Parallel querying of distributed ontologies with shared vocabulary

Index-based Concurrent SPARQL query processing

Sharjeel Aslam, Vassil Vassilev and Karim Ouazzane

1 Introduction

The World Wide Web is used and acclaimed as a data storage and sharing centre.
The conventional approaches for searching on World Wide Web are, Keywords similarities and contents categorization, however due to insufficient performance by theses searches, resulted with huge unconnected or unwanted data as well.
To conquered the issues of future intelligent World Wide Web, idea of Semantic Web was launched in 2001 with the new framework for World Wide Web for data retrieval. Improvement of XML, RDF, OWL and development of ontology development tool Jena helped to initiate the reality of Semantic Web applications. Described semantic web technologies raised more questions regarding retrieval data on Semantic Web and need for new search engine on distribute ontologies
Author is going to propose a system to fetch and process distributed SPARQL query. Results of SPARQL queries come in RDF graph category. If we investigate start of RDF query language, then we reach to this conclusion that all languages for the most part have the same features as SQL. These languages are SEROL, RDQL and RQL and so on.
Proposed distributed query mechanism will access distributed RDF store to fetch required information. The purpose for this system is not just producing single query results but fetching meaningful and linked information from diverse sources. To accomplish this our proposed framework will convert main SPARQL query into sub queries and sending sub queries to required repository to get required data

2 Literature Review

One of the unique quality of current technology is that ontologies are treated as monolithic by inference engines and software's of ontology administration. Whenever requirement comes to build layers between ontologies under distributed heterogeneous environment then common practice is to develop global unified ontology and applying reasoning on it. When the requirement comes to reapply existing facts of ontology at some stage in creation of new ontology, precisely similar method is used. Replication in newly developed ontology is the compulsory part for reused ontology and later on additional reasoning is applied on composed ontology. [1]

Now we take a look on issues on current semantic search engines in a summarised way as follows:

Few intelligent semantic search engine do not perform very well when need to improve precision and low recall. *Ding's* introduced semantic flash search engine, it is exposed that search engine's resources are based on top-50 retrieved results from the Google, cannot categorised as semantic search engine, option can be there for low precision and high [2]
Inside an intelligent semantic search engine, recognition of user's purpose of search plays vital part. For example, call for term's analysis system was proposed by *chiung-Hon leon lee* to identify the purpose of search by the user. Motivational factor behind the reason was to give more flexibility to user [3].
Few search engines presented the option for the term search. If user entered the word which can have multip meanings then results can be the alternatives list of words by search engine [4].
If user have a distinctive domain knowledge and query does not contain all required components then problem can be arise among non-phrased wrong queries [5]

Single word entered by the user can have many similar or associated meanings

Numerous semantic descriptions can be utilised to symbolised the similar proposed meaning. The above described problems may obstruct the Semantic Web applications when they have to conclude the definite meaning of assured textual resources in the context of ontologies [6].

In general, Inference techniques are incompetent on linked ontologies due to its massive size.
The building of reasoning procedures while permitting to maintain the ontologies isolated.

Details on the WWW may not be trustworthy because of the evolving environment and the absence of upgraded insights. Interaction among predicates can be the reason of intermediate output which cannot be as real as required. [7]

After executing the query, environment cannot be remain static. On the WWW information may not change quickly but rather in different situations like information streams, queries may take long time for execution which can effect information attributes and running cost as well. It can effect server performance as well due to workload. Described scenario make it compulsory to adapt strategies for changing environment.
Query optimisers ordinarily change to a heuristic methodology when queries turn out to be excessively complicated using dynamic tools. [8]
Request required various information from different sources then execution model after using optimizing techniques does not full fill requirements. The principle issue in this circumstance is that cardinality gauges change too rapidly for the streamlining [9]. Additionally, Schedule the early results and planning for executions turns out to be extremely perplexing

3 Overview of the Distributed RDF Algebra

As I have proposed a system to fetch and process distributed SPARQL query. As we realize that SPARQL is suggested by W3C for getting to RDF components and it has all the elements which are essential for querying RDF data sets. [10] Results of SPARQL queries come in RDF graph category. If we investigate start of RDF query language, then we reach to this conclusion that all languages for the most part have the same features as SQL. These languages are SEROL, RDQL and RQL and so on.
My proposed distributed query mechanism has following principle stages

1. Accessing distributed RDF stores
2. Creating new RDF data after fetching and linking information from diverse sources.

