Practical Knowledge Graphs with GraphDB - SDM Lab 2

Yutao Chen[*], Ziyong Zhang[*]

{yutao.chen,ziyong.zhang}@estudiantat.upc.edu

1 Introduction

This project is completed for SDM Lab 2. In this project, we will use the python RDFLib library to complete ontology creation, including the TBOX definition, and ABOX definition, Create the final ontology, and then we will define the TBOX and ABOX Save it in Turtle format and upload it to GraphDB. Finally, we will use SPARQL queries to test whether our Ontology is complete. The data used in the project will continue to use the data from Lab 1 on research publication domains, all codes mentioned in this report will be shown on GitHub: https://github.com/Ziyong-Zhang/SDM_lab2.

2 TBOX definition

2.1 Definition

We have used the RDFLib to define the TBOX, and showed our visualization on gra.fo, and in Figure 1.

In the TBOX, the URI is defined in Namespace12.

```
    URI = Namespace("http://SDM_lab_2.org/")
```

Listing 1: Namespace Denifition

To present the relation written in the description: "Authors write research papers that can be published in the proceedings of a conference or workshop (a conference is a well-established forum while a workshop is typically associated to new trends still being explored), or in a journal." We defined some Classes and Properties

- **Classes:**
  - **Author**
  - **ResearchPaper**
  - **Proceedings**
  - **Conference**
    - **RegularConference**
    - **Workshop**
- **Properties:**
  - **Author – Write → ResearchPaper**
  - **ResearchPaper – IsInProceeding → Proceedings**
  - **Conference – BelongTo → Proceedings**
  - **Workshop – WorkshopIn → Proceedings**
  - **RegularConference – ConIn → Proceedings**

The RDFLib definition example code is listed in 2, the full version of the code could refer to 1_TBOX_create.py.

```
# Author
graph.add((URI.Author, RDF.type, RDFS.Class))
graph.add((URI.Author, RDFS.subClassOf, URI.Person))
graph.add((URI.Author, RDFS.label, Literal("Author")
    ))

# Workshop
graph.add((URI.Workshop, RDF.type, RDFS.Class))
graph.add((URI.Workshop, RDFS.subClassOf, URI.
    Conference))
graph.add((URI.Workshop, RDFS.label, Literal("
    Workshop")))

# Write
graph.add((URI.Write, RDF.type, RDF.Property))
graph.add((URI.Write, RDFS.domain, URI.Author))
graph.add((URI.Write, RDFS.range, URI.ResearchPaper)
    )
graph.add((URI.Write, RDFS.label, Literal("Write")))
```

Listing 2: Tbox Denifition

Continually, to present the "A conference/workshop is organized in terms of editions. Each edition of a conference/workshop is held in a given city (venue) at a specific period of time of a given year.", to present venue and its relation, we defined the Classes and Properties:

- **Classes:**

[*]Universitat Politècnica de Catalunya (UPC), Erasmus Mundus Big Data Management and Analytics (BDMA)

