Querying XML Data by the Nested Relational Sequence Database System

Ho Lam, Lau and Wilfred Ng
Department of Computer Science
The Hong Kong University of Science and Technology
{lauhl, wilfred}@cs.ust.hk

Abstract

In this concise paper, we present the Nested Relational Sequence Model (NRSM), which is an extension of the Nested Relational Data Model in order to handle XML data. We also introduce a set of algebraic operations pertaining to the Nested Relational Sequence Model, which is an extension of the Nested Relational Data Model in order to handle XML data, to manipulate XML documents via NRS relations. We demonstrate NRS operations by examples and illustrate how XML queries can be formulated within the NRSM. We also introduce the ongoing work of translating XQuery into NRS operations.

1. Introduction

XML [12] is becoming a standard format for representing and exchanging data on the World-Wide-Web. With the growth of electronic commerce, the quantity of XML documents on the Web has rapidly increased. To cope with the large amount of XML documents, we need an XML DataBase Management System (DBMS) that is efficient enough to handle storage, management and retrieval of XML documents. This depends on establishing an effective data model in a formal manner, which serves as a foundation for the development of future XML DBMS.

We propose the Nested Relational Sequence Model (NRSM) [5, 6, 7], which is an extension of the well-established Nested Relational Data Model (NRDM) [4, 10, 11] in order to cater for the nesting structure and node ordering of XML documents. Like the NRDM, the NRSM is capable of supporting composite and multi-valued attributes, which are essential for representing hierarchically structured information such as XML data. In addition, the NRSM extends the NRDM to support ordering of XML data by allowing nested tuple sequences in a *nested sequence relation* (or a NRS relation). An important feature in our model is that XML data can be collapsed into the same node as follows: data that has the same tag label along the same path are mapped to a sequence of data nodes under the same label node in a merged XML data tree.

One benefit of viewing XML data as nested data within the NRSM is that we are able to eliminate a substantial amount of redundancy in XML documents. Another benefit is that based on the NRSM we are able to preserve the original structure of XML documents, that means, in general XML documents can be retrieved from NRS relations in our system without loss of information. Descriptive information such as "comments" and "processing instructions" can also be stored in an NRS relation. The proposed method for mapping between XML documents and NRS relations is straightforward enough to implement on top of most common DBMSs such as Oracle. Comparing with other approaches in XML modeling [2, 8, 9], our approach requires extra efforts on grouping those data that shares the same tag labels. We also find that the preparation time for loading an XML document into the NRSD System is longer than other proposed data models such as the *attribute inlining* approach proposed in [3]. In addition, we need to consume extra resources to reconstruct the original XML documents from the NRSD System.

Figure 1 presents our main idea of the mapping between an XML document and an NRS relation. We model the document as a *merged data tree* (or simply a *data tree*), where the leaf nodes are *sequences of data values,* called the *data nodes*, and the non-leaf nodes are the labels of XML tags and attributes, called the *label nodes*. Figure 2 illustrates a data tree which incorporates three types of order as follows: (1) the *data order* resulting from the sequence of data elements in a data node; (2) the *sibling order* resulting from the left to right sides for label nodes which share the same parent; and (3) the *ancestor order* resulting from different levels of the label nodes in the data tree.

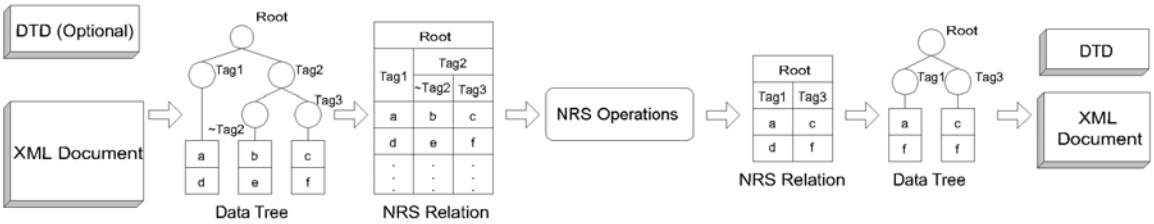

Figure 1. Mapping an XML document to an NRS relation and retrieving under NRS operations

