Querying XML Data Based on Nested Relational Sequence Model

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

ABSTRACT

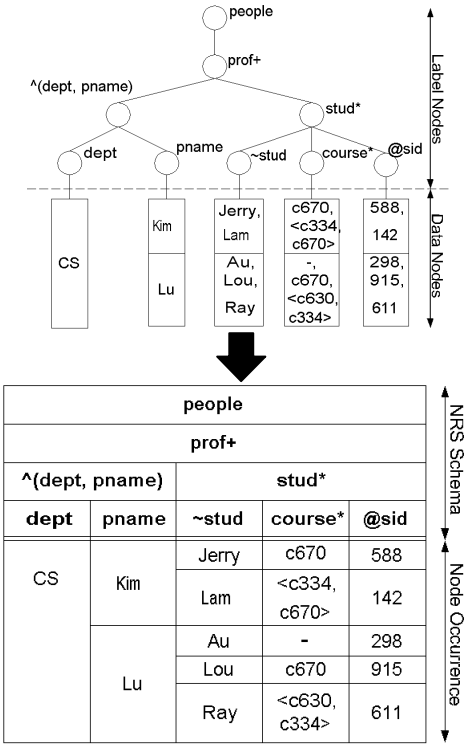
We present an XML query language based on the Nested Relational Sequence Model (the NRSM), which is an extension of the Nested Relational Data Model in order to handle XML data. The query language provides the users with a set of formal algebraic operations for manipulating NRS relations. We also introduce our Java-based implementation of a user-friendly web interface for querying XML data.

Keywords

XML Databases, Nested Relational Sequence Model, Data Trees, Query Languages.

1. INTRODUCTION

We propose the Nested Relational Sequence Model (NRSM) [6], which is an extension of the well-established Nested Relational Data Model (NRDM) [2,3,4] in order to cater for the two important features of nesting structure and node ordering in XML documents [1,5]. The NRSM supports composite and multi-valued attributes, which are essential for representing hierarchically structured objects 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 that has the same label along the same path can be collapsed into the same data node. This eliminates a substantial amount of redundancy in an XML document. A NRS relation *R* is defined by *R = (N, O, S)*, where *N* is the *NRS name*, *O* is the *NRS occurrence* and *S* is the *NRS schema*. In Figure 1 we show an example of mapping of an XML data tree into a NRS relation. Within the NRSM we define a set of algebraic operations, which is employed to formulate a query over a NRS relation that contains XML data. These operations enable users to retrieve XML information and to integrate XML documents in a systematic manner as follows: by taking one or more NRS relations as an input, a NRS query returns a NRS relation as an output result which represents an XML data tree. We summarize these operations in Table 1 given below.



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

Figure 1: Mapping an XML data tree into a NRS relation *R*.

Types	Names	Descriptions
Nested Table Operations	Nest (η)	Create a new structure of the selected attributes (tags) and the relevant data in their corresponding cells.
	UnNest (μ)	The reverse operation of η, which flattens a (or a subset of) NRS relation into a sequence of flat tuples.
Unary Operations	Projection (π)	Return the selected columns in the NRS relation.
	Selection (σ)	Return a sequence of tuples from a NRS relation according to some selection conditions.
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. The order of the first relation is maintained.
	Product (×)	Concatenate the second NRS relation onto the selected group of the first NRS relation.
Manipulating Operations	Insert /InsertData	Insert new columns/values into a NRS relation before or after a specify location. (Insert at the end by default.)
	Delete /DeleteData	Delete selected columns/tuple values in the NRS relation.
	Update /UpdateData	Update the schema/tuple values of a selected location of cells.
	New	Create a new NRS relation.
Aggregate Operations	Count	Count the number of tuples that satisfy some conditions
	Sum	Return the total of the selected locations of cells, where the data in given columns is numeric.
	Avg	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.

Table 1: A brief description of the NRS operations

2. QUERYING XML VIA NRS RELATIONS

The NRS operations described in Table 1 are as expressive as the emerged query languages for XML data [1]. We use XPath [5] to specify the location of a cell in a given NRS relation. Users are able to use a set of algebraic operations to perform queries and to transform a NRS relation into an XML view. These operations preserve the order of tuples in the original NRS relation. We now demonstrate some examples to show how to query an XML data tree via its corresponding NRS relation using NRS operations. These queries are based on the XML document and the corresponding NRS relation shown in Figure 1. We abbreviate this NRS relation as *R* in the queries Q_1 to Q_4 given below.

