2004－DBS－134　(I)　(30)
2004／7／13

実体化XMLビューの差分更新手法

能登谷淳一†　　古瀬　一隆††

† 秋田県立大学 システム科学技術学部　〒 015–0055 秋田県本荘市土谷字海老ノ口 84–4
†† 筑波大学 電子・情報工学系　〒 305–8573 茨城県つくば市天王台 1–1–1
E-mail: †notoya@akita-pu.ac.jp, ††furuse@is.tsukuba.ac.jp

あらまし　本研究では問い合わせ言語として XQuery を利用する XML データベースシステムのための実体化ビュー差分更新手法を提案する．本研究で提案する差分更新手法は，多重集合代数および FlexKey による順序付けに基づく方法であり，既存の XML ビュー管理手法と比較し以下の三点の特徴を持つ．(1)XQuery の演算対象である XML 文書構造とノードのリストの双方を実体化ビューとして管理する．(2)XQuery 中で利用される二種類の順序，即ち文書中の要素出現順およびシーケンス中の要素出現順を維持する．(3) 実体化ビュー更新の一括実行 (Deferred update) に利用可能である．これらの特徴により，XML データベースシステムにおける実体化ビューの管理と利用が関係データベースシステムと同程度の水準で実現可能となる．

キーワード　XML, XQuery, 実体化ビュー, 差分更新

Incremental Update for Materialized XML Views

Juichi NOTOYA† and Kazutaka FURUSE††

† Faculty of Systems Science and Technology, Akita Prefectural University
Ebinokuchi 84–4, Tsuchiya, Honjyo, Akita 015–0055 Japan
†† Institute of Information Sciences and Electronics, University of Tsukuba
Tennodai 1–1–1, Tsukuba, Ibaraki 305–8573 Japan
E-mail: †notoya@akita-pu.ac.jp, ††furuse@is.tsukuba.ac.jp

Abstract　In this paper we present an incremental maintenance method for XQuery materialized views. This method is based on the bag algebra and the FlexKey ordering method. It has three novelties. 1) It maintains both lists of nodes and XML document structures as materialized views. 2) It maintains both sequence orders and XML document orders in materialized views. 3) It can be used for both immediate update and deferred update. These features enable XML DBMSs to maintain and use materialized views as capable as RDBMSs.

Key words　XML, XQuery, Materialized View, Incremental Update

1. はじめに

XML データベースに対する実体化ビューの利用が XML データベースの問い合わせ処理の分野で注目されている．

実体化ビューは関係データベースシステムにおいては，問い合わせ処理の高速化，可用性の増大などを目的として広く利用されてきた．XML データベースシステムでは，これらの用途に加えて異種データの統合利用への応用など，実体化ビューの応用範囲の拡大が

期待され，重要性が認識されている．

実体化ビューを利用したデータベースの効率的運用のためには，実体化ビューの差分更新技術が不可欠である．関係データベースシステム上の実体化ビューの管理に関しては，関係代数を理論的背景とした差分更新方式が提案され利用されてきた [1]．これに対し，問い合わせ言語として XQuery [2] を利用する多くの XML データベースシステムでは，実体化ビューの差分更新機能は提供されていない．従来の関係代数の演算対象である集合，多重集合が順序を持たない集まりであるのに対し，XQuery は多様な意味の順序を持つノードシーケンスを操作対象とする問い合わせ言語である．そのため，関係データベース上の実体化ビュー差分更新に関する研究はそのままでは XML データベースには適用することはできなかった．

Dimitrova らによるビュー更新手法である VOX [3] は，ノードシーケンスの順序を維持することを目標とした XML データベース上の実体化ビュー更新に関する初期の研究である．VOX においては，XQuery の演算対象の一部を構成するノードのリストを多重集合を利用して表現することで，XQuery 問い合わせの大部分に対して正しく動作する実体化ビュー更新手法を提供した．しかし VOX においては，XQuery の演算対象の構成要素のうち，XML 文書構造そのものに対する対応が不十分であったことから，以下のような問題があった．

- タグ付け演算を行った結果に対してパスステップ演算を行った場合，順序が保存されない．
- パスステップ演算において，子孫方向の軸以外を扱うことができない．
- アドホックなタプル ID 生成手法，ノード ID 生成手法を必要とする．
- 差分更新式の計算にビュー定義式の部分式の実体化を必要とし，式変形のみによる差分更新式の導出ができない．

