International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 4 (2012) pp. 208 -218
© MIR Labs, www.mirlabs.net/ijcisim/index.html

A Framework for SPARQL Query Processing, Optimization and Execution with Illustrations

Sanjay Kumar Malik[1] , **SAM Rizvi**[2]
[1] University School of Information Technology, GGS Indraprastha University, New Delhi
sdmalik@hotmail.com

[2] Deptt. of Computer Science,Jamia Millia Islamia, New Delhi
samsam_rizvi@yahoo.com

Abstract: **The vision of Semantic web is to allow intelligent description and interchange of integrated data from various distributed web resources. A structure for this metadata on web is known as Resource Description Framework (RDF) where data is in the form of XML (Extended Markup Language). A query language is used to retrieve such large RDF data effectively and efficiently which is known as SPARQL (Standard Protocol and RDF Query Language) which involves Query Processing, Optimization and Execution. In this paper, we propose a framework for SPARQL Query Processing, Optimization and Execution with various SPARQL illustrations in Twinkle and Jena ARQ. A "Furniture RDF" has been illustrated with "Filtering RDF using Twinkle" and "Filtering RDF using Jena ARQ on Eclipse" based on Java source code obtained after executing Eclipse.**

Keywords: RDF, SPARQL, Query Processing, Optimization, Execution, JENA ARQ, TWINKLE, Eclipse

I. Introduction

Sir Tim Berner's LEE, inventor of WWW(World Wide Web) and his W3C(World Wide Web Consortium) team along with others are working hard towards taking the current web to semantic web. The objective of the semantic web is to provide a better platform for the knowledge representation of linked data to allow machine processing on a global scale by adding logic,inference and rule systems to the web which allows data to be shared across different applications and boundaries[1].

Two important technologies for semantic web are XML(Extended Markup Language) and RDF(Resource Description Framework) where RDF was developed to extend XML which tends to make work easier for automated services and autonomous agents by providing semantic capability[1]. Therefore, information on web should be expressed in a meaningful way accessible to machines which may be achieved by Resource Description Framework (RDF) as a basic data format aiming to represent information about resources on the web[2].

Semantic Web requires much more expressive power than using ontology languages like XML,XMLS(XML Schema), RDF, RDFS(RDF Schema) and OWL(Web Ontology Language) used to describe the semantics and reasoning of resources/metadata which are available on the web and also identify the relationship between them.

There are a number of RDF query languages available, but Connected Services Framework (CSF) Profile Manager only supports queries written in SPARQL. A query language is used for querying RDF graphs known as SPARQL which stands for "Simple protocol and RDF query Language" ,which is basically an RDF query language that defines a data access protocol and standard query language to be used with the RDF data model.

SPARQL works for any RDF mapped data source. SPARQL query language has some similarity with SQL constructs and there are some tools available as open source like TWINKLE 2.0, Jena Framework with ARQ [3,4] processor on which SPARQL can be executed and tested. SPARQL Query can also be used to retrieve data from RDFS as well as from OWL[5] which can be created and executed on Ontology tool like Protégé which involves Query processing, Optimization and Execution. "Query Processing" is the internal steps taken by the Query Engine for the evaluation of the Query and requires some transformation and rewriting methodologies for the execution without changing the outcome of the query. Query engine executes the query but there is a requirement to have optimization concepts so that the query can retrieve data efficiently as it affects the execution performance. "Query optimization" defines some of the rules for rewriting query which transform execution structure of query. "Query Execution" is the step in which query engine evaluates the SPARQL query by generating the QEP (Query Execution Plan).

In this paper, Section 2 presents a framework for Query Processing, Optimization and Execution and steps for SPARQL Query Engine Evaluation. Section 3, 4 and 5 discusses about Query processing, Query Optimization and Query Execution respectively with various SPARQL illustrations using Twinkle and Jena ARQ in Section 6. Section 7 refers to a "Furniture RDF" illustrated with "Filtering RDF using Twinkle" and "Filtering RDF using Jena ARQ on Eclipse" based on Java source code obtained after executing Eclipse.

II. Proposed framework for query processing, optimization and query execution

We propose framework for Query Processing, Optimization and Query Execution as shown in figure1.

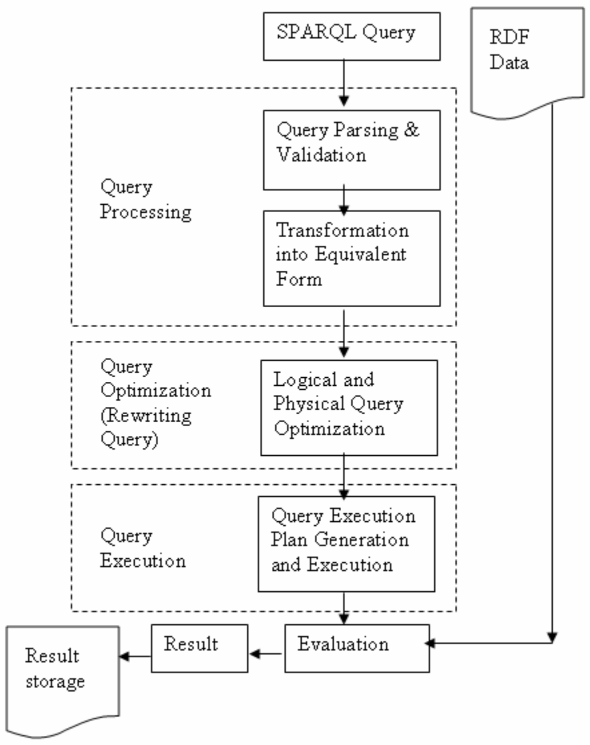

SPARQL Query
RDF Data
Query Processing
Query Parsing & Validation
Transformation into Equivalent Form
Query Optimization (Rewriting Query)
Logical and Physical Query Optimization
Query Execution
Query Execution Plan Generation and Execution
Result storage
Result
Evaluation

Figure 1. A Framework for SPARQL Query Processing, Optimization and Execution

It focus on the internal structure and transformation of the SPARQL query during the evaluation. It first shows the processing step in which Parsing and validation is done for the query and then equivalent transformation takes place in form of operator tree which is further passed to next transformation of Query optimization and then emphasise on the rewriting rule of the query for the better performance.

