On Contextual Queries for XML Documents Using Namespaces

Hiroko Kinutani, Masatoshi Yoshikawa, Yohei Yamamoto,
Takahiro Morimoto and Shunsuke Umeura
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara 630-0101, Japan
{hiroko-k, yosikawa, yohei-y, taka-mo, uemura}@is.aist-nara.ac.jp

Abstract

The "Namespaces in XML" and the emerging "XML Schema" are important standards in the family of the XML (eXtensible Markup Language). These concepts enable us to use element types and attributes defined as external namespaces or schemas with own element types and attributes. Consequently, there will be a large number of well–formed XML documents having different top–level structures and having the same bottom–level structures. The emergence of a new type of the structure of XML documents requires a new type of efficient retrieval mechanism.
In this paper, we describe the retrieval of the context nodes by specifying the scope of subtree using namespaces in the bottom–level structure of XML documents automatically.
keywords: XML, Namespace, Structured Documents, Ambiguous Queries

1 Introduction

XML[4], recommended by W3C(World Wide Web Consortium) and related specifications, such as "Namespaces in XML"[5], "XML Schema"[6, 7], show us new technologies to process XML documents having new types of structures. By referring schemas defined as namespaces, we are able to exchange documents and data with element types and attributes having common definitions. Using these features, an XML document can refer any namespace. In this situation, XML documents can be classified into the following four types:

1. Valid documents with only one independent schema;
2. Valid documents referring other schemas;
3. Well–formed documents including valid components referring other schemas; and
4. Well–formed documents with no schema.

Most of conventional researches and techniques are directed toward 1. and 2.. In this paper, we focus on 3. to make sure the features of these

```
1 <?xml version="1.0"?>
2 <books xmlns:dc=
  "http://purl.org/metadata/dublin_core/dc.xsd"
    xmlns:isbn="http://isbn.org/isbn.xsd">
3 <publisher pcode="8303">
4 <dc:publisher>NAIST Publishing Inc.
    </dc:publisher>
5 </publisher>
6 <book>
7 <creators>
8 <dc:creator>Hiroko</dc:creator>
9 <dc:creator>Yohei</dc:creator>
10 </creators>
11 <dc:title>XML Processing</dc:title>
12 <isbn:number>4-89470-073-0</isbn:number>
13 </book>
14 <book>
15 <dc:creator>Taka</dc:creator>
16 <dc:title>Multimedia Databases
    </dc:title>
17 <isbn:number>0-589092-2</isbn:number>
18 </book>
19 </books>
```

Figure 1: A sample digital library XML document

XML documents, and we introduce the notion of on contextual queries for XML documents using namespaces.

This paper is organized as follows. First, we describe XML Schema, XML namespaces and the problems of queries using those specifications in Section 2. In Section 3, we propose a function **context** to retrieve context nodes using namespaces. Finally, we conclude and show future work in Section 4.

2 XML schema and namespace

The XML Namespace Recommendation[5] extends the data model of XML to allow element

