A Data Modeling and Query Processing Scheme for Integration of Structured Document Repositories and Relational Databases

Atsuyuki Morishima

Doctoral Degree Program in Engineering

Hiroyuki Kitagawa

Institute of Information Sciences and Electronics

University of Tsukuba
1-1-1 Tennohdai, Tsukuba, Ibaraki 305, Japan
{*mori, kitagawa*}*@dblab.is.tsukuba.ac.jp*

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 first present a data model named the NR/SD model as a basic data modeling framework for integration of structured documents and relational databases. Then, we discuss the query processing and optimization scheme for environments including structured document repositories 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. This feature poses the following issues in query processing: (1) Utilization of the local query processing capability of the document repository and the relational database, and (2) Efficient manipulation of structured document data whose volume is potentially quite large. We discuss the query processing and optimization scheme mainly focusing on these issues.

Keywords integration of heterogeneous information resources, document databases, data model, query processing

1 Introduction

Today, a huge amount of information is accumulated in a variety of physically and/or logically distributed, autonomous, and heterogeneous information repositories. Thus, integration of such repositories has been one of the hot research issues [11][13][20]. Databases and structured documents are representatives of important information resources. In addition to well-structured data as in the conventional databases, semi-structured data such as structured documents as described in SGML [15] have been widely used, and have increased significance in applications such as digital libraries [26], CALS [6], WWW [17], and hypermedia descriptions [14].

Proceedings of the Fifth International Conference on Database Systems for Advanced Applications, Melbourne, Australia, April 1–4, 1997.

The objective of this research is to provide a framework for the seamless integration of structured document repositories and relational databases. One of the promising approaches to integrate heterogeneous information resources is to use software modules or agents called *mediators* and *wrappers* [20][25]. We follow this approach to attain integration of structured document repositories and relational databases (Figure 1). The mediator acts as a coordinator and receives users' requests. After the analysis of the request, it dispatches wrappers to local information repositories.[1] Each wrapper issues local commands to the local information repository and receives intermediate result. Then, the wrapper translates it into a predefined data representation form and sends it back to the mediator. Finally, the mediator collects data returned by the wrappers and produces the final result.

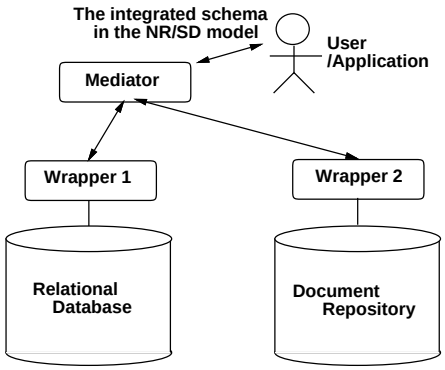

Figure 1: Integration of document repositories and relational databases

In this paper, we first present a new data model named the *NR/SD model* as a basic modeling framework for integration of structured documents and relational databases. Then, we present a query processing and optimization scheme for the NR/SD model in the mediator-based integration framework.

Data modeling constructs of the NR/SD model are nested relational structures and the abstract data type concept. The NR/SD model treats raw structured documents as values of an abstract data type named the *structured document type*. The structured document type has a number of associated functions to retrieve text elements contained

[1] In this context, we do not assume that wrappers always reside at local information repositories as in TSIMMIS [20].

in the structured documents. They are based on the region algebra [9], which serves as a formal basis to model text manipulation and is also used in real text retrieval applications [23]. Thus, the NR/SD model provides basic constructs to model data both in structured documents and relational databases, and operators for their manipulation.

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 embedded in 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 based on the nested relational algebra operators, such as join and nest. In addition, we can get required data in the form of structured documents as well as relations. This is useful in such cases as WWW browsers are used as the user interface. Furthermore, the operators in the NR/SD model can also be used to develop various user views on top of the stored structured documents in analogy to views in relational databases [8].

We discuss query processing and optimization issues related to the NR/SD model. In query processing in the environment as shown in Figure 1, we have to pay attention to the following points:

(1) We should make the best of the local query processing capability of the document repository and the relational database. Needless to say, we should use the capability of the document repository for text manipulation and that of the relational database for relation manipulation. It is important to note that, in the NR/SD model, we can convert nested relational structures into structured documents and vice versa. Therefore, for example, data represented in nested relational structures may originally resides in the document repository. Then, when some nested algebra operators are applied to the nested relational structures, it is desirable to make use of the local filtering capability of the document repository to support the operation. The query processing and optimization scheme presented in the paper incorporates rules to cope with such issues as well as conventional query optimization rules.

(2) Since structured documents could contain a large amount of data, naive transfer of documents would require the mediator to have large work space. It would also bring about very large intermediate query processing cost and data transfer cost. Our query processing and optimization scheme incorporates the concept of *abstract value* to alleviate the problem. Operations of structured documents in the NR/SD model often require only higher-level text elements and document structures, and detailed parts are only necessary to obtain the final query results. In such cases, the use of abstract values allows us to defer transfer of detailed document data until the final query processing phase. Then, we can reduce the work space and the intermediate query processing cost of the mediator, and sometimes attain the reduction of the total data transfer cost.

The rest of this paper is organized as follows. In Section 2, related works are briefly surveyed. Section 3 describes an example scenario of the integration of structured document repositories and relational databases. This example is used in the remaining part of the paper. Section 4 gives overview of the data structures and operators in the NR/SD model. In Section 5, we specify data manipulation of the example scenario in the NR/SD model. Section 6 presents the query processing and optimization scheme. Section 7 is the conclusion.

2 Related Works

There are a number of approaches to integration of heterogeneous information resources, in particular including semi-structured data as well as well-structured data. One approach is to extract common structures and properties of data stored in the heterogeneous information repositories and to provide a data model which can uniformly accommodate them. An advantage of this approach is that users can manipulate different kinds of data in only a single uniform modeling and operational framework. CPL [4], UnQL [5], and OEM [20] follow this approach. CPL provides a rich set of data types including lists and variants to prepare for the heterogeneity of data. In contrast of this, UnQL and OEM use only simple nested data structures into which original data are abstracted and translated. Each of those models provides a set of operations based on the comprehension syntax and/or SQL.

