Engineering Sciences International Research Journal
ISSN (Print) : 2330 – 4338 ; Volume 1 Issue 1 ǁ February. 2013 ǁ

MashQL Editor using Query Detection Algorithm

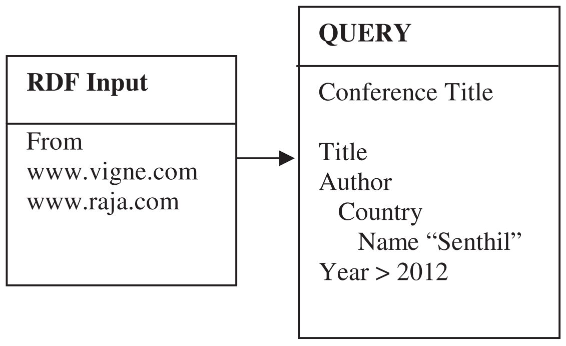
U. Vignesh[1], P. Senthilraja[2]

Abstract: We present a query detection algorithm on a existing query formulation language (mashQL) to identify data easily in a database and to produce the structured data on screen as a result. This algorithm on mashQL gives an advantage that the user, who gives a query, unknown of schema structure. To illustrate this query detection power of mashQL without any constraints, such as generality loss, etc. we choose the local database scenario. In background, the mashQL queries are automatically translated into SPARQL queries and they are executed as SPARQL queries, which can be decomposed into highly performing fragments that detects the data on which way a data is partitioned into cluster in a RDF data management system. Which, can also acts as a server. MashQL is also called a query-by-diagram language. These concepts can be applied over a library information system, musical player etc., which includes large scale of aspects controlled on to it. By this application, we can provide very effectively accurate and efficient interaction between system and user. MashQL editor has the search text box top of screen and provides search query results. MashQL editor allows user to search and get data called as challenge. By this query detection algorithm, complexity and responsibility of understanding data source are moved from user to query editor. Due to local database scenario, we structure the required amount of data's to the database.

Keywords: Query detection, mashQL editor, challenge, search box.

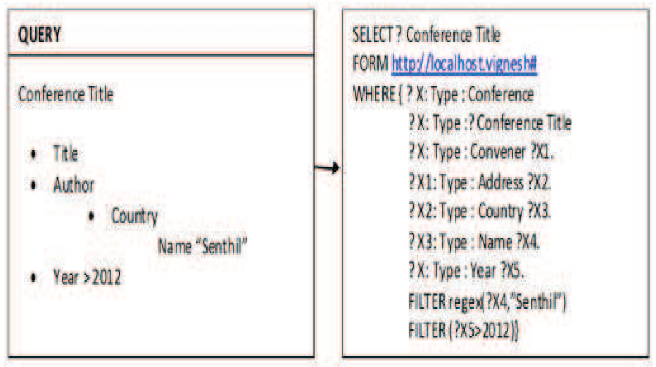
1. INTRODUCTION

In a mashQL Language, the various algorithms have been implemented, but the query detecting algorithm overcomes those algorithms based on performances, such as fast response time, scalability and its detecting accuracy. The mashQL Language has been defined with its greater characteristics of designing aspects such as, a mashQL searching box at the top of the screen on a mashQL Editor. Then, when the input has been given such as data in the format of a query, it works on it a following basis. RDF Input, mashQL, SPARQL, RESULT



Figu. 1. MashQL queries

The mashQL Language on a query Detection algorithm deals with the screen on a mashQL as the mashQL Query Editor. In mashQL Query Editor, at first we give a query in the search box, the RDF input has been identified to load data from the loader in the server. The RDF input, on a background, translated into a SPARQL, which is a basic query execution language in normal aspects. Then the query is executed in a format of an SPARQL. The SPARQL, then do, its job i.e interacting with the server based on a query that we had given, then this query has been run on a server for an accurate result. After, the results noted, they has to be displayed on a screen.



```
SELECT ? Conference Title
FORM http://localhost.vignesh#
WHERE { ? X: Type : Conference
        ? X: Type :? Conference Title
        ? X: Type : Convener ?X1.
        ? X1: Type : Address ?X2.
        ? X2: Type : Country ?X3.
        ? X3: Type : Name ?X4.
        ? X: Type : Year ?X5.
        FILTER regex( ?X4,"Senthil")
        FILTER ( ?X5>2012)}
```

Fig. 2. MashQL mappings to SPARQL

The user deals with the result, that the SPARQL displays on a screen, after the result have been viewed, an further investigation of an result, if the user needs an clarification, then again an mashQL editor has been executed once again by clicking the further search option on a screen. If this algorithm has been defined and executed on online schema, the mashQL has to be run through a query regarded website on an whole based as an query, which is a very considerable drawback for an real time and their varying time also increases based on an evaluation.

2. RELATED WORK

Query detection is the art of extracting related details easily from a database for a given query. In background, query that has been given by the user to mashQL editor translated into SPARQL and executed in the same format for further processing to be continued for identification of details to produce a result.

