Issue Report

Issue Link : https://github.com/RDFLib/rdflib/issues/798

Description:

This issue is related to not working of SPARQL Aggregation query that is, "*The query seems to execute, but when I try to iterate over its rows and print them, Jupyter Notebook goes on thinking forever and doesn't return anything. Could it be that I've made a mistake and have created some kind of recursive infinite loop? Any help would be much appreciated.*"

Given Ontology snippet:

```
<owl:Class rdf:about="http://human.owl#NCI_C12909"><rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Hematopoietic_System</rdfs:lab
el><rdfs:subClassOf
rdf:resource="http://human.owl#NCI_C41166"/>rdfs:subClassOfowl:Restriction<owl:onProperty
rdf:resource="http://human.owl#UNDEFINED_part_of"/><owl:someValuesFrom
rdf:resource="http://human.owl#NCI_C41165"/></owl:Restriction></rdfs:subClassOf><oboInO
wl:hasRelatedSynonym
rdf:resource="http://human.owl#genid7506"/><oboInOwl:hasRelatedSynonym
rdf:resource="http://human.owl#genid7507"/><oboInOwl:hasRelatedSynonym
rdf:resource="http://human.owl#genid7508"/><oboInOwl:hasRelatedSynonym
rdf:resource="http://human.owl#genid7509"/><oboInOwl:hasRelatedSynonym
rdf:resource="http://human.owl#genid7510"/><oboInOwl:hasRelatedSynonym
rdf:resource="http://human.owl#genid7511"/><oboInOwl:hasRelatedSynonym
rdf:resource="http://human.owl#genid7513"/></owl:Class>
```

Given SPARQL Query:

```
"""
SELECT ?node ?nodeLabel ?superclass ?superclassLabel (group_concat(DISTINCT ?node2) as
?node2s) (group_concat(DISTINCT ?node2Label) as ?node2Labels) where {
?node rdf:type owl:Class .
?node rdfs:subClassOf ?superclass .
OPTIONAL { ?node rdfs:subClassOf ?restriction }
OPTIONAL { ?restriction a owl:Restriction }
OPTIONAL { ?restriction owl:someValuesFrom ?node2 }
?node rdfs:label ?nodeLabel .
?superclass rdfs:label ?superclassLabel .
OPTIONAL { ?node2 rdfs:label ?node2Label }
}
group by ?node ?nodeLabel ?superclass ?superclassLabel ?node2 ?node2Label
LIMIT 10
```

"""

Solution For the given issue:

We observed this issue and found some solutions as given below:

1. After observing we found given ontology snippet was an RDF/XML formatted ontology. We debugged this ontology using protégé and filled all the missing part of it. We inserted different required prefixes. We eliminated some bugs they were in given ontology and marked as red.

Corrected Ontology:

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.semanticweb.org/gaurav/ontologies/2020/3/untitled-ontology-
34#"
    xml:base="http://www.semanticweb.org/gaurav/ontologies/2020/3/untitled-ontology-34"
    xmlns:ns="http://www.semanticweb.org/gaurav/ontologies/2020/3/untitled-ontology-34"
    xmlns:ns1="ns:"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:oboInOwl="http://www.geneontology.org/formats/oboInOwl#"

xmlns:cpannotationschema="http://www.ontologydesignpatterns.org/schemas/cpannotationsc
hema.owl#">
<owl:Class rdf:about="http://human.owl#NCI_C12909">
<rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Hematopoietic_System</rdfs:lab
el>
<rdfs:subClassOf rdf:resource="http://human.owl#NCI_C41166"/>
<rdfs:subClassOf owl:Restriction="http://www.semanticweb.org/gaurav"/>
<owl:onProperty rdf:resource="http://human.owl#UNDEFINED_part_of"/>
<owl:someValuesFrom rdf:resource="http://human.owl#NCI_C41165"/>
 <oboInOwl:hasRelatedSynonym
rdf:resource="http://www.semanticweb.org/gaurav/ontologies/2020/3/"/>
<oboInOwl:hasRelatedSynonym rdf:resource="http://human.owl#genid7506"/>
<oboInOwl:hasRelatedSynonym rdf:resource="http://human.owl#genid7507"/>
<oboInOwl:hasRelatedSynonym rdf:resource="http://human.owl#genid7508"/>
<oboInOwl:hasRelatedSynonym rdf:resource="http://human.owl#genid7509"/>
<oboInOwl:hasRelatedSynonym rdf:resource="http://human.owl#genid7510"/>
```

```
<oboInOwl:hasRelatedSynonym rdf:resource="http://human.owl#genid7511"/>
<oboInOwl:hasRelatedSynonym rdf:resource="http://human.owl#genid7513"/>
 </owl:Class>
 </rdf:RDF>
```

2. We also found a similar obo ontology which is in project's ontology directory named as OboOntologyHuman.owl. It's size is 83 MB and this is similar to the given obo ontology. So we can assume that size of given ontology in issue was similar to this ontology.
3. After observing the given query with the help of Fuseki server we inserted some prefixes in the query and this query works fine. But this issue was related to this query's running time which was sufficient enough to show that jupyter notebook is in infinite loop. So we observed this issue and try to optimize the query as following:

Optimized Query:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
SELECT ?node ?nodeLabel ?superclass ?superclassLabel (group_concat(DISTINCT ?node2) as
?node2s) (group_concat(DISTINCT ?node2Label) as ?node2Labels)
where {
?node rdf:type owl:Class .
  ?node rdfs:subClassOf ?superclass .
   OPTIONAL { ?node rdfs:subClassOf ?restriction.
           ?restriction a owl:Restriction ;
             owl:someValuesFrom ?node2 .
              ?node2 rdfs:label ?node2Label .
   }
   ?node rdfs:label ?nodeLabel .
   ?superclass rdfs:label ?superclassLabel .
}
group by ?node ?nodeLabel ?superclass ?superclassLabel ?node2 ?node2Label
LIMIT 10
```

This query runs faster and results are same with the previous query.

Why this optimization is correct?

1. If we have four separate independent optional clauses, the query engine has to always execute all the four optional operations, regardless of how many of those operations turn out to have a result. However, if you have a single optional clause, and the first pattern in that optional clause does not have a result, you have immediately eliminated 3 operations because the other three patterns no longer need resolution. They all fail or succeed together.

2. In this case output will be same. The reason is being that all four optional patterns are about the presence of the same construction (an OWL restriction), and they will usually all be present or absent together in this ontology so resultant triples will be same.