International Journal of Internet, Broadcasting and Communication Vol.5 No.1 14-17 (2013)
http://dx.doi.org/10.7236/IJIBC.2013.5.1.14

IJIBC 13-1-4

An Architecture for Efficient RDF Data Management Using Structure Index with Relation-Based Data Partitioning Approach

Duc Nguyen[1], Sang-yoonOh[2]

[1]*Department of Computer Engineering, Ajou University, Korea*
nguyenminhduc@ajou.ac.kr

[2]*Department of Computer Engineering, Ajou University, Korea*
syoh@ajou.ac.kr

Abstract

RDF data is widely used for exchanging data nowadays to enable semantic web era. This leads to the need for storing and retrieving these data efficiently and effectively. Recently, the structure index in graph-based perspective is considered as a promising approach to deal with issues of complex query graphs. However, even though there are many researches based on structure indexing, there can be a better architectural approach instead of addressing the issue as a part. In this research, we propose architecture for storing, query processing and retrieving RDF data in efficient manner using structure indexing. Our research utilizes research results from iStore and 2 relation-based approaches and we focus on improving query processing to reduce the time of loading data and I/O cost.

Key words : *Structure index, RDF graph data, Data partitioning, RDF data management*

1. Introduction

RDF data [2] is widely used for the semantic web data nowadays. A lot of researches are targeting to exploit those data to enable semantic web era by making use of and inferring new and useful information from them. It is also a one of the most important technologies in semantic web to represent metadata and exchange information by exploiting relationships through RDF statements. Besides, RDF graph which is a visual representation of triples data becomes more and more popular due to its simplicity and ease of understanding and reading this semi-structured data. Hence, many organizations as well as web sites have created more and more RDF data (up to billions of triples). RDF data can be found everywhere at the moment and is distributed over the semantic web. This turns into the need of maintaining this vast amount of data in an optimized way. Then people can be able to find the relationships inferred from RDF statements for new useful info and query for necessary data more conveniently.

This leads to the need for storing and retrieving these data efficiently and effectively. Recent studies [3, 4, 5, 8] show that there are several perspectives of ways to do this: relational, entity and graph-based approaches. Storing data with traditional way of relation [3] requires many join operations. Entity perspective [5], which applies IR technologies, focuses on keyword and entities search. Among these perspectives, the structure index in graph-based perspective is considered as a promising approach to deal with issues that need to be

Manuscript Received: Nov. 11, 2012 / Revised: Jan. 10, 2013 / Accepted: Feb. 8, 2013
Corresponding Author: syoh@ajou.ac.kr
Tel:+82-31-219-3620, Fax: +82-31-219-3620,
Dept. of Computer Engineering, Ajou University, Korea

solved with complex query graphs. To achieve the efficiency of RDF data management, we apply the graph-based perspective which offers graph theory and graph database to find the similar structure inside RDF data to create index. In this paper, we propose architecture for storing, query processing and retrieving RDF data in an efficient manner using structure index. Our research utilizes research results from iStore [4] and 2 relation-based approaches[8] and we focus on improving query processing to reduce the time of loading data and I/O cost by partitioning data and storing the intermediate structure index graph.

