A Data Modeling Approach to the Seamless Information Exchange among Structured Documents and Databases

Atsuyuki Morishima* Hiroyuki Kitagawa†

*Doctoral Degree Program in Engineering

†Institute of Information Sciences and Electronics

June 1996

ISE-TR-96-133

Abstract

Integration of heterogeneous information resources has been one of the most important issues in recent advanced application environments. In addition to conventional databases, structured documents have been recognized as important information resources recently. In this paper, we present a data model named the *NR/SD model*, which is used as a basic data modeling framework for the seamless integration of structured documents and relational databases. The NR/SD model combines an abstract data type named the *structured document type* and the nested relational structures, and features operators named *converters* to dynamically convert structured documents into nested relational structures and vice versa. Therefore, we can manipulate information in either forms of structured documents and relations. The operators can also be used to develop user views on the stored structured documents. We show data structures and operators in the NR/SD model and its applicability to system federation environments. Some basic properties of the converters are also studied.

1 Introduction

Advanced applications today often require accesses to information not centralized in one place but scattered in a variety of physically and/or logically distributed information repositories. Thus, integration of heterogeneous information resources has been one of the hot research issues [1][2][3]. Databases and structured documents are representatives of important information resources. In addition to the conventional databases, structured documents such as those described in SGML [4] have been widely used, and have increased significance in applications such as digital libraries [5], CALS [6], WWW [7], and hypermedia descriptions [8].

The objective of this research is to provide a framework for the seamless integration of structured documents and conventional databases. In this paper, we present a data model named the *NR/SD model* as a basic data modeling framework, and discuss its applicability to the integration of structured documents and relational databases. Data modeling constructs of the NR/SD model are nested relational structures and the abstract data type concept. The nested relational structures [9][10][11] are useful in that (1) they have primitive constructs to represent logical structures embedded in structured documents, and (2) they suit well to the widely-used relational database structures as natural extensions of relations. The NR/SD model provides an abstract data type named the *structured document type* as a container to store raw structured document data. Thus, the NR/SD model provides basic constructs to model data either in structured documents and relational databases. The NR/SD model features operators, called *converters*, to dynamically convert structured documents into nested relational structures and vice versa. The converters allow us to represent the same logical structures either as instance-level structures inside structured documents or as schema-level structures of nested relations. The former representation is appropriate in manipulating a collection of heterogeneous data objects, while the latter is appropriate from the viewpoint of data restructuring and querying. The operators in the NR/SD model including the converters, can be used to manipulate data both in the structured document repositories and relational databases. In addition, we can get required data in the form of structured documents as well as relations. Today, WWW is often used to retrieve data. In such environments, the retrieval results need to be in structured documents. Furthermore, the operators in the NR/SD model can be used to develop various user views on top of the stored structured documents in analogy to views in relational databases [12].

Several approaches have been proposed to achieve the integration of structured documents and databases [13][14][15][16][17][18][19][20][21]. Although their objectives are somewhat different from one another, most of them intend to achieve the integration of structured documents and databases by introducing schema level constructs to represent data in structured documents. Atlas [14] represents document data in nested relations and provides querying facilities. Christophides and others extended the object-oriented data model of O_2 to represent SGML DTDs [15]. Yan and others [16], COINS[17], and Volz and others[18] all provide the view of structured documents within the framework of the object-oriented database schema. NST [19] is an algebra along this approach. All those approaches map structural information of structured documents into the database schema. Therefore, we have to design schema level constructs such as classes and types for each type of structured documents within the framework of the given data model in advance. Moreover, we are forced to manage documents of different types with different object types, and it is not easy to manipulate heterogeneous document collections. As aforementioned, the NR/SD model allows us to handle document structures either at the schema level or at the instance level, and these problems are largely alleviated.

T/RDBMS [20][21] combines the relational data model and the abstract data type to represent structured documents. In T/RDBMS, information inside structured documents can be viewed as a predefined collection of relations and queried in the extended SQL. However, document structures cannot be converted to the schema level constructs. Moreover, no means is provided for transforming relational structures into structured documents. This asymmetric bridge between the structured document world and the relational database world prohibits us from utilizing the full power of the relational data model for manipulating structured documents. For example, we cannot apply the relational algebra to restructure structured documents. The NR/SD model allows us to first transform structured documents into nested relational structures with the converters, then to manipulate them with the nested relational algebra operators, and finally to transform the result data into structured documents again. Also, the modeling concepts in T/RDBMS cannot be used to develop "virtual" structured documents as user views.

Yoshikawa and others proposed another approach to the integration of those two world [22]. Their approach provides a general mechanism to make reference links from components of SGML documents to database objects. Therefore, the linkage specification of structured documents and databases must be specified in advance at the instance level.

A family of region algebras were proposed to manipulate structured documents [23][24][25]. They are focussed on structured documents and do not consider conventional databases. Moreover, they are essentially index algebras to be mainly used for retrieval, so that they cannot be used for restructuring structured documents.

The rest of this paper is organized as follows. In Section 2, an example scenario is shown to explain the integration of structured documents and the relational database in the NR/SD model. Section 3 gives the data structures and operators in the NR/SD model. In Section 4, we present application of the operators to the sample scenario in Section 2 and show its applicability. In Section 5, we study some basic properties of the converters. Section 6 is the conclusion.

2 Example Scenario

In this section, we show an example integration scenario of structured documents and the relational database to illustrate situations in which the NR/SD model is used. Suppose that we have two information repositories. One is a relational database repository which manages the faculty data of some university. The other is a document repository which manages papers and books published, in the form of structured documents. To attain the integration of these two information repositories, we follow the system federation approach as in [1][26] and prepare a mediator and two wrappers as shown in Figure 1. The mediator has to do two types of job: One is to communicate with the two repositories through the wrappers, and the other is to provide the integrated view of the underlying two repositories. The wrappers retrieve the information requested by the mediator from the repositories and pass it in the requested form.

In this sample context, the NR/SD model can be used to provide the integrated schema of these two repositories. As shown in Figure 2, the integrated schema consists of three relations. The relations "Faculty" and "Department" directly represent data in the relational database. The relation "Document" has an attribute "Doc" and represents the collection of structured documents in the document repository. The domain of "Doc" is the structured document type, and each "Doc" value is a raw structured document in the document repository. As mentioned above, the document repository has two types of structured document, that is, papers and books. Therefore, the "Doc" attribute values have two different structures.

In this example, suppose we want to get a set of report structured documents, each of which contains a department name and lists the papers and books written by faculty members in the department (Figure 3). It is often the case that the result data needs to be in structured documents when retrieval is done through WWW. Each item in the list includes the title, author name, and publication year. In addition, it should include the journal name, volume, etc. for journal papers, the conference name for conference papers, and the publisher name for books. The contents of the papers and books are not included in the list.

