

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

森嶋 厚行

北川 博之

筑波大学大学院 工学研究科

筑波大学 電子・情報工学系

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。我々が提案している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ためのデータモデルである NR/SD モデルでは、構造化文書を抽象データ型“構造化文書型”の値として扱い、構造化文書と入れ子型リレーション構造の動的変換を利用したデータ操作が可能である。しかし、NR/SD モデルを現実の応用に適用するためにはいくつかの問題点の解決を図ることが必要であることが明らかとなった。本稿では、NR/SD モデルのコンセプトを引き継ぎかつ現実の応用により対応した改良モデルである NR/SD+ について説明する。NR/SD+ 固有の特徴として、(1)SGML 等に現れる様々な文書構造に柔軟に対応可能、(2)多様な構造を持つ構造化文書の下位要素に対してより直接的な操作が可能、という点がある。NR/SD+ では構造化文書間の参照構造を動的に生成する仕組みを利用して統合操作を実現する。

Integration of Structured Documents and Databases Featuring Dynamic Creation of References

Atsuyuki Morishima*

Hiroyuki Kitagawa†

*Doctoral Degree Program in Engineering, Univ. of Tsukuba

†Institute of Information Sciences and Electronics, Univ. of Tsukuba

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 *NR/SD+*, a revised version of NR/SD model we developed as a basic data modeling framework for the seamless integration of structured documents and relational databases. NR/SD+ (and 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. Features of NR/SD+ include its flexibility to cope with various structural constructs which occur in SGML-compliant documents, and capability of direct extraction and manipulation of text elements which appear inside the document hierarchy. These features are achieved by dynamic creation of references to structured document fragments.

1 はじめに

近年、コンピュータネットワークの発達に従い、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, 5]。各種ある情報資源の中でも、構造化文書は WWW やデジタル図書館などで主要な役割を果たしており、最も重要な情報資源の一つである。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統合利用環境の研究開発を行なっている [3, 4]。本統合利用環境では、構造化文書を格納する文書リポジトリとリレーショナルデータベースを統合する基本的枠組み

として、ソフトウェアモジュールであるラッパーとメダイエータを利用する (図 1)。ラッパーは各情報資源を共通モデルに変換し、メダイエータはそのモデルに基づいた統合スキーマと操作環境を提供する。我々はこの統合利用環境で用いる共通モデルとして、NR/SD モデルを開発した [3, 4]。

NR/SD モデルは、入れ子型リレーショナルモデルに、構造化文書を扱うための抽象データ型“構造化文書型”(SD 型)を導入したものである。NR/SD モデルでは、入れ子型リレーショナル代数演算を提供すると

同時に、SD 型の値 (SD 値) に対して文書内容にもとづく検索が適用可能である。したがって、リレーションと構造化文書に対する操作を共に提供している。

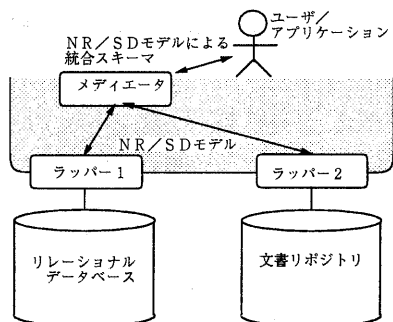


図1: 文書リポジトリとリレーションデータベースの統合利用環境

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、(1) 動的 (2) 双方向 (3) 部分的に相互変換する演算子“コンバータ”にある。(1)と(2)の性質により、同一のデータに対して入れ子型リレーション代数系と文書検索操作の両者が適用できる。したがって次のようなことが可能である。(a) 入れ子型リレーション代数系を利用して、構造化文書の構造変換操作等を行なう。(b) 入れ子型リレーション構造を構造化文書に変換し、型と独立した検索やメタデータを手がかりとした検索等を行なう。さらに(3)の性質をもつことから、データの共通構造を入れ子型リレーション構造で、非共通構造を構造化文書として表現することにより、(c) 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ことが可能である。これは、構造化文書では、スキーマとインスタンスが明確に区別されるリレーション構造と異なり、文書構造情報が各文書中に隠蔽され、全ての構造化文書が単一ドメインに属するという性質を利用している。このような、データ構造を抽象化して扱う能力は、異種情報資源の統合利用に威力を発揮するのはもちろんのこと、データ構造の規定が緩やかでインスタンスごとに構造が異なるような、構造化文書を始めた準構造化データの操作に、効果的であると考えられる。