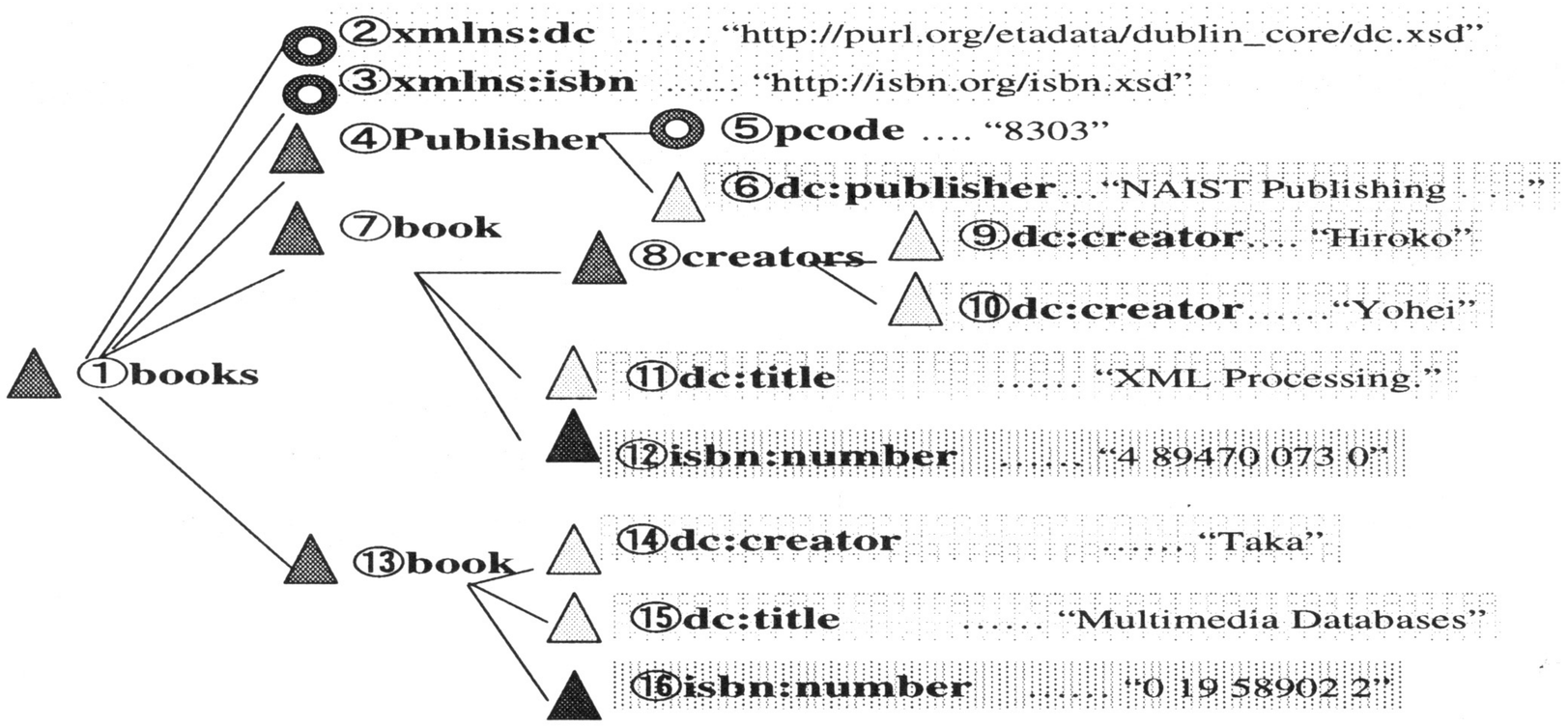

Figure 2: Tree representation of the XML document in Fig.1

type names and attribute names to be qualified with URIs. On the other hand, XML Schema describes that the URI refer to an instance of XML schema. It means that a set of names defined in a namespace is treated similarly as the set of element types and attribute names defined in the referring schema. Therefore, we can refer some namespaces combining some schema modules in order to create a desired schema.

Fig. 1 is a sample XML document of digital library. In line 2, two kinds of namespaces are defined. One of them refers to the XML schema defined at "http://purl.org/metadata/dublin_core/dc.xsd" and element types and attributes defined there are qualified with prefix "dc". Similarly, the XML schema defined at "http://isbn.org/isbn.xsd" qualifies element types and attributes with prefix "isbn". The structure of element types and attributes qualified with these prefixes are the same in every XML document referring to these namespaces. In contrast, element types and attributes with no prefix are independent of other XML documents. Consequently, XML documents referring to the same namespaces have different top–level structures and the same bottom–level structure.

Fig. 2 depicts a tree representation of the XML document in Fig. 1. Nodes are numbered from 1 to 16 in depth first order; element nodes with circle and attribute nodes with triangle. The nodes belonging to the scope of a reserved namespace with prefix "xmlns" are 2 and 3. The nodes belonging to the scope of a namespace with prefix "dc" are 6, 9, 10, 11, 14 and 15. Also, the nodes belonging to the scope of a namespace with prefix "isbn" are 12 and 16. The rest of element types and attributes,1, 4, 5, 7, 8 and 13 do not have any namespace, and do not have schema definitions anywhere.