Table 1. Brief description of NRS operations

Type	Name	Description
Nesting Operations	NEST (η)	Create a new structure of the selected attributes (tags) into a sequence of nested tuple.
	UNNEST (μ)	The reverse operation of η, which flattens an (or a subset of) NRS relation into a sequence of flat tuples.
Ordering Operations	ORDERBY (ω)	Reorder the tuple in value order by sorting the selected column(s). The default sorting direction is ascending.
	REARRANGE (κ)	Reorder all the children of the selected tags with the given arrangement.
	SWAP (χ)	Exchange the position of two columns and changing their sibling order.
	GROUPBY (φ)	Group the NRS relation by a selected tag.
Unary Operations	PROJECT (π)	Return the selected columns in the NRS relation.
	SELECE (σ)	Return a sequence of tuples from an NRS relation according to some selection conditions.
Structural Operations	NEW (⊗)	Create a new NRS relation by stating the schema of the new relation.
	INSERT (⊕)	Insert new columns/values into an NRS relation before or after a specify location. (Insert at the end by default.)
	DELETE (∅)	Delete selected columns/tuple values in the NRS relation.
Aggregate Operations	COUNT (ς)	Count the number of tuples that satisfy some conditions.
	SUM (Σ)	Return the total of the selected locations of cells, where the data in given columns is numeric.
	AVG (α)	Return the average of the selected locations of cells, where the data in given columns is numeric.
	MIN (⊤)	Return the smallest of the selected locations of cells.
	MAX (⊥)	Return the largest of the selected locations of cells.
Binary Operations	UNION (∪)	Return a sequence of tuples in either the first NRS relation or second NRS relation over the same NRS schema.
	DIFFERENCE (−)	Return a sequence of tuples in the first NRS relation but not in the second NRS relation over the same NRS schema.
	PRODUCT (×)	Concatenate the second NR relation onto the selected group of the first NRS relation.

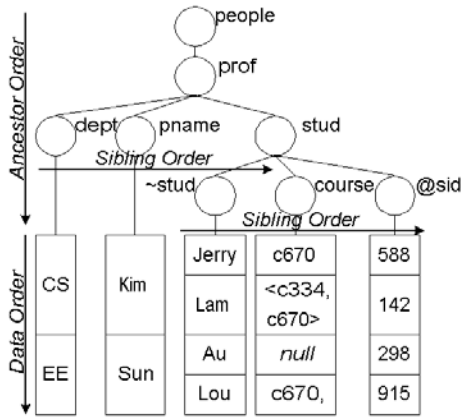

Figure 2. A data tree showing the three types of order

The XML semantics and the order of the document structure are preserved in an NRS relation and the mapping between a data tree and an NRS relation is reversible. After mapping XML documents into NRS relations, we are able to apply a sequence of NRS operations on the relations to formulate useful queries. A brief description of the NRS operations is shown in Table 1. The output of a sequence of NRS operations is always an "order-preserving" NRS relation, which preserves three types of orderings of XML data: data, sibling and ancestor orders. An NRS relation can be naturally converted back to XML data. Our suggested method of conversion is also applicable to usual flat relations, since they can be regarded as a special case of NRS relations having an imposed sibling order on tuples.

2. Related Work

The *Nested Relational Data Model* (NRDM) is one of the NF^2 relational data models proposed in 1979. In the NRDM, data is modelled as a collection of tables, named *Nested Tables* (NTs), which can have nested columns. Under the NRDM, a variety of form-based documents can be formally and uniformly handled as NTs within database systems.

The closest research with the NRSM is the QSByE [1]. QSByE represents semistructured data using nested tables with structural variants, they objective is to reduce the complexity in typical query languages for semistructured data. It allows distinct rows to contain object with distinct structures. Similar to our approach, it extends the nested relational operators to implement a QBE-like query language. Silva represents semistructured data as an OEM tree, each tag is represent as a distinct subtree and each subtree is composed of distinct atomic components. However, if the document contains many tags with the same label, the OEM tree grows rapidly. Moreover, the orders of semistructured data are not discussed.