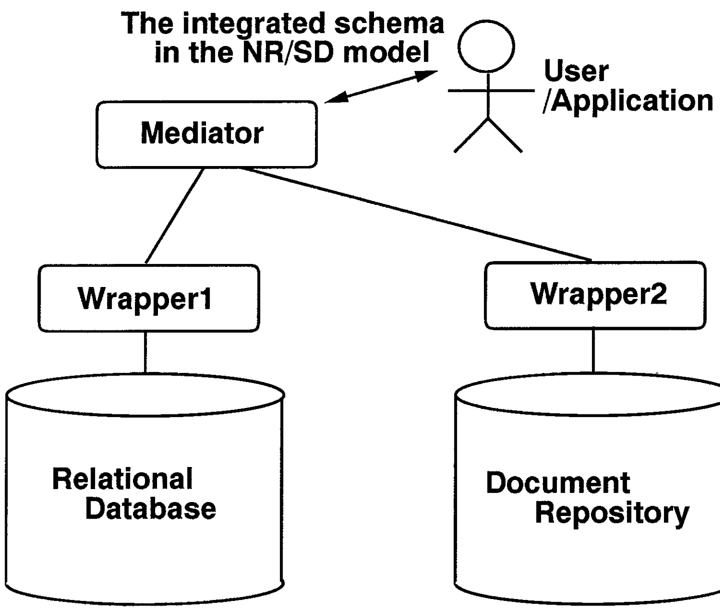

Figure 1. Example situation

Faculty					
FID	Name	D-Name	Rank	Salary	Speciality

Department	
D-Name	Head

Document
Doc

Figure 2. Integrated schema in the NR/SD model

```
<table>
<dep>Department A</dep>
<pubs>
<pub><title>``On structured documents''</title><a-name>T. Jone</a-name>
<pub-info><p-pub-info><journal><j-name>A-journal<j-name><vol>1</vol>
<no>8</no><year>1992</year></journal></p-pub-info></pub-info></pub>
<pub><title>``Relational Databases''</title><a-name>G. Mark</a-name>
<pub-info><b-pub-info><publisher>B</publisher><year>1995</year>
</b-pub-info></pub-info></pub>
<pub><title>``Access control in a multidatabase''</title><a-name>H. Smith
</a-name><pub-info><p-pub-info><proceedings><c-name>CIKM</c-name><month>May
</month><year>1992</year></proceedings></p-pub-info></pub-info></pub>
                                 ⋮
```

Figure 3. Sample report structured document

We can satisfy the requirement with the operators in the NR/SD model. Specifically, we first extract the author name and affiliation information from relation "Document" and represent it in nested relational structures with the converters. Then, we select relevant data, join them with the faculty relation data, and so on. Finally, we convert the data into structured documents again.

3 NR/SD Data model

In this section, we define the data structures and operators in the NR/SD model. As mentioned before, nested relational structures, and the abstract data type, *structured document type*, are basic data modeling constructs of the NR/SD model. In addition to the ordinal nested relational algebra operators, the NR/SD model features six *converters*, to convert structured documents into nested relational structures and vice veasa.

3.1 NR/SD Data Structures

We define the nested relational structures following the formalism of Fischer and Thomas [10]. A *relation scheme* S is a set of rules of the form $A_i = (A^i_1, \ldots, A^i_n)$. In this context, the order of $A^i_1, \ldots, A^i_n$ is significant. An example of relation scheme T is as follows.

$$T = \{A = (B, C, D), D = (E, F)\}$$

T has two rules. $A, B, \ldots, F$ are called *attributes.* We call attributes which appear on the left side of some rules, namely A and D, *higher-order attributes*, and the others, namely B, C, E, and F, *zero-order attributes.* Let E_S denote the set of attributes in S, namely $E_T = \{A, B, C, D, E, F\}$. Each attribute can appear at most once on the right side of some rule and also on the left side of another rule. S has just one *external attribute*, denoted by R_S, which appears only on the left side of some rule, namely $R_T = A$.

Instances are defined for each attribute. If A_i is a zero-order attribute, an instance of A_i is a value from the set $dom(A_i)$, called the *domain* of A_i. As defined in Subsection 3.2, $dom(A_i)$ can be the *structured document type* as well as ordinal primitive data types such as Integer and String. Values of the structured document type are called *SD values*, and values in the other domains are called *ordinal values.* If A_i is a higher-order attribute and $A_i = (A^i_1, A^i_2, \ldots, A^i_n)$, then an instance of A_i is a set of tuples such that each component of a tuple is an instance of A^i_j. Instances of higher-order attributes are called *composite values.*

The *relation* $\langle S, r\rangle$ is a pair of the relation scheme S and an instance r of R_S. Figure 4 shows relation $\langle T, r_0\rangle$ in tabular form. Here, $dom(B)$ is String, $dom(E)$ is Integer, and $dom(C)$ and $dom(F)$ are the structured document type defined in Subsection 3.2. Often we refer to the relation simply by its instance r when there is no ambiguity.

A			
B	C	D	
		E	F
abc	`<table><dep>` `Department...`	1	...
		2	...
def	...	3	...
		4	...

Figure 4. Relation $\langle T, r_0\rangle$

3.2 Structured Document Type

Here, we define the *structured document type* denoted by SD. A value of the structured document type SD is the following pair of a *DTD* (Document Type Definition) and text in which tags are embedded according to the DTD.

$$SD = \{\langle d, c\rangle | d \text{ is a } DTD \wedge c \text{ is a tagged text which conforms to } d\}$$

Figure 5 shows an example SD value. The DTD in the upper box represents the document structure. Inside the lower box is the tagged text. The DTD in the NR/SD model is similar to that in SGML, although we do not consider exception structures and recursions for simplicity. A tagged text is divided into *elements* surrounded by a *begin tag* <*gi*> and an *end tag* </*gi*>, where *gi* is a *generic identifier* representing the *element type*. Elements can be nested within other elements. For example, the tagged text in Figure 5 has element "memo," and the "memo" has "prolog," "date,"... etc. as sub-elements.

The DTD prescribes how the elements can be hierarchically constructed by sub-elements. Consider the DTD in Figure 5. Each line in the DTD is an *element type definition*. An element "memo" is a sequence of "prolog" and "body." An element "body" consists of zero or more "para" elements. An element "prolog" is again a sequence of "date," "from," "to," and "subject." An element "to" contains either "faxno" or "mailaddr."

Formally, a DTD is a set of *element type definitions*, which has one of the following forms.

- $g = \mathbf{seq}(g_1, \ldots, g_n)$
- $g = \mathbf{rep}(g_1)$
- $g = \mathbf{or}(\{g_1, \ldots, g_n\})$, if $j \neq i$ then $g_i \neq g_j$.
- $g = \mathbf{ptext}$

Here, g is a generic identifier of the defined element type, and g_i is the generic identifier of its sub-element type. Element type definitions of the form $g = \mathbf{ptext}$ can be omitted for concise presentation. In Figure 5, element type definitions such as "para=**ptext**" are omitted. Element types defined in a DTD must form a rooted DAG structure. The DTD in Figure 5 forms the rooted DAG (tree in this case) shown in Figure 6. Each node in the DAG represents an element type and has a generic identifier g and the structure specification (**seq**, **rep**, **or**, or **ptext**). We call the generic identifier of the root node, the *root generic identifier*. The root generic identifier of a DTD d is denoted by $root(d)$. If d stands for the DTD in Figure 5, $root(d) =$ "memo."

$$\begin{array}{lcl}
\text{memo} & = & \mathbf{seq}(\text{prolog, body}) \\
\text{body} & = & \mathbf{rep}(\text{para}) \\
\text{prolog} & = & \mathbf{seq}(\text{date, from, to, subject}) \\
\text{to} & = & \mathbf{or}(\{\text{faxno, mailaddr}\})
\end{array}$$

```
<memo>
<prolog>
<date>March 27, 1996</date>
<from>A. Morishima</from>
<to><faxno>XX-XXXX</faxno></to>
<subject>An example</subject>
</prolog>
<body>
<para>This is a value of structured document
 type. ...</para>
<para>...</para>
</body>
</memo>
```