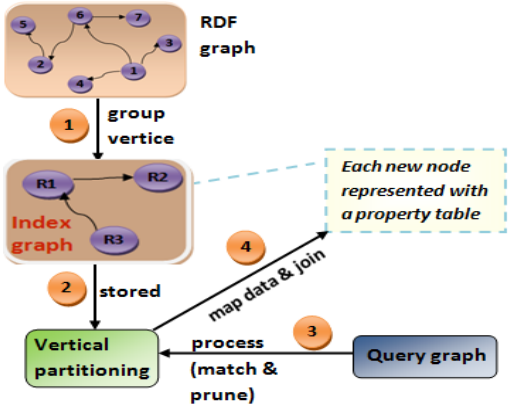
2. Related work

So far there are several researches [6, 7, 9] that built and introduced their structure index approaches with different types of grouping RDF data. gStore [6] used signature with bit-strings to encode vertices, edges and created structure index with a balance tree for the signature data graph. It focused on supporting queries with wildcard and had good performance when processing simple queries. Picalausa, F. at. el. [9] recently proposed a new architecture for structure indexing. They grouped a set of triples (instead of resource vertices) to build nodes for the index graph and define a new concept *equality type* to make edges for the connection between these nodes. Among those approaches, in our opinion, iStore [4] can be considered as the state of the art in the domain of graph-based approach in general, and structure indexing in particular, for RDF data. Thanh Tran et. al. [4] proposed a structure index (grouping vertices with similar structure) and data partitioning schema to store, retrieve data, and process query graph. Their approach provided high scalability where the query matching task is not affected by the size of data, but the index graph's size which depends on the structure of RDF data. However, iStore [4] used the graph theory of *homomorphism* (k-bisimulation) to find the vertices having similar structure. This graph theory is known to be NP-complete problem and can be solved in polynomial time [1] if the graph is just a single edge (single triple in RDF data). This means that it will take longer time to process with vertices that have many adjacent edges. Furthermore, when adding new RDF data, especially a new triple with existing subject (vertex) and predicate (edge), it is difficult for iStore [4] to determine the group (node) which this new RDF data can be inserted to. iStore [4] uses a 3-column (*subject, predicate, object*) table to store all triples of the subjects belong to the grouped vertices. Hence, fetching data for all triple patterns of a same variable needs to join data later to get full result.

3. Architecture for RDF data management

Different from iStore [4], our approach with property tables can get whole structure row of data matched by given predicates in the query and does not need to join data. Also, we can find the similar vertices with polynomial time complexity $O(n^2)$ (n is the number of vertices) using the lists of incoming/outgoing labels (described below). With new data that has existing subject and predicate, these lists can easily find the associated representative node for this new RDF data to insert to.

The following figure shows our research methodology to store, retrieve data and process query efficiently:



3.1 Structure index graph

Structure index is an ideal data structure in graph-based approach for indexing technology. This kind of index isa summary graph revised from the source RDF graph by grouping the vertices (or parts of data)based on some frequent patterns or structural similarity.

$$\begin{cases} \mathbf{v1} \rightarrow (\{\mathbf{v1\_incoming\_labels}\},\{\mathbf{v1\_outgoing\_labels}\}) \\ \mathbf{v2} \rightarrow (\{\mathbf{v2\_incoming\_labels}\},\{\mathbf{v2\_outgoing\_labels}\}) \\ \quad \ldots \ldots \\ \mathbf{vN} \rightarrow (\{\mathbf{vN\_incoming\_labels}\},\{\mathbf{vN\_outgoing\_labels}\}) \end{cases}$$

In iStore [4], vertices are combined into a group called *extension* node. In this paper, we also use a similar concept, *representative* (R) *node*, to exhibit a group of similar vertices which have the same lists of distinct incoming and outgoing labels. To do this, we maintain a list of vertices in RDF graph data associated with their adjacent edges' labels as follow:

These *representative* nodes form a new graph called structure index of original RDF graph data.

Partition data into representative nodes

R1 — Outgoing edges

Subject	predicate1	predicate2	...	predicateN
v1	$o1_1$	$o1_2$	...	$o1_N$
v2	$o2_1$	$o2_2$	...	$o2_N$

The RDF data is then partitioned to these R nodes, each of which is represented with a property table [5] (i.e. ***v1, v2*** are grouped into node **R1**) containing only outgoing labels (properties) of internal vertices. For fast data lookup, we employ 2 indexes (***R, p***) and (***R, s***) on these tables (nodes) to get the list of <*subject, object*> for given predicate (with matching R node) and to get the full row of data for given subject (with matching R node), respectively.