In our approach, the tree model is adopted because we recognize that the structure of XML documents are essentially an ordered hierarchical tree and transforming XML documents into trees is natural and straightforward. We observe that if information with the same tag is grouped together and the values are stored under the same node, we can save much space and remove the unwanted redundancy. Therefore, we propose a tree representation called the *merged data tree* (or simply *data tree*), in which XML data has the same label along the same path is collapsed into the same node and the XML data are mapped to a sequence of data nodes as shown in Figure 2.

3. NRS Operations and Running Examples

We define a set of algebraic operations on the NRSM, which is employed to formulate a query over an NRS relation. These operations enable users to retrieve XML information from NRS relations. The output result of these operations is an NRS relation, taking one or more NRS relations as input. These operations have been briefly introduced in Table 1. We now demonstrate some examples to show how to perform queries by using NRS operations. The first example highlights the fact that the most common kind of queries can be expressed as NRS operations in the NRSM. In this section, the example queries are operating the NRS relaton *R* as shown in Figure 3, which is built from the XML document "sample.xml" as shown in Figure 4.

.

R

people				
prof+				
^(dept, pname)		stud*		
dept	**pname**	**~stud**	**course***	**@sid**
CS	Kim	Jerry	c670	588
		Lam	<c334, c670>	142
EE	Sun	Au	-	298
		Lou	c670	915
		Ray	<c630, c334>	611

Figure 3. The corresponding NRS Relation *R* of sample.xml

```
<people>
    <prof>
        <dept>CS</dept>
        <pname>Kim</pname>
        <stud sid="588">Jerry
            <course>c670</course>
        </stud>
        <stud sid="142">Lam
            <course>c334</course>
            <course>c670</course>
        </stud>
    </prof>
    <prof>
        <dept>EE</dept>
        <pname>Sun</pname>
        <stud sid="298">Au
        </stud>
        <stud sid="915">Lou
            <course>c670</course>
        </stud>
        <stud sid="611">Ray
            <course>c630</course>
            <course>c334</course>
        </stud>
    </prof>
</people>
```

Figure 4. The sample XML (sample.xml)

We now illustrate NRS operations by the following queries.

(Q_1) List the names of the students who are taking the course "c670".

The query Q_1 can be expressed in XQuery [14] as follows:

```
FOR $s IN
    document("sample.xml")//people/prof/stud
WHERE $s/course = "c670"
RETURN $s/text()
```

We first transform the XML document "*sample.xml*" and its DTD into the corresponding NRS relation, *R*, which is shown in Figure 3. Then we perform the following sequence of operations:

1. $S \leftarrow \pi_{(/people/prof/stud)}(R)$
2. $T \leftarrow \sigma_{(/stud/course="c670")}(S)$
3. $R_{result} \leftarrow \pi_{(/stud/\sim stud)}(T)$

For the sake of clarity, we use the temporary NRS relations, *S* and *T*, for storing the intermediate results in each step of processing. In the first step, we project on "*people/prof/stud*" over *R*. Then we perform the selection according to the condition *"/stud/course = "c670""* over *S*. Finally, we project on "*/stud/~stud*" over *T*, which generates the required results R_{result}. The temporary NRS relations *S*, *T* and R_{result} for this query are shown in Figure 5.

stud*		
~stud	**course***	**@sid**
Jerry	c670	588
Lam	c334	142
	c670	
Au	-	298
Lou	c670	915
Ray	c630	611
	c334	

S

stud*		
~stud	**course***	**@sid**
Jerry	c670	588
Lam	c334	142
	c670	
Lou	c670	915

