Integrated Querying and Restructuring of the World Wide Web and Databases

Atsuyuki Morishima
Doctoral Degree Program in Eng., Univ. of Tsukuba
mori@dblab.is.tsukuba.ac.jp

Hiroyuki Kitagawa
Inst. of Info. Sci. and Elec., Univ. of Tsukuba
kitagawa@is.tsukuba.ac.jp

Abstract

Today, a huge number of information sources are available via computer networks. Thus, integration of heterogeneous information sources has been one of the most important issues in recent advanced application environments. In particular, with the broad acceptance of the World Wide Web (or Web), there is a great deal of demand for integration of the Web and databases. In this paper, we propose *WebNR/SD* as a basic framework for integration of the Web, relational databases, and structured document repositories. WebNR/SD combines an abstract data type named the *structured document type* and the nested relational structures to accommodate structured documents (Web pages) and relations. WebNR/SD also has a data type named *Hlink type* to deal with hyper-text links among Web pages, and operators to dynamically import and export Web pages on demand. WebNR/SD can dynamically convert Web pages into nested relational structures and vice versa. In this paper, we focus on integration of the Web and relational databases, and amalgamation of data in Web pages and relations is explained. In addition to querying different information sources, we can construct hyper-text views over the underlying information sources.

1 Introduction

Today, a huge number of information sources are available via computer networks. Thus, integration of heterogeneous information sources has been one of the most important issues in recent advanced application environments [4][5][9][15][24]. In particular, with the broad acceptance of the World Wide Web (or Web), there is a great deal of demand for integration of Web information resources and databases [16]. Web consists of a lot of structured documents (or *pages* in the context of Web), written in HTML. HTML documents are SGML documents whose inner document structures are prescribed by a predefined document type definition. In addition, XML (*Extensible Markup Language*)[10] has been developed as a next generation language for Web page description. XML documents are also SGML documents, and allow Web pages to have user-defined document types.

In the previous work, we proposed a data model, named *NR/SD*, as a basic framework for integration of structured document repositories (such as SGML files) and relational databases. We studied basic properties of its operators, and developed a query processing and optimization scheme in the integration environment [20] [21]. In this paper, we incorporate the Web into our integration scheme.

Data modeling constructs of NR/SD are nested relations [13] and an abstract data type named the *structured document type* (or shortly, SD type). NR/SD treats raw structured documents as values of SD type. NR/SD provides the nested relational algebra operators, and a number of functions associated with SD type to retrieve text elements contained in structured documents. Thus, NR/SD provides basic constructs to model data both in structured documents and relational databases, and operators for their manipulation.

NR/SD features operators, called *converters*, to dynamically convert structured documents into nested relational structures and vice versa. Thus, we can represent the same data either as atomic values of SD type (structured documents) or as nested relational structures. 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. With NR/SD, we can retrieve data in the form of structured documents or relations, and can also restructure data objects to develop various views on top of the stored structured documents and relational databases.

In [22], we extended NR/SD to *NR/SD+*. The main difference is that converters are more primitive in NR/SD+. Therefore, NR/SD+ is more flexible to accommodate various structures which occur in SGML documents. Moreover, in NR/SD+, converters can directly manipulate text elements inside structured documents. These features make NR/SD+ more suitable for practical integration scenarios.

In this paper, we propose *WebNR/SD* to incorporate the Web as an information source in addi-

International Symposium on Digital Media Information Base (DMIB'97), Nara, Japan, November 1997

tion to structured document repositories and relational databases. Although Web contains a collection of structured documents as conventional structured document repositories, it differs from them in the following points: (1) Web pages usually have hyper-text links to relevant pages. (2) The number of Web pages is virtually innumerable and they are autonomously changing. Therefore, it is impractical to manage all the Web pages in the world with a static database scheme.

To cope with this first point, WebNR/SD introduces a new type named the *Hlink type*. To address the second point, we provide *Navigate* and *Import* operators. They enable us to fetch Web pages from the outside Web world on demand. Navigate gets hyper-text links to the Web pages which meet user-specified conditions. Import is to obtain the contents of the Web pages as SD type values.

In addition to Navigate and Import, WebNR/SD features *Export* operator. It exports SD type values as Web pages to the outside Web world. Thus, in addition to querying different information sources, we can construct *hyper-text views* over the underlying information sources.

In the following discussion, we assume that Web pages are written in XML, and that Web pages have user-defined inner document structures. This enables us to use tag information more effectively taking care of user-specified data semantics.

The rest of this paper is organized as follows. Section 2 presents an application scenario of WebNR/SD. Section 3 gives brief description of NR/SD+, the basis of WebNR/SD. Section 4 defines data types and operators specific to WebNR/SD. Section 5 shows how data manipulation in the application scenario can be attained in WebNR/SD. We briefly survey related works in Section 6, and conclude in Section 7.

2 Application Scenario

2.1 Integration Environment

One of the promising approaches to integrate heterogeneous information sources is to use software modules or agents called *mediators* and *wrappers* [24][27]. We follow this approach to attain integration of the Web, relational databases, and structured document repositories (Figure 1). The mediator acts as a coordinator, and first it dispatches wrappers to information sources[1]. Wrappers for relational databases and document repositories communicate with local information sources directly, while the wrapper for the Web communicates with a number of Web servers to get Web pages. Wrappers provide the mediator with schema information on local information sources. In particular, wrappers for the Web and structured document repositories construct relational views over them. The mediator provides clients with an integrated schema on those information sources. Clients may be user interface modules, or application processes which communicate with the mediator. After the mediator gets the user's request from a client, it analyzes the request, decomposes it into local processing requests, and sends them to wrappers. Each wrapper issues local commands to the local information source. Local commands may be simple requests to get particular data items such as Web page fetches, or complicated queries which utilize querying capability of local information sources, such as SQL commands. The wrapper receives the intermediate result, translates it into WebNR/SD data representation, and sends it back to the mediator. Finally, the mediator collects data from the wrappers and produces the final result, which is returned to the client.

