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+とその問合せ処理

森嶋厚行[†] 北川博之[††]

近年，ネットワークを通じた各種情報源へのアクセスが容易となるに従い，異種分散情報源の統合利用が重要な課題となっている．各種情報源の中でも，リレーショナルデータベースと構造化文書は最も代表的かつ重要な情報源である．特に構造化文書は，電子図書館やWWW，CALSなど幅広いアプリケーションで利用されるようになり，近年その重要性が増大している．本稿では，構造化文書とリレーショナルデータベースを対象とした統合利用環境における，統合データモデルNR/SD+と問合せ処理方式について述べる．NR/SD+は入れ子型リレーション構造に抽象データ型である「構造化文書型」を導入したデータモデルである．リレーション構造と構造化文書型の値を動的に相互変換する演算子を用意することにより，これらの統合利用を実現する．また，本統合利用環境では，次の点を考慮した問合せ処理を行う．(1) 問合せ式が構造化文書とリレーション構造の動的な変換を含む場合であっても，各情報源のローカルな問合せ処理機能の有効利用を図る．(2) 大量の構造化文書のうち統合操作に本当に必要なデータ部分のみに注目することにより，問合せ処理の効率化を図る．

NR/SD+ Data Model and Its Query Processing —For Integration of Structured Documents and Relational Databases

ATSUYUKI MORISHIMA[†] and HIROYUKI KITAGAWA[††]

Integration of heterogeneous information sources 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 sources recently. In this paper, we first present a data model named NR/SD+ as a basic framework for integration of structured documents and relational databases. Then, we discuss a query processing and optimization scheme for environments including structured document repositories and relational databases. NR/SD+ 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. 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.

1. はじめに

近年，ネットワークを通じた各種情報源へのアクセスが容易となるに従い，異種分散情報源の統合利用が重要な課題となっている[10),18)]．各種情報源の中でも，リレーショナルデータベースと構造化文書は最も代表的かつ重要な情報源である．特に構造化文書は，電子図書館やWWW，CALSなど幅広いアプリケーションで利用されるようになり，近年その重要性が増大している．

我々は，構造化文書とリレーショナルデータベースを対象とした異種情報源統合利用方式の研究開発を行っている．異種情報源の統合利用のための主要なアプローチの1つとして，ソフトウエアモジュールであるメディエータとラッパーを利用するものがある[18),23)]．我々もこのアプローチを採用する（図1）．ラッパーは各情報源のデータを統合データモデルに変換し，メディエータはそのモデルに基づいた統合スキーマと操

† 筑波大学大学院工学研究科
Doctoral Program in Engineering, University of Tsukuba
†† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

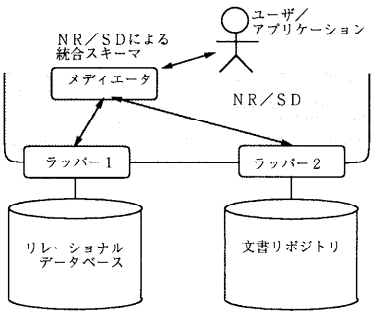


図1 統合利用環境
Fig. 1 Integration environment.

作環境を提供する．我々はこの統合利用環境で利用される統合データモデルNR/SDを提案し，NR/SDの演算子間に成立する基本的な性質や，NR/SDに基づく統合利用環境における問合せ処理方式の研究を進めてきた[15),16)]．本稿では，NR/SDの拡張であり現実の応用により適したモデルであるNR/SD+[17)]と，NR/SD+に基づいた本統合利用環境における問合せ処理と最適化について述べる．

NR/SD+は，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目的とした統合データモデルである．NR/SD+では，入れ子型リレーショナルモデル[11)]に抽象データ型である「**構造化文書型**」(*structured document type*, **SD型**)を導入する．各構造化文書は，それぞれSD型のアトミックな値(**SD値**)として扱われる．NR/SD+では入れ子型リレーショナル代数演算子による演算に加え，SD値に対して文書内容に基づく検索操作が可能である．

NR/SD+の特徴は，構造化文書(SD値)と入れ子型リレーション構造を，動的に相互変換可能なことである．この変換は**コンバータ**(*converter*)と呼ぶ演算子群を用いて行う．コンバータを利用することにより，NR/SD+では以下のようなデータ操作が可能となる．

(1) 必要に応じて，同一のデータに対し入れ子型リレーショナル代数系と文書検索操作の両者の演算を使い分ける．たとえば，以下のような操作を行うことが可能である．(a) 入れ子型リレーショナル代数系を利用して，構造化文書として与えられたデータの構造変換操作等を行う．(b) 入れ子型リレーション構造を構造化文書に変換し，スキーマ構造と独立した検索やメタデータを手がかりとした検索を行う．

(2) 異なる構造のデータを適当な抽象度で抽象化して統一的に操作する．コンバータによる相互変換は，対象データに対して部分的に行うことができる．したがって，データの一部を入れ子型リレーション構造で，他の部分をSD値として持つことが可能である．部分的に異なる構造を持つ複数のデータに対し，その共通部分構造を入れ子型リレーション構造で保持し，非共通構造はSD値として扱うことにより，入れ子型リレーション構造のレベルでは細部の差異を隠蔽した適当な抽象度を持つ同一のスキーマ構造を持たせることができる．

このほかにも，NR/SD+はデータベース検索の結果を構造化文書の形式で求める操作や，構造化文書に対するビューを定義するための機構として利用できる．

本統合利用環境における問合せ処理において，(1) ラッパーが各情報源中の全データをNR/SD+に変換後，メディエータに転送し，(2) メディエータがそれらのデータに対して処理を行う，という単純な手順では，処理上の無駄が多い．本稿では，問合せ処理の際に以下のような方式を用いることにより，処理効率の向上を図ることを提案する．

(1) 問合せを構成する操作のうち，複数の情報源にまたがらない操作については，各情報源の検索機構や問合せ処理機構を用いてローカルに処理を行う．リレーションと構造化文書の相互変換が行われる場合においても各情報源の機能を有効利用するために，入れ子型リレーショナル代数演算と文書内容検索を横断した問合せ変換規則を導入し，問合せ式の変換を行う．

(2) 構造化文書をラッパーからメディエータに転送する際，その転送の一部を，メディエータでの問合せ処理に必要な時期まで遅延する．これによって，メディエータにおける作業領域と処理コストの削減を行う．

本稿は，以下のように構成されている．2章では，構造化文書とリレーショナルデータベースの統合利用例を示す．3章では，NR/SD+のデータ構造と演算子を説明する．4章では統合利用例に対するNR/SD+を用いたデータ操作を示す．5章では問合せ処理と最適化について説明する．6章では，関連研究について述べる．7章では本稿のまとめと今後の課題について述べる．

2. 統合利用例

図2に，構造化文書とリレーショナルデータベースの統合利用の例を示す．リレーショナルデータベースにA大学の教官情報リレーション“Faculty”が格納されている．一方，文書リポジトリには，多数の論文が，それぞれ構造化文書の形式で格納されているとする．

このとき，A大学の各教官ごとに，所属学科，専門分野，担当科目などの教官情報と，その教官によって

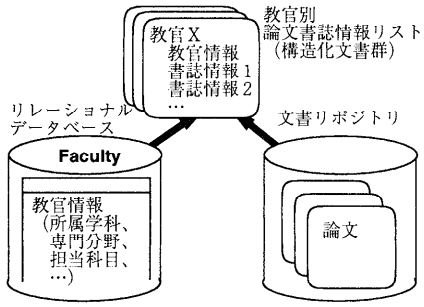


図2 統合利用例
Fig. 2 Application example.

