Ontology-Based Data Access of Animals with Ontop – A Tutorial –

Frances Gillis-Webber
Department of Computer Science
University of Cape Town
South Africa

C. Maria Keet
Department of Computer Science
University of Cape Town
South Africa

The aim of this tutorial is to demonstrate the concept of Ontology-Based Data Access (OBDA), where one queries the data residing in a database through the ontology. We use the Ontop framework for this, which is compatible with the Protégé ontology development environment (ODE), and MySQL is used as the relational database. The following tasks are demonstrated in this tutorial:

- the implementation of Ontop and the connection to a data source;
- the creation of a mapping in the mapping layer;
- using the virtualised data to (a) query in SPARQL and (b) materialise as non-virtualised triples.

The remainder of this tutorial is structured as follows: Section 1 presents a brief overview; the installation of Ontop in Protégé is described in Section 2, and Section 3 demonstrates the mapping process. Section 4 concludes with a demonstration of the use of the data from Section 3. Note: similar tutorial material (and on which the general instructions of this tutorial are based) for a subject domain other than elephants inhabiting one of the 10 largest national parks in the world can be found at `https://github.com/ontop/ontop/wiki` and `https://ontop-vkg.org/tutorial/`.

1 Introduction

Ontology-Based Data Access (OBDA) connects the TBox of the ontology to the ABox with the data that is stored in a relational database (rather than that in an OWL file), with a mapping layer as the intermediary. The relational data which is retrieved as results to a query can then be returned as-is or it can be virtualised into RDF by way of mappings, to generate a knowledge graph [3]. This thus transforms the composite into a knowledge base, with data independence between the TBox and the ABox, but without losing the ability to query this data using SPARQL.

Motivations for this type of architecture, as well as the technical details, are described in Chapter 8 of the ontology engineering textbook.

2 Installing Ontop in Protégé

In this section, where to download Ontop is specified, as well as other required software for Microsoft Windows. The section is concluded with a description of the processes required to install Ontop and set up a connection to the database.

2.1 What to download

The following material is required for this tutorial:

- **Protégé 5.x**, which can be downloaded from `https://protege.stanford.edu/`.
- **Java Runtime Environment** (JRE) 8.x.
- **Ontop**, which can be downloaded from `https://github.com/ontop/ontop/releases`.
- **African Wildlife Ontology for ODBA** (AWO), version `4obda.owl`, which is downloadable from `https://people.cs.uct.ac.za/~mkeet/OEbook/ontologies/` or `http://www.meteck.org/teaching/OEbook/ontologies/`.
- **Animals database** (ADB), which is downloadable from `https://www.movebank.org/movebank/#page%3Dstudies%2Cpath%3Dstudy736029750` (the DOI for the data is 10.5441/001/1.403h24q5.).
- **MySQL**. If installing ADB on your localhost: `https://dev.mysql.com/downloads/mysql/`.

2.2 Setting up Ontop

The steps required to implement Ontop are shown below. It is assumed that Protégé has already been installed.

1. Go to the most stable release of Ontop on **Github**.
2. Scroll down to the **Assets** sub-section and download the **it.unibz.inf.ontop.protege-[version].jar** file.

3. If Protégé is open, close the application.
4. Go to the Protégé directory and navigate to the **plugins** folder.
5. Move the downloaded .jar file to this folder.
6. After opening Protégé, Ontop should now be visible in the menu bar.
7. In the menu bar, click on **Window** and then **Tabs**.
8. Select **Ontop Mappings**, and then select **Ontop SPARQL**. Both tabs should now display in the main window, as shown in Figure 1.

