Querying RDF and OWL Data Source using SPARQL

Naveen Kumar
Dept. of Computer Science and Engineering
AIACT&R, GGSIPU
New Delhi, India
naveen79599@gmail.com

Suresh Kumar
Dept .of Computer Science and Engineering
AIACT&R, GGSIPU
New Delhi, India
sureshpoonia@aiactr.ac.in

***Abstract*— The Semantic web is the extension of WWW , "web of document" that provide a support for "web of data" it gives an easier way to find ,share ,reuse and combine information. The semantic web can be best known as the web of linked data that enables people to create data stores on the web, build vocabularies, and write rules for handling data. It is based on machine-readable information and builds on XML technology's capability to define customized tagging schemes and RDF's (R*esource* D*escription Framework*) flexible approach to representing data .The key challenge for many semantic web application is to access RDF and OWL data source, as a solution to this challenge, SPARQL, the w3c Recommendation for an RDF query language, supports querying of multiple RDF graphs and OWL data. In this paper we propose a framework for querying RDF data and OWL data using SPARQL. We perform querying the data using "TWINKLE" and "PROTEGE" tool and we also provide an experimental result of improving query performance by optimizing the query.**

Keywords— SPARQL , Query processing , Query Optimization, Semantic web, RDF, OWL.

I. INTRODUCTION

The Semantic Web initiative of the World-Wide Web Consortium (W3C) has been active for the last few years and has attracted interest and doubt in equal measure. The initiative was inspired by Sir Tim Berners-Lee, founder of semantic web with the goal of providing a more flexible, integrated, automatic self-adapting Web, richer and more interactive experience for users. 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 developing the Semantic Web are: Extensible Markup Language (XML) and the Resource Description Framework (RDF) [2]. RDF is an XML-based standard for describing resources that exist on the Web, intranets, and extranets. RDF builds on existing XML and URI (Uniform Resource Identifier) technologies. RDF statements are often referred to as "triples" that consist of a subject, predicate, and object, which correspond to a resource (subject) a property (predicate), and a property value (object). 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[3]. Semantic web include many of the other technology also which makes it more expressive and powerful like XML, RDF, RDFS (RDF schema), OWL (web Ontology Language) , SPARQL etc. that are used to describe the semantics and reasoning of resources/metadata which are available on the web and also identify the relationship between them . SPARQL [4] (Simple protocol and RDF query Language) is a query language used to define a data access protocol and a standard query language to be used with the RDF data. SPARQL has been developed to extract information from data encoded using RDF. From a technical point of view, RDF databases are collections of (subject, predicate, object) triples. Each triple encodes the binary relation predicate between subject and object, i.e. represents a single knowledge fact. Due to their homogeneous structure, RDF databases can be seen as labeled directed graphs, where each triple defines an edge from the subject to the object node under label predicate [5]. There are some tools available as open source where we can execute the SPARQL query and test the results for example TWINKLE 2.0 [6],Jena framework with ARQ[7] , DARQ [8]etc. SPARQL query can also be executed on ontology editor tool like protégé [9] to retrieve data from RDFS and OWL.

In this paper we propose a framework for querying RDF and OWL data using SPARQL which include a brief explanation of query processing, Optimization and Execution in Section III. We perform querying the RDF data using "TWINKLE" tool with illustration in section IV and we also provide an experimental result of improving query performance by optimizing the query in section V, then in section VI we query the owl data using "PROTEGE" tool and finally leads to conclusion in section VII.

