Format SPARQL Query Results into HTML Report

Dr Sunitha Abburu,
Professor & Director, Dept of Computer Applications,
Adhiyamaan College of Engineering
Hosur,Tamilnadu,India

G.Suresh Babu
JRF, Dept of M.C.A
Adhiyamaan College of Engineering
Hosur, Tamilnadu, India

***Abstract*—SPARQL is one of the powerful query languages for querying semantic data. It is recognized by the W3C as a query language for RDF. As an efficient query language for RDF, it has defined several query result formats such as CSV, TSV and XML etc. These formats are not attractive, understandable and readable. The results need to be converted in an appropriate format so that user can easily understand. The above formats require additional transformations or tool support to represent the query result in user readable format. The main aim of this paper is to propose a method to build HTML report dynamically for SPARQL query results. This enables SPARQL query result display, in HTML report format easily, in an attractive understandable format without the support of any additional or external tools or transformation.**

Keywords—SPARQL query; Oracle database 11g semantic store; Jena adapter; HTML report.

I. INTRODUCTION

The goal of semantic web [1] is to extend the current web standards and technology so that machine understands the web content. The knowledge representation technology used in the semantic web is ontology. An ontology is a common, shared and formal description of important concepts in specific domain [2]. Researchers have developed several ontology languages such as RDF, RDFS, and OWL etc. There are several query languages based on the ontology data format [3]. For example XPath and XQuery query languages for XML format, RDQL [4] and SPARQL [5] for RDF format and OWL-QL for OWL format of ontologies. Among the ontology query languages, SPARQL is one of the most efficient query languages for the Semantic Web [6] and it is recommend by W3C.

The SPARQL query language for RDF has several query result forms such as CSV, TSV [7] and XML etc. These formats are not clear to the user to realize and analyze query results. There is a need to represent SPARQL query result in an attractive format so that user can easily understand the SPARQL query results. The above SPARQL query result formats require additional conversions or tool support to represent query results in user readable format. This paper proposed a method to represent SPARQL query results in more attractive and user graspable format.

The rest of the paper is organized as follows. Section II describes survey on SPARQL query result formats. Section III defines the problem statement and over view of the proposed system architecture. Section IV describes the implementation details of the proposed system. Section V shows results of the proposed method. Finally section VI concludes.

II. RELATED RESEARCH WORK

Most RDF stores use one of the common RDF query languages like RDQL [4] or SPARQL [5]. Among them SPARQL is efficient query language for semantic web and it has W3C recommendation. Basically the SPARQL language for RDF defines several query result forms such as CSV, TSV [7], and XML etc. The SPARQL binds the variables of query results into XML notation. The conversion process of SPARQL query results into XML format is described by RDF data access working group (DAWG). DAWG has described four implementations. Among them two implementations produce SPARQL result in XML format and other two implementations, consumes the query results that are in XML format. The producer implementations are Joseki [8] and AllegroGraph [9]. The producer implementations produce SPARQL query results in XML format. The two consumers are python [10] and XSL Transform (XSLT). Python parses the format into an internal graph to check for correctness. XSLT consumes SPARQL query results XML format and generates an HTML document [11].

SPARQL 1.1 query results CSV (comma separated values) and TSV (tab separated values) formats provide simple and easy to process formats for the transmission of tabular data. These formats are supported as input to many tools like spreadsheets. SPARQL 1.1 query results in JSON format [12] is designed to represent the query results in an array object. The results of a SELECT query are serilialized as an array, where each array element is one "row" of the query results. BIRT [13], an open source Eclipse-based reporting system that can be used to generate charts and other reports from input data. BIRT takes input from relational databases or spreadsheets. TopBraid Composer [14] integrates with BIRT to generate reports for SPARQL query results [15]. TopBraid Composer's Maestro Edition provides an interface between any OWL/RDF data source and BIRT.

The SPARQL query result formats CSV, TSV, JSON and XML require additional transformations or tool support to represent query result in an appropriate format. BIRT is not developed specifically for SPARQL query results. To use BIRT SPARQL query result needs to be converted into any BIRT specific input format. Our approach enables the user to build HTML document dynamically for variable binding SPARQL query results and browse the constructed HTML document automatically to view the report.