Figure 5. SD value

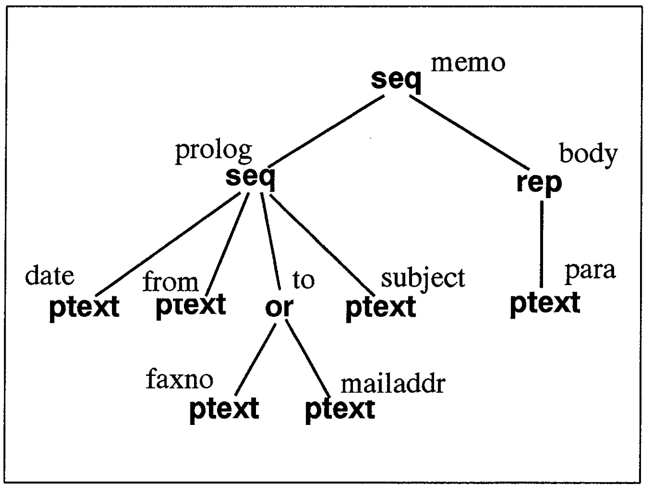

Figure 6. The DAG representation of DTD

For convenience in the later discussion, we introduce the linear notation of a DTD. For a DTD d with $root(d) = g$, the linear notation $LN(g)$ is derived as follows.

1. If $g = \textbf{ptext} \in d$, then $LN(g)$ is g:**ptext**.
2. If $g = \textbf{seq}(g_1, \ldots, g_n) \in d$, then $LN(g) = g : \textbf{seq}(LN(g_1), \ldots, LN(g_n))$.
3. If $g = \textbf{rep}(g_1) \in d$, then $LN(g) = g : \textbf{rep}(LN(g_1))$.

4. If $g = \mathbf{or}(g_1, \ldots, g_n) \in d$, then $LN(g) = g : \mathbf{or}(g_1, \ldots, LN(g_n))$.

The DTD in Figure 5 is represented below in the linear notation.

memo:**seq**(prolog:**seq**(date:**ptext**, from:**ptext**, to:**or**({faxno:**ptext**, mailaddr:**ptext**}), subject:**ptext**), body:**rep**(para:**ptext**))

3.3 Converters

The NR/SD model provides the six operators: *Rep-unpack*, *Seq-unpack*, *Rep-pack*, *Seq-pack*, *Or-append*, and *Or-remove*, which are generically called *converters*. The converters transform SD values into nested relational structures and vice versa. XX-unpacks and XX-packs reorganize nested structures of relations. XX-upacks extract the top-level structures embedded in SD values and represent them in the nested relational structures. XX-packs attain the conversions in the opposite direction. Or-append and Or-remove are mainly used to make some preparations for XX-unpacks and XX-packs.

Rep-unpack

Rep-unpack (**RU**) takes a relation containing SD values, and extracts the repetition (**rep**) structures at their roots. For example, relation r_1 (Figure 7) is transformed into relation r_2 (Figure 8) by the following Rep-unpack:

$$r_2 := \mathbf{RU}_{B:(C,D),E}(r_1).$$

R	
A	B
1	⟨ a:**rep**(b:**or**({c:**seq**(d:**text**,e:**ptext**),f:**ptext**})) , "<a><b><c><d>T1</d><e>T2</e></c></b> <b><f>T3</f></b> <b><c><d>T4</d><e>T5</e></c></b></a>"⟩
2	⟨ g:**rep**(h:**seq**(i:**text**,j:**ptext**,k:**ptext**)) , "<g><h><i>T6</i><j>T7</j><k>T8</k></h> <h><i>T9</i><j>T10</j><k>T11</k></h></g>"⟩

Figure 7. Relation r_1

R			
A	B		E
	C	D	
1	1	⟨ b:**or**({c:**seq**(d:**text**,e:**ptext**),f:**ptext**}) , "<b><c><d>T1</d><e>T2</e></c></b>"⟩	a
	2	⟨ b:**or**({c:**seq**(d:**text**,e:**ptext**),f:**ptext**}) , "<b><f>T3</f></b>"⟩	
	3	⟨ b:**or**({c:**seq**(d:**text**,e:**ptext**),f:**ptext**}) , "<b><c><d>T4</d><e>T5</e></c></b>"⟩	
2	1	⟨ h:**seq**(i:**text**,j:**ptext**,k:**ptext**) , "<h><i>T6</i><j>T7</j><k>T8</k></h>"⟩	g
	2	⟨ h:**seq**(i:**text**,j:**ptext**,k:**ptext**) , "<h><i>T9</i><j>T10</j><k>T11</k></h>"⟩	

Figure 8. Relation r_2

Definition 1. Let $\langle S, r\rangle$ be a relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m)$, $dom(A_i) = SD$ for some $1 \leq i \leq m$, and $\forall t \in r \exists g, d, c(t[A_i] = \langle g : \mathbf{rep}(d), c\rangle)$. Then, $\mathbf{RU}\ _{A_i:(O,B),G}(\langle S, r\rangle) = \langle S', r'\rangle$, where B, O and G are new attributes,

$$\begin{aligned} S' &= (S - \{R_S = (A_1, \ldots, A_m)\}) \cup \{R_S = (A_1, \ldots, A_m, G), A_i = (O, B)\}, \\ r' &= \{t | \exists u \in r, \exists g, d, c, n(u[A_i] = \langle g : \mathbf{rep}(d), c\rangle \wedge n = \#sub\text{-}el(c) \\ &\qquad \wedge t = u \text{ except } t[G] = g \text{ and } t[A_i] = \{(1, \langle d, sub\text{-}el(c, 1)\rangle), \ldots, (n, \langle d, sub\text{-}el(c, n)\rangle)\})\}, \end{aligned}$$

#sub-el(c) is the number of the direct sub-elements of c, and *sub-el(c, i)* is the i-th direct sub-element of c. □

Seq-unpack

Seq-unpack (**SU**) takes a relation containing SD values and extracts the sequence (**seq**) structures at their roots. For example, relation r_3 (Figure 9) is transformed into relation r_4 (Figure 10) by the following Seq-unpack:

$$r_4 := \mathbf{SU}_{B=(C,D),E}(r_3).$$

R	
A	B
1	⟨ a:**seq**(b:**rep**(c:**ptext**),d:**seq**(e:**ptext**,f:**ptext**)) , “<a><b><c>T1</c><c>T2</c></b><d><e>T3</e><f>T4</f></d></a>”⟩
2	⟨ g:**seq**(h:**or**({i:**ptext**,j:**ptext**}),k:**rep**(l:**ptext**)) , “<g><h><j>T5</j></h><k><l>T6</l><l>T7</l></k></g>”⟩

Figure 9. Relation r_3

R			
A	C	D	E
1	⟨ b:**rep**(c:**ptext**) , “<b><c>T1</c><c>T2</c></b>”⟩	⟨ d:**seq**(e:**ptext**,f:**ptext**) , “<d><e>T3</e><f>T4</f></d>”⟩	a
2	⟨ h:**or**({i:**ptext**,j:**ptext**}) , “<h><j>T5</j></h>”⟩	⟨ k:**rep**(l:**ptext**) , “<k><l>T6</l><l>T7</l></k>”⟩	g