T

stud*
~stud
Jerry
Lam
Lou

R_{result}

Figure 5. Generated NRS relations for answering the query Q_1

Then we transform the resulting NRS relation by using its corresponding DTD. The conformed XML document is represented as follows:

```
<!ELEMENT stud(#PCDATA)>
<stud>Jerry</stud>
<stud>Lam</stud>
<stud>Lou</stud>
```

(Q_2) Create a new XML document with the format given below, where *X* is the number of professors, *Y* is the number of students in the department and *Z* is the name of the department:

```
<university>
    <dept numP="X" numS="Y"> Z</dept>
</university>
```

First, we create a new NRS relation named "*university*" with the given schema "*university (dept (~dept, @numP, @numS))*" using the operation "*New*". Then we insert the value of "*dept*" corresponding to the "*dept*" of *R*. Each "*dept*" has two attributes, "*numP*" and "*numS*". Thus, we use the aggregate function "*Count*" to find out the number of professors and students in the "*dept*". The operations are shown below and the corresponding NRS relations are shown in Figure 6.

1. $S \leftarrow New_{(university(dept(\sim dept,\ @numP,\ @numS))}$;

2. $T \leftarrow Insert_{(university/dept/\sim dept\ =\ R/people/prof/dept)}(S)$;

3. For each data value under "university/dept/":
$R_{result} \leftarrow Insert_{(university/dept/@numP\ =\ Count(/people/prof/pname)(U)),\ university/dept/@numS\ =\ Count(/people/prof/stud/\sim stud)(U))}(T)$;
where $U = \sigma_{(R/people/prof/dept=\ university/dept)}(R)$.

University		
Dept		
~dept	@numP	@numS

S

university		
dept		
~dept	@numP	@numS
CS		
EE		

T

university		
dept		
~dept	@numP	@numS
CS	1	2
EE	1	3

R_{result}

Figure 6. Generated NRS relations for the query Q_2.

The resulting NRS relation and its corresponding DTD and XML document is represented as follows:

```
<university>
  <dept numP="1" numS="2">CS</dept>
  <dept numP="1" numS="3">EE</dept>
</university>
```

4. Translating XQuery into NRS Operations

In this section, we introduce the undergoing research on translating Xquery [14] into expressions of NRS operations. We show how to translate the XQuery expression "*FOR-WHERE-RETURN*", which is the most basic and the most common form of XQuery, into sequences of NRS operations:

```
FOR $b IN path1,
    $c IN path2, …
WHERE condition1, condition2,…
RETURN
<tag1>
{
    <tag2>
        $b/path3
    </tag2>
    <tag3>
        $c/path4
    </tag3>
    …
}
</tag1>
```

Before we translate the XQuery expressions into sequences of NRS operations, we map the XML documents into NRS relations, $\{R_1, R_2, \ldots\}$. Since a query may involve several XML documents, we need to map them into NRS relations before we apply NRS operations. After the mapping, the following sequence of NRS operations is performed:

1. $S_1 \leftarrow \pi_{(path1)}(R_1), S_2 \leftarrow \pi_{(path2)}(R_2), \ldots$
2. $T_1 \leftarrow \sigma_{(condition1)}(S_i), T_2 \leftarrow \sigma_{(condition2)}(S_i), \ldots$
3. $U \leftarrow \otimes_{(tag1(tag2,\ tag3))}$
4. $R_{result} \leftarrow \oplus_{(tag2\ =\ T1/path3,tag3\ =\ T2/path4,\ \ldots)}(U)$

First, we translate the XQuery expressions "*FOR $x IN pathx*" into a sequence of project operations (π) and store the result into intermediate NRS relations S_1, S_2, …. Second, basing on the conditions stated in the XQuery expression "*WHERE condition1, condition2,…*", we perform the select operations (σ) on the involved intermediate NRS relations, S_i. For example, if "*conditionx"* in the XQuery expression is equal to *"$b/name = "abc"* ", we know that S_1 is involved and we perform the select operation on S_1. Now, we obtain the required data in the intermediate NRS relation T_1, T_2, … .The third step is to return the results in the specified format stated in the "*RETURN*" expression. To achieve this, we perform the new operation ($\otimes$) to create a new NRS relation with the schema corresponding to the one stated in the XQuery "*RETURN*" expression. Finally, we perform the insert operations ($\oplus$) to insert the corresponding results into the new NRS relation. Since the mechanism of handling the hierarchical structure of the returned format is very complex, in the current stage of our translating algorithm, we can only handle the structure of at most two nested levels.