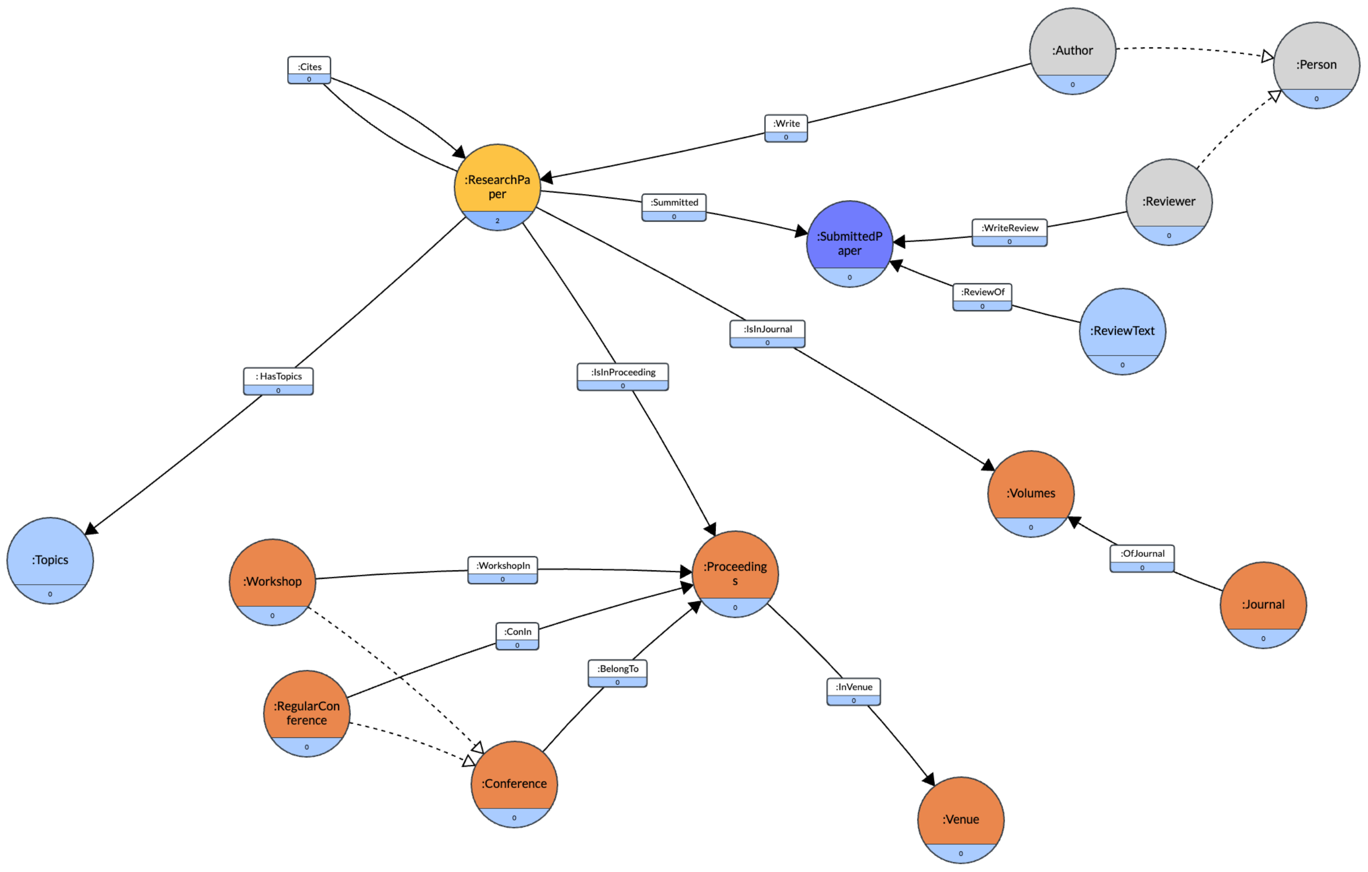


Figure 1: TBOX Logics

  - **Venue**
- **Properties:**
  - **Proceedings – InVenue → Venue**

And the RDFLib code is Listing3:

```
# Venue
graph.add((URI.Venue, RDF.type, RDFS.Class))
graph.add((URI.Venue, RDFS.label, Literal("Venue")))

# InVenue
graph.add((URI.InVenue, RDF.type, RDF.Property))
graph.add((URI.InVenue, RDFS.domain, URI.Proceedings
    ))
graph.add((URI.InVenue, RDFS.range, URI.Venue))
graph.add((URI.InVenue, RDFS.label, Literal("InVenue
    ")))
```

Listing 3: Tbox Denifition

For the requirement ”Proceedings are published records which include all the papers presented in an edition of a conference or workshop. Oppositely, journals do not hold joint meeting events and, like a magazine, a journal publishes accepted papers in terms of volumes. There can be various volumes of a journal per year.”, we defined Class Volumes to present the volumes and defined their relations in:

- **Classes:**
  - **Volumes**
- **Properties:**
  - **ResearchPaper – IsInJournal → Volumes**

For the requirement: ” A paper can be written by many authors, however only one of them acts as corresponding author. A paper can be cited by another paper, meaning their content is related.” We defined Property Cites to present the relation of cite.