Another promising approach to the integration is the *hybrid* approach. In this approach, existing data modeling frameworks fit for different information repositories are combined, or additional facilities are incorporated into a well-known existing data modeling framework. This approach intends to provide a natural combination of familiar frameworks rather than introducing a novel but unfamiliar data model. It also allows us to make use of the well-known established technologies including storage management, index structure, and query processing. In addition, the translation overhead is generally smaller than the former approach. The integration through the NR/SD model falls in this category.

There are several studies along the line of the hybrid approach relating to structured documents and databases [1][2][3][7][10][21][22][24][27]. Abiteboul and others [1][2] proposed the notion of structuring schema which specifies how data contained in text should be incorporated into databases. Atlas [22] 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 [7]. COINS [10], Volz and others [24], and Yan and others [27] all proposed schemes to jointly use the DBMS and the IR system to manage structured documents. T/RDBMS [3] is similar to our approach in that it combines the relational data model and the abstract data type representing structured documents. In T/RDBMS, information inside structured documents can be viewed as a predefined collection of relations and queried in the extended SQL.

Features of the NR/SD model in the light of those studies are as follows.

- The NR/SD model provides a framework for *symmetrically* and *dynamically* amalga-

mating structured documents and databases. The converters transform structured documents into relational structures and vice versa. Therefore, users can restructure existing documents based on nested relational algebra and create new documents from relational structures.

- The converters enable incremental conversion between structured documents and nested relational structures. For example, given a collection of documents with different structures, we can extract common substructures and represent them in nested relational structures, ignoring detailed structures.

NST [12] and Järvelin and Niemi [16] use nested relational structures as a basic modeling framework of structured documents. However, they are not supposed to work for integration of structured documents and databases.

Yoshikawa and others proposed another approach to the integration of structured documents and databases [28]. Their approach is to provide a general mechanism to make reference links from components of SGML documents to database objects and is different from ours.

Integration based on object-oriented models is also another promising approach [19]. Although object-oriented models are powerful and practically useful frameworks, they are too rich to be used as a formal basis for discussion on the above mentioned symmetric, dynamic, and incremental data transformation. If we focus on structuring aspect of object-oriented models such as complex objects, many concepts in the NR/SD model could be mapped into the world of object-oriented models.

Although a lot of works have been done on query processing in distributed database and multidatabase environments [11][13], there is little work dedicated to query processing involving structured documents and databases. Abiteboul and others [1] proposed a grammar-based approach to incorporate text data into the object-oriented database. They presented an optimization technique to push down selections and projections so that they could be executed in the text parser. Volz and others [24] outlined query processing in environments where structured documents are cooperatively managed by the OODBMS and the IR system. They mainly discussed dynamic derivation of relevance values of text data which is not actually stored in the IR system. Yan and others [27] discussed query processing issues which occur when the external function capability of the OODBMS is used for incorporating the functionality of the IR system. The query processing issues discussed in the previous works are different from ours. Objectives of T/RDBMS [3] include integration of the document repository and the database. However, they did not describe concrete query processing and optimization schemes.

3 Example Scenario

In this section, we show an example integration scenario of a structured document repository and a relational database in the environment shown in Figure 1. The relational database is supposed to manage faculty data of some university (say, A university), and the document repository is assumed to store papers in the form of structured documents. Figure 2 shows the schema of the relation "Faculty" managed in the relational database.

Assume that we want to get a set of structured documents, each of which contains the faculty name, academic information such as his/her speciality and the course he/she teaches[2], and the list of his/her publications for each faculty member of A university. Then, also assume that we need to select documents for faculty members whose specialities are database or who teach some database course.

Faculty					
FID	Name	D-Name	Speciality	Course	Rank

Figure 2: Relation "Faculty"

In the NR/SD model, this request is specified as a sequence of data manipulation steps. The outline is as follows and the complete specification is given in Section 5.

- Relevant data is extracted from the structured documents and is grouped by the author.
- It is joined with data in the "Faculty" relation and new structured documents which contain the above information are created.
- Documents for the qualified faculty members are selected.

4 NR/SD model

In this section, we introduce the data structures and operators in the NR/SD model. Our explanation here is rather illustrative omitting formal definitions because of space limitation. Formal definitions of data structures and the converters are given in [18]. As mentioned before, nested relational structures and the abstract data type, *structured document type*, are basic data modeling constructs of the NR/SD model. Figure 3 shows relation r_0. In the NR/SD model, the order of attributes of a relation is significant. The domains of attributes B and E are String and Integer, respectively. Types such as String and Integer are called *ordinary types*. The domains of C and F are *the structured document type* explained in Subsection 4.1.

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

Figure 3: Relation r_0

4.1 Structured Document Type

A value of the *structured document type* or *SD type* is a pair of a *DTD* (Document Type Definition) and text in which tags are embedded according to the DTD. We call a value of SD type an *SD value*.

Figure 4 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

[2]For simplicity, each faculty member is assumed to have one speciality and teach only one course.

in SGML, although we do not consider exceptions, recursions, and SGML attributes for simplicity. A tagged text is divided into *elements* surrounded by a *begin tag* <*g*> and an *end tag* </*g*>, where *g* is a *generic identifier* representing the *element type*. Elements can be nested within other elements. For example, the tagged text in Figure 4 has an element "letter," and the "letter" has "from," "date," "to," etc. as sub-elements.

The DTD prescribes how the elements can be hierarchically constructed by sub-elements. In Figure 4, each line in the DTD is an *element type definition*. An element "letter" is a sequence of "from," "date," "to," "salutation," "body," and "signature" (**seq** structure). An element "body" consists of zero or more "para" elements (**rep** structure). An element "addr" contains either "home" or "office" (**or** structure). The element types such as "from," "home," and "para" have no internal structures.