III. PROPOSED SYSTEM ARCHITECTURE

From the literature study, the conclusion is that, there is no direct method to represent SPARQL query result in user understandable format.All the existing formats require additional conversions or tool support to represent query results in user understandable format. The problem is to design and implement a method to represent results of a SPARQL SELECT query that is executed on semantic data which is stored in the oracle database 11g semantic store using Jena adapter [16].

The method should enable the user to build HTML document dynamically for variable binding SPARQL query results and browse the constructed HTML document automatically to view the query result.

The architecture describes generation of HTML page for variable binding SPARQL query results. Fig1 shows the system architecture. The user interface allows user to write SPARQL SELECT query. The query is executed on semantic data stored in the oracle database using the oracle Jena adapter. The HTML file construction section lists the variables involved in the query and extracts the variable binding values. A HTML Document is constructed with the variables and query results. Finally the constructed HTML document is displayed to view the report using any web browser like internet explorer, etc.

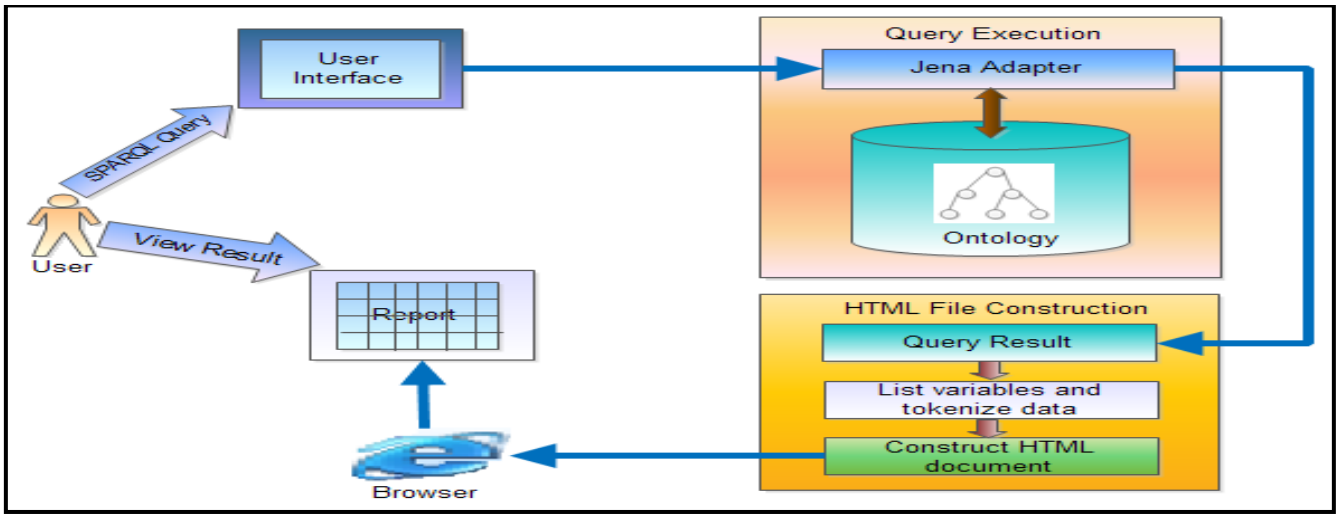

Fig. 1. Proposed System Architecture

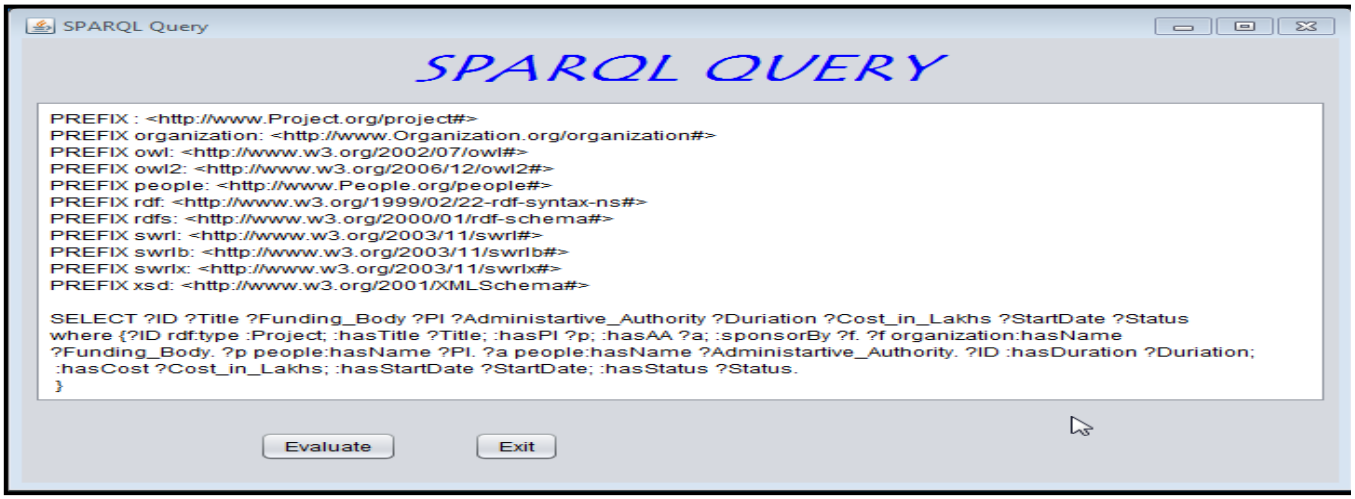

Fig.2. User interface to write SPARQL query

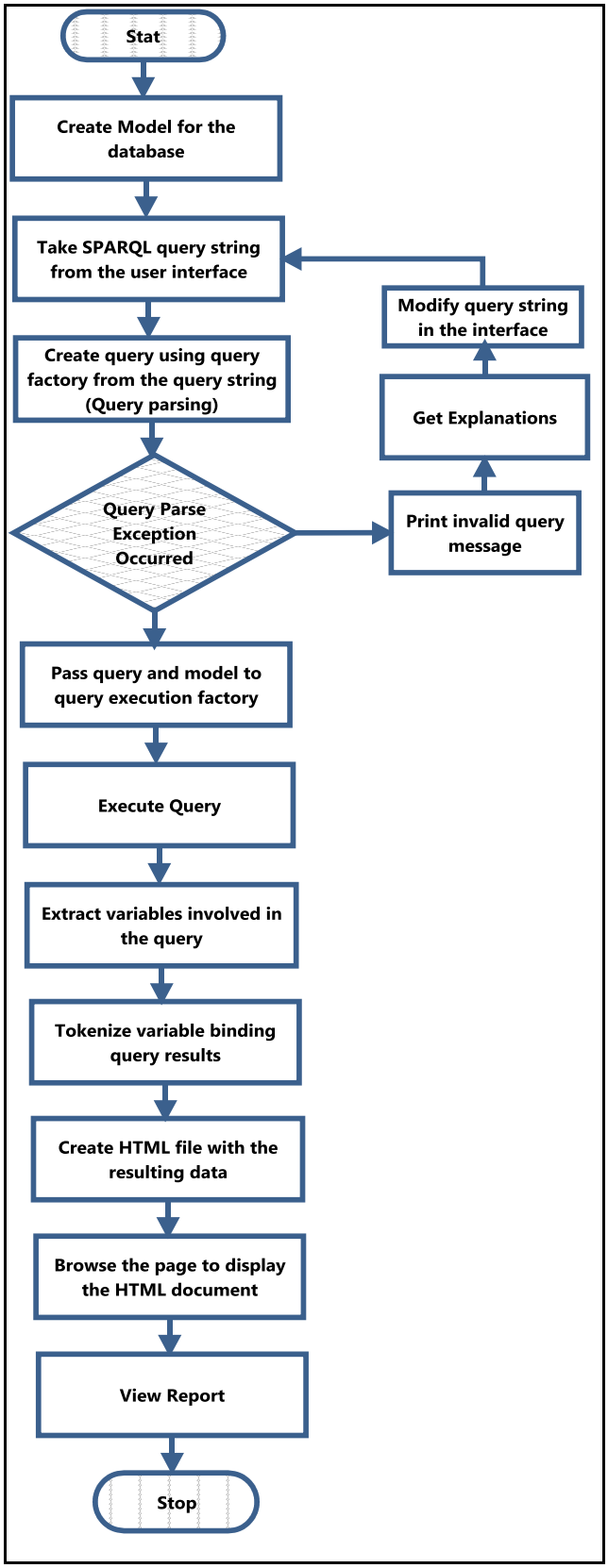

