Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 9, September 2014, pg.121 – 133

RESEARCH ARTICLE

A QUERY FORMULATION LANGUAGE FOR DATA WEB

Jagadish Kumar. Talagapu[1], G. Ravi[2]
C.S.E & JNTUH
Jagadish.tlgp@gmail.com, g.raviraja@gmail.com
[1]M.Tech IV semester, Dept. of CSE, Malla Reddy College of Engineering and Technology, Hyderabad
[2]Associate Professor, Dept. of CSE, Malla Reddy College of Engineering and Technology, Hyderabad

Abstract— This trend of structured data on the web is shifting the focus of the web technologies toward new paradigms of structured –data retrieval. Traditional search engine cannot serve such data as the result of a keyword-based. Query will not be precise or clean, because the query itself is still ambiguous although the underlying data are structured. To expose the massive amount of structured data on the web to its full potential, people should be able to query these data easily and effectively. We present a query formulation language(called MashQL) in order to easily query and fuse structured data on the web the main novelty of MashQL is that it allows people to navigate, explore, query and mash up data sources without prior knowledge about the schema, structure, vocabulary, or any technical details of the sources.

Index Terms— Query Formulation, Semantic Web, Data Web, RDF, SPARQL, Indexing Methods

1. Introduction

Data mining, the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most important Information in their data warehouses. Users to easily search and consume structured data are a Known challenge that receives recently a great attention from the Web and the Data web Communities. Companies, Communities, Research Labs such as Google Base, Yahoo Local, Freebase, Upcoming, Flicker, eBay, Amazon, and LinkedIn are all competing on publishing structured data in the web through APIs and slowly started to adopt metadata standards. For example, Yahoo started to support websites embedding RDF and micro formats, by better presenting them in the search results; MySpace also started to adopt RDF for profile and data portability; Google, Upcoming, Slide share, Digg, the Whitehouse, and many others started to publish their content in RDF a, a forthcoming W3C standard for embedding RDF inside web pages so that content can be better understood, searched, and filtered.

1.1 Scope of the project:

This trend of structured data on the Web (Data Web) is shifting the focus of Web technologies towards new paradigms of *structured-data retrieval*. Traditional search engines can serve such data as the results of a keyword based query will be precise or clean, because the query itself is still decided although the underlying data is structured. In this project we apply query formulation language .a user can write a keyword, based on the keyword to display the relevant information. For Example Web user wants to retrieve "Lara's articles 2007".In our Application User select type article or storybooks, here user need article so user select Article Based on article display author name, here user need Lara's article so user Select Lara's Based on author name display Lara's articles published years, here user need "Lara's articles 2007". So user select year 2007 Then user go to search finally web user get the Lara's articles 2007.

1.2 Challenge:

The huge database consists of both related and relevant information. To retrieve structured information few challenges are encountered before formulating a query, user has to know in "what way the data is structured" and "what are the labels of those data elements". This is because the end users are not expected to investigate about the schema. Instead the user every time search and filter for structured information. Users only use drop-down lists generated dynamically, as they interact with the query editor. We even do not assume that a data source should have -an offline or inline- schema. To define the query formulation power of MashQL, the Data Web is chosen as an application scenario. We also chose RDF as a data model and SPARQL as a backend query language. This is because RDF is themost primitive data model, and thus, MashQL can be similarly used for querying, e.g., databases and XML. Formulating queries should be fast and should not need any programming skills.

Example:

Web user wants to retrieve "Lara's articles 2007".In our Application. User select type article or books, or songs here user need article so user select Article Based on article display author name, here user need Lara's article so user Select Lara's Based on author name display Lara's articles published years, here user need "Lara's articles 2007". RDF data is queried using SPARQL. So user select year 2007 Then user go to search finally web user get the Lara's articles 2007. Query conditions in SPARQL are called triple-patterns (RDF represents data as a directed labeled graph. A graph is a set of triples of the form <Subject, Predicate, Object>. Subjects and Predicates must be URIs, an Object can be either a URI or a Literal.), and evaluated as pattern-filling rather than truth- evaluation if compared with SQL. This is a robust way for querying schema-free data, as changes to data do not cause queries to break; however, it poses hard query formulation challenges. Before writing a query, one has to be fully-aware of the property labels and data structures query requires one to manually investigate the data itself before querying it. This issue becomes challenging in the case of large datasets; and even more complex when querying multiple sources, as predicates have to be explicitly union-ed. It allow people to easily query and consume structured data is a known challenge in different areas.

http://example1.com: http://example2.com

```
A1 rdf:Type bibo:Article
:A1 :Title "Data Web"
:A1 :Author "Tom Lara"
:A1 :Author "Bob Hacker"
:A1 :Year 2007
:A2 rdf:Type bibo:Article
:A2 :Title "Semantic Web"
:A2 :Author "Tom Lara"
:A2 :Year 2005
```

```
:B1 :Title "Linked Data"
:B1 :Author "Lara T."
:B1 :PubYear 2008
:B1 :Publisher "Springer"
:B2 :Title "Data on the
Web"
:B2 :Author "Abiteboul S."
```

Figure: 1. SPARQL query over two RDF data sources.