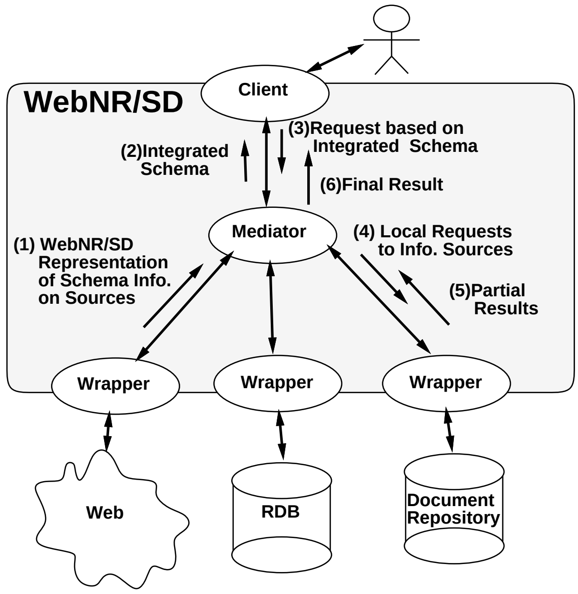

Figure 1: Integration environment

2.2 Example Scenario

In this subsection, we show an example scenario in the integration environment shown in Figure 1 and its technical requirements. Assume that we have a relational database which manages information on a CS department of some university. It contains a relation "Member" (Figure 2(a)). Also, there are Web pages originating in "CS department home page" (Figure 2(b)). The department has a number of research groups. Each group has Web pages which list publications of the group for each year (we call them PL pages). They are indirectly linked from the CS department home page (Figure 2(c)). PL pages of different groups may have different inner page structures. For example, every PL page of Group A categorizes publication items by projects, while that of Group B annotates each publication item with its abstract. However, every publication item in all PL pages is assumed to have substructure shown in Figure 3. Figure 4 shows examples of publication items.

The requirement here is to get a *hyper-text view* over those information sources. It consists of new Web pages (new PL pages), each of which lists publications of a faculty member (Figure 2(d)). It also has an index Web page which contains hyper-text links to the new PL pages and the faculty members' information originally stored in the relation "Member" (Figure 2(e)).

[1] In this context, we do not assume that wrappers always reside at local information sources as in TSIMMIS [24].

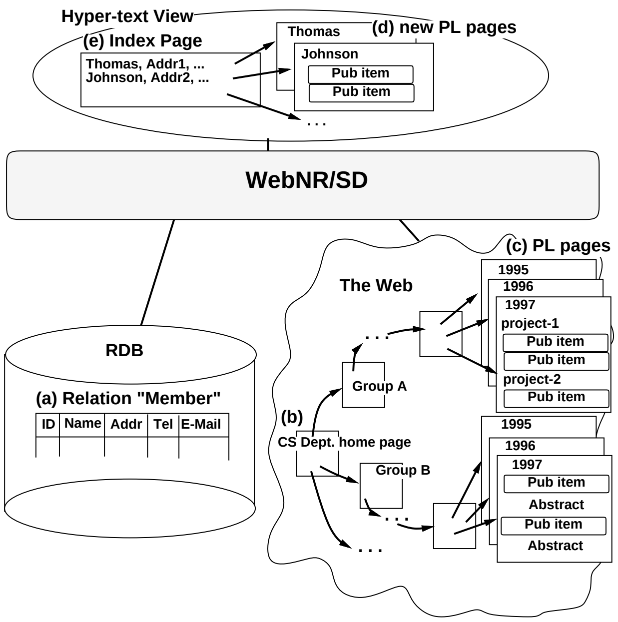

Figure 2: Hyper-text View over the Web and a Relational Database

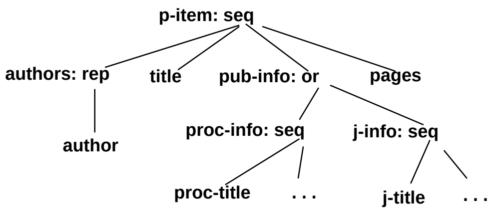

Figure 3: Structure of a publication item

```
<p-item><authors><author>T. Johnson</author>
<author>G. Mark</author></authors><title>
On structured documents</title><pub-info><proc-
info><proc-title>CODAS</proc-title><date><month>
Nov.</month><year>1996</year></date></proc-info>
</pub-info><pages>128-187</pages></p-item>
    . .
<p-item><authors><author>H. Smith</author>
</authors><title>Access control in a multidatabase
</title><pub-info><j-info><j-title>A-journal
</j-title><vol>1</vol><no>7</no><date><month>June
</month><year>1994</year></date></j-info></pub-
info><pages>32-41</pages></p-item>
    . .
```

Figure 4: Sample publication items contained in PL pages

This example scenario requires the following problems to be solved.

P1 To develop an integrated schema over the Web and the relational database.

P2 To amalgamate data objects originally contained in the Web pages and the relational database.

P3 To cope with heterogeneous structures in the original PL pages.

P4 To restructure data objects and construct new Web pages and new hyper-text link structures.