2.1 Queries of XML documents using namespaces

Until now, many query languages for XML documents have been proposed[3]. However, the query language of XML documents has not been recommended by W3C yet. XQL[1, 2] is a query languages for XML documents. XQL contains several features including joins, text containment and extensible functions. XQL also includes the queries using namespaces. In XQL definition, a 'context' is the set of nodes against which a query operates. XQL allows a query to select between using the current context as the input context and using the root context as the input context. A query prefixed with '/' uses the root context. A query may use the child operator '/', and a query may also use the '//' operator to indicate recursive descendants. Attribute names are always prefixed with '@'. They are treated as children of the elements to which they belong. Wildcards '*' select all nodes in the context. The filter operator '[ ]' filters the set of nodes to its left based on the conditions inside the brackets. Multiple conditions may be combined using Boolean operators; "and", or, "union"(|), "intersect" and "both"(~). The followings are the examples of XQL using elements qualified with namespaces:

(1) Find all number elements belong to the namespace "http://isbn.org/isbn.xsd".

```
isbn := ''http://isbn.org/isbn.xsd'';

//isbn:number
```

(2) Find all books where the units attributes belong to the namespace "http://ecommerce.org/edi.xsd" is equal to "Yen".

```
edi := ''http://ecommerce.org/edi.xsd'';

//book[@edi:units=''Yen'']
```

(3) Find all books where the publication date elements belong to the namespace "http://purl.org/metadata/dublin_core/dc.xsd " is before January 1, 1999.

```
dc := ''http://purl.org/metadata
   /dublin_core/dc.xsd'';

//book[.//*[dc:date]
   lt date(''1999-01-01'')]
```

The followings are the issues of queries for XML documents with namespaces:

- Since XML recommendation allows well–formed documents, documents with unknown structure can be the target to retrieve. Consequently, it is important for queries based on ambiguous structure conditions.
- While the set of nodes can be retrieved with namespaces, it is difficult to retrieve the fragments which has relations with the specified namespaces semantically. Therefore, it is difficult to merge subtrees from one document source into another document subtree with the namespace's elements effectively.

3 Find context nodes using namespaces

As XML namespace and the XML Schema are becoming widely used, there will be many well–formed XML documents with the same namespaces. Those XML documents have different top–level structures and the same bottom–level structure. Each of those XML documents has no schema definition as a whole, but the part of those documents are referring to the same schema definitions as the namespaces. It is difficult for users and applications to know the exact path to query those XML documents. They can specify only the elements or the attributes belonging to those namespaces. Consequently, when we want to know the relationship between default namespace's element/attribute and query namespace's element/attribute, it is necessary to extract the domain of the context relative to the specified namespace in those XML documents.

In this section, we investigate how to find context nodes in XML documents when the name of element types or attributes belonging to certain namespaces are specified in queries.

For XML documents used in this section, we assume that those documents (1)use namespaces,and (2)have no relation among each namespaces. Therefore, in the domain of namespace A, element types or attributes belonging to B should not include element types or attributes of A and vice versa.(This assumption is suitable for the case where it is not allowed to import other schemas when we define a schema.) The notation of queries is based on XQL.

3.1 Context of simple XML queries

Let us consider the case to manage well–formed digital library XML documents in Fig 1. We suppose that user who issues queries know the Dublin Core namespace in which the book schema is defined with element types; "title" and "creator".

At first, we study simple XML queries using a namespace.

Query 1a:

```
dc:=''http://purl.org/metadata/
    dublin_core/dc.xsd'';

//*[dc:title contains ''XML'']
```

In the *Query 1a*, from the root node of this document structure, we search nodes through every path which is the element node named "title" defined at Dublin Core namespace and which contains the character strings "XML". The result of this query is a parent node set, at least one of whose child node satisfies the above filter conditions. That is the node 7 in Fig. 2. The path expression of this result is /books/book[0]. It is recognized that we extract the node 7 subtree as a contextual unit with related elements of dc:title.

Query 1b:

```
//*[dc:creator]
```

In the *Query 1b*, from the root node, we search nodes through every path which is the element node named "creator" defined at Dublin Core namespace. The result of this query is a parent node set, at least one of whose child node satisfies the above filter conditions. Those are node 8 and 13 in Fig. 2. Considering that dc:creator indicates author or writer, the node 7 and 13 are more appropriate. When the authors of well–formed XML documents use ad-hoc element sets, it is difficult to extract these nodes using these parent/child, ascendant/descendant operators in XQL.