SPARQL Query:

```
PREFIX S1:<http://example1.com>
PREFIX S2:<http://example2.com>
 SELECT ?ArticleTitle
  FROM <http://example1.com>
  FOM <http://example2.com>
  WHERE {{{?X S1:Title ?ArticleTitle} UNION
  {?X S2:Title ?ArticleTitle}}
  {{?X S1:Author ?X1} UNION {?X S2:Author ?X1}}
  {{?X S1:Year ?X2} UNION {?X S2:PubYear ?X2}}
  FILTER regex(?X1, "^Lara")
  FILTER (?X2 > 2007)}
```

Results:

ArticleTitle
Data Web
Linked Data

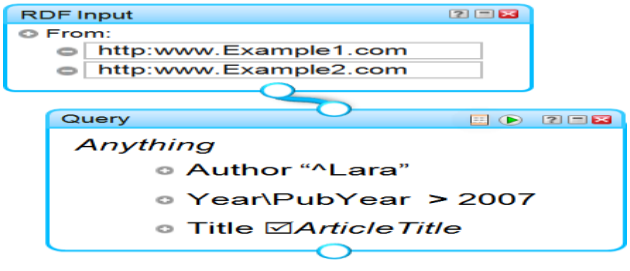


Figure: 2. the same SPARQL query Figure 1, but in Mash QL.

We propose an interactive query formulation language, called MashQL. The novelty of MashQL (compared with related work) is that it considers all of the above assumptions together. Being a language -not merely an interface and, at the same time, assuming data to be schema-free is one of the key challenges addressed in the context of MashQL design and development. Without loss of generality, this article focuses on the Data Web scenario. We regard the Web as a database, where each data source is seen as table. In this view, a data mashup becomes a query involving multiple data sources.

To illustrate the power of MashQL we chose to focus on querying RDF, which is the most primitive data model, hence, other models -as XML and relational databases - can be easily mapped into it . We give a bird's-eye view of MashQL in Figure 2 which shows the same query as in Figure 1 written in MashQL. The first module specifies the query input, the second module specifies the query body, and the output is piped into a third module (not shown here) that renders the results into HTML or XML, or as RDF input to other queries. Each MashQL query is seen as a tree; the root is called the query subject. Each branch is a restriction on a property of the subject. Branches can be expanded to allow sub trees, called query paths, which allows one to navigate through the underlying dataset and build complex queries. Formulating a query is an interactive process: First, the editor queries a given dataset (as a black-box) to find the main concepts, from which the query subject can be selected. The editor then finds the possible properties for this subject (e.g. Title, Author, Year). The user selects a property and restricts it using a function. In this way, users can navigate and query a data source without any prior knowledge about it. The symbol "_" indicates a projection, i.e., appear in the results. When querying multiple sources, two properties (or two instances) are considered the same if and only if they have the same URI. To help end-users not seeing cryptic URI,

the editor normalizes URIs by detecting different namespaces of same properties and optionally combines them together (Section6). In case of different same spaces and property labels (e.g.,S1:Year and S2:PubYear), the user can choose the union operator "\" to combine them. Although MashQL can be used, in a sense, for data integration, but this is not a goal per se. Data integration requires not only syntax, but also semantic integration, which is not supported in MashQL. MashQL allows people to spot different labels of same properties.

- Query Language: The notational system and constructs that make MashQL an expressive and yet intuitive query language, supporting all constructs of SPARQL.

- Query Formulation Algorithm: This algorithm is used by the Mash QL editor. Its novelty is that it one to navigate through and query a data graph(s) without assuming the end-user to

- Know the schema or the data to adhere to a schema.

- Graph Signature Index: Because of assumption 2 (data is schema-free), the previous algorithm has to query the whole dataset in real-time, which can be a performance bottleneck because such queries may involve many self-joins. Hence, the interactivity of Mash QL might be unacceptable. Thus, we propose a new way of indexing RDF, which we call the Graph Signature. The size of a Graph Signature is typically much smaller than the original graph, yielding fast response-time queries.