SPARQL Query Engine evaluation steps accomplishes many tasks for the evaluation of the SPARQL query. These steps are shown below:

1) SPARQL Query is written and passed to the Query Engine for the evaluation.
2) Query Engine then scans and parses for the SPARQL syntax and order of the semantic like keywords as SELECT, WHERE etc and then Validate for the RDF attributes.
3) Query Engine then Transform it into equivalent form.
4) After transformation into equivalent form, Rewriting of query is done by first logical optimization and then with physical optimization.
5) After performing optimization steps, it generates an QEP (Query Execution Plan) to execute the optimized execution for RDF data [6].
6) After all these steps, if all the steps are performed error free then engine take RDF file which is passed with SPARQL query for the evaluation step. Finally it returns the result. (Many Execution tools also provide the feature for storing the results which is represented as Result storage.

III. Query Processing

A high level query expressed in any high level query language is first being scanned, parsed and validated. The role of scanner is to identify the language tokens like keywords, attribute name and relation name, parser checks the syntax of the query to make sure that the query is formulated according to the grammar, then validated check attributes and relation names are valid or not. The query is converted into a tree data structure called query tree, which is internal representation of the query.

Another way of representing the query is using graph data structure called query graph. For the query execution the processor apply some optimization techniques on query graph or tree and optimize that graph for processing and produces an execution plan. Then, further the code to execute the plan is generated by query code generator. The runtime DB processor runs the query code to generate the result of the query[7].

SPARQL Query has a processing cycle to retrieve the data. In SPARQL query processing, SPARQL query is firstly parsed by the parser for any syntax error in which the keywords of SPARQL are identified. It also verifies the SPARQL query order e.g. SELECT, where FILTER option is in order or not. Validation is performed where RDF attributes are checked within SPARQL query.

Query Processing transforms this SPARQL then into its equivalent operator tree as shown in figure 4. This operator tree presents the execution sequence of the queries and is used further for the optimization purpose which is based on the SPARQL algebra.

IV. Query Optimization

Query optimization is the process of selecting the most suitable strategy for processing a query. Due to declarative nature of SPARQL(SPARQL Protocol and RDF Query Language), a query engine has to choose an efficient way to evaluate a query. All RDF repositories provide querying capabilities, but some of them do require manual interaction which minimizes the query execution time.

Query engines for ontological data based on graph models execute user queries without considering any optimization and especially for large ontologies, optimization techniques are required to ensure that query results are delivered within reasonable time.

Query optimization is the core concept of reordering / rewriting of SPARQL using some approach as selectivity estimation, which demonstrate how triples pattern reordering according to their selectivity affects the query execution performance[8]. The SPARQL query graph model and the transformation rules to rewrite a query into a semantically equivalent one, was proposed to find an efficient query execution plan.

One of the significant rule for optimization is rewriting FILTER variables.

V. Query Execution

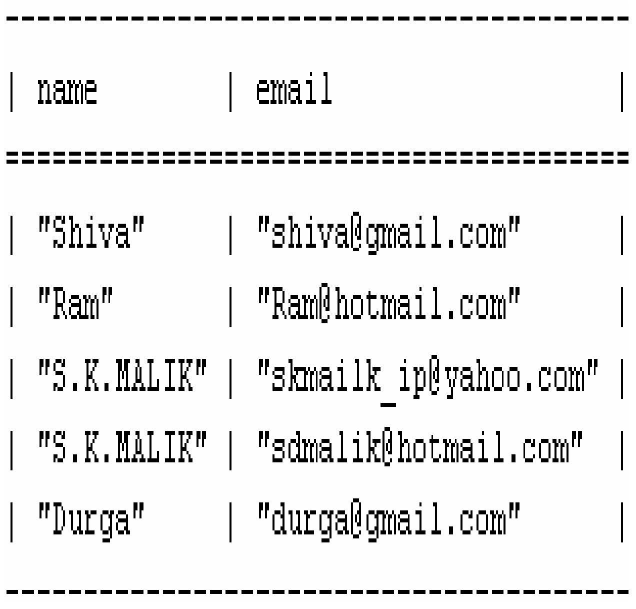
As for the evaluation of SPARQL, its Engine requires QEP to be generated, which is the code of execution of optimized operator tree. To Execute SPARQL Query, we have different tools available as open source on which SPARQL query can be executed. Among these tools TWINKLE and Jena ARQ is most popular. One sample SPARQL Query and its execution is demonstrated on JENA ARQ.

Query 1:- The below query will find the name and email of all employees.

Query syntax:-

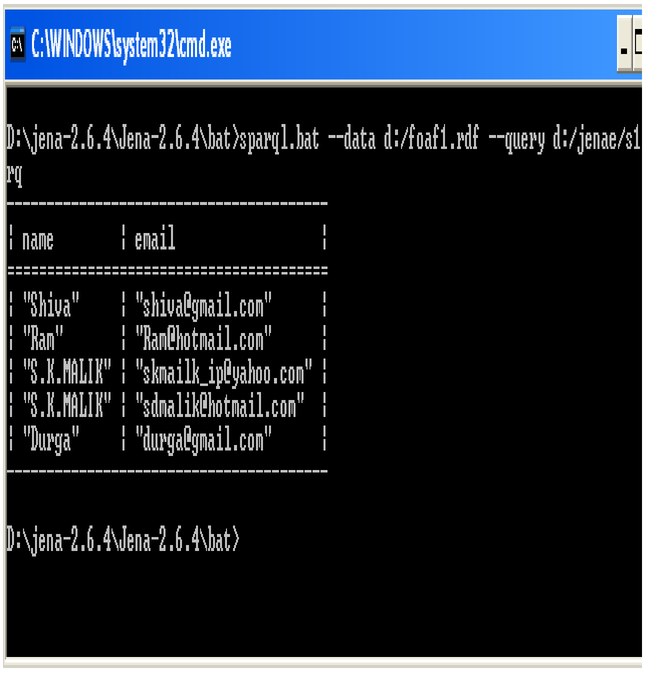
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-
ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
?person rdf:type foaf:Person .
?person foaf:name ?name .
OPTIONAL { ?person foaf:mbox_sha1sum ?email }
}
```

The Execution Output with Jena ARQ and Twinkle Tool is as shown below:



