データベースシステム 109－11
（1996. 7. 24）

構造化文書とデータベースの統合利用のためのデータモデル

森嶋 厚行* 北川 博之†
*筑波大学 工学研究科
†筑波大学 電子・情報工学系

概要

近年、異種情報資源の統合利用が重要な課題となっている。特に、構造化文書が広範囲なアプリケーションで利用されるに従い、構造化文書とデータベースの統合利用の必要性が高まっている。本稿では、この統合利用環境におけるデータ操作系と、その応用について述べる。構造化文書を扱うために、抽象データ型である構造化文書型を導入する。文書の構造情報を、抽象データ型のインスタンスレベルとデータベースのスキーマレベルで相互変換するためのオペレータを用意し、さらに、文書構造階層の深部にあるデータを処理するための機構を用意する。この枠組を使えば、構造化文書中のデータとデータベース中のデータをシームレスに扱うことができる。

A Data Model for the Integration of Structured Documents and Databases

Atsuyuki Morishima* Hiroyuki Kitagawa†

*Doctoral Degree Program in Engineering, Univ. of Tsukuba
†Institute of Information Sciences and Electronics, Univ. of Tsukuba

abstract

Integration of heterogeneous information resources has been one of the most important issues in recent advanced application environments. In addition to conventional databases, structured documents have been recognized as important information resources recently. In this paper, we present a data model named the *NR/SD model*, which is used as a basic data modeling framework for the seamless integration of structured documents and relational databases. The NR/SD model combines an abstract data type named the *structured document type* and the nested relational structures, and features operators named *converters* to dynamically convert structured documents into nested relational structures and vice versa. We also introduce a mechanism which allows concise expressions in extracting data embedded in structured documents, and show its applicability.

1 はじめに

近年、ネットワーク環境等における情報の共有の需要が高まるに従って、異種分散情報資源の統合利用が重要な課題となっている [2]。代表的な情報資源であるデータベースに加え、特に最近その重要性を増しているものに、WWW や CALS で利用されている構造化文書がある。構造化文書は、文書構造とそれに従いタグづけされたテキストから構成される。

我々は、構造化文書とデータベースの統合利用環境について研究中であり、統合利用のためのデータモデルとして、NR/SD モデルを提案している [3]。NR/SD モデルでは、構造化文書を扱うための抽象データ型である“構造化文書型”を、入れ子型リレーショナルモデルに導入することにより、構造化文書とリレーショナルデータベースの統合利用をおこなう枠組を提供する。NR/SD モデルの特徴は、構造化文書と入れ子型リレーション構造を、必要に応じて

相互変換するためのオペレータ“コンバータ”(converter)にある。コンバータを適用すれば、構造化文書中に存在する文書構造情報を、スキーマレベルに持ち上げたり、逆にデータベースのスキーマ構造を、構造化文書の文書構造情報としてインスタンスレベルに落すことができる。

したがって、NR/SDモデルを利用すれば、以下のような応用が可能になる。(1) 入れ子型リレーショナルデータモデルのパワーを利用して、構造化文書群の操作を行なう。(2) スキーマ構造の異なるデータベース中のデータを統一的に操作する。(3) WWWのような環境において、データベース検索の結果を構造化文書の形式で求める。(4) 構造化文書に対するビューを定義する。

本稿では、まず、NR/SDモデルの概要を示す。続いて、ある種のデータ操作を簡潔に行なうために、NR/SDモデルの構造化文書型に付属するオペレータを定義する。具体的には、このオペレータは、構造化文書に対して、リージョン代数(region algebra)[1]を適用するためのものである。最後に、NR/SDモデルの応用例を示し、このオペレータを利用しない場合と利用する場合のデータ操作記述を比較する。

