Transforming RDF via RDF Schema into XML Documents for XPath Query Language

Win Lai Hnin, Khin Nwe Ni Tun
University of Computer Studies, Yangon, Myanmar
winlaihnin.84@gmail,knntun@gmail.com

Abstract

The Semantic Web is an extension of the current Web that will allow to find, share and combine information more easily. To harvest such power requires robust and scalable data repositories that can store RDF data. Most of the existing RDF storage techniques rely on relation model and relational database technologies for these tasks. The mis-match between the graph model of the RDF data and the rigid 2D tables of relational model jeopardized the scalability of such repositories and frequently renders a repository inefficient for some types of data and queries. This paper proposes the rules that can transform RDF data into XML document. These papers discusses the idea of collection of subject, predicate and object from RDF graph model and this collection is used to transform RDF data into XML documents and store in the XML repository and extract the XML data by the XML query language.

1. Introduction

Most of the web sites today are designed for human reading, not for computer understanding. Computers essentially play a role in parsing web pages for displaying and processing jobs. They have no reliable way to draw the semantics from a page. The Semantic Web will improve the meaningful content of the web pages. It is not completely a new generation of web, but an expansion of the current one. The meaning in the Semantic Web is mostly represented by Resource Description Framework (RDF). RDF encrypts these meanings in the sets of triples that build meaningful webs about related things. These are recognized by the Universal Resource Identifiers
(URIs) which tie meanings to a unique definition so that users can easily find them and their relationships on the web.

However, a considerable amount of resources is available in eXtensible Markup Language (XML) rather than in RDF. The main success of XML is its flexibility. Users can define their own tags to describe elements in the XML document. Nested, tagged elements are the building blocks of XML. Each tagged element has a sequence of zero or more attribute/value pairs, and a sequence of zero or more subelements. XPath is a declarative query language for XML that provides simple syntax for addressing parts of an XML document. XPath can specify sets of nodes and sets of paths in an XML document tree.

The needs to develop applications on the Semantic Web and support search in RDF data call for RDF repositories to be reliable and robust. As in the context of RDB and XML, the selection of storage models is critical to a data repository as it is the dominating factor to determine how to evaluate queries and how the system behaves when scales up.

The rest of this paper is organized as follows. Section 2 describes the background theory of the proposed system. Section 3 discusses the design and implementation of the proposed system. Section 4 describes XML storage to extract the XML data. Finally, we conclude our paper.

1.1 Related Work

Most of the existing RDF data repositories [2, 3, 4] rely on relational models for data storage and evaluate SPARQL queries by rewriting them into SQL queries and then executing them in the RDB engine. Among them there are two major directions:(1) keeping the simple triple data model of RDF data, e.g. *triple store* [4]; and (2) decomposing RDF triples into relations, either based on predicates, e.g. *vertical partition* or based on semantics, e.g. *property table* [3].

The *triple store* does not scale well as the evaluation of a complex SPARQL query invokes many self-joins. Various indexing techniques [1] were proposed as remedies, at the cost of huge increase in storage space and decrease in the scalability and update efficiency. The *vertical partition* [2] works well for SPARQL queries when all predicates in the WHERE clause are known. Otherwise, all tables have to be accessed and results unioned. The *property table* incurs small number of joins for some queries because a selection in one property table can match multiple simple access patterns. However it suffers storage redundancy and large overhead in query evaluation [2].

The proposal of serializing RDF graph into XML trees to utilize existing XML technologies [7] focused on representing all RDF features such as blank node in XML, but pays less or no attention to the efficiency of RDF data storage and query evaluation. It either leads to XML data [7] with large redundancy or flat XML data [4] that cannot take full advantage of XML query evaluation techniques.

Mo Zhou and Yuqing Wu [5] proposed the two RDF-to-XML decomposition algorithms for the decomposition in two steps: (1) the schema-level decomposition which maps an RDF schema to a set of XML schemas and (2) the data-level decomposition which maps RDF data to a set of XML documents

conforming to the XML schemas which brings inefficient in mapping RDF data to a set of XML documents conforming to the XML schemas in some applications.

Steve Battle [12] proposed Gloze approach that is the bidirectional mapping between XML and RDF. The Gloze approach showed how the content of this vanilla XML may be modeled in RDF, allowing XML to be mapped into RDF. This approach allowed to directly interpreting on XML document as an RDF model without passing through RDF/XML and uses XML schema as the basis for describing how XML is mapped into RDF and back again.