また，VOX は XML のための問い合わせ代数として XAT を利用しているが

- XQuery ではノードシーケンスの展開を for 句で行うことができるのに対し，XAT においては展開とパスステップの機能が単一の演算子に結合されており，分離できない．
- XQuery Formal Semantics [4] における基本構文である if-then-else 式を扱うことができない．

などの点から，XQuery のための実体化ビュー更新手法の代数的基盤としては XAT は必ずしも適当ではないと考えられる．

本研究では，XQuery のための標準的な問い合わせ代数である XQuery-Core [4] の演算を多重集合代数を用いて表現し，その演算対象であるノードシーケンスを代数的に取り扱うことにより，上記の問題を解決した差分更新手法を提供する．また，多重集合代数の利用により，本研究で提案する差分更新手法は更新操作の一括実行（Deffered update）[5] に適したものとなる．

2. 多重集合による XQuery データ表現

本研究では，XQuery-Core の各演算を多重集合代数を用いて表現することにより，XML ビュー差分更新の代数的基盤とする．本研究で利用する多重集合代数は，以下のように再帰的に定義される型 t の上の代数である．

$$t ::= \mathbb{D}_i \,|\, \mathrm{unit} \,|\, t \times t \,|\, \mathbb{Z}^t$$

ここで $\mathbb{D}_i$ は基底型の一つを表す．unit は空タプル $()$ を唯一つの元として持つ型である．$s \times t$ は型の直積，$\mathbb{Z}^t$ は t の多重集合型を表す．

本研究では，多重集合代数の演算子として，BALG 代数 [6] および NRC 代数 [7] で利用される演算子表記法を組み合わせて利用する．本研究で利用する多重集合代数の基本演算子は，加算的合併 $e_1 \uplus e_2$，加算的合併に関する逆元の導出 $\dot{-}e$，シングルトン生成 $\{\!|e|\!\}$，タプル生成 $(e_1\ , e_2)$，射影 $\pi_i e$，ラムダ抽象 $\lambda x\,.\,e$，関数適用 $f(e)$，関数の自然拡張 $\mathrm{ext}(f)$，論理否定 $\neg e$ の 9 種類である．また，NRC 代数と同様にマップ演算，条件演算などの複合演算を適宜利用する．

BALG 代数と本研究で使用する多重集合代数との本質的違いは，本研究で利用する代数中の多重集合が元の出現数として負の整数を許すため，任意の多重集合 x に対し加法的合併に関する逆元 $\dot{-}x$ が必ず存在する点である．すなわち，本研究で利用する多重集合代数は，加算的合併演算に関して群をなす．また，多重集合代数は関数の自然拡張演算と加算的合併演算に関してモナド [8] をなす．本研究では，多重集合代数上で成立する群とモナドの規則を利用した式の変形によって XQuery-Core 上の差分更新式を導出する．

2.1 FlexKey による順序付けと識別

本研究では XML 文書構造およびノード間の順序を多重集合上で表すために，FlexKey エンコーディング手法 [9] を用いる．

FlexKey エンコーディングを利用してノードの識別子を生成することにより，単一のデータ構造によって XML 文書構造中のノードを一意に特定し，さらにノードの文書構造中での位置を表現することが可能となる．また，FlexKey の稠密性より，XML 文書へのノードの挿入や削除に際して，挿入や削除の対象となる XML 文書中のノードの識別子の変更を必要としないという特徴がある．この特徴は XML データベースの実体化ビュー上でノードの順序付けを表現する際に特に有用であると考えられる．

FlexKey エンコーディング手法においては，XML 文書中の各ノードにはローカルキーが付与される．ローカルキーは文字列であり，ノード n_2 がノード n_1 の兄ノードであるとき，n_1 のローカルキー lk_1 と n_2 のローカルキー lk_2 の間に辞書式順序関係 $lk_1 \prec lk_2$ が成立する．本稿ではローカルキー文字列を構成するアルファベットを Σ_α とする．

各ノードの FlexKey は，そのノードのローカルキーと親ノードの FlexKey により再帰的に定義される．すなわち，ノード n_i の FlexKeyfk_i は，

$$fk_i = \begin{cases} lk_i & n_i \text{ が根ノードのとき} \\ fk_j \text{ `/'} \ lk_i & \text{それ以外} \end{cases}$$