書かれた論文の書誌情報を，すべてまとめた構造化文書を作成したい．さらに，作成された構造化文書群の中から，専門分野や担当科目が database に関連する教官の文書を選択すると仮定する．

NR/SD+では，このデータ操作要求は以下のようなデータ操作の並びとして表現される．より具体的な操作内容については 4 章で示す．

操作 1 論文群から書誌情報を抽出する．
操作 2 抽出した書誌情報と “Faculty” を結合する．
操作 3 要求される構造化文書群を作成する．
操作 4 それら構造化文書群から，専門分野や担当科目が “database” に関連する教官の文書を選択する．

3. NR/SD+

本章では，NR/SD+のデータ構造と演算子について述べる．1 章で述べたとおり，NR/SD+のデータ構造は，入れ子型リレーション構造に抽象データ型である**構造化文書型**（*structured document type*, **SD 型**）を導入したものである．**図 3** のリレーション r_0 において，$dom(A)$（属性 A のドメイン，以下同様）は文字列型，$dom(D)$ は整数型，$dom(B)$ と $dom(E)$ は構造化文書型である．

3.1 構造化文書型

各構造化文書は抽象データ型である SD 型の値（**SD 値**）として扱われる．SD 値は，文書構造を表す DTD（*Document Type Definition*）と，その DTD に従ったタグ付きテキストから構成される．テキスト中で開始タグ <*g*> と終了タグ </*g*> で区切られた部分を**要素**（*element*）と呼ぶ．ここで，g を**共通識別子**（*generic identifier*）と呼び，これが**要素型**を示す．各要素型の構造は，DTD を構成する**要素型定義**で与えられる．**図 4** の SD 値は，paper や title 等の要素から構成されている．paper は，副要素 title や authors 等を並べた列構造（**seq** 構造）を持つ．pub-info は，

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

図3 リレーション r_0
Fig. 3 Relation r_0.

paper = **seq**(title, authors, pub-info, content, ref)
authors = **rep**(author)
author = **seq**(a-name, affiliation)
pub-info= **or**(proc, journal)
proc = **seq**(c-name, month, year)
journal = **seq**(j-name, vol, no, year)
content = **seq**(abstract, **rep**(section))
ref = **rep**(ref-item)
ref-item = **seq**(title, authors, pub-info)
...

```
<paper>
<title>Integration of Heterogeneous
Information Repositories</title><authors>
<author><a-name>T. Jhonson</a-name>
<affiliation>A-Univ</affiliation></author>
<author><a-name>G. Mark</a-name><affiliation>
B-Univ</affiliation></author></authors>
<pub-info><proc><c-name>X Conf.</c-name>
<month>June</month><year>1997</year>
</proc></pub-info><content> ...
```

図4 SD 値
Fig. 4 SD value.

proc(eedings) もしくは journal のどちらかを内部に含む（**or** 構造）．title，a-name などは内部構造を持たない要素である（**text** 構造）．本来 DTD にはこれらの定義（title=**text** など）も存在するが，図 4 ではこれらを省略している．これらのほかにも，SD 値では SGML 文書に現れる様々な文書構造が利用可能である☆．

以下では，表記を簡潔にするために DTD を線形表記することがある．たとえば，DTD “a=**seq**(b, c), b=**rep**(d), c=**and**(e, f)” を “a: **seq**(b: **rep**(d), c: **and**(e, f))” と表記する．線形表記では再帰構造を直接記述できない等の制限があるが，以下の議論は一般の DTD を持つ SD 値についても成立する．また，要素型定義のうち，議論と本質的に無関係なものを省略する．

3.2 コンバータ

2 章に示したような操作を行うためには，SD 値と入れ子型リレーション構造を相互に変換できることが必要である．このための演算子群が**コンバータ**である．

☆ 要素がどの順序で現れてもよい「順不同構造」，任意位置に要素が現れる「添加構造」，要素の中に同じ型の要素が現れる「再帰構造」など．

r_1:

A	B
1	⟨ paper:**seq**(title, authors:**rep**(author), pub-info, content, ref:**rep**(ref-item:**seq**(title, authors:**rep**(author), pub-info))), "<paper><title>T1</title><authors><author>A1</author><author>A2</author></authors> ... <ref><ref-item><title>T2</title><authors><author>A3</author><author>A4</author></authors> <pub-info>P1</pub-info></ref-item> ... </ref></paper>" ⟩

r_2:

A	B	C	
		O	D
1	⟨ paper:**seq**(title, authors:**rep**(author), pub-info, content, ref:**rep**(ref-item:**seq**(title, authors:**rep**(author), pub-info))), "<paper><title>T1</title><authors>&x.1;&x.2;</authors> ... <ref><ref-item><title>T2</title><authors> <author>A3</author><author>A4</author></authors> <pub-info>P1</pub-info></ref-item> ... </ref></paper>" ⟩	1	⟨ author, "<author>A1</author>" ⟩
		2	⟨ author, "<author>A2</author>" ⟩

図5 コンバータの実行例
Fig. 5 Examples of converters.

プリミティブなコンバータとしてUnpackとPackを用意する．Unpackは，SD値中の要素群を値として含む新たな副リレーション構造を作成する．副リレーション構造中に抽出されるべき要素群は，Unpackのパラメータで指定する．Packは，副リレーション構造中の要素から構成される新たなSD値を作成する．まず，Unpackのパラメータにおける要素指定について説明し，続いてコンバータUnpackとPackの説明を行う．UnpackとPackの形式的な定義は付録A.1に示す．

要素指定：Unpackでの要素指定にはリージョン代数式[7),8)]を用いる．リージョン代数式を用いることにより，SD値を構成する要素群のうち，指定条件を満たす要素の集合が特定される．**リージョン**（*region*）とは，テキスト中の連続する領域のことである．リージョン代数は，リージョンの集合を操作する代数である．NR/SD+では，リージョン集合を，文書中の各要素に対応するものに限定して考える．たとえば，タグ付きテキスト "<cities> <c> Nara </c> <c> Nice </c> <capitals> <c> Tokyo </c> <c> Paris </c> </capitals> </cities>" に対して，リージョン集合 "c" は，{ "<c> Nara </c>," "<c> Nice </c>," "<c> Tokyo </c>," "<c> Paris </c>" } を表し，リージョン集合 "capitals" は，{ "<capitals> <c> Tokyo </c> <c> Paris </c> </capitals>" } を表す．"**I**" は特殊なリージョン集合であり，タグ付きテキスト全体を構成する最上位要素ただ1つを含むリージョン集合を表す．この例では，{ "<cities> <c> Nara </c> <c> Nice </c> <capitals> <c> Tokyo </c> <c> Paris </c> </capitals> </cities>" } である．

リージョン代数では，和演算子（∪）や差演算子（−）といった通常の集合演算子のほかに，固有の演算子である包含演算子（⊃），被包含演算子（⊂），選択演算子などが用意される．包含演算子と被包含演算子は次のように定義される．

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

ただし，$r \supset s$ は要素 r が要素 s を完全に包含するときに成立する（$r \supset r$ は不成立）．$r \subset s$ も同様に定義される．先のタグ付きテキストに対して，リージョン代数式 "c − c⊂ capitals" は，{ "<c> Nara </c>," "<c> Nice </c> " を表す．

選択演算子（$\sigma[w](e)$）はリージョン集合 e に含まれる要素のうち，単語 "w" を含む要素を選択する．先のタグ付きテキストに対して，リージョン代数式 "σ["Nice"](c)" は { "<c> Nice </c>" } を返す．

Unpack：コンバータUnpack（**U**）は，SD値から，そのSD値中の要素群を値として含む新たな副リレーション構造を作成する．2章の例では**操作1**で利用される．**図5**に次式の適用結果を示す．

$$r_2 \leftarrow \mathbf{U}_{B \to C(O, D[author - author \subset ref])\ as\ x}(r_1)$$

この例は，属性B中の論文データから著者の情報を抽出して，属性Cの副リレーション構造を作成するものである．属性C中の各副リレーションには，属性B中のSD値を構成する各要素に対応するSD値群のうち要素指定の条件 $author - author \subset ref$ に合致するものが格納される．したがって，要素型authorの要素のうち，refに含まれるものを除いたものが，属性DのSD値として格納される．これにより，各論文から，当該論文の著者情報だけを抽出する．r_2 の属性BのSD値に現れる文字列 &x.n; を**SDリファレンス**（*SD reference*）と呼ぶ．SDリファレンスは，

Unpackによって抽出された副リレーション構造中のSD値への参照を表す．次のPackでは，この参照関係を用いることでUnpackの逆操作を可能とする．SDリファレンスの頭部（この例では“x”）はデータ操作上の目的に応じてユーザが指定する．以下では，SDリファレンスを含む参照元のSD値を**マスタ**（*master*），参照先のSD値を**デリバティブ**（*derivative*）と呼ぶ（**図6**）．