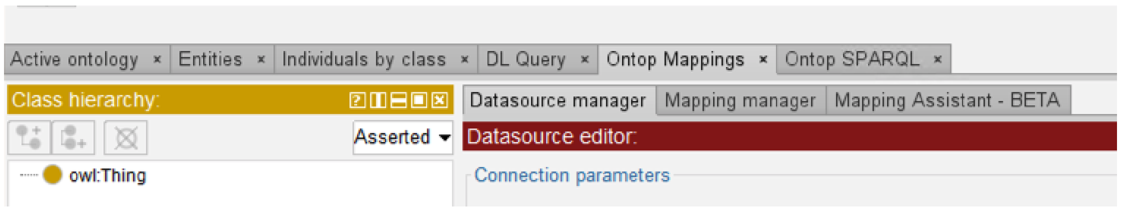


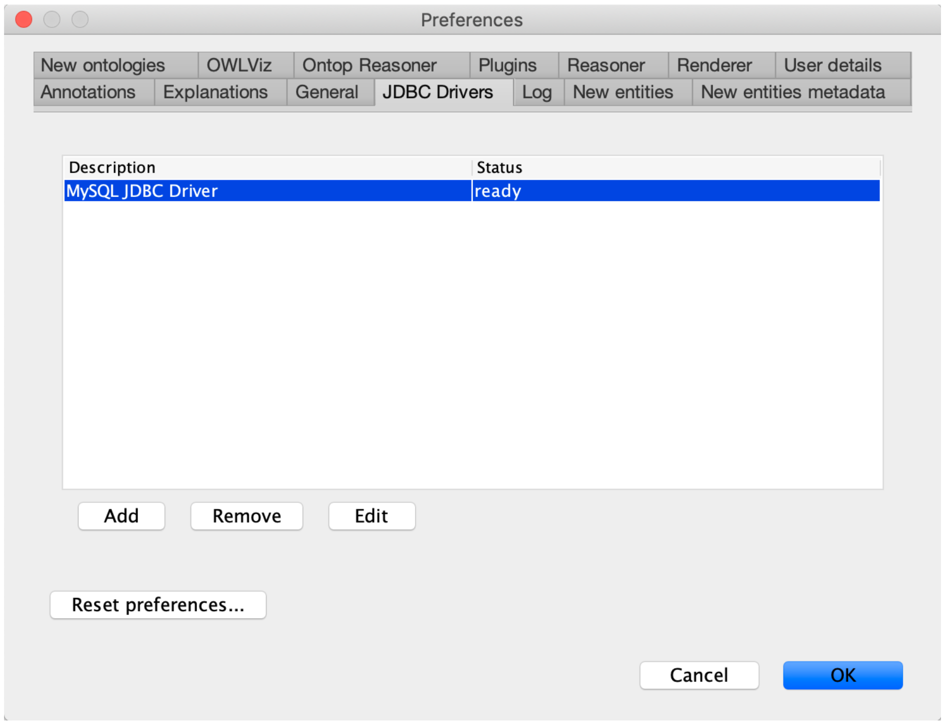
Figure 1: Ontop installed in Protégé

2.3 Setting up the JDBC driver

Ontop uses the Java DataBase Connectivity Framework (JDBC) to connect to data sources. Before setting up the database connection, the JDBC driver needs to be installed. Installing the JDBC driver for MySQL is described below; for other databases, go to `https://github.com/ontop/ontop/wiki/ObdalibPluginJDBC`.

1. In your web browser, search for the **MySQL Connector/J** driver. At time of writing, it can be found at: `https://dev.mysql.com/downloads/connector/j/`.
2. For the Windows and Mac operating system (OS), select **Platform Independent** from the **Operating System** dropdown.
3. Download the compressed file appropriate for your OS, uncompress it, and save it to a location of your choice.
4. Open Protégé, and go to File > Preferences (Windows) or Protégé > Preferences (Mac).
5. In the pane that opens, select the **JDBC Drivers** tab.
6. If the JDBC driver for MySQL is not visible, then click the **Add** button.

7. In the **Description** field, enter a descriptor, for eg. 'MySQL JDBC Driver'.
8. Select **com.mysql.jdbc.Driver** in the **Class Name** dropdown.
9. Browse to the .jar file of the driver downloaded in Section 3.2 and then click **OK**.
10. Your installed driver should have a status of **ready** in Protégé, as shown in Figure 2.
11. Click **OK** to exit the Preferences pane.



Description	Status
MySQL JDBC Driver	ready

Figure 2: Ontop installed in Protégé

2.4 Setting up the database connection in Protégé

The database first needs to be installed. The database can be hosted in the cloud or it can be installed on your localhost. If installing locally, make sure MySQL server is set up and running. Connect to MySQL server and run/import the ADB script. This script will create the database (with the name 'animals'), its structure and it will populate the tables with data. Before proceeding, check that the database has been set up and you can query the data therein.