5. Conclusions

In this paper, we have introduced the Nested Relation Sequence Model (NRSM). The NRSM is desirable for managing XML since technologies developed for the existing DBMSs can be naturally

adapted to our system. We believe that the NRSM is a simple and natural database model for handling the storage and querying for XML documents, since it allows us to address the important features of multi-valued nesting and different kinds of ordering in XML data.

We have also presented a set of NRS operators, which is a combination of a refined version of existing relation algebra for nested relations [10] and some newly defined operations such as the rearrange and the swap operations [6]. The set of NRS operators provides us a basis for manipulating NRS relations in an optimized way, in which the nesting and ordering operations are useful in transforming and restructuring the XML relations to other forms. We have also demonstrated with examples how to formulate XML queries in terms of NRS operations and showed that querying using NRS operations are compatible with XQuery. Currently, we are able to translate the XQuery of the form "FOR-WHERE-RETURN" by using the NRS operations. Finally, we have discussed the ongoing research on translating XQuery expressions into sequences of NRS operations. We are going to compare the efficiency of the NRSM with other data model and perform experiments on the efficiency of the NRS operations.

References

[1] A.S. da Silva, I.M.E. Filha, A.H.F. Laender, and D.W. Embley, "Representing and Querying Semistructured Web Data Using Nested Tables with Structural Variants", *Proceedings of ER*, 2002

[2] D. Beech, A. Malhotra and M. Rys. "A Formal Data Model and Algebra for XML", *Communication to the W3C*, 1999.

[3] D. Florescu and D. Kossmann. "A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database", *Technical Report 3680*, INRIA Rocquencourt, France, 1999.

[4] H. Kitagawa and T. L. Kunii. *The Unnormalized Relational Data Model, For Office Form Processor Design*, Springer-Verlag, 1989.

[5] H. L. Lau and W. Ng, "Querying XML Data Based on Nested Relational Sequence Model", *Proceedings of Poster Track of WWW*, Honolulu, 2002.

[6] H. L. Lau and W. Ng, "The Development of Nested Relational Sequence Model to Support XML Databases", *Proceedings of the International Conference on Information and Knowledge Engineering (IKE'02)*, Las Vegas, USA, 2002, pp. 374-380.

[7] H. L. Lau and W. Ng, *The Development of the Nested Relational Sequence Model to support XML Databases,* Master Thesis of the Department of Computer Science, HKUST, 2002.

[8] I.M.E. Filha, A.S. da Silva, A.H.F. Laender, and D.W. Embley, "Using Nested Tables for Representing and Querying Semistructured Web Data", *The Fourteenth International Conference on Advanced Information Systems Engineering* Toronto, Canada, 2002.

[9] M. Fernandez, J. Simeon and P. Wadler. "An Algebra for XML Query.", *In Foundations of Software Technology and Theoretical Computer Science, number 1974, 2000.*

[10] M. Levene. *The Nested UniversityRelation Database Model,* Springer-Verlag, 1992.

[11] P. Atzeni and V. De Antonellis. *Relational Database Model Theory,* The Benjamin / Cummings Publishing Company, INC., 1993.

[12] T. Bray, J. Paoli, C. M. Sperberg-McQueen and E. Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", *W3C Recommendation,* 2000.

[13] W. Ng. "Maintaining Consistency of Integrated XML Trees", *International Conference on Web-Age Information Management WAIM'2002*. LNCS Vol. 2419, Beijing, China, 2002, pp 145 -157.

[14] World Wide Web Consortium. "XQuery 1.0 and XPath 2.0 Data Model", In: http://www.w3.org/TR/query-datamodel/, 2002.