SDM - Knowledge Graph

Rebekka Sihvola
Dmitriy Chukhray

May 29, 2024

Contents

Chapter 1

B.1 TBOX Definition

The first step in the creation of a knowledge graph for the research publication domain is the proper formation of a conceptual model, the so-called TBOX. TBOX was modelled in accordance with the lab's requirements except for proceedings' chairmen and journals' editors as these concepts were deemed irrelevant to us and our raw data did not have information about those people. We also decided to get rid of Affiliation and Proceeding classes as querying can be done without them and this way we did not need to create additional classes and respective subclass properties. Both Conference and Workshop classes were merged into one class called Edition. The distinction between them is implied by two different properties relating class Paper and class Edition through respective properties published_conference and published_workshop. The TBOX was written using the RDFLib library for Python. After the execution of the Python script, TBOX is saved in turtle format with a file name "tbox.ttl". The figure 1 is a visual representation of the created TBOX.

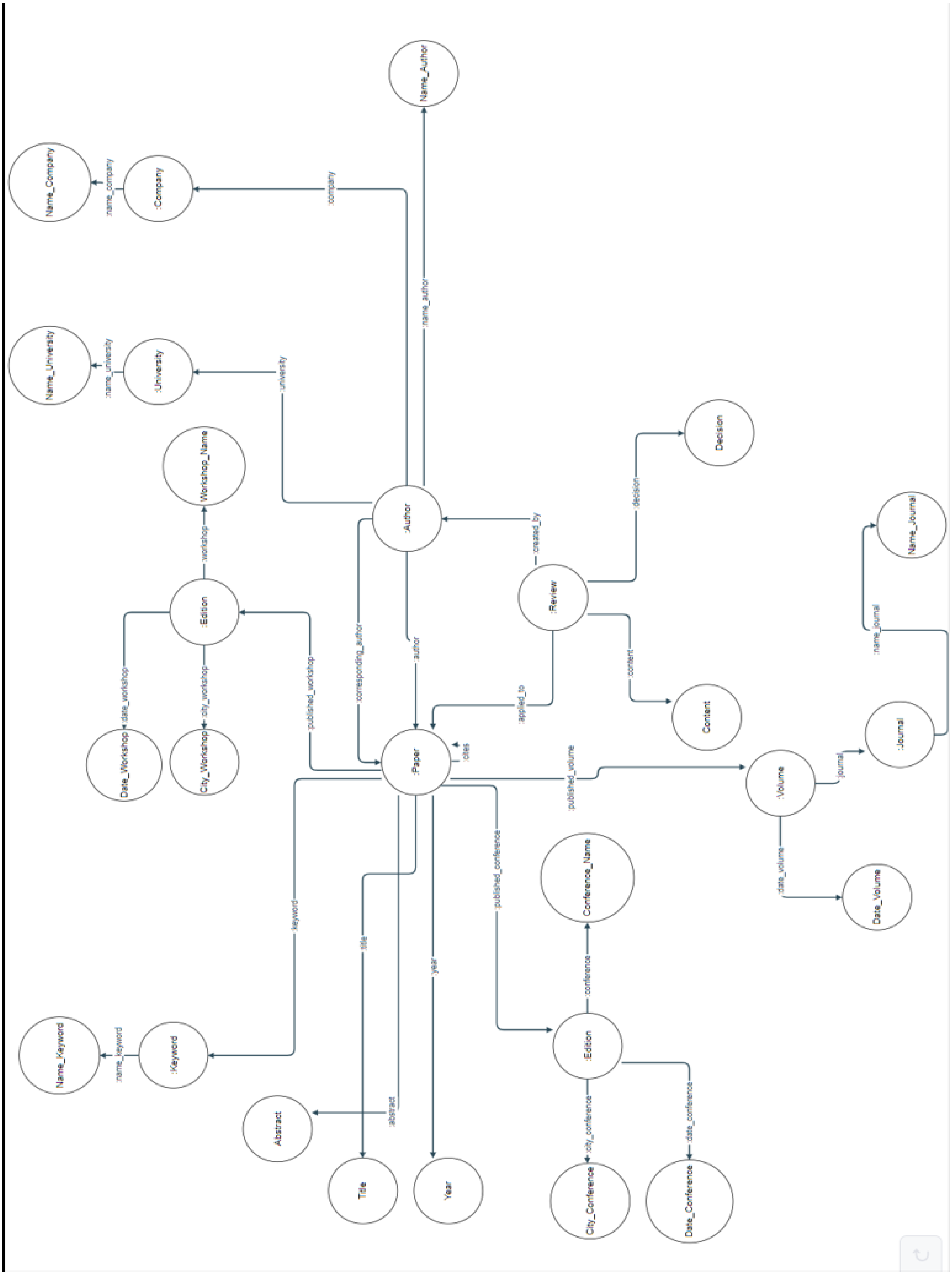

Figure 1.1: Project Architecture Overview.