2 構造化文書とデータベースの統合利用環境

NR/SDモデルは、典型的には図1のような環境で利用されることを想定している。ここには2種類の情報資源が存在する。一方はリレーショナルデータベースであり、もう一方は構造化文書群が格納されている文書リポジトリである。これらの統合を行なうために、メディエータとラッパ[4]を使用する。メディエータはユーザに統合スキーマを提供するものであり、ラッパはメディエータの要求に応じて各情報資源のデータを提供するものである。NR/SDモデルの目的は、このような環境で、リレーショナルデータベース中の情報と、構造化文書中の情報を共に利用したデータ操作を行ない、結果を、必要に応じてリレーションもしくは構造化文書のいずれの形式でも得られるようにすることである。

まず、メディエータが、NR/SDモデルを用いてユーザに情報資源の統合スキーマを提供する。この統合スキーマでは、文書リポジトリ中の構造化文書を、抽象データ型である構造化文書型の値として扱う。統合スキーマの例を図2に示す。ここで、属性Cの値が構造化文書型である。このような統合スキーマのもとで、NR/SDモデルでのデータ操作をおこなうことにより、必要な情報を求める。

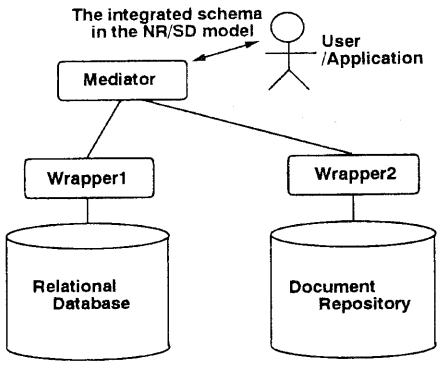

図1. 統合利用環境

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

図2 統合スキーマ例

3 NR/SDモデル

NR/SDモデルは、入れ子型リレーショナルモデルと、抽象データ型である“構造化文書型”(SD型)を組み合わせ、さらに入れ子型リレーション構造と構造化文書を相互変換するためのオペレータである“コンバータ”を導入したものである。ここでは、まず、SD型について述べ、つづいてコンバータについて説明する。

3.1 SD型

SD型の値(SD値)は、文書構造を表すDTD(Data Type Definition)と、そのDTDに従ったタグ付きテキストの組である。SD値の例を図3に示す。図3では上部がDTD、下部がタグ付きテキストを表す。タグ付きテキスト中で、開始タグ <*gi*> と終了タグ </*gi*> に囲まれた部分を要素(element)と呼び、*gi*を要素のジェネリック識別子と呼ぶ。各要素の定義は、DTDの各行で行なう。図3の例では、要素“memo”は“prolog”と“body”からなる列で構成され、“body”は、“para”の0個以上の繰返しで構成されることを表す。また、“to”は、“faxno’もしくは“mailaddr”から構成されることを示す。“para”等の左辺に現れない要素は、タグなしテキスト(ptext)であることを示す。

DTDにもとづき要素間の包含関係をグラフ化すると、DAG構造となる(図4)。

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

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

図 3. SD 値

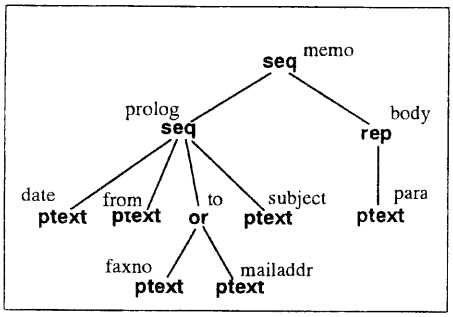

図 4. DTD の DAG 表現

3.2 コンバータ

ここでは、SD 値と入れ子型リレーション構造の相互変換を行なう 6 つのコンバータを説明する。

Rep-unpack, Seq-unpack

Rep-unpack(**RU**)/Seq-unpack(**SU**) は、SD 値を含むリレーションに適用し、SD 値の最上位構造にある繰返し構造 (**rep**) / 列構造 (**seq**) をスキーマレベルに持ち上げる。