- **Properties:**
  - **ResearchPaper – Cites → ResearchPaper**

And its RDFLib code is code 4

```
# Cites
graph.add((URI.Cites, RDF.type, RDF.Property))
graph.add((URI.Cites, RDFS.domain, URI.ResearchPaper
    ))
```

```
graph.add((URI.Cites, RDFS.range, URI.ResearchPaper)
    )
graph.add((URI.Cites, RDFS.label, Literal("Cites")))
```

Listing 4: Tbox Denifition

Then, for the requirement ”A paper can be about one or more topics, specified by means of keywords (e.g., property graph, graph processing, data quality, etc.). A paper must also contain an abstract (i.e., a summary of its content).”, to present its topics, we introduce the Class Topics Code:

- **Class:**
  - **Topics**
- **Property:**
  - **HasTopics**

The last part of requirement is: ” Finally, we also want to include in the graph the concept of review. When a paper is submitted to a conference/workshop or a journal, the conference chair or the journal editor assigns a set of reviewers (typically three) to each paper. Reviewers are scientists and therefore they are relevant authors (i.e., they have published papers in relevant conferences or journals). Obviously, the author of a certain paper cannot be reviewer of her own paper.” To present this, we add a new Class **Person**, which contains the subclass Author and Reviewer, so we could better query without too much limitation in the query code, the Classes and Properties are:

- **Class:**
  - **SubmittedPaper**
  - **Reviewer**
  - **ReviewText**
  - **Person**
- **Property:**
  - **Submitted**
  - **WriteReview**
  - **ReviewOf**

To sum up, in this section we introduced our thinking and corresponding code about the requirement and provided some part of the code, the full version of the code is available on 1_TBOX_create.py.

2.2 Hierachy and Assumptions

The hierary of the TBOX defined is shown as Figure 2. The following assumptions are made related to the TBOX modelling:

- All Research Papers are considered as Submitted Papers. Therefore, the numbers of instances for both classes would be the same.
- Since the cited papers are randomly assigned based on the research paper, the domain and range of the Property Cites here are both Class Research Paper. In reality, it is not always the case.
- Different types of conferences (workshop, regular conference) are considered as subclasses of the Conference Class.
- Authors and Reviewers Classes are considered as subclass of Person Class.
- Class Topics is derived from the keywords from the research papers, which will be shown in the process of creating the ABOX.

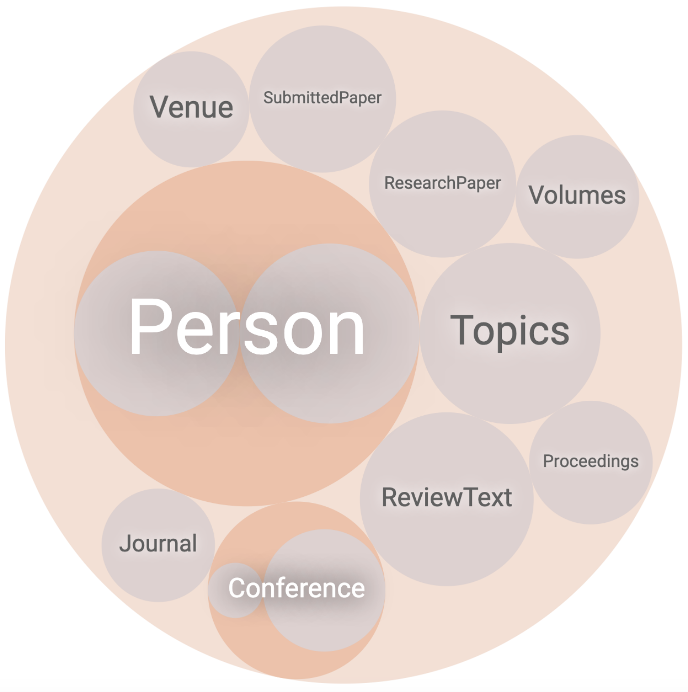


Figure 2: Class Hierarchy

There are 14 classes in total in the TBOX, shown as both Figure 1 and Figure2. The domain and range of different properties is demonstrated as Table 1.

