Finding Resources from Middle of RDF Graph and at Sub-Query Level in Suffix Array Based RDF Indexing Using RDQL Queries

Hikmat Ullah Khan and Tahir Afzal Malik

***Abstract*—These Semantic Web promises to describe semantic information about its resources based on metadata concept. There are various schemes for querying metadata described with RDF and RDFS for effective query retrieval. Akioyoshi Matono et al. proposed indexing and query processing scheme for path based RDF query using suffix array. Sung Wan Kim proposed improved and efficient scheme for query processing based on longest common prefix concept. The results were shown using RDQL queries. There are two deficiencies, one that the proposed scheme deals with forward or backward queries and does not target resources in the middle of RDQL query. Second, various forms of RDQL queries especially based on sub-query concepts can not be answered from both the above cited schemes. In our proposed work, we have formally discussed proposed schemes to remove both these deficiencies and proposed respective solutions. The proposed work has been analyzed empirically and have found correct. The work will help to improve effectiveness in the search and retrieval of resources from suffix based RDF indexing using RDQL queries.**

***Index Terms*—Semantic Web, RDF Graph, Suffix Array, RDQL**

I. Introduction

In current years, the role of web has been revolutionized globally in about all fields of life. Semantic Web [1], [2] promises to add semantics and sense to web content. The open secret behind all such promises is metadata. For the purposes of advanced processing, currently available metadata on web is insufficient, both in terms of quality and quantity. The Semantic Web, on the other hand, makes it possible to perform high-level processes, such as question-answer, reasoning, deduction, semantic searches etc.

RDF (Resource Description Framework) [3] is framework to describe data and their semantics. RDFS (Resource Description Framework Schema) [4] is used to specify schematic information, for instance, resources, properties, and classes. It exploits web link structure to use URIs to assign to object subject relationships known as triplets. This model benefits us with using and mixing of structured as well as semi-structured data across diverse applications. RDQL (RDF Data Query Language) [5] is the query language that has been implemented in many RDF systems for extracting information from RDF graphs.

The Paper has been formulated as follows: After introduction of the domain knowledge, then background information about the terms and concepts under consideration like suffix array, basic and improved indexing of RDF graph based on suffix array. Then problem definition has been discussed briefly, then detailed problem has been defined and proposed methods for solution of the problem have been discussed in formal way. This work is somewhat different than our previous works [6], [7] but those works introduced us to lead towards semantic web. .It is worth mentioning that figures related to all concepts and sample RDF graph for explanation have been taken same as those defined in [8], [9] for ease of understanding for the reader.

Manuscript received February 28, 2012; revised May 6, 2012.

H. U. Khan is with the Department of Computer Science, COMSATS Institute of Information Technology, Attock 43600, Pakistan (e-mail: hikmatullah@comsats.edu.pk).

T. A. Malik is with the Al-Faisal University, College of Business and Tourism, Abha 61321, Kingdom of Saudi Arabia (e-mail: t.a.malik@pscabha.edu.sa)..

II. Background Information

A. Suffix Array

A suffix at the position i in a given text, is the sub-sequence beginning from i-th position to the end. For the sample text ' mississippi', the suffix at position 6 is 'ssippt'. If we sort all suffixes of string then putting it an array forms suffix array [10]. In other words, it is a list of all extracted suffixes of text in lexicographical order. Whenever suffix pattern repeats in the text, all the suffixes appear consecutively in the suffix array. By applying binary search on the suffix array, a specific string pattern can be retrieved. With the help of suffix array, for instance, taking text "abracadabra" as sample text to form the suffix array, first, index values are assigned to the sample text. Index values depict positions where binary search can be applied. In this example, as index values are specified character by character, so the sample text can be searched with the help of the suffix array.

Text	a	b	r	a	c	a	d	a	b	r	a
Index	0	1	2	3	4	5	6	7	8	9	10

Fig. 1. Assigning index points to input.

Second, index points are sorted according to their corresponding suffixes. The correspondence between the index points and the suffixes have been shown in Fig. 2 a) in indexed based order while the sorted suffixes have been shown in Fig. 2 b).