以下の 2 式の演算結果を図 5 に示す。図 5 では、SD 値の DTD は、要素定義を直接入れ子にして表現している。例えば図 4 の DTD は、memo: **seq** (prolog: **seq** (date: **ptext**, from: **ptext**, to: **or** ({faxno: **ptext**, mailaddr: **ptext**}), subject: **ptext**), body:**rep**(para:**ptext**)) と表される。

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

Rep-pack, Seq-pack

Req-pack (**RP**) / *Seq-pack* (**SP**) は、SD 値を含むリレーションに適用し、サブリレーション / 属性の部分列を、繰返し構造 (**rep**)/ 列構造 (**seq**) として SD 値の最上位構造に挿入する。以下の演算結果を図 5 に示す。

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

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

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

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

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

図 5. (上から) リレーション r_1, r_2, r_3, r_4

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

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

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

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

図6. (上から) リレーション r_5, r_6, r_7, r_8

なお、$\mathbf{RP}_{B,g}(r_2)$ のように、パラメータとして明示的にジェネリック識別子を指定する場合は、r_2 における属性 E のような、ジェネリック識別子用の属性は用いない。

Or-remove

Or-remove (**OR**) は SD 値を含むリレーションに適用し、SD 値の最上位構造にある選択構造 (**or**) を除去する。**OR** を利用して、最上位構造を **rep** もしくは **seq** にすれば、その結果に対して、さらに Rep-unpack/Seq-unpack を適用できる。例えば、図 6 のリレーション r_5 に次式を適用すると、Seq-unpack を適用可能なリレーション r_6 が得られる。

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

Or-append

Or-append (**OA**) は SD 値を含むリレーションに適用し、SD 値の最上位構造に、選択構造 (**or**) を追加する。**OA** を利用すれば、SD 値群の DTD を統一できるので、さらに Rep-pack オペレータを適用できる。例えば、図 6 のリレーション r_7 は複数の DTD を持つので、そのままでは Rep-pack オペレータを適用できない。次のように **OA** を適用すれば、Rep-pack を適用可能なリレーション r_8 が得られる。

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

3.3 NR/SD 代数

NR/SD 代数は、前述のコンバータの他に、通常の入れ子型リレーショナル代数オペレータ (図 7)、複合オペレータ、型変換オペレータからなる。複合オペレータは、各単項オペレータを拡張したもので、パラメータで属性指定する際に、入れ子型リレーションの内部属性を直接指定できる。例えば、複合オペレータ $\mathbf{RP*}_B(r)$ の B は入れ子リレーションの任意の深さの内部属性であってよい。

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	$r_1 - r_2$

図 7. 入れ子型リレーショナル代数オペレータ

型変換オペレータ $\mathbf{DT}_{att,type}(r)$ は、リレーション r の属性 att の型を $type$ に変更する。$type$ がSD型の場合は、ジェネリック識別子も共に指定する。例えば、$\mathbf{DT}_{A,SD(para)}(r_1)$ では、リレーション r_1 の属性 A の値 "1" が、SD型の値 ⟨ para=**ptext**, " <para>1 </para> "⟩ となる。

4 NR/SD モデルの拡張

コンバータの主要な利用目的には、以下のようなものがある。(1) 文書構造をスキーマ構造に反映することにより、文書のリストラクチャリングをおこなう。(2) スキーマ構造の一部をインスタンス中に隠蔽し、異なるスキーマ構造を持つデータを統一的に操作する。(3) 文書を細分化していくことによって、文書構造下位にある部分文書を抽出する。(3) の場合、目的の部分文書に到達するまで、構造化文書の最上位構造から順にコンバータを適用していくことになる。しかしこの操作は、複数のDTDが混在する場合や目的の部分文書が文書構造の深部にある場合には、非常に複雑なものとなる。その上、その部分文書を得る過程で得られた入れ子型リレーション構造はあくまでも副生成物であり、本来不必要なものである。

