RDF/XML 저장을 위한 Trie 기반 모델

오정정*,김주리*,이현창**,한성국*

*원광대학교 컴퓨터공학부, **원광대학교 정보전자상거래학부

A Trie-Based Model for RDF/XML Storage

Sumit Bisai, Kim, Ju-Ri, Lee, Hyun-Chang, Han, Sung-Kook

Wonkwang University

E-mail : {cyanic, hclglory, skhan}@wonkwang.ac.kr

요 약

인터넷상에서 데이터 양은 빠른 속도로 증가하고 있으며, 더불어 RDF/XML 문서 크기 또한 상당히 크게 양산되고 있다. 이러한 상황은 효율적인 저장 모델을 요구하고 있으며, 저장된 데이터에 대해 효율적으로 질의를 처리할 수 있어야 한다. 이와 같은 많은 데이터를 저장하고 처리하기 위한 도구들이 존재하지만 이들 대부분은 과도한 조인연산, 데이터 및 스키마 관리에 있어서 문제점들을 안고 있다. 이에 본 연구에서는 문제점을 최소화하기 위해 Trie와 벡터(vector)를 사용하여 데이터를 저장할 수 있는 모델을 제안하며, 다음과 같은 내용을 중심으로 살펴본다. 먼저, 과도한 조인 연산을 피하며, 데이터를 지역화(localized) 및 캐쉬(cached)되게 저장한다. RDF 그래프를 상하로 이동하는 복잡도를 동일하게 하며, 스키마 정보와 데이터의 의미를 함께 저장하고 메모리 내(in-memory) 실행 문제 등을 해결한다. 이렇게 함으로서 의미 추출과 결과를 확대 시키는데 소요되는 시간을 최적화시킬 수 있다.

1. Introduction

The rapid increase of data present in the web demands development of efficient storage schemes to facilitate quick extraction of the data. Moreover, under the current technology, data is often stored as text, making them non-interpretable by machines. These factors led to the requirement of an intelligent agent which can understand and accordingly interpret the data and thus paved the way for Semantic Web [1].

The Semantic Web architecture is consisted of several layers, which as a whole is known as Semantic Web cake [2]. One such layer is the Resource Description Framework (RDF) [3], which is a language for representing the worldthrough triplets. It provides a framework for representing the metadata, along with support for serializing and exchanging of data models. It does so with the help of identifying things by Uniform Resource Identifiers (URI). Further, it also assigns each element a property or property value. This paper proposes a model for storing this RDF document efficiently using a Trie [4].

Presently there are tools which do store RDF documentsand perform queries over them. But these schemes suffer from certain drawbacks:

- Models like Jena [5] store the data in the form of 5 different tables. Some queries require multiple join operation. Many a times the size of these tables is also large, and hence join operation becomes quite expensive.
- Another major problem that the present models face is moving up the RDF graph [6] while extracting data. Even for simple queries

sometimes the time extends to a good number of hours. [7]

· There are several tools and query languages which can query over the triple very efficiently, but they have no specific functionality or semantics to discriminate betweenschema and data information. [8]

The proposed model takes care of all above drawbacks. The rest of the paper is organized as follows. Section 2 describes the data structures used by these models to store the RDF/XML documents along with the notations used. The final section covers the effectiveness of the model

2. The Proposed Algorithm and the Data Structures Used

The whole algorithm revolves around precise understanding of the RDF/XML document or the RDF graph. Each line in the RDF/XML document is parsed well by interpreting the meaning of rdf:about, rdf:id, rdf:Description, rdfs:Class and etc.

2.1 Introduction of terms for defining the proposed model

· **Definition 1** [Leaf value pair]: Each leaf of the Trie has a set of pair of numbers in the form of (a, b), where a is the row number and b the column number of both the Vectors.