しかし、NR/SD モデルを現実の統合利用環境へ適用する際には、以下のような問題がある。

(1) コンバータによる変換の対象となる、入れ子型リレーション構造とSD 値の文書構造の關係に制約がある。つまり、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、この制約は必ずしも必然性のあるものではない。例えば、NR/SD モデルでは文書の列構造とリレーションの属性列の相互変換を行なうコンバータが用意されているが、文書の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るのは、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と、多くのコンバータの導入が必要となる。

(2) 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 値の階層構造にお

ける上位要素群に限定されている。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。さらに、異なる DTD を持つ文書が混在する場合は、DTD ごとに場合分けをしたうえで、上記の操作を行わなければならない。

そこで、本稿ではNR/SD モデルを改良しこれらの問題点を解決したモデルであるNR/SD+ について説明する。NR/SD+ はNR/SD モデルと同等以上のデータ操作記述力を保持し、さらに以下のような固有の特徴をもつ。(1) コンバータによる変換対象となる、文書構造とリレーション構造の關係が直交している。(2) 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ない。(3)SGML 文書等に現れる例外構造、再帰構造、などに対応している。

2 NR/SD+ のデータ構造

NR/SD+ のデータ構造は、NR/SD モデルと同様、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”(SD 型)を導入したものである。ただし、SD 型は例外構造、再帰構造などを扱えるように拡張している。

2.1 SD 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 (Data 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。テキスト中で同じ名前のタグで区切られた部分を要素と呼ぶ。各要素の構造の定義はDTD の要素型定義で行なう。図2のSD 値は、proceedings や paper 等の要素から構成されている。proceedings は、副要素 confinfo や committees 等を並べた列構造を持つ。ただし、sponsors は省略可能である。confinfo は、副要素 c-name 等の順不同の並びからなる。p-body は副要素 section の繰返し構造からなり、内部の任意位置に fig を含むことができる。

proceedings	= seq(confinfo, committees, opt(sponsors) toc, rep(paper), index)
confinfo	= and(c-name, venue, date)
paper	= seq(title, authors, abstract, p-body, reference)
authors	= rep(author)
author	= seq(name, affiliation)
p-body	= rep(section) + fig
section	= seq(sectitle, rep(para), opt(rep(section)))
...	
<pre> <proceedings> <confinfo><c-name>International Conference on Integration of Heterogeneous Information Repositories/<c-name><date>June 4, 1997</date> <venue>Tsukuba, Japan/<venue></confinfo> <committees> ... </committees> <toc> ... </toc> <paper> ... </pre>	

図2: SD 値 (一部)