Pack：Pack（**P**）は，Unpackの逆演算に相当し，マスタとなるSD値に副リレーション構造中の要素群（デリバティブ）を埋め込むことにより，新たなSD値を作成する．統合利用例では**操作3**で使用される．図5に次式の適用結果を示す．

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

ここでは，属性Cの各副リレーションのデリバティブを，r_2 の属性Bのマスタの構造化文書に埋め込むことにより，r_1 を得る．すなわち，マスタ中のSDリファレンスがそれぞれ対応するデリバティブのテキスト部によって置換される．対応するデリバティブが存在しないSDリファレンスがあった場合は，それらはPack後も元のままとする．

3.3 マスタコンストラクタ

Packを用いてリレーション構造をSD値に変換する際には，マスタとしての役割を果たすSD値が必要である．しかし，該当するリレーション構造が直接Unpackによって作成されたものでないためにマスタが存在しない場合や，Unpackで作成されたマスタとは異なる，目的に応じた別のマスタを利用したい場合がある．このような状況に対処するため，**マスタコンストラクタ**（*master constructor*）と呼ぶ演算子群を用意する．マスタコンストラクタは，デリバティブに矛盾しないDTDとSDリファレンスを持つマスタを作成する．以下では，**seq**構造と**rep**構造をそれぞれ最上位構造として持つマスタを生成する *Sequence master constructor*（**SC**）と *Repetition master constructor*（**RC**）を考える．**図7**は次式の適用結果である．

$r_4 \leftarrow \mathbf{SC}_{C(O,D)\ as\ x,B,G}(r_3)$

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

属性Gの値は新たに生成されるマスタの最上位要素の共通識別子として使用される．統合利用例の**操作3**では新たな文書構造を持つSD値を作成するために，マスタコンストラクタを利用する．

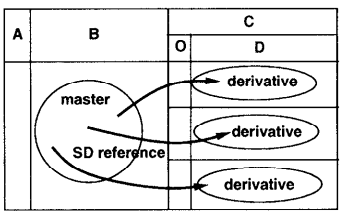


図6 マスタとデリバティブ
Fig. 6 Master and derivatives.

3.4 NR/SD+代数

NR/SD+代数は，プリミティブな演算子として，先に説明したコンバータとマスタコンストラクタのほかに，通常の入れ子型リレーショナル代数演算子（**図8**）と**改名演算子**[14]，さらに**ドメイン変換子**を持つ．改名演算子は，リレーションの属性名を変更する．ドメイン変換子 $\gamma_{Attr,type}(r)$ は，属性 *Attr* のドメインを *type* に変更する．ただし，*type* がSD型の場合は，テキストのタグに利用される共通識別子 g を明示的に指定する必要がある．たとえば，$\gamma_{A,SD(para)}(r)$ は属性 A のドメインをSD型に変更し，属性 A 中の値“v”をSD値〈 para, “ <para>v</para> ”〉に変

r_3:

A	C		G
	O	D	
1	1	〈 title, “<title>T1</title>” 〉	pub
	2	〈 pub-info, “<pub-info>P</pub-info>”〉	

r_4:

A	C		B
	O	D	
1	1	〈 title, “<title>T1</title>” 〉	〈 pub:**seq**(title, pub-info), “<pub>&x.1;&x.2;</pub>”〉
	2	〈 pub-info, “<pub-info>P</pub-info>”〉	

r_5:

A	C		G
	O	D	
1	1	〈 pub, “<pub>P1</pub>” 〉	pubs
	2	〈 pub, “<pub>P2</pub>”〉	

r_6:

A	C		B
	O	D	
1	1	〈 pub, “<pub>P1</pub>”〉	〈 pubs:**rep**(pub), “<pubs>&x.1;&x.2;</pubs>”〉
	2	〈 pub, “<pub>P2</pub>”〉	

図7 マスタコンストラクタ**SC**と**RC**の実行例
Fig. 7 Master constructors **SC** and **RC**.

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8 入れ子型リレーショナル代数演算子
Fig. 8 Nested relational algebra operators.

換する．ドメイン変換子を用いることにより，もともと文字列などで与えられたデータをSD値に変換してコンバータによる操作対象としたり，また，SD値中の一部のデータを抽出して文字列や整数として提示や比較を行うことが可能となる．

これらプリミティブな演算子をベースとして，式の簡潔な記述を行うための**複合演算子**を定義する．結合演算子 $\bowtie_p$ は複合演算子の1つである．他の複合演算子としては，選択条件としてリージョン代数式を指定する**拡張選択演算子** $\sigma_{\rho(attr,expr)}(r)$ がある．ここで，*attr* はSD型の属性であり，*expr* はリージョン代数式による要素指定である．この演算子は，属性 *attr* のSD値に *expr* を適用した結果が要素の非空集合となるようなタプルを選択するものであり，$\rho(attr, expr)$ はこの条件で真となる疑似述語と見なすことができる．たとえば，SD型の属性 A を持つ単項リレーションのインスタンス r が {⟨ a:**rep**(b:**or**(c, d)), "<a> <c> w1 </c> " ⟩, ⟨ a:**rep**(b:**or**(c, d)), "<a> <d> w2 </d> " ⟩} であるとすると，$\sigma_{\rho(A,b\supset c)}(r)$ は {⟨ a:**rep**(b:**or**(c, d)), "<a> <c> w1 </c> " ⟩} となる．

また，以下のような**複合コンバータ**を，プリミティブなコンバータと他の演算子の組合せとして定義することができる．複合コンバータは，プリミティブなコンバータとは異なるパラメータを持つ．

- $\mathbf{U}_{A_i \to (B_1[e_1]\ as\ x_1,\ldots,B_n[e_n]\ as\ x_n)}(r)$：Unpackの拡張であり，複数のUnpack操作をまとめたものである．UnpackとUnnestの組合せとして定義される．**図9**において，$r_8 \leftarrow \mathbf{U}_{B \to (C[month]\ as\ x, D[year]\ as\ y)}(r_7)$ を適用することにより，month要素を抽出した属性 C とyear要素を抽出した属性 D を持つリレーション r_8 を得る．
- $\mathbf{P}_{(A_i,\ldots,A_j)\to B:SC,G}(r)$：マスタコンストラクタ **SC** とPackを利用して定義される．属性 $A_i, \ldots, A_j$ 中のSD値をデリバティブとして，**seq** 構造を最上位構造に持つ新たなSD値を属性 B に作成する．図9において，$r_7 \leftarrow \mathbf{P}_{(C,D)\to B:SC,G}(r_9)$ を適用することにより，r_9 の属性 C の値と属性 D の値を **seq** 構造で結合したSD値を属性 B に持つリレーション r_7 を得る．
- $\mathbf{P}_{A_i(D)\to B:RC,G}(r)$：マスタコンストラクタ **RC** とPackを利用して定義される．属性 D 中のSD値をデリバティブとして，**rep** 構造を最上位構造に持つ新たなSD値を属性 B に作成する．図9において，$r_{11} \leftarrow \mathbf{P}_{C(D)\to B:RC,G}(r_{10})$ を適用することにより，r_{10} の属性 C の各副リレーションに含まれる属性 D の値を **rep** 構造で結合したSD値を属性 B に持つリレーション r_{11} を得る☆．

4. NR/SD+による統合操作の記述

ここでは，2章で示した統合操作例をNR/SD+で記述する．NR/SD+の統合スキーマでは，リレーショナルデータベース中のリレーションはそのまま表現し，文書リポジトリに格納されている構造化文書群はSD型の属性を持つ単項リレーション "Document" として表現する．**図10**に，この例での統合スキーマを示す．"Faculty" の属性FID, D-Name, F-Titleは，それぞれ教官の識別子，所属学科名，役職名を表す．"Document" 中のSD値は**図11**のDTDを持つ☆☆．問合せ結果である教官別論文書誌情報リストは，SD型属性を持つ**図12**の単項リレーションに格納され，それぞれ**図13**のDTDを持つこととする．このとき，統合操作例の各操作は以下のようになる．具体的な問合せ式は付録A.2に示す．

操作1 コンバータUnpackとUnnestを用いて，"Document" 中のSD値群から書誌情報を抽出する．ただし，各論文の参考文献の書誌情報を抽出しないように，たとえばauthors要素の抽出の際には，要素指定 authors − authors ⊂ ref を利用する．なお，affiliationがA大学ではない著者の情報は除去する．

操作2 結合演算子を用いて，抽出した書誌情報とリレーション "Faculty" を結合し，射影演算子で必要な属性のみを残す．

操作3 コンバータPackとNestを用いて，図13の構造を持つSD値を作成する．

操作4 拡張選択演算子を用いて，作成されたSD値群のうちacademic-info要素に単語 "database" を持つものを選択する．

☆ 厳密には，この演算ではデリバティブの出現順序を決定する必要があるが，本稿ではこれについての説明は省略する．

☆☆ 図4のDTDをDAG表記したもの．また，ここでは簡単化のためaffiliationには所属機関名が記載されているものとする．

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>"⟩

r_9:

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

r_{10}:

A	C / D	G
1	⟨ pub, "<pub>P1</pub>" ⟩	pubs
	⟨ pub, "<pub>P2</pub>"⟩	

r_{11}:

A	B
1	⟨ pubs:**rep**(pub), "<pubs><pub>P1</pub> <pub>P2</pub></pubs>"⟩

図 9 複合コンバータの実行例
Fig. 9 Examples of composite converters.