These values refer to the position of that URI or literal in two Vectors. These are the indices of each resource and predicate and they are distinct in nature. No two of them have the same index value. Fig 1 presents the structure of (a, b) tuple. In the figure 1, the leaf value of "http://www.swl.com/IITian/Sumit" is (0, 0). The advantage of using indices is that instead of storing long character strings we store in the form of numbers which are easy to compare. The procedure for finding 'a'and 'b' is explained in Section 2.2

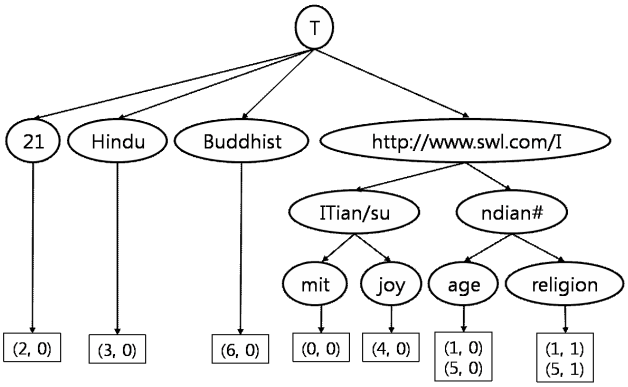

Figure 1.A typical trie structure of RDF/XML

· **Definition 2** [Index Vector]: The Index Vector, V-1 establishes a link between the previous, current and the next URI or literal which came across while parsing the RDF/XML document.

· **Definition 3** [Reference Vector]: The Reference Vector, V-2 is quite similar to V-1. It contains the literals, labels, statements, URIs etc. They are stored at positions obtained by the values of leaf in the Trie.

So, instead of storing the long URI values in Trie or Index Vector, it stores only indices. Reference Vector is used to get back the URI values, literal values, etc of these indices when required.

· **Definition 4** [URI Parts]: An URI is split in two parts called P and S, where P is the substring before the last '/' or '#' in the URI value and S is the remaining substring.

· **Definition 5** [Index Vector pair]: At each entry of V-1 two numeric indices [x, y] are stored, where both x and y are of type (a, b).

The first index, xpoints to the Resource or Property from where it came to that particular URI (node/arc) in the RDF graph. The other index ypoints to the Resource or Property to which the URI (node/arc) follows to in RDF graph. Thus it respects the Triple format of storing the data but not exactly in the sequence of Subject followed by Predicate and finally Object. With this technique it also stores

schema information like subClassOf, subPropertyOf, domain, range etc.

For example, V-1 has values of the form [{1, 0}, {3, 0}]. This is represented as [x, y], where both {1, 0} and {3, 0} are of types (a, b).

2.2 Explanation of the Algorithm

Now, we will describe the construction of Trie structure along with the fillings of Index vector V-1 and the Reference vector V-2 using the above algorithm in details.

Example 1 This example covers cases of handling the following.

(a) Two or more RDF graphs

(b) Literals be it String Literals orTyped. Here only String Literal is explained but the same rule holds for Typed Literals.

(c) Two or more RDF graphs or rdf:about having the same

- o Predicate value
- o Literal value

Figure 2 provides the RDF/XML representation of the underlying example. Figure 1 presents the corresponding Trie.

```
<? Xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:cd="http://www.swl.com/ Indian#">
<rdf:Description rdf:about="http://www.swl.com/IITian/Sumit">
<cd:age>21</cd:age>
<cd:religion>Hindu</cd: religion>
</rdf:Description>
<rdf:Description rdf:about="http://www.swl.com/IITian/Sujoy">
<cd:age>21</cd:age>
<cd: religion>Buddhist</cd: religion>
</rdf:Description>
</rdf:RDF>
```

Figure 2. The RDF/XML Representation of Example 1

Table 1. The Vectors corresponding to Example 1

The 'a' values	Position (a, b)	V-1 value	V-2 value
0	0, 0	[{_}, {1, 0}]	http://www.swl.com/IITian/Sumit
1	1, 0	[{0, 0}, {2, 0}]	http://www.swl.com/Indian#age
	1, 1	[{0, 0}, {3, 0}]	http://www.swl.com/Indian#religion
2	2, 0	[{1, 0}, {_}] [{5, 0}, {_}]	21
3	3, 0	[{1, 1}, {_}]	Hindu
4	4, 0	[{_}, {5, 0}]	http://www.swl.com/IITian/Sujoy
5	5, 0	[{4, 0}, {2, 0}]	http://www.swl.com/Indian#age
	5, 1	[{4, 0}, {6, 0}]	http://www.swl.com/Indian#religion
6	6, 0	[{5, 1}, {_}]	Buddhist

3. Accessing Knowledge for Query Answering

SPARQL [9] the mostly used query language for querying RDF has most of its queries in triple pattern, where each of the Subject, Predicate and Object can be a variable. These queries match against a given pattern. But in the mentioned scheme such a pattern is not required. Here we discuss two queries each covering different query patterns.