- Implementation and Evaluation: We present two implementations of MashQL: a server-side. mashup editor, and a Firefox add-on extension. We evaluate the response-time of MashQL on two large.

- Datasets: DBLP and DBPedia; and compare it with Oracle's Semantic Technology. We will show queries can be answered instantly, regardless of the data size.

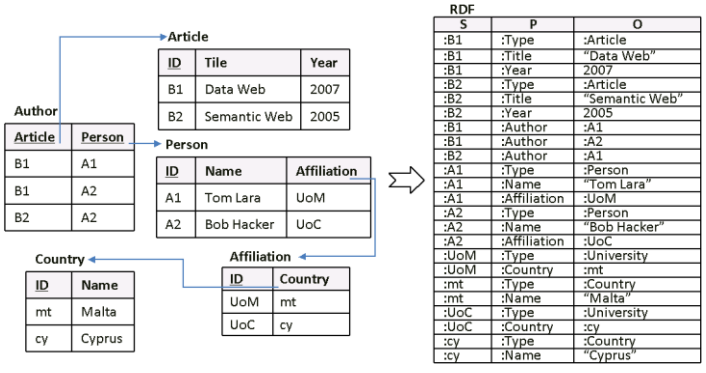
2. Related Work

There are several approaches have been proposed by the database community to query structured data sources, such as *query-by-example* and conceptual *queries*. These types of approaches are not used by casual users. This is because they still assume knowledge about the relational/conceptual schema. In the natural language processing community, it has been proposed to allow people to write *queries as natural language sentences*, and then translate these sentences into a formal language (SQL or XML Query). However, these approaches are challenged with the language ambiguity and the "free mapping" between sentences and data schemes. Several approaches and RDF Author are proposing to *represent triple patterns graphically* as ellipses connected with arrows. However, these approaches assume advanced knowledge of RDF and SPARQL. Some mash up editors (e.g., Yahoo Pipes, Popfly, sMash) allow people to write query scripts inside a module, and visualize these modules and their inputs help formulating what is inside these boxes. In XML databases, the lore query language has been proposed to allow people to query XML data graphically, and without prior knowledge about the data.

3. Definition of MashQL

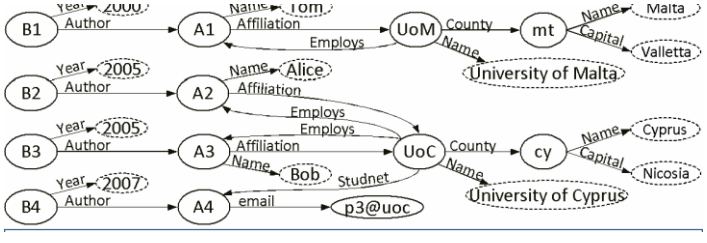
The concatenation of syntax and semantics is known as MashQL. The idea of MashQL is to allow people to query, mash up, and pipeline structured data intuitively. In the background MashQL queries are automatically translated into and executed as SPARQL queries. The novelty of MashQL is that it allows one to query (and navigate) an RDF graph without any prior knowledge about its structure or technical details; as well as it supports query pipelines as a built-in concept. RDF represents data as a directed labeled graph. A graph is a set of triples of the form <Subject, Predicate, Object>. Subjects and Predicates must be URIs, an Object can be either a URI or a Literal. The only difference with the RDF model is that we allow an identifier to be any form of a key (i.e. weaker than a URI). Allowing this, would simplify the use of MashQL for querying databases. Relational databases (or XML) can be

mapped easily to this primitive data model. The primary key of a table is seen as a subject, a column label as a predicate, and the data-entry in that column as an object. Foreign keys represent relationships between data elements across tables.



RDF provides a rich data model where the entities and relationships can be described in details. The relationships in the RDF framework are class objects, which mean that the relationship between the objects may be created arbitrarily and stored separately from the objects. This makes RDF suitable for dynamically changing, shared and distributed nature of web. Also RDF's simple data model and the ability of modeling data which differ from resource to resource, led to its increasing usage in applications unrelated to semantic web activity.