Property Name	rdfs: domain	rdfs: range
Cites	:ResearchPaper	:ResearchPaper
HasTopics	:ResearchPaper	:Topics
Submitted	:ResearchPaper	:SubmittedPaper
PaperTitle	:ResearchPaper	XSD.string
PaperAbstract	:ResearchPaper	XSD.string
Write	:Author	:ResearchPaper
WriteReview	:Reviewer	:SubmittedPaper
ReviewOf	:ReviewText	:SubmittedPaper
OfJournal	:Journal	:Volumes
InVenue	:Proceedings	:Venue
BelongTo	:Conference	:Proceedings
WorkshopIn	:Workshop	:Proceedings
ConIn	:RegularConference	:Proceedings
IsInProceeding	:ResearchPaper	:Proceedings
IsInJournal	:ResearchPaper	:Volumes

Table 1: TBOX Properties

3 ABOX Definition & Create the final Ontology

The data is reused from the research publications domain from the Assignment of Lab 1. In this section, we will make use of RDFLib to convert CSV to RDFS, we put the ABOX definition and create ontology together because this would make it easier to build the schema, the code is available at 2_ABOX_create.ipynb.

3.1 Namespace and Data Import

We created the namespace as $http://SDM\_lab\_2.org/$ and imported the research publications domain data from the folder.

```python
URI = Namespace("http://SDM_lab_2.org/")
g = Graph()

csv_dir = '/Users/SDM/Lab_2/Lab_doc/data/CSVs/'

csv_files = [f for f in os.listdir(csv_dir) if f.
    endswith('.csv')]

dataframes = {}
for csv_file in csv_files:
    file_path = os.path.join(csv_dir, csv_file)
    df_name = os.path.splitext(csv_file)[0].replace(
    '-', '_')
    dataframes[df_name] = pd.read_csv(file_path)

for df_name in dataframes.keys():
    print(df_name)
```

Listing 5: Namespace and Data Import

3.2 Author – [Write] → ResearchPaper

The ABOX definition code and ontology create code are put in the iteration:

```python
g.add((URI.Author, URI.Write, URI.ResearchPaper))

def Write_process(df):
    for _, record in df.iterrows():
        # Create URIs
        Author_id_uri = URIRef(URI.Author + "_" +
    str(record['author_id']))
        ResearchPaper_id_uri = URIRef(URI.
    ResearchPaper + "_" + str(record['paper_id']))

        g.add((Author_id_uri, RDF.type, URI.Author))
        g.add((ResearchPaper_id_uri, RDF.type, URI.
    ResearchPaper))

        g.add((Author_id_uri, URI.Write,
    ResearchPaper_id_uri))

Write_process(dataframes['author_write'])
```

Listing 6: Author – [Write] → ResearchPaper

In the code $URIRef(URI.Author + ”_” + str(record['author_id']))$, we are creating ABOX URI Instances, the code $g.add((Author_id_uri, RDF.type, URI.Author))$ is creating a relation between ABOX and TBOX by RDF.type, and the code $g.add((Author_id_uri, URI.Write, ResearchPaper_id_uri))$ is creating the relation between ABOX Instances. All this happened under the iteration which iterate all the rows in the data dataframes['$author_write$'].

