**Semantic Data Management**

GraphDB – Knowledge Graph

Dmitriy Chukhray

Rebekka Sihvola

**B.1 TBOX Definition**

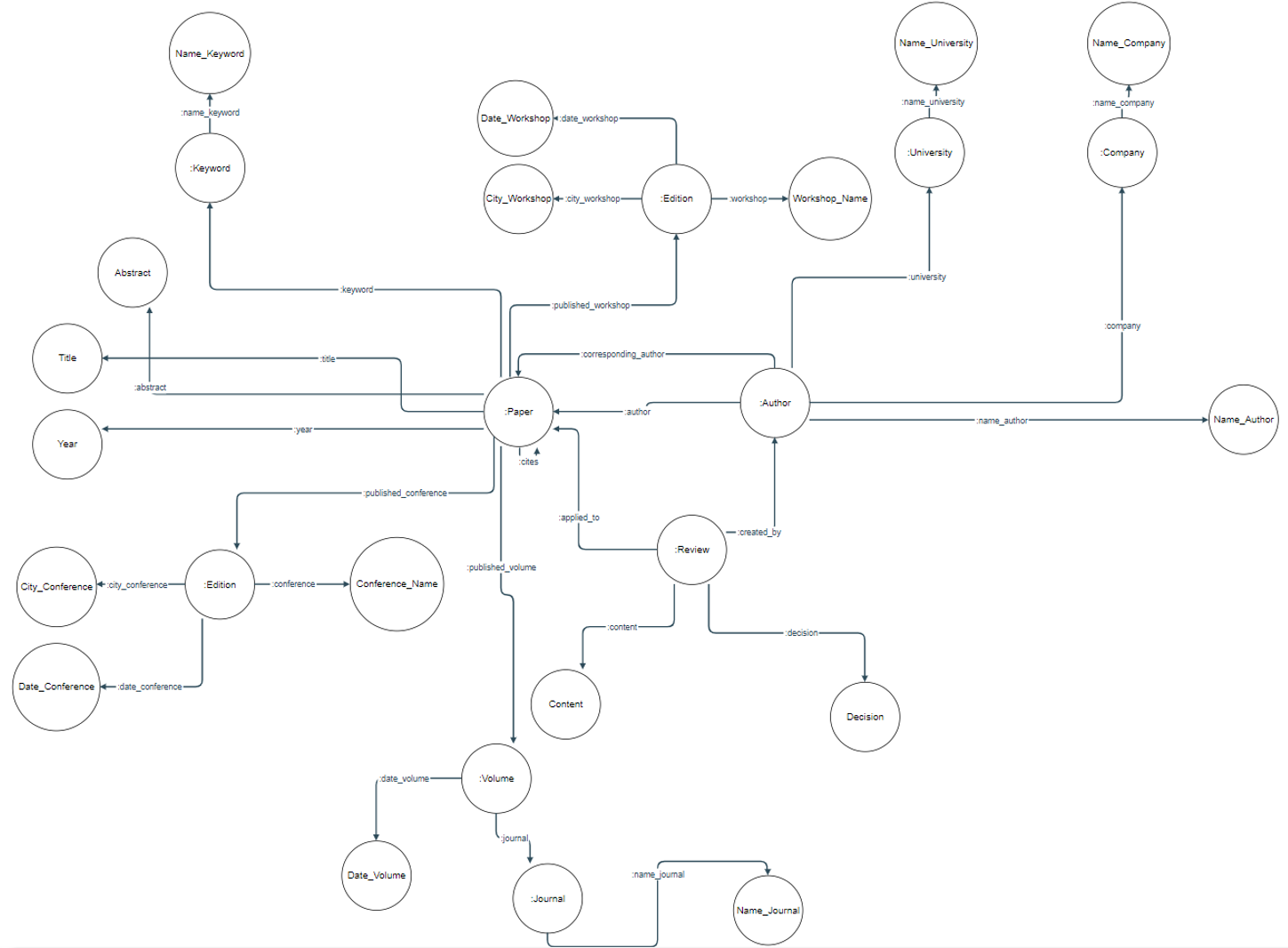
The first step in the creation of knowledge graph for the research publication domain is a proper formation of conceptual model, the so-called TBOX. TBOX was modeled in accordance with lab’s requirement except for proceedings’ chairmen and journals’ editors as they deemed irrelevant to us and our raw 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 RDFLib library for python. After the execution of the python script TBOX is being saved in turtle format with a file name “tbox.ttl”. The figure 1 is a visual representation of the created TBOX.

Figure 1: Visual representation of TBOX

**B.2 ABOX definition**

To create the ABOX we used 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 being saved in turtle format with a file name “abox.ttl”. However, we run into some problem 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 “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.

**B.3 Create the final ontology**

To create the connection between TBOX and ABOX we used 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 the section A of this lab. RDFS (Optimized) was chosen as the inference entailment regime. This inference regime has all 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 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.

**B.4 Querying the ontology**

All queries are going to use 3 prefixes. We are going to list them now but for 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  
 }  
}

From our TBOX schema we know that all properties in the knowledge graph are of rdf:Property meta model class linked through rdf:type. Union conjunction is a logical OR operator that adds returns of two separate queries. Since class Edition represent both conferences and workshops the way to make sure we only look for properties whose domain is conference we include a triplet that has a specific label which is conference. For journals this is not needed because 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 of rdfs:Class meta model class linked through rdf:type, 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,