Our approach to the problems in WebNR/SD is addressed in the following sections. Concrete data manipulation steps in WebNR/SD are shown in Section 5.

3 NR/SD+

In this section, we briefly present NR/SD+, the basis of WebNR/SD. Note that we focus on points indispensable for introducing WebNR/SD. Therefore, the description of NR/SD+ here is not complete. More detailed description is given in [22].

3.1 Data Structure

As mentioned before, nested relational structures and the structured document type are basic data modeling constructs of NR/SD+. Figure 5 shows example relation r_0. The domains of attributes A and D are String and Integer, respectively. The domains of B and E are *the structured document type*, explained below.

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

Figure 5: Relation r_0

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 6 shows an example SD value. The DTD in the upper box represents the document structure. Inside the lower box is the tagged text. A tagged text is divided into *elements* surrounded by a *start tag* <*g*> and an *end tag* </*g*>, where *g* is a *generic identifier* representing the *element type*. Elements can be nested within other elements. The DTD prescribes how the elements can be hierarchically constructed by sub-elements.

In Figure 6, each line in the DTD is an *element type definition*. An element "pl-page-B" is a sequence of "page-title," "year," and "pub-list" (**seq** structure). An element "pub-list" consists of zero or more "list-item" elements (**rep** structure). An element "pub-info" contains either "proc-info" or "j-info" (**or** structure). An element "proc-info" has **seq** structure and can contain at most one "venue" element (**opt** structure). The element types such as "page-title," "year," and "author" have no internal structures (**text** structure). We usually omit their definitions such as page-title=**text** for simplicity.

An integrated schema in NR/SD+ represents relations stored in the databases as they are, and a set of structured documents stored in each document repository as a unary relation with an SD type attribute.

pl-page-B	=	**seq**(page-title, year, pub-list)
pub-list	=	**rep**(list-item)
list-item	=	**seq**(p-item, abstract)
p-item	=	**seq**(authors, title, pub-info, pages)
pub-info	=	**or**(proc-info, j-info)
proc-info	=	**seq**(proc-title, **opt**(venue), date)
j-info	=	**seq**(j-title, vol, no, date)
authors	=	**rep**(author)
date	=	**seq**(month, year)

```
<pl-page-B>
<page-title>Publication-List 1997</page-title><year>
1997</year><pub-list><list-item><p-item><authors>
<author>A. white</author><author>T.Jackson</author>
</authors><title>Integration of Web and DB</title>
<pub-info><proc-info><proc-title> ...
...
```

Figure 6: Sample SD value

3.2 Converters

NR/SD+ features two *converters*: *Unpack* and *Pack*. The converters transform SD values into nested relational structures and vice versa. *Unpack* constructs sub-relation structures which store text elements originally contained in SD values. The text element specification contained in the parameter of Unpack specifies which elements of the SD values are to be contained in the sub-relations. Conversely, *Pack* constructs SD values from sub-relation structures containing text elements. We first explain text element specification in Unpack and then the converters.

Text element specification

We use expressions of *the region algebra* [7] to specify which text elements are to be contained in sub-relation structures in Unpack. A *region* is a contiguous part of text. In this context, we only consider regions which correspond to text elements. The region algebra is a set-at-a-time algebra. An expression e of the region algebra is generated by the following rule:

$$\begin{aligned} e \to\ & R_i|e\cup e|e\cap e|e-e \\ & |e\supset e|e\subset e|e<e|e>e|\sigma[w](e)|(e) \end{aligned}$$

where R_i can be a generic identifier (e.g. “date”). The region algebra expression “*gi*” returns the set of *gi*-typed text elements. For example, for the tagged text “<c> <a> w1 w2   w3   w4 w1   w1 w5 </c>”, the region algebra expression “a” returns {“<a> w1 w2   w3   w4 w1 ”}, and “b” returns the set of text elements {“ w1 w2 ,” “ w3 ,” “ w4 w1 ,” “ w1 w5 ”}.

Union ($\cup$), intersection ($\cap$), difference ($-$) are ordinary set operators. The *including* ($\supset$), *included* ($\subset$), *follows* ($>$), and *precedes* ($<$) operators are defined as follows:

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

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 same tagged text as the above example, the region algebra expression “$\sigma['w1'](b\subset a)$” returns {“ w1 w2 ,” “ w4 w1 ” }.

Unpack

Unpack (**U**) constructs sub-relation structures which store text elements originally contained in SD values. Figure 7 gives an example of Unpack, where

$$r_2 := \mathbf{U}_{B\to C(O,D[\sigma['T1'](c\subset(b\cup f))])\ as\ x}(r_1).$$

This example constructs sub-relation structures for attribute C (with sub-attributes O and D) in the result relation r_2. Attribute D includes SD values representing text elements which are extracted from SD values of attribute B in r_1, according to the text element specification $\sigma['T1'](c\subset(b\cup f))$. Thus, SD values in attribute D represent such c-type text elements that they are contained in text elements of type b or f and they contain the word “T1.”