3 NR/SD+ 代数

NR/SD+ 代数の主要な演算子は、コンバータ、マスタコンストラクタ、マスタアダプタ、入れ子型リレーション代数演算子である。

A	B		
1	$\langle a:\text{rep}(c:\text{and}(d, b)), \langle a \rangle \langle c \rangle \langle b \rangle T1 \langle /b \rangle \langle d \rangle T2 \langle /d \rangle \langle /c \rangle \langle c \rangle \langle d \rangle T3 \langle /d \rangle \langle b \rangle T4 \langle /b \rangle \langle /c \rangle \langle c \rangle \langle b \rangle T5 \langle /b \rangle \langle d \rangle T6 \langle /d \rangle \langle /c \rangle \langle /a \rangle \rangle$		
2	$\langle d:\text{rep}(b), \langle d \rangle \langle b \rangle T7 \langle /b \rangle \langle b \rangle T8 \langle /b \rangle \langle b \rangle T9 \langle /b \rangle \langle /d \rangle \rangle$		
3	$\langle e:\text{seq}(b, c:\text{rep}(b)), \langle e \rangle \langle b \rangle T10 \langle /b \rangle \langle c \rangle \langle b \rangle T11 \langle /b \rangle \langle b \rangle T12 \langle /b \rangle \langle /c \rangle \langle /e \rangle \rangle$		
A	B	C	
		O	D
1	$\langle a:\text{rep}(c:\text{and}(d, b)), \langle a \rangle \langle c \rangle \langle b \rangle T1 \langle /b \rangle \langle d \rangle T2 \langle /d \rangle \langle /c \rangle \langle c \rangle \langle d \rangle T3 \langle /d \rangle \langle b \rangle T4 \langle /b \rangle \langle /c \rangle \langle c \rangle \langle b \rangle T5 \langle /b \rangle \langle d \rangle T6 \langle /d \rangle \langle /c \rangle \langle /a \rangle \rangle$	1	$\langle b, \langle b \rangle T1 \langle /b \rangle \rangle$
2	$\langle d:\text{rep}(b), \langle d \rangle \langle b \rangle T7 \langle /b \rangle \langle b \rangle T8 \langle /b \rangle \langle b \rangle T9 \langle /b \rangle \langle /d \rangle \rangle$	2	$\langle b, \langle b \rangle T5 \langle /b \rangle \rangle$
3	$\langle e:\text{seq}(b, c:\text{rep}(b)), \langle e \rangle \langle b \rangle T10 \langle /b \rangle \langle c \rangle \langle b \rangle T11 \langle /b \rangle \langle b \rangle T12 \langle /b \rangle \langle /c \rangle \langle /e \rangle \rangle$	1	$\langle b, \langle b \rangle T7 \langle /b \rangle \rangle$
		1	$\langle b, \langle b \rangle T10 \langle /b \rangle \rangle$
		2	$\langle b, \langle b \rangle T11 \langle /b \rangle \rangle$

図3: コンバータ U/P の適用例 (リレーション r_1 (上) と r_2 (下))

3.1 コンバータ

NR/SD+ は2つのコンバータ Unpack と Pack を持つ。コンバータ Unpack (U) は SD 値中の要素群を値として持つ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リージョン代数式を用いて指定する。抽出すべき要素群の指定にリージョン式を用いることで、SD 値中の上位要素群以外の要素とリレーション構造の変換が可能となる。Pack (P) は副リレーション構造中の要素を持つような SD 値を作成する。まず、リージョン代数 [1] について説明し、続いてコンバータ Unpack と Pack の説明を行なう。

リージョン代数

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。リージョンは組 (開始位置, 終了位置) であらわされる。例えば, “<name> <fn> Taro </fn> <ln> Tsukuba </ln> </name>” に対するリージョン (2, 4) は, “<fn> Taro </fn>” を表す。NR/SD+ ではこの例のように、リージョンを文書中の各要素に対応するものに限定して考える。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り、リージョン代数式 e は以下の構文規則を満たす。

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

ただし, R_i は総称識別子 (“para” など), 自然数 (“3” など), “I”, “A” のいずれかである。 R_i が総称識別子の場合は, リージョン代数式 “ R_i ” は, タグ付きテキスト中の要素のうち, 総称識別子が R_i であるものに対応するリージョンの集合を表す。 R_i が自然数 i の場合は, タグ付きテキスト中で, いずれかの要素の第 i 副要素として現れる要素に対応するリージョンの集合を表す。 R_i が “I” である場合は, タグ付きテキスト全体を構成する要素に対応するリージョンただ一つを含む集合を表す。“A” は $U_i R_i$ を表す。和演算子 (U), 積演算子 (∩), 差演算子 (−) は通常の集合演算子である。包含演算子 (⊃), 後置演算子 (>) は, 以下のよう

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

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真であり, $r > s$ は r が s より後方に位置するときに真である。

被包含演算子 $\subset$ と 前置演算子 $<$ も同様に定義される。また, 直接包含演算子 ($\supset_d$), 直接被包含演算子 ($\subset_d$) は以下のように定義される。

$$R \supset_d S = \{r \in R : \exists s \in S, r \supset s \wedge \neg \exists t \in A, r \supset t \supset s\}$$

$$R \subset_d S = \{r \in R : \exists s \in S, r \subset s \wedge \neg \exists t \in A, r \subset t \subset s\}$$