```
---------------------------------------
| name        | email                 |
=======================================
| "Shiva"     | "shiva@gmail.com"     |
| "Ram"       | "Ram@hotmail.com"     |
| "S.K.MALIK" | "skmailk_ip@yahoo.com" |
| "S.K.MALIK" | "sdmalik@hotmail.com"  |
| "Durga"     | "durga@gmail.com"     |
---------------------------------------
```

Figure 2. Execution with Jena ARQ



```
C:\WINDOWS\system32\cmd.exe

D:\jena-2.6.4\Jena-2.6.4\bat>sparql.bat --data d:/foaf1.rdf --query d:/jenae/s1
rq
---------------------------------------
| name        | email                 |
=======================================
| "Shiva"     | "shiva@gmail.com"     |
| "Ram"       | "Ram@hotmail.com"     |
| "S.K.MALIK" | "skmailk_ip@yahoo.com" |
| "S.K.MALIK" | "sdmalik@hotmail.com"  |
| "Durga"     | "durga@gmail.com"     |
---------------------------------------

D:\jena-2.6.4\Jena-2.6.4\bat>
```

Figure 3. Execution with TWINKLE tool.

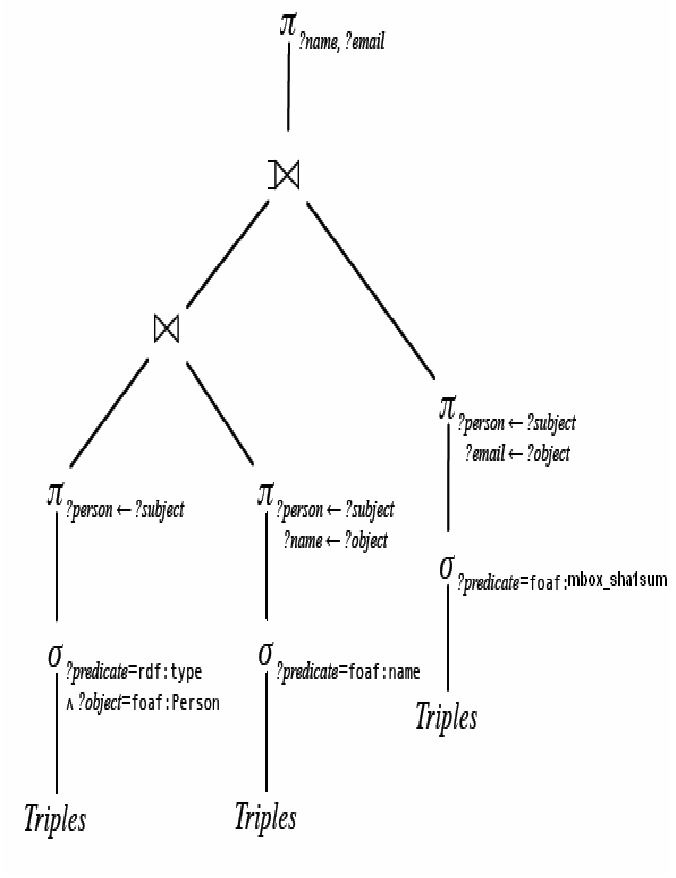

π ?name, ?email
⋈
⋈
π ?person ← ?subject
π ?person ← ?subject ?name ← ?object
π ?person ← ?subject ?email ← ?object
σ ?predicate=rdf:type ∧ ?object=foaf:Person
σ ?predicate=foaf:name
σ ?predicate=foaf:mbox_sha1sum
Triples
Triples
Triples

Figure 4. Operator Tree Equivalent to SPARQL Query

An operator tree (query tree) corresponds to the above SPARQL query which shows the execution of operators in SPARQL[9].

VI. SPARQL Illustrations

In this section, we present the SPARQL queries execution in which Projection, Selection, and then one example of rewriting rule (rewrite Filter variable) is demonstrated.

A. *Applying Projection*

Query 1: The below query will find the name, course, subject and marks of the student.

Syntax:-
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?student_name ?course ?subject ?marks
WHERE {
  ?x rdf:type foaf:Person .
  ?x foaf:studentname ?student_name .
  ?x foaf:course ?course .
  ?x foaf:subject ?subject .
  ?x foaf:marks ?marks .
}
order by ?student_name
```

The Execution Output with Jena ARQ and Twinkle Tool is as shown below:

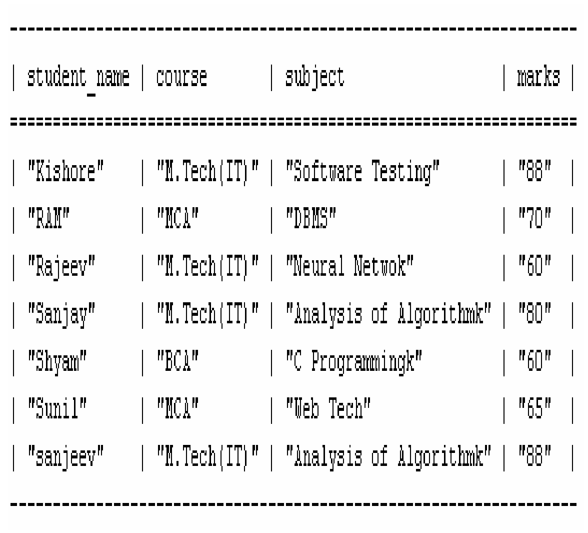
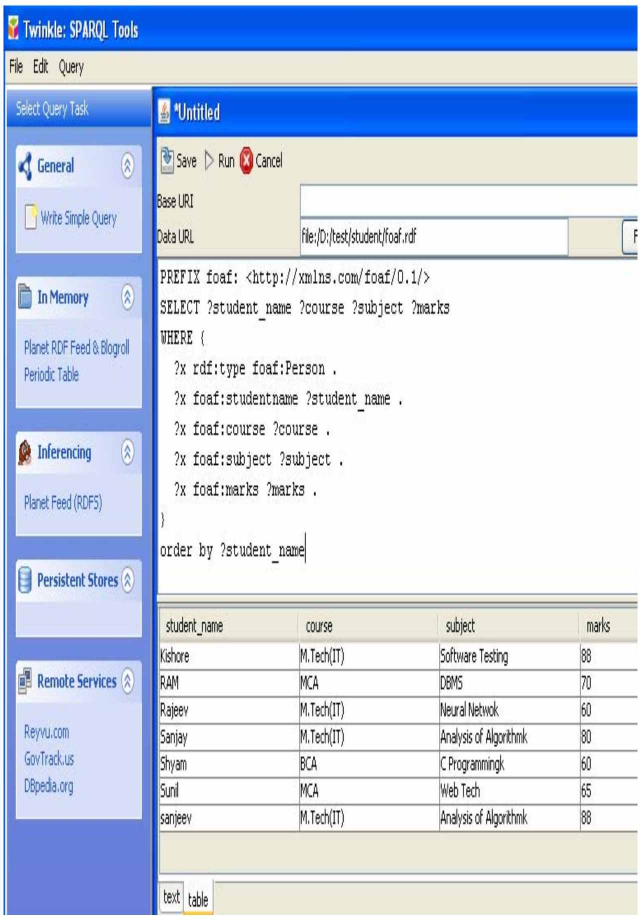

```
-----------------------------------------------------------------
| student_name | course     | subject                | marks |
=================================================================
| "Kishore"    | "M.Tech(IT)" | "Software Testing"     | "88"  |
| "RAM"        | "MCA"      | "DBMS"                 | "70"  |
| "Rajeev"     | "M.Tech(IT)" | "Neural Netwok"        | "60"  |
| "Sanjay"     | "M.Tech(IT)" | "Analysis of Algorithmk" | "80"  |
| "Shyam"      | "BCA"      | "C Programmingk"       | "60"  |
| "Sunil"      | "MCA"      | "Web Tech"             | "65"  |
| "sanjeev"    | "M.Tech(IT)" | "Analysis of Algorithmk" | "88"  |
-----------------------------------------------------------------
```