Element types defined in a DTD must form a rooted DAG structure. The DTD in Figure 4 forms the rooted DAG (tree in this case) shown in Figure 5. The DTD in Figure 4 is also represented in the linear form "letter: **seq** (from: **ptext**, date: **ptext**, to: **seq**(name: **ptext**, addr: **or**(home: **ptext**, office: **ptext**)), salutation: **ptext**, body: **rep**(para: **ptext**), signature: **ptext**)."

letter = **seq**(from, date, to, salutation, body, signature)
to = **seq**(name, addr)
addr = **or**(home, office)
body = **rep**(para)

```
<letter>
<from>1-3-28 tennohdai ...</from>
<date>April 1, 1997</date>
<to><name>Prof. Alexander White</name><addr>
<office>DB Lab., Inst. of ...</office></addr></to>
<body>
<salutation>Dear ...</salutation>
<para>...</para>...<para>...</para>
</body>
<signature>Thomas Jackson</signature>
</letter>
```

Figure 4: Sample SD value

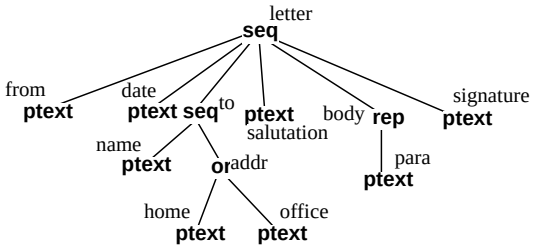

Figure 5: DAG representation of DTD

4.2 Functions Associated with SD Type

The structured document type SD has a number of associated functions. They are based on the region algebra [9], and are used to retrieve elements contained in SD values. In addition to ordinary text retrieval, they facilitate element retrieval which depends on document structures and information embedded in tags.

A *region* is a contiguous part of text. In this context, we only consider regions which correspond to elements. The region algebra is a set-at-a-time algebra. An expression e of the region algebra is generated by the following rule:

$$e \rightarrow R_i | e \cup e | e \cap e | e - e \\ | e \supset e | e \subset e | e < e | e > e | \sigma[w](e) | (e)$$

where R_i is a generic identifier (e.g. "para") or "**doc**." If R_i is a generic identifier, the region algebra expression "R_i" returns the set of elements whose generic identifiers are R_i. If R_i is "**doc**," it returns a singleton set which contains one element corresponding to the whole tagged text. Union ($\cup$), intersection ($\cap$), difference ($-$) are ordinary set operators. The *including* ($\supset$), *included* ($\subset$), *follows* ($>$), and *precedes* ($<$) operators are defined as follows:

$$R \supset S = \{r \in R : \exists s \in S, r \supset s\} \\ R \subset S = \{r \in R : \exists s \in S, r \subset s\} \\ R > S = \{r \in R : \exists s \in S, r > s\} \\ R < S = \{r \in R : \exists s \in S, r < s\}$$

where $r \supset s$ holds when element r strictly includes element s, $r > s$ holds when r follows s (i.e. the begin position of r is after the end position of s), and $r \subset s$ and $r < s$ are defined in a similar way. The selection (σ) operator selects elements which include occurrences of the word "w."

For example, for the tagged text "`<c> <a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b> </a> <b> w1 w5 </b> </c>`", the region algebra expression "a" returns the set of elements {"`<a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b> </a>`"}, "b" returns {"`<b> w1 w2 </b>`," "`<b> w3 </b>`," "`<b> w4 w1 </b>`," "`<b> w1 w5 </b>`"}, "**doc**" returns {"`<c> <a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b> </a> <b> w1 w5 </b> </c>`"}, and "σ["w1"]$(b) \subset a$" returns {"`<b> w1 w2 </b>`," "`<b> w4 w1 </b>`" }.

The NR/SD model provides the *apply* operator (denoted by α) to extract elements from SD values contained in relations. Region algebra expressions are used to give the extraction specification. Suppose that r is a relation which has an SD type attribute $attr_1$, $expr$ is a region algebra expression, and $attr_2$ is a new attribute name. Then, $\alpha_{attr_1,expr,attr_2}(r)$ adds the new attribute $attr_2$ to the original relation r. The attribute $attr_2$ stores the SD values which are returned as the result of applying the region algebra expression $expr$ to the SD value in the attribute $attr_1$. For example, if $expr$ is given as "σ["w1"]$(b) \subset a$," and the $attr_1$ value of a target tuple is ⟨ c:**seq**(a:**rep**(b:**ptext**),b:**ptext**), "`<c> <a> <b> w1 w2 </b> <b> w3 </b> <b> w4 w1 </b> </a> <b> w1 w5 </b> </c>`" ⟩, then the new $attr_2$ value is { ⟨b:**ptext**, "`<b> w1 w2 </b>`"⟩, ⟨ b:**ptext**, "`<b> w4 w1 </b>`"⟩}.

4.3 Converters

The NR/SD model features the six operators: *Rep-unpack*, *Seq-unpack*, *Rep-pack*, *Seq-pack*, *Or-append*, and *Or-remove*, which are generically called the *converters*. The converters transform SD values into nested relational structures and vice versa.

Rep-unpack/Seq-unpack

Rep-unpack (**RU**) transforms the top-level **rep** structures embedded in the SD values into nested relational structures. Seq-unpack(**SU**) deals with the **seq** structures instead of the **rep** structures. Figure 6 gives examples of Rep-unpack and Seq-unpack operators, where

r_1:

A	B
x	⟨ a:**rep**(b:**or**(c:**seq**(d:**ptext**,e:**ptext**),f:**ptext**)) , "<a><f>W1</f><c><d>W2</d><e>W3</e></c>"⟩
y	⟨ g:**rep**(h:**seq**(i:**ptext**,j:**ptext**,k:**ptext**)) , "<g><h>W4<j>W5</j><k>W6</k></h><h>W7<j>W8</j><k>W9</k></h><h>W10<j>W11</j><k>W12</k></h></g>"⟩

r_2:

A	B		E
	C	D	
x	1	⟨ b:**or**(c:**seq**(d:**ptext**,e:**ptext**),f:**ptext**) , "<f>W1</f>"⟩	a
	2	⟨ b:**or**(c:**seq**(d:**ptext**,e:**ptext**),f:**ptext**) , "<c><d>W2</d><e>W3</e></c>"⟩	
y	1	⟨ h:**seq**(i:**ptext**,j:**ptext**,k:**ptext**) , "<h>W4<j>W5</j><k>W6</k></h>"⟩	g
	2	⟨ h:**seq**(i:**ptext**,j:**ptext**,k:**ptext**) , "<h>W7<j>W8</j><k>W9</k></h>"⟩	
	3	⟨ h:**seq**(i:**ptext**,j:**ptext**,k:**ptext**) , "<h>W10<j>W11</j><k>W12</k></h>"⟩	

r_3:

A	B
x	⟨ a:**seq**(b:**rep**(c:**ptext**),d:**or**(e:**ptext**,f:**ptext**,g:**ptext**)) , "<a><c>W1</c><c>W2</c><d><e>W3</e></d>"⟩
y	⟨ h:**seq**(i:**seq**(j:**ptext**,k:**ptext**),l:**rep**(m:**ptext**)) , "<h><j>W4</j><k>W5</k><l><m>W6</m><m>W7</m></l></h>"⟩

r_4:

A	C	D	E
x	⟨ b:**rep**(c:**ptext**) , "<c>W1</c><c>W2</c>"⟩	⟨ d:**or**(e:**ptext**,f:**ptext**,g:**ptext**), "<d><e>W3</e></d>"⟩	a
y	⟨ i:**seq**(j:**ptext**,k:**ptext**), "<j>W4</j><k>W5</k>"⟩	⟨ l:**rep**(m:**ptext**) , "<l><m>W6</m><m>W7</m></l>"⟩	h

Figure 6: Sample relations (1)

$$r_2 \leftarrow \mathbf{RU}_{B:(C,D),E}(r_1)$$
$$r_4 \leftarrow \mathbf{SU}_{B=(C,D),E}(r_3).$$

Attribute C in r_2 is used to keep the order of elements, and E in r_2 and r_4 is to represent extracted generic identifiers.

Rep-pack/Seq-pack

Rep-pack (**RP**) transforms sub-relations into the **rep** structures in SD values. Seq-pack (**SP**) deals with the **seq** structures instead of the **rep** structures. Figure 6 gives examples of Rep-pack and Seq-pack operators, too, where

$$r_1 \leftarrow \mathbf{RP}_B(r_2)$$
$$r_3 \leftarrow \mathbf{SP}_{B=(C,D)}(r_4).$$

In the above example of Rep-pack, the top-level generic identifier used for the new SD value in r_1 is automatically determined by the rightmost attribute E in r_2. The same remark applies to Seq-pack. We can explicitly specify the generic identifier as a parameter instead of using the rightmost attribute. In this case, we specify the expressions like $r_1 \leftarrow \mathbf{RP}_{B,g}(r_2)$ and $r_3 \leftarrow \mathbf{SP}_{B=(C,D),g}(r_4)$, where g is a given generic identifier.

Or-remove

Or-remove (**OR**) removes the top-level **or** structures. This operator can be used to prepare for further applications of Rep-unpack or Seq-unpack operators, which require the root structure of target SD values to be **rep** or **seq**. For example, the following Or-remove transforms relation r_5 into relation r_6 as shown in Figure 7.

$$r_6 \leftarrow \mathbf{OR}_{B,C}(r_5)$$

Then, we can apply Seq-unpack to r_6. Attribute C represents removed generic identifiers.

Or-append

Or-append (**OA**) adds the **or** structures at the top level. This operator can be used to prepare for Rep-pack operator, which requires the target SD value sets to have the same element type at their roots. For example, we cannot directly apply Rep-pack to relation r_7 shown in Figure 7. However, the following Or-append yields relation r_8, to which we can apply Rep-pack:

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

Here, the rightmost attribute E in r_7 is used to determine the top-level generic identifier.

4.4 NR/SD Algebra

The *NR/SD algebra* consists of the converters, the apply operator, the *domain translator*, and the ordinary nested relational algebra operators listed in Figure 8. Domain translator $\mathbf{DT}_{attr,type}(r)$ changes the type of attribute *attr* into *type*. When *type* is SD type, a generic identifier must be also specified. For example, $\mathbf{DT}_{A,SD(para)}(r_1)$ changes the type of attribute A of relation r_1 into SD type and value "v" in the attribute A into SD value ⟨ para=**ptext**, " <para>v</para> "⟩.

In addition to the above primitive operators, *composite* operators are defined as their combinations. Join operator $\bowtie_p$ is among them. We define extended selection operator. It selects tuples whose SD values satisfy the given selection condition. The selection condition is specified by a region algebra expression *expr*. The extended selection $\sigma_{\rho(attr,expr)}(r)$ is defined as a composite operator $\pi_{\neg A}(\sigma_{A\neq\phi}(\alpha_{attr,expr,A}(r)))$, where $\neg A$ denotes all attributes in r other than A. The pseudo predicate $\rho(attr, expr)$ holds if *expr* returns a non-empty set of SD values for the SD value in attribute *attr*.

For example, let r be a unary relation with attribute A and contains the set of SD values { ⟨ a:**rep**(b:**or**(c:**ptext**, d:**ptext**)), "<a> <c> w1 </c> " ⟩, ⟨ a:**rep**(b:**or**(c:**ptext**, d:**ptext**)), "<a> <d> w2 </d> " ⟩}. Then, $\sigma_{\rho(A,b\supset c)}(r)$ returns the singleton set of an SD value { ⟨ a:**rep**(b:**or**(c:**ptext**, d:**ptext**)), "<a> <c> w1 </c> " ⟩}.

r_5:

A	B
x	⟨ a:**or**(b:**seq**(c:**rep**(d:**ptext**),e:**ptext**),f:**seq**(g:**ptext**,h:**ptext**)) , "<a><c><d>W1</d><d>W2</d></c><e>W3</e>"⟩
y	⟨ i:**or**(f:**seq**(g:**ptext**,h:**ptext**),j:**or**(k:**ptext**,l:**ptext**)) , "<f><g>W4</g><h>W5</h></f>"⟩
z	⟨ m:**or**(n:**seq**(o:**ptext**,p:**ptext**),f:**seq**(g:**ptext**,h:**ptext**)) , "<m><n><o>W6</o> W7 </n></m>"⟩

r_6:

A	B	C
x	⟨ b:**seq**(c:**rep**(d:**ptext**),e:**ptext**) , "<c><d>W1</d><d>W2</d></c><e>W3</e>"⟩	a
y	⟨ f:**seq**(g:**ptext**,h:**ptext**) , "<f><g>W4</g><h>W5</h></f>"⟩	i
z	⟨ n:**seq**(o:**ptext**,p:**ptext**) , "<n><o>W6</o> W7 </n>"⟩	m

r_7:

A	B		E
	C	D	
x	1	⟨ b:**seq**(c:**ptext**,d:**ptext**) , "<c>W1</c><d>W2</d>"⟩	a
	2	⟨ e:**rep**(f:**ptext**) , "<e><f>W3</f><f>W4</f></e>"⟩	
	3	⟨ b:**seq**(c:**ptext**,d:**ptext**) , "<c>W5</c><d>W6</d>"⟩	
y	1	⟨ h:**or**(i:**ptext**,j:**ptext**) , "<h>W7</h>"⟩	g
	2	⟨ k:**seq**(l:**ptext**,m:**ptext**) , "<k><l>W8</l><m>W9</m></k>"⟩	

r_8:

A	B	
	C	D
x	1	⟨ a:**or**(b:**seq**(c:**ptext**,d:**ptext**),e:**rep**(f:**ptext**)) , "<a><c>W1</c><d>W2</d>"⟩
	2	⟨ a:**or**(b:**seq**(c:**ptext**,d:**ptext**),e:**rep**(f:**ptext**)) , "<a><e><f>W3</f><f>W4</f></e>"⟩
	3	⟨ a:**or**(b:**seq**(c:**ptext**,d:**ptext**),e:**rep**(f:**ptext**)) , "<a><c>W5</c><d>W6</d>"⟩
y	1	⟨ g:**or**(h:**or**(i:**ptext**,j:**ptext**),k:**seq**(l:**ptext**,m:**ptext**)) , "<g><h>W7</h></g>"⟩
	2	⟨ g:**or**(h:**or**(i:**ptext**,j:**ptext**),k:**seq**(l:**ptext**,m:**ptext**)) , "<g><k><l>W8</l><m>W9</m></k></g>"⟩

Figure 7: Sample relations (2)

Some unary composite operators are also defined so that they can directly manipulate internal structures inside relations. For example, the extended Seq-pack operator $\mathbf{SP^*}_B(r)$ allows B to be any internal attribute inside the relation r. Their formal definitions are given in [18].

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 8: Nested relational algebra operators

5 Query Specification

In this section, we apply the NR/SD model to the example scenario described in Section 3. In the NR/SD model, the structured document repositoriy is viewed as a unary relation "Document," the domain of whose unique attribute "doc" is SD type (Figure 9). Each structured document in "Document" is assumed to have the DTD shown in Figure 10. The query result, namely a set of structured documents, is modeled as a relation shown in Figure 11. We assume that the result structured documents are required to have the DTD shown in Figure 12. The data manipulation request in Section 3 can be specified as follows. In the following NR/SD algebra expressions, we omit specifications of the domain translators for notational simplicity.

Document
Doc

Figure 9: Relation "Document"

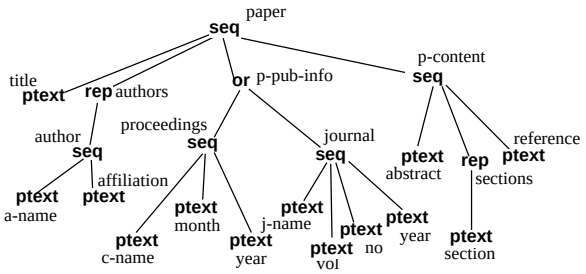

Figure 10: DTD in attribute "Doc"

Result
Pub-Table

Figure 11: Required relation

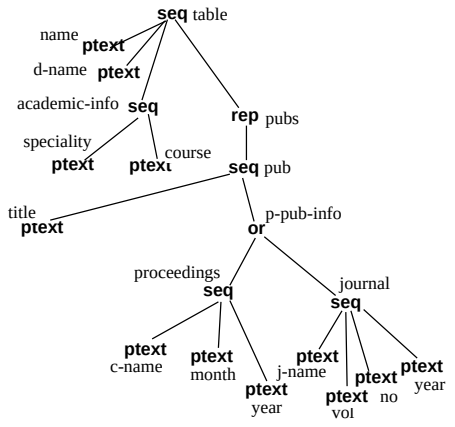

Figure 12: Required DTD in attribute "Pub-Table"

(1) First, we unpack higher-level structures of the SD values contained in relation "Document."

$$r_9 \leftarrow \mathbf{SU}_{Author=(A\text{-}Name,Affiliation),G_3}(\mu_{Authors}(\mathbf{RU}_{Authors:(O_1,Author),G_2}(\mathbf{SU}_{Doc=(Title,Authors,Pub\text{-}Info,Content),G_1}(Document))))$$

(2) Then, we select from relation r_9 those tuples whose "Affiliation" values are "A-univ," remove unnecessary attributes, and apply Nest to group publications by the author.

$$r_{10} \leftarrow \nu_{Pubs=(Title,Title,Pub\text{-}Info)}(\pi_{A\text{-}Name,Title,Title,Pub\text{-}Info}(\sigma_{Affiliation=\text{"A-univ"}}(r_9)))$$

(3) We joins the relation r_{10} with relation "Faculty," and remove unnecessary attributes.

$$r_{11} \leftarrow \pi_{Name,D\text{-}Name,Speciality,Course,Pubs}(Faculty \bowtie_{Name=A\text{-}Name} (r_{10}))$$

(4) We transform r_{11} into structured documents by packing operators.

$$r_{12} \leftarrow \mathbf{SP}_{Pub\text{-}Table=(Name,D\text{-}Name,A\text{-}Info,Pubs),table}(\mathbf{SP}_{A\text{-}Info=(Speciality,Course),academic-info}(\mathbf{RP}_{Pubs,pubs}(\mathbf{SP*}_{Pub=(Title,Pub\text{-}Info),pub}(r_{11}))))$$

(5) Finally, we select the documents which contain the word "database" in their "academic-info" elements.

$$r_{13} \leftarrow \sigma_{\rho(Pub\text{-}Table,\sigma[''database'']}(academic-info))(r_{12})$$

6 Query Processing and Optimization

We present a query processing and optimization scheme for NR/SD algebra expressions. The mediator-based framework as described in Section 1 is assumed. In the query processing and optimizations scheme, we can use *abstract SD values* instead of real SD values to reduce the work space and the intermediate query processing cost of the mediator. Two operators are included for this purpose in addition to the NR/SD algebra operators. In this section, we first discuss abstract SD values, and then the query processing and optimization scheme is presented.

6.1 Abstraction of SD values

We present the concept of the abstract SD value and operators for abstraction and materialization of SD values.