SD values in attribute B of relation r_2 contain *SD references*, denoted by “&x.n;.” SD references refer to SD values stored in the sub-relation structures. The header of SD reference (i.e. “x” in this example) is specified as a parameter of Unpack. In r_2, we call SD values in attribute B *master SD values* (or *masters*), and those in attribute D *derivative SD values* (or *derivatives*) (Figure 8).

r_1:

A	B
1	⟨ b:**rep**(c), “<c>T1 T2</c><c>T3 T4</c><c>T1 T5</c>” ⟩
2	⟨ a:**seq**(d:**seq**(c,c),b:**rep**(c)), “<a><d><c>T1</c><c>T2</c></d><c>T6</c><c>T1</c>” ⟩
3	⟨ e:**seq**(b:**rep**(c), f:**seq**(c,g)), “<e><c>T1</c><c>T4 T6</c><f><c>T7 T1</c><g>T3</g></f></e>” ⟩

r_2:

A	B	C	
		O	D
1	⟨ b:**rep**(c),“&x.1;<c>T3 T4</c>&x.2;” ⟩	1	⟨ c, “<c>T1 T2</c>” ⟩
		2	⟨ c, “<c>T1 T5</c>” ⟩
2	⟨ a:**seq**(d:**seq**(c,c),b:**rep**(c)), “<a><d><c>T1</c><c>T2</c></d><c>T6</c>&x.1;” ⟩	1	⟨ c, “<c>T1</c>” ⟩
3	⟨ e:**seq**(b:**rep**(c), f:**seq**(c,g)), “<e>&x.1;<c>T4 T6</c><f>&x.2;<g>T3</g></f></e>” ⟩	1	⟨ c, “<c>T1</c>” ⟩
		2	⟨ c, “<c>T7 T1</c>” ⟩

Figure 7: Relations r_1 and r_2

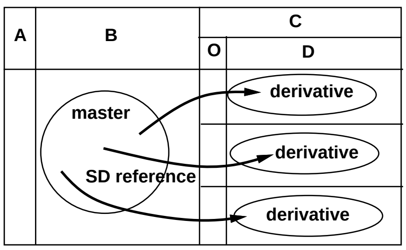

Figure 8: Master and derivatives

Pack

Pack ($\mathbf{P}$) constructs SD values from sub-relation structures containing text elements. Figure 7 gives an example of Pack, too, where

$$r_1 := \mathbf{P}_{C(O,D)\ as\ x \to B}(r_2).$$

This Pack restores original SD values in attribute B from sub-relations in attribute C of r_2 and masters in attribute B. The masters are used as templates, and SD references are replaced with text elements in derivatives.

3.3 Master Constructors

Pack operator requires masters in constructing SD values from sub-relation structures. Although masters could be created by Unpack, as shown above, sometimes we need other masters in order to get SD values which have other inner document structures. Also, if we want to construct SD values from sub-relations which are *not* directly created by Unpack, it is necessary to explicitly provide masters. *Master constructors* can be used to create masters which have DTDs and SD references consistent with existing derivatives. Here, we explain *Sequence master constructor* (**SC**) and *Repetition master constructor* (**RC**). **SC** provides **seq**-structured masters, while **RC** provides **rep**-structured masters.

Figure 9 gives examples of **SC** and **RC**, where

$$r_4 := \mathbf{SC}_{C(O,D)\ as\ x, B, G}(r_3), \text{ and}$$

$$r_6 := \mathbf{RC}_{C(O,D)\ as\ x, B, G}(r_5).$$

In both examples, attribute G serves to designate generic identifiers of the top text elements of new masters in attribute B.

r_3:

A	C		G
	O	D	
1	1	⟨ a, "<a>T1" ⟩	f
	2	⟨ b, "T2"⟩	
	3	⟨ c, "<c>T3</c>"⟩	

r_4:

A	C		B
	O	D	
1	1	⟨ a, "<a>T1" ⟩	⟨ f:**seq**(a,b,c), "<f>&x.1;&x.2;&x.3;</f>"⟩
	2	⟨ b, "T2"⟩	
	3	⟨ c, "<c>T3</c>"⟩	

r_5:

A	C		G
	O	D	
1	1	⟨ a, "<a>T1" ⟩	f
	2	⟨ a, "<a>T2"⟩	
	3	⟨ a, "<a>T3"⟩	

r_6:

A	C		B
	O	D	
1	1	⟨ a, "<a>T1" ⟩	⟨ f:**rep**(a), "<f>&x.1;&x.2;&x.3;</f>"⟩
	2	⟨ a, "<a>T2"⟩	
	3	⟨ a, "<a>T3"⟩	

Figure 9: Examples of master constructors

3.4 NR/SD+ Algebra

NR/SD+ algebra includes converters, master constructors, ordinary nested relational algebra operators (Figure 10) and *Domain translator*, as primitive operators. Domain translator $\gamma_{Attr,type}(r)$ changes the domain of attribute *Attr* into *type*. When *type* is SD type, a generic identifier must be also specified. For example, $\gamma_{A,SD(para)}(r)$ changes the domain of attribute A into SD type and value "v" in the attribute A into SD value ⟨ para, " <para>v</para> "⟩.

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 10: Nested relational algebra operators

Composite operators are defined as combinations of the primitive operators. Join operator $\bowtie_p$ is among them. The followings are variations of converters defined as composite operators. They have different parameter specification syntax.

- $\mathbf{U}_{A_i \to (B_1[e_1]\ as\ x_1, \ldots, B_n[e_n]\ as\ x_n)}(r)$ is an extended version of Unpack. It represents a sequence of primitive Unpacks. Figure 11 shows an example, where $r_8 := \mathbf{U}_{B \to (C[month]\ as\ x, D[year]\ as\ y)}(r_7)$. Masters are in attribute B in the result relation r_8, and derivatives are in new attributes C and D.

- $\mathbf{P}_{(A_i,\ldots,A_j) \to B:SC,G}(r)$ involves master constructor **SC** and Pack. It creates new SD values in attribute B using SD values in attributes $A_i, \ldots, A_j$ as derivatives. $\mathbf{P}_{(C,D) \to B:SC,G}(r_9)$ to the relation r_9 in Figure 12 yields relation r_7 in Figure 11.