II. RELATED WORK

Recently, some efforts arose in querying the data in the form of RDF and OWL using SPARQL in semantic web. Querying

data has been a research topic in the field of database systems and information retrieval for a long time. In [10] Oren et al. present a technique for answering queries over RDF data through an evolutionary search algorithm, using fingerprinting and Bloom filters for rapid approximate evaluation of generated solutions. Their evolutionary approach has several advantages compared to traditional database-style query answering. Jiewen Huang et al in [11] introduce a scalable RDF data management system that is up to three orders of magnitude more efficient than popular multi-node RDF data management systems. They introduce techniques for leveraging state-of-the-art single node RDF-store technology, partitioning the data across nodes in a manner that helps accelerate query processing through locality optimizations and many other techniques. Research on query optimization for SPARQL includes query rewriting [12] or basic reordering of triple patterns based on their selectivity [13]. Optimization for queries on local repositories has also focused on the use of specialized indices for RDF or efficient storage in relational databases, e.g. [14, 15]. In this paper we perform querying of RDF and OWL data with the help of TWINKLE and PROTÉGÉ tool using SPARQL, in our experimental work we also optimize the query to improve the query performance.

III. PROPOSED FRAMEWORK

We propose a framework for querying RDF and Ontological data which includes query processing, optimizing, and execution shown below in fig 1.

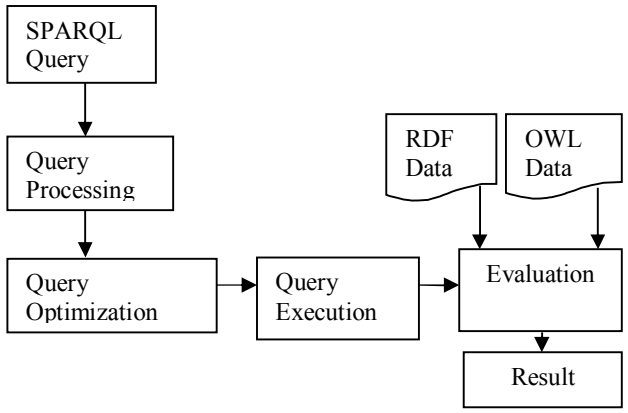

Fig 1. A framework for querying RDF and OWL data.

During the first phase of this framework which is query processing, the query has been parsed, scanned and validated. The scanner scans or identify the language token like keywords, attribute name and the relation , The SPARQL query is firstly parsed by the parser for any syntax error in which the keywords of SPARQL are identified and verifies the SPARQL query order e.g. SELECT, WHERE, FILTER , OPTION is in order or not. Then Validation check is performed to check whether the attribute and relation name is valid or not. The SPARQL query processor then transform the query into its equivalent query tree. 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 [16]. In second phase during query optimization, the task of the query optimizer is to build a feasible and cost-effective query execution plan considering limitations on the access patterns. To build the plan we use logical and physical query optimization [17]. The major focus of this phase is to reordering or rewriting the SPARQL query using some approach as selectivity estimation, which demonstrate how triples pattern reordering according to their selectivity affects the query execution performance [18].

After performing query optimization SPARQL query engine generate the query execution plan (QEP) to execute the optimized execution for RDF data .The query engine take RDF or Owl file which is passed with SPARQL query for the evaluation step. And finally it returns the result. Some tools also provide a mechanism to store the result for example TWINKLE 2.0.

IV. SPARQL ILLUSTRATIONS

To Execute SPARQL Query, we have different tools like TWINKLE, Jena ARQ, DARQ available as open source on which SPARQL query can be executed. Among these tools we make use of TWINKLE tool for executing our SPARQL queries. In this section, We Present SPARQL query execution in which Projection, Selection and rewriting rule for optimizing the query is demonstrated on the books RDF data which have developed.

A. Applying Projection

Query 1: The below query will find the name, isbn number , author and category of the book.

Syntax:-

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

B. Applyingselection using Filter

Query 2:- The below query will find the details of the books whose price is less than or equal to 500 and comes under technical category.