The connection URL in Protégé for MySQL is of the following format:

```
jdbc:mysql://hostname:port/animals?sessionVariables=sql_mode='ANSI'
```

These tasks describe how to write the correct connection URL for the **animals** database we will be using.

1. Replace **hostname** with the database server name. If you are hosting the database locally, then replace **hostname** with **localhost**.

2. If the port number is known, replace **port** with this number. If not used, then remove **:port**.

3. The text **animals** is the name of the database. Make sure to keep the querystring parameters unchanged.

To set up the connection in Protégé:

1. Open AWO in Protégé.

2. Select the **Ontop Mappings** tab, and then the **Datasource manager** tab.

3. Enter the connection URL tailored to your configuration in the field provided.

4. Enter the username and password in the Database **Username** and **Password** fields.

5. Select **com.mysql.jdbc.Driver** from the Driver class dropdown.

6. Click the **Test Connection** button.

7. If there is a successful connection, a **Connection is OK** message should be shown, as shown in Figure 3.

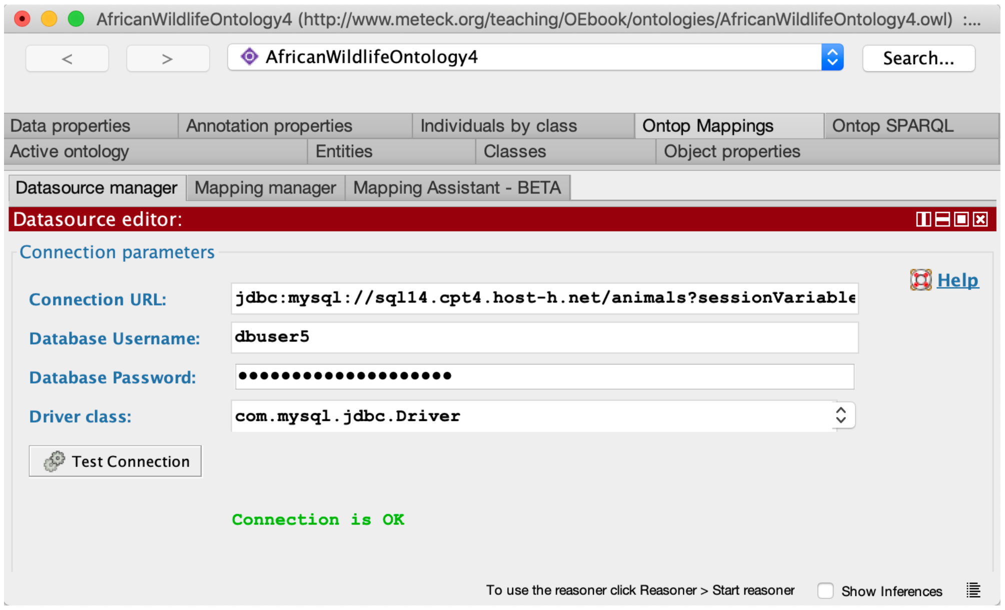
3 The mapping process

In this section, an exercise is presented on creating mapping axioms between the database and the tutorial ontology. That is, we are going to link elements from the ontology to queries over the database. First,

1. Open the ontology (the OBDA version of AWO) in Protégé.

2. Choose some vocabulary element you wish to retrieve the instances of (e.g., *Elephant*).

Creating a mapping axiom is then a three-step process in Protégé:



Connection URL: jdbc:mysql://sql14.cpt4.host-h.net/animals?sessionVariable

Database Username: dbuser5

Database Password: ••••••••••••••••••••

Driver class: com.mysql.jdbc.Driver

Connection is OK

Figure 3: A successful connection to MySQL

1. *Source:* write an SQL query that will retrieve the instances (or tuples) from the database that are the instances of the chosen element (all elephants, in this case), and then test this query to verify it is the correct query. This is the body of the mapping.

2. *Target:* map the fields returned in the SQL query to a triples template. This is the head of the mapping.

3. *Save:* edit the mapping ID to a unique, more memorable identifier (optional), and then save the mapping.

In AWO, there is a class *Elephant*. This is the common name of the mammal, *Loxodonta africana*. Likewise, there are other sub-classes of *Animal* in AWO (see Figure 4), such as *Giraffe*, *Lion* and *Warthog*.