Stefan Bischof, Stefan Decker and Thomas Krennwallner [11] proposed mapping method between RDF and XML with XSPARQL. The XSPARQL language is combined XQuery and SPARQL, allow querying XML and RDF data using the same framework and respectively transform one format into the other. XSPARQL provides concise and intuitive solutions for mapping between RDF and XML in either direction, addressing both the use cases of W3C GRDDL (Gleaning Resource Description from Dialects of Language) working group and SAWSDL that describe an initial implementation of an XSPARQL engine, available for user evaluation.

1.2 Overview of the Proposed System

Semi-structured data model organizes data entries in a tree structure and represents the

semantic relationships among them via containment relationships. Tree pattern matching is at the core of the query languages for XML, e.g. XPath and XQuery. We observe the similarity between RDF and XML, in term of data representation (e.g. using links to represent relationships among data instances) and query
(e.g. tree pattern matching in XML and graph pattern matching in RDF) and propose to leverage the sophisticate storage management and query evaluation techniques of XML data repositories to store and query RDF data.

Specifically this system collect the subject, predicate and object from RDF graph model and transform RDF data into XML document and store XML repository to extract information by XPath queries. Figure (1) shows the architecture of proposed system.

The contribution of the proposed system is as follows:

- The idea of collection of subject, predicate and object from RDF graph model and rules of transformation from RDF data into XML documents.
- XML-based RDF data storage that doesn't depend on the XML schema.
- The XML query processing to extract information from XML repositories.

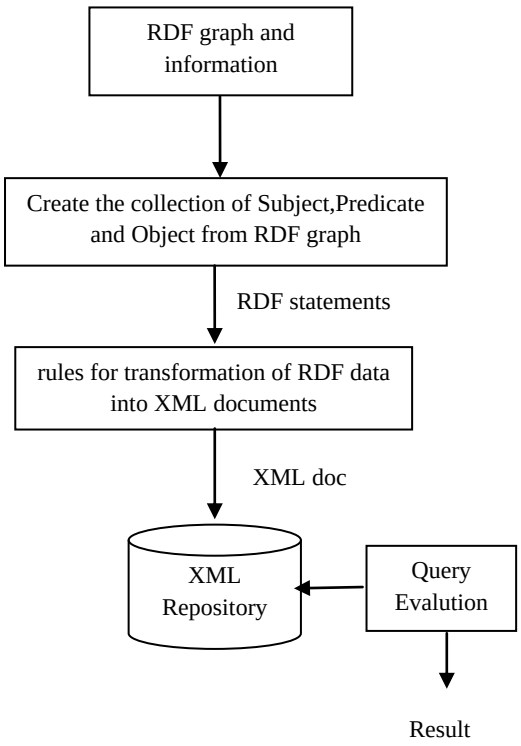


Figure 1: System Architecture

RDF data are significantly different from XML data in syntax and data model: RDF data and schema are directed graphs with both nodes and edges labeled, while XML data are trees with only nodes labeled. Although our work, as other RDF storage approaches, is syntax independent, the difference between the data models brings substantial challenges to storing and querying RDF data using XML techniques, in transforming graphs into trees, keeping storage efficiency.

2. Background Theory

2.1 Knowledge Representation

There are three essential requirements for arbitrary language used for data interchange on the web:

(1) Language should have the ability to describe any form of data to satisfy all the potential need.

(2) The represented data should be easily accessed by other organizations and its supported software, such as parsers or query APIs, should be reusable (syntactic operability).

(3) It should have definitions for mappings between terms in the data (semantic interoperability).

2.2 RDF

The vision of the Semantic Web is to allow everybody to publish interlinked machine-processable

information with the ease of publishing a web page. The basis for this vision is a standardized logical data model called Resource Description Framework (RDF). RDF data is a collection of statements, called triples of the form (s, p, o), where s is a subject, p is a predicate, and o is an object; each triple states the relation between the subject and the object. A collection of triples can be represented as a directed typed graph, with nodes representing subjects and objects and edges representing predicates, connecting subject nodes to object nodes. Basic RDF data model consist of three objects:

Resources **:** an element, a URI, a literal,…

Properties **:**directed relations between two resources

Statement **:**combination of a resource, a property and a value.

2.3 XML

XML is a meta-language that enables designers to create their own customized tags to provide functionality not available with HTML. XML is a restricted version of SGML, designed especially for Web documents. SGML allows document to be logically separated into two: one that defines the structure of the document (DTD), other containing the text itself. XML retains key SGML advantages. XML is not intended as a replacement for SGML or HTML. It is a data format for exchanging data on the web, between databases and elsewhere. Elements or tags are most common form of markup. First element must be a root element, which can contain other (sub) elements. XML document must have one root element. Element begins with start-tag and end-tag. XML element is case-sensitive. Attributes are name-value pairs that contain descriptive information about an element. A given attribute may only occur once within a tag, while (sub) elements with same tag may be repeated.