で与えられる．本稿では，FlexKey 構築時の区切り記号として ‘/’ を用いる．記号 ‘/’ は Σ_α に属さない記号であり，全ての $x \in \Sigma_\alpha$ に対して ‘/’ $\prec x$ である．

このように構成された FlexKey は XML 文書中の任意のノード n_1, n_2 に対し，以下の条件をみたす．

（1） $n_1 = n_2$ のとき，かつこのときに限り $fk_1 = fk_2$ である．

（2） n_1 が n_2 の祖先であるとき，かつこのときに限り fk_1 は fk_2 の接頭辞である．

（3） n_1 が n_2 より文書中で先に出現するとき，$fk_1 \prec fk2$ である．

一般にノードシーケンス中ではノードは文書順とは異なる順序で並べられる．そこで，FlexKey エンコーディング手法においては FlexKey の再構成により，ノードシーケンス中でのノードの並び順を表現する手法が与えられている．任意の FlexKey 文字列を記号 ‘:’ を区切り記号として連結した文字列を FlexOrder 文字列と呼ぶ．記号 ‘:’ は Σ_α に属さない記号であり，全ての $x \in \Sigma_\alpha$ に対して ‘:’ $\prec x$ である．このように構成された FlexOrder 文字列の辞書式順序により，ノードシーケンス中でのノードの位置を表す．

本研究ではこれらに加えて，FlexKey 文字列，FlexOrder 文字列をローカルキー文字列に変換する手法を提供する．関数 enc(s) は引数に与えられた文字列 s 中の Σ_α に属さない文字 s_i を Σ_α の文字列 ‘#$n_{i1} \ldots n_{im}$;’ に辞書式順序を保存したまま変換する関数である．記号 ‘#’ $\in \Sigma_\alpha$ を任意の記号 $x \in \Sigma_\alpha$ に対して ‘#’ $\preceq x$ であるように定めることにより，関数 enc を用いて FlexKey 文字列および FlexOrder 文字列を辞書式順序を保ったままローカルキー文字列に変換することが可能である．

2.2 シーケンスシグネチャによるノードシーケンスの識別

XML データベースへの問い合わせの特徴として，エレメント構築演算の存在がある．XQuery-Core におけるエレメント構築演算は，ノードシーケンスおよびエレメント名から新たなエレメントノードを生成し，生成したエレメントノードを唯一の元とするノードシーケンスを返す演算である．

FlexKey を利用したノード識別を行う系においては，新たに生成するエレメントノードに対し，条件（1）～（3）を満足する FlexKey を与える必要がある．VOX では新たに生成されたノードに対してはアドホックに FlexKey の付与を行っていた．その結果，式によって生成された XML 文書に対しては，条件（2）および（3）が成立せず，文書構造中でのノードの位置を特定するために XDOM スケルトンと呼ばれる付加的なデータ構造を必要とした．

本研究では文字列として構成される識別子であるシーケンスシグネチャをノードシーケンスに付与し，エレメント構築演算時の FlexKey 生成に利用する．シーケンスシグネチャは，XQuery-Core の問い合わせ式がデータベース状態からノードシーケンスを関数的に決定することを利用する．すなわち，問い合わせ式の表現そのものをノードシーケンスの識別子として用いる．例えば，問い合わせ式 E_1, E_2 に対し，E_1 によって生成されるノードシーケンスのシグネチャを sig_1，E_2 によって生成されるノードシーケンスのシグネチャを sig_2 とする．このとき，$sig_1 = sig_2$ なら

ば，E_1 と E_2 は等価な式であり，任意のデータベース状態に対し，式 E_1 と E_2 は互いに等しいノードシーケンスを返す．したがって，シーケンスシグネチャとデータベース状態が決定すると，ノードシーケンスの値は一意に定まり，この意味でシーケンスシグネチャはノードシーケンスの識別子であると言える．