The purpose for this system is not just producing single query results but fetching meaningful and linked information from diverse sources. To accomplish this our proposed framework will examine single main query into small chunks and afterward get information accordingly.
Our strategy utilizes the fundamental pattern of RDF information. RDF utilizes <Subject, Object, Predicate> tuple model as it portrays that Subject S has property P which holds O value. Subject and Predicate are depicted as URIs and Object goes under URIs classification or can be a literal. This straightforward and simple to utilize information expression strategy accomplish query handling optimization of RDF store.
The accompanying area is going to portray design of our proposed technique. This segment will highlight and depict the significance of query analysis and retrieval of RDF in distributed atmosphere which is the objective of our proposed framework
The principle segment of our SPARQL query structure is a centralised storage strategy where RDF triples are stored after fetching required information and centralised processing system helps to link and communicate with distributed sources.

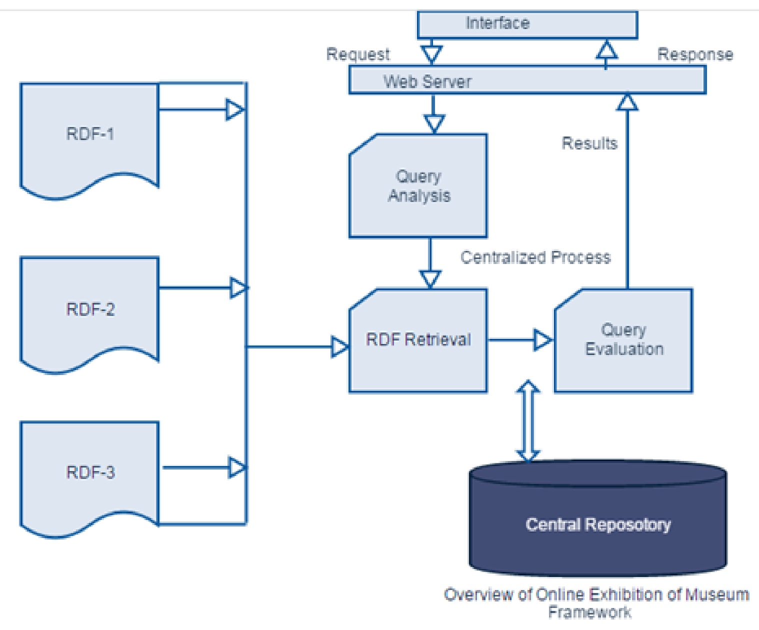


Figure 1 Framework

Fundamental part of centralized process is to investigating or analysing query in a small chunks and fetching information from remote areas. Toward the end query will be assessed against central repository.

As we will create client interface for taking input as string which will processed on centralized method via HTTP. We are going to utilize information document which will hold all data which are important to make association with sesame or jena and RDF stores. All procedure will have performed by centralized process.

3.1 Indexing

The general methodology of our procedure is NOT simply concentrating on results from each remote repository however formulating and analysing final output after fetching all RFD data. Using this methodology joins both dispersed data and new RDF store.

Presently i am going to enhance clarification about centralized procedure that how it functions. To access remote RDF initially need to build up secure association then pre-handling query stage is done which is a technique for linguistically analysis of query and afterward isolate into little chunks. This analysis give us access to RDF statements which gives triple pattern and it stores in a local and centralized RDF repository. Reasoning can be apply to this fetched RDF data. Central repository holds RDF data temporarily and it will be over right when new requested query analysis begins. Query analysis only begins when all RDF triples are fetched from all distributed sources and final output are delivered after analysing central repository.