Simple search box, the user has an highly interaction between the system, but their working process was an failure consideration. Since, in a simple search box, the user just can give a keyword, then system works on that keyword and gives its suggestion immediately to auto complete a keyword. It has a basic fundamental data scheme, the problem it undergone is that it doesn't play the role of a Query Language.

Graphical Queries are queries that are very much considerable for an semi-structured data, here the users formulate the queries in the form of a filling tables and are of the major disadvantage is that the user must be known of the schema that they are to be submit the data in an schematized format and the user must known the technical details of an schema.

Generalized Queries are the naturally characterized queries for a language description. Here, in this type of queries, it permits a user to write their queries in a natural language sentences. Then, there natural language sentence has been translated into a formal language, such as SQL, XML etc. Here, the user not required to have knowledge about their schema. The main disadvantage, that generalized queries undergo is that language ambiguity and mapping between multiple meaning of keywords that we submit.

Formulated Queries are the queries that have a conceptual meaning on towards them. Such as the Query can details with an OPM, UML diagram in care of a diagrammatic format. If the people select a particular part of a figure, then it is translated into SQL as user convenience. The major drawback in formulated Queries is that user must be aware of a schema.

RDF 3X & SHARD is storage system, such as it can be suffered as a database. In RDF3X, a one-node store, it constructs index over the possible permutations of the given object & gives the result, whereas, SHARD is open source, triple storage system. It considers data in the format of flat files. The number of iterations has been executed on an SPARQL Query & Output the result.

3. OUR PROPOSAL

3.1 Architecture

The Query detection algorithm architecture gives an full overview of an mashQL language using Query detection algorithm. In the given architecture at first the user 'A' enter an information of system of some large class of applications, such as library information system, musical player etc., and the information system the user gives an query to the client system, the query that the user submits to system in an format of unknown scheme to him.

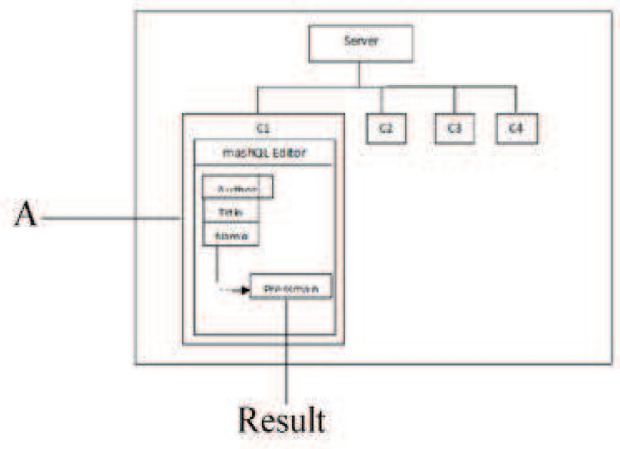


Fig. 3. System architecture

Fig. 3 shows the mashQL using query detection algorithm architecture. The client system, which is formatted with the mashQL Editor considers the query and translate to SPARQL. The query executes in a manner of SPARQL to the server and from the server, it loads a related data's to the query and displays it on the screen as a result.

3.2 QDT ALGORITHM

This query detection algorithm (QDT) deals with the query from user to editor, and it is most probably responsible for understanding a source of data. It includes the following Query models to be followed, viz

- Identify the dataset.
- Query object selection.
- Adding object filter to detect.
- Results format.

$$Q \in Q_k : \pi_O \sigma_p = \text{'Data'}(G)$$
$$Q \in Q_L : \pi_s(G) \pi_O(\sigma_{O \in URI}(G))$$
$$Q \in S$$

Fig. 4. Dataset identification

$$(Q \in Q_{kj} \text{->} P \in \pi_p \, \sigma_p = \text{'Data'}(G)$$
$$(Q \in Q_L) \text{->} P \in \pi_p \, \sigma_p = \text{'Subject'}(G)$$
$$(Q \in S) \text{->} P \in \pi_p \, \sigma_p = \sigma(G)$$
$$P \in S$$

Fig. 5. Query selection

$$(Q \in Q_L) \wedge (P \in S) \text{->} O \in \pi_{o1} (\sigma_{O \in URI}(G))$$
$$(Q \in Q_L) \wedge (P \notin S) \text{->} O \in \pi_{o1} (\sigma_{O \in URI}(G))$$
$$(Q \in Q_K) \wedge (P \in S) \text{->} O \in \pi_{o1} (\sigma_{O \in URI}(G))$$
$$(Q \in Q_K) \wedge (P \notin S) \text{->} O \in \pi_{o1} (\sigma_{O \in URI}(G))$$
$$(Q \in S) \wedge (P \in S) \text{->} O \in \pi_{o1} (\sigma_{O \in URI}(G))$$
$$(Q \in S) \wedge (P \notin S) \text{->} O \in \pi_{o1} (\sigma_{O \in URI}(G))$$

Fig. 6. Filter inclusion

4. QDT AND CONTROL FLOW

The overall process of mashQL using QDT and its control flow gives an overview of how the system works through it. In fig. 7 the control flow diagram shows the clear overview. The user A gives an input query to the system by using search box. Then the query searching process was done through SPARQL. Thus the QDT plays the role to achieve the fast response time in a searching process for finding a given query by the user. If the related details of query occurs in a multiple databases then those databases also occurs in a result page for the user to select. If the user selects and gives further query to process then the same process continues as from the mashQL editor.