Figure 10. Relation r_4

Definition 2. Let $\langle S, r\rangle$ be relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m)$, $dom(A_i) = SD$ for some $1 \leq i \leq m$, and $\exists k \forall t \in r \exists g, d_1, \ldots, d_k, c(t[A_i] = \langle g : \mathbf{seq}(d_1, \ldots, d_k), c\rangle)$. Then, $\mathbf{SU}_{A_i=(B_1,\ldots,B_k),G}(\langle S, r\rangle) = \langle S', r'\rangle$, where $B_1, \ldots, B_k$ and G are new attributes,

$$S' = (S - \{R_S = (A_1, \ldots, A_m)\}) \cup \{R_S = (A_1, \ldots, A_{i-1}, B_1, \ldots, B_k, A_{i+1}, \ldots, A_m, G)\},$$

and

$$\begin{aligned} r' = \{t | \exists u \in r, \exists g, d_1, \ldots, d_k, c(u[A_i] = \langle g : \mathbf{seq}(d_1, \ldots, d_k), c\rangle \\ \wedge t[G] = g \wedge t[A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m] = u[A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m] \\ \wedge 1 \leq \forall l \leq k(t[B_l] = \langle d_l, sub\text{-}el(c, l)\rangle))\}. \end{aligned}$$ □

Rep-pack

Rep-pack (**RP**) takes a relation containing SD values, and embeds sub-relation structures into SD values as repetition (**rep**) structures. For example, relation r_2 (Figure 8) is transformed into relation r_1 (Figure 7) by the following Rep-pack:

$$r_1 := \mathbf{RP}_B(r_2).$$

Definition 3. Let $\langle S, r\rangle$ be a relation. Assume that S has the rules $R_S = (A_1, \ldots, A_m, G)$ and $A_i = (O, B)$, $dom(B) = SD$, $\leq$ is a total order relation over $dom(O)$, and $\forall t \in r \exists d \forall v \in t[A_i] \exists c (v[B] = \langle d, c\rangle)$. Then, $\mathbf{RP}_{A_i}(\langle S, r\rangle) = \langle S', r'\rangle$, where

$$\begin{aligned}
S' &= (S - \{R_S = (A_1, \ldots, A_m, G), A_i = (O, B)\}) \cup \{R_S = (A_1, \ldots, A_m)\}, \\
r' &= \{t | \exists u \in r, \exists g, d, i_1, \ldots, i_n, c_1, \ldots, c_n \\
&\qquad (u[G] = g \wedge u[A_i] = \{(i_1, \langle d, c_1\rangle), \ldots, (i_n, \langle d, c_n\rangle)\} \wedge i_1 \leq \ldots \leq i_n \\
&\qquad \wedge t = u \text{ except } t[A_i] = \langle g : \mathbf{rep}(d), add\_tag(concat(c_1, \ldots, c_n), g)\rangle)\},
\end{aligned}$$

$concat(c_1, \ldots, c_n)$ is the concatenation of the tagged texts $c_1, \ldots, c_n$, and add_tag([tagged_text], g) is the tagged text "<g>[tagged_text]</g>." For example, add_tag("T1", a) is "<a>T1</a>."
□

In the above definition, the generic identifier g is given by the corresponding value of attribute G. We can explicitly specify the generic identifier g as a parameter instead of specifying attribute G. In this case, the expression would be $\mathbf{RP}_{B,g}(r_2)$, where g is a given generic identifier. We omit the formal definition of this version of Rep_pack.

Seq-pack

Seq-pack (**SP**) takes a relation containing SD values, and embeds attribute sub-sequences into SD values as sequence (**seq**) structures. For example, relation r_4 (Figure 10) is transformed into relation r_3 (Figure 9) by the following Seq-pack:

$$r_3 := \mathbf{SP}_{B=(C,D)}(r_4).$$

Definition 4. Let $\langle S, r\rangle$ be a relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m, G)$, and $dom(A_i) = SD, \ldots, dom(A_j) = SD$ for some $1 \leq i \leq j \leq m$. Then, $\mathbf{SP}_{B=(A_i,\ldots,A_j)}(\langle S, r\rangle) = \langle S', r'\rangle$, where B is a new attribute,

$$S' = (S - \{R_S = (A_1, \ldots, A_m, G)\}) \cup \{R_S = (A_1, \ldots, A_{i-1}, B, A_{j+1}, \ldots, A_m)\},$$

and

$$\begin{aligned}
r' = \{t | \exists u \in r, \exists g, d_i, \ldots, d_j, c_i, \ldots, c_j (u[G] = g \wedge u[X_i] = \langle d_i, c_i\rangle \wedge, \ldots, \wedge u[X_j] = \langle d_j, c_j\rangle \\
\wedge t[A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m] = u[A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_m] \\
\wedge t[B] = \langle g : \mathbf{seq}(d_i, \ldots, d_j), add\_tag(concat(c_i, \ldots, c_j), g)\rangle)\}.
\end{aligned}$$
□

As in Rep-pack, the generic identifier g can be explicitly specified as a parameter to Seq-pack.

Or-remove

Or-remove (**OR**) takes a relation containing SD values, and removes the top "or" (**or**) structures at their roots. This operator can be used to prepare for further applications of Rep-unpack and Seq-unpack operators, which require the root structure of target SD values to be repetition (**rep**) or sequence (**seq**). For example, the following Or-remove transforms relation r_5 (Figure 11) into relation r_6 (Figure 12), to which we can apply Seq-unpack:

$$r_6 := \mathbf{OR}_{B,C}(r_5).$$

R	
A	B
1	⟨ a:**or**({b:**seq**(c:**rep**(d:**text**),e:**ptext**),f:**seq**(g:**ptext**,h:**ptext**)}) , "`<a><b><c>T1</c><c>T2</c></b><d>T2</d></b></a>`"⟩
2	⟨ i:**or**({f:**seq**(g:**text**,h:**ptext**),j:**seq**(k:**ptext**,l:**or**({m:**ptext**,n:**ptext**}))}) , "`<i><j><k>T3</k><l><m>T4</m></l></j></i>`"⟩
3	⟨ o:**or**({p:**seq**(q:**text**,r:**ptext**),f:**seq**(g:**ptext**,h:**ptext**)}) , "`<o><p><q>T5</q><r>T6</r></p></o>`"⟩

Figure 11. Relation r_5

R		
A	B	C
1	⟨ b:**seq**(c:**rep**(d:**text**),e:**ptext**) , "`<b><c>T1</c><c>T2</c></b><d>T2</d></b>`"⟩	a
2	⟨ j:**seq**(k:**ptext**,l:**or**({m:**ptext**,n:**ptext**})) , "`<j><k>T3</k><l><m>T4</m></l></j>`"⟩	i
3	⟨ p:**seq**(q:**text**,r:**ptext**) , "`<p><q>T5</q><r>T6</r></p>`"⟩	o