通常の問い合わせ式によって得られるノードシーケンスのシグネチャは各演算の定義式（表 1）により得られるが，XQuery-Core 問い合わせにおいては演算の結果として得られるノードシーケンスの他に，演算の内部で一時的に生成され利用されるノードシーケンスが存在する．すなわち XQuery-Core の FLWR 構文においては，入力ノードシーケンスは単一の要素からなるノードシーケンス（シングルトンシーケンスと呼ぶ）に分解され，各々のシングルトンシーケンスに対して特定の処理が適用される．FlexOrder を利用したノードシーケンスの実現においては，入力ノードシーケンスの各々の要素は FlexOrder 文字列によって特定可能であるので，入力ノードシーケンス s のシグネチャ sig とシングルトンシーケンスの要素 x を特定する FlexOrder 文字列 $order_x$ の連結によって，シングルトンシーケンスのシグネチャ $\mathrm{Ssig}(sig, order_x)$ とする．(注1)

$$\mathrm{Ssig}(sig, order_x) = sig \text{ `[' } order_x \text{ `]'}$$

2.3 多重集合代数による XQuery の実現

本研究では XQuery のノードシーケンスを多重集合代数の定義域 t の部分型 NS の値として表す．

$$\mathrm{NS} = \mathrm{SI} \times \mathbb{Z}^{\mathrm{FK}\times t} \times \mathbb{Z}^{\mathrm{FK}\times \mathrm{FO}}$$

ここで，$\mathrm{SI}, \mathrm{FK}, \mathrm{FO}$ はシーケンスシグネチャ文字列，FlexKey 文字列，FlexOrder 文字列の定義域であり，各々$\mathbb{D}_i$ の一つとする．すなわち，ノードシーケンス $s = (S, T, L) : \mathrm{NS}$ は，シーケンスシグネチャ S，XML 文書構造 T，およびノードリスト L の三つの属性を持つ組として表現される．XML 文書構造 T は XML 文書上のノードの識別子である FlexKey ID と，そのノードの持つ値 V の二つの属性を持つ関係である．ノードリスト L はノードリスト上のノードの識別子である FlexKey ID と，ノード中での位置を表す FlexOrder O の二つの属性を持つ関係である．

上記のデータ構造と多重集合代数の演算を利用した XQuery-Core のノードシーケンス演算の実現を表 1 に示す．ここでは，表記の簡単化のため XQuery-Core の通常の表記法ではなく，XQuery-Core の演算と同等の能力を持つ演算子と，ラムダ抽象，関数適用および XQuery Functions and Operators [10] によって定義される外部関数を用いた表記法を利用する．また，シーケンスシグネチャの導出式中において，「“E”」の形式の表記は，式 E の（値ではなく）表現そのものを表す．

2.3.1 データベース参照

XQuery-Core において，データベース中の XML 文書の参照は組み込み関数である doc を用いて行われる．本稿ではこれを演算子の一つとして扱い，δ によって表す．

$$\frac{r : \mathrm{URI}}{\delta[r] : \mathrm{NS}}$$

この演算により，URI r によって示されるデータベースに格納された XML 文書構造 T と，その最上位要素 t からなるシングルトンシーケンス $(\text{`}\boldsymbol{\delta}\text{[' } r \text{ `]'}, T, \{\!\{t\}\!\})$ を得る．

2.3.2 連　　接

複数のノードシーケンスを連接し，単一のノードシーケンスを構成する演算は，XQuery-Core においてはカンマ構文によって記述される．本稿では，これを $\overset{o}{\uplus}$ によって表す．

$$\frac{E_1 : \mathrm{NS} \quad E_2 : \mathrm{NS}}{E_1 \overset{o}{\uplus} E_2 : \mathrm{NS}}$$

この演算により，E_1 の後に E_2 を連接したノードシーケンスを得る．

2.3.3 関 数 拡 張

XQuery においては，ノードシーケンス中の各要素を対象とする関数適用を行うため，FLWR 構文が用いられる．本稿では，これを演算子 β によって表す．

$$\frac{E : \mathrm{NS} \quad F : \mathrm{NS} \to \mathrm{NS}}{\beta[F]E : \mathrm{NS}}$$

この演算により，E の各要素からなるシングルトンシーケンスに関数 F を適用した結果を全て連接して構成されるノードシーケンスを得る．

(注1)：式中では $\mathrm{Ssig}(\pi_{\mathrm{S}} E, \pi_{\mathrm{O}} x)$ を単に Ssig と省略して記述する．