ADB contains data of animals tagged (collared) for scientific research in national parks in South Africa [1], whose data analysis has been published recently [2]. There are five tables: *speciesList*, *studies*, *nationalParks*, *countries*, *animals* and *animalTags*. The table *speciesList* lists the animals from the class *Animal* in AWO. The table *studies* contains the list of studies included in this database. The table *nationalParks* contains the parks in which these studies were conducted. The table *countries* contains the countries in which the parks are located. The table *animals* lists the animals which have been tagged. The table *animalTags* lists the tags associated with an animal from the *animals* table. An animal may have one or more tags during its lifetime.

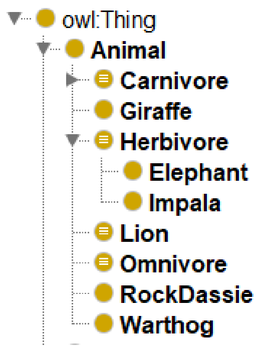


Figure 4: The class *Animal*

GOAL
Using SPARQL, we want to retrieve all tagged elephants. To do this, we need to do the following:

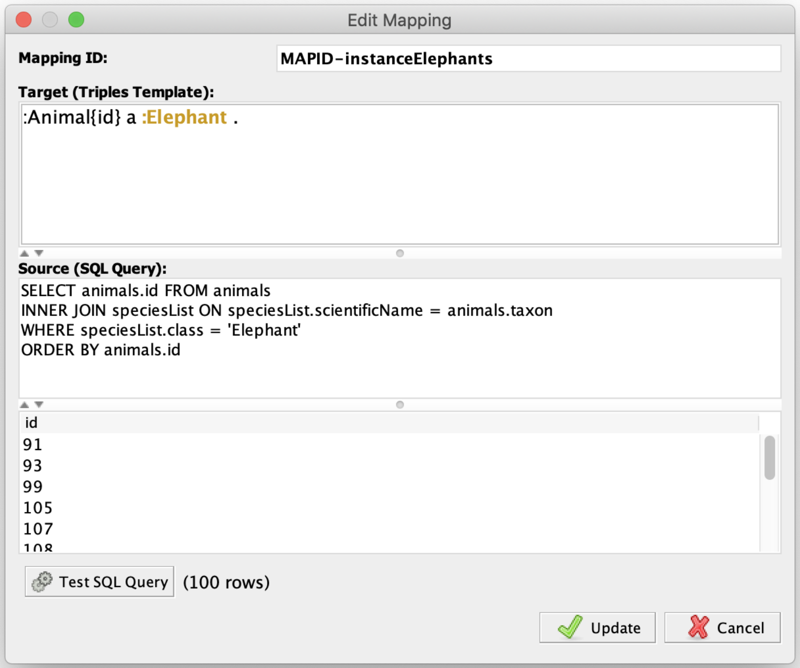
1. Create a mapping axiom in Protégé.
2. Write a SPARQL query to query this data.

3.1 Writing the SQL query for the body of the mapping

1. In Protégé, select the **Ontop Mappings** tab, and then the **Mapping manager** tab.
2. Click the **Create** button.
3. In the **Source (SQL Query)** box, enter the SQL query as shown in Listing 1.
4. Then click the **Test SQL Query** button.
5. If the results returned are similar to those shown in Figure 5, proceed to the next section, otherwise go back to point 3.

```
SELECT animals.id FROM animals
INNER JOIN speciesList ON speciesList.scientificName = animals.taxon
WHERE speciesList.class = 'Elephant'
ORDER BY animals.id
```

Listing 1: Example SQL query for retrieving elephants



Mapping ID: MAPID-instanceElephants

Target (Triples Template):

```
:Animal{id} a :Elephant .
```

Source (SQL Query):

```
SELECT animals.id FROM animals
INNER JOIN speciesList ON speciesList.scientificName = animals.taxon
WHERE speciesList.class = 'Elephant'
ORDER BY animals.id
```

```
id
91
93
99
105
107
108
```

(100 rows)

Figure 5: A mapping for *Elephant*.

3.2 Writing the triples template for the head of the mapping

1. In the **Target (Triples Template)** box, enter the triples as shown in Listing 2. The completed mapping should look similar to that shown in Figure 5.