Figure 12. Relation r_6

Definition 5. Let $\langle S, r\rangle$ be a relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m)$, $dom(A_i) = SD$ for some $1 \le i \le m$, and $\forall t \in r \exists g, d_1, \ldots, d_k, c(t[A_i] = \langle g : \mathbf{or}(\{d_1, \ldots, d_k\}), c\rangle)$. Then, $\mathbf{OR}_{A_i,G}(\langle S, r\rangle) = \langle S', r'\rangle$, where G is a new attribute,

$$S' = (S - \{R_S = (A_1, \ldots, A_m)\}) \cup \{R_S = (A_1, \ldots, A_m, G)\},$$

and

$$\begin{aligned} r' = \{t | \exists u \in r, \exists g, g_1, \ldots, g_k, d_1, \ldots, d_k, c(\\ u[A_i] = \langle g : \mathbf{or}(\{g_1 : d_1, \ldots, g_k : d_k\}), c\rangle \wedge c = \text{"<}g\text{><}g_i\text{>...</}g_i\text{></}g\text{>"} \\ \wedge t = u \text{ except } t[G] = g \text{ and } t[A_i] = \langle g_i : d_i, \textit{sub-el}(c, 1)\rangle)\}. \end{aligned} \quad \square$$

Or-append

Or-append (**OA**) takes a relation containing SD values, and add the "or" (**or**) structures at their roots. This operator can be used to prepare for Rep-pack operator, which requires the target SD value set to have the same element type at their roots. For example, we cannot directly apply Rep-pack to relation r_7 (Figure 13). However, the following Or-append yields relation r_8 (Figure 14), to which we can apply Rep-pack:

$$r_8 := \mathbf{OA}_D(r_7).$$

R			
A	B		E
	C	D	
1	1	⟨ c:**seq**(d:**text**,e:**ptext**) , “<c><d>T1</d><e>T2</e></c>”⟩	b
	2	⟨ f:**or**({g:**ptext**,h:**ptext**}) , “<f><g>T3</g></f>”⟩	
	3	⟨ c:**seq**(d:**text**,e:**ptext**) , “<c><d>T4</d><e>T5</e></c>”⟩	
2	1	⟨ j:**rep**(k:**ptext**) , “<j><k>T6</k><k>T7</k></j>”⟩	i
	2	⟨ l:**seq**(m:**text**,n:**ptext**) , “<l><m>T8</m><n>T9</n></l>”⟩	

Figure 13. Relation r_7

R		
A	B	
	C	D
1	1	⟨ b:**or**({c:**seq**(d:**text**,e:**ptext**),f:**or**({g:**ptext**,h:**ptext**})}) , “<b><c><d>T1</d><e>T2</e></c></b>”⟩
	2	⟨ b:**or**({c:**seq**(d:**text**,e:**ptext**),f:**or**({g:**ptext**,h:**ptext**})}) , “<b><f><g>T3</g></f></b>”⟩
	3	⟨ b:**or**({c:**seq**(d:**text**,e:**ptext**),f:**or**({g:**ptext**,h:**ptext**})}) , “<b><c><d>T4</d><e>T5</e></c></b>”⟩
2	1	⟨ i:**or**({j:**rep**(k:**ptext**),l:**seq**(m:**text**,n:**ptext**)}) , “<i><j><k>T6</k><k>T7</k></j></i>”⟩
	2	⟨ i:**or**({j:**rep**(k:**ptext**),l:**seq**(m:**text**,n:**ptext**)}) , “<i><l><m>T8</m><n>T9</n></l></i>”⟩

Figure 14. Relation r_8

Definition 6. Let $\langle S, r\rangle$ be a relation. Assume that S has the rules $R_S = (A_1, \ldots, A_m, G)$ and $A_i = (B_1, \ldots, B_n)$, and $dom(B_j) = SD$ for some $1 \leq j \leq n$. Then, $\mathbf{OA}_{B_j}(\langle S, r\rangle) = \langle S', r'\rangle$, where

$$S' \;=\; (S - \{R_S = (A_1, \ldots, A_m, G)\}) \cup \{R_S = (A_1, \ldots, A_m)\}$$

and

$$\begin{aligned} r' \;=\; & \{t \mid \exists u \in r, \exists g, d(u[G] = g \wedge d = g : \mathbf{or}(\{d' \mid \exists w \in u[A_i], \exists c(w[B_j] = \langle d', c\rangle)\}) \\ & \qquad \wedge t = u \text{ except } t[A_i] = \{v \mid \exists w \in u[A_i], \exists d', c(w[B_j] = \langle d', c\rangle \\ & \qquad\qquad \wedge v = w \text{ except } v[B_j] = \langle d, add\_tag(c, g)\rangle)\}\}\}. \end{aligned}$$ □

As in Rep-pack, the generic identifier g can be explicitly specified as a parameter to Or-append.

3.4 NR/SD Algebra

The *NR/SD algebra* consists of the six converters and the operators explained here. The operators other than the converters fall into two groups. The first group includes counterparts of ordinal nested relational algebra operators, such as Selection, Projection, Nest, Unnest (Figure 15) [10]. The second group includes composite operators to enable concise expressions. Here, we briefly describe these operators. We also explain how to translate the structured document type into ordinal types and vice veasa.

Nested Relational Algebra Operators

Operators in Figure 15 are also operators of the NR/SD algebra. Selection takes selection condition p for selecting tuples. In the NR/SD model, applicability of some converters depends on the DTD structures of SD values. For this reason, Selection here is extended to be able to select tuples based on equality of DTDs of SD values in tuples. For example, Selection $\sigma_{\mathrm{DTD}(B)=a:\mathbf{or}(\{b:\mathbf{ptext},c:\mathbf{ptext}\})}(r_1)$ selects tuples whose DTDs of attribute B ($dom(B) = SD$) values are $a : \mathbf{or}(\{b : \mathbf{ptext}, c : \mathbf{ptext}\})$.

As mentioned in Subsection 3.1, the order of attributes is significant in the NR/SD model. The main reason is that sequence structures in structured documents are converted into attribute sequences and vice versa by Seq-Unpack and Seq-Pack. Therefore, $\pi_{A,B}(r) \neq \pi_{B,A}(r)$ and $r_1 \times r_2 \neq r_2 \times r_1$.

Selection	$\sigma_p(r)$
Projection	$\pi_{A_{i1},\ldots,A_{im}}(r)$
Cartesian product	$r_1 \times r_2$
Nest	$\nu_{A=(B_1,\ldots,B_m)}(r)$
Unnest	$\mu_A(r)$
Union	$r_1 \cup r_2$
Difference	$r_1 - r_2$

Figure 15. Nested relational algebra operators

Composite Operators

The operators we have defined above are not sometimes convenient in formulating practical queries concisely, since they can only be applicable to the outermost structures of relations. We can define composite operators which can directly manipulate internal attributes inside relations. For this purpose, given a relation $\langle S, r\rangle$, we define *Tag* operator τ_{A_i} for Unnest $\mu_{A_i}(r)$ as $\tau_{A_i}(r) = \pi_{Attributes,A_i}(r \times r)$, where *Attributes* stands for the children of the external attribute R_S. Namely, Tag operation makes a copy of the target attribute A_i of Unnest $\mu_{A_i}(r)$ to assure the reversibility of Unnest [27]. Then, for example, we define the composite operator **RU*** for Rep-unpack **RU** as follows:

$$\mathbf{RU*}_{A:(B,C),G}(r) = \Psi(\mathbf{RU}_{A:(B,C),G}(\Phi(r))),$$

where $\Phi(r)$ denotes a minimum sequence of pairs of Tag and Unnest such that $\mathbf{RU}_{A:(B,C),G}(\Phi(r))$ is well-defined, and $\Psi(r)$ is a sequence of pairs of Nest and Projection which recovers the original nesting of attributes as the inverse of Φ. Similar composite operators can be defined for any unary operators we have defined.