Abstract SD values

Since structured documents could contain a large amount of data, naive transfer of documents from the document repository would require the mediator to have large work space. We introduce the notion of *abstract SD values* (ASD values) to alleviate the problem. Operations of structured documents in the NR/SD model often require only higher-level elements and document structures, and detailed parts are only necessary to obtain the final query results. Thus, we can transfer ASD values containing only partial information required by the mediator to derive intermediate results. The use of ASD values can also reduce the intermediate query processing cost of the mediator.

The following is a sample ASD value for the SD value in the first tuple in relation r_1 (Figure 6).

⟨ a:**rep**(b:UNKNOWN) , "<a> [id1, 3, 5]   [id1, 8, 15] " ⟩

In the DTD part of the ASD value, special structure "UNKNOWN" is introduced. This means that the internal structure of element type "b" is unknown. In the tagged text part, each "b" element does not contain actual text data. Rather, each "b" element in the ASD value contains a triplet named *marker*, which specifies the location of actual text data stored in the document repository. The marker consists of the identifier of a document in the document repository, the begin position of the designated element in the document, and its end position.

Note that the result of applying Rep-unpack operator $\mathbf{RU}_{B:(C,D),E}$ to relation r_1 shown in Figure 6 can be derived, enen if the SD value is replaced by the ASD value.

ASD values at several abstraction levels can be generated. The most abstract ASD value has only one UNKNOWN structure (e.g. ⟨ a:UNKNOWN , "<a> [id1, 2, 16] " ⟩). However, the result of unpacking cannot be derived with this ASD value, since it does not contain enough information on internal structures.

Abstraction operator

Operator $\mathbf{SDA}_{attr,as}(r)$ (SD Abstraction) transforms SD values in attribute *attr* into ASD values. We call this process *abstraction*. Abstraction specification *as* determines the abstraction level, prescribing the DTD structure of the ASD values. Suppose that attribute A has an SD value with DTD "a:**rep**(b:**seq**(c:**seq**(d:**ptext**, e:**ptext**), f:**or**(g:**rep**(h:**ptext**), i:**seq**(j:**ptext**,k:**ptext**)))." Application of $\mathbf{SDA}_{A,\mathbf{rep}(\mathbf{seq}(\text{UNKNOWN},*))}(r)$ transforms it into an ASD value with DTD " a:**rep**(b:**seq**(c:UNKNOWN, f:**or**(g:**rep**(h:**ptext**), i:**seq**(j:**ptext**, k:**ptext**))))," where "*" in the abstraction spcification requires full details.

Materialization operator

Operator $\mathbf{SDM}_{attr}(r)$ (SD Materialization) transforms ASD values in attribute *attr* into SD values. We call this process *materialization*. For example, the ASD value ⟨ a:**rep**(b:UNKNOWN) , "<a> [id1, 3, 5]   [id1, 8, 15] " ⟩ reverts to the original SD value with **SDM** operator.

6.2 Outline of Query Processing

We show the outline of the query processing and optimization scheme. We assume that the relational database can execute relational algebra expressions, and that the document repository has text retrieval capability based on the region algebra and can directly execute $\sigma_{\rho(attr,expr)}(r)$. The mediator is assumed to maintain the schema information of the relational database and the DTD information of documents stored in the document repository. In the following discussion, for simplicity, we also assume that given queries just once refer to one relation in the document repository and another relation in the relational database. However, the discussion can be extend to more general cases without difficulty.

Figure 13 shows the basic query processing framework. The mediator receives a query expressed in NR/SD algebra and transforms it into the form $F(G, H)$[3]. The subexpression G consists of only relational algebra operators applied to a relation stored in the relational database. Wrapper 1 is responsible for processing G. The wrapper submits G to the relational database, and transfers the result $Ans1$ to the mediator. The subexpression H has the form $\mathbf{SDA}_{attr,as}(\sigma_{\rho(attr,expr)}(R))$, where R is a unary relation with attribute $attr$ which models the set of structured documents stored in the document repository. Wrapper 2 submits $\sigma_{\rho(attr,expr)}(R)$ to the document repository, receives the result from the document repository, and translates it into a unary relation consisting of a set of SD values. Then, wrapper 2 executes the **SDA** operator to transforms the SD values into abstract SD values according to the abstraction specification as. After that, the wrapper 2 transfers the result relation $Ans2$ to the mediator. The mediator executes $F(Ans1, Ans2)$ to obtain the final result Ans. In the process, it may fetch some SD values from the wrapper 2 when it executes **SDM** operators contained in F.

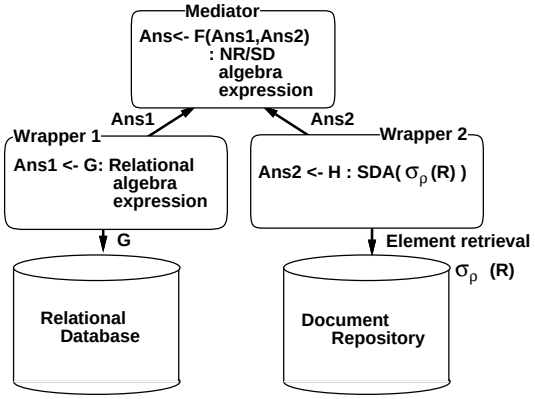

Figure 13: Query processing framework

The mediator derives $F(G, H)$ in the following two steps.

[Step 1] The mediator decomposes the given query expression into $F'(G, H')$, where F' and H' are bases for F and H, respectively. As mentioned before, the subexpression G contains only relational algebra operators applied to a relation in the relational database. H' has the form $\sigma_{\rho(attr,expr)}(R)$, where R is a unary relation which models the document repository. F' consists of NR/SD operators applied to data obtained from both local repositories. For example, suppose that we have a binary relation $R_1(A_1, A_2)$ in the relational database and that we view the document repository as a unary relation $R_2(B)$. Then, the expression $\sigma_{A_2=c}(R_1) \bowtie_{A_1=B_1} \mathbf{SU}_{B=(B_1,B_2),G}(\sigma_{\rho(B,expr)}(R_2))$ is decomposed into F', G, and H' as follows.

$$F': \quad Ans \leftarrow Ans1 \bowtie_{A_1=B_1} \mathbf{SU}_{B=(B_1,B_2),G}(Ans2)$$
$$G: \quad Ans1 \leftarrow \sigma_{A_2=c}(R_1)$$
$$H': \quad Ans2 \leftarrow \sigma_{\rho(B,expr)}(R_2).$$

[3]Actually, $F(G, H)$ includes SD abstractions and materializations in addition to operators explained in Subsection 4.4.