Fig.3. Execution process of SPARQL query and construction of HTML

IV. Implementation

The proposed architecture is implemented using Java NetBeans IDE and Jena API. A simple user interface is designed for writing SPARQL query using Java NetBeans IDE. Fig2 shows the interface for writing SPARQL query. Implementation of the proposed method has two phases. First phase consist query execution process using the Oracle Jena adapter and the second phase describes generation of HTML document for SPARQL query results. The entire process is described in fig3

Phase1: SPARQL query execution

Executing SPARQL query using Jena adapter has the following steps.

a) Create model for the ontology store

b) Model

```
model=ModelOracleSem.createOracleSemModel(oracle,"RDF Model");
```

c) Take SPARQL query string from the interface

d) String q=txtQuery.getText();

e) Create query using QueryFactory.

f) Query query=QueryFactory.create(q, Syntax.syntaxARQ);

g) Pass query and model to the QueryExecutionFactory and execute the query

```
QueryExecution
exec=QueryExecutionFactory.create(query, model);
```

Phase2: HTML document construction for SPARQL query results

Constructing HTML document from the variable binding SPARQL query result has the following steps.

a) List out the variables involved in the query

```
ResultSet rs=exec.execSelect();
List l=rs.getResultVars();
```

b) Extract variable binding values (it is an iterative process)

```
QuerySolution qs=rs.nextSolution();
String val=qs.get(l.get(i).toString()).toString();
```