選択演算子 (σ) は単語 “ w ” を含む要素を選択する。

Unpack/Pack

以下の式は Unpack (U) と Pack (P) の適用例である。結果は図3に示す。

$$r_2 := U_{B \rightarrow C(O, D[b \cap 1])} \text{ as } x(r_1) \quad (1)$$

$$r_1 := P_{C(O, D) \text{ as } x \rightarrow B(r_2)} \quad (2)$$

r_2 は r_1 に (1) 式を適用した結果である。ここでは, リージョン代数式として $b \cap 1$ が指定されている。したがって, 属性 C の各副リレーションには, 要素型 b を持つ要素のうち, 任意の要素の第一副要素として現れるものを持つ SD 値が含まれる。 r_2 の属性 B の SD 値に現れる $\&x.n$; を SD リファレンス (SD reference) と呼ぶ。SD リファレンスは, Unpack によって作成された副リレーション構造中の SD 値を指す参照である。SD リファレンスの参照元の 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4)。

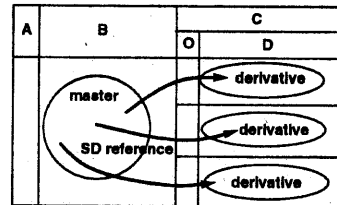


図4: マスタとデリバティブ

一方, r_2 に (2) 式を適用すると, 属性 C の各副リレーションを, r_2 の属性 B の SD 値 (マスタ) をテンプレートとして構造化文書に変換し, r_1 を得る。この際, マスタ中の SD リファレンスがそれぞれデリバティブのテキスト部によって置換される。P のパラメータとしてマスタを直接指定することも可能である。

3.2 マスタコンストラクタとマスタアダプタ

Pack を用いてリレーション構造を SD 値に変換する際には必ずマスタが必要とされる。しかし、そのリレーション構造が 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。コンバータとマスタコンストラクタを組み合わせることにより、変換対象となる文書構造とリレーション構造の直交化が図れる。

一方、Unpack によって作成されたマスタを用いて Pack を行なう際にも問題が生じる場合がある。NR/SD+ におけるコンバータの主要な目的の一つに、構造化文書と入れ子型リレーション構造を相互変換することによって、異なる操作体系による操作を提供するというものがある。しかし、Unpack → 入れ子型リレーション代数を用いた操作 → Pack の過程において、(a) デリバティブの個数に変更が生じた場合、また (b) デリバティブの文書構造に変更が生じた場合には、Pack 時に、Unpack によって作成されたマスタとデリバティブ間に不一致が生じる。不一致には (a) に起因する SD リファレンスの不一致と (b) に起因する DTD の不一致がある。これらの不一致を解決するために、マスタアダプタ (master adapter) と呼ぶ演算子群を用意する。マスタアダプタはデリバティブに基づいてマスタを変更することにより、不一致を解決する。

図 5 にコンバータとマスタコンストラクタ、マスタアダプタの関係を示す。

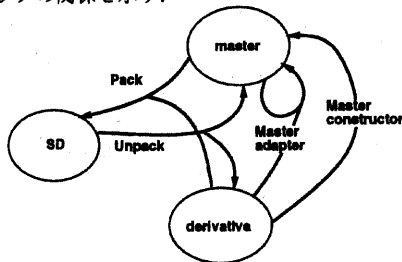


図 5: コンバータとマスタコンストラクタ、マスタアダプタの関係

マスタコンストラクタ

マスタコンストラクタ SC の適用例を次式に示す。マスタコンストラクタ SC は、副リレーション中の要素群をデリバティブとするような列構造を持つマスタを作成する (図 6)。

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

他に $\text{RC}_{C(O,D) \text{ as } x, B, G}(r)$, $\text{OC}_{C(O,D) \text{ as } x, B, G}(r)$ などがあり、それぞれ属性 D 中の値をデリバティブとするようなマスタを属性 B に作成する。その際、属性 G 中の値をタグとして利用する。作成されるマスタはそれぞれ、rep 構造、or 構造を持つ。