Figure 5. Execution Output with Jena ARQ

```
SELECT ?student_name ?course ?subject ?marks
WHERE {
  ?x rdf:type foaf:Person .
  ?x foaf:studentname ?student_name .
  ?x foaf:course ?course .
  ?x foaf:subject ?subject .
  ?x foaf:marks ?marks .
FILTER(?marks >= "70")
}
order by ?student_name
```

The Execution Output with Jena ARQ and Twinkle Tool are as shown below in figure 6 and 7:



Base URI
Data URL file:/D:/test/student/foaf.rdf

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?student_name ?course ?subject ?marks
WHERE {
  ?x rdf:type foaf:Person .
  ?x foaf:studentname ?student_name .
  ?x foaf:course ?course .
  ?x foaf:subject ?subject .
  ?x foaf:marks ?marks .
}
order by ?student_name
```

student_name	course	subject	marks
Kishore	M.Tech(IT)	Software Testing	88
RAM	MCA	DBMS	70
Rajeev	M.Tech(IT)	Neural Netwok	60
Sanjay	M.Tech(IT)	Analysis of Algorithmk	80
Shyam	BCA	C Programmingk	60
Sunil	MCA	Web Tech	65
sanjeev	M.Tech(IT)	Analysis of Algorithmk	88

Figure 6. Execution Output with Twinkle

B. *Applying Selection using Filter*

Query 2: The below query will find the detail of student will marks greater than equal to 70.

Syntax:-
```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
```

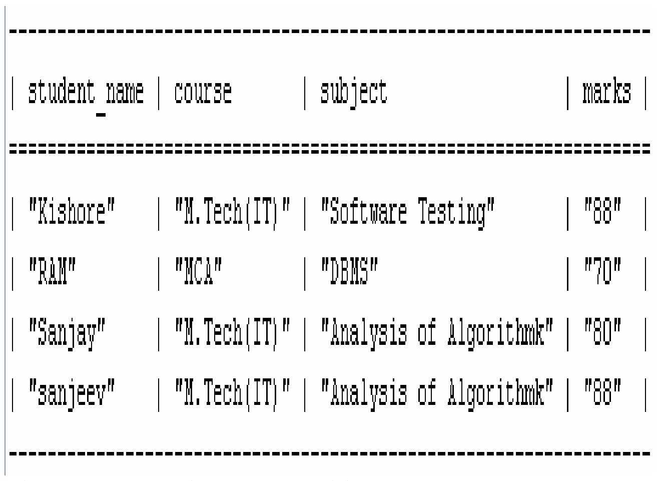
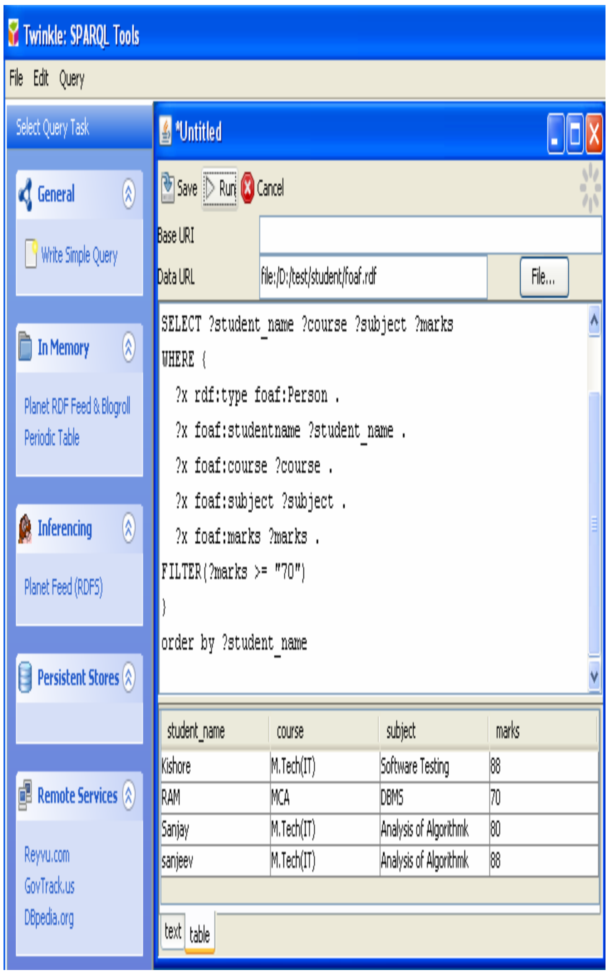