Suffix											Index
a	b	r	a	c	a	d	a	b	r	a	0
	b	r	a	c	a	d	a	b	r	a	1
		r	a	c	a	d	a	b	r	a	2
			a	c	a	d	a	b	r	a	3
				c	a	d	a	b	r	a	4
					a	d	a	b	r	a	5
						d	a	b	r	a	6
							a	b	r	a	7
								b	r	a	8
									r	a	9
										a	10

Sorted Suffix											Index
a											10
a	b	r	a								7
a	b	r	a	c	a	d	a	b	r	a	0
a	c	a	d	a	b	r	a				3
a	d	a	b	r	a						5
b	r	a									8
b	r	a	c	a	d	a	b	r	a		1
c	a	d	a	b	r	a					4
d	a	b	r	a							6
r	a										9
r	a	c	a	d	a	b	r	a			2

Fig. 2. a) The extracted suffixes: before sorting. b). the extracted suffixes: after sorting.

Resultantly, after sorting, the index values results in the suffix array for sample text [11].

10	7	0	3	5	8	1	4	6	9	2

Fig. 3. The final constructed suffix array.

B. Indexing Of Rdf Graph Based On Suffix Array

Akioyoshi Matono *et al.* [8] introduced an indexing scheme using suffix array to efficiently process the path-based queries over RDF data [8]. This scheme treats RDF data as a Directed Acyclic Graph and extracts all paths in the form of an alternation of labels of nodes and labels of arcs from root nodes to terminal nodes.

An example RDF graph is shown in the figure 4 to demonstrate the use of suffix array for indexing.

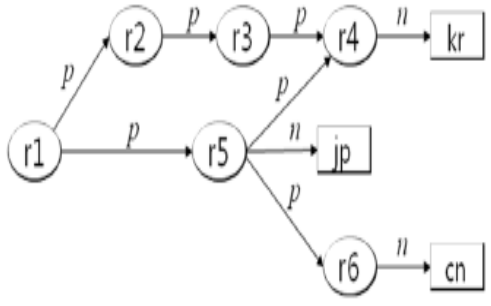

Fig. 4. An example of RDF graph

An integer pair (pid, idx) has been assigned as an index point for each suffix, since suffixes are extracted from different paths. The steps to generate the suffix array are shown in Figure 5 showing that all the suffixes are sorted in lexicographical order and duplicate suffixes are eliminated.

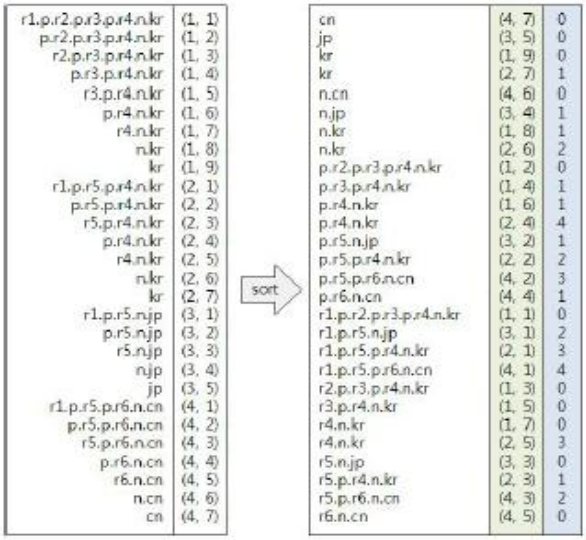

Fig. 5. Suffix array index construction

C. Improved Indexing of RDF Graph Based on Suffix Array

Sung Wan Kim proposed two schemes to improve query processing performance [9]. We hence proposed an index structure to reduce binary search space and introduced a query evaluation approach to reduce the overhead caused by repeating direct pattern matching. It devised algorithms for querying Forward and Backward RDQL queries on RDF graph using LCP (longest common prefix concept). The value of LCP_i [p] maintains the length of the longest common prefix from the suffix patterns of SA_i [p] and SA_i [p-1]. Finally, experimental evaluations demonstrated the proposed approach improves performance compared to the previous approach for path-based RDF queries.

III. Problem Definition

There are two problems. Firstly, there is no method to find Resource, using RDQL query to retrieve resource mentioned in middle of RDQL query, from RDF graph based on Suffix Array technique. For example, following Fig. 4, if we have no mechanism to find *r*3 from *r*1 *p r*2 *p r*3 *p r*4 *n kr* pattern. For details, the proposed mechanism provides sufficient example for understanding.

Secondly, as there are various possible shapes of RDQL thus to meet the requirement of those there is no mechanism to find from SA for other possible shapes of RDQL as cited on sub-query etc. We discuss the scenario and then solution in proposed method.