この問題点を緩和するために、本節では、リージョン代数 [1] をNR/SD代数に導入する。ここでは、まずリージョン代数について説明し、つづいて、導入するための機構を述べる。具体的には、SD型に付随するオペレータを用いて、構造化文書にリージョン代数式を適用する。

4.1 リージョン代数

リージョン代数 (region algebra) は、リージョンの集合に対する演算をおこなう代数である。リージョンとは、文書の部分文字列を表すもので、組 ⟨ 開始位置, 終了位置 ⟩ で定義される。開始位置と終了位置を先頭からの単語数で表すと、文書 "Integration of Structured Documents and Databases" に対するリージョン $\langle 3,5 \rangle$ は、"Structured Documents and" を表す。

リージョン代数式 e は以下で定義される。

$$\begin{aligned} e \rightarrow \quad & R_i | e \cup e | e \cap e | e - e \\ & | e \supset e | e \subset e | e < e | e > e | \sigma_p(e) | (e) \end{aligned}$$

R_i はあらかじめ決められたリージョンの集合の名前である。例えば、para という名前で、要素 "para" に対応するリージョンの集合を表現する、と定めることができる。union ($\cup$)、intersection ($\cap$)、defference ($-$) は通常の集合演算である。including ($\supset$)、included ($\subset$) は以下のように定義される。

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

ここで、$r \supset s$ は、リージョン r がリージョン s を一部として含むことを表す。$r \subset s$ は r が s の一部であることを表す。

follows ($>$) と precedes ($<$) は以下のように定義される。

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

$r > s$ は、リージョン r がリージョン s より後に位置することを、$r < s$ は r が s に先行することを表す。

selection (σ) はリージョン集合に対する選択演算を行なう。

4.2 SD 値オペレータ

リージョン代数をNR/SD代数に導入するために、SD型に付随するオペレータを定義する。"get_RG" は、SD値に対してリージョン代数式を適用し、リージョンの集合を返す。また、"get_SD" は、SD値にリージョンを与えると、そのリージョンに対応する部分文書を、SD値として返す。

- get_RG(SD value, region algebra expression) →region-set
- get_SD(SD value, region)→SD value

ここで、SD value はSD値、region は、抽象データ型として用意するリージョン型の値、region-set はリージョン型の属性をもつ単項リレーションをそれぞれ示す。

さらに、これらSD型に付随するオペレータによる操作を可能とするためのApply演算子 (α) を、NR/SD代数に追加する。

$$\alpha_{f,att}(r)$$

r はSD型の属性をもつリレーション、f は、SD型に付随するオペレータ、att は f の適用結果を格納するための新たな属性名である。

5 アプリケーション

ここでは、構造化文書とリレーショナルデータベースを横断した問合せの例を示す。まず、リージョン代数オペレータを使用せず解を求めるための式を示し、続いて、リージョン代数オペレータを導入することによって式の一部が簡潔になることを示す。

図1の状況において、リレーショナルデータベースにはT大学の教職員データが、文書リポジトリには論文データベースが格納されているとする。このとき、NR/SDモデルによる統合スキーマが図8のようになると仮定する。論文データベースは、1論文あたり1構造化文書で格納されており、各論文のDTDは図9で表される。