Syntax:-

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

```
SELECT  ?name ?isbn ?category ?author ?price
WHERE { ?x rdf:type foaf:book .
?x foaf:name ?name .
?x foaf:isbn ?isbn .
?x foaf:category ?category.
?x foaf:author ?author .
?x foaf:price ?price .
FILTER(?price <="500" && ?category="technical")
 }
order by ?name
```

The execution output with Twinkle Tool for query 1 and 2 is shown below in Figure 2 and 3:

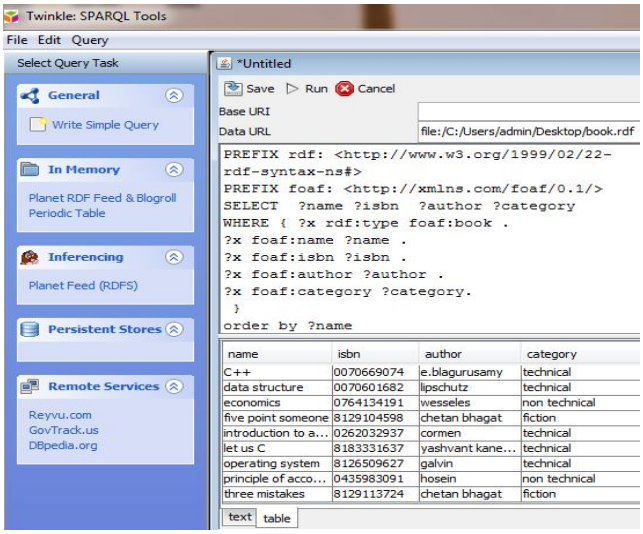

Fig 2. Execution output with Twinkle Tool

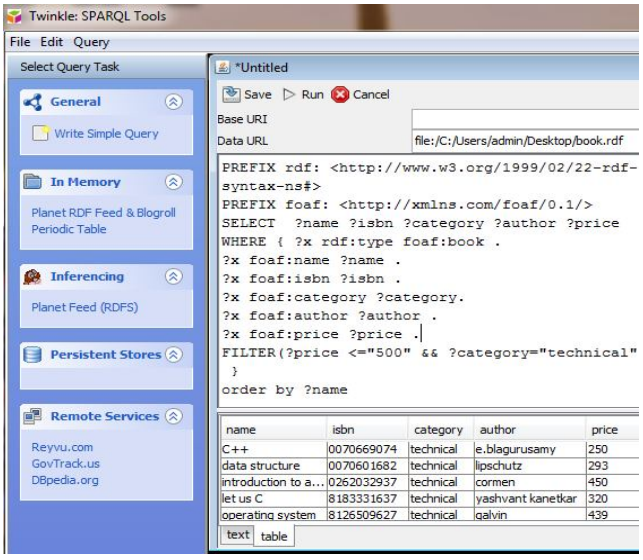

Fig 3. Execution output with Twinkle Tool

C. Applying Selection using regex function in Filter clause

Query 3: The Below query will find the details of the books written by chetan bhagat and has price less than or equal to 200.

Syntax:-

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT  ?name ?isbn ?category ?author ?price
WHERE { ?x rdf:type foaf:book .
?x foaf:name ?name .
?x foaf:isbn ?isbn .
?x foaf:category ?category.
?x foaf:author ?author .
?x foaf:price ?price .
FILTER(?price <="200" && regex(?author,"chetan"))
 }
order by ?name
```

D. Applying Rewrite Filter variable rule of Optimization

Query 4: The below query will find the details of the books whose price is less than or equal to 500 and comes under technical category.

Syntax:-

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT  ?name ?isbn ?category ?author ?price
WHERE { ?x rdf:type foaf:book .
?x foaf:name ?name .
?x foaf:isbn ?isbn .
?x foaf:category ?category.
{?x foaf:category "technical"}
?x foaf:author ?author .
?x foaf:price ?price .
FILTER(?price <="500")
 }
order by ?name
```

The execution output with Twinkle Tool for query 3 and 4 is shown in Figure 4 and 5.

In illustration of query 4 we apply the rewriting of the filter variable rule in which first query engine will execute “filter” condition to reduce the internal processing and therefore leads to reduce the time required for execution of query results in improving query performance after optimizing the query.