The example here is very simple. Actually, in the decomposition of the query expression into $F'(G, H')$, the mediator tries to push locally executable operators down into G and H', and to make F' include as few operators as possible. This process is carried out based on a number of rewriting rules. In addition to well-known algebraic optimization techniques such as selection push-down, they include rules to utilize local filtering capability of the document repository to support nested relational algebra operators, and conversely, to use the algebraic capability of the relational database to support region-algebra-based data manipulation. Some of the rules are discussed in Subsection 6.3.

[Step 2] The mediator transforms F' and H' into F and H, respectively. First, H is derived as $\mathbf{SDA}_{attr,as}(H')$. The abstraction specification as is determined based on F'. Consider the sample expressions F' and H' shown in Step 1. F' implies that the mediator does not need the actual SD values in attribute B_2 until the join operation is finished. The values in B_2 are parts of SD values contained in R_2. Thus, as is determined to be "**seq**(*,UNKNOWN)" and H becomes

$$Ans2 \leftarrow \mathbf{SDA}_{B,\mathbf{seq}(*,\mathrm{UNKNOWN})}(\sigma_{\rho(B,expr)}(R_2)).$$

F is obtained by inserting **SDM** operators into F' at places where materialized SD values are needed. In the above example, **SDM** is only required to obtain the final Ans. Thus, F is constructed as follows.

$$Ans \leftarrow \mathbf{SDM}_{B_2}(Ans1 \bowtie_{A_1=B_1} \mathbf{SU}_{B=(B_1,B_2),G}(Ans2))$$

In the above explanation, we have shown the use of ASD values in the simplest form. Actually, we can tune the abstraction level based on the available work space in the mediator, the local query processing cost, and the data transfer cost. By giving the abstraction specification "*," we can even insist that real SD values should directly be used instead of ASD values. Discussion on this issue is beyond the scope of the paper, and we consider only this simple use of ASD values in the paper.

6.3 Query Processing Example

In this subsection, we show query processing steps for the NR/SD algebra expressions given in Section 5.

[Step 1] As mentioned before, the mediator has a number of rewriting rules. In Figure 14, we show rules specific to the NR/SD algebra and used in this example. Rules RL1 and RL2 push selection down based on the commutativity of Selection and Seqpack. RL3 and RL4 are rules to screen SD values in advance to discard those which cannot satisfy the outer-most selection condition. These rules are mainly used to make the document repository sift structured documents before data transfer to the mediator. RL5 pushes Selection down through **DT** operator. Note that, by this rule, the selection condition becomes a word containment predicate "∋" on relations. RL6 facilitates further application of other rewriting rules related to $\sigma_{\rho(attr,expr)}$.

(RL1) $\sigma_{\rho(A,expr)}(\mathbf{SP}_{A=(A_1,\ldots,A_n),gi}(r)) \rightsquigarrow \mathbf{SP}_{A=(A_1,\ldots,A_n),gi}(\sigma_{\rho(A_{i1},expr)\vee\ldots\vee\rho(A_{im},expr)}(r))$,
where $expr$ includes no gi and **doc**, and A_{ij} is an attribute in $\{A_{i1},\ldots,A_{im}\}$ such that the predicate $\rho(A_{ij},expr)$ may hold.

(RL2) $\sigma_{\rho(A,\sigma[w](gi))}(\mathbf{SP}_{A=(A_1,\ldots,A_n),gi}(r)) \rightsquigarrow \mathbf{SP}_{A=(A_1,\ldots,A_n),gi}(\sigma_{\rho(A_1,\sigma[w](\mathbf{doc}))\vee\ldots\vee\rho(A_n,\sigma[w](\mathbf{doc}))}(r))$

(RL3) $\sigma_{\rho(A_i,expr)}(\mathbf{SU}_{A=(A_1,\ldots,A_n),G}(r)) \rightsquigarrow \sigma_{\rho(A_i,expr)}(\mathbf{SU}_{A=(A_1,\ldots,A_n),G}(\sigma_{\rho(A,expr)}(r)))$

(RL4) $\sigma_{\rho(A_1,expr)}(\mu_A(\mathbf{RU}_{A:(O,A_1),G}(r))) \rightsquigarrow \sigma_{\rho(A_1,expr)}(\mu_A(\mathbf{RU}_{A:(O,A_1),G}(\sigma_{\rho(A,expr)}(r))))$

(RL5) $\sigma_{\rho(A,\sigma[w](\mathbf{doc}))}(\mathbf{DT}_{A,SD(gi)}(r)) \rightsquigarrow \mathbf{DT}_{A,SD(gi)}(\sigma_{A\ni w}(r))$,
where attribute A of relation r is of String type.

(RL6) $\sigma_{A=w}(\mathbf{DT}_{A,String}(r)) \rightsquigarrow \sigma_{A=w}(\mathbf{DT}_{A,String}(\sigma_{\rho(A,\sigma[w](\mathbf{doc}))}(r)))$,
where attribute A of relation r is of SD type.

Figure 14: Sample rewriting rules

We show the result of Step 1 below. Note that the extended selection on SD values in the original expression to yield r_{13} is pushed down into G. Thus, this selection is executed by the relational database. Moreover, $\sigma_{\rho(Doc,\sigma[''A\text{-}univ''](\mathbf{doc}))}(Document)$ in H' discards documents not including the word "A-univ," which reduces data transfer from the document repository to the mediator.