T 大学教職員データベース Faculty

FID	Name	D-Name	Rank	Speciality

論文データベース Document

Doc

図 8. 統合スキーマ

この時、以下の問合せを行う。

Q. T 大学に現在属している教員が書いた論文のうち、他の論文に参照されているものを求めよ。結果は著者別にまとめ、それぞれ構造化文書とせよ。結果の構造化文書の DTD は、図 10 に示すものとする。

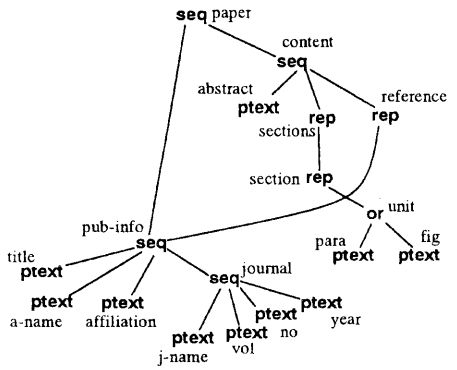

図 9. 論文の DTD

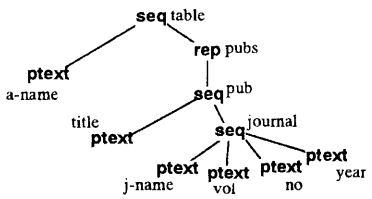

図 10. 要求される DTD

この解を求める操作を表す式は以下のようになる。

step 1 Document 中の構造化文書群から、被参照論文情報 (reference の下位にある pub-info 以下の部分文書) を抽出する。

$$r_1 := \pi_{Pub}(\mu_{Reference}(\mathbf{RU}_{Reference:(O_1,Pub),G_3}(\mathbf{SU}_{Content=(Abstract,Sections,Reference),G_2}(\mathbf{SU}_{Doc=(Pub\text{-}Info,Content),G_1}(Document)))))$$

step 2 被参照論文の情報をリレーションに展開する。

$$r_2 := \mathbf{SU}_{Pub=(Title,A\text{-}Name,Affiliation,Journal),G_4}(r_1)$$

step 3 T 大学に現在属している教員が書いた論文を選択し、さらに、著者名でネスト演算をおこなう。

$$r_3 := \sigma_{A\text{-}Name=Name}(\mathbf{DT}_{Name,SD(a\text{-}name)}(\sigma_{Affiliation=\text{“T-univ”}}(\mathbf{DT}_{Affiliation,String}(r_2)) \times Faculty))$$

$$r_4 := \nu_{Publications=(Title',Title,Journal)}(\pi_{A\text{-}Name,Title',Title,Journal}(r_3))$$

step 4 step3 の結果リレーションを、構造化文書に変換する。

$$r_5 := \mathbf{SP}_{Table=(A\text{-}Name,Publications),table}(\mathbf{RP}_{Publications,pubs}(\mathbf{SP^*}_{Pub=(Title,Journal),pub}(r_4)))$$

リージョン代数式の適用オペレータを利用すると step1 の式が、以下の式 **step 1'** に書き換えられる。

$$r_1 := \pi_{Pub}(\alpha_{get\_SD(Doc,R\text{-}Pub'),Pub}(\mu_{R\text{-}Pub}(\alpha_{get\_RG(Doc.pub\text{-}info\subset reference),R\text{-}Pub}(Document))))$$

コンバータのみを利用した場合、論文 DTD の reference までの構造が変更されると、それに対応した新たな step 1 式が必要となる。しかも、reference までの構造が複雑になるほど、その式もより複雑なものとなる。一方、step 1' 式はその影響を受けないので、無変更で良い。さらに、そのような複数の論文 DTD が混在する状況でもこの式 1 つで対応できる。

6 おわりに

本稿では、構造化文書とデータベースの統合利用のためのデータモデル NR/SD モデルを示し、このモデルを用いて、これらの情報資源を横断したデータ操作が可能であることを示した。また、SD 値に対してリージョン代数式を適用することにより、ある種のデータ操作が簡潔に記述できることを示した。今後の課題は、NR/SD モデルに基づく実際の操作環境の検討や、統合環境の実装アーキテクチャなどである。

参考文献

[1] M. P. Consens and T. Milo, “Algebras for Querying Text Regions,” in *Proc. ACM PODS*, 1995, pp. 11-22.

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

[3] A. Morishima and H. Kitagawa, “A Data Modeling Approach to the Seamless Information Exchange among Structured Documents and Databases,” Technical Report ISE-TR-96-133, University of Tsukuba, 1996.

[4] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman, “MedMaker: A Mediation System Based on Declarative Specifications,” in *Proc. ICDE*, 1996, pp. 132-141.