- $\mathbf{P}_{A_i(D,\leq) \to B:RC,G}(r)$ involves master constructor **RC** and Pack. It creates new SD values in attribute B using SD values in attribute D as derivatives. If we do not care about the order of derivatives in new SD values, ordering specification $\leq$ can be omitted. Figure 13 shows an example, where $r_{11} := \mathbf{P}_{C(D) \to B:RC,G}(r_{10})$.

Moreover, for all operators which require attribute names as their parameters, we can explicitly give constant values instead of attribute names. They are defined as composite operators. For example, $\mathbf{P}_{(A_i,\ldots,A_j) \to B:SC,'gi'}(r)$ specifies generic identifier $'gi'$ directly instead of attribute name G. This is defined as $\mathbf{P}_{(A_i,\ldots,A_j) \to B:SC,G}(r \times gi_{(G)})$ where $gi_{(G)}$ denotes a unary relation with attribute G, containing only one value $'gi'$.

r_7:

A	B
1	⟨ date:**seq**(month, year), "<date><month>June</month> <year>1970</year></date>"⟩

r_8:

A	B	C	D
1	⟨ date:**seq**(month, year), "<date>&x.1;&y.1;</date>"⟩	⟨ month, "<month>June</month>"⟩	⟨ year, "<year>1970</year>"⟩

Figure 11: Relations r_7 and r_8

r_9:

A	C	D	G
1	⟨ month, "<month>June</month>"⟩	⟨ year, "<year>1970</year>"⟩	date

Figure 12: Relation r_9

r_{10}:

A	C D	G
1	⟨ member, "<member>T1</member>" ⟩	members
	⟨ member, "<member>T2</member>"⟩	
	⟨ member, "<member>T3</member>"⟩	

r_{11}:

A	B
1	⟨ members:**rep**(member), "<members><member>T1</member> <member>T2</member> <member>T3</member></members>"⟩

Figure 13: Relations r_{10} and r_{11}

4 WebNR/SD

WebNR/SD is an extension of NR/SD+. WebNR/SD introduces *Hlink type* to deal with hyper-text link structures on Web pages, and four additional operators.

4.1 Hlink Type

First, we introduce a new element structure **hlink** into SD type. Elements with **hlink** structure are called *linking elements* and represent hyper-text links to Web pages just as anchor elements in HTML documents [2]. An element with **hlink** structure is assumed to have no sub-elements, like elements with **text** structure. The **hlink** structure is associated with the attribute "href," whose value designates a URL. For example, "`<a href="http://T.ac.jp/p1.xml">T Univ</a>`" is an element with **hlink** structure. A value of *Hlink type* (an Hlink value) is defined as an SD value which consists of only one element with **hlink** structure. Since Hlink values are also SD values, operators applicable to SD values can also be applied to Hlink values.

4.2 WebNR/SD Algebra

WebNR/SD algebra consists of NR/SD+ algebra operators and four additional operators: Export, Import, Navigate, and URL generator.

Export and Import

Export and *Import* incorporate the Web into our framework. Export (**E**) exports SD values stored in a relation to the Web world as Web pages. Import (**I**) imports Web pages residing in the Web world into a relation as SD values. Figure 14 gives an example of Export and Import, where

$$r_{13} := \mathbf{E}_{B,U,L,G}(r_{12}), \text{ and}$$

$$r_{12} := \mathbf{I}_{B,U,L,G}(r_{13}).$$

$\mathbf{E}_{B,U,L,G}(r_{12})$ creates Web pages whose URLs are given in attribute U. The Web pages' contents correspond to SD values stored in attribute B of r_{12}. Attribute B in the result relation r_{13} has Hlink values referring to those pages and containing character strings originally stored in attribute L. $\mathbf{I}_{B,U,L,G}(r_{13})$ works in the opposite direction. Attribute B in the result relation r_{12} obtains SD values corresponding to Web pages which are referred to by Hlink values stored in attribute B of r_{13}.

r_{12}:

A	B	U	L	G
1	⟨ b:**seq**(c:**rep**(d),e) , "<c><d>T1 T2</d> </c><e>T3</e>" ⟩	http://.../index.xml	Page A	a

r_{13}:

A	B
1	⟨ a:**hlink**, " Page A" ⟩

Figure 14: Relations r_{12} and r_{13}

Navigate operator

Navigate (**N**) navigates the Web according to a *path regular expression* specified as a parameter of this operator. In path regular expressions, hyper-text links are represented by $\rightarrow$ (a link whose destination and source pages reside in the same Web server) and $\Rightarrow$ (a link whose destination and source pages reside in different Web servers), while Web pages are represented by character strings (*labels*) and a period. For example, $B \rightarrow . \Rightarrow C \rightarrow D$ is a path regular expression that

[2]Actually, linking elements are defined as elements having "xml-link" attribute in XML. In this paper, we assume that **hlink** structure is used to represent linking elements for simplicity.

represents the set of paths that start a Web page B in a Web server X, followed by a page in the same server X and a page C in a different server Y($\neq X$), and end with a page D in the server Y.

If there is hyper-text link structure shown in Figure 15 in the Web, Figure 16 gives the result of Navigate under the above path regular expression, where

$$r_{15} := \mathbf{N}_{B\to.\Rightarrow C\to D,E}(r_{14}).$$

Labels in path regular expressions are used to associate Web pages with relational attributes. Figure 16 shows two important points: (1) Attribute E of the result relation r_{15} contains the set of paths specified by the path regular expression, where each page on the paths is represented by an Hlink value referring to the Web page instead of the contents of the page itself. (2) Links to those pages corresponding to the period do not appear in sub-relations of attribute E.