G:

$$Ans1 \leftarrow \pi_{Name,D-Name,Speciality,Course}(\sigma_{Speciality\ni''database''\vee Course\ni''database''}(Faculty))$$

H':

$$Ans2 \leftarrow \sigma_{\rho(Doc,\sigma[''A-univ''](\mathbf{doc}))}(Document)$$

F' consists of the following expressions:

$$z_1 \leftarrow \mathbf{SU}_{Author=(A\text{-}Name,Affiliation),G_3}(\mu_{Authors}(\mathbf{RU}_{Authors:(O_1,Author),G_2}(\mathbf{SU}_{Doc=(Title,Authors,Pub\text{-}Info,Content),G_1}(Ans2))))$$

$$z_2 \leftarrow \nu_{Pubs=(Title,Title,Pub\text{-}Info)}(\pi_{A\text{-}Name,Title,Title,Pub\text{-}Info}(\sigma_{Affiliation=\text{“A-univ”}}(z_1)))$$

$$z_3 \leftarrow \pi_{Name,D\text{-}Name,Speciality,Course,Pubs}(Ans1 \bowtie_{Name=A\text{-}Name} (z_2))$$

$$z \leftarrow \mathbf{SP}_{Pub\text{-}Table=(Name,D\text{-}Name,A\text{-}Info,Pubs),table}(\mathbf{SP}_{A\text{-}Info=(Speciality,Course),academic-info}(\mathbf{RP}_{Pubs,pubs}(\mathbf{SP^*}_{Pub=(Title,Pub\text{-}Info),pub}(z_3))))$$

[Step 2] F' implies that the mediator needs only real SD values in attributes "Affiliation" and "A-Name." No other SD values are required until the final result is required. SD values in "Affiliation" and "A-Name" correspond to "affiliation" and "a-name" elements contained in the structured documents in the document repository. Thus, the abstraction specification in H is determined to be **seq**(UNKNOWN, **rep**(**seq**(*,*)), UNKNOWN, UNKNOWN), and we get the following expression.

H:

$$Ans2 \leftarrow \mathbf{SDA}_{Doc,\mathbf{seq}(\text{UNKNOWN},\mathbf{rep}(\mathbf{seq}(*,*)),\text{UNKNOWN},\text{UNKNOWN})}(\sigma_{\rho(Doc,\sigma[''A-univ''](\mathbf{doc}))}(Document))$$

Finally, **SDM** operator is inserted into F' to obtain the final query result, and we get the following F.

F:

$$z_1 \leftarrow \mathbf{SU}_{Author=(A\text{-}Name,Affiliation),G_3}(\mu_{Authors}(\mathbf{RU}_{Authors:(O_1,Author),G_2}(\mathbf{SU}_{Doc=(Title,Authors,Pub\text{-}Info,Content),G_1}(Ans2))))$$

$$z_2 \leftarrow \nu_{Pubs=(Title,Title,Pub\text{-}Info)}(\pi_{A\text{-}Name,Title,Title,Pub\text{-}Info}(\sigma_{Affiliation=\text{“A-univ”}}(z_1)))$$

$$z_3 \leftarrow \pi_{Name,D\text{-}Name,Speciality,Course,Pubs}(Ans1 \bowtie_{Name=A\text{-}Name} (z_2))$$

$$z \leftarrow \mathbf{SP}_{Pub\text{-}Table=(Name,D\text{-}Name,A\text{-}Info,Pubs),table}(\mathbf{SP}_{A\text{-}Info=(Speciality,Course),academic-info}(\mathbf{RP}_{Pubs,pubs}(\mathbf{SDM}_{Pub}(\mathbf{SP^*}_{Pub=(Title,Pub\text{-}Info),pub}(z_3)))))$$

7 Conclusion

Structured documents and databases are widely recognized as important information resources, and they are often maintained in distributed, heterogeneous, and autonomous information repositories. Thus, integration of document repositories and databases is one of the significant issues even though it presents a lot of challenging problems to us.

In this paper, we have presented the NR/SD model as a basic modeling framework for integration of structured documents and relational databases, and have described a query processing and optimization scheme for the NR/SD model in the mediator-based integration framework. The NR/SD model uses the nested relational structures incorporating the structured document type. The converters attain dynamic conversion of structured documents into nested relational structures and vice versa. The query processing and optimization scheme supports the dynamic property of the NR/SD model, paying attention to utilization of local query processing capability and efficient manipulation of structured document data.

Elaborated study of the query processing and optimization scheme incorporating cost models and index utilization is one of important future research

issues. They also include caching, view maintenance, and development of user-friendly query languages. Treatment of grammatical constructs such as exceptions and recursions ignored in the paper is another future research issue. These issues will be discussed in forth-coming papers.

Acknowledgements

This work was supported in part by the Ministry of Education, Science, Sports and Culture, Japan through the Grand-in-Aid for Scientific Research on Priority Areas #08244101.

References

[1] S. Abiteboul, S. Cluet, and T. Milo. Querying and Updating the File. *Proc. 19th VLDB Conf.*, pp. 73-84, 1993.

[2] S. Abiteboul, S. Cluet, and T. Milo. A Database Interface for File Update. *Proc. ACM SIGMOD Conf.*, pp. 386-397, 1995.

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

[4] P. Buneman, S. B. Davidson, K. Hart, and C. Overton. A Data Transformation System for Biological Data Sources. *Proc. 21st VLDB Conf.*, pp. 158-169, 1995.

[5] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A Query Language and Optimization Techniques for Unstructrured Data. *Proc. ACM SIGMOD Conf.*, pp. 505-516, 1996.

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

[7] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From Structured Documents to Novel Query Facilities. *Proc. ACM SIGMOD Conf.*, pp. 313-324, May 1994.

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

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

[10] 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. *Proc. ACM SIGIR Conf.*, pp. 223-232, 1992.

[11] A. K. Elmagarmid and C. Pu, (eds.). Special Issue on Heterogeneous Databases. *ACM Computing Survey*, vol. 22, no. 3, 1990.

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

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

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

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

[16] Kalervo Järvelin and Timo Niemi. An NF^2 Relational Interface for Document Retrieval, Restructuring and Aggregation. *Proc. SIGIR Conf.*, pp. 102-110, 1995.

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

[18] A. Morishima and H. Kitagawa. A Data Modeling Approach to the Seamless Information Exchange among Structured Documents and Databases. *Proc. 1997 ACM Symposium on Applied Computing*, San Jose, Feb. 1997.

[19] E. Pitoura, O. Bukhres, and A. Elmagarmid. Object Orientation in Multidatabase Systems. *ACM Computing Surveys*, vol. 27, no. 2, pp. 141-195, June 1995.

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

[21] 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, pp. 272-283, 1994.

[22] R. Sacks-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, pp. 454-470, June 1995.

[23] A. Salminen and F. W. Tompa. PAT expressions: an algebra for text search. *Acta Linguistica Hungarica* 41, 1-4 (1992-93) 1994.

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

[25] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, pp. 38-49, March 1992.

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

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

[28] M. Yoshikawa, O. Ichikawa, and S. Uemura. Amalgamating SGML Documents and Databases. *Proc. 5th International Conf. on Extending Database Technology*, March 1996.