3.2 Index graph management

We manage and store the index graph (with representative nodes and existing predicates) by using vertical partitioning [3] approach, which represents each predicate (between representative nodes) with a 2-column table of subject, object. This helps us to find the matches of query graph to the structure index graph by retrieving and joining data from these tables. For each match, we get a list of pairs (*representative node, variable)* in which the *variable* is considered to be matched with the found node.

3.3 Query processing

In this step, query graph is processed to find a match with the index graph via the predicate tables as described above. After getting a query match with structure index, we can prune it by removing the triple patterns which have the *undistinguishedvariables* - variables that are only objects (leaf nodes) and do not exist in the SELECT section of the query. In case there is no match found or the matching query is pruned completely, we can skip data processing step to reduce I/O cost.

3.4 Data processing

As aforementioned, for each matching query with variable set $\{x_1, x_2, x_3, \ldots x_i, \ldots, x_m\}$, we have a list of mapping pairs (x_i, R_i)in which $R_i$is a representative node ***i*** in the index graph. Then appropriate RDF data can be retrieved for each variable based on property tables of associated R node. For the queries in which any part of triple patterns has a certain structure, i.e. where the predicates are of a same *subject* variable, data is only retrieved once without joining step. This helps to reduce loading data time. Besides, fetching data for a matching query from the index graph takes less time than loading all data from source data and joining them to get the result.

4. Conclusion

In this paper, we proposed an efficient RDF data management architecture which reduces the time of loading data and I/O cost using structure indexing with relation-based data partitioning models. We believe our approach can help other researchers to get easily an insight of the structure index applied to manage RDF

data with graph-based perspective. Then they are able to find out new approaches to solve the issues of managing RDF data.

However, our work also has some challenging obstacles to address; maintaining the lists of incoming/outgoing labels may face a scalability issue when RDF data is quite larger. Besides, diverse structure RDF data makes the grouping method for different structural vertices become ineffective. Updating/adding new RDF data can be a big challenge with relation-based data partitioning models.

Acknowledgement

This work was jointly supported by the MKE, Korea under the ITRC support program supervised by NIPA (NIPA-2012-(H0301-12-2003)) and Basic Science Research Program through the NRF of Korea (No. 2012-0003198).

References

[1] JIS, M. Tech CSE Syllabus. "Computers and Intractability A Guide to the Theory of NP Completeness." (1979).
[2] http://www.w3.org/RDF/ 2012
[3] D. J. Abadi, A. Marcus, S. R. Madden, and K. Hollenbach. Scalable Semantic Web Data Management using vertical partitioning. In VLDB, 2007.
[4] Thanh Tran , Gunter Ladwig, Sebastian Rudolph, “iStore: Efficient RDF Data Management Using Structure Indexes for General graph Structured Data”, Institute AIFB Karlsruhe Institute of Technology, 2009.
[5] Wang, H., Liu, Q., Penin, T., Fu, L., Zhang, L., Tran, T., Yu, Y., Pan, Y.: Semplore: A scalable IR approach to search the Web of Data. Web Semantics: Science, Services and Agents on the World Wide Web 7(3), 177{188 (2009)
[6] Zou, Lei, Jinghui Mo, Lei Chen, and Dongyan Zhao. "gStore: answering SPARQL queries via subgraph matching." Proceedings of the VLDB Endowment 4, no. 8 (2011): 482-493.
[7] Tran, Thanh, and Günter Ladwig. "Structure index for RDF data." - Workshop on Semantic Data Management (SemData@ VLDB). 2010.
[8] Luo, Y., Picalausa, F., Fletcher, G. H., Hidders, J., & Vansummeren, S. (2012). Storing and indexing massive rdf datasets. Semantic Search over the Web, 31-60.
[9] Picalausa, F., Luo, Y., Fletcher, G., Hidders, J., & Vansummeren, S. (2012). A structural approach to indexing triples - The Semantic Web: Research and Applications, 406-421.