3.2 Operators

This document proposed the formal specification/model for operations to access distributed ontologies and an algorithm to divide main SPARQL query into sub queries to fetch data from multiple repositories. I am proposing different operations, select, project, union, generalization and specialization

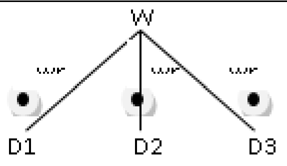
$\boldsymbol{\pi}$ sign is used for project which takes S O and P as input and source is schema

$$\boldsymbol{\pi}_{\begin{bmatrix} S? \\ O? \end{bmatrix}}(\boldsymbol{source})$$

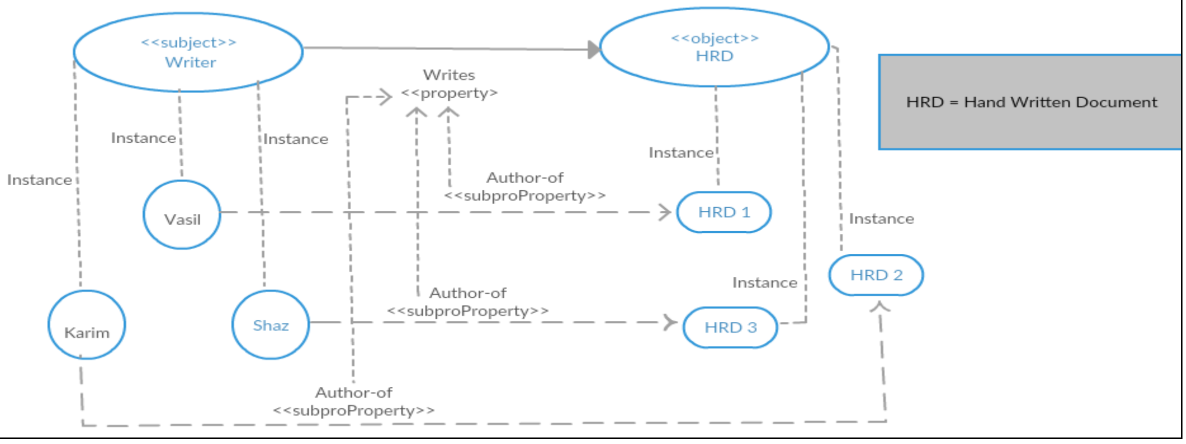
As we know that triplet has three elements, *Subject, Predicate and Object.* Project operator, $\boldsymbol{\pi}$,operator will be used to extract information about subject and object from schema and source replace with the schema name.

Example: Exhibit a list of resources from Museum about writer and handwritten documents

$$\boldsymbol{\pi}_{\begin{bmatrix} ?\boldsymbol{writer} \leftarrow ?\boldsymbol{S} \\ ?\boldsymbol{HD{-}Doc} \leftarrow ?\boldsymbol{O} \end{bmatrix}}(\boldsymbol{Schema})$$



W = Writer
wr = writes
D1 = Document 1
D2 = Document 2
D3 = Document 3



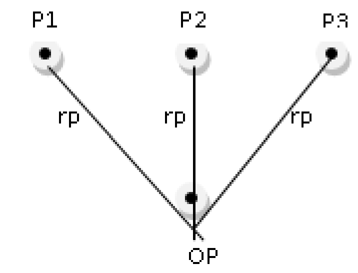
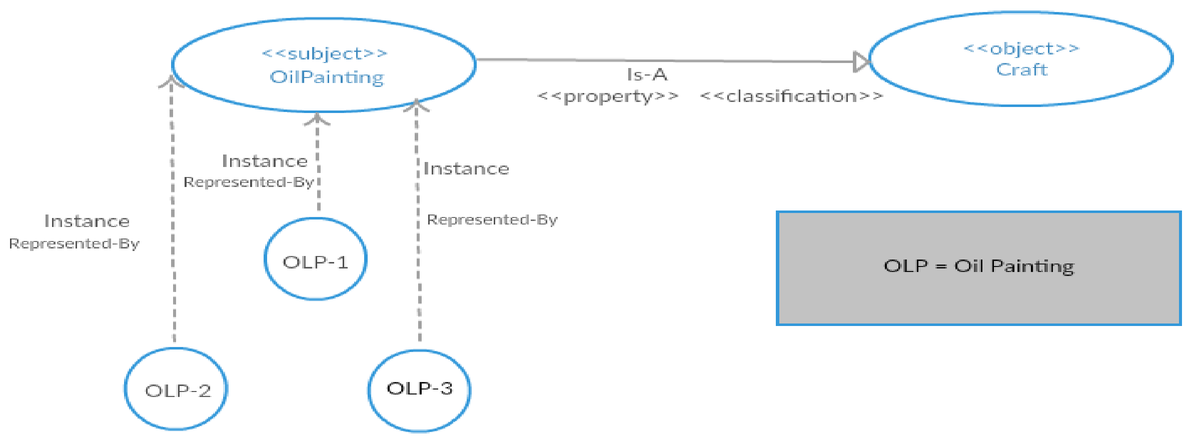
1.1 Select:

1.1.1 Syntax:

$$\boldsymbol{\sigma}_{[\boldsymbol{logic}]}(\boldsymbol{source})$$

1.1.2 Explanation:

$\boldsymbol{\sigma}$ sign is used for select and bring all required sources or nodes which meets the condition. Arithmetic ,Comparison or Boolean operators can be utilized along with constants or strings inside the "$\boldsymbol{logic}$" . and source replace with the schema name.

Example: Exhibit all paintings of OilPainting from Schema	
$\sigma_{painting=oilpainting}\ (Schema)$	
	**P1** = Painting 1 **P2** = Painting 2 **P3** = Painting 3 **OP** = Oil Painting **rp** = Represented-By
 <<subject>> OilPainting Is-A <<property>> <<classification>> <<object>> Craft Instance Represented-By Instance Represented-By Instance Represented-By OLP-1 OLP-2 OLP-3 OLP = Oil Painting	

1.2 Join

1.2.1 Syntax:

$$\pi_{[?X\ ,?Y]}\boldsymbol{\sigma}_{[\boldsymbol{condition}]}(\boldsymbol{source})$$

$$\bowtie$$

$$\pi_{[?X\ ,?Z]}\boldsymbol{\sigma}_{[\boldsymbol{condition}]}(\boldsymbol{source})$$

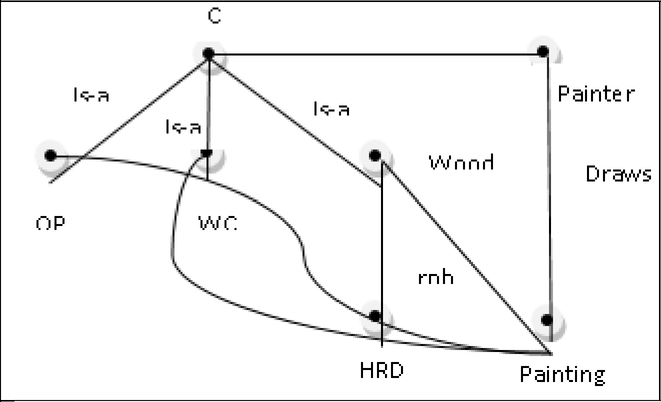
1.2.2 Syntax Explanation:

$\bowtie$ sign is used for a join. It combines triplets from a single or multiple sources according to requested query. We are using both operators, Project and Select, in our syntax. Project operator, π, takes two parameters, ? X and ?Y . As we know that triplet has three elements, *Subject, Predicate and Object. So ?X represent subject and ? Y represent object. Select operator*, $\boldsymbol{\sigma}$**,** will be used for a condition and Arithmetic ,Comparison or Boolean operators can be utilized along with constants or strings inside the "$\boldsymbol{condition}$" . and source replace with the schema name.

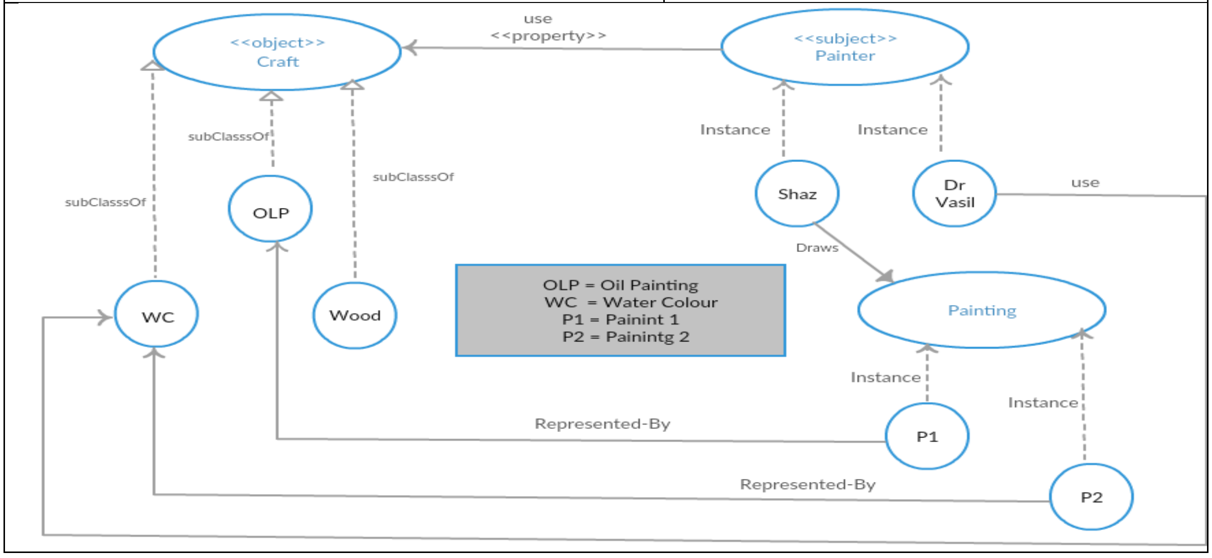
Following example will explain that how join operator works