c) Constructing HTML document with the listed variables and binding values. The following code shown in fig4 constructs HTML document dynamically.

d) Pass the constructed HTML document to a web browser to display the result

```
Desktop dt=Desktop.getDesktop();
dt.browse(new URI("result.html"));
```

V. Results

```
PrintWriter pw=new PrintWriter(new File("result.html"));
pw.print("<html><body bgcolor=\"#EAE6F5\">");
pw.print("<h2 align=center><font color=\"#FF00FF\">SPARQL
RESULT</font></h2>");
pw.print("<table border=1 align=\"center\">");
pw.print("<tr>");
for(int i=0;i<l.size();i++)
   pw.print("<th bgcolor=\"#FFA500\"><font
size=6>"+l.get(i)+"</font></th>");
pw.write("</tr>");
pw.print("<tbody bgcolor=\"#C0C0C0\">");
while(rs.hasNext())
{
   QuerySolution qs=rs.nextSolution();
    pw.print("<tr>");
    for(int i=0;i<l.size();i++)
   {
        val=qs.get(l.get(i).toString()).toString();
        pw.print("<td>"+val+"</td>");
    }
    pw.print("</tr>");
 }
 pw.print("</tbody></table>");
 pw.print("</body></html>");
 pw.close();
```

Fig.4. Code to generate HTML document for SPARQL query result

To evaluate the proposed method, OWL ontology for R&D projects is constructed using NeOn toolkit [17] and loaded into the oracle database 11g semantic store. This section describes the difference between actual SPARQL query result format and the proposed method output.

To show the difference, a sample SPARQL SELECT query is taken to print all the principal investigators involved in various research projects.

