Query Processing for RDF data Using XML Repository

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

Abstract

The Semantic Web, which represents a web of knowledge, offers us new opportunities to search for knowledge and information. To harvest such search 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 jeopardizes the scalability of such repositories and frequently renders a repository inefficient for some types of data and queries. In this paper, we propose a system that can store RDF data in the XML repository. This system serializes RDF data into RDF/XML and then maps into a XML document. We discuss the basic idea of serializing RDF data into RDF/XML and then mapping of RDF/XML to XML document.

1. Introduction

RDF (Resource Description Framework) is a W3C recommended language for describing linked data of the Semantic Web in the form of triples. RDF data is represented as labeled directed graphs, where each triple defines an edge from the subject node to the object node under label predicate. RDF has a number of possible serializations, including an XML serialization (RDF/XML), popularly used as the format for exchanging RDF data. In general, the order of statements in RDF is not important, and thus the order in which XML tags occur in RDF/XML can vary greatly whilst still preserving semantics.

The Extensible Markup Language (XML) developed by the WWW Consortium (W3C) [W3C98] is recognized today as the current standard for establishing interoperability on the Web. Since it provides a standard syntax for representing data, it is assumed to be a key enabling technology for exchange of information on the Web and within corporations. As a consequence, integration of XML data from multiple external sources is becoming a critical task. XPath is a declarative query language for XML that provides simple syntax for addressing parts of an XML document. With XPath, collections of elements can be retrieved by specifying a directory-like path, with zero or more conditions placed on the path.

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: In the next section, we discuss the translation from RDF to XML. In section 2.1, we discuss the RDF data model and then in section 2.2, we describe the serializing from RDF to RDF/XML and then we describe the XML document in section 2.3. And then we describe XPath query language for XML in section 3. Finally, we conclude our paper.

1.1 Related Work

Most of the existing RDF data repositories [2, 4, 6] 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* [6]; and (2) decomposing RDF triples into relations, either based on predicates, e.g. *vertical partition* or based on semantics, e.g. *property table* [4].

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. For example, the RDF data in Fig. 1(a) are stored in five tables. All need to be accessed to evaluate the SPARQL query above. 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].

An alternative approach [9] preserves the graph nature of RDF data by storing RDF graphs in an object-relational database. However, this separates the RDF schema and RDF primary data, hich brings difficulties in evaluating queries containing both schema and data instances.

The proposal of serializing RDF graph into XML trees to utilize existing XML technologies [3, 5, 8] 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 [8] with large redundancy or flat XML data [5] that cannot take full advantage of XML query evaluation techniques.

Mo Zhou and Yuqing Wu [7] 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.

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 terms 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 sophisticated storage management and query evaluation techniques of XML data repositories. Specifically we propose to serialize RDF data into RDF/XML and map RDF/XML to XML documents in an XML repository and XPath queries to be evaluated against the XML data using the latest XML query evaluation techniques.

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 and mapping graph pattern queries into tree pattern queries.

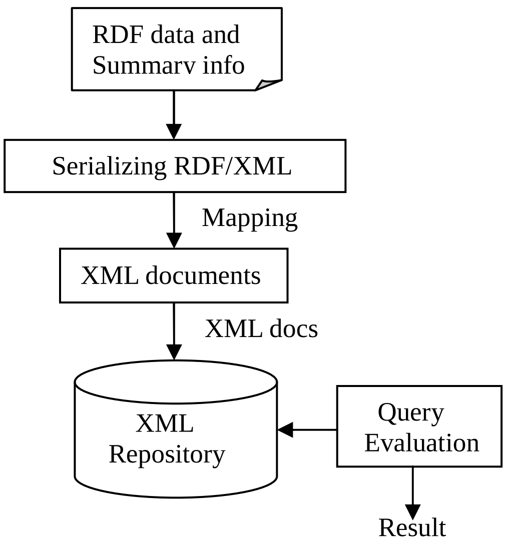

Figure (1): System Architecture

Our contribution can be summarized as follows:

- We propose an XML-based RDF storage that doesn't depend on the XML schema.
- We propose serializing from RDF into RDF/XML and the mapping from RDF/XML to XML documents.
- We discuss the XPath queries to extract information from XML repositories.

2. Translation from RDF to XML

2.1 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.

Example:

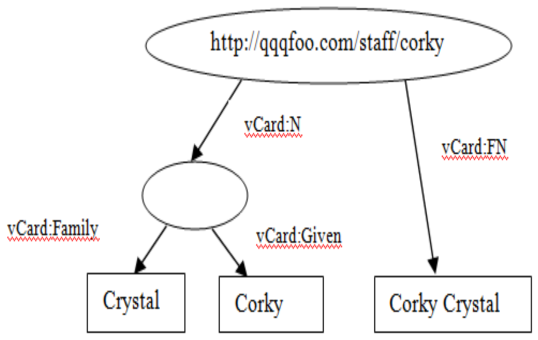

Figure (2): RDF data

In this example, one could create triple with the subject http://qqqfoo.com/staff/corky, the predicate vCard:FN, and a value as object which is identified by a literal "Corky Crystal". RDF offers the concept of blank nodes (also known as anonymous resources). Blank nodes allow the creation of resources without needing a URIref, since the node itself provides the necessary connectivity between the various other parts of the graph. In Figure (2), one could create another triple with the subject http://qqqfoo.com/staff/corky, the predicate vCard:N and the blank node that represents the data type properties vCard:Family and vCard:Given. The range of Family and Given properties is of type string. The RDF database integrates vocabulary from different namespaces, i.e. the standard RDF namespace rdf, a user-defined namespace vCard, as well as the namespaces foaf, dc and dcterms. The standard namespace rdf provides some basic vocabulary with predefined semantics, such as rdf:type used for typing URIs. Next, foaf provides domain-specific vocabulary to describe persons in a uniform way and dc and dcterms provide predefined vocabularies for describing bibliographic entities.