Faculty:

FID	Name	D-Name	Speciality	Course	F-Title

Document:

Doc

図 10 統合スキーマ
Fig. 10 Integrated schema.

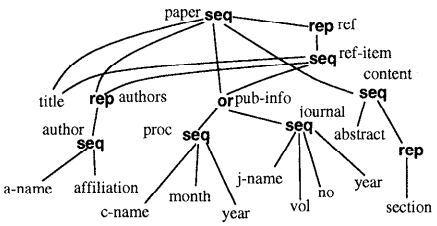


図 11 属性 "Doc" 中の SD 値の DTD
Fig. 11 DTD in attribute "Doc".

Table

図 12 結果のリレーション "Ans"
Fig. 12 Required relation "Ans".

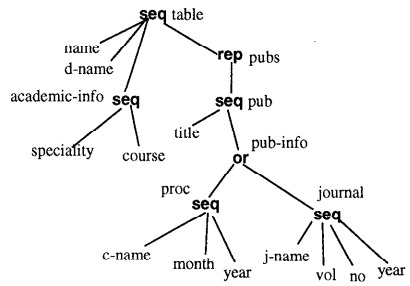


図 13 属性 "Table" 中の SD 値に対して要求される DTD
Fig. 13 Required DTD in attribute "Table".

5. 問合せ処理と最適化

ここでは，本統合利用環境における問合せ処理と最適化について述べる．まず，メディエータの作業領域と処理コストの削減を図るための ASD 値の利用について述べる．また，ASD 値を利用するために，NR/SD+代数演算子に加えて，新たな 2 つの演算子を導入する．次に，問合せ処理と最適化の概要について説明する．本章の説明には，説明用の簡単な例を利用する．2 章で述べた統合利用例に対する問合せ式（付録 A.2）への本処理方式の適用は，付録 A.5 に示す．

5.1 ASD 値の利用

構造化文書は一般に大量のデータ量を持つため，ラッパーからメディエータへの転送コストやメディエータでの作業領域が膨大なものとなる可能性がある．方，NR/SD+による構造化文書（SD 値）の操作はしばしば文書の一部分のみを必要とし，文書の詳細部は，特定の演算の適用時や最終結果を求める時点まで不必要という場合が存在する．そこで，ラッパーからメディエータへの SD 値の転送の一部を，問合せ処理に必要な時期まで遅延する方式を検討する．問合せ処理に不必要な詳細部分を抽象化した SD 値を **ASD 値**（*abstract SD value*）と呼ぶ．ASD 値の抽象化された各部には，テキスト本体は含まれておらず，その代わりにその部分の実体が文書リポジトリ中のどこに格納されているかを示す情報が格納される．

図 14 の r_{12} の SD 値を ASD 値に変換した例が r_{13} である．ASD 値の DTD では，**rep** や **seq** 構造のほかに，特殊な要素構造である "UNKNOWN" が用いられる．"d:UNKNOWN" は，要素型 "d" の内部構

r_{12}:

A	B
1	⟨ a:**seq**(b:**rep**(c), d:**seq**(e,e)), "<a><c>T1</c><c>T2</c> <d><e>T3</e><e>T4</e></d>''⟩

r_{13}:

A	B
1	⟨ a:**seq**(b:**rep**(c), d:UNKNOWN), "<a><c>T1</c><c>T2</c> <d> [id1, 11, 16] </d>"⟩

図14 SD値とASD値の変換例
Fig. 14 Example of transformation between SD values and ASD values.

造が省略されていることを表す．ASD値のタグ付きテキスト部において，要素型 "d" に対応する部分には，本来のテキストデータではなく，代わりに，マーカ（*marker*）と呼ぶ3つ組（[id1, 11, 16]）が現れる．マーカの構成要素は，それぞれ文書リポジトリ中に存在する文書の識別子，その文書内における実際の要素のテキストデータの開始位置，終了位置である．

重要な点は，SD値をASD値に抽象化することにより文書の詳細部が不明な状況であっても，適用可能な演算が存在することである．たとえば，リレーション r_{12} と r_{13} のどちらに対しても，Unpack演算 $\mathbf{U}_{B\to C(O,D[c\subset b])\ as\ x}$ は要素型 "d" の内部のデータが不要なため，適用可能である．

1つのSD値に対応して，様々な抽象度のASD値が存在する．たとえば，最も抽象度の高いASD値は，DTDがただ1つのUNKNOWN構造で構成されているものである．r_{12} のSD値に対しては，この場合のASD値は ⟨ a:UNKNOWN, "<a> [id1, 2, 17] " ⟩ となる．しかし，このASD値を含むリレーションに対して上記のUnpack演算を適用することは不可能である．なぜなら，抽象化された部分の中にUnpackの要素指定に現れる要素型 "c," "b" が存在する可能性があるからである．したがって，ASD値の利用の際には，元のSD値のDTDと問合せ式の両者を考慮した抽象度の決定が必要である．

SD値抽象化演算子

SD値抽象化演算子 $\mathbf{SDA}_{Attr,abs}(r)$（SD Abstraction）は，属性 *attr* 中のSD値をASD値に変換する．抽象度を指定するのが，抽象化指定子（abstraction specifier）*abs* である．*abs* は $(g_1,\ldots,g_n;g_{n+1},\ldots,g_m)$ という形式を持つ．これは，要素型 $g_1,\ldots,g_m$ の要素は必ずASD値中にも残存させること，特に，要素型 $g_1,\ldots,g_n$ の要素の内部にはUNKNOWN構造を持つ要素をいっさい含めないことを指定する．たとえば，上に述べたようにASD値に対してUnpack演算を適用可能とするためには，Unpackのパラメータにおいて要素指定のリージョン式に現れる要素型を $g_1,\ldots,g_m$ に含めておけばよい．また，特にリージョン式中である単語を含むか否かの判定が必要となる要素型は $g_1,\ldots,g_n$ に含めなくてはならない．本演算子が適用された際，ASD値が持つべきDTDを当該SD値ごとに決定する必要がある．その具体的方法は，付録A.3に示す．

SD値具体化演算子

SD値具体化演算子 $\mathbf{SDM}_{attr}(r)$（SD Materialization）は属性 *attr* 中のASD値をSD値に変換する．たとえば，r_{13} に**SDM**演算子を適用することにより r_{12} が得られる．

5.2 問合せ処理の概要

本節では，問合せ処理と最適化の概要について述べる．ここでは，リレーショナルデータベースと文書リポジトリの持つローカルな問合せ処理能力について以下を仮定する．すなわち，リレーショナルデータベースはリレーショナル代数式を処理可能である．文書リポジトリはリージョン代数に基づくテキスト検索機能を持ち，$\sigma_{\rho(attr,expr)}(r)$ に相当する処理を実行可能である．また，メディエータは，リレーショナルデータベースのスキーマ情報と文書リポジトリに格納されている構造化文書のDTDを参照可能であるとする．議論を簡単にするため，以下では，メディエータに投入される問合せを，文書リポジトリを表現するリレーションとリレーショナルデータベース中の1つのリレーションをそれぞれ1度ずつ参照する問合せに限定して説明する．しかし，多数のリレーションを参照するようなより一般的な問合せに対して，以下の手法を拡張することは可能である．