表 1　多重集合代数による XQuery-Core 演算の実現
Table 1　Bag algebra representations of the XQuery-Core operators

e	$\pi_{\mathrm{S}}e$	$\pi_{\mathrm{T}}e$	$\pi_{\mathrm{L}}e$
$E_1 \overset{o}{\uplus} E_2$	$\pi_{\mathrm{S}}E_1$ ‘$\overset{o}{\uplus}$’ $\pi_{\mathrm{S}}E_2$	$\pi_{\mathrm{T}}E_1 \uplus \pi_{\mathrm{T}}E_2$	$\mathrm{map}(\lambda x\,.\,(\pi_{\mathrm{ID}}x\ ,\ \text{‘0:’}\ \pi_{\mathrm{O}}x))\pi_{\mathrm{L}}E_1 \uplus \mathrm{map}(\lambda x\,.\,(\pi_{\mathrm{ID}}x\ ,\ \text{‘1:’}\ \pi_{\mathrm{O}}x))\pi_{\mathrm{L}}E_2$
$\beta[F]E$	‘$\boldsymbol{\beta}$[’ “F” ‘]’ $\pi_{\mathrm{S}}E$	$\mathrm{ext}(\lambda x\,.\,\pi_{\mathrm{T}}F(\mathrm{Ssig}\ ,\ \pi_{\mathrm{T}}E\ ,\ \{\!\{x\}\!\}))\pi_{\mathrm{L}}E$	$\mathrm{ext}(\lambda x\,.\,\mathrm{map}(\lambda y\,.\,(\pi_{\mathrm{ID}}y\ ,\ \pi_{\mathrm{O}}x\ \text{‘:’}\ \pi_{\mathrm{O}}y))\pi_{\mathrm{L}}F(\mathrm{Ssig}\,,\pi_{\mathrm{T}}E\,,\{\!\{x\}\!\}))\pi_{\mathrm{L}}E$
$\rho[F]E$	‘$\boldsymbol{\rho}$[’ “F” ‘]’ $\pi_{\mathrm{S}}E$	$\pi_{\mathrm{T}}E$	$\mathrm{ext}(\lambda x\,.\,\mathrm{ext}(\lambda y\,.\,\mathrm{ext}(\lambda z.\ \text{if}\ (\pi_{\mathrm{ID}}y = \pi_{\mathrm{ID}}z)\ \text{then}\ \{\!\{(\pi_{\mathrm{ID}}x\ ,\ \mathrm{enc}(\pi_{\mathrm{V}}y))\}\!\}\ \text{else}\ \emptyset)\pi_{\mathrm{L}}F(\mathrm{Ssig}\ ,\ \pi_{\mathrm{T}}E\ ,\ \{\!\{x\}\!\}))\pi_{\mathrm{T}}F(\mathrm{Ssig}\ ,\ \pi_{\mathrm{T}}E\,,\{\!\{x\}\!\}))\pi_{\mathrm{L}}E$
$\phi[a]E$	‘$\boldsymbol{\phi}$[’ “a” ‘]’ $\pi_{\mathrm{S}}E$	$\pi_{\mathrm{T}}E$	$\mathrm{ext}(\lambda x\,.\,\mathrm{ext}(\lambda y.\ \text{if}\ a(\pi_{\mathrm{ID}}y\ ,\ \pi_{\mathrm{ID}}x)\ \text{then}\ \{\!\{(\pi_{\mathrm{ID}}y\ \ ,\ \ \pi_{\mathrm{O}}x\ \text{‘:’}\ \pi_{\mathrm{ID}}y)\}\!\}\ \text{else}\ \emptyset)\pi_2E)\pi_3E$
$\nu[n]E$	‘$\boldsymbol{\nu}$[’ “n” ‘]’ $\pi_{\mathrm{S}}E$	$\{\!\{(\mathrm{enc}(\text{‘}\boldsymbol{\nu}\text{[’}\ \text{“}n\text{”}\ \text{‘]’}\ \pi_{\mathrm{S}}E)\ ,\ n)\}\!\}\ \uplus\ \mathrm{ext}(\lambda x\ .\ \mathrm{ext}(\lambda y.\ \text{if}\ \mathrm{match}(\text{‘\^{}’}\ \pi_{\mathrm{ID}}x\ \text{‘/’}\ ,\ \pi_{\mathrm{ID}}y)\ \text{then}\ \{\!\{(\mathrm{replace}(\text{‘\^{}’}\ \pi_{\mathrm{ID}}x\ \text{‘/’}\ ,\ \mathrm{enc}(\text{‘}\boldsymbol{\nu}\text{[’}\ \text{“}n\text{”}\ \text{‘]’}\ \pi_{\mathrm{S}}E)\ \text{‘/’}\ \mathrm{enc}(\pi_{\mathrm{O}}x)\ \text{‘/’}\ ,\ \pi_{\mathrm{ID}}y)\ ,\ \pi_{\mathrm{V}}y)\}\!\}\ \text{else}\ \emptyset)\pi_{\mathrm{T}}E)\pi_{\mathrm{L}}E$	$\{\!\{(\mathrm{enc}(\text{‘}\boldsymbol{\nu}\text{[’}\ \text{“}n\text{”}\ \text{‘]’}\ \pi_{\mathrm{S}}E)\,,\text{‘0’})\}\!\}$
IF $(E_1\ ,\ E_2\ ,\ E_3)$	‘IF(’ $\pi_{\mathrm{S}}E_1$ ‘,’ “E_2” ‘,’ “E_3” ‘)’	if $\pi_{\mathrm{L}}E_1$ then $\pi_{\mathrm{T}}E_2$ else $\pi_{\mathrm{T}}E_3$	if $\pi_{\mathrm{L}}E_1$ then $\pi_{\mathrm{L}}E_2$ else $\pi_{\mathrm{L}}E_3$