3.3 ResearchPaper – [submitted] → SubmittedPaper

The ABOX definition code and ontology create code are put in the iteration:

```python
g.add((URI.ResearchPaper, URI.Submitted, URI.
    SubmittedPaper))

g.add((URI.ResearchPaper, URI.PaperTitle, XSD.string
    ))

g.add((URI.ResearchPaper, URI.PaperAbstract, XSD.
    string))

def Submitted_process(df):
    for _, record in df.iterrows():
        # Create URIs
        paper_id_uri = URIRef(URI.ResearchPaper + "_
    " + str(record['DOI']))
        submitted_paper_id_uri = URIRef(URI.
    SubmittedPaper + "_" + str(record['DOI']))
```

```python

        g.add((submitted_paper_id_uri, RDF.type, URI
    .SubmittedPaper))

        # Add the relationship to the graph
        g.add((paper_id_uri, URI.Submitted,
    submitted_paper_id_uri))

        g.add((paper_id_uri, URI.PaperTitle, Literal
    (record['title'], datatype=XSD.string)))
        g.add((paper_id_uri, URI.PaperAbstract,
    Literal(record['abstract'], datatype=XSD.string
    )))

Submitted_process(dataframes['paper'])
```

Listing 7: ResearchPaper – [submitted] $\rightarrow$ SubmittedPaper

This is another example, we create the ABOX definition, and link it with TBOX. The full version of the code is available at $2_ABOX_create.ipynb$.

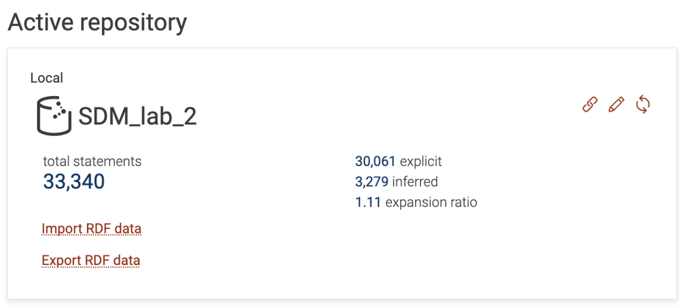
3.4 Final Ontology

As is shown previously, the connection between TBOX and ABOX is done while creating the ABOX. Using the *graph.add()* function, the instances are added by iterating over the datasets. *RDF.type* from the namespace indicates that the added instance is a type of a class from the namespace. For example, the code adding ResearchPaper is shown as Listing 8.

```python
g.add((ResearchPaper_id_uri, RDF.type, URI.
    ResearchPaper))
```

Listing 8: Adding Research Paper in ABOX

The repository aftering importing into GraphDB is shown as Figure 3. The class relationship and number of links among the instances are shown as Figure 4.


Active repository

Local

SDM_lab_2

total statements
33,340

30,061 explicit
3,279 inferred
1.11 expansion ratio

Import RDF data

Export RDF data

Figure 3: GraphDB Repository

The number of instances in therms of classes is shown as Table 2.

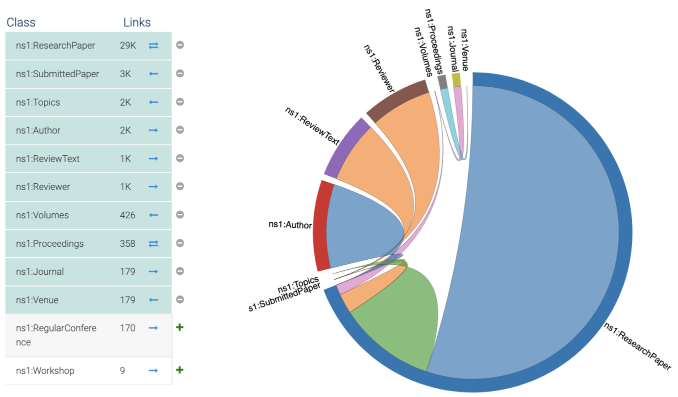

Class	Links
ns1:ResearchPaper	29K ⇄
ns1:SubmittedPaper	3K ←
ns1:Topics	2K ←
ns1:Author	2K →
ns1:ReviewText	1K →
ns1:Reviewer	1K →
ns1:Volumes	426 ←
ns1:Proceedings	358 ⇄
ns1:Journal	179 →
ns1:Venue	179 ←
ns1:RegularConference	170 →
ns1:Workshop	9 →

