XML Project 4 - Report

Artyom K.
Ekeuh A.

May 2021

1 Introduction

This report is written about the implementation of fourth project. The first part of the project is to convert RDF into OWL representation. The second is about querying SPARQL endpoint. At the beginning we describe the code structure and then we go into architectural decisions we made during development.

2 Code Structure

We separated first and second part of the assignment into two files.

- part1.xml - represents OWL structure that had to be transformed.
- part2.sparql - the file for SPARQL queries.

3 Architectural Decisions

3.1 Conversion RDF to OWL

We decided to convert RDF into OWL document. We have selected Manchester syntax for doing that. The reason because Manchester syntax is a user-friendly syntax that is easy to read and write. We started the document with importing all prefixes that are being used in the document. After doing that we specify all properties that we have. Most of the properties have annotations, domain and range. Then, we have classes that are owl:Thing. Most of the classes have annotations and sub-classes. At the end of the document we specify individuals. We would say that the conversion is straightforward and there is not much to write about it, since the document is itself self explanatory.

3.2 SPARQL Queries

3.2.1 Query 1

In the first query we need to find unique affiliations. To do that we use distinct value that removes all possible duplicates. We use predicate *hasAffiliation* to

find affiliations.

3.2.2 Query 2

The second query is similar to the first query, but to find all CSO topics we use another predicate - *hasCsoEnhancedtopic*.

3.2.3 Query 3

In the third query first we need to find papers and corresponding journal name. We find the connection by use of *hasJourName* predicate. Then we use *hasCsoEnhancedTopic* to find topics for those journals. Then we grouping by topic and name and ordering in descend order to find the top 10.

3.2.4 Query 4

In the fourth query we need to find papers of iswc series and to count their references. First, we find papers and their conference names by *hasConfName* predicate. Also, we use *hasReference* predicate to get references for that paper. Finally, we use filter with contains function in order to find only conferenses that are from *iswc* series. To get them in decrease order we use a combination group by and order by with DESC function to get results in decrease order.

3.2.5 Query 5

This query finds all authors that have affiliation with ULB and their papers. First of all we use *hasPaper* predicate. Then we use a combination of *hasAffiliationDistribution* and *hasAffiliation* to find affiliations. Then we filter out only those affiliations that related to ULB by use of *contains* function.

3.2.6 Query 6

In this query we have federated query that connects to remote endpoint in order to get abstract for a specified paper. We use special construct to connect to the remote endpoint. The construct is:

```
SERVICE URI {
    # BODY
}
```

3.2.7 Query 7

The final query is a combination of query 5 and query 6. We need to find authors with affiliation with ULB and get abstracts from remote endpoint by use of federated query.

```
PREFIX aida35kschema: <http://aida.kmi.open.ac.uk/aida35k/ontology#>
PREFIX aida35k: <http://aida.kmi.open.ac.uk/aida35k/resource/>
PREFIX ns4: <http://purl.org/dc/terms/>

select ?paper ?abstract  where {
    ?author aida35kschema:hasPaper ?paper .
    ?paper aida35kschema:hasAffiliationDistribution ?ad .
    ?ad aida35kschema:hasAffiliation ?affiliation .
    filter contains(?affiliation,"bruxelles") .
    BIND(STRAFTER(str(?paper),
    "http://aida.kmi.open.ac.uk/aida35k/resource/p_")
    AS ?resource_id).
    bind(IRI(concat("https://makg.org/entity/", ?resource_id)) as ?uri)
    SERVICE <https://makg.org/sparql> {
        ?uri ns4:abstract ?abstract
    }
}
```

We use string after function to get only resource id from the paper from our endpoint. Then we use another bind function to concat full URL. Finally we use SERVICe to get abstract by use of following triple:

```
    ?uri ns4:abstract ?abstract
```

3.3 Conclusion

To conclude, we have converted RDF into OWL. Then we have succesfully were able to write all 7 queries that were required in this assignment.