マスタアダプタ

マスタアダプタは、Unpack によって作成されたリレーション構造の操作結果を、マスタの構造に反映させる。したがって、構造化文書群に対するリレーションアルビューの操作を通じて、元の構造化文書の操作を行なうことができる。マスタアダプタには、Text Adapter (TA) と DTD Adapter (DA) がある。Text Adapter は、マスタとデリバティブ間における SD リファレンスの不一致を解決するために、マスタのタグ付きテキスト部を変更する。DTD Adapter は、マスタとデリバティブ間の DTD の不一致を解決するために、マスタの DTD を変更する。

- $\text{TA}_{M, C(O,D) \text{ as } x}(r)$: 属性 M のマスタの SD リファレンス $\&x.i$; と属性 C の副リレーションに含まれるデリバティブの不一致を、マスタのタグ付きテキスト部の変更によって解決する。テキスト部の変更で解決不可能なマスタは変更しない。
- $\text{DA}_{M, C(O,D) \text{ as } x}(r)$: 属性 M のマスタの DTD と属性 C の副リレーションに含まれるデリバティブの不一致を、マスタの DTD の変更によって解決する。DTD の変更では解決不可能なマスタについては変更しない。

次式と図 7 を用いて Text Adapter の適用例を示す。

$$r_6 = \text{TA}_{B, C(O,D) \text{ as } x}(r_5)$$

3.3 その他の演算子

NR/SD+ では、コンバータ、マスタコンストラクタ、マスタアダプタ、入れ子型リレーション代数演算子の他に、以下の演算子を持つ。

プリミティブな演算子

- ドメイン変換子 $\gamma_{A, \text{type}}(r)$: 入れ子型リレーション r の属性 A の型を type に変更する。
- $\tau_{A \rightarrow G}(r)$: A 中の SD 値の最上位要素のタグを取り出して属性 G に入れる。
- $o_{C(O,D), \leq}$: 属性 C の副リレーション構造に新たな属性 O を追加する。属性 O の値には、属性 D の値間の順序関係 $\leq$ に従って自然数を入れる。 $\leq$ が省略された場合は任意の順に自然数を割り当てる。
- $\delta_{A \rightarrow B}(r)$: 入れ子型リレーション r 中の属性名を A から B に変更する。

複合演算子

- $\text{AS}_{C(O,D) \rightarrow (A_1, A_2, \dots, A_n)}(r)$: 入れ子型リレーション r が持つ副リレーション構造 $C(O,D)$ を除去し、属性 D 中の値を持つ属性列 $A_1, A_2, \dots, A_n$ を追加する。
- $\text{TS}_{(A_1, A_2, \dots, A_n) \rightarrow C(O,D)}(r)$: 入れ子型リレーション r がもつ属性列構造 $A_1, A_2, \dots, A_n$ を除去し、これらの属性中の値を属性 D の値として持つような副リレーション構造 $C(O,D)$ を追加する。
- $\text{U}^+_{B \rightarrow (B_1[e_1] \text{ as } z_1, \dots, B_n[e_n] \text{ as } z_n)}(r)$: 属性 B の SD 値からマスタとデリバティブを作成する。マスタは属性 B に、デリバティブは新たな属性列 $(B_1, \dots, B_n)$ に格納される。

A	C		G
	O	D	
1	1	$\langle a, \langle a \rangle T1 \langle /a \rangle \rangle$	f
	2	$\langle b, \langle b \rangle T2 \langle /b \rangle \rangle$	
	3	$\langle c, \langle c \rangle T3 \langle /c \rangle \rangle$	

A	C		B
	O	D	
1	1	$\langle a, \langle a \rangle T1 \langle /a \rangle \rangle$	$\langle f:seq(a,b,c), \langle f \rangle x.1; x.2; x.3; \langle /f \rangle \rangle$
	2	$\langle b, \langle b \rangle T2 \langle /b \rangle \rangle$	
	3	$\langle c, \langle c \rangle T3 \langle /c \rangle \rangle$	