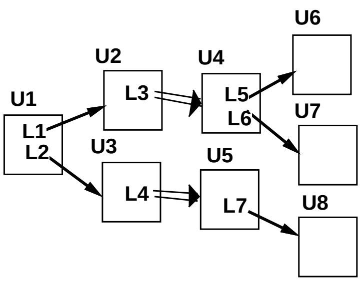

Figure 15: Example hyper-text link structure (Un are URLs and Ln are character strings in linking elements)

Path regular expressions can represent alternation and repetition structures. For example, $A(\to .)* \to B \mid A \Rightarrow B$ represents the set of paths which start with a page A, followed by one or more local links leading to B, or from a page A to B via a global link. '$*/n_1..n_2/$' can be used as syntactic sugar. For example, $A(\to .) * /2..3/ \to B$ is equivalent to $A \to . \to . \to B \mid A \to . \to . \to . \to B$.

We can also specify selection conditions on Web pages in path regular expressions. The selection condition is given by a region algebra expression and placed just after a label or a period. The selection condition holds if the region algebra expression returns a non-empty set of text elements for the Web page bound to the preceding label or period. For example, assume that the contents of the Web pages at URLs "U4" and "U5" in Figure 15 are ⟨ e:**seq**(a:**hlink**,b:**or**(c, d),a:**hlink**), "`<e> <a href="U6"> L5 </a> <b> <c> w1 </c> </b> <a href="U7"> L6 </a> </e>`" ⟩, and ⟨ f:**seq**(b:**or**(c, d), a:**hlink**), "`<f> <b> <d> w2 </d> </b> <a href="U8"> L7 </a></f>`" ⟩, respectively. Then, $\mathbf{N}_{B\to.\Rightarrow C[b\supset c]\to D,E}(r_{14})$ returns the relation which is same as r_{15} except that its attribute E contains only the first two subtuples of r_{15}.

Figure 17 shows the BNF specification of path regular expressions [3].

[3] In this paper, we assume that every path specified by a path regular expression gives non-empty binding to all the labels. $A \to B \mid A \to C$ is an example which violates the restriction.

URL generator

URL generator (**URL**) generates new unique URLs. It can be used before Export operation in creating new Web pages. An example of **URL** is shown in Figure 18, where

$$r_{17} := \mathbf{URL}_U(r_{16}).$$

$\mathbf{URL}_U(r_{16})$ extends relation r_{16} by adding a new attribute U, which contains generated URLs.

5 Query Specification

In this section, we apply WebNR/SD to the example scenario described in Section 2. First, we briefly outline how to deal with the problems **P1** through **P4** mentioned in Subsection 2.2 in WebNR/SD.

P1 In WebNR/SD, the Web is modeled as a collection of Hlink values. The Hlink values work as anchor points to fetch necessary Web pages with Navigate and Import operators. In the example scenario, we use "CS department home page" as a starting point to fetch Web pages. The Web wrapper provides a unary relation "WWW" which only contains an Hlink value referring to the "CS department home page" (Figure 19). Relation "Member" in the relational database is represented as it is. The relation "WWW" and "Member" form the integrated schema.

P2 Once Web pages are imported, they are represented as SD values. Thus, amalgamation of data in Web pages and the relational database is achieved by converters, Domain translator, and nested relational algebra operators.

P3 In spite of structural difference among the original PL pages, Unpack allows us to extract publication items from the PL pages with a text element specification. The operation would be $\mathbf{U}_{A\to B(O,C[p\text{-}item])\ as\ x}(r)$ if attribute A stores the original PL pages.

P4 Construction of new Web pages can be attained by a combination of converters, nested relational algebra operators, and master constructors. First, we construct (sub-)relational structures which store text elements originally contained in PL pages by using Unpack operator. Next, we restructure them using operators such as join and nest. Finally, we transform the relational structures into SD values using master constructors and Pack. Export exports them to the outside Web world and also obtains Hlink values referring to those pages. The Hlink values are used as subelements of the index page. Thus, new hyper-text link structures are created in a similar way.

r_{14}:

A	B
1	⟨ a:**hlink**, “” ⟩

r_{15}:

A	B	E	
		C	D
1	⟨ a:**hlink**, “” ⟩	⟨ a:**hlink**, “L3” ⟩	⟨ a:**hlink**, “L5” ⟩
		⟨ a:**hlink**, “L3” ⟩	⟨ a:**hlink**, “L6” ⟩
		⟨ a:**hlink**, “L4” ⟩	⟨ a:**hlink**, “L7” ⟩

Figure 16: Relations r_{14} and r_{15}

```
path_reg_expr ::= label p_expr { '|' label p_expr }
p_expr        ::= link page [ '[' condition ']' ]
                | p_expr { p_expr }
                | p_expr { '|' p_expr }
                | p_expr '*' [ '/' num '..' num '/' ]
                | '(' p_expr ')'
link          ::= '->' | '=>'
page          ::= label | '.'
```

Figure 17: Path regular expression syntax

r_{16}:

A	B
1	b1
2	b2

r_{17}:

A	B	U
1	b1	http://.../new/1
2	b2	http://.../new/2

Figure 18: Relations r_{16} and r_{17}

Figure 20 shows the element type definitions for publication items shown in Figure 3, contained in original PL pages. We assume that every PL page has a text element “page-title” which contains the word “publication-list.” Figures 21 and 22 show DTDs for the index page and new PL pages, respectively. Then, the data manipulation request in Section 2 can be specified as follows. In the following WebNR/SD algebra expressions, we omit the domain translators for notational simplicity. Also, we may omit text element specifications, headers of SD references, and generic identifiers, in parameters, when there is no possibility of confusion. For example, $\mathbf{U}_{B \to (Authors[authors]\ as\ Authors, Title[title]\ as\ Title)}(r)$ is simply represented as $\mathbf{U}_{B \to (Authors, Title)}(r)$. Also, $\mathbf{P}_{(Author,Pubs) \to P\text{-}List:SC,'p\text{-}list'}(r)$ is represented as $\mathbf{P}_{(Author,Pubs) \to P\text{-}List:SC}(r)$.

(1) First, we import all the PL pages residing in the same Web server as the CS department home page.

$$r_{18} := \mathbf{I}_{D,U,L,G}(\mu_C(\mathbf{N}_{Page(\to.)*\to D[\sigma['publication\text{-}list'](page\text{-}title)],C}(WWW)))$$

(2) Then, relevant text elements are extracted by unpacking the imported Web pages.

$$r_{19} := \mu_{As}(\mathbf{U}_{Authors \to As(O_2, Author)}(\mathbf{U}_{P\text{-}Item \to (Authors, Title, Pub\text{-}Info, Pages)}(\mu_P(\mathbf{U}_{D \to P(O_1, P\text{-}Item)}(r_{18})))))$$

(3) The relation r_{19} is joined with relation “Member,” and unnecessary attributes are dropped.

$$r_{20} := \pi_{Name,Addr,Tel,E\text{-}Mail,Author,Title,Pub\text{-}Info,Pages}(r_{19} \bowtie_{Author=Name} Member)$$

(4) New SD values are constructed following the DTD shown in Figure 22.

$$r_{21} := \mathbf{P}_{P(New\text{-}P\text{-}Item) \to Pubs:RC}(\nu_{P=(New\text{-}P\text{-}Item)}(\mathbf{P}_{(Title,Pub\text{-}Info,Pages) \to New\text{-}P\text{-}Item:SC}(r_{20})))$$

$$r_{22} := \mathbf{P}_{(Author,Pubs) \to P\text{-}List:SC}(r_{21})$$

(5) The SD values are exported as new PL pages, and Hlink values are set to link to those Web pages.

$$r_{23} := \mathbf{E}_{P\text{-}List,U_1,'publications','a'}(\mathbf{URL}_{U_1}(r_{22}))$$

(6) Finally, the index page is constructed following the DTD shown in Figure 21. It is also exported as a new Web page.

$$r_{24} := \mathbf{P}_{Members(Member) \to Table:RC}(\nu_{Members=(Member)}(\mathbf{P}_{(Name,Addr,Tel,E\text{-}Mail,P\text{-}List) \to Member:SC}(r_{23})))$$

$$r_{25} := \mathbf{E}_{Table,U_2,'index\text{-}page','a'}(\mathbf{URL}_{U_2}(r_{24}))$$

Page
⟨ a:**hlink**, “”⟩

Figure 19: Relation WWW

p-item	=	**seq**(authors, title, pub-info, pages)
pub-info	=	**or**(proc-info, j-info)
proc-info	=	**seq**(proc-title, **opt**(venue), date)
j-info	=	**seq**(j-title, vol, no, date)
authors	=	**rep**(author)
date	=	**seq**(month, year)

Figure 20: Element definitions for publication items

table	=	**rep**(member)
member	=	**seq**(name, addr, tel, e-mail, a)
a	=	**hlink**

Figure 21: DTD for the index page

p-list	=	**seq**(author, pubs)
pubs	=	**rep**(new-p-item)
new-p-item	=	**seq**(title, pub-info, pages)
pub-info	=	**or**(proc-info, j-info)
proc-info	=	**seq**(proc-title, **opt**(venue), date)
j-info	=	**seq**(j-title, vol, no, date)
date	=	**seq**(month, year)

Figure 22: DTD for new PL pages

6 Related Work

As demonstrated in Section 5, WebNR/SD features dynamic and symmetric amalgamation of the Web and databases. It can be used for integrated querying and restructuring of information residing in the Web, relational databases, and structured document repositories. Since management of the Web information and integration of heterogeneous information sources are hot topics, there are a number of works related to this study.

Infomaster [14] provides integrated access to multiple distributed heterogeneous information sources. STRUDEL [11] is a Web-site management system which constructs Web pages from underlying information sources. They are similar to WebNR/SD in that they can create Web pages based on different information sources including Web pages, and that they use wrappers to translate information sources into common data representations. Infomaster uses extensions of predicates, and STRUDEL uses an graph data model as pivot data representations. Since they use rather abstract pivot data representations, they could be applicable to various information sources. However, the translation overhead by wrappers is not negligible. Moreover, it is difficult to explicitly distinguish metadata (such as database schemas and DTDs) from data in STRUDEL. WebNR/SD is more dedicated to integration of the Web, relational databases, and structured document repositories, and the translation overhead is far smaller. Metadata management in WebNR/SD would facilitate manipulation of a large volume of data.

There are a number of academic works and commercial implementations in which Web pages are used as the interface to databases [16][23]. There, new Web pages are constructed on the fly to show query results. Techniques such as the CGI and dedicated Web servers are also included in this category. These approaches are very weak in providing common data models for data integration. Thus, it is often the case that, except for querying relational databases and specifying HTML-based query results, users have to write application programs from scratch.