2. If desired, edit the **Mapping ID** to `MAPID-instanceElephants`, and then click the **Update** button.

3. The newly created mapping will now show in the Mapping manager pane (see Figure 6).

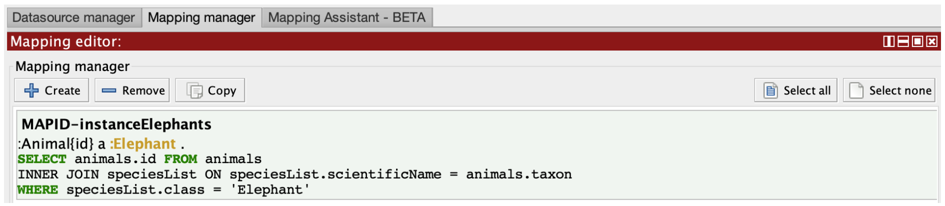
```
:Animal{id} a :Elephant .
```

Listing 2: Example triples template

The **column** returned in the SQL query is mapped to the equivalent **{column}** in the triples template.

Points to note:

- The order of each mapped column in the triples template must match the order of the column names returned in the SQL query.
- A column from the SQL query can only be mapped once in the triples template.
- Only inner joins are currently supported by Ontop.



```
MAPID-instanceElephants
:Animal{id} a :Elephant .
SELECT animals.id FROM animals
INNER JOIN speciesList ON speciesList.scientificName = animals.taxon
WHERE speciesList.class = 'Elephant'
```

Figure 6: A mapping in Mapping manager

Now that the mapping between ADB and AWO has been created, the virtualised data can be queried using SPARQL.

3.3 Querying the data using SPARQL

1. In Protégé, select the **Ontop SPARQL** tab.
2. In the **Query Editor** box, enter the prefix declarations and the SPARQL query as shown in Listing 3.
3. Click **Reasoner** in the menu bar.
4. Make sure the **Ontop x.x** reasoner is selected, and then click **Start reasoner**.
5. Go back to the **Query Editor** box in the Ontop SPARQL tab and click the **Execute** button. The results should look similar to those in Figure 7.
6. If you edit your SPARQL query, go back to **Reasoner** in the menu bar and click **Synchronise reasoner** before executing the query.

```
PREFIX : <http://www.meteck.org/teaching/OEbook/ontologies/
AfricanWildlifeOntology4obda.owl#>

SELECT DISTINCT *
WHERE { ?animalId a :Elephant }
```

Listing 3: A SPARQL query using the template from the created mapping

The triple pattern enclosed within the curly braces:

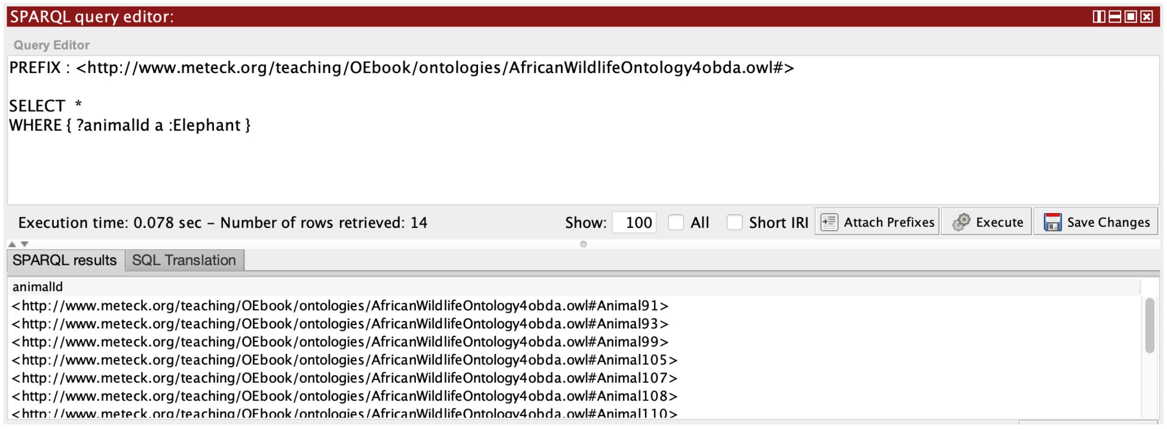
```
?animalId a :Elephant
```

is the same pattern as that used in the triples template:

```
:Animal{id} a :Elephant .
```

The mapped values from the database are now replaced by SPARQL variables.