問合せ処理の基本的な流れを**図15**に示す．メディエータは，問合せを表すNR/SD+代数式を受け取ると，それを $F(G,H)$ の形式のNR/SD+代数式に変換する☆．部分式 G を構成する演算子は，リレーショナルデータベース中のリレーションに適用されるリレーショナル代数演算子である．ラッパー1は G をリレーショナルデータベースに投入し，その結果である *Ans*1 をメディエータに転送する．部分式 H は $\mathbf{SDA}_{attr,abs}(\sigma_{\rho(attr,expr)}(R))$ の形式をしている．ここで，R は属性 *attr* を持つ単項リレーションであり，文書リポジトリ中の構造化文書群を表す．ラッパー2は H の部分式 $\sigma_{\rho(attr,expr)}(R)$ に相当する検索命令

☆ 実際には，$F(G,H)$ はNR/SD+代数演算子のほかに，SDA演算子とSDM演算子を含む．

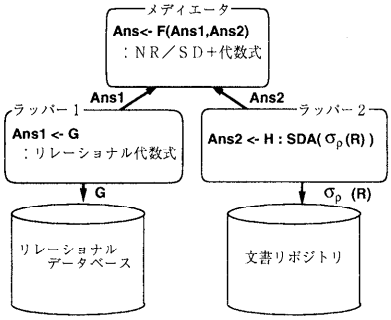


図15 問合せ処理の概要
Fig. 15 Query processing framework.

を文書リポジトリに投入し，その結果からSD値を含む単項リレーションを作成する．続いて**SDA**演算子を実行し，抽象化指定子 abs に基づいてSD値をASD値に変換する．最後にラッパー2はその結果であるリレーション $Ans2$ をメディエータに転送する．メディエータは $F(Ans1, Ans2)$ を実行し，最終結果 Ans を得る．以下に示すように，F 中には**SDM**演算子が含まれるが，その実行時には，メディエータがラッパー2に対して，ASD値の具体化に必要なデータの転送を要求する．

メディエータに投入された問合せから $F(G, H)$ を求める手順は以下のとおりである．

[Step 1] 問合せを $F'(G, H')$ の形式に分解する．ここで，F' と H' は，それぞれ前述の F と H の基となる式である．部分式 G は，リレーショナルデータベース中のリレーションに適用されるリレーショナル代数演算子のみから構成される．H' は $\sigma_{\rho(attr,expr)}(R)$ という形式をしている．ここで，R は文書リポジトリに対応する単項リレーションである．F' はNR/SD+代数演算子から構成され，G と H' の結果に対して適用される．たとえば，リレーショナルデータベースに二項リレーション $R_1(A_1, A_2)$ が存在し，文書リポジトリに対して単項リレーションのスキーマ $R_2(B)$ が割り当てられていると仮定する．このとき，問合せ式 $\sigma_{A_2=c}(R_1) \bowtie_{A_1=B_1} \mathbf{U}_{B\to(B_1[b_1],B_2[b_2])}(\sigma_{\rho(B,expr)}(R_2))$ を分解して得られる F', G, H' は以下のようになる．

F' : Ans
$\quad \leftarrow Ans1 \bowtie_{A_1=B_1} \mathbf{U}_{B\to(B_1[b1],B_2[b2])}(Ans2)$
G : $Ans1 \leftarrow \sigma_{A_2=c}(R_1)$
H' : $Ans2 \leftarrow \sigma_{\rho(B,expr)}(R_2)$

この例では非常に単純な分解によって $F'(G, H')$ を求めたが，実際には問合せを $F'(G, H')$ へと分解する際には，ローカルに実行できる選択演算子等は G と H' へ移動させ，F' 中の演算子をできる限り少なくする．この変換は，NR/SD+代数式の変換規則群を利用して行われる．この変換規則群は，通常リレーショナル代数式の最適化で一般的に使われる規則のほかに，選択演算子とコンバータの可換性に関する規則を含む．これらにより，文書リポジトリの検索機能を用いて入れ子型リレーショナル代数演算の支援を行ったり，逆にリレーショナルデータベースの問合せ処理能力を用いてリージョン代数式に基づくテキスト検索の支援を行うことができる．付録A.4に，変換規則の一部を示す．

[Step 2] F' と H' を基に，F と H を作成する．まず，H を $\mathbf{SDA}_{attr,abs}(H')$ とする．抽象化指定子 abs は F' に基づいて決定される．Step 1で示した式 F' から，Unpackに必要な要素型は b_1 と b_2 であり，かつ，属性 B_2 のSD値（要素型 b_2 の要素を持つ）の内容は，結合演算が終了するまで不必要であることが分かる．したがって abs を "$(b_1; b_2)$" と指定することができ，H は，

$$Ans2 \leftarrow \mathbf{SDA}_{B,(b_1;b_2)}(\sigma_{\rho(B,expr)}(R_2))$$

となる．

また，F' 中でASD値でなくSD値が必要な場所に**SDM**演算子を挿入することにより F が作成される．上の例では，最終結果 Ans を得る時点でのみ**SDM**演算子が必要である．したがって，F は以下のようになる．

$$Ans \leftarrow \mathbf{SDM}_{B_2}(\\ Ans1 \bowtie_{A_1=B_1} \mathbf{U}_{B\to(B_1[b_1],B_2[b_2])}(Ans2))$$

ここでは，ASD値の最も基本的な使用を示したが，実際にはメディエータで利用可能な作業領域，各情報源での問合せ処理コスト，ラッパー/メディエータ間のデータ転送コストなどを考慮して，ASD値の利用法を制御する必要がある．

6. 関連研究

異種分散情報源の統合については様々な研究が行われている．第1のアプローチとして，異種情報源のデータを均一に表現可能な抽象データモデルを用意し，そのデータモデルに基いた統合操作を行うものがある．この手法の利点の1つは，利用者が多様な異種情報源を単一の操作体系の下で操作可能であることである．CPL[4], UnQL[5], OEM[18]などがこのようなアプローチを採用している．CPLではリストやバリアントなどを含む豊富な型を用意することにより，様々なデータ構造に対応している．UnQLとOEMでは，すべての情報源を，明示的なスキーマを持たない

ラベル付き有向グラフとして表現する．以上のデータモデルではcomprehension syntaxやOQLに基づいたデータ操作体系を用意している．

第2のアプローチとして，ハイブリッドな手法を用いた研究がある．これは，各種情報源に対する既存のデータモデルや操作体系を組み合わせたり，それらを拡張することによって，異種情報源の統合を行うものである．このアプローチでは，まったく新しいデータモデルを導入する代わりに，利用者にとって比較的馴染みのある既存のデータモデルや操作体系に基づいた統合が行われる．また，既存のデータモデルに対してすでに確立した記憶管理やインデックス，問合せ処理といった技術の応用も比較的容易である．さらに，各情報源の持つデータを統合データモデルに基づく記述と相互変換する際のオーバヘッドも，第1のアプローチに比べて一般的に小さいといえる．NR/SD+を用いた統合も，このハイブリッドな手法に分類される．

構造化文書とデータベースを対象としたハイブリッドな統合に関しても，これまでにいくつかの研究が行われている[1),2),6),9),12),13),21),22),24)]．特に，T/RDBMS[3)]では，本研究と同様に，リレーショナルデータモデルと構造化文書を扱う抽象データ型を組み合わせている．これらの関連研究に対するNR/SD+の特徴は，構造化文書とリレーション構造を**動的，双方向，部分的**に変換することにより，同一データに対して構造化文書による表現と入れ子型リレーションによる表現の両者やその混在を可能としている点である．また，必要に応じて同一のデータに対し入れ子型リレーショナル代数系と文書検索操作の両者の演算を使い分けることや，構造の異なる複数のデータに対して，共通の部分構造を入れ子型リレーション構造で保持し，非共通構造はSD値として扱うことにより，入れ子型リレーション構造のレベルでは細部の差異を隠蔽した共通のスキーマを割り当て，統一的な操作を行うといったことが可能である．

我々が以前に提案したNR/SD[15),16)]はNR/SD+の基となった統合データモデルであり，NR/SD+と同様，コンバータによる動的，双方向，部分的な相互変換を実現している．しかし，現実の応用への適用の際に以下のような問題点がある．

