ECE_5DA04_TP – Big Graph Databases
Lab 3: Obi-Wan

Garima Gaur

16th January 2025

Submission deadline: Friday, 31 January, 2025 at 23:55:00 (Paris time)
Acknowledgement: Thanks to Maxime Buron for sharing his Obi-Wan tutorial. This assignment is largely based on the tutorial.

1 Lab Organization

- **Obi-Wan High-level Overview**: Please find the presentation under `Lab 3 slides` link at Moodle.
- **ObiWan Setup**: Download obiwan-setup.zip available under the file tab of Lab 3 at Moodle.
- **References**:
  - **Git Repository:** `https://gitlab.inria.fr/cedar/obi-wan`
  - **Publication:** Refer to "Ontology-Based RDF Integration of Heterogeneous Data" (`https://inria.hal.science/hal-02446427/document`) published in EDBT 2020 for the better and deeper understanding of GLAV mappings and Obi-Wan implementation.

2 Assignment

In this lab session, we use the ontology-based data access system **Obi-Wan**, which performs the RDF integration of heterogeneous data sources. This system has been developped by Maxime Buron during its PhD thesis. Obi-Wan allows querying heterogeneous data sources (currently PostgreSQL, SQLite, MongoDB, Jena TDB) through an RDFS ontology. Obi-Wan handles global-local-as-view (GLAV) mappings to integrate the sources within a virtual RDF graph. The goal of this lab session is first to get familiar with Obi-Wan and then learn to integrate data sources.

Obi-Wan Project Layout: Unzip the `obiwan-setup.zip` and move to the directory `sw-ris`. This directory contains an example of Obi-Wan project. Usually, an Obi-Wan project contains the 4 following files:

- *`obi-wan`*`.properties` : the file setting the behaviour of Obi-Wan
- *`ris`*`.json` : the RDF Integration System file containing mappings
- *`ontology`*`.nt` : the RDFS ontology in N-triples format
- `querysession.properties` : this file is necessary for technical reasons when we use materialization-based query answering (you should not modify it)

Data Sources: For this assignment, we will integrate two datasets about the Star Wars. In the `sw-ris` directory, you will find two corresponding SQLite database files:

- `sw_bank.db` contains tables about Star Wars characters
- `sw_imdb.db` contains tables about Star Wars movies

The database in `sw_bank.db` contains two tables:

- `vehicle`
  - `character`: the name of the driving character
  - `vehicleType`: the type of vehicule driven by the character
- `lightsaber`
  - `character`: the name of the character using the weapon
  - `saber`: the name of the lightsaber

The database in `sw_imdb.db` contains three tables:

- `title`
  - `tconst`: alphanumeric unique identifier of the title
  - `primaryTitle`: the more popular title
  - `startYear`: represents the release year of a title
  - `runtimeMinutes`: primary runtime of the title, in minutes
- `person`
  - `nconst`: alphanumeric unique identifier of the name/person
  - `primaryName`: name by which the person is most often credited
  - `birthYear`: in YYYY format
  - `deathYear`: in YYYY format if applicable, else NULL
- `casting`
  - `tconst`: alphanumeric unique identifier of the title
  - `nconst`: alphanumeric unique identifier of the name/person
  - `category`: the category of job that person was in ('actor', 'actress', 'director' or 'composer')
  - `character`: the name of the character played if applicable, else NULL

Getting started: Start the Obi-Wan server from `sw-ris` directory using the following command:

```
java -jar <path to obiwan.jar> server obi-wan.properties
```

Once the server is launched, you can access to the RIS visualization at `http://localhost:8080/ris/sw-ris/index.html` and the SPARQL endpoint at `http://localhost:8080/client/index.html`.

	Rule name	Entailment rule	
$\mathcal{R}$	rdfs5	$(p_1, :\text{subproperty}, p_2), (p_2, :\text{subproperty}, p_3) \rightarrow (p_1, :\text{subproperty}, p_3)$	$\mathcal{R}_{\text{onto}}$
	rdfs11	$(s, :\text{subclass}, o), (o, :\text{subclass}, o_1) \rightarrow (s, :\text{subclass}, o_1)$	
	ext1	$(p, :\text{domain}, o), (o, :\text{subclass}, o_1) \rightarrow (p, :\text{domain}, o_1)$	
	ext2	$(p, :\text{range}, o), (o, :\text{subclass}, o_1) \rightarrow (p, :\text{range}, o_1)$	
	ext3	$(p, :\text{subproperty}, p_1), (p_1, :\text{domain}, o) \rightarrow (p, :\text{domain}, o)$	
	ext4	$(p, :\text{subproperty}, p_1), (p_1, :\text{range}, o) \rightarrow (p, :\text{range}, o)$	
	rdfs2	$(p, :\text{domain}, o), (s_1, p, o_1) \rightarrow (s_1, :\text{type}, o)$	$\mathcal{R}_{\text{data}}$
	rdfs3	$(p, :\text{range}, o), (s_1, p, o_1) \rightarrow (o_1, :\text{type}, o)$	
	rdfs7	$(p_1, :\text{subproperty}, p_2), (s, p_1, o) \rightarrow (s, p_2, o)$	
	rdfs9	$(s, :\text{subclass}, o), (s_1, :\text{type}, s) \rightarrow (s_1, :\text{type}, o)$	