2.2 Serializing from RDF to RDF/XML

RDF/XML is the widespread serialization format for RDF graphs. RDF/XML is the normative syntax for writing RDF. The success of RDF/XML lies in its early availability and the number of tools that support RDF/XML processing. Therefore, RDF/XML is the recommended syntax for applications to exchange RDF information. The basic principle of RDF/XML files is the mapping of RDF nodes and arcs into XML elements, attributes, element content, and attribute values. Probably the most prominent serialization format is RDF/XML which allows encoding RDF databases as XML trees. The basic idea behind RDF/XML is to split the RDF graph into small, tree-structured chunks, which are then described in XML with the help of predefined tags and attributes. The RDF/XML format was primarily designed to be processed by computers. We propose pseudo code for serializing from RDF into RDF/XML is:

```
Input ⟵ Subject, Predicate, Object
Output ⟵ Serializing RDF/XML

IF (Subject is root node)
  THEN display root node in the about attribute of the Description element and then display predicate and object.
     IF (Object is BagID)
        THEN display Bag Element and then display
               its properties and value
        ELSE IF (Object is SeqID)
            THEN display Seq Element and then
                   display its properties and value
           ELSE IF (Object is AltID)
               THEN display Alt Element and then
                      Display its properties and value
           ENDIF
        ENDIF
      ENDIF
ENDIF
```

Pseudo code for serializing RDF/XML

An RDF graph can be considered as a collection of paths of the form- node, predicate arc, node, predicate arc, node, predicate arc … node, which cover the entire graph. In RDF/XML, these turn into sequences of elements inside elements which alternate between elements for nodes and predicate arcs. This has been called a series of node/ arc stripes, where the node at the start of the sequence turns into the outermost element; the next predicate arc turns into a child element, and so on.

Example:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf =
"http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:vCard =
"http://www.w3.org/2001/vcard-rdf/3.0#">

   <rdf:Description rdf:about =
"http://qqqfoo.com/staff/corky" >
      <vCard:FN> Corky Crystal </vCard:FN>
      <vCard:N rdf:parseType="Resource">
         <vCard:Family> Crystal </vCard:Family>
         <vCard:Given>  Corky </vCard:Given>
      </vCard:N>
</rdf:Description>
</rdf:RDF>
```

RDF/XML for Figure (2)

In this example, the outer rdf:RDF XML element encloses the scope of the RDF/XML. The inner rdf:Description element is the "frame-style" block of properties, all about the resource with URI http://qqqfoo.com/staff/corky. Here the element vCard:FN represents the property with the value "Corky Crystal". This element encodes for the URI reference that is defined by the namespace name (URI) for "vCard" which in this case is http://www.w3.org/2001/vcard-rdf/3.0#/ concatenate with the local name of the element (FN) giving the URI http://www.w3.org/2001/vcard-rdf/3.0#/FN. When a property has a URI value, an rdf:resource attribute is used on the empty property element with the URI as the attribute value. A property value can have an XML content when the parseType="Resource" attribute is used on the property element.

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.

In this paper, we map the RDF/XML to XML document. RDF/XML has an XML syntax that has a specific meaning. Every Description element describes a resource. Every attribute or nested element inside a Description is a property of that resource. Tags and attributes have a specific meaning and we can refer to resources by using URIs. The following is an example of XML-tagged document, contained in the file **staff.xml**.

```
 <?xml version="1.0"?>
<Staff id="corky"
xmlsn:vCard="http://www.w3.org/2001/vcard-
rdf/3.0#"
     <vCard:FN> Corky Crystal </vCard:FN>
     <vCard:N>
           <vCard:Family> Crystal </vCard:Family>
           <vCard:Given> Corky </vCard:Given>
     </vCard:N>
</Staff>
```

staff.xml document for Figure (2)

3. XPath Query Language

XPath 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. Evaluating an XPath expression results in one of the following:

- A node set
- A Boolean
- A floating-point number
- A String of Unicode character

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.

The easiest way to demonstrate a predicate is to discuss two similar expressions along multiple axes. Examples of XPath queries against staff.xml document are the following:

(1) /Staff/N/Family (selects Family element that is children of N element that is children of the root element Staff).
(2) ///Family (selects Family element in the document.
(3) /Staff/* (selects all child elements of the root element Staff).
(4) /Staff[@id] (selects the id attribute of the Staff element).

4. Conclusion

To answer the increasing demands on RDF repository, we 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 query processing techniques. We identified the system that serializing from RDF into RDF/XML and the mapping from RDF/XML to XML documents. In addition, our approach is efficient for time consuming in translation from RDF 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] Dave Beckett. RDF/XML syntax specification (revised). W3C Recommendation, February 2004.

[4] J. J. Carroll, *et al.* Jena: implementing the semantic web recommendations. In *WWW*, 2004.

[5] J. J. Carroll, *et al.* Rdf triples in xml. In *WWW*, 2004.

[6] L. Sidirourgos, *et al.* Column-store support for RDF data management: not all swans are white. *PVLDB*, 1(2):1553–1563, 2008.

[7] Mo Zhou and Yuqing Wu. XML-Based RDF Data Management for Efficient Query Processing, 2010.

[8] Norman Walsh. Rdf twig: accessing rdf graphs in xslt. *In Proc. Extreme Markup Languages*, 2003.

[9] S. Alexaki, *et al.* The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. In *SemWeb*, 2001.

[10] T. Neumann, *et al.* The RDF-3X engine for scalable management of RDF data. *VLDB J.*, 19(1):91–113, 2010.