Functions of WebNR/SD include querying of the Web. Recently, a number of languages to query the Web have emerged, although they are not intended for amalgamating the Web and databases. Their queries can be based on hyper-text link structures and/or pages' contents. However, their capabilities of exploiting pages' inner structures are generally much restricted. W3QL [17] and WebSQL [19] are SQL-like query languages, and RAW [12] is an extension of the relational algebra. They are dedicated to querying and provide no way to restructure Web pages. WebLog [18] is a logic-based language which enable us to query and restructure Web pages, exploiting various extensible built-in predicates. However, WebLog deals with Web pages at rather abstract level. A page in WebLog is defined as a set of *rel-infons*, each of which is a group of related information appearing in the page. Mapping between WebLog pages and actual Web pages is outside the scope of WebLog and has to be done by users. Instead, WebNR/SD directly manipulates internal page structures when querying and restructuring Web pages, and query expressions decide complete internal page structures.

To facilitate Web search, there are many index servers that maintain indices for Web pages by using robots and so on. They are mainly used to get sets of Web pages which may match query conditions. We can define an operator to utilize their capability in WebNR/SD. Like Navigate, it can be used to obtain hyper-text links to Web pages. Then, the Web pages are fetched by Import, and further manipulated by other operators.

WebNR/SD is designed as an extension of NR/SD+, a basic modeling framework to integrate structured documents and databases. There are several studies on integration of structured documents and databases [1][2][3][6][8][25][26][28]. Abiteboul and others [1][2] proposed the notion of structuring schema which specifies how data contained in text can be incorporated into databases. Atlas [25] 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 [6]. COINS [8], Volz and others [26], and Yan and others [28] 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 NR/SD+ in the light of those studies are as follows.

- NR/SD+ provides a framework for *symmetrically* and *dynamically* amalgamating 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 *partial 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.

WebNR/SD inherits these features from NR/SD+ and incorporates the Web into its integration scheme.

Yoshikawa and others proposed another approach to the integration of structured documents and databases [29]. Their approach is to provide a general mechanism to make predefined reference links from components of SGML documents to database objects and is different from ours.

7 Conclusion

In this paper, we have proposed WebNR/SD to attain integration of the Web, relational databases, and structured document repositories. We have focused on integration of the Web and relational databases in this context. WebNR/SD features Hlink type to deal with hyper-text links among Web pages, Navigate and Import operators to fetch Web pages from the outside Web world on demand, and Export operator to export Web pages. It also has converters to dynamically convert Web pages into nested relational structures and vice versa. In addition to querying different information sources, WebNR/SD provides hyper-text views over the underlying information sources.

Future research issues include user-friendly query languages and user interface, and efficient implementation architecture of mediators and wrappers which execute WebNR/SD operations. They also include more support of features specific to SGML and XML documents which are not considered in this paper, such as attributes of text elements and various kinds of hyper-text links.

Acknowledgement

This work was supported in part by the Ministry of Education, Science, Sports and Culture, Japan through the Grand-in-Aid for Scientific Research #08244101 and #09680321, and by University of Tsukuba Special Project Research.

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] 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.

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

[8] 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.

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

[10] Extensible Markup Language (XML). World Wide Web Consortium Working Draft. `http://www.w3.org/TR/`.

[11] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. STRUDEL: A Web-site Management System. *Proc. SIGMOD Conf.*, Tucson, pp. 549-552, May 1997.

[12] T. Fiebig, J. Weiss, and G. Moerkotte. RAW: A Relational Algebra for the Web. Proc. of the Workshop on Management of Semi-structured Data, pp. 34-41, May 1997.

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

[14] M. R. Genesereth, A. M. Keller, O. M. Duschka. Infomaster: An Information Integration System. *Proc. SIGMOD Conf.*, Tucson, pp. 539-542, May 1997.

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

[16] Ralf Kramer. Databases on the Web: Technologies for Federation Architectures and Case Studies. *Proc. SIGMOD Conf.*, Tucson, pp. 503-506, May 1997.

[17] D. Konopnicki and O. Shmueli. W3QS: A Query System for the World-Wide Web. Proc. of VLDB Conf., pp. 54-65, 1995.

[18] L. Lakshmannan, F. Sadri, and I. Subramanian. A Declarative Language for Querying and Restructuring the Web. Proc. of the 6th Int. Workshop on Research Issues in Data Eng. (RIDE'96), Feb. 1996.

[19] A. O. Mendelzon, G. A. Mihaila, and T. Milo. Querying the World Wide Web. Proc. PDIS'96, pp. 80-91, Dec. 1996.

[20] 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,* (ACM SAC'97), San Jose, pp. 78-87, Feb. 1997.

[21] A. Morishima and H. Kitagawa. A Data Modeling and Query Processing Scheme for Integration of Structured Document Repositories and Relational Databases. *Proc. of the Fifth International Conference on Database Systems for Advanced Applications* (DASFAA'97), Melbourne, pp. 145-154, April 1997.

[22] A. Morishima and H. Kitagawa. Integration of Structured Documents and Databases Featuring Dynamic Creation of References. IPSJ SIG Notes, vol. 97, no. 64, 97-DBS-113-4, 1997.

[23] T. Nguyen and V. Srinivasan. Accessing Relational Databases from the World Wide Web. *Proc. SIGMOD Conf.*, Montreal, pp. 529-540, June 1996.

[24] 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.

[25] 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.

[26] 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.

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

[28] 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.

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