V. EVALUATION

In this section we evaluate the performance of the TWINKLE query engine. For our experiment we make use of books RDF data source which we have created containing 250 triples. This tool is running on following system configuration, intel core i3 processor, 3.05GHZ with 2 GB of RAM.

We execute four example queries with and without optimization on this tool. For queries without optimization we

used default execution strategies without any changes, i.e. rewriting filter variables etc. The example queries are listed in listing 1 to 4.

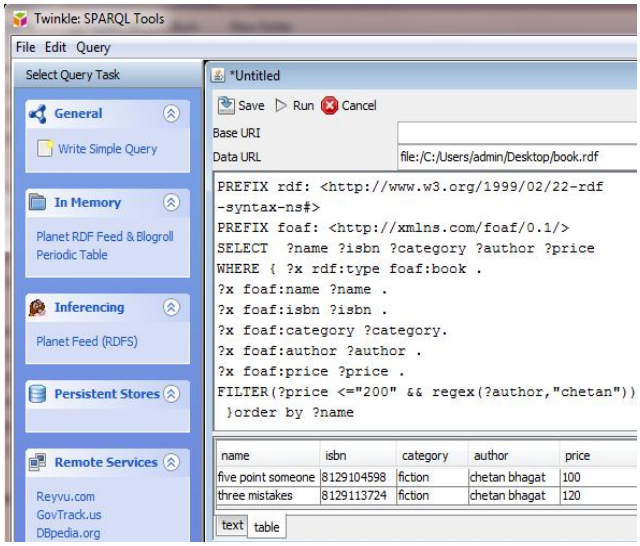

Fig 4. Execution output with Twinkle Tool

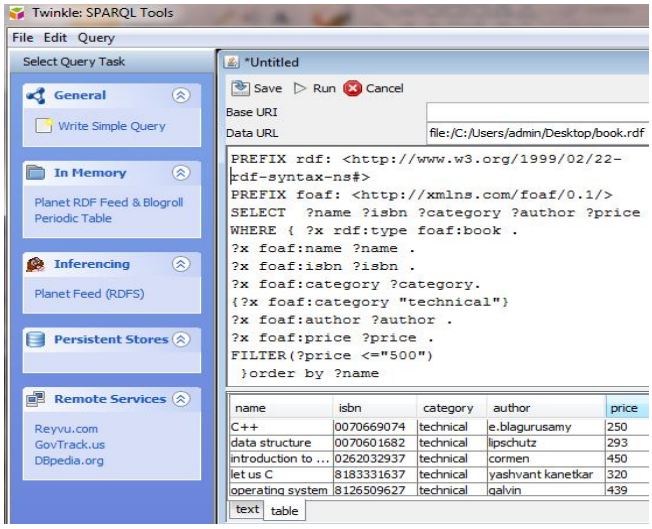

Fig 5. Execution output with Twinkle Tool

```
/*  Find the name, author, category of the book */
SELECT  ?name  ?author ?category
WHERE { ?x rdf:type foaf:book .
?x foaf:name ?name .
?x foaf:author ?author .
?x foaf:category ?category.  }
```

Listing 1. Q1

```
/* Find the name, author, category and price of the book
whose price is less than 500 and are technical */
SELECT  ?name ?author ?category ?price
WHERE { ?x rdf:type foaf:book .
?x foaf:name ?name .
?x foaf:author ?author .
?x foaf:category ?category.
?x foaf:price ?price .
FILTER(?price <"500" && ?category="technical")
}
```

Listing 2. Q2

```
/* Find the name, author, price of the book written by
chetan bhagat and has price less than 200 */
SELECT  ?name  ?author ?price
WHERE { ?x rdf:type foaf:book .
?x foaf:name ?name .
?x foaf:author ?author .
?x foaf:price ?price .
FILTER(?price <="200" &&  regex(?author,"chetan"))
}
```