The goal was to retrieve all tagged elephants. Using the data from ADB, this has returned 14 elephants (as instances of the Elephant class).



```
PREFIX : <http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#>

SELECT *
WHERE { ?animalId a :Elephant }
```

Execution time: 0.078 sec – Number of rows retrieved: 14

```
animalId
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal91>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal93>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal99>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal105>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal107>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal108>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal110>
```

Figure 7: Querying virtual triples using SPARQL, which for our sample database lists the collared (tagged) elephants in the bottom pane.

4 Advanced Goal

Building on the tasks from the previous sections, we now move onto a more advanced goal. We want to identify, using SPARQL, the instances of Elephant which were collared for scientific research within South African national parks. To do this, we need to do the following:

1. Create virtual instances of the sub-class *SouthAfricaNationalPark* from the *nationalParks* table in ADB.

2. Create virtual triples using the *tagged-in* property from the instances of the *Elephant* class to the relevant instances of the *SouthAfricaNationalPark* sub-class.

4.1 Virtual instances of *SouthAfricaNationalPark*

1. Create a new mapping in the Mapping Manager.
2. In the **Source (SQL Query)** box, enter the SQL query as shown in Listing 4.
3. In the **Target (Triples Template)** box, enter the triples as shown in Listing 5.
4. Edit the **Mapping ID** to `MAPID-instanceSouthAfricaNationalParks`, and then click the **Accept** button.
5. The newly created mapping will now show in the Mapping manager pane.

```
SELECT nationalParks.shortName from nationalParks
WHERE nationalParks.countriesId = 1
```

Listing 4: SQL query for South African National Parks

```
:{shortName}NationalPark a :SouthAfricaNationalPark .
```

Listing 5: Triples template for SouthAfricaNationalPark instances

4.2 Virtual triples using the *tagged-in* object property

1. Create a new mapping in the Mapping Manager.
2. In the **Source (SQL Query)** box, enter the SQL query as shown in Listing 6.
3. In the **Target (Triples Template)** box, enter the triples as shown in Listing 7.
4. Edit the **Mapping ID** to `MAPID-triplesTaggedIn`, and then click the **Accept** button.
5. The newly created mapping will now show in the Mapping manager pane.

```
SELECT animals.id, nationalParks.shortName FROM animalTags
INNER JOIN animals ON animals.id = animalTags.animalsId
INNER JOIN speciesList ON speciesList.scientificName = animals.taxon
INNER JOIN studies ON studies.id = animalTags.studiesId
INNER JOIN nationalParks ON nationalParks.id = studies.nationalParksId
WHERE speciesList.class = 'Elephant'
ORDER BY animals.id
```

Listing 6: SQL query for the *tagged-in* object property

```
:Animal{id} :tagged-in :{shortName}NationalPark .
```

Listing 7: Triples template for the *tagged-in* object property

There should be three mappings in Protégé:

1. `MAPID-instanceElephants`
2. `MAPID-instanceSouthAfricaNationalParks`
3. `MAPID-triplesTaggedIn`

The virtualised data between ADB and AWO can now be queried using SPARQL.

4.3 Querying the data using SPARQL

1. In the **SPARQL Query Editor** box, enter the prefix declarations and the SPARQL query as shown in Listing 8.
2. Click **Reasoner** in the menu bar, and then click **Start / Synchronize reasoner**.
3. Go back to the **Query Editor** box in the Ontop SPARQL tab and click the **Execute** button. The results should look similar to those in Figure 8.

```
PREFIX : <http://www.meteck.org/teaching/OEbook/ontologies/
AfricanWildlifeOntology4obda.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT *
WHERE {
        ?nationalPark a :SouthAfricaNationalPark .
        ?animalId a :Elephant ; :tagged-in ?nationalPark .
}
```