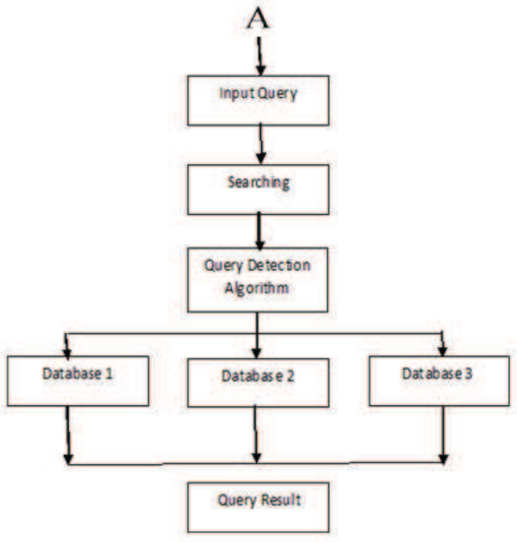


Fig. 7. Query detection control flow

5. PERFORMANCE ANALYSES

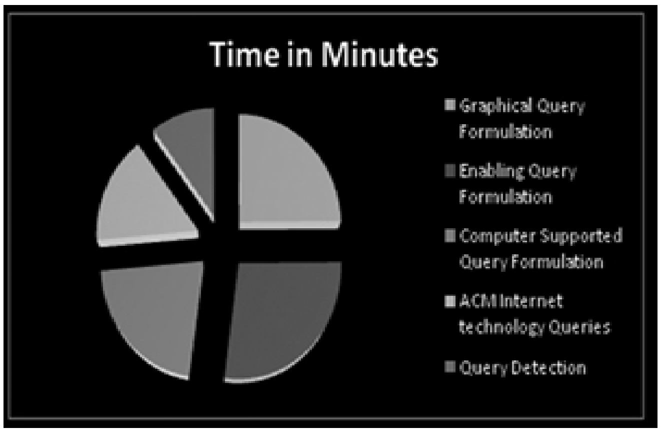


Fig. 8. Comparison chart

Fig. 8 presents the results for the comparison of the algorithms that have been previously used in an existing mahsQL editor for finding a query related results. The results gives a knowledge that the query detecting algorithm overcome existing algorithms on the fast response time to query given by the user.

6. CONCLUSION

We proposed a Query Detection algorithm with its architecture in a mashQL. We have specified various algorithms in comparison with Query detection algorithm. Based on the chart nature of comparison graph, we proposed that our algorithm has fast response time. Our architecture is able to perform upto 3 orders of different queries performed by different users in paralled. For future work, we plan to add a text categorization component to our algorithm and explore how updates can be handled efficiently by our architecture.

7. REFERENCES

[1] Mustafa Jarrar, Marios D.Dikaiakos "A Query Formulation Language for the Data Web", 2012

[2] Petropoulos, M, Papakonstantinouy, Vassalos V: Graphical Query Interfaces for Semistructured Data ACM Internet Technology, 5(2). 2005

[3] Prud'hommeaux E, Seaborne A: SPARQL Query Language for RDF. 2008

[4] Popescu A, Etzioni O, Kautz H: Towards a theory of natural language interfaces to databases. 8th Con on Intelligent user interfaces. 2003

[5] Russell A, Smart R, Braines D, Shadbolt R.: NITELIGHT: A Graphical Tool for Semantic Query Construction. SWUI Workshop. 2008.

[6] Steer D, Miller L, Brickley D: RDFAuthor: Enabling everyone to author rdf," WWW'02 Developers Day, 2002.

[7] Stockmeyer L, Meyer A: Word problems requiring exponential time. STOC'73.

[8] Tummarello G, Polleres A, Morbidoni C: Who the FOAF knows Alice? ISWC Workshops. 2007

[9] Savvides C: MashQL: A Step towards Semantic Pipes. M.Sc. Thesis. Computer Science dept., University of Cyprus, May 2010.

* * *

[1]Mr. U.Vignesh has been serving as an Assistant Professor with the Department of Information Technology (IT), Mookambigai College of Engineering affiliated to Anna University, Chennai. He obtained his B.Tech and M.Tech degree in Information Technology from Anna University, Chennai. His research interest includes Data mining, Distributed systems, Image Processing.

[2]Mr.P.Senthilraja is an Assistant professor in the Department of Computer Science and Engineering (CSE), Sudharsan Engineering College affiliated to Anna University, Chennai. He did his B.Tech in IT and M.E in CSE Anna University, Chennai. His research interest includes Image Processing and Cloud Computing.