Listing 3. Q3

```
/* Find the name, author, price of the book whose price is
less than 400 */
SELECT  ?name  ?author ?price
WHERE { ?x rdf:type foaf:book .
?x foaf:name ?name .
?x foaf:author ?author .
OPTIONAL {?x foaf:price ?price .  FILTER
(?price<"400"}
}
```

Listing 4. Q4

Table 1 contains the time required by each query (optimized and unoptimized) to execute, Figure 7 shows the query execution times. The experiments show that our optimizations significantly improve query performance. For query Q1 the execution times of optimized and unoptimized execution are almost same, this is due to the fact that the query plans for both cases are the same. For queries Q2 to Q4 the unoptimized queries took longer than that of optimized ones. Rewriting of the filter and optional variable rules or basic reordering the triple pattern based on their selectivity tends to reduce the time required for internal processing and thus execution time of the optimized queries is quiet reasonable . Our evaluations show that even with a very limited amount of statistical information it is possible to generate query plans that perform relatively well.

Table1: Execution time(sec) required by each query

Queries	Optimized	Unoptimized
Q1	2	2
Q2	3	4
Q3	3	5
Q4	2	5

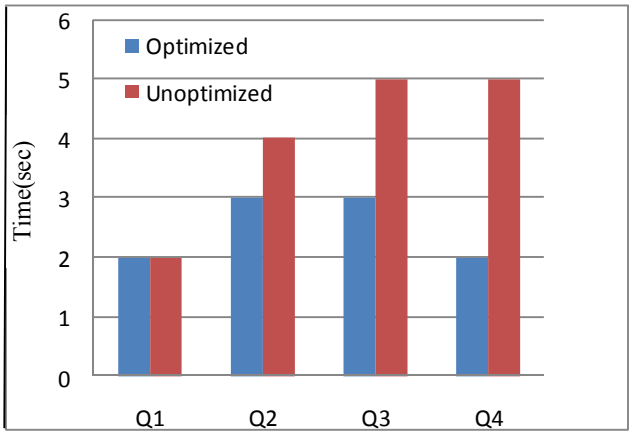

Fig 6. Query Execution times

VI. QUERYING OWL DATA IN PROTÉGÉ

We create an ontology of books domain using protégé tool. Protégé is a free, open source ontology editor and knowledge base framework. An Ontology is a "formal, explicit specification of a shared conceptualization of a domain of interest". Thus, an Ontology is the attempt to express an exhaustive conceptual scheme within a given domain, typically a hierarchical data structure containing all the relevant entities, their relations and the rules within that domain [19]. Following is the OWL Code Snippet obtained while creating the ontology of books domain. We would execute a SPARQL query on this code in protégé and obtain the result of ontology class hierarchy (subject/object).The concept of books domain ontology is shown below:

```
<?xml version="1.0"?>
<rdf:RDF
   xmlns="http://www.owl-
ontologies.com/Ontology1365941879.owl#"
   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/Ontology1365941879.owl">
  <owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="Fictional">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Category"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Availability">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="book"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Author">
    <rdfs:subClassOf rdf:resource="#book"/>
  </owl:Class>
  <owl:Class rdf:ID="Above_500">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="Price"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:ID="Technical">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Category"/>
    </rdfs:subClassOf>
  </owl:Class>……..
```

For executing a SPARQL query on books ontology we make use of protégé 3.3.1 and execute the following query shown below in SPARQL query panel of Protégé.