図 6: マスタコンストラクタ SC の適用例 (リレーション r_3 (左) と r_4 (右))

A	B	C	
		O	D
1	$\langle a:rep(b:seq(c, d)), \langle a \rangle \langle b \rangle \langle c \rangle T1 \langle /c \rangle x.1; \langle /b \rangle \langle b \rangle x.2; \langle d \rangle T4 \langle /d \rangle \langle /b \rangle \langle /a \rangle \rangle$	1	$\langle d, \langle d \rangle T2 \langle /d \rangle \rangle$
		2	$\langle c, \langle c \rangle T3 \langle /c \rangle \rangle$
		3	$\langle d, \langle d \rangle T5 \langle /d \rangle \rangle$

A	B	C	
		O	D
1	$\langle a:rep(b:seq(c, d)), \langle a \rangle \langle b \rangle \langle c \rangle T1 \langle /c \rangle x.1; \langle /b \rangle \langle b \rangle x.2; \langle d \rangle T4 \langle /d \rangle \langle /b \rangle \langle c \rangle \langle /c \rangle x.3; \langle /b \rangle \langle /a \rangle \rangle$	1	$\langle d, \langle d \rangle T2 \langle /d \rangle \rangle$
		2	$\langle c, \langle c \rangle T3 \langle /c \rangle \rangle$
		3	$\langle d, \langle d \rangle T5 \langle /d \rangle \rangle$

図 7: マスタアダプタ TA の適用例 (リレーション r_5 (上) と r_6 (下))

- $P^+(B_1, as\ x_1, \dots, B_n, as\ x_n) \rightarrow B(r)$: 属性 B の SD 値をマスタとし、属性列 $(B_1, \dots, B_n)$ の値をデリバティブとして Pack を行なう。
- $P-RC_{C(O,D) \rightarrow B, gi}(r)$: $C(O, D)$ 中のデリバティブに基づき RC でマスタを作成し新たな属性 B に格納した後、 P を行なう。
- $P-SC-TS_{(B_1, \dots, B_n) \rightarrow B, gi}(r)$: $(B_1, \dots, B_n)$ 中のデリバティブに基づき SC でマスタを作成し新たな属性 B に格納した後、 P を行なう。

4 NR/SD+ を用いた統合操作例

統合利用環境に、文書リポジトリと 2 つのリレーショナルデータベースが存在すると仮定する。文書リポジトリには、図 2 の DTD を持つ予稿集と、予稿集とはかなり異なる DTD (図 8) を持つ書籍が格納されている。リレーショナルデータベース A には T 大学の教員リレーション “Faculty” と学科リレーション “Department” が格納されている。また、リレーショナルデータベース B には T 大学の書籍部の在庫情報リレーション “Stock” が格納されている。メディアータ上ではこれらは図 9 のスキーマを通じて操作できる。ここで、文書リポジトリは、SD 型の属性を持つ単項リレーション “Document” として表現されている。ここでは以下の各問合せに対する NR/SD+ によるデータ操作を示す。

Q1. T 大学の教授によって書かれた論文と書籍の書誌情報を求めよ。結果は、学科ごとにまとめた構造化文書とせよ。

$$r_7 := U_{Author \rightarrow (Name, Affiliation)}(\mu Authors(\mu_{Pub \rightarrow Authors}(O_2, Author[author \cup authorinfo]) (\mu_{Pub \rightarrow (Title)}(\mu Pubs(U_{Doc \rightarrow Pubs}(O_1, Pub[paper \cup book]) (U_{Doc \rightarrow (Pub-Info[confinfo \cup b-pubinfo]) (Document))))))))$$

$$r_8 := \sigma_{Affiliation='T-univ'}(r_7) \bowtie_{Name=Name} \sigma_{F-Title='Prof'}(Faculty) \bowtie_{DID=DID} Department$$

$$r_9 := \pi_{DID, D-Name, Title, Name, Pub-Info}(r_8)$$

$$r_{10} := P-SC-TS_{(DID, D-Name, Pubs) \rightarrow Table}(P-RC_{Publications}(O_3, Pub) \rightarrow Pubs(\cup Publications(O_3, Pub) (\cup Publications=(Pub)(P^+(Title, Name, Pub-Info) \rightarrow Pub[pub:seq(title, name, pi:or(confinfo, b-pubinfo)), \langle \langle pub \rangle \& Title.1; \& Name.1; \langle pi \rangle \& Pub-Info.1; \langle /pi \rangle \langle /pub \rangle \rangle] (r_9))))))$$