Join Case 1: Exhibit all paintings of all painters from schemas where used crafts is watercolour

$$\pi_{[?painter\ ,?painting,?craft]}\boldsymbol{\sigma}_{[\boldsymbol{craft='watercolour'}]}(\boldsymbol{Schema\ A})$$
$$\bowtie$$
$$\pi_{[?painter,?painting\ ,?craft]}\boldsymbol{\sigma}_{[\boldsymbol{B.craft=A.craft}]}(\boldsymbol{Schema\ B})$$



C = Craft
OP = Oil Painting
WC = Water Colour
Wood = Wood
HRD =Hand written Documents
rpb = Represented By



1.3 Generalization:

1.3.1 Explanation:

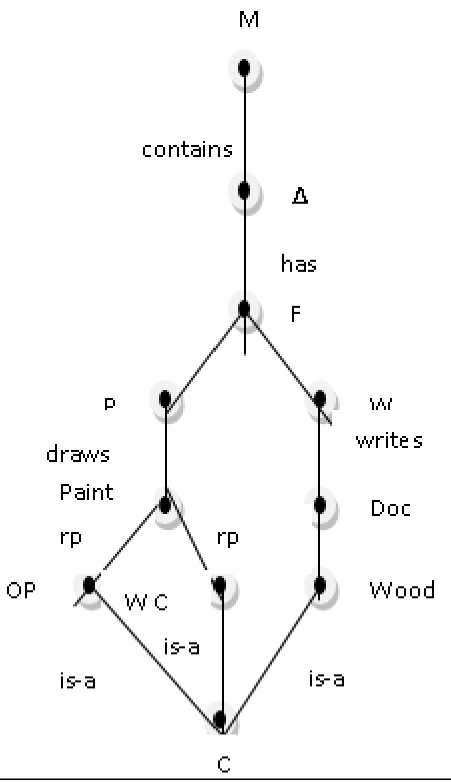
Generalization is the process of extracting common characteristics from one or more classes and combining them into a generalized superclass It is used to get hierarchies of classes and subclasses in up level, up to defined level. In our case we are going up to 1 level.

$$\boldsymbol{Gen}_{(\boldsymbol{rdfs}:\boldsymbol{class}(?\boldsymbol{class}),\boldsymbol{n-level})}(\boldsymbol{Source})$$

1.3.2 Explanation:

As we know that triplet has three elements, *Subject, Predicate and Object. Subject and object always represents classes or subclasses.* Generalization operator , ***Gen*** ,operator will be used to extract parent class up to 1 level of mentioned class (**?class**). Source will be replaced with the schema

Example 1**:** Exhibit all the hierarchies of painting at level 1

$\boldsymbol{Gen}_{(rdfs:class(?painting)1_,)(Schema)}$	
 M contains A has F P W draws writes Paint Doc rp rp OP W C Wood is-a is-a is-a C	**M** = Museum **A** =Artefact **F** = Founder **P** = Painter **W** = Writer **Paint** = Painting **Doc** = Documents **OP** = Oil Painting **WC** = Water Colour **Wood** = Wood **C** = Craft

Specialization:

