**Semantic Data Management**

GraphDB – Knowledge Graph

Dmitriy Chukhray

Rebekka Sihvola

**B.1 TBOX Definition**

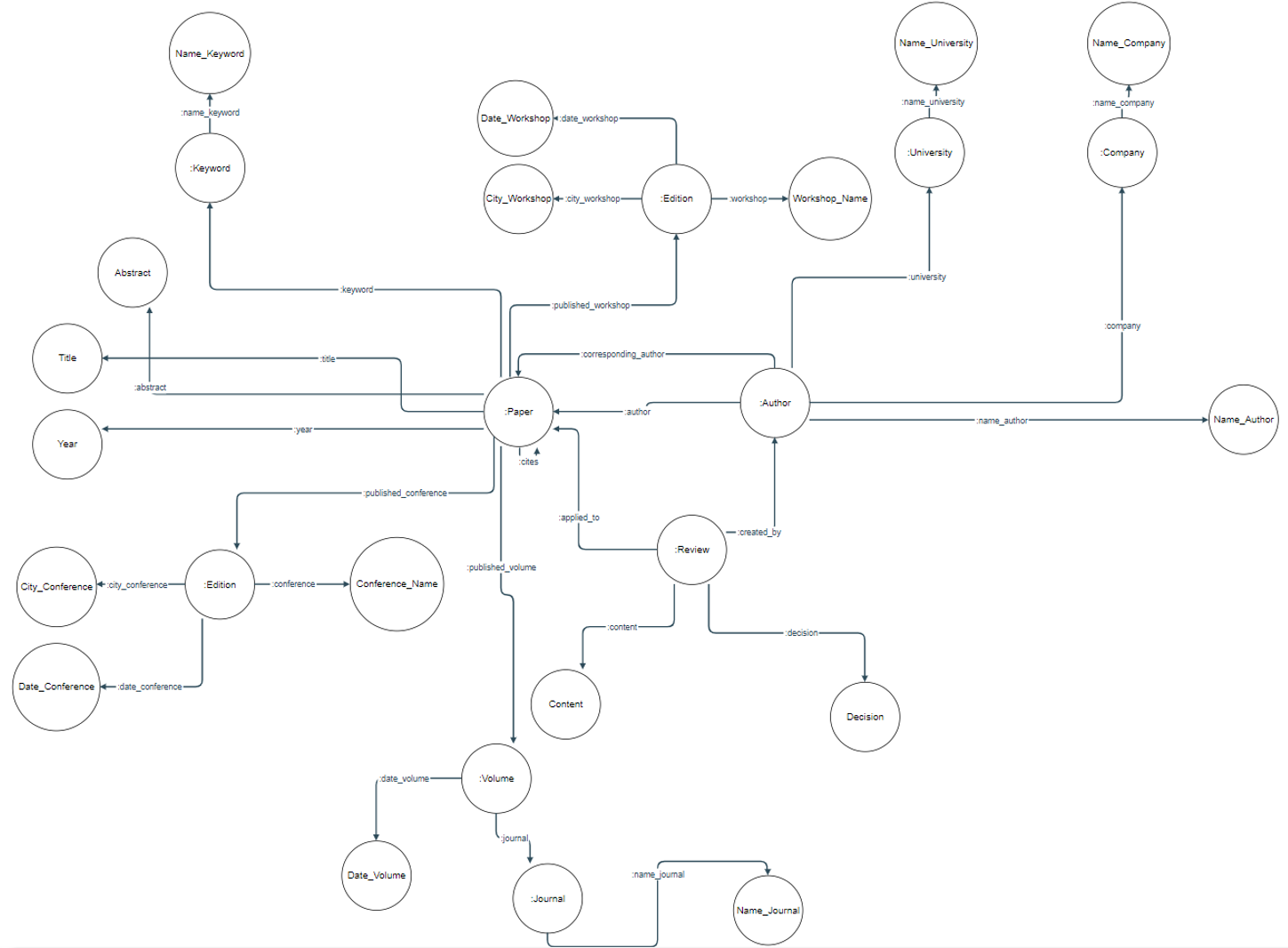
The first step in the creation of knowledge graph for the research publication domain is a proper formation of conceptual model, the so-called TBOX. TBOX was modeled in accordance with lab’s requirement except for proceedings’ chairmen and journals’ editors as they deemed irrelevant to us and our raw 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.