2.3.4　整　　列

XQuery においては，XQuery-Core の FLWR 構文への拡張として出力リスト中での要素の順序変更を行う FLWOR 構文が提供されている．ここでは，FLWOR 構文は，ノードシーケンスの整列を行う演算子 ρ と関数拡張演算 β の組み合わせによって表現されるものと考える．

$$\frac{E : \mathrm{NS} \quad F : \mathrm{NS} \to \mathrm{NS}}{\rho[f]E : \mathrm{NS}}$$

この演算により，E と同一の要素を持ち，関数 F によって得られる内容に従った整列順を持つノードシーケンスを得る．

2.3.5　パスステップ

パスステップ演算は XML 文書構造をノードリストとして抽出する演算である．XAT ではパスステップ演算の軸として child 軸および descendant 軸の一部のみを利用可能であった．これに対し，本研究では XQuery で定義される全ての軸に対するパスステップ演算に対応するため，軸テスト述語を導入する．

$$\frac{E : \mathrm{NS} \quad a : \mathrm{FK} \times \mathrm{FK} \to \mathrm{boolean}}{\phi[a]E : \mathrm{NS}}$$

この演算により，E に属するノードから見て軸 a の位置に存在にするノードからなるノードシーケンスを得る．

軸テスト $a : \mathrm{FK} \times \mathrm{FK} \to \mathrm{boolean}$ は以下のように定義される FlexKey 上の二項関係である．これらは，第一引数に与えられた FlexKey が，第二引数で与えられた FlexKey の軸上に存在するとき真となる．

$$\begin{aligned}
\mathrm{child}(x,y) &= \mathrm{match}(\text{‘\^{}’}\ y\ \text{‘/[\^{}/]+’}, x)\\
\mathrm{descendant}(x,y) &= \mathrm{match}(\text{‘\^{}’}\ y\ \text{‘/.+’}, x)\\
\mathrm{parent}(x,y) &= \mathrm{match}(\text{‘\^{}’}\ x\ \text{‘/[\^{}/]+’}, y)\\
\mathrm{ancestor}(x,y) &= \mathrm{match}(\text{‘\^{}’}\ x\ \text{‘/.+’}, y)\\
\mathrm{following}(x,y) &= \neg\mathrm{descendant}(x,y)\\
&\quad \wedge \mathrm{strcmp}(x,y) > 0\\
\mathrm{preceding}(x,y) &= \neg\mathrm{ancestor}(x,y)\\
&\quad \wedge \mathrm{strcmp}(x,y) < 0
\end{aligned}$$

2.3.6　エレメント構築

XQuery にはドキュメント構築，エレメント構築など種々の構築演算が存在するが，本研究ではそれらを全てエレメント構築演算の一種であるものとして

扱う．

$$\frac{E : \mathrm{NS} \quad n : \mathrm{string}}{\nu[n]E : \mathrm{NS}}$$

この演算により，内容として E を持つ n エレメントからなるシングルトンシーケンスを得る．

2.3.7 条 件 演 算

XQuery-Core においては，条件演算は基本演算の一つであり，FLWR 構文の where 句として表現される選択演算等多くの演算が条件演算を用いて定義される．

$$\frac{E_1 : \mathrm{NS} \quad E_2 : \mathrm{NS} \quad E_3 : \mathrm{NS}}{\mathrm{IF}\ (E_1, E_2, E_3) : \mathrm{NS}}$$