この操作例は、NR/SD+ が、構造化文書とリレーションという表現の異種性と、DTD の相違というデータ構造の異種性に対処できることと、コンバータが構造化文書中の下位部分と入れ子型リレーション構造を直接相互変換できることを示す。結果の構造化文書の DTD を図 10 に示す。

book	= seq(title, b-authors, b-pubinfo, toc, b-body, index)
b-authors	= rep(authorinfo)
authorinfo	= seq(name, profile)
profile	= and(affiliation, history, interests)
b-pubinfo	= seq(opt(series, publisher, year)
b-body	= rep(chapter) + fig
...	

図 8: 書籍の DTD

Faculty					
FID	Name	DID	F-Title	Course	Speciality

Department		
DID	D-Name	Head

Stock			
PID	Title	Publisher	Num

Document
Doc

図 9: 統合スキーマ例

table	= seq(did, d-name, pubs)
pubs	= rep(pub)
pub	= seq(title, name, pi)
pi	= or(confinfo, b-pubinfo)

図 10: Q1 の結果作成される構造化文書の DTD

次はマスタアダプタを利用して、構造化文書の部分的な構造変換を行う例である。

Q2. Q1 で作成した、図 10 の DTD をもつ書誌情報リストが存在する。このリストから論文を削除して書籍だけを残し、さらに各書籍について大学の書籍部での在庫情報を付加せよ。

論文の除去:

$$\begin{aligned} r_{11} &:= \mathbf{U}_{Table \rightarrow Pubs}(O_4, Pub[confinfo \cup b-pubinfo])(r_{10}) \\ r_{12} &:= \nu_{Pubs=(O_4, Pub)}(\sigma_{dtd(Pub)=b-pubinfo}(\mu_{Pubs}(r_{11}))) \\ r_{13} &:= \mathbf{P}_{Pubs \rightarrow Table}(\mathbf{TA}_{Table, Pubs}(r_{12})) \\ \text{在庫情報の追加:} \\ r_{14} &:= \mathbf{U}_{B-Pubinfo \rightarrow (Publisher)}^+ \\ &\quad \mathbf{U}_{Pub \rightarrow (Title, Name, B-Pubinfo)}^+ \\ &\quad \mu_{Pubs}(\mathbf{U}_{Table \rightarrow Pubs}(O_4, Pub)(r_{13}))) \\ r_{15} &:= \mathbf{P}_{(Publisher) \rightarrow B-Pubinfo}^+ \\ &\quad \pi_{Table, O_4, Title, Name, B-Pubinfo, Publisher, Num} \\ &\quad r_{14} \bowtie_{Title=Title \wedge Publisher=Publisher} Stock)) \\ r_{16} &:= \nu_{Pubs=(O_4, Pub)}(\\ &\quad \mathbf{P-SC-TS}_{(Title, Name, B-Pubinfo, Num) \rightarrow Pub}(r_{15})) \\ r_{17} &:= \mathbf{P}_{Pubs \rightarrow Table}(\mathbf{DA}_{Table, Pubs}(r_{16})) \end{aligned}$$

ここで重要な点は、図 10 の DTD において、要素“pub”より上位の文書構造がいかに複雑であろうと、その文書の構造が本質的に「“confinfo”もしくは“b-pubinfo”の繰返し」や「“pub”の繰返し」とみなすことができるかぎり、マスタアダプタを用いた上式は無変更で適用可能であるということである。