Translation between the Structured Document Type and Ordinal Types

In the manipulation of data in structured documents and relational databases, it is sometimes necessary to translate ordinal values such as integers and strings into primitive SD values (namely, SD values which have only one start tag and end tag.) and vice versa. For example, we need to translate a string "abc" into an SD value $\langle g : \textbf{ptext}, \text{"<}g\text{>abc</}g\text{>"}\rangle$ and an integer 1 into $\langle g : \textbf{ptext}, \text{"<}g\text{>1</}g\text{>"}\rangle$. The *domain translator* (**DT**) perform such type conversion. Formally, let $\langle S, r\rangle$ a relation, and assume A_i is a zero-order attribute in S. Then, $\textbf{DT}_{A_i,T}(r)$ changes $dom(A_i)$ into T. When $T = SD$, we also have to specify a generic identifier g, for example $\textbf{DT}_{A_i,SD(g)}(r)$, to be used in start and end tags.

4 Application

In this section, we show sample uses of the NR/SD algebra in the context of the scenario explained in Section 2. Two examples are presented here. The first example shows data retrieval from the document repository and the result is given as a relation. The second one is the data retrieval discussed in Section 2. This example shows how the NR/SD algebra is used to manipulate collections of heterogeneous data and to put the retrieval result into structured documents.

As mentioned in Section 2, the integrated schema consists of three relations "Faculty," "Department," and "Document." We assume each structured document in "Document" is based on DTD1 or DTD2, whose DAG structures are shown in Figure 16.

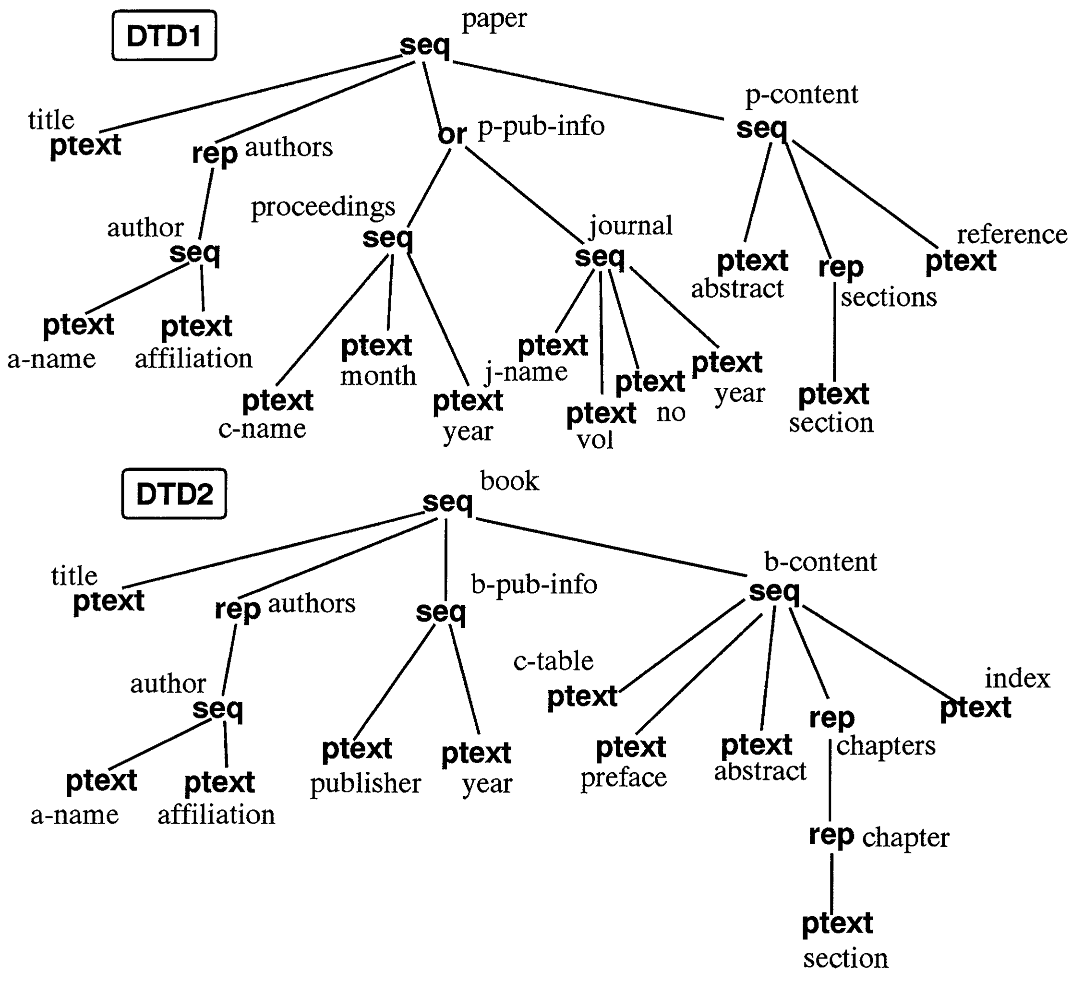

Figure 16. DTDs in relation "Document"

Example 1 : Retrieve information on books from the document repository which were published in 1990. The result should be given as a relation with the schema { R=(Title, Author, Affiliation, Publisher)}.

First, we extract the root sequence (**seq**) structures in structured documents in relation "Document," and get the corresponding attribute sequence with Seq-unpack. Then, we can select only tuples whose root generic identifiers are "book" as follows. Figure 17 illustrates the intermediate relation r_9.

$$r_9 \;:=\; \sigma_{G_1=book}(\mathbf{SU}_{Doc=(Title,Authors,Pub\text{-}Info,Content),G_1}(Document))$$

Document				
Title	Authors	Pub-Info	Content	G_1
				book

Figure 17. Relation r_9

Next, we extract repetition (**rep**) and sequence (**seq**) structures at the root of SD values in attributes "Authors" and "Pub-Info" one after another with Rep-unpack and Seq-unpack. Finally, we select tuples whose value of attribute "Year" is 1990 and which have the first author name, and get the final result with Projection.

$$\begin{aligned} r_{10} \;&:=\; \mathbf{SU}_{Author=(A\text{-}Name,Affiliation),G_4}(\mu_{Authors}(\mathbf{RU}_{Authors:(O_1,Author),G_3}(\\ &\qquad\qquad \mathbf{SU}_{Pub\text{-}Info=(Publisher,Year),G_2}(r_9)))) \\ r_{11} \;&:=\; \pi_{Title,Author,Affiliation,Publisher}(\sigma_{Year=1990\wedge O_1=1}(\mathbf{DT}_{Year,Integer}(r_{10}))) \end{aligned}$$

Example 2 : Retrieve information on papers and books written by faculty members in "University A." The result should be grouped by department and in the form of structured document whose DTD is shown in Figure 18. We assume that the relational database contains the data of "University A."