Specialization is the reverse process of Genera___ation, means, creating new sub classes from an existing class. In our case we are going bottom up to 1 level

<u>Abstract from Museum schema</u>

1.3.3 Syntax:

$$\boldsymbol{Spec}_{(rdfs:class(?class)_,n-level)(Schema)}$$

As we know that triplet has three elements, *Subject, Predicate and Object. Subject and object always represents classes or subclasses.* Specialization operator , ***Spec***,operator will be used to extract child classes up to 1 level of mentioned class (**?class**). Source will be replaced with the schema

Example 1**:** Exhibit all the hierarchies of craft down to 1 level (bottom)
$\boldsymbol{Spec}_{(rdfs:class(?craft)1_,)(Schema)}$

IV Distributed SPARQL queries

The general methodology of our procedure is NOT simply concentrating on results from each remote repository however formulating and analysing final output after fetching all RFD data. Using this methodology joins both dispersed data and new RDF store.
Presently I am going to enhance clarification about centralized procedure that how it functions. To access remote RDF initially need to build up secure association then pre-handling query stage is done which is a technique for linguistically analysis of query and afterward isolate into little chunks. This analysis give us access to RDF statements which gives triple pattern and it stores in a local and centralized RDF repository. Reasoning can be apply to this fetched RDF data. Central repository holds RDF data temporarily and it will be over right when new

requested query analysis begins. Query analysis only begins when all RDF triples are fetched from all distributed sources and final output are delivered after analysing central repository.

Algorithm 1 . Translating SPARQL query into Algebraic expression

```
    Step1.   Initialise String for SPARQL query
    Step2.   Initialise list of models
    Step3.   Create Function transformToAlgebricForm which receive queryString and model
       Step4.    Create Query of given sparql query string using create method of
QueryFactory.
       Step5:   Create the pattern element of created Query
        Step6.   Create Op object to compile the query
        Step7. Optimize the Algebra expression
        Step8.  initialize variable varMap as HashMap and allocate memory to Vars  and put
                    those into varMap
         Step9.  Create object of NodeTransform with varMap
         Step10.  Call transform method to get query into relational algebraic form
```

Algorithm 2 . Converting main SPARQL query into sub queries

```
Step1.   Create function generateSubQry which receive Linked Hash Map of triplePath and set
of Strings containing required model names
Step2. Declare variable parentModels as Set of Model and assign keySet of MaodelMap
Step3. Declare variable modelTripleMap with key Model and value as LinkedHashSet of
TriplePath
Step4.  Declare variable triplesForModel as LinkedHashSet<TriplePath>
Step5      Begin For loop
                  get the key of entry into tripleName
                  get the value of entry into set of String
                  if modelSet contains modelname
                             Add triplename to triplesForModel
                end if
Step6.  Save the model and triplesForModel to map modelTripleMap
Step7.     End of for
```

V Execution of SPARQL queries in distributed ontologies

Algorithm 3 . Sending SPARQL subquery to fetch required results

```
Step1.   Create function runqueryonModel which takes modelTripleMap and modelcollection as
input
Step2     Declare parentmodel
Step3     Declare variable Map<String, String> subQryDetails
Step4     Begin loop    // for each model existingModel from parentModel
Step5      Get model name of existingModel and prefix of ExistingModel
Step6      Execute the query using queryExecution engine to receive the resultset of
           executed query
Step7      if ResultSet has next element
                  add modelname and query to subQryDetails
                  split the modelname with "." and store it into array fname
                  create object of file with "subquery" appended to fname
Step8      End if
Step9     End Loop
Step10   End function
```

Algorithm 4 . Combining results

```
Step 1 Create function runQueryonModels(List<Model> modelCollection, String queryFinal)
Step 2 get substring of query with index of select and last index of }
Step 3 Declare  Function ReadableIndex.createReadableIndex(FileFilter)
Step 4 Declare variable Map<String, String>subQryDetails

Map<String, String> subQryDetails = new HashMap<>();
Step 5 Begin For loop -for each model existingModel from parentModel
Step 5.1. :      get model name of existingModel
Step 5.2. :      get the prefix of ExistingModel
Step 5.3. :      Execute the query using queryExecution engine
Step 5.4. :       get the resultset of executed query
Step 5.5. :        if ResultSet has next element
step 5.5.1. :              add modelname and query to subQryDetails and return it
step 5.5.2. :              split the modelname with "." and store it into array fname
step 5.5.3. :              create object of file with "subquery" appended to fname
step 5.5.4. :              create fileoutputstream of above mentioned file
step 5.5.5. :             write above result to mentioned file using ResultSetFormatter

Step 5.6:           end if
Step 6          . Step close fileoutputstream and queryEngine.
Step 7. End loop.
Step 8. get  Map<String, String>subQryDetails i.e list of subqueries
Step 9.  combine subqueries with string append operation

Step 10.get the list of models
Step 11 iterate over each model and execute appended query using queryEngine
Step 12 create object of fileWriter and write query results to csv file.
```