IV. Proposed Solutions

A. Finding Resource from Middle Portion of RDF Graph

It is important that as the Suffix array based query retrieval mechanism [8] is giving us both forward query as well as backward, thus let us take the advantage of both such algorithms [12]. It is notable that we take the same RDF graph example as used in earlier work for consistency and ease of understanding. The proposed method can be decomposed into following steps:

1) First split the query intending to find resource in the middle of the RDF graph. But here arises question where to split the query, it is the point where we have mentioned the resource which we want to have output. After split, we have two sub queries, let us name the query from starting triple to the required output point as Sub-Query_Forwarding as the other sub-query that is formed with same select but where part but having only those remaining triples in this sub-query, which are in the Main Query but not in the Sub_Query_Forwarding. Let us denote it as Sub_Query_Backward.
2) Then, using the algorithm for Suffix Array, execute the Sub-Query_Forwarding and put the result into Set Result_Subset_Forward.
3) Then, using the algorithm for Suffix Array, execute the Sub-Query_Backward and put the result into Set Result_Subset_Backwartd.
4) Finally, taking intersection of both Result_Cubset_Forward and Result_Subset_Backward gives the desired output named as Output_Set

Let us elaborate our proposed method with the help of an example.

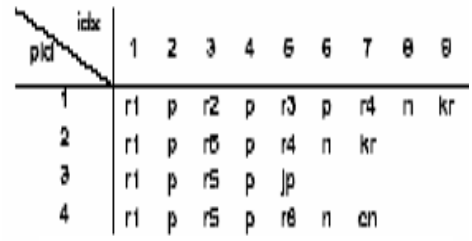

Figure 6: Path Information Table for RDQL Query Sample

Our intended RDQL is

Select ?y
where (*r*1 *p r*5), (*r*5 *p* ?*y*), (?*y n cn*)

Then after first step, we get Sub_Query_Forward as follows

Select ?*y*
where (*r*1 *p r*5), (*r*5 *p* ?*y*),
while Sub_Query_Backward is given as
Select ?*y*
where (?*y n cn*)

In second step, as we execute both queries, After executing sub_Query_Forard, we have Result_Subset_Backwartd = {*r*4, *r*6}, while after executing sub_Query_Backward, Result_Subset_Backward = { *r*6 }. Then taking intersection of both sets, we get the final desired output as Output_Set {r6}

B. Sub-Query Based Resource Retrieval from RDF Graph

Let us see what are the different scenarios in RQDL having Sub-query within one main RDQL query, that may not answered from Suffix array technique and algorithm mentioned in [9].

1) One Triplet Query having Two Variables

To elaborate all the simple technique each for different scenarios for sub query processing, the following example have been taken.

Select ?*x*, ?*ri*
where (?*x px* ?*ri*)

First search only suffixes of length 2 only in the suffix table and then find the *px* as second last element in the Two length suffixes only and then find the resource *ri*, found as last word. Then put that result in the formation of actual query to be answered, which is.

Select ?*x*
where (?*x px Ri*)

It is notable that for intermediate result, for notation purpose Rx has been used; for instance for SELECT ?*x*, the retrieved intermediate will be depicted as Rx.

2) Sub-Query: Up to Two Variables

Now, let us discuss the proper sub-query format for the case where two variables are concerned. For instance,

Select ?*y*
where (?*x p r*1) (?*x p* ?*y*)

Applying our proposed scheme, we get Sub_Query_1

Select?*x*
where (?*x p r*1)

It return set of resources, let us denote as Result_Sub_Query_1 (denote it as *Rx*), for each resource, we apply the Sub_Query_2, i.e.,

Select ?*y*
where (*Rx p* ?*y*)

3) Sub-Query: Up to Three Variables

Let us now, explain the proposed scheme for sub-query having three variables. For instance,

select ?*x* ?*z*
where (?*x p* ?*y*) (?*y p* ?*z*)

can be reshaped into the following two sub-queries:

select ?*x* ?*y*
where (?*x p* ?*y*)

Let us denote the results of ?*x* and ?*y* as *Rx*, and *Ry* respectively. Now *Ry* will be used for retrieval of ?*z* part.

Select ?*z*
where (*Rx p Ry*) (*Ry p* ?*z*)

Now we have result of *rx* and *rz*, which is required one. However, it is important that the intermediate variable, such as y in this case can be blank node, so proper computational handling of blank node must be managed in the implementation, which is intuitive one as each blank do has unique URI.

4) Handling Filters for RDQL Query processing on RDF Graph based on Suffix Array

For applying filters in RDQL queries, the proposed scheme is again different ones.