Chapter 2

B.2 ABOX Definition

To create the ABOX we used the RDFLib library for Python. To properly create the ABOX we had to iterate through each preprocessed CSV file row by row and specify the correct columns for each knowledge graph property. After the execution of the Python script ABOX is saved in turtle format with a file name “abox.ttl”. However, we run into some problems with the property created_by which links classes Review and Author. For some reason, the output turtle file created by the Python script changes authors’ IDs from integer data type to float, even though we explicitly tried to set it to both string or integer and the preprocessed data has author IDs stored as integer data type. That is why we created a small additional Python script called “ttl.py” that takes the “abox.ttl” file as the input and produces the final ABOX turtle file called “abox_nozeros.ttl” with correct data types, which is going to be used in the next section.

Chapter 3

B.3 Create the final ontology

To create the connection between TBOX and ABOX we used the RDFLib library for Python. The script for the connection uses two previously created TBOX and ABOX turtle files as inputs ("tbox.ttl" and "abox_nozeros.ttl") and by the end of its execution it creates "tbox_plus_abox.ttl" turtle file. Then all we have left to do is to upload the newly created turtle file using GraphDB's import button as it was instructed in section A of this lab. RDFS (Optimized) was chosen as the inference entailment regime. This inference regime has all the rdf:type links we needed except for ¡property rdf:type RDF.Property¿ and that is why for every single property we had to specify them explicitly. Despite that, RDFS (Optimized) created some of the rdf:type links due to its inference which were ¡class rdf:type RDFS.Class¿ for all the existing objects of property triplets as properties' domains and ranges are always considered to be rdf:type of RDFS.Class according to RDFS (Optimized) ruleset.

Table 3.1: Number of classes and properties

Classes	Properties
9	24

Table 3.2: Number of instances per main classes

Review	Author	Keyword	Company	University	Paper	Volume	Edition	Journal
8471	3636	2500	1819	1816	1577	570	273	189

Table 3.3: Total number of triplets for main properties

Triplets
68357

Chapter 4

B.4 Querying the ontology

All queries are going to use 3 prefixes. We are going to list them now but for a cleaner look, we are not going to repeat them in every single query.

```
PREFIX lab2: <http://sdmlab2.org>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

Query 1. Find all authors. Let's assume we want to return both authors' IDs and names.

```
SELECT ?author ?name
WHERE {
    ?author rdf:type lab2:Author .
    ?author lab2:name_author ?name .
}
```

From our TBOX schema we know that all author instances are linked to lab2:Author classes through rdf:type link. These instances linked through lab2:name_author should return their corresponding names.

Query 2. Find all properties whose domain is Author

```
SELECT ?property
WHERE {
    ?property rdf:type rdf:Property .
    ?property rdfs:domain lab2:Author .
}
```

From our TBOX schema we know that all properties in the knowledge graph are of rdf:Property meta model class linked through rdf:type. By adding the second triplet we make sure we only return properties whose domain is lab2:Author class.

Query 3. Find all properties whose domain is either Conference or Journal.

```
SELECT ?property
WHERE {
  {
    ?property rdfs:domain lab2:Edition .
    ?property rdfs:label ?propertyName .
    FILTER (CONTAINS(UCASE(?propertyName), "CONFERENCE"))
  }
  UNION
  {
    ?property rdfs:domain lab2:Journal .
  }
}
```

Union conjunction is a logical OR operator that adds returns of two separate queries. Since class Edition represents both conferences and workshops the way to make sure we only look for properties whose domain is a conference we include a triplet that has a specific label which is conference. For journals, this is not needed because the Journal class represents only one concept.

Query 4. Find all the papers written by a given author that where published in database conferences.