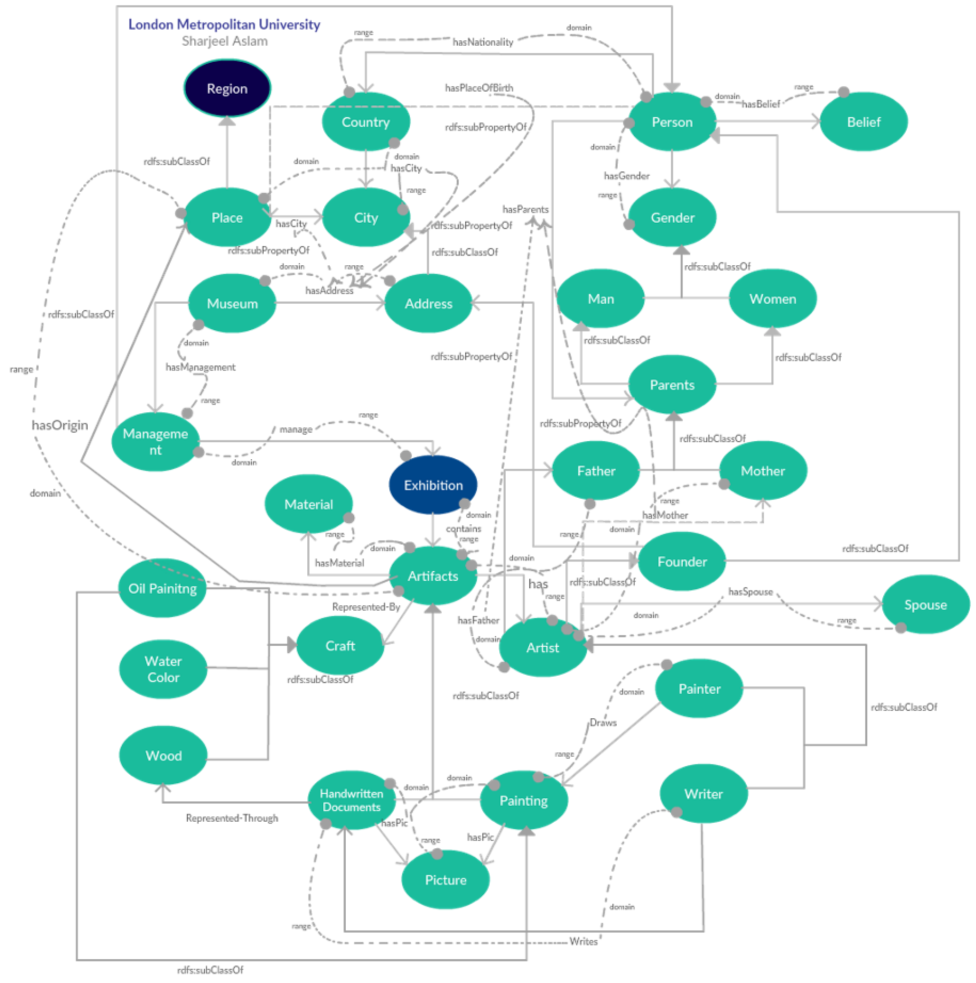
VI Conclusion and future development

The conventional approaches for searching on World Wide Web are, Keywords similarities and contents categorization, however due to insufficient performance by theses searches, resulted with huge unconnected or unwanted data as well. We have proposed index mechanisms which store triplet's information from all participated RDF files into one single repositories. We successfully proposed and implemented algorithms using Java To transform main SPARQL query into algebraic expression, converting main SPARQL queries into sub queries and then sending each sub query into required repository to get data.