Figure 4: Class Relationship

Class	Number of Instances
Reviewer	1057
Author	1942
Person	2999
ResearchPaper	483
SubmittedPaper	483
Topics	2016
ReviewText	1491
Journal	116
Volume	179
Proceedings	179
Venue	130
Workshop	9
RegularConference	170
Conference	179

Table 2: Instances Numbers

4 Querying the ontology by SPARQL

In this section, we will use 6 queries to test whether our creation of TBOX and ABOX is correct.

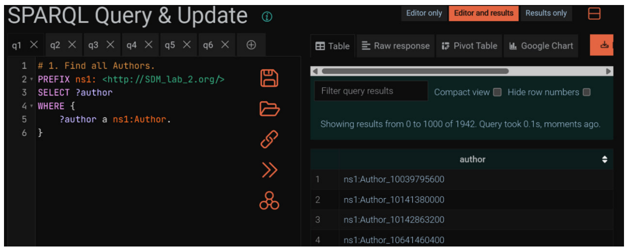
4.1 Find all Authors.

In this code we first set the namespace same as TBOX and ABOX, then we select the Author by setting the SELECT Clause to? author, the WHERE clause we use a which is short of rdf:type for asking all the subjects having a type ns1:Author.

```sparql
# 1. Find all Authors.
PREFIX ns1: <http://SDM_lab_2.org/>
SELECT ?author
WHERE {
    ?author a ns1:Author.
}
```

Listing 9: Find all Authors

The running result:



```
# 1. Find all Authors.
PREFIX ns1: <http://SDM_lab_2.org/>
SELECT ?author
WHERE {
    ?author a ns1:Author.
}
```

Showing results from 0 to 1000 of 1942. Query took 0.1s, moments ago.

	author
1	ns1:Author_10039795600
2	ns1:Author_10141380000
3	ns1:Author_10142863200
4	ns1:Author_10641460400

Figure 5: Q1 Running Result

4.2 Find all properties whose domain is Author.

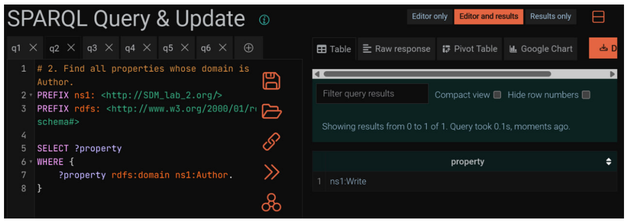
In this code, we are making use of rdfs:domain to find the properties whose domain is Author:

```
# 2. Find all properties whose domain is Author.
PREFIX ns1: <http://SDM_lab_2.org/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?property
WHERE {
    ?property rdfs:domain ns1:Author.
}
```

Listing 10: Find all properties whose domain is Author.

The running result:



```
# 2. Find all properties whose domain is
Author.
PREFIX ns1: <http://SDM_lab_2.org/>
PREFIX rdfs: <http://www.w3.org/2000/01/r
schema#>

SELECT ?property
WHERE {
    ?property rdfs:domain ns1:Author.
}
```

Showing results from 0 to 1 of 1. Query took 0.1s, moments ago.

	property
1	ns1:Write

Figure 6: Q2 Running Result

4.3 Find all properties whose domain is either Conference or Journal.

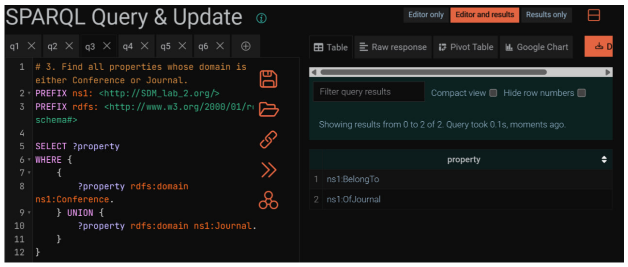
In this code, we are making use of rdfs:domain to find the properties whose domain is either Conference or Journal:

```
# Find all properties whose domain is either
    Conference or Journal.
PREFIX ns1: <http://SDM_lab_2.org/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?property
WHERE {
    {
        ?property rdfs:domain ns1:Conference.
    } UNION {
        ?property rdfs:domain ns1:Journal.
    }
}
```

Listing 11: Find all properties whose domain is either Conference or Journal.