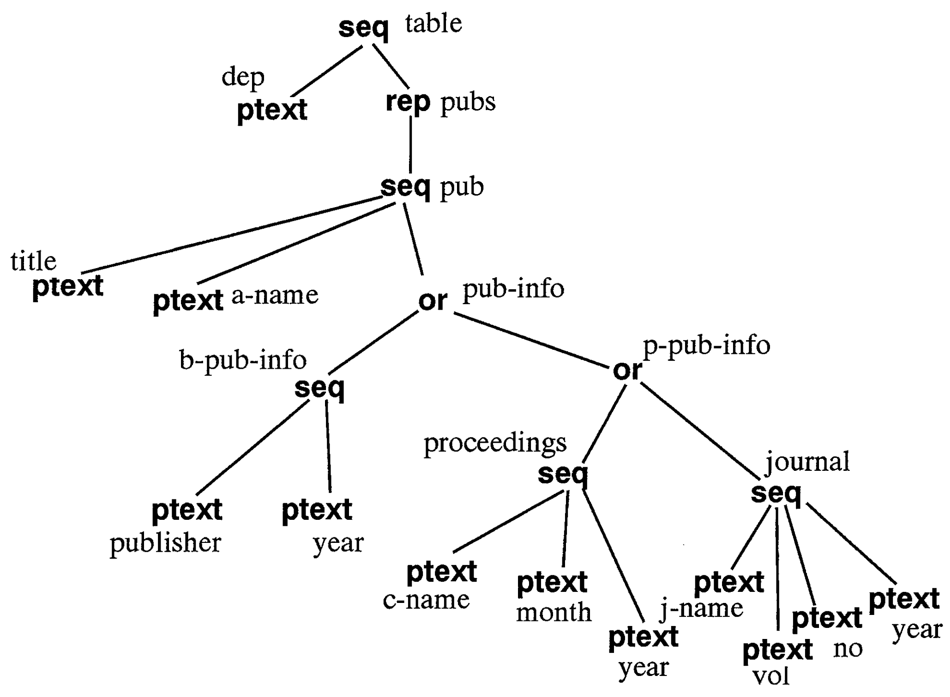

Figure 18. DTD in Example 2

The operation for the this example is divided into three steps: (1) transformation of structured documents into nested relational structures with the converters, (2) manipulation with the nested relational algebra operators, and (3) transformation of the intermediate data into structured documents again with the converters.

[Step 1]

Structures of SD values in relation "Document" is extracted, and "Document" is transformed into relation r_{12} (Figure 19) as follows:

$$r_{12} \;:=\; \mathbf{SU}_{Author=(A\text{-}Name,Affiliation),G_3}(\mu_{Authors}(\mathbf{RU}_{Authors:(O_1,Author),G_2}(\\ \mathbf{SU}_{Doc=(Title,Authors,Pub\text{-}Info,Content),G_1}(Document)))).$$

Document								
Title	O_1	A-Name	Affiliation	Pub-Info	Content	G_1	G_2	G_3
							authors	author

Figure 19. Relation r_{12}

[Step 2]

First, we select only tuples whose "Affiliation" value is "A-univ" from relation r_{12}, and joins the intermediate relation with relations "Faculty" and "Department." (Join is represented as a combination of Selection and Cartesian product below.) Then, we project out unnecessary attributes and apply Nest to group publications by the department (D-Name). The result at the end of Step 2 is shown in Figure 20.

$$\begin{aligned} r_{13} &:= \sigma_{A\text{-}Name=Name}(\mathbf{DT}_{Name,SD(a\text{-}name)}(\sigma_{Affiliation=\text{“A-univ”}}(\\ &\qquad \mathbf{DT}_{Affiliation,String}(r_{12})) \times \sigma_{D\text{-}Name=D\text{-}Name}(Faculty \times Department))) \\ r_{14} &:= \nu_{Publications=(Name,Title,A\text{-}Name,Pub\text{-}Info)}(\\ &\qquad \pi_{D\text{-}Name,Name,Title,A\text{-}Name,Pub\text{-}Info}(r_{13})) \end{aligned}$$

T				
D-Name	Publications			
	Name	Title	A-Name	Pub-Info

Figure 20. Relation r_{14}

[Step 3]

Step 3 transforms r_{14} into structured documents. Figure 21 shows the result relation r_{15}. Note that the SD values in the attribute “Pub-Info” of r_{14} have two kinds of DTD for papers and for books. Therefore, we need Or-append (**OA**) before Rep-pack (**RP**) on the attribute “Publications,” to unify the two DTDs. “Name” is used in Rep-pack to specify the order of elements in the repetition (**rep**) structure.

$$\begin{aligned} r_{15} &:= \mathbf{SP}_{Table=(D\text{-}Name,Publications),table}(\mathbf{RP}_{Publications,pubs}(\\ &\quad \mathbf{SP*}_{Pub=(Title,A\text{-}Name,Pub\text{-}Info),pub}(\mathbf{OA}_{Pub\text{-}Info,pub\text{-}info}(\mathbf{DT}_{D\text{-}name,SD(dep)}(r_{14}))))) \end{aligned}$$

Document
Table

Figure 21. Relation r_{15}

5 Basic Properties of Converters

The main concern about the converters is whether the original structures changed by a converter can be recovered again by other converters. We show their basic properties as the following propositions. We omit the proofs because they can be derived from the definitions of the converters without difficulty.

Propositions 1 and 2 assure that XX-packs are reversible with XX-unpacks. Proposition 3 assures that Or-append is reversible with Or-remove.

Proposition 1. Let $\langle S, r\rangle$ be a relation. Assume that S has the rules $R_S = (A_1, \ldots, A_m, G)$ and $A_i = (O, B)$, $dom(B) = SD$, and $\forall t \in r \exists d \forall v \in t[A_i] \exists c(v[B] = \langle d, c\rangle)$. Then,

$$\mathbf{RU}_{A_i:(O,B),G}(\mathbf{RP}_{A_i}(\langle S, r\rangle)) = \langle S, r\rangle. \qquad \square$$

Proposition 2. Let $\langle S, r \rangle$ be a relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m, G)$, $dom(A_i) = SD, \ldots, dom(A_j) = SD$ for some $1 \leq i \leq j \leq m$, and B is a new attribute. Then,

$$\mathbf{SU}_{B=(A_i,\ldots,A_j),G}(\mathbf{SP}_{B=(A_i,\ldots,A_j)}(\langle S, r \rangle)) = \langle S, r \rangle. \qquad \Box$$

Proposition 3. Let $\langle S, r \rangle$ be a relation. Assume that S has the rules $R_S = (A_1, \ldots, A_m, G)$ and $A_i = (B_1, \ldots, B_n)$, and $dom(B_j) = SD$ for some $1 \leq j \leq n$. Then,

$$\pi_{A_1,\ldots,A_m,G}(\nu_{A_i=(B_1,\ldots,B_n)}(\mathbf{OR}_{B_j,G}(\mu_{A_i}(\tau_{A_i}(\mathbf{OA}_{B_j}(\langle S, r \rangle)))))) = \langle S, r \rangle. \qquad \Box$$

Propositions 4 and 5 assure that XX-unpacks are reversible with XX-packs. However, Proposition 6 says Or-remove is not always reversible with Or-append.