References

[1] http://subs.emis.de/LNI/Proceedings/Proceedings29/GI-Proceedings.29-9.pdf.
[2] D. Ding, J. Yang, Q. Li, L. Wang, and W. Liu, "Towards a flash search engine based on expressive semantics," in Proceedings of WWW Alt.'04 New York, 2004, pp. 472-473.
[3] Chiung-Hon Leon Lee, Alan Liu, "Toward Intention Aware Semantic Web Service Systems," scc, vol. 1, pp.69-76, 2005 IEEE International Conference on Services Computing (SCC'05) Vol-1, 2005.
[4] D. Tumer, M. A. Shah, and Y. Bitirim, 'An Empirical Evaluation on Semantic Search Performance of Keyword-Based and Semantic Search Engines: Google, Yahoo, Msn and Hakia', in *Fourth International Conference on Internet Monitoring and Protection, 2009. ICIMP '09*, 2009, pp. 51 –55.
[5] Z. Gefu and H. Zhao-hui, 'Design of a Semantic Search Engine System for Apparel', in *2010 International Conference on E-Business and E-Government (ICEE)*, 2010, pp. 1414 –1417.
[6] Z. ZhiHao, H. JiPing, D. Ting, and W. Yu, 'Semantic Web Service Similarity Ranking Proposal Based on Semantic Space Vector Model', in *2012 Second International Conference on Intelligent System Design and Engineering Application (ISDEA)*, 2012, pp. 917 –920.
[7] H. S. Pinto, A. G´omez-P´erez, and J. P. Martins. Some issues on ontology integration. In *Proceedings of the Workshop on Ontologies and Problem Solving Methods (IJCAI-99)*,1999.
[8] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2006.
[9] B. Omelayenko. RDFT: A Mapping Meta-Ontology for Business Integration. In *Proceedings of the Workshop on Knowledge Transformation for the Semantic for the Semantic Web at the 15th European Conference on Artificial Intelligence (KTSW2002)*, pages 77– 84, Lyon, France, 23 July 2002.
[10] D. Tumer, M. A. Shah, and Y. Bitirim, 'An Empirical Evaluation on Semantic Search Performance of Keyword-Based and Semantic Search Engines: Google, Yahoo, Msn and Hakia', in *Fourth International Conference on Internet Monitoring and Protection, 2009. ICIMP '09*, 2009, pp. 51 –55.
[11] Z. Gefu and H. Zhao-hui, 'Design of a Semantic Search Engine System for Apparel', in *2010 International Conference on E-Business and E-Government (ICEE)*, 2010, pp. 1414 –1417.
[12] Z. ZhiHao, H. JiPing, D. Ting, and W. Yu, 'Semantic Web Service Similarity Ranking Proposal Based on Semantic Space Vector Model', in *2012 Second International Conference on Intelligent System Design and Engineering Application (ISDEA)*, 2012, pp. 917 –920.
[13] H. S. Pinto, A. G´omez-P´erez, and J. P. Martins. Some issues on ontology integration. In *Proceedings of the Workshop on Ontologies and Problem Solving Methods (IJCAI-99)*,1999.
[14] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2006.
[15] B. Omelayenko. RDFT: A Mapping Meta-Ontology for Business Integration. In *Proceedings of the Workshop on Knowledge Transformation for the Semantic for the Semantic Web at the 15th European Conference on Artificial Intelligence (KTSW2002)*, pages 77–84, Lyon, France, 23 July 2002.

Appendix A: RDF Schema of the museum ontology


London Metropolitan University
Sharjeel Aslam
Region
Country
Person
Belief
Place
City
Gender
Museum
Address
Man
Women
Parents
Manageme nt
Exhibition
Father
Mother
Material
Artifacts
Founder
Oil Painitng
Craft
Artist
Spouse
Water Color
Painter
Wood
Handwritten Documents
Painting
Writer
Picture
hasNationality
hasPlaceOfBirth
rdfs:subPropertyOf
hasBelief
hasGender
hasCity
hasParents
hasAddress
hasManagement
hasOrigin
manage
contains
hasMaterial
Represented-By
hasMother
hasFather
has
hasSpouse
Draws
hasPic
Represented-Through
Writes
rdfs:subClassOf
domain
range