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

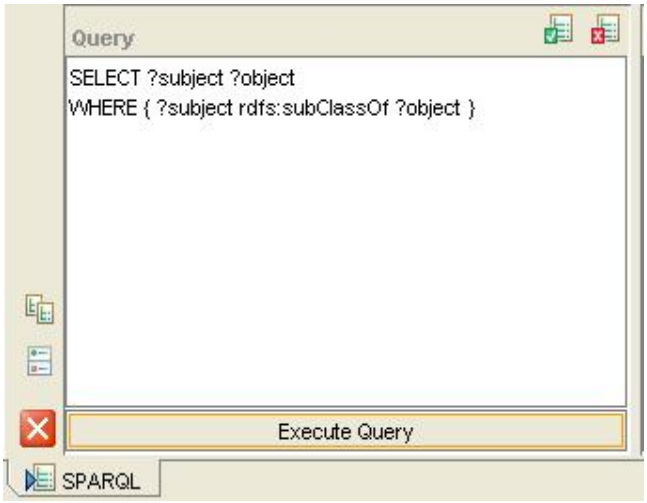

Fig 7. SPARQL query panel

When the above SPARQL Query is executed on the OWL file of books ontology created in Protégé , following result is obtained.

Results

subject	object
Category	book
Available	Availability
Price	book
Non_technical	Category
Fictional	Category
Below_500	Price
Technical	Category
Availability	book
Author	book
Above_500	Price
Not_available	Availability

Fig 8. Query output in protégé.

The above result snapshot displays the output of SPARQL query executed using protégé which displays 'Subject' and 'Object' showing the subclass and superclass relationships, For example; "Category(subject) is the subclass of book(object)".

VII. Conclusion

In this paper, we have presented a framework for querying RDF and OWL data source which contains a brief explanation of query processing, optimization and execution. We executes SPARQL query on various tool like Twinkle and protégé with various examples which may be useful for better query information processing and retrieval. Our experiments shows that the optimized query can improve query performance and allow SPARQL queries to answer in a reasonable time.

References

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

[2] The Semantic Web, by TIM BERNERS-LEE, JAMES HENDLER and ORA LASSILA ,Scientific American: Feature Article: The Semantic Web: May 2001.

[3] 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.

[4] Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. W3C Recommendation (January 2008) Retrieved June 11, 2009, from http://www.w3.org/TR/rdf-sparql-query/

[5] Schmidt, Michael, Michael Meier, and Georg Lausen. "Foundations of SPARQL query optimization." Proceedings of the 13th International Conference on Database Theory. ACM, 2010.

[6] http://www.ldodds.com/projects/twinkle/

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

[8] http://darq.sf.net/

[9] http://protege.stanford.edu/

[10] Oren, Eyal, Christophe Guéret, and Stefan Schlobach. "Anytime query answering in RDF through evolutionary algorithms." *The Semantic Web-ISWC 2008*. Springer Berlin Heidelberg, 2008. 98-113.

[11] Huang, Jiewen, Daniel J. Abadi, and Kun Ren. "Scalable SPARQL querying of large RDF graphs." *Proceedings of the VLDB Endowment* 4.11 (2011): 1123-1134.

[12] Hartig, O., Heese, R.: The sparql query graph model for query optimization. In: 4th European Semantic Web Conference (ESWC). (2007) 564-578

[13] Abraham Bernstein, Christoph Kiefer, M.S.: OptARQ: A SPARQL Optimization Approach based on Triple Pattern Selectivity Estimation. Technical Report ifi- 2007.03, Department of Informatics, University of Zurich (2007)

[14] Harth, A., Decker, S.: Optimized index structures for querying rdf from the web. In: Third Latin AmericanWeb Congress (LA-WEB),Washington, DC, USA, IEEE Computer Society (2005) 71

[15] Harris, S., Gibbins, N.: 3store: E_cient bulk rdf storage. In: PSSS - Practical and Scalable Semantic Systems. (2003)

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

[17] Quilitz, B., Leser, U. (2008). Querying distributed RDF data sources with SPARQL. Proceedings of the 5th European Semantic Web Conference (ESWC2008).

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

[19] M. Karyda, T. Balopoulos , et. Al. "An ontology for secure e-government applications", Proceedings of the First International Conference on Availability, Reliability and Security (ARES'06) 0-7695-2567-9/06 , 2006 IEEE.