3. Design and Implementation

Our procedure has two main steps. The first one presents the idea to collect subject, predicate and object from the RDF graph model. The second uses this collection to transform RDF data to XML documents.

3.1 Rules for RDF Transformation

In this stage, first create the collection of subject, predicate and object from the RDF graph model as an input. These collections are used to extract element, subelement and attribute. The idea of this step is as follows:

- The object of the first statement is root element of document.
- For each subclass (predicate with rdfs:Class, rdfs:subClassof,rdf:Property) we decide whether they are element or subelement or attribute of the document.
- For data value of every element, we can predict the type of predicate in RDF.

RDF statement is a collection of triples of Subject, Predicate and Object; each triple states the relation between the subject and the object. Predicate is the main building block of RDF statements. The rules of deriving element, subelement and attribute are as follows:

IF Predicate is rdfs:Resource
THEN the object of this predicate is root element of the XML document

ELSEIF Predicate is rdfs:Class
THEN the object of this predicate is the element of the document

ELSEIF Predicate is rdfs:subClassof
THEN the subject of this predicate is subelement of its objects in the document

ELSEIF Predicate is rdf:Property
THEN the object of this predicate is attribute of its subjects in the document

ELSEIF Predicate is rdf:type
THEN the subject of this predicate is the attribute value of its object In the document

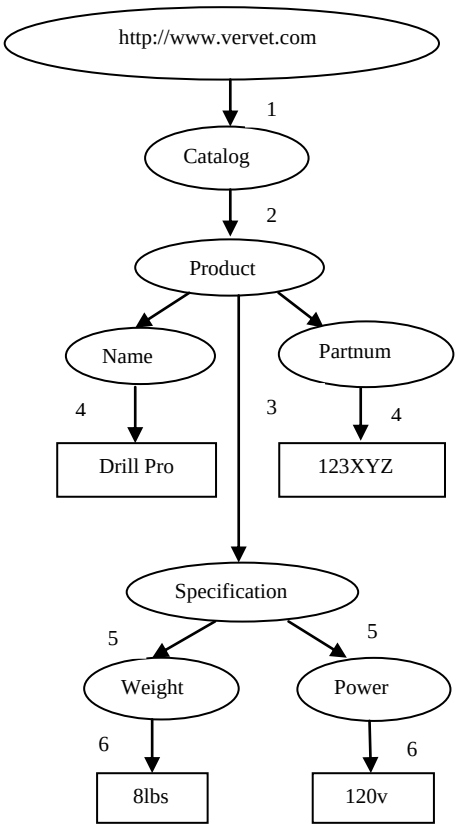
ELSEIF Predicate is rdfs:domain
THEN the object of this predicate is the attribute of its subject in the document

ELSEIF Predicate is rdf:value
THEN the object of this predicate is the value of the document

Rules for RDF Transformation

3.2 Example of the proposed system

In order to illustrate for our procedure, we use sample files at http://www.vervet.com/. This website supports free download of the XML editor, XMLPro. We choose files describing product, because these kinds of files are so popular on the web as well as in the electronic business. The graph description of the RDF triples is presented in the Figure 2.



1. rdfs:Resource
2. rdfs:Class
3. rdfs:subClassof
4. rdf:Property
5. rdfs:domain
6. rdf:value

Figure 2: RDF graph

Table 1 is the RDF statements for Figure 2 that represent the meaning of the data as well as the relationship between data. For example, Name is a property of Product and its value is Drill Pro.

Table 1: RDF statements from RDF graph

Subject	Predicate	Object
http://www.vervet.com	rdfs:Resource	Catalog
Catalog	rdfs:Class	Product
Product	rdf:Property	Name
Name	rdf:value	"Drill Pro"
Product	rdf:Property	Partnum
Partnum	rdf:value	"123XYZ"
Product	rdfs:Class	Specifications
Specifications	rdf:Property	Weight
Weight	rdf:value	"8lbs"
Specifications	rdf:Property	Power
Power	rdf:value	"120v"

Using rules in section 3.1, we derive element, subelement and their corresponding attribute as below:

Root element: Catalog, Element: Product (Attribute: Name, Partnum). Subelement: Specification (Attribute: Weight, Power). After having the set of elements, subelements and attributes from the previous step, we can produce XML documents by using algorithm in section 3.2. Following is XML document:

```
<Catalog>
<Product Name= "Drill Pro" Partnum= "123XYZ">
   <Specification  Weight= "8lbs"
     Power= "120v"/>
</Product>
</Catalog>
```

The above XML document is interpreted by RDF triples in the table 1.