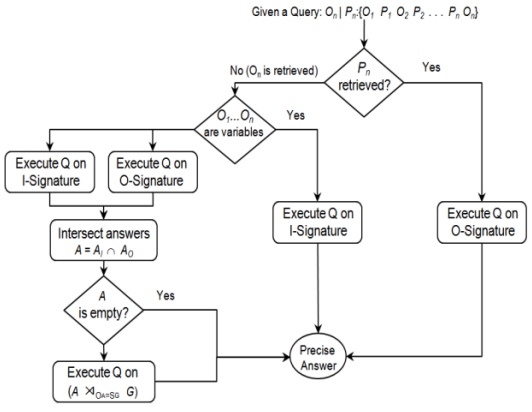
It is easy to write, flexible, and without constrains. However RDF has been criticized that the XML syntax for RDF is verbose and the triplet (subject, predicate, object expressions) notation is not expressive enough.

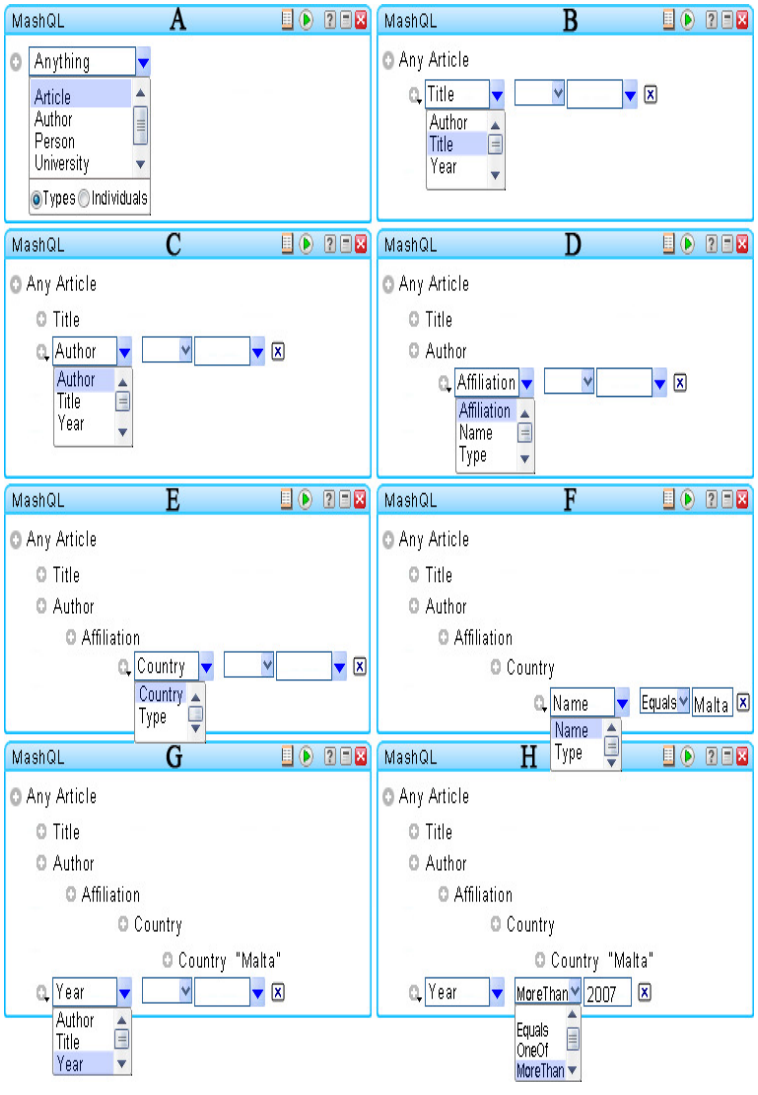


4. Implementation

There are some lists of the possible properties for this subject is dynamically generated to add a restriction on the chosen subject. The user can choose any one of the subject from the list of properties such as {Title, Author, Publisher, Year}; if the subject is a variable, the list will be the set of all properties in the dataset. Users can use any optional or unbound to filter the data. If a property is prefixed with "maybe" this property is considered optional, if it is prefixed with "without" it is considered unbound, and if it is not prefixed then it is required. Users can choose an object filter such as Equals, Contains, Doesn't contain, One Of, Between, More Than, Not, etc., or may select an object identifier from a list, which is generated from the set of the possible objects, depending on the previously chosen subject and predicate. Furthermore, users can also click on the restriction icon to expand the tree, i.e., declare a query path. The symbol can be used before subject, property, or object variables to indicate that this variable will be returned in the results. For example, the results of the above query are a one-column table that contains the list of all retrieved titles.

ARCHITECTURE:





QUERY FORMULATION ALGORITHM:-

This algorithm is used by the MashQL editor. Its novel is that it one to navigate through and query a data graph (s) without assuming the end-user to know the schema or the data to adhere to a schema.