この演算により，E_1 が真であるときには E_2 の値を，偽であるときには E_3 の値を得る．ノードシーケンスは，その第三項が $\emptyset$ と等しいときに偽，それ以外のときに真と見做される．

3. 実体化 XML ビューの差分更新

XQuery によって記述された問い合わせのうち，XML データベースに格納された XML 文書構造の状態を参照し，単一のノードシーケンスを得るようなものを XQuery ビューと呼ぶ．

いま，XML データベースの状態が s であるときビュー E を評価して得られる値を $E_{(s)}$ とする．このとき，

$$E_{(s')} = E_{(s)} \Leftarrow \Delta E_{(s,s')}$$

をみたす差分適用演算 $\Leftarrow$ と差分導出関数 $\Delta E_{(s,s')}$ が存在し，これらが $E_{(s')}$ を直接評価する場合と比較して十分小さなコストで計算可能であるならば，$E_{(s)}$ の値を実体化ビューとして保持することによって $E_{(s')}$ の計算コストを削減可能である．

本研究では，データベースの状態変化として，XML データベース上の文書へのノードの追加および削除によって表される更新を扱う．

3.1 差分導出関数

本研究では演算子によって構成されるビュー定義式を各演算子の被演算項である部分式に分解して考え，各部分式の差分値からビュー定義域の差分値を構成する．

差分導出関数により得られる差分値を表す型として，ノードシーケンスと同一の型 NS を利用する．また，差分適用演算 $\Leftarrow$ として以下の演算を考える．

$$(S, T, L) \Leftarrow (\Delta S, \Delta T, \Delta L) = (S, T \uplus \Delta T, L \uplus \Delta L)$$

データベース参照演算は部分式を持たず，データベースの変化そのものが差分となる．

$$\delta[r]_{(s')} = \delta[r]_{(s)} \Leftarrow \Delta\delta[r]_{(s,s')}$$
$$\Delta\delta[r] = (\text{`}\boldsymbol{\delta}\text{[' } r \text{ ']'}, \blacktriangle T \uplus (\dot{-}\blacktriangledown T), \emptyset)$$

ここで，$\blacktriangle T$ および $\blacktriangledown T$ は，データベース状態の s から s' への変化の過程で追加および削除されたノード集合である．

表 2 にデータベース参照演算以外の各演算子に対する差分導出関数(注2)を示す．演算子 o に対する差分導出関数 o^{Δ} は以下の等式をみたす関数であり，各演算子の定義より容易に導出可能である．

$$oE_{(s')} = o(E_{(s)} \Leftarrow \Delta E_{(s,s')})$$
$$= oE_{(s)} \Leftarrow o^{\Delta}(E_{(s)}, \Delta E_{(s,s')})$$

各演算子に対する差分導出関数を利用することにより，それらの演算子によって構成されるビュー定義式の差分導出関数を計算可能である．

以下に本手法による差分導出の例を示す．XQuery 言語によるビュー定義

```
element pub {
  for $x in doc("bib.xmll")/books/book
    if ($x/author/text()='notoya')
      then $x/title else()
}
```

は，演算子を利用した式

$$\begin{aligned} E = &\nu[\text{`}\texttt{pub}\text{'}]\beta[\lambda x.\ \mathrm{IF}\ ( \\ &(\beta[\lambda y.\ \mathrm{IF}\ (y = \text{`}\texttt{notoya}\text{'}, y, \emptyset)]\phi[\mathrm{child}] \\ &\beta[\lambda y.\ \mathrm{IF}\ (y = \text{`}\texttt{author}\text{'}, y, \emptyset)]\phi[\mathrm{child}]x)\ , \\ &(\beta[\lambda y.\ \mathrm{IF}\ (y = \text{`}\texttt{title}\text{'}, y, \emptyset)]\phi[\mathrm{child}]x), \emptyset)] \\ &\beta[\lambda y.\ \mathrm{IF}\ (y = \text{`}\texttt{book}\text{'}, y, \emptyset)]\phi[\mathrm{child}] \\ &\beta[\lambda y.\ \mathrm{IF}\ (y = \text{`}\texttt{books}\text{'}, y, \emptyset)]\phi[\mathrm{child}]\delta[\text{`}\texttt{bib.xml}