To extract maximal domain of the context concerning the specified namespaces' conditions, we introduce a function **context**. This function computes a set of context nodes to specify subtrees relative to the specified namespace in the bottom–level structure of XML documents automatically. This **context** function can be implemented as a user defined function in XQL.

Definition 1 (Context) *For a set of nodes in XML document, the function* **context** *computes a set of nodes of subtrees, each of those subtrees is a maximal subtree having different path expressions each other except sibling relationship path expression including specified namespace's element or attribute. Consequently, both the argument and value of the* **context** *are sets of nodes in the XML documents.*

Query 1c:

```
(1) context(//*[dc:title contains ''XML''])
= context( {/books/book[0] })
= { /books/book[0] }

(2) context(//*[dc:creator])
= context( {/books/book[0]/creators,
   /books/book[1] })
= { /books/book[0], /books/book[1] }
```

In (1) and (2) of the *Query 1c*, the function **context** are computed by the result nodes of XML queries in *Query 1a* and *Query 1b*, respectively. In Fig. 2, the result node of (1) is 7, and result nodes of (2) are both 7 and 13. The XML document are divided into disjoint fragment by "dc:creator".

3.2 Context of conjunctive XML queries

Next, we study conjunctive XML queries using namespaces. Consider the case to extract subtree that has at least one creator element and the title element which contains the character strings "XML" simultaneously, and both elements are defined at Dublin Core namespace. *Query 2a* is a query specifying this condition in XQL. The "and" operator combines two XQL query. For instance, let q1 and q2 be XML queries, "q1 and q2" returns true when the intersection of q1 and q2 is non–empty, else returns false.

Query 2a:

```
//*[.//*[dc:creator] and
    .//*[dc:title contains ''XML'']]
```

In the *Query 2a*, from the root node of this document structure, node are searched through every path which has descendant element "dc:creator" and descendant element which contains character strings "XML" simultaneously. The result nodes are both 1 and 7. However, node 1 is not appropriate because there are two /books/book/dc:title paths which are not in a sibling relationship. Therefore, the subtree rooted with the node 1 is too large for the context of our intension.

Query 2b is the modification of above *Query 2a*. The value of the function **context** is a set of nodes of subtrees each of those subtrees is a maximal subtree having different path expressions each other except sibling relationship path expression including specified namespace's element type or attribute. That is node 7 in Fig 2.

Query 2b:

```
context(//*[.//[dc:creator] and
    .//[dc:title contains ''XML'']])
= context( { /books, /books/book[0] })
= {/books/book[0] }
```

In this section, We have introduced both context of simple XML queries and context of conjunctive queries.

4 Conclusions

In this paper, we have articulated the problems of queries with namespaces and introduced a function **context** to retrieve context nodes using namespaces.

The followings are the future work:

- Developing algorithm of fully–operational function **context**;
- Designing indices suitable for XML documents with namespaces. Especially, for the purpose of bottom-up search in the tree; and
- Implementation and evaluation of this algorithm.

References

[1] Jonathan Robie, Joe Lapp, and David Schach: "XML Query Language (XQL)", http://www.w3.org/TandS/QL/QL98/pp/xql.html, Sep 1998.

[2] Jonathan Robie: "XQL (XML Query Language)", http://metalab.unc.edu/xql/xql-proposal.xml, Aug 1999.

[3] World Wide Web Consortium: "QL'98 - The Query Languages Workshop", http://www.w3.org/TandS/QL/QL98/, December 1998.

[4] World Wide Web Consortium: "Extensible Markup Language (XML) 1.0", http://www.w3.org/TR/1998/REC-xml-19980210, W3C Recommendation 10-February-1998, Feb 1998.

[5] World Wide Web Consortium: "Namespaces in XML", http://www.w3.org/TR/1999/REC-xml-names-19990114/, W3C Recommendation 14-January-1999. Jan 1999.

[6] World Wide Web Consortium: "XML Schema Part 1: Structures", http://www.w3.org/1999/05/06-xmlschema-1/, W3C Working Draft 06-May-1999, May 1999.

[7] World Wide Web Consortium: "XML Schema Part 2: Datatypes", http://www.w3.org/1999/05/06-xmlschema-2/, W3C Working Draft 06-May-1999,May 1999.