For instance,
Select ?*x*
where (?*x n* ?*y*)
AND ?*y* >= 24

It is important to note that ?*y* is literal value as n is data property not object property, which is depicted as p in the given example. The query will not be split into sub queries but it looks for suffixes of length 3. Then it will retrieve only those suffixes of length 3 whose data property matches satisfies the condition given in the query.

V. Conclusion

Suffix array has been used for indexing in many perspectives but here we have examined the suffix array based RDF indexing using RDQL queries. The paper has

been formulated on the work done by [8], [9] in this domain. There were deficiencies in those works that have been addressed and solutions have been presented. First, in RDF graph, resources in the middle portion of RDF graph can not be retrieved, which has been solved by dividing the query into two queries for processing. Secondly, many other deficiencies which exist in the existing works at sub-query level has been explored, discussed and then solution of all such have been presented. This improves the effectiveness of the indexing mechanism due to targeting middle nodes in the RDF Graph. The work can be used in semantic caches using suffix arrays based on RDF Indexing and using RDQL queries to improve effectiveness.

ACKNOWLEDGMENT

The first author is thankful to Prof. Dr. Abdul Qadir, for his comments and suggestions. Dr. Abdul Qadir, an expert in the Semantic Web and Semantic Cache, is currently serving as Head, Centre for Distributed and Semantic Computing and Dean, Faculty of Engineering and Applied Sciences, Muhammad Ali Jinnah University, Islamabad.

REFERENCES

[1] What the Semantic Web can represent. T. Berners-Lee. [Online]. Available: http://www.w3.org/DesignIssues/RDFnot.html .1998
[2] World Wide Web Consortium: Semantic Web. [Online]. Available: http://www.w3c.org/2001/sw/ 2001
[3] World Wide Web Consortium: Resource Description Framework (RDF) Model and Syntax Specification. [Online]. Available: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ W3C Recommendation, 22 February 1999.
[4] World Wide Web Consortium: Resource Description Framework (RDF) Schema Specification 1.0. [Online]. Available: http://www.w3.org/TR/2000/CR-rdf-schema-20000327/ W3C Candidate Recommendation 27 March 2000.
[5] Andy Seaborne. RDQL - A Query Language for RDF. [Online]. Available:http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/, 2004.
[6] U. Manber and E.W. Myers. "Suffix Arrays: A New Method for On-Line String Searches," *First ACM-SIAM Symposium on Discrete Algorithms*. pp. 319-327, January 1990. Brief Introduction to Suffix Array. [Online]. Available: http://sary.sourceforge.net/docs/suffix-array.html
[7] Brief Introduction to Suffix Array [Online]. Available: http://sary.sourceforge.net/docs/suffix-array.html
[8] A. Matono, T. Amagasa, M. Yoshikawa, and S. Uemura, "An Indexing Scheme for RDF and RDF Schema based on Suffix Arrays," In *the Proc. of the First International Workshop on Semantic Web and databases (SWDB)*, pp. 151-168, September 2003.
[9] S. W. Kim, "Improved Processing of Path Query on RDF Data Using Suffix Array," *Journal of Convergence Information Technology*, vol. 4, no. 3. September 2009.
[10] B. Liu and B. Hu, "Path Queries Based RDF Index," In *the Proc. of the First International Conference on Semantics, Knowledge, and Grid (SKG)* , pp. 91-93. 2006.
[11] T. A. Malik and H. U. Khan, "Perforamnce Measurement using Distributed Perforamnce knowledge management system: Empirical case study of Coca Cola Enterprises," *International Review of Business Research Papers*, vol. 6, no. 1, pp. 250-282, Feburary 2010.
[12] M. Shoaib, M. N. Yasin, H. U. Khan, M. I. Saeed, and M. S. H. Khiyal, "Relational WordNet model for semantic search in Holy Quran," *Presented at the International Conference on Emerging Technologies*, ICET 2009, pp. 29-34, October 19-20, 2009.

Hikmat Ullah Khan is serving as Assistant Professor, Department of Computer Science, COMSATS Institute of Information Technology, Attock, Pakistan. He got his Master degree in Computer Science from International Islamic University, Islamabad. His research interest includes but not limited to Semantic Web, Information Retrieval, Data Mining, and Social Network Mining.

Tahir Afzal Malik is serving as Lecturer in Al-Faisal University, College of Business and Tourism, Abha 61321, Kingdom of Saudi Arabia. He did his Masters in Computer Science from International Islamic University Islamabad. He received his Master degree in Business Administration from Bradford, UK. His research interests include both computer science as well as management sciences like Quality Assurance, Networking, and Management Information System.