(1) NR/SDでは，特定の文書構造–特定の入れ子型リレーション構造の組に対してコンバータが定義されている．しかし，この組合せとしては現実には種々のものが考えられる．たとえば，NR/SDでは文書の要素列構造とリレーションの属性列の相互変換を行うコンバータが用意されているが，要素列構造を副リレーション構造と相互変換するコンバータも考えられる．また，特定の文書構造–入れ子型リレーション構造の組の数だけコンバータを用意していては，SGML文書等に現れるような様々な文書構造に対応しようとする際に，多数のコンバータの導入が必要となる．NR/SD+では，変換対象となる文書構造の制約をなくしている．

(2) NR/SDでは，コンバータによってSD値と相互変換されるリレーション構造中の値が，そのSD値中の要素の階層構造における上位要素群に限定される．したがって，SD値の下位要素群に対してコンバータを適用するためには，コンバータ以外の演算を含めた複数の演算を，DTDに従って複雑に組み合わせ適用する必要がある．NR/SD+では，コンバータによりSD値と相互変換されるリレーション構造中の値が，そのSD値の階層構造における上位要素に限定されることはない．

NR/SD+でこれらの問題点を解決するために導入されたのが，リージョン代数式を用いた要素指定と，マスタの概念である．リージョン代数式を用いた要素指定は，コンバータUnpackに関して，上記の問題点の解決に貢献している．NR/SD+のUnpackでは，抽出対象となる要素群の決定が，リージョン代数式に基づきインスタンスベースで行われる．その結果，文書構造にかかわらずUnpackが適用可能となり，かつ抽出対象となる要素群を文書の上位要素群に限定する必要がなくなった．一方，マスタの導入は，コンバータPackに関して，上記の問題点の解決に貢献している．マスタの満たすべき条件は，所定のSDリファレンスを含むことだけであるので，SD値と，リレーション構造中に含まれるデリバティブの複雑な関係を表現可能である．したがって，NR/SD+のPackでは，作成されるSD値の文書構造が，特定の文書構造に限定されるという制約がなくなり，また，SD値の下位構造に直接デリバティブを埋め込むことが可能となった．

オブジェクト指向モデルを用いた異種情報源の統合[19),20)]についてもいくつかの研究が行われている．NR/SD+の特徴であるコンバータを用いた構造化文書とデータベースの動的，双方向，部分的な変換の概念を，オブジェクト指向モデルの枠組みの中で議論することは興味深い今後の研究課題である．ただし，その場合，動的に生成されるオブジェクトのクラスやクラス階層，メソッドの定義，OIDの割当てなどを含めた議論が必要となる．

吉川ら[25)]の研究では，SGML文書に対して，データベースのオブジェクトへの参照を埋め込むための仕組みを提案している．

構造化文書とデータベースを対象とした統合利用環境における問合せ処理に関して，いくつかの研究が存在する[1),7),22),24)]．これらの関連研究は，(1) 問合せ処理や最適化が，データベースをフロントエンドとした非対称型の統合形態の範囲で議論されていること，(2) ASD 値のようなデータ転送の遅延の枠組みがないこと，といった点で本稿の問合せ処理とは異なるが，データベースと文書リポジトリによる問合せ処理の分担という点で深い関連を持つ．特に，文献 7) での，リージョン代数式の最適化や文書リポジトリに実際に存在するインデックスを考慮した問合せ処理に関する議論は，本稿の問合せ処理の枠組みでも有効であると考えられる．

7. おわりに

本稿では，構造化文書とリレーショナルデータベースを対象とした異種情報源統合利用環境における，統合データモデル NR/SD+と問合せ処理について述べた．NR/SD+は，入れ子型リレーション構造に抽象データ型である構造化文書型を導入したデータモデルである．入れ子型リレーショナル代数系によるリレーション操作と構造化文書に対する文書検索操作に加えて，入れ子型リレーション構造と構造化文書の動的，双方向，部分的な変換操作が可能であり，これらを用いて構造化文書とリレーショナルデータベースの統合操作を実現する．また，本統合利用環境では，動的な変換操作を含む問合せ式を効率的に実行するために，問合せ式の変換規則を利用して各情報源の問合せ処理機能の有効利用を図る．さらに，構造化文書の一部のデータ転送の遅延を行い，統合操作に必要な作業領域や中間処理コストの削減を行う．

今後の課題としては，NR/SD+では現在扱っていない構造化文書固有のデータ構造である，要素に付随する属性などへの対応を行うことや，利用者向け問合せ言語，視覚的統合操作環境の開発などがある．NR/SD+代数の問合せ記述力の評価も重要な課題の1つである．また，問合せ処理におけるインデックスの利用，ASD 値の抽象度のより細かな制御，ビューの具体化などの，より高度な最適化の検討も今後の課題である．

謝辞 貴重なコメントを下さった査読者の方々に感謝いたします．本研究の一部は，文部省科学研究費補助金重点領域研究「高度データベース」(08244101)，基盤研究 (C)(09680321)，ならびに筑波大学「東西言語文化の類型論」特別プロジェクトの助成による．

参考文献

1) Abiteboul, S., Cluet, S. and Milo, T.: Querying and Updating the File, *Proc. 19th VLDB Conf.*, pp.73–84 (1993).
2) Abiteboul, S., Cluet, S. and Milo, T.: A Database Interface for File Update, *Proc. ACM SIGMOD Conf.*, pp.386–397 (1995).
3) Blake, G.E., Consens, M.P., Kilpeläinen, P., Larson, P., Snider, T. and Tompa, F.: 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) Buneman, P., Davidson, S., Hart, K. and Overton, C.: A Data Transformation System for Biological Data Sources, *Proc. 21st VLDB Conf.*, pp.158–169 (1995).
5) Buneman, P., Davidson, S., Hillebrand, G. and Suciu, D.: A Query Language and Optimization Techniques for Unstructrured Data, *Proc. ACM SIGMOD Conf.*, pp.505–516 (1996).
6) Christophides, V., Abiteboul, S., Cluet, S. and Scholl, M.: From Structured Documents to Novel Query Facilities, *Proc. ACM SIGMOD Conf.*, pp.313–324 (1994).
7) Consens, M.P. and Milo, T.: Optimizing Queries on Files, *Proc. ACM SIGMOD Conf.*, Minneapolis, Minnesota, pp.301–312 (1994).
8) Consens, M.P. and Milo, T.: Algebras for Querying Text Regions, *Proc. ACM Symposium on Principles of Database Systems*, pp.11–22 (1995).
9) Croft, W.B., Smith, L.A. and Turtle, H.R.: A Loosely-Coupled Integration of a Text Retrieval System and an Object-Oriented Database System, *Proc. ACM SIGIR Conf.*, pp.223–232 (1992).
10) Elmagarmid, A.K. and Pu, C. (Eds.): Special Issues on Heterogeneous Databases, *ACM Computing Surveys*, Vol.20, No.3 (1990).
11) Fischer, P.C. and Thomas, S.J.: Operators for Non-First-Normal-Form Relations, *Proc. IEEE COMPSAC83*, Chicago, pp.464–475 (1983).
12) Güting, R.H., Zicari, R. and Choy, D.M.: An Algebra for Structured Office Documents, *ACM Trans. Office Information Systems*, Vol.7, No.4, pp.123–157 (1989).
13) Järvelin, K. and Niemi, T.: An NF^2 Relational Interface for Document Retrieval, Restructuring and Aggregation, *Proc. SIGIR Conf.*, pp.102–110 (1995).