(Q_1) List the names of the students who are taking the course "*c670* " inside the "*people*" element.
$S \leftarrow \pi_{(people/prof/stud)}(R);\ T \leftarrow \sigma_{(/course="c670")}(S);\ R_{result} \leftarrow \pi_{(/\sim stud)}(T);$

(Q_2) Transform the NRS relation into a flat relation, list the "*dept, pname, ~stud, course* and *@sid*" inside the "*people*" element.
$S \leftarrow \mu_{(people/pof/stud)}(R);\ T \leftarrow \mu_{(people/pro/\hat{}(dept|pname))}(S);\ R_{result} \leftarrow \mu_{(people/pof)}(T);$

(Q_3) Assume we have another NRS relation R_2, which shares the same DTD as *R* and may contains duplicated tuples as *R*. List the duplicated tuples.
$S \leftarrow R - R_2;\ R_{result} \leftarrow R - S;$

(Q_4) 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 numProf="X" numStud="Y">Z</dept>
</university>
```

$S \leftarrow New_{(university)};\ R_{result} \leftarrow Insert_{(university/dept = R/people/prof/dept\ ,\ university/dept/@numProf = Count(R/people/prof/pname),\ university/dept/@numStud = Count(R/people/prof/stud/\sim stud))}(S);$

3. IMPLEMENTING THE NRSM CLIENT

In Figure 2 we show the user interface of the NRSM client. The client consists of three main components as follows: (1) a list of NRS operations is on the left column, (2) a main panel is at the central frame showing the expanding tree, and (3) text fields are at the bottom showing the selected nested tuples in the NRS relation. In the main panel there are four menu tabs, which show the XML tree and its corresponding NRS relation, the XML document, the DTD and the statistics for the querying results. There are three steps to initiate a query over the document. First, the schema for the tree is identified first and then the table displays the corresponding part of the NRS relation. Second, the sub-schema in the NRS relation is selected and then the path for the selected sub-schema is displayed in the text field. Finally, the users click on an appropriate NRS operation button in the left frame to perform queries.

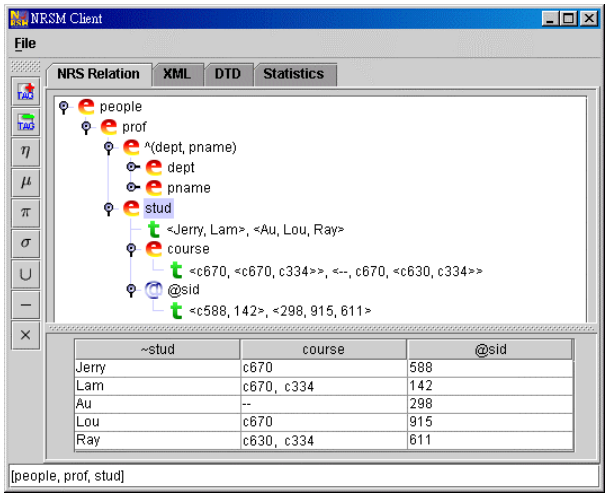


Figure 2: The interface of the NRSM client.

4. CONCLUSIONS

In this poster, we have proposed a query language based on the NRSM, which enables users to perform queries and transform the NRS relations into different XML document structures. We have also demonstrated with some examples of using NRS operations to formulate XML queries. Finally, we have introduced our on-going work related to the implementation of the NRSM client. We are further studying the algorithms on optimizing NRS queries and the strategies for converting XQuery [5] into a NRS querying expression.

REFERENCES

1. World Wide Web Consortium. *XML 1.0 specification*. In: http://www.w3.org/ TR/2000/REC-xml-20001006, (2000).
2. M. Levene. *The Nested UniversityRelation Database Model*. Springer-Verlag, (1992).
3. H. Kitagawa and T. L. Kunii. *The Unnormalized Relational Data Model*. For Office Form Processor Design. Springer-Verlag, (1989).
4. P. Atzeni and V. De Antonellis. *Relational Database Model Theory*. The Benjamin /Cummings Publishing Company, inc., (1993).
5. World Wide Web Consortium. *XQuery 1.0 and XPath 2.0 Data Model*. In: http: //www.w3.org/TR/query-datamodel, (2001).
6. H. L. Lau and W. Ng. *Storing XML Documents in Nested Relational Sequence Relations.* Research Note. The Hong Kong University of Science and Technology, (2002).