Proposition 4. Let $\langle S, r \rangle$ be a relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m)$, $dom(A_i) = SD$ for some $1 \leq i \leq m$, $\forall t \in r \exists g, d, c(t[A_i] = \langle g : \mathbf{rep}(d), c \rangle)$, and B, O, G are new attributes. Then,

$$\mathbf{RP}_{A_i}(\mathbf{RU}_{A_i:(O,B),G}(\langle S, r \rangle)) = \langle S, r \rangle. \qquad \Box$$

Proposition 5. Let $\langle S, r \rangle$ be a relation. Assume that S has the rule $R_S = (A_1, A_2, \ldots, A_m)$, vv$dom(A_i) = SD$ for some $1 \leq i \leq m$, $\exists k \forall t \in r \exists g, d_1, \ldots, d_k, c(t[A_i] = \langle g : \mathbf{seq}(d_1, \ldots, d_k), c \rangle)$, and $B_1, \ldots, B_k, G$ are new attributes. Then,

$$\mathbf{SP}_{A_i=(B_1,\ldots,B_k)}(\mathbf{SU}_{A_i=(B_1,\ldots,B_k),G}(\langle S, r \rangle)) = \langle S, r \rangle. \qquad \Box$$

Proposition 6. Let $\langle S, r \rangle$ be a relation. Assume that S has the rule $R_S = (A_1, \ldots, A_m)$, $dom(A_i) = SD$ for some $1 \leq i \leq m$, and G is a new attribute. Then,

$$\mu_A(\mathbf{OA}_{A_i}(\nu_{A=(A_1,\ldots,A_m)}(\mathbf{OR}_{A_i,G}(\langle S, r \rangle)))) = \langle S, r \rangle$$

does not always hold. $\Box$

6 Conclusion

With the recent advances in information technology such as digital libraries, WWW, and CALS, structured documents have been widely recognized as important information resources. In this paper, we have proposed the NR/SD model for the seamless integration of structured documents and relational databases. The NR/SD model uses the nested relational structures incorporating the structured document type and provides a number of algebra operations, to manipulate data in structured documents and relations. In particular, the converters attain dynamic conversion of structured documents into nested relational structures and vice veasa. We have shown that the NR/SD algebra enables seamless manipulation of structured documents and relational databases. With the NR/SD algebra, we can formally specify the process of deriving necessary information in the form of structured document as required in WWW environments. Also, we can develop user views on the document repository as in the relational database. We are sure that the NR/SD model can work as a useful formal modeling framework in various system integration contexts. Finally, we have studied the reversibility of operations by the converters as one of their basic properties.

Future research issues include more detailed analysis of properties of the operators, development of user-friendly query languages, and system architecture and implementation schemes to support the NR/SD model. In the context of the scenario in Section 2, caching and optimization techniques are also indispensable issues. These issues will be discussed in forth-coming papers.

Acknowledgement

The authors appreciate collaboration with many members of Database Laboratory, University of Tsukuba. This work was partially supported by the Grand-in-Aid for Science Research from the Ministry of Education, Science, and Culture of Japan.

References

[1] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, "Object Exchange Across Heterogeneous Information Sources," in *Proc. 11th International Conference on Data Engineering*, Mar. 1995, pp. 251-260.

[2] A. R. Hurson, M. W. Bright, and S. Pakzad, (eds.), *Multidatabase Systems: An Advanced Solution for Global Information Sharing*, IEEE Computer Society Press, 1994.

[3] S. T. March, (ed.), *Special Issue on Heterogenenous Databases*, ACM Computing Survey, vol. 22, no. 3, 1990.

[4] *Information Processing – Text and Office System – Standard Generalized Markup Language (SGML)*, ISO 8879, 1986.

[5] G. Wiederhold, "Digital Libraries, value, and productivity," *Communications of the ACM*, vol. 38, no. 4, April 1995, pp. 85-96.

[6] *CALS Implementation Guide*, MIL-HDBK-59B, 1994.

[7] E. Krol, *The Whole Internet: User's Guide & Catalog, Second Edition*, O'Reilly & Associates, Inc., 1994.

[8] *Hypermedia/Time-based Structuring Language (Hytime)*, ISO/IEC 10744, 1992.

[9] S. Abiteboul, P. C. Fischer, and H. J. Scheck, (eds.), *Nested Relations and Complex Objects in Databases*, no. 361 in Lecture Notes in Computer Science, Springer-Verlag, 1989.

[10] P. C. Fischer and S. J. Thomas, "Operators for Non-First-Normal-Form Relations," in *Proc. IEEE COMPSAC83*, Chicago, Nov. 1983, pp. 464-475.

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

[12] D. D. Chamberlin, J. N. Gray, and I. L. Traiger, "Views, Authorization, and Locking in a Relational Database system," in *Proc. National Computer Conference*, Anaheim, 1975, pp. 425-430.

[13] R. Sacks-Davis, T. Arnold-Moore, and J. Zobel, " Database systems for structured documents," *Proc. International Symposium on Advanced Database Technologies and Their Integration,* Nara, Japan, 1994, pp. 272-283.

[14] R. Sack-Davis, A. Kent, K. Ramamohanarao, J. Thom, and J. Zobel, "Atlas: A Nested Relational Database System for Text Applications," *IEEE Trans. Knowledge and Data Engineering,* vol.7, no. 3, June 1995, pp. 454-470.

[15] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl, "From Structured Documents to Novel Query Facilities," in *Proc. ACM SIGMOD International Conference on Management of Data,* May 1994, pp. 313-324.

[16] T. W. Yan and J. Annevelink, "Integrating a Structured-Text Retrieval System with an Object-Oriented Database System," in *Proc. 20th VLDB Conference*, Santiago, Chile, 1994, pp. 740-749.

[17] W. B. Croft, L. A. Smith, and H. R. Turtle, "A Loosely-Coupled Integration of a Text Retrieval System and an Object-Oriented Database System," in *Proc. ACM SIGIR Conference*, 1992, pp. 223-232.

[18] M. Volz, K. Aberer, and K. Böhm, "Applying a Flexible OODBMS-IRS-Coupling to Structured Document Handling," in *Proc. 12th International Conference on Data Engineering*, 1996.

[19] R. H. Güting, R. Zicari, and D. M. Choy, "An Algebra for Structured Office Documents," *ACM Trans. Office Information Systems*, vol. 7, no. 4, Apr. 1989, pp. 123-157.

[20] G. E. Blake, M.P. Consens, P. Kilpeläinen, P. Larson, T. Snider, and F. Tompa, "Text/Relational Database Management Systems: Harmonizing SQL and SGML," in *Proc. International Conference on Applications of Databases*, no.819 in Lecture Notes in Computer Science, Vadstena, Sweden, 1994, pp. 267-280.

[21] G. E. Blake, M. P. Consens, I. J. Davis, P. Kilpeläinen, P. Larson, T. Snider, and F. W. Tompa, "Text/Relational Database Management systems: Overview and Proposed SQL Extensions," Technical Report CS-95-25, UW Centre for the New OED and Text Research, Department of Computer Science, University of Waterloo, June 1995.

[22] M. Yoshikawa, O. Ichikawa, and S. Umemura, "Amalgamating SGML Documents and Databases," in *Proc. 5th International Conference on Extending Database Technology*, March 1996.

[23] F. J. Burkowski, "An Algebra for Hierarchically Organized Text-dominated Databases," *Information Processing & Management*, vol.28, no. 3, 1992, pp. 333-348.

[24] C. L. A. Clarke, G. V. Cormack, and F. J. Burkowski, "An Algebra for Structured Text Search and a Framework for its Implementation," *The Computer Journal*, vol. 38, no. 1, 1995, pp. 43-56.

[25] M. P. Consens and T. Milo, "Algebras for Querying Text Regions," in *Proc. ACM Symposium on Principles of Database Systems*, 1995, pp. 11-22.

[26] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman, "MedMaker: A mediation system based on declarative specifications," available by anonymous ftp from db.stanfora.edu as the file pub/papakonstantinou/1995/medmaker.ps, 1995.

[27] M. Gyssens and D. Van Gucht, "The Expressiveness of Query Languages for Nested Relations," *Data Engineering*, vol. 11, no. 3, 1988, pp. 48-55.