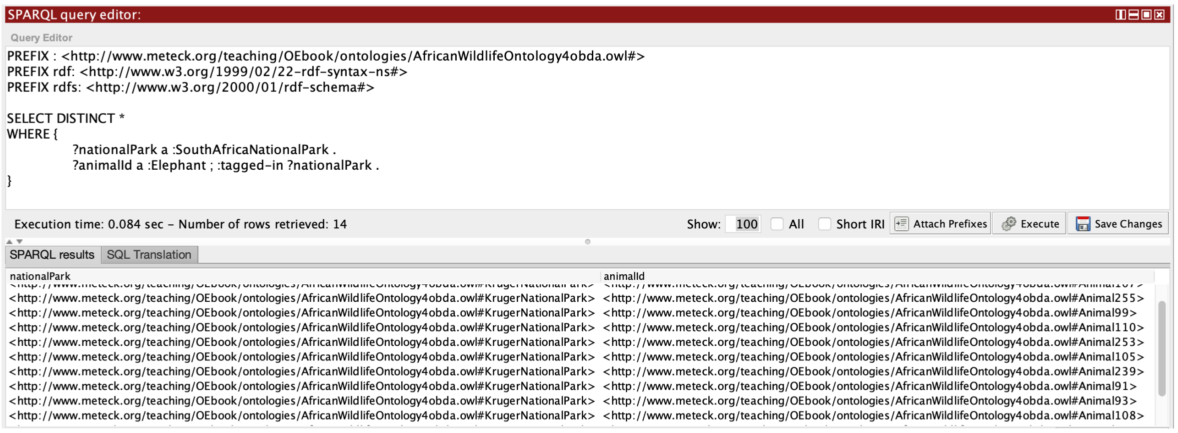
Listing 8: The SPARQL query to retrieve the elephants tagged in SA national parks.

5 Working with the virtualised data

5.1 Materialising the triples

The data from the mappings does not have to remain virtualised, it can be materialised as well.

1. In Protégé, click **Ontop** in the menu bar.
2. Then click **Materialize triples**.
3. In the window that opens, the user is presented with the option to add the virtualised triples to the current ontology or dump the ontology and the virtualised triples to an external file.



```
PREFIX : <http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT DISTINCT *
WHERE {
        ?nationalPark a :SouthAfricaNationalPark .
        ?animalId a :Elephant ; :tagged-in ?nationalPark .
}
```

Execution time: 0.084 sec – Number of rows retrieved: 14

nationalPark	animalId
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#KrugerNationalPark>	<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal255>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#KrugerNationalPark>	<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal99>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#KrugerNationalPark>	<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal110>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#KrugerNationalPark>	<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal253>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#KrugerNationalPark>	<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal105>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#KrugerNationalPark>	<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal239>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#KrugerNationalPark>	<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal91>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#KrugerNationalPark>	<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal93>
<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#KrugerNationalPark>	<http://www.meteck.org/teaching/OEbook/ontologies/AfricanWildlifeOntology4obda.owl#Animal108>

Figure 8: OBDA query in SPARQL (top pane) with the instances (bottom pane), being the tagged elephants in SA national parks.

4. Select a preferred option and follow the screen prompts.

Figure 9 shows an example of materialised triples of a virtual instance, dumped to an external file, using the mapping from Section 3.2.

```
<rdf:Description rdf:about="http://www.meteck.org/teaching/OEbook/ontologies/
AfricanWildlifeOntology4obda.owl#Animal91">
        <rdf:type rdf:resource="http://www.meteck.org/teaching/OEbook/ontologies/
AfricanWildlifeOntology4obda.owl#Animal"/>
</rdf:Description>
```

Figure 9: Materialised triples of a virtual instance

References

[1] Slotow, R., Thaker, M., Vanak, A.: Data from: Fine-scale tracking of ambient temperature and movement reveals shuttling behavior of elephants to water (2019), retrieved from Movebank Data Repository, `https://doi.org/10.5441/001/1.403h24q5`

[2] Thaker, M., Gupte, P.R., Prins, H.H.T., Slotow, R., Vanak, A.T.: Fine-scale tracking of ambient temperature and movement reveals shuttling behavior of elephants to water. Front. Ecol. Evol. **7**(4) (2019). https://doi.org/10.3389/fevo.2019.00004

[3] Xiao, G., Ding, L., Cogrel, B., Calvanese, D.: Virtual knowledge graphs: An overview of systems and use cases. Data Intelligence **1**, 201–223 (2019)