The running result:



```
# 3. Find all properties whose domain is
either Conference or Journal.
PREFIX ns1: <http://SDM_lab_2.org/>
PREFIX rdfs: <http://www.w3.org/2000/01/r
schema#>

SELECT ?property
WHERE {
    {
        ?property rdfs:domain
ns1:Conference.
    } UNION {
        ?property rdfs:domain ns1:Journal.
    }
}
```

Showing results from 0 to 2 of 2. Query took 0.1s, moments ago.

	property
1	ns1:BelongTo
2	ns1:OfJournal

Figure 7: Q3 Running Result

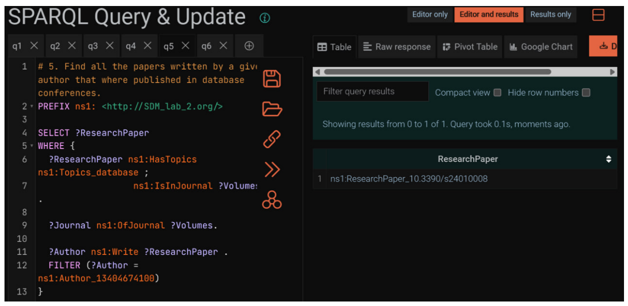
4.4 Find all the papers written by a given author that where published in database Journals.

As mentioned in Q4, we are basicly doing the same thing as Q4, but this time we searched in Journals, and we get the result correct:

```
# 5. Find all the papers written by a given author
    that where published in database Journals.
PREFIX ns1: <http://SDM_lab_2.org/>
SELECT ?ResearchPaper
WHERE {
  ?ResearchPaper ns1:HasTopics ns1:Topics_database ;
                 ns1:IsInJournal ?Volumes .
  ?Journal ns1:OfJournal ?Volumes.
  ?Author ns1:Write ?ResearchPaper .
  FILTER (?Author = ns1:Author_13404674100)
}
```

Listing 12: Find all properties whose domain is either Conference or Journal.

The running result:



```
# 5. Find all the papers written by a give
author that where published in database
conferences.
PREFIX ns1: <http://SDM_lab_2.org/>

SELECT ?ResearchPaper
WHERE {
  ?ResearchPaper ns1:HasTopics
ns1:Topics_database ;
                 ns1:IsInJournal ?Volume:
.

  ?Journal ns1:OfJournal ?Volumes.

  ?Author ns1:Write ?ResearchPaper .
  FILTER (?Author =
ns1:Author_13404674100)
}
```

Showing results from 0 to 1 of 1. Query took 0.1s, moments ago.

	ResearchPaper
1	ns1:ResearchPaper_10.3390/s24010008

Figure 8: Q5 Running Result

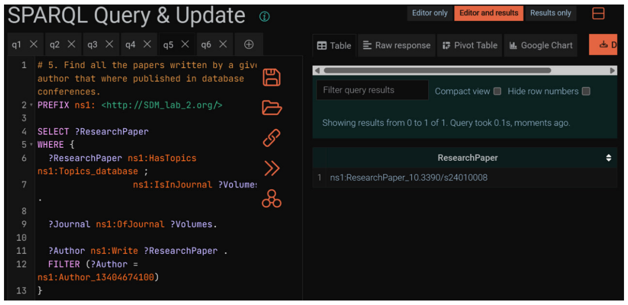
4.5 Find all the papers written by a given author that where published in database conferences.