```
-----------------------------------------------------------------
| student_name | course     | subject                | marks |
=================================================================
| "Kishore"    | "M.Tech(IT)" | "Software Testing"     | "88"  |
| "RAM"        | "MCA"      | "DBMS"                 | "70"  |
| "Sanjay"     | "M.Tech(IT)" | "Analysis of Algorithmk" | "80"  |
| "sanjeev"    | "M.Tech(IT)" | "Analysis of Algorithmk" | "88"  |
-----------------------------------------------------------------
```

Figure 7. Execution Output with Jena ARQ



file:/D:/test/student/foaf.rdf

```
SELECT ?student_name ?course ?subject ?marks
WHERE {
  ?x rdf:type foaf:Person .
  ?x foaf:studentname ?student_name .
  ?x foaf:course ?course .
  ?x foaf:subject ?subject .
  ?x foaf:marks ?marks .
FILTER(?marks >= "70")
}
order by ?student_name
```

student_name	course	subject	marks
Kishore	M.Tech(IT)	Software Testing	88
RAM	MCA	DBMS	70
Sanjay	M.Tech(IT)	Analysis of Algorithmk	80
sanjeev	M.Tech(IT)	Analysis of Algorithmk	88

Figure 8. Execution Output with Twinkle

C. Applying Selection with more condition in Filter Clause

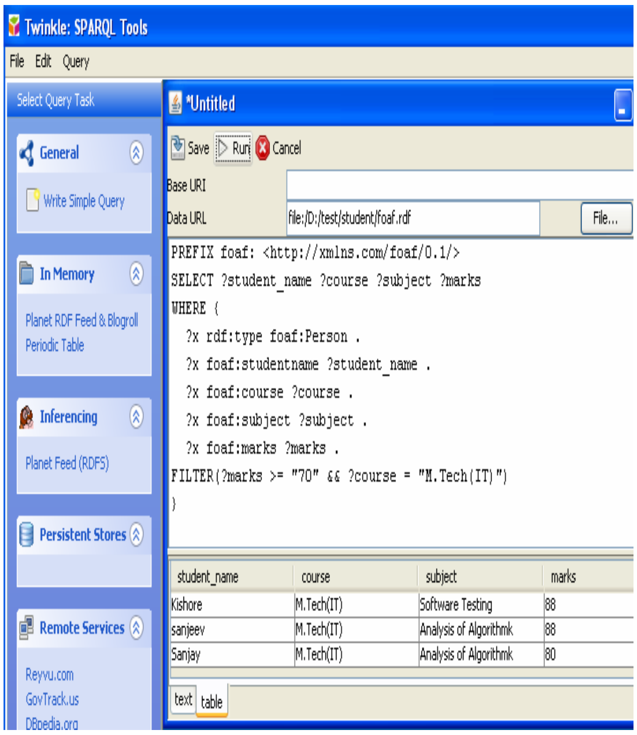
Query 3: The below query will find the detail of student will marks greater than equal to 70 and course is M.Tech(IT).

Syntax:-
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?student_name ?course ?subject ?marks
WHERE {
 ?x rdf:type foaf:Person .
 ?x foaf:studentname ?student_name .
 ?x foaf:course ?course .
 ?x foaf:subject ?subject .
 ?x foaf:marks ?marks .
FILTER(?marks >= "70" && ?course = "M.Tech(IT)") }

The Execution Output with Jena ARQ and Twinkle Tool is as shown below :

```
---------------------------------------------------------------
| student_name | course       | subject                | marks |
===============================================================
| "Kishore"    | "M.Tech(IT)" | "Software Testing"     | "88"  |
| "sanjeev"    | "M.Tech(IT)" | "Analysis of Algorithmk" | "88"  |
| "Sanjay"     | "M.Tech(IT)" | "Analysis of Algorithmk" | "80"  |
---------------------------------------------------------------
```

Figure 9. Execution Output with Jena ARQ



file:/D:/test/student/foaf.rdf

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?student_name ?course ?subject ?marks
WHERE {
  ?x rdf:type foaf:Person .
  ?x foaf:studentname ?student_name .
  ?x foaf:course ?course .
  ?x foaf:subject ?subject .
  ?x foaf:marks ?marks .
FILTER(?marks >= "70" && ?course = "M.Tech(IT)")
}
```

student_name	course	subject	marks
Kishore	M.Tech(IT)	Software Testing	88
sanjeev	M.Tech(IT)	Analysis of Algorithmk	88
Sanjay	M.Tech(IT)	Analysis of Algorithmk	80

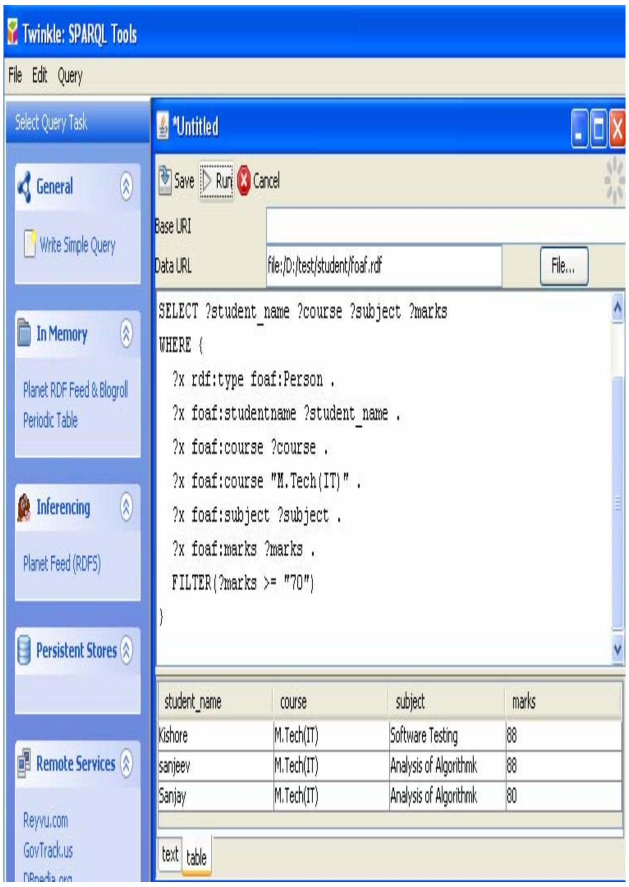
Figure 10. Execution Output with Twinkle

D. Applying Rewrite Filter Variable Rule of optimization

Query 4: The below query will find the detail of student with marks greater than equal to 70 and course is M.Tech(IT).