Select the query subject:

That is, after specifying the dataset, users can select s from a dropdown list that contain, either.

I. ST: the set of the union of the subject-types in G, such as Article; or and object identifiers in the dataset.
II. SI: the union of all subject and object identifiers in the dataset.
III. A user-defined subject label. In the latter case, the subject is seen as a variable (SIV) displayed on italics a variable label anything.

Select a property:

Depending on the chosen subject (s), a list of the possibilities properties for this subject is generated. There are four possibilities:

I. If (S I ST), such as Article, the list will be the set of all properties that the instance of this subject-type have(e.g, Title, Author, Year).
II. If (S I SI), such as A1, the list will be the set of all properties that this particular instance (s) has.
III. If the subject is a variable (S I V), the list will be the set of all properties in the dataset.
IV. Users can also choose the property to be a variable by introducing their own label.

Add an object filter:

There are three of filters the user can use to restrict P: a filtering function, an object identifier, or a query path.

I. A filtering function can be selected from a list.
II. if a user want to add an object identifier as a filter, a list of the possible objects will be generated.

Example: if a user previously choose any article as a subject, as a property, the list of objects identifiers would. The following formalization specify what the list of objects identifiers may contain.

Algorithm:

Step 0: Specify the data set G in the Input module. G can be one or a merge of multiple Data graphs.

Step 1: Select the query subject S. That is, after specifying the data set, users can select S From a drop-down list.

Step 2: Select a property P. Depending on the chosen subject(s) in step 1, a list of the possible properties for this subject is generated

Step 3: Add an object filter on P. There are three types of filters the user can use to restrict P:

1) A filtering function can be selected from a list.

2) If a user wants to add an object identifier as a filter, a list of the possible objects will be generated.

3) Users can also choose to expand the property P to declare a path on it Repeat Steps 2-3 (until the user stops).

Step 4: The symbol before a variable is used to indicate that it will be returned in the Results.

Hardware and Software Requirements:-

HARDWARE REQUIREMENTS:

Processor : Pentium Dual Core 2.00GHZ

Hard disk : 40 GB

Ram : 2GB (minimum)

SOFTWARE REQUIREMENTS

Operating system: windows

Front end : Microsoft visual studio .Net 2005

Coding language: C#

Backend : SQL server 2005

Existing system:

The main problem is that this approach is fundamentally bounded with the language ambiguity multiple meanings of terms and the mapping between these terms and the element of a data schema allow people with limited IT-skills to explore and query one or multiple data sources with prior knowledge about the schema, structure, vocabulary, or any technical details of the these sources.

We do not assume that a data source should have -an offline or an inline schema. The rapid growth of structured data on the web has created a high demand for making this content more reusable and consumable.

Disadvantage:

- Everybody must have a knowledge about the process.
- Content would not display if the keyword not matching.
  - Existing algorithm is pattern matching algorithm.

Proposed system:

We present a novel query formulation algorithm, by which the complexity and the responsibility of understanding a data source are moved from the user to query editor. It allows end user to easily navigate and query an unknown data graph. We addressed the challenge of achieving interactive performance during query formulation by introducing a new approach for indexing RDF data. We present two different scenario of MashQL and evaluate our implementation on two large datasets. In future, we plan to use our approach on keyword-search.

- Proposed algorithm is query formulation algorithm.

Advantages:

- It saves the time and the user spending cost.
- It is allow the user to dynamically create a new file through the web.
- MashQL can be similarly used for querying relational databases and XML.
- MashQL can be used to query and mash up the Data Web as simple as filtering and piping web feeds.
- Easily query and fuse structured data on the web.
- Easy to users can navigate and query a data source without any prior knowledge about it.

5. GRAPH INDEXING

The main aim is querying the data web in such a way that, the data are schema free. By query formulation, at the backend queries are needed to executed on the whole data set in real time, because there is offline schema that can be used instead. In such a user-interaction setting, the response-time is an important factor that needs to be taken into consideration and which should be small, preferably within 100 ms. Achieving such a short interaction time for background queries with graph-shaped data is even more challenging, because the exploration of a graph stored in a relational table G(S,P,O) can be expensive as this table needs to be self-joined many times. A query with *n* levels involves *n-1* joins. Pre-computing and materializing all possible MashQL's background queries is not an option since the space requirements are too high; thus an efficient RDF indexing is needed. Several approaches have been proposed to index RDF, such as Oracle3, and RDF3X5. Although these approaches have shown good performance –a query with a medium complexity costs some seconds- however, this performance is unacceptable for an interactive query formulation session, especially in the case of large graphs.