4. XML Storage

The first approach to storing XML documents is to employ traditional databases such as relational database or object-oriented database as the underlying storage. The second is to develop a specialized system, which is known as native storage. The underlying storage representation has a significant impact on the efficiency of query processing. Basically, a storage strategy can be defined as efficient if the system manages to retrieve data accurately; use storage resources competently and update data and schema correctly. This system uses native XML database because it has many advantages to support time-consuming.

A native storage basically means building a specialized data manager that contains XML as its fundamental unit of its logical model. These data are stored and retrieved in their original structure, with no mapping process required. Nevertheless, the NXD requires a particular underlying physical storage model, which can be a custom database or any typical database model. Using this approach may work best, especially on scalability, data retrieval and handling of huge amounts of data. Nevertheless, it is not suitable when integration between various heterogeneous XML documents is needed. TIMBER, XBase and Natix are some examples of native storage.

XPath query language is designed for XML documents. It provides a single syntax that we can use for queries, addressing and patterns. Fundamentally, an XPath is an expressing. Specifically, identity constraints require the resultant node set to contain only elements or attributes. Fragment identifiers restrict the resultant node set to contain only elements. Location paths nominally provide the grammar for typical XPath expressions for XML schemas. In an XML schema, all location paths are either relative to an enclosing component (for identity constraints) or relative to an entire XML document (for locating schema components). One of the general features of a location path is the ability to navigate along a number of axes. An *axis* specifies a direction of movement in the node tree. For example, you might specify a *child* node, an *attribute* node, an *ancestor* node, or a *descendant* node. The XPath Recommendation defines 13 axes. An identity constraint is limited to containing only the axes *child*, *attribute*, and *descendant-or-self*. Furthermore, an

identity constraint can only use the shortcut notation for these axes. Predicates are very powerful, but slightly confusing when first encountered. A predicate is strictly a filter. A predicate filters out desired nodes from a node set. Examples of XPath queries for the resultant of XML document are the following:

(1) /Catalog/* (selects all child elements of the root element Catalog)
(2) /// Specification (selects Specification element in the document)
(3) /Product [@Name] (selects Name attribute of the Product element)

5. Conclusion

To answer the increasing demands on RDF repository, carefully studied the existing RDF data management systems, identified the preferred properties of an RDF repository and proposed to take advantage of the latest XML data storage and efficient query processing techniques. In this paper, we have proposed rules to transform RDF data into XML documents by using RDF schema vocabularies. Our proposed method enables languages used in procedure do their jobs as their original functions. XML is used for describing data, RDF for providing triple statements about data and RDF schema for supporting vocabularies to describe the relationship among data. In addition, our approach is efficient for time consuming in translation from RDF data to XML documents for supporting Semantic Web applications in various domains.

References

[1] C. Weiss, *et al.* Hexastore: sextuple indexing for semantic web data management. *PVLDB*, 1(1):1008–1019, 2008.
[2] D. J. Abadi, *et al.* Scalable Semantic Web Data Management Using Vertical Partitioning. In *VLDB*, 2007.
[3] J. J. Carroll, *et al.* Jena: implementing the semantic web recommendations. In *WWW*, 2004.
[4] L. Sidirourgos, *et al.* Column-store support for RDF data management: not all swans are white. *PVLDB*, 1(2):1553–1563, 2008.
[5] Mo Zhou and Yuqing Wu. XML-Based RDF Data Management for Efficient Query Processing, 2010.
[6] Michel Klein, "Interpreting XML via an RDF Schema", *Database and Expert Systems Applications*, 2002.
[7] Norman Walsh. Rdf twig: accessing rdf graphs in xslt. *In Proc. Extreme Markup Languages*, 2003.
[8] Pham Thi Thu Thuy, Young-Koo Lee, Sung young Lee and Byeong-Soo Jeong, "Transforming Valid XML Documents into RDF via RDF Schema", 2006.
[9] Peter Patel-Schneider, and Jérôme Siméon, "The Yin/Yang Web: XML syntax and RDF Semantics", *11th International WWW conference*, Hawaii, 2002.
[10] S. Alexaki, *et al.* The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. In *SemWeb*, 2001.
[11] S. Bischof, S. Decker, K. Thomas, N. Lopes, A. Polleres. "Mapping Between RDF and XML with XSPARQL", April 2011.
[12] S Battle. "Gloze: XML to RDF and back again", http://www.hpl.hp.com/personal/steve-battle.
[13] Sergey Melnik, "Bridging the gap between RDF and XML", Dec 1999.
[14] T. Neumann, *et al.* The RDF-3X engine for scalable management of RDF data. *VLDB J.*, 19(1):91–113, 2010.
[15] Tim Berners_Lee, "Why RDF model is different from the XML model", Sep 1998, available at: http://www.w3.org/DesignIssues/RDF-XML.html.