5 NR/SD 代数演算子との関係

NR/SD+ 代数の演算子を組み合わせることにより、NR/SD 代数の演算子と等価な演算が定義できる。以下に、NR/SD のコンバータと等価な NR/SD+ 代数式を示す。

$$\begin{aligned} \mathbf{RU}_{B:(O,D),G}(r) &= \delta_{E \rightarrow B}(\pi_{-B}(\tau_{B \rightarrow G}(\mathbf{U}_{B \rightarrow E(O,D[AC_d I])} \text{ as } x(r)))) \\ \mathbf{SU}_{B=(B_1, \dots, B_n),G}(r) &= \pi_{-B}(\tau_{B \rightarrow G}(\mathbf{AS}_{E(O,D) \rightarrow (B_1, \dots, B_n)}(\\ &\quad \mathbf{U}_{B \rightarrow E(O,D[AC_d I])} \text{ as } x(r)))) \\ \mathbf{RP}_{B:(O,D),G}(r) &= \delta_{E \rightarrow B}(\mathbf{P}_{B(O,D)} \text{ as } x \rightarrow E(\mathbf{RC}_{B(O,D)} \text{ as } x, E, G(r))) \\ \mathbf{SP}_{B=(B_1, \dots, B_n),G}(r) &= \mathbf{P}_{E(O,D)} \text{ as } x \rightarrow B(\mathbf{SC}_{E(O,D)} \text{ as } x, B, G(\\ &\quad \mathbf{TS}_{(B_1, \dots, B_n) \rightarrow E(O,D)}(r))) \\ \mathbf{OR}_{B,G}(r) &= \delta_{D \rightarrow B}(\pi_{-(B,O)}(\tau_{B \rightarrow G}(\mu_E(\\ &\quad \mathbf{U}_{B \rightarrow E(O,D[AC_d I])} \text{ as } x(r)))) \\ \mathbf{OA}_{D,G}(r) &= \nu_{C=(O,D)}(\delta_{E \rightarrow D}(\mathbf{P}_{F(O_1,H)} \text{ as } x \rightarrow E(\\ &\quad \mathbf{TS}_{(D) \rightarrow F(O_1,H)}(\mu_C(\mathbf{OC}_{C(O,D)} \text{ as } x, E, G(r)))))) \end{aligned}$$

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 モデルにおけるいくつ

かの問題点を解決したモデルである。NR/SD+ について説明した。具体的な改良点は以下の通りである。

(1) NR/SD モデルでは「特定の文書構造-特定の入れ子型リレーション構造」ごとに存在したコンバータの機能を、よりプリミティブな演算子の組合せで実現することにより、変換対象となる文書構造と入れ子型リレーション構造の関係を直交化した。またその結果、多様な文書構造に対応する際に追加しなければならない演算子の個数が減り、柔軟な対応が可能になった。

(2) Unpack におけるリージョン代数式の利用と SD リファレンスにより、SD 値と入れ子型リレーション構造の変換において、入れ子型リレーション構造中の SD 値(デリバティブ)が必ずしも変換対象となる SD 値の上位要素群に対応する必要がなくなった。

また、NR/SD+ を用いた統合操作例を示し、さらに NR/SD モデルのコンバータと等価な NR/SD+ 代数式を示した。

現在、NR/SD+ に基づく統合利用環境のプロトタイプシステムを開発中である。また、視覚的なユーザインタフェースやモバイルエージェントを利用した統合利用方式、WWW 環境への NR/SD+ の適用などについても研究を進めている。特に今後の WWW 環境では、XML[6] の導入により DTD で定義された文書構造が重要な意味を持つようになると考えられるため、NR/SD+ を WWW とリレーショナルデータベースの統合利用に適用することが効果的であると考えられる。

謝辞

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)ならびに基盤研究(C)(09680321)の助成による。

参考文献

- [1] M. P. Consens and T. Milo, “Algebras for Querying Text Regions,” *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,” *Proc. ACM SAC'97*, pp. 78-87, Feb. 1997.
- [4] A. Morishima and H. Kitagawa, “A Data Modeling and Query Processing Scheme for Integration of Document Repositories and Relational Databases,” *Proc. DASFAA '97*, pp. 145-154, April 1997.
- [5] 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.
- [6] “Extensible Markup Language (XML),” World Wide Web Consortium Working Draft, <http://www.w3.org/pub/WWW/TR/WD-xml.html>.