Compute the Graph-Signature:

Procedure ComputeOSignature(G, S_o)

begin

S_o= (a copy of) *G.* Stable = false

Group nodes having the same property labels.

while (Stable □True) do

foreach node *A* in S_o do

for each path P_I from *A* into a node *B* do

$X = P\text{i-}1(ext[B]^G)$

if ($A \not\subseteq X$) then replace *A* by ($A \cap X$) and ($A - X$)

if there was no split then Stable=True

end

Procedure ComputeISignature(G, S_I)

begin

S_I = (a copy of) *G*. Stable = false

Group nodes having the same incoming properties

while (Stable □True) do

foreach node *A* in S_I do

foreach P_i path into *A* from a node *B* do

$X = P_i(ext[B]^G)$

if ($A \nsubseteq X$) then replace *A* by (*A* □*X*) and (*A* - *X*)

if there was no split then Stable=True

end

6. Conclusion and Future Work

We propose a query formulation language, called MashQL. We have specified four assumptions that a data web query language should have, and shown how MashQL implements all of them. The language-design and the performance complexities of MashQL are fundamentally tackled. We have designed and formally specified the syntax and the semantics of MashQL, as a language, not merely a single purpose interface. We have also specified the query formulation algorithm, by which the complexity of understanding a data source (even it is schema free) are moved to the query editor. We addressed the challenge of achieving interactive performance during query formulation by introducing a new approach for indexing RDF data. We presented two different implementation scenarios of MashQL and evaluated our implementation on two large datasets.

References:

1. http://www.scribd.com/doc/75915018/A-Query-Formulation-Language-for-the-Data-Web-Abstract-by-Coreieeeprojects
2. http://jpinfotech.blogspot.in/2012/08/a-query-formulation-language-for-data.html
3. http://dl.acm.org/citation.cfm?id=1874607
4. Jayapandian M, Jagadish H"Expressive Query Specification through Form Customization" 2008.
5. "Querying the Data Web" Jarrar M, Dikaiakos M. 2009.
6. "Data Guides: Enabling Query Formulation and Optimization Semistructured Databases" Goldman R, Widom J 2009.
7. A Data Mashup Language for the Data Web" Jarrar M, Dikaiakos M 2009.
8. "An Efficient SQL-based RDF Querying" Scheme E. Chong, S. Das, G. Eadon, and J. Srinivasan 2005.
9. "MashQL for the Data Web" E. Chong, S. Das, G. Eadon, and J. Srinivasan 2005.
10. "How Useful Are Natural Language Interfaces to the Semantic Web for Casual End-Users? "E. Kaufmann and A. Bernstein 2007.
11. "Towards a Theory of Natural Language Interfaces to Databases" A. Popescu, O. Etzioni, and H. Kautz,2003.
12. "MashQL: A Query-by-Diagram Topping SPARQL Towards Semantic Data Mashups" A. Popescu, O. Etzioni, and H. Kautz,2003.
13. "Query-by-Example: a data base" languageM. Zloof,1977.
14. M. Jarrar and M. Dikaiakos, "A Query Formulation Language for the Data Web", *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 5, may 2012.
15. M. Jarrar and M. Dikaiakos, "A Data Mashup Language for the Data Web," *Proc. WWW2009 Workshop Linked Data on the Web (LDOW)*, 2009.
16. M. Jarrar and M. Dikaiakos, "MashQL: Querying the Data Web," *IEEE Internet Computing*, vol. 14, no. 3, pp. 58-67, May 2010.
17. M. Jarrar and M. Dikaiakos, "Querying the Data Web," *Technical Report* TR-09-04, Dept. of Computer Science, Univ. of Cyprus, http://encs.birzeit.edu/klab/publications/TARMD10.pdf.htm, Nov. 2009.
18. M. Jarrar and M. Dikaiakos, "MashQL: A Query-by-Diagram Topping SPARQL Towards Semantic Data Mashups", *ONISW'08*, October 30, 2008, Napa Valley, California, USA.

Authors Biography

First Author: Jagadish Kumar. Talagapu

Roll no:12N36D5830
M.Tech in CSE from JNTUH,
Malla Reddy College of Engineering and Technology, Hyderabad

Second Author: G. Ravi

M.Tech in CSE from JNTUH,
Associate Professor, Dept. of CSE,
Malla Reddy College of Engineering and Technology, Hyderabad