SPARQL SELECT Query: To print various R&D project details

SELECT ?ID ?Title ?Funding_Body ?PI ?Administartive_Authority ?Duriation ?Cost_in_Lakhs ?StartDate ?Status where {?ID rdf:type :Project; :hasTitle ?Title; :hasPI ?p; :hasAA ?a; :sponsorBy ?f. ?f organization:hasName ?Funding_Body. ?p people:hasName ?PI. ?a people:hasName ?Administartive_Authority. ?ID :hasDuration ?Duriation; :hasCost ?Cost_in_Lakhs; :hasStartDate ?StartDate; :hasStatus ?Status. }

The query is executed and fig5 shows actual format of SPARQL query result. The format has unnecessary data and query results are combined with variables. It is not effective, difficult to read and understand.

Fig6 shows output of the above SPARQL query using proposed method. The resulting format is effective understandable and readable.

VI. Conclusion

This paper presents a method to represent results of SPARQL query executed on semantic data stored in the oracle 11g database. This method uses the Oracle Jena adapter to execute SPARQL query. The proposed method is implemented using Java code and HTML elements to render the query results in presentable format. Compared to other approaches, this approach does not require intermediate

```
( ?Funding_Body = "DRDO" ) ( ?Cost_in_Lakhs = "4.85600014E1"^^xsd:float ) ( ?Title = "Rooting System" ) ( ?Status = "On going" ) ( ?ID =
<http://www.Project.org/project#P0002> ) ( ?StartDate = "2011-05-12T00:00:00"^^xsd:dateTime ) ( ?Administartive_Authority = "Prof. K. Lakshmi" ) ( ?PI =
"Prof. M. R. Naidu" ) ( ?Duriation = "3 Years" )
( ?Funding_Body = "DRDO" ) ( ?Cost_in_Lakhs = "4.05600014E1"^^xsd:float ) ( ?Title = "Knowledge Representation System" ) ( ?Status = "Completed" ) (
?ID = <http://www.Project.org/project#P0004> ) ( ?StartDate = "2006-05-26T00:00:00"^^xsd:dateTime ) ( ?Administartive_Authority = "Dr. Gurdeep" ) ( ?PI =
"Prof. Mohanlal" ) ( ?Duriation = "2 Years" )
( ?Funding_Body = "ISRO" ) ( ?Cost_in_Lakhs = "7.52600021E1"^^xsd:float ) ( ?Title = "Resource Management System" ) ( ?Status = "Completed" ) ( ?ID =
<http://www.Project.org/project#P0005> ) ( ?StartDate = "2007-01-24T00:00:00"^^xsd:dateTime ) ( ?Administartive_Authority = "Dr. M. Jawahar" ) ( ?PI =
"Dr. A. Usha Rani" ) ( ?Duriation = "3 Years" )
( ?Funding_Body = "MOES" ) ( ?Cost_in_Lakhs = "1.05E1"^^xsd:float ) ( ?Title = "Managing Cloud in the Internet" ) ( ?Status = "Completed" ) ( ?ID =
<http://www.Project.org/project#P0001> ) ( ?StartDate = "2009-03-24T00:00:00"^^xsd:dateTime ) ( ?Administartive_Authority = "Dr. Ajaipal" ) ( ?PI = "Dr. L.
Kanhaiya Lal" ) ( ?Duriation = "2 Years" )
( ?Funding_Body = "MOES" ) ( ?Cost_in_Lakhs = "3.03999996E1"^^xsd:float ) ( ?Title = "Knowledge Managment System" ) ( ?Status = "On going" ) ( ?ID =
<http://www.Project.org/project#P0003> ) ( ?StartDate = "2011-06-25T00:00:00"^^xsd:dateTime ) ( ?Administartive_Authority = "Prof. M K Naidu" ) ( ?PI =
"Prof. Dimpu Rani" ) ( ?Duriation = "3 Years" )
```