```
SELECT (?pTitle AS ?paper_title) (?confName AS ?conferenceName)
WHERE
{
    ?paper rdf:type lab2:Paper .
    ?paper lab2:title ?pTitle .
    ?paper lab2:published_conference ?conference .
    ?paper lab2:keyword ?keyword .

    ?keyword rdf:type lab2:Keyword .
    ?keyword lab2:name_keyword "database" .

    ?author rdf:type lab2:Author .
    ?author lab2:name_author "Di Liu" .
    ?author lab2:author ?paper .
    ?conference lab2:conference ?confName .
}
```

Let's assume our given author is Di Liu and database conferences have only papers with keyword database. From our TBOX schema we know that all classes in the knowledge graph are linked through rdf:type to their corresponding URIs, therefore we need to make sure that ?paper is linked to lab2:Paper through rdf:type, ?keyword is linked to lab2:Keyword through rdf:type, and ?author is linked to lab2:Author through rdf:type. To return the names of the papers we need to traverse along lab2:title property. Since we are only interested in the papers published in conferences we need to add the triplet that connects ?paper to ?conference through lab2:published_conference property. Then, we check all keyword instances attached to the set of papers we are interested in and return only those that have "database" keyword. The same idea is applied to the authors of papers, we only want papers written by Di Liu.

Query 5. Find all the authors whose papers were never rejected.

```
SELECT DISTINCT ?author ?authorname ?papertitle
WHERE
{
    ?author rdf:type lab2:Author .
    ?paper rdf:type lab2:Paper .
    ?review rdf:type lab2:Review .
    ?author lab2:author ?paper .
    ?review lab2:applied_to ?paper .
    ?review lab2:decision "Accepted" .
    ?author lab2:name_author ?authorname .
    ?paper lab2:title ?papertitle .

    MINUS {
        ?rejectedPaper rdf:type lab2:Paper .
        ?rejectedReview rdf:type lab2:Review .
        ?author lab2:author ?rejectedPaper .
        ?rejectedReview lab2:applied_to ?rejectedPaper .
        ?rejectedReview lab2:decision "Rejected" .
    }
}
```

From our TBOX schema we know that all classes in the knowledge graph are linked through rdf:type to their corresponding URIs, therefore we need to make sure that ?paper is linked to lab2:Paper through rdf:type, ?review is linked to lab2:Review through rdf:type, ?author is linked to lab2:Author through rdf:type, ?rejectedPaper is linked to lab2:Paper through rdf:type, and ?rejectedReview is linked to lab2:Review through rdf:type. Then, in the first part of the query, before the MINUS statement, we retrieve authors who wrote paper(s) that received a review with the decision "Accepted". The query should return all authors whose papers were never rejected and without the second part of the query, the result might still have some authors with rejected papers. That is why we need the MINUS statement. The triplets inside of the MINUS statement return all authors who wrote papers that received "Rejected" decision. In the end, the authors we get in the second part of the query get excluded from the authors we get in the first part of the query and this is how we get authors without a single rejected paper. It is worth noting that most of the authors returned by this query wrote only one paper, but there are some authors that wrote two or three papers and none of them were rejected.

Query 6. Return the count of rejected papers for each university in a decreasing order.

```
SELECT ?universities ?universitiesnames (COUNT(?papers) AS ?rejectedPapersCount)
WHERE {
    ?authors rdf:type lab2:Author .
    ?universities rdf:type lab2:University .
    ?papers rdf:type lab2:Paper .
    ?reviews rdf:type lab2:Review .
    ?authors lab2:university ?universities .
    ?universities lab2:name_university ?universitiesnames .
    ?authors lab2:author ?papers .
    ?reviews lab2:applied_to ?papers .
    ?reviews lab2:decision "Rejected" .
}
GROUP BY ?universities ?universitiesnames
ORDER BY DESC(?rejectedPapersCount)
```

From our TBOX schema we know that all classes in the knowledge graph are linked through rdf:type to their corresponding URIs, therefore we need to make sure that ?authors is linked to lab2:Author through rdf:type, ?universities is linked to lab2:University through rdf:type, ?papers is linked to lab2:Paper through rdf:type, and ?reviews is linked to lab2:Review through rdf:type. Then, we want to make sure that authors who wrote papers are affiliated with universities. The decisions for those papers should only be "Rejected". The last part of the query is needed to aggregate data per university and its corresponding name in the decreasing order.