Figure 1: Entailment Rules used by Obi-Wan

2.1 Integration Task

You are required to declare new mappings (in `ris.json`) or/and modify RDFS ontology (in `ontology.nt`) in order to complete the following tasks on the integrated virtual RDF graph. The list of entailment rules that you may need to refer to are presented in Figure 2.1. For evaluating the queries specified below, use the Obi-Wan SPARQL endpoint mentioned above. Note that you should add the following prefixes to the queries:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sw: <https://www.starwars.com/databank/>
PREFIX imdb: <http://www.imdb.com/>
```

1. Update the mapping named "title" so that a user can obtain some answer to the SPARQL query:

   ```
   SELECT * WHERE {
     ?movie imdb:title ?title .
     ?movie imdb:releaseYear ?year .
   }
   ```

2. Create a new mapping named "actress" that extracts from the table `casting` in the database `sw_imdb.db` who are the actresses playing in movies, and that enables answering to the SPARQL query:

   ```
   SELECT ?name ?title WHERE {
     ?actress imdb:fullName ?name .
     ?actress imdb:actressIn ?movie .
     ?movie imdb:title ?title .
   }
   ```

3. Create another mapping for the male actors to answer the following query:

   ```
   SELECT ?name ?title WHERE {
     ?actor imdb:fullName ?name .
     ?actor imdb:maleActorIn ?movie .
     ?movie imdb:title ?title .
   }
   ```

4. We want to be able to get who is playing in which movie using a single SPARQL query and not having one query for the actresses and another for the male actors. Update the ontology in a way so that the following query gets an answer:

```
SELECT ?name ?title WHERE {
  ?actor imdb:fullName ?name .
  ?actor imdb:actorIn ?movie .
  ?movie imdb:title ?title .
}
```

5. Add a new mapping to answer the following query asking for the actor's name with the type of object used by a character they play:

```
SELECT ?actorName ?character ?objectType WHERE {
  ?actor imdb:fullName ?actorName .
  ?actor imdb:plays ?character .
  ?character sw:uses ?object .
  ?object rdf:type ?objectType .
}
```

6. Add new mappings and update the ontology such that the following query list down all the people involved in the movies:

```
SELECT ?personName ?title WHERE {
  ?person imdb:fullName ?personName .
  ?person imdb:workedIn ?movie .
  ?movie imdb:title ?title .
}
```

7. Write a mapping populating the class `sw:CharactersFromFirstTrilogy` of characters appearing in the first trilogy of movies.

8. Write a mapping populating the class `sw:youngArtist` of people who were at the age below 21 when they acted in the movies.

9. Add a single triple to the ontology to populate the class `imdb:Movie` with the movies.

10. Imagine a way to sufficiently populate the graph with new mappings and ontological triples to obtain interesting answers to the query:

```
SELECT ?person ?p ?movie WHERE {
  ?person rdf:type imdb:Person .
  ?person ?p ?movie .
  ?p rdfs:subPropertyOf imdb:castsIn .
}
```

Note the following for pleasant experience:

- Start by copying an existing mapping and modify it to obtain what you want. A small error in the JSON entry will throw errors.
- Kill (`Ctl +C`) and rerun the server using the command `java -jar <path to obiwan.jar> server obi-wan.properties` to evaluate the queries after each update to the `ris.json` or/and `ontology.nt`

3 Submission Instructions

You are required to prepare the following three files:

1. `report.pdf`: For each task in the assignment, you are requried to specify what changes you made to which file. Along with the changes in the mappings or/and ontology, you are required to run queries specified and report their answers.

2. `ris.json`: The final version of the `ris.json` that you get after solving all the integration tasks in the assignment.

3. `ontology.nt`: Similar to the final version of `ris.json`, submit the final version of this file containing all the exisitng as well as the new ontology triples while solving the tasks.

For the submission on the Moodle, upload a single zip folder containing all the three files. Please name your submission as `LASTNAME_FIRSTNAME.zip` with first and last name in capital letters. Notice that the submission deadline is Friday, 31*st* January, 2025 at 23:55:00 (Paris time). Kindly adhere to the deadline.