14) Maier, D.: *The Theory of Relational Databases*, Computer Science Press (1983).
15) Morishima, A. and Kitagawa, H.: 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 (1997).
16) Morishima, A. and Kitagawa, H.: A Data Modeling and Query Processing Scheme for Integration of Structured Document Repositories and Relational Databases, *Proc. 5th International Conference on Database Systems for Advanced Applications (DASFAA '97)*, Melbourne, pp.145–154 (1997).
17) 森嶋厚行, 北川博之：参照の導入による構造化文書とデータベースの統合操作の検討, 情報処理学会データベースシステム研究会研究報告, Vol.97, No.64, pp.21–26 (1997).
18) Papakonstantinou, Y., Garcia-Molina, H. and Widom, J.: Object Exchange Across Heterogeneous Information Sources, *Proc. 11th Data Engineering Conf.*, pp.251–260 (1995).
19) Pitoura, E., Bukhres, O. and Elmagarmid, A.: Object Orientation in Multidatabase Systems, *ACM Computing Surveys*, Vol.27, No.2, pp.141–195 (1995).
20) Roth, M.T. and Schwarz, P.: Don't Scrap It, Wrap It! A Wrapper Architecture for Legacy Data Sources, *Proc. 23rd VLDB Conf.*, Athens, Greece (1997).
21) Sacks-Davis, R., Kent, A., Ramamohanarao, K., Thom, J. and Zobel, J.: Atlas: A Nested Relational Database System for Text Applications, *IEEE Trans. Knowledge and Data Engineering*, Vol.7, No.3, pp.454–470 (1995).
22) Volz, M., Aberer, K. and Böhm, K.: Applying a Flexible OODBMS-IRS-Coupling to Structured Document Handling, *Proc. 12th Data Engineering Conf.* (1996).
23) Wiederhold, G.: Mediators in the Architecture of Future Information Systems, *IEEE Computer*, pp.38–49 (1992).
24) Yan, T.W. and Annevelink, J.: Integrating a Structured-Text Retrieval System with an Object-Oriented Database System, *Proc. 20th VLDB Conf.*, Santiago, Chile, pp.740–749 (1994).
25) Yoshikawa, M., Ichikawa, O. and Uemura, S.: Amalgamating SGML Documents and Databases, *Proc. 5th International Conf. on Extending Database Technology* (1996).

付　　録

A.1 コンバータの定義

プリミティブなコンバータUnpackとPackの定義を示す.

Unpack：リレーション r が属性 $A_1,\ldots,A_m$ を持つとする. このうち, A_i はSD型とする. ここで, 新たな属性名 B, O, C, リージョン代数式 e, 文字列 x に対して, $r' = \mathbf{U}_{A_i \to B(O,C[e])\ as\ x}(r)$ とする. このとき, r' のスキーマは, r に新たな属性 B を追加したものであり, 属性 B は整数型の副属性 O とSD型の副属性 C を持つ. また, r' のインスタンスは以下のとおりである.

$$\begin{aligned} r' = \quad & \{t \mid \exists u \in r, \exists d, c, n (u[A_i] = \langle d, c \rangle \\ & \wedge n = \#rl(c,e) \wedge t = u \text{ except} \\ & t[A_i] = \langle d, c^{rl(c,e)}_{SDref\_list(x,\#rl(c,e))} \rangle \\ & \text{and } t[B] = \{(1, SD(\langle d, c \rangle, rl(c,e), 1)), \ldots, \\ & \qquad (n, SD(\langle d, c \rangle, rl(c,e), n))\})\} \end{aligned}$$

ここで, $rl(c,e)$ は, タグ付きテキスト c にリージョン代数式 e を適用した結果のリージョン群から構成されるリスト, $\#rl(c,e)$ は $rl(c,e)$ の長さ, $SDref\_list(x,n)$ は x を頭部とする n 個のSDリファレンスから構成されるリスト $[x.1, x.2, \ldots, x.n]$, $c^{[x_1,\ldots,x_p]}_{[y_1,\ldots,y_p]}$ はテキスト c 中の各 x_i を y_i で置換したもの, $SD(v,l,i)$ は, SD値 v を構成する要素群のうち, リージョンリスト l の第 i 番目のリージョンが表す要素に対応したSD値である.

Pack：リレーション r が属性 $A_1,\ldots,A_m$ を持つとする. このうち, 属性 A_i はSD型とし, 属性 A_j は整数型の副属性 O とSD型の副属性 B を持つとする. ここで, 文字列 x に対して, $r' = \mathbf{P}_{A_j(O,B)\ as\ x \to A_i}(r)$ とする. このとき, r' のスキーマは, r から属性 A_j を除去したものである. また, r' のインスタンスは以下のとおりである.

$$\begin{aligned} r' = \{t \mid & \exists u \in r, \exists d, c, d_1, c_1, \ldots, d_n, c_n (u[A_i] - \langle d, c \rangle \\ & \wedge u[A_j] = \{(1, \langle d_1, c_1 \rangle), \ldots, (n, \langle d_n, c_n \rangle)\} \\ & \wedge t = u \text{ except } t[A_i] = \langle d, c^{SDref\_list(x,n)}_{[c_1,\ldots,c_n]} \rangle)\} \end{aligned}$$

A.2 統合操作例を実現するNR/SD+問合せ式

以下では, 式を簡潔にするため, ドメイン変換子を省略する. また, コンバータのパラメータにおける, リージョン代数式による要素指定, SDリファレンスの指定, 共通識別子の指定を以下のように省略する. すなわち,

(RL1) $\sigma_{\rho(A,expr)}(\mathbf{P}_{(A_1,\ldots,A_n)\to A:SC,gi}(r)) \rightsquigarrow \mathbf{P}_{(A_1,\ldots,A_n)\to A:SC,gi}(\sigma_{\rho(A_{i1},expr)\vee\ldots\vee\rho(A_{im},expr)}(r))$
ただし，$expr$ は gi，$\mathbf{I}$ を含まないリージョン代数式．また，$A_{i1}\ldots A_{im}$ は，属性 $A_1,\ldots,A_n$ のうち述語 $\rho(A_i,expr)$ が成立する可能性のある属性．

(RL2) $\sigma_{\rho(A,\sigma[w](gi))}(\mathbf{P}_{(A_1,\ldots,A_n)\to A:SC,gi}(r)) \rightsquigarrow \mathbf{P}_{(A_1,\ldots,A_n)\to A:SC,gi}(\sigma_{\rho(A_1,\sigma[w](\mathbf{I}))\vee\ldots\vee\rho(A_n,\sigma[w](\mathbf{I}))}(r))$

(RL3) $\sigma_{\rho(A_1,expr)}(\mu_A(\mathbf{U}_{A\to(O,A_1[e]\ as\ e),G}(r))) \rightsquigarrow \sigma_{\rho(A_1,expr)}(\mu_A(\mathbf{U}_{A\to(O,A_1[c]\ as\ e)}(\sigma_{\rho(A,expr)}(r))))$

(RL4) $\sigma_{\rho(A,\sigma[w](\mathbf{I}))}(\gamma_{A,SD(gi)}(r)) \rightsquigarrow \gamma_{A,SD(gi)}(\sigma_{A\ni w}(r))$
ただし A は文字列型の属性．

(RL5) $\sigma_{A=w}(\gamma_{A,String}(r)) \rightsquigarrow \sigma_{A=w}(\gamma_{A,String}(\sigma_{\rho(A,\sigma[w](\mathbf{I}))}(r)))$
ただし A は SD 型の属性．

図 16 コンバータ，選択演算子，ドメイン変換子に関する変換規則
Fig. 16 Query rewriting rules.

$r' \leftarrow \mathbf{U}_{Author\to(A\text{-}Name[a\text{-}name]\ as\ a\text{-}name,\ Affiliation[affiliation]\ as\ affiliation)}(r)$
$r'' \leftarrow \mathbf{P}_{Ps(Pub)\to Pubs:RC,'pubs'}(r)$

を，それぞれ以下のように表記する．

$r' \leftarrow \mathbf{U}_{Author\to(A\text{-}Name,Affiliation)}(r)$
$r'' \leftarrow \mathbf{P}_{Ps(Pub)\to Pubs:RC}(r)$

また，演算子のパラメータにおいて，属性名の指定の代わりに明示的に定数を指定したものを一部用いる．これらは複合演算子として定義される．

操作 1

$result_1$
$\leftarrow \sigma_{Affiliation=\text{"A-univ"}}($
$\mathbf{U}_{Author\to(A\text{-}Name,Affiliation)}($
$\mu_{As}(\mathbf{U}_{Authors\to As(O_1,Author)}($
$\mathbf{U}_{Doc\to(Title[title-title\subset ref],\ Authors[authors-authors\subset ref],\ Pub\text{-}Info[pub\text{-}info-pub\text{-}info\subset ref])}($
$Document)))))$

操作 2

$result_2$
$\leftarrow \pi_{Name,D\text{-}Name,Speciality,Course,Title,Pub\text{-}Info}($
$Faculty \bowtie_{Name=A\text{-}Name} (result_1))$