3.1 Query 1: Extraction of all information of a particular resource

Now let's run a query over the RDF graph. Suppose the request is to find all the students of India, where URI of students is "http://swl.com/IITD#students" and URI of India is "http://swl.com/India". Here the subject is 'India' and the predicate is 'student'.

・The scheme first matches the subject and the predicate URI in the Trie and get the (a, b) values of the relevant leaves. Here the values are {0, 0} and {1, 0} respectively.

・After going to {0, 0} th position in the V-1, it chooses that particular pair whose 'y' value is {1, 0}. Asthere is only one pair present it picks up that value. (The case of the presence of multiple pair is explained below).

・Now it moves on to {1, 0}th position in the V-1. Here also only one pair is present and its 'y' value is {2, 0}.

・Again the scheme moves on to {2, 0}th position, but here it finds a set of 3 pairs. The URI value of the leaves can be obtained from V-2, which shows it to be a Blank node.

・Moving in the same fashion to all the three pairs gives the type and full description of the Blank Node.

・The first pair points to {3, 0} and then further to {3, 1}. It then gets the URI value of {3, 1} from V-2 which identifies itstype as Bag.

・At position {3, 2} there is a link to {4, 0} whose URI value is obtained from V2 as "http://swl.com/IITian/Sumit". Similarly from {3, 3} there is a link to {4, 1} whose URI valueis "http://swl.com/IITian/Sujoy".

Thus the query turns out to be complete by saying that it is a container of type Bag and having values of 'Sumit' and 'Sujoy'.

4. Conclusion and Future Work

In this work we propose a model for the storage of RDF data by using Trie and Vectors. We have proposed algorithms for the insertion, deletion and modification of data in the storage model. We have also taken three examples covering various structures of RDF graph. Due to space constraint we can't take examples consisting RDFS. But the same algorithms can be used with the schema added to the RDF.

Besides so many factors mentioned above one major focus over this whole work is localization of data. This adds a lot to the query processing.

The points mentioned in Section 4 clearly indicate that the Trie-based model has enough potential to be an effective model for storing RDF/XML documents. However, to justify the claim in a stronger way one needs to compare the efficacy of the model with the existing storage schemes, and compare their efficiencies by running them on a large number of RDF/XML documents. We are presently working in this direction. We intend to test the different schemes on the LUBL.

[참고문헌]

[1] Grigoris Antoniou and Frank van Harmelen: A Semantic Web Primer, The MIT Press, Cambridge, Massachusetts, London, England.

[2] T. Berners-Lee et al. Primer: Semantic Web "Layer Cake", "http://www.w3.org/2004/Talks/0412-RDF-functions/slide4-0.html"

[3] T. Berners-Lee et al: RDF Primer, "http://www.w3.org/TR/REC-rdf-syntax/".

[4] Michael T.Goodrichand RobertoTamassia: Data Structures and Algorithms in Java

[5] Kevin Wilkinson, Craig Sayers, Harumi Kuno: Efficient RDF Storage and Retrieval in Jena2

[6] Dieter Fensel: Ontologies, A silver bullet for knowledge Management and Electronic Commerce

[7] T. Berners-Lee et al: RDF Semantics, W3C Recommendation 10 February 2004, "http://www.w3.org/TR/rdf-mt"

[8] Jeen Broekstra, Arjohn Kampman and Frank van Harmelen, Sesame: A Generic Architecture for Storing Querying RDF and RDF Schema.

[9] w3c, "http://www.w3.org/TR/rdf-sparql-query/"

[10] Nick Drummond, Matthew Horridge, Robert Stevens, Chris Wroe, Sandra Sampaio, The University of Manchester, Pizza Ontology, http://www.co-ode.org/ontologies/pizza