Fig. 5. SPARQL query result actual format

SPARQL RESULT

ID	Title	Funding_Body	PI	Administartive_Authority	Duriation	Cost_in_Lakhs	StartDate	Status
P0002	Rooting System	DRDO	Prof. M. R. Naidu	Prof. K. Lakshmi	3 Years	4.85600014E1	2011-05-12T00:00:00	On going
P0004	Knowledge Representation System	DRDO	Prof. Mohanlal	Dr. Gurdeep	2 Years	4.05600014E1	2006-05-26T00:00:00	Completed
P0005	Resource Management System	ISRO	Dr. A. Usha Rani	Dr. M. Jawahar	3 Years	7.52600021E1	2007-01-24T00:00:00	Completed
P0001	Managing Cloud in the Internet	MOES	Dr. L. Kanhaiya Lal	Dr. Ajaipal	2 Years	1.05E1	2009-03-24T00:00:00	Completed
P0003	Knowledge Managment System	MOES	Prof. Dimpu Rani	Prof. M K Naidu	3 Years	3.03999996E1	2011-06-25T00:00:00	On going

Fig. 6. Variable binding SPARQL query result using proposed method

transformations to present query results in readable format and user need not to take much effort to view the result. The proposed method is examined with R&D project OWL ontology loaded in the oracle 11g database. As shown in the results, the proposed method constructs an HTML report dynamically for SPARQL query results and displays automatically using a web browser to view the result.

ACKNOWLEDGMENT

The work presented in this paper is done as part of a sponsored project founded by government of India, Ministry of Defence, DRDO (ER&IPR), and done in the labs of Adhiyamaan College of Engineering where the author is working as a Professor& Director in the department of Master of Computer Applications. The author would like to express her sincere thanks to DRDO for providing the support.

REFERENCES

[1] N. Shadbolt, W. Hall and T. Berners-Lee, "The Semantic Web Revisited", Intelligent Systems, IEEE, 2006, vol. 21, no. 3, pp. 96–101.

[2] O. Lassila, F. van Harmelen, I. Horrocks, J. Hendler, D.L. Mcguinness, "The Semantic Web and its Languages", Intelligent Systems and their Applications, IEEE, 2000, Vol. 15, Issue 6, pp. 67-73.

[3] J. Bailey, F. Bry, T. Furche, S. Schaffert, "Semantic Web Query Languages", Encyclopedia of Database Systems, Springer, 2009, pp. 2583-2586.

[4] Seaborne A, "RDQL – A query language for RDF W3C Member Submission".Available at http://www.w3.org/Submission /2004/SUBM-RDQL-20040109.

[5] E. Prud'hommeaux and A. Seaborne, "SPARQL query language for RDF", Technical report, W3C Recommendation, 2008. Available on http://www.w3.org/TR/rdf-sparql-query/

[6] S. M. Patil and D. M. Jadhav, " Semantic Search using Ontology and RDBMS for Cricket", International Journal of Computer Applications, May 2012, Vol. 4, No.14, pp.26-31.

[7] Andy Seaborne, "SPARQL 1.1 Query Results CSV and TSV Formats", 2012, available at http://www.w3.org/TR/2012/WD-sparql11-results-csv-tsv-20120501/

[8] http://www.joseki.org/

[9] http://www.franz.com/agraph/allegrograph/

[10] http://www.franz.com/agraph/support/documentation/current/python-tutorial/python-tutorial-40.html.

[11] Dave Beckett and Jeen Broekstra, "Format SPARQL Query Results XML Format into xhtml (XSLT)", 2013, available at http://www.w3.org/TR/rdf-sparql-XMLres/

[12] Andy Seaborne, "SPARQL 1.1 Query Results JSON Format", 2013, The Apache Software Foundation, available at http://www.w3.org/TR/2013/REC-sparql11-results-json-20130321/

[13] BIRT Tutorial, version 8.2, May 2011; available at http://www.eclipse.org/birt/phoenix/tutorial/

[14] www.topbraidcomposer.org.

[15] Moritz Weiten, "OntoSTUDIO as a Ontology Engineering Environment", Semantic Knowledge Management, Springer Book, Chapter 5, 2009, pp.51-60.

[16] Chuck Murray, "Oracle Database Semantic Technologies Developer's Guide", 11g Release 2 (11.2), may 2012.

[17] NeOn toolkit: http://neon-toolkit.org/wiki/Main_Page.