Syntax
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?student_name ?course ?subject ?marks
WHERE {
 ?x rdf:type foaf:Person .
 ?x foaf:studentname ?student_name .
 ?x foaf:course ?course .
 ?x foaf:course "M.Tech(IT)" .
 ?x foaf:subject ?subject .
 ?x foaf:marks ?marks .
 FILTER(?marks >= "70")
}

The Execution Output with Twinkle Tool is as shown below:



file:/D:/test/student/foaf.rdf

```
SELECT ?student_name ?course ?subject ?marks
WHERE {
  ?x rdf:type foaf:Person .
  ?x foaf:studentname ?student_name .
  ?x foaf:course ?course .
  ?x foaf:course "M.Tech(IT)" .
  ?x foaf:subject ?subject .
  ?x foaf:marks ?marks .
  FILTER(?marks >= "70")
}
```

student_name	course	subject	marks
Kishore	M.Tech(IT)	Software Testing	88
sanjeev	M.Tech(IT)	Analysis of Algorithmk	88
Sanjay	M.Tech(IT)	Analysis of Algorithmk	80

Figure 11. Execution Output with Twinkle

In this illustration, we apply the filter "M.Tech(IT)" clause before with the foaf:course. This is known as rewriting filter variable rule in which first, engine will execute "filter" condition to reduce the internal processing.

VI. SPARQL Illustration: "Furniture RDF"

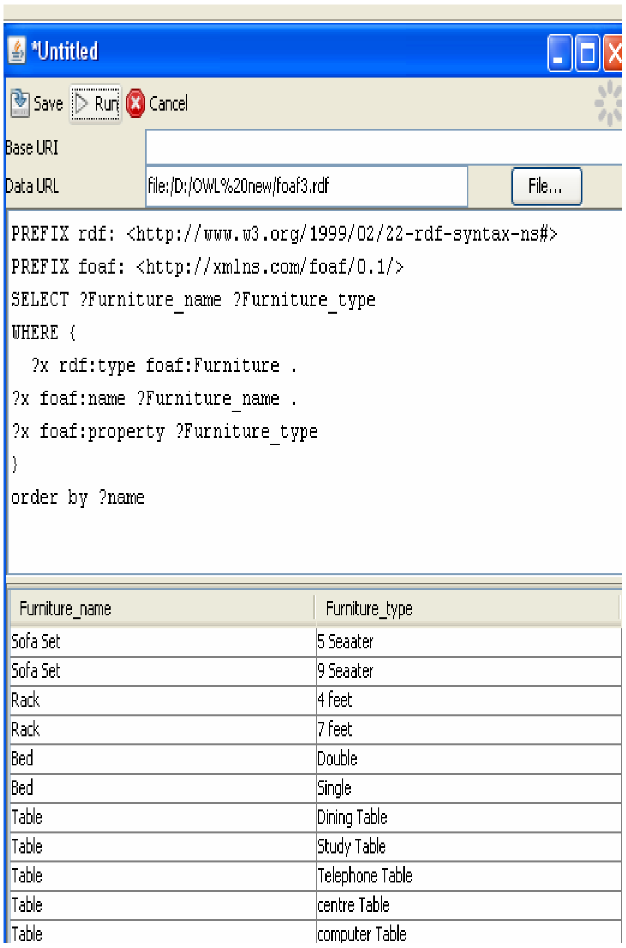
We take another example of a "Furniture RDF" with "furniture_name" and "furniture_type". SPARQL is a query language which is used to retrive data from RDF. We create a "Furniture RDF" with "name" and "property" triples with fields "name" and "type". Sparql Query Syntax for filtering data from RDF file is shown below:

```
SYNTAX :
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-
ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?type
        WHERE {
        ?x rdf:type foaf:Furniture .
        ?x foaf:name ?name .
        ?x foaf:property ?type
        }
        order by ?name
```

The above query is using the namespace "rdf" which will filter the data using the "foaf :Furniture". It Further executes over the said "Furniture RDF" file with two fields, "name" and "type"

A. Filtering RDF using Twinkle

When we execute the above SPARQL in TWINKLE Tool which is an open source editor for executing SPARQL, the following result output is obtained with "furniture name" and "furniture type".



file:/D:/OWL%20new/foaf3.rdf

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?Furniture_name ?Furniture_type
WHERE {
  ?x rdf:type foaf:Furniture .
?x foaf:name ?Furniture_name .
?x foaf:property ?Furniture_type
}
order by ?name
```

Furniture_name	Furniture_type
Sofa Set	5 Seaater
Sofa Set	9 Seaater
Rack	4 feet
Rack	7 feet
Bed	Double
Bed	Single
Table	Dining Table
Table	Study Table
Table	Telephone Table
Table	centre Table
Table	computer Table

Figure 12. SPARQL Query Execution in TWINKLE

Output:-

```
-------------------------------------
| Furniture_name | Furniture_type    |
=====================================
| "Sofa Set"     | "5 Seaater"       |
| "Sofa Set"     | "9 Seaater"       |
| "Rack"         | "4 feet"          |
| "Rack"         | "7 feet"          |
| "Bed"          | "Double"          |
| "Bed"          | "Single"          |
| "Table"        | "Dining Table"    |
| "Table"        | "Study Table"     |
| "Table"        | "Telephone Table" |
| "Table"        | "centre Table"    |
| "Table"        | "computer Table"  |
```

Figure 13. Data Retrieved from Furniture RDF

B. Filtering RDF using Jena ARQ on Eclipse

We create an RDF which describes the Furniture of different types with their properties using Jena (a toolkit for parsing RDF using Java). Now, using Eclipse (open source java editor), Jena and ARQ tool, the source code of java will be compiled into "TestFurniture.class" file. Using java interpreter, when the class file will be executed, it will fire the sparql query over the furniture RDF file and generate the result which is ordered by the "name" field in the said Furniture RDF, in alphabetical order. Following are the source code and output which demonstrates the filtering of data from the Furniture RDF file:

C. Java Source Code **: (**obtained after executing Eclipse**)**

Following source code is written in Java on Eclipse:

```
public class TestFurniture {
public static void main(String[] args)throws Exception {
InputStream in = new FileInputStream(new
File("FurnitureRdf.rdf"));
Model model =
ModelFactory.createMemModelMaker().createModel(null);
model.read(in,null);
in.close();
String queryString ="PREFIX rdf:
<http://www.w3.org/1999/02/22-rdf-syntax-ns#>" +
"PREFIX foaf: <http://xmlns.com/foaf/0.1/>"+
"SELECT ?name ?type"+
"    WHERE {"+
"?x rdf:type foaf:Furniture . "+
"           ?x foaf:name ?name . "+
"           ?x foaf:property ?type "+
"           }"+
"order by ?name";
Query query = QueryFactory.create(queryString);
QueryExecution qe = QueryExecutionFactory.create(query,
model);
ResultSet results = qe.execSelect();
ResultSetFormatter.out(System.out, results, query);
qe.close();
           } }
```

The following output is obtained after executing the above code:

OUTPUT

```
---------------------------------
| name       | type              |
=================================
| "Bed"      | "Double"          |
| "Bed"      | "Single"          |
| "Rack"     | "4 feet"          |
| "Rack"     | "7 feet"          |
| "Sofa Set" | "5 Seaater"       |
| "Sofa Set" | "9 Seaater"       |
| "Table"    | "Dining Table"    |
| "Table"    | "Study Table"     |
| "Table"    | "Telephone Table" |
| "Table"    | "centre Table"    |
| "Table"    | "computer Table"  |
---------------------------------
```

Figure 14. Data retrieved from "Furniture RDF" using Jena and ARQ with Eclipse

From the above output, we infer from the furniture RDF file, the different "names" of the furniture and their "types".

VII. Executing SPARQL on OWL in Protege

Following is the OWL Code Snippet obtained while creating the Ontology of IP University(IPU) using Protégé. We would execute a SPARQL Query on this code in protégé and obtain the output of ontology class hierarchy(subject/object).

IPU OWL FILE SNIPPET

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns:xsp="http://www.owl-
ontologies.com/2005/08/07/xsp.owl#"
xmlns:swrlb="http://www.w3.org/2003/11/swrlb#"
   xmlns="http://www.owl-
ontologies.com/Ontology1277277573.owl#"
 xmlns:swrl="http://www.w3.org/2003/11/swrl#"
xmlns:protege="http://protege.stanford.edu/plugins/owl/prot
ege#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xml:base="http://www.owl-
ontologies.com/Ontology1277277573.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Vice_Chancellor">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Vice Chancellor</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="GGS_IP_University"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="USCT">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >USCT</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Univ_School_of_Studies_USS_"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Academics">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="IPU_Campus_Administration"/>
    </rdfs:subClassOf>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Academics</rdfs:label>
  </owl:Class>
  <owl:Class rdf:ID="Asst_Registrars">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Asst Registrars</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Dyp_Registrar"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="USBT">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >USBT</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class
rdf:about="#Univ_School_of_Studies_USS_"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#Dyp_Registrar">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Dyp Registrar</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Controller_of_Examination"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Data_Entry_Operators">
    <rdfs:subClassOf rdf:resource="#Asst_Registrars"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Data Entry Operators</rdfs:label>
  </owl:Class>
    <owl:FunctionalProperty rdf:ID="Id">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string
"/>
    <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Id of COE</rdfs:comment>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Id</rdfs:label>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Qualification">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string
"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Qualification</rdfs:label>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Intake">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Intake</rdfs:label>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="IGIT_Class10029">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
    <rdfs:comment
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >ID of COE which is Unique</rdfs:comment>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >IGIT_Class10029</rdfs:label>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string
"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Year">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Year</rdfs:label>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Add">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Add</rdfs:label>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string
"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Sal">
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#float"/
>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Sal</rdfs:label>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Roll_No.">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string
"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Roll No.</rdfs:label>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Designation">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string
"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Designation</rdfs:label>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description
rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
          <owl:Class
rdf:about="#Controller_of_Examination"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="DOJ">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >DOJ</rdfs:label>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string
"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description
rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
          <owl:Class
rdf:about="#Controller_of_Examination"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Program">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string
"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Program</rdfs:label>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Ph._No.">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Ph. No.</rdfs:label>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#int"/>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description
rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
          <owl:Class
rdf:about="#Controller_of_Examination"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="Name">
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description
rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
          <owl:Class
rdf:about="#Controller_of_Examination"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Name</rdfs:label>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string
"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="NAME">
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string
"/>
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >NAME</rdfs:label>
  </owl:FunctionalProperty>
  <owl:FunctionalProperty rdf:ID="ID">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >ID</rdfs:label>
    <rdfs:domain>
      <owl:Class>
        <owl:unionOf rdf:parseType="Collection">
          <rdf:Description
rdf:about="http://www.w3.org/2002/07/owl#Thing"/>
          <owl:Class
rdf:about="#Controller_of_Examination"/>
        </owl:unionOf>
      </owl:Class>
    </rdfs:domain>
    <rdfs:range
rdf:resource="http://www.w3.org/2001/XMLSchema#string
"/>
    <rdf:type
rdf:resource="http://www.w3.org/2002/07/owl#DatatypePro
perty"/>
  </owl:FunctionalProperty>
  <Controller_of_Examination
rdf:about="IGIT_Class10032">
    <ID
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >1</ID>
    <Designation
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Professor , COE</Designation>
    <DOJ
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >2006</DOJ>
    <Name
rdf:datatype="http://www.w3.org/2001/XMLSchema#string
"
    >Prof.  Yogesh Singh</Name>
    <Ph._No.
rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
    >23900400</Ph._No.>
  </Controller_of_Examination>
</rdf:RDF>
<!-- Created with Protege (with OWL Plugin 3.4.1, Build
536)  http://protege.stanford.edu -->
```

Executing a SPARQL query on IPU Ontology (already created before) Using Protégé

Steps for executing SPARQL in Protégé 3.3.1 are:

1. Select ‘Open SPARQL Query Panel’ from OWL menu from the menu bar in protégé tool. (Version 3.3.1)

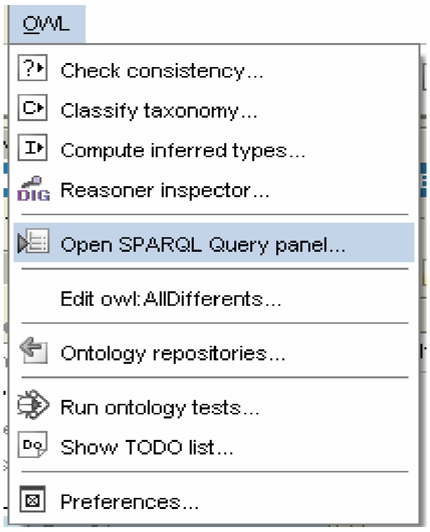
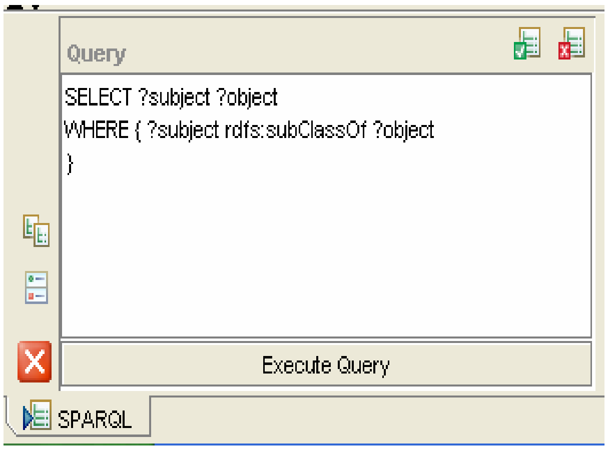


Figure 15. “Open SPARQL Query panel” of OWL Menu

2. Write the Following Query in the Query Panel and Press the ‘Execute Button’ at the bottom of the panel and the result will be displayed on the Right Side Section of the Query Panel.

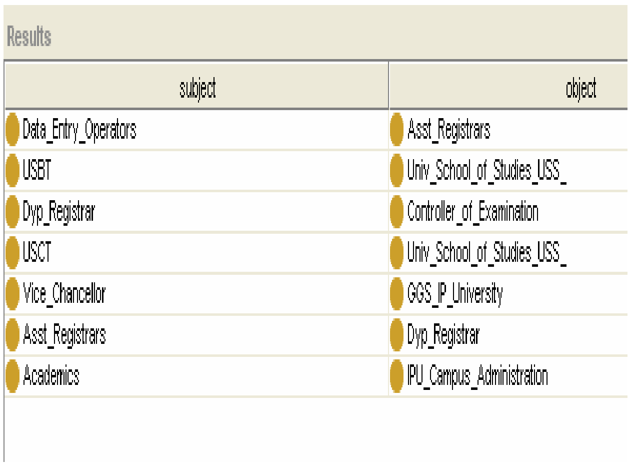
SPARQL Syntax:-
SELECT ?subject ?object
WHERE { ?subject rdfs:subClassOf ?object }



```
SELECT ?subject ?object
WHERE { ?subject rdfs:subClassOf ?object
}
```

Figure 16. Query written in the Query Panel

When the above SPARQL Query is executed on the OWL file of IPU Ontology created in Protégé , following result is obtained:



Results

subject	object
Data_Entry_Operators	Asst_Registrars
USBT	Univ_School_of_Studies_USS_
Dyp_Registrar	Controller_of_Examination
USCT	Univ_School_of_Studies_USS_
Vice_Chancellor	GGS_IP_University
Asst_Registrars	Dyp_Registrar
Academics	IPU_Campus_Administration

Figure 17. Output generated in Protégé

The above output snapshot displays the results of the SPARQL executed using Protégé which displays 'Subject' and 'Object' showing the subclass and superclass relationships. Eg; "Vice Chancellor(subject) is the subclass of "GGS_IP_University(object)"

VIII. Conclusions and future work

In this paper, we have presented a framework of SPARQL Query Processing, Optimization and Execution and SPARQL Execution using various tools like Twinkle and Jena ARQ illustrated with various examples which may be useful for better query information processing and retrieval. It also illustrates Filtering RDF using Jena ARQ on Eclipse and executing a SPARQL Query on OWL Code to obtain the relationship hierarchy output.

References

[1] Berners Lee, Godel and Tuning."Thinking on the web", *Wiley*, Preface, pp xvii-xviii

[2] Groppe Sven, Groppe Jinghua, Kukulenz Dirk, Linnemann Volker, M. Clerc, M. "A SPARQL Engine for Streaming RDF Data", Proceedings of Third IEEE International Conference on Signal-Image Technologies and Internet-Based System, SITIS'07 Pages(s): 167-174

[3] http://jena.sourceforge.net/ARQ/

[4] http://jena.sourceforge.net/ARQ/Tutorial/query1.html.

[5] Jerome Euzenat,"Processing Ontology Alignment with SPARQL" 2006.

[6] Olaf Hartig and Ralf Heese, "The SPARQL Query Graph Model for Query Optimization", Humboldt-University zu Berlin.

[7] Eliase M. Navathe, "Fundamentals of DBMS" : Query Processing and Optimization".

[8] Abraham Bernstein, Christoph Keifer, Markus Stocker, "A SPARQL Optimization Approach based on Triple Pattern Selectivity Estimation" , A Techinal Report. March 2007

[9] Richard Cyganiak, Digital Media Systems Laboratory, "A Relational Algebra for SPARQL", HPL-2005-170, HP Laboratories, Bristol, 2005.

Author Biographies

Sanjay Kumar Malik has more than 14 years of experience in academics and industry in India and abroad(Dubai & USA) and is presently working as Asst Professor in University School of Information Technology of Technology, GGS Indraprastha University(Delhi Govt.), Delhi. He is MCA and acquired MTech(IT) from GGS Indraprastha University, Delhi and presently pursuing PhD from GGS IP University and published various research papers in National/International Journal/Conferences of repute and attended conferences in USA.

Dr. S.A.M. Rizvi is a Ph.D. in Computer Science & Engineering, presently working as an Associate Professor at the Department of Computer Science, Jamia Millia Islamia (Central Government University), New Delhi, INDIA having more than 26 years of experience in India and abroad. An Expert in Software Engineering, who has published numerous papers in the field of Software Engineering, MIS, Mathematical Modeling,Bioinformatics and Web based applications