操作 3

$result_3$
$\leftarrow \mathbf{P}_{(Name,D\text{-}Name,A\text{-}Info,Pubs)\to Table:SC}($
$\mathbf{P}_{(Speciality,Course)\to A\text{-}Info:SC,'academic\text{-}info'}($
$\mathbf{P}_{Ps(Pub)\to Pubs:RC}(\nu_{Ps=(Pub)}($
$\mathbf{P}_{(Title,Pub\text{-}Info)\to Pub:SC}(result_2)))))$

操作 4

Ans
$\leftarrow \sigma_{\rho(Table,\sigma[''database''](academic\text{-}info))}(result_3)$

A.3 ASD 値が持つ DTD の決定方法

抽象化指定子 $(g_1,\ldots,g_n;g_{n+1},\ldots,g_m)$ が与えられたとき，SD 値から生成される ASD 値の DTD は以下の手順で決定する．まず，$A = \{g_1,\ldots,g_n\}$，$B = \{g_{n+1},\ldots,g_m\}$ とし，$C = \{g' | \exists g \in A \cup B(may\_contain(g',g))\}$ とする．ここで，$may\_contain(g',g)$ は，元の SD 値の DTD において要素型 g' の要素が要素型 g の要素を内部に含みうると定義されているとき成立する．たとえば，属性 D に DTD "a:**rep**(b:**seq**(c:**seq**(d:**text**,e:**text**),f:**or**(g:**rep**(h:**text**),i:**seq**(j:**text**,k:**text**))))" を持つ SD 値が存在すると仮定し，$\mathbf{SDA}_{D,(f;c)}(r)$ を適用するとき，$A=\{f\}$，$B=\{c\}$，$C=\{a,b\}$ となる．このとき，元の SD 値の DTD 中で定義される要素型のうち，ASD 値でも元の要素型定義が保存される要素型は，

$C \cup A \cup SE(A)$

と決定する．ここで，$SE(A)$ は A 中の要素型を定義するために必要なすべての下位要素型の集合である．先の例では $SE(A) = \{g,h,i,j,k\}$ である．また，これら以外の要素型は UNKNOWN 構造とする．したがって，変換後の ASD 値は DTD "a:**rep**(b:**seq**(c:UNKNOWN, f:**or**(g:**rep**(h:**text**), i:**seq**(j:**text**, k:**text**))))" を持つ．

A.4 変換規則

図 16 は 2 章の統合利用例に対する問合せ処理（付録 A.5）で用いられる変換規則である．RL1 から RL3 は，コンバータと選択演算子の可換性に基づく変換規則である．RL1 と RL2 は選択演算と複合コンバータ $\mathbf{P}_{(A_1,\ldots,A_n)\to A:SC,gi}(r)$ の交換則に基づき，選択演算を早い段階で適用するものである．RL3 は，コンバータによる構造化文書からリレーションへの変換後に選択演算を行う式に対して，コンバータ適用前に，あらかじめ選択条件を満たす可能性のない構造化文書を除去する規則である．これらの規則を利用して，データ

をメディエータに転送する前に，各情報源がデータの絞り込みを行うことができる．

RL4 と RL5 は各情報源の問合せ処理能力を活用するために，問合せ式中の各演算を，情報源で処理可能なドメインでの演算に変換するための変換規則である．RL4 は SD 型ドメインの選択条件を持つ選択演算を，文字列ドメインでの選択演算としてドメイン変換子 γ より先に行うためのものである．この規則によって，SD 値に関する選択条件が，文字列の単語包含述語 "∋" に書き換えられる．また，RL5 を適用することにより，文字列ドメインでの選択演算に先立って，SD 型ドメインでの選択演算を行う．この規則は演算子 $\sigma_{\rho(attr,expr)}$ を含まない問合せ式に対して，他の変換規則を適用可能にするという働きもある．

A.5 統合利用例に対する問合せ処理

2 章の統合利用例の問合せ式（付録 A.2）に対する，5 章の問合せ処理の適用を以下に示す．

[Step 1] メディエータが，変換規則を利用して問合せ式を変形する．問合せ式に付録 A.4 の変換規則を適用すると，以下の式が得られる．まず，元の問合せ式においてメディエータが Ans を求める処理（操作 4）中にあった，SD 値に関する拡張選択演算の代わりに，G が "Faculty" に対する選択演算を行う．この選択演算はリレーショナルデータベースでローカルに実行される．さらに，H' 中の $\sigma_{\rho(Doc,\sigma[''A\text{-}univ'']\langle\mathbf{I}\rangle)}(Document)$ によって，文書リポジトリ中の文書のうち "A-univ" を含まないものがあらかじめ除去されるので，メディエータに転送しなければならないデータ量が削減される．

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

$$H': Ans2 \leftarrow \sigma_{\rho(Doc,\sigma[''A\text{-}univ'']\langle\mathbf{I}\rangle)}(Document)$$

$$F': Ans \leftarrow \mathbf{P}_{(Name,D\text{-}Name,A\text{-}Info,Pubs)\rightarrow Table:SC}(\mathbf{P}_{(Speciality,Course)\rightarrow A\text{-}Info:SC,'academic\text{-}info'}(\mathbf{P}_{Ps(Pub)\rightarrow Pubs:RC}(\nu_{Ps=(Pub)}(\mathbf{P}_{(Title,Pub\text{-}Info)\rightarrow Pub:SC}(\pi_{Name,D\text{-}Name,Speciality,Course,Title,Pub\text{-}Info}(Ans1 \bowtie_{Name=A\text{-}Name} (\sigma_{Affiliation=\text{“A-univ”}}(\mathbf{U}_{Author\rightarrow(A\text{-}Name,Affiliation)}(\mu_{As}(\mathbf{U}_{Authors\rightarrow As(O_1,Author)}(\mathbf{U}_{Doc\rightarrow(Title[title-title\subset ref],Authors[authors-authors\subset ref],Pub\text{-}Info[pub\text{-}info-pub\text{-}info\subset ref])}(Ans2)))))))))))$$

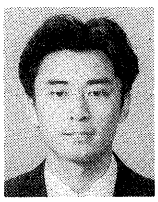
[Step 2] F' より，Unpack 演算時に不可欠な要素型が "title," "authors," "ref," "pub-info," "author," "a-name," "affiliation" であることが分かる．さらに，メディエータが ASD 値ではない具体的な SD 値を必要とする要素型は，選択条件に現れる属性 "A-Name" と "Affiliation" に含まれる，"a-name" と "affiliation" であることが分かる．他の属性の SD 値は，最終結果を求める段階まで具体化される必要がない．したがって，H における抽象化指定子は，(a-name, affiliation; title, authors, ref, pub-info, author) となる．その結果，H は次式となる．

$$H: Ans2 \leftarrow \mathbf{SDA}_{Doc,(a\text{-}name,affiliation;title,authors,ref,pub\text{-}info,author)}(\sigma_{\rho(Doc,\sigma[''A\text{-}univ'']\langle\mathbf{I}\rangle)}(Document))$$

最後に，F' において SD 値の具体化が必要な位置に **SDM** 演算子を挿入し，F を得る．

$$F: Ans \leftarrow \mathbf{SDM}_{Table}(F')$$

(平成 9 年 8 月 29 日受付)
(平成 10 年 2 月 2 日採録)



森嶋 厚行（学生会員）

1970 年生．1993 年筑波大学第三学群情報学類卒業．現在，同大学大学院工学研究科在学中．異種分散情報源統合利用方式，半構造化データ管理等に興味を持つ．ACM 会員．

北川 博之（正会員）

1955 年生．1978 年東京大学理学部物理学科卒業．1980 年同大学大学院理学系研究科修士課程修了．日本電気（株）勤務の後，1988 年筑波大学電子・情報工学系講師．同助教授を経て，現在，同大学電子・情報工学系教授．理学博士（東京大学）．異種分散情報源統合，文書データベース，問合せ処理，分散情報検索等に興味を持つ．著書「データベースシステム」(昭晃堂)「The Unnormalized Relational Data Model」(共著，Springer-Verlag) 等．電子情報通信学会，日本ソフトウェア科学会，ACM，IEEE-CS 各会員．