In this code, we are making use ns1:HasTopics and ns1:Write to find the Author who write the paper in the database topics. Then we make use of FILTER to select a specific given author, and continuously we should make sure the paper is from the Proceedings of Conferences. Still, because our dataset is too small we don't have a database topics paper in the Conference, we will implement finding this paper in the Journal in Query 5.

```
PREFIX ns1: <http://SDM_lab_2.org/>
SELECT ?ResearchPaper
WHERE {
  ?ResearchPaper ns1:HasTopics ns1:Topics_database ;
                 ns1:IsInJournal ?Volumes .
  ?Journal ns1:OfJournal ?Volumes.
  ?Author ns1:Write ?ResearchPaper .
  FILTER (?Author = ns1:Author_13404674100)
}
```

Listing 13: Find all the papers written by a given author that where published in database conferences.

The running result:



```
# 5. Find all the papers written by a giv
author that where published in database
conferences.
PREFIX ns1: <http://SDM_lab_2.org/>

SELECT ?ResearchPaper
WHERE {
  ?ResearchPaper ns1:HasTopics
ns1:Topics_database ;
                 ns1:IsInJournal ?Volume:
.

  ?Journal ns1:OfJournal ?Volumes.

  ?Author ns1:Write ?ResearchPaper .
  FILTER (?Author =
ns1:Author_13404674100)
}
```

Showing results from 0 to 1 of 1. Query took 0.1s, moments ago.

	ResearchPaper
1	ns1:ResearchPaper_10.3390/s24010008

Figure 9: Q5 Running Result

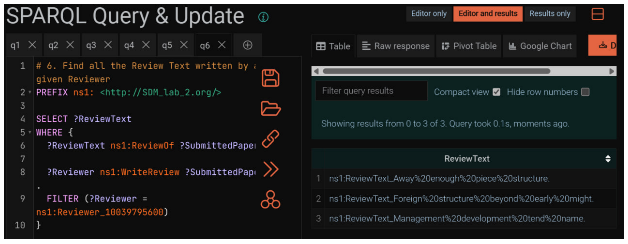
4.6 Find all the Review Text written by a given Reviewer.

In the last query, we decided to test the Reviewer and Review-Text:

```
# 6. Find all the Review Text written by a given
    Reviewer
PREFIX ns1: <http://SDM_lab_2.org/>
SELECT ?ReviewText
WHERE {
  ?ReviewText ns1:ReviewOf ?SubmittedPaper.
  ?Reviewer ns1:WriteReview ?SubmittedPaper .
  FILTER (?Reviewer = ns1:Reviewer_10039795600)
}
```

Listing 14: Find all the Review Text written by a given Reviewer.

The running result:



```
# 6. Find all the Review Text written by a
given Reviewer
PREFIX ns1: <http://SDM_lab_2.org/>

SELECT ?ReviewText
WHERE {
  ?ReviewText ns1:ReviewOf ?SubmittedPape

  ?Reviewer ns1:WriteReview ?SubmittedPap
.
  FILTER (?Reviewer =
ns1:Reviewer_10039795600)
}
```

Showing results from 0 to 3 of 3. Query took 0.1s, moments ago.

	ReviewText
1	ns1:ReviewText_Away%20enough%20piece%20structure.
2	ns1:ReviewText_Foreign%20structure%20beyond%20early%20might.
3	ns1:ReviewText_Management%20development%20tend%20name.

Figure 10: Q6 Running Result

5 Conclusion

In this project, we created TBOX and ABOX based on research publications domain data, the main working library used in TBOX definition, ABOX definition, and Ontology creation is RDFLib [2], in the last part, the GraphDB [1] is used to run 6 queries to test the creation of Ontology. According to the experiential running results, all designs work perfectly. The code of this project is available at github.com/Ziyong-Zhang/SDM_lab2/.

References

[1] Neo4j. Getting Started with Neo4j: Graph Database, current.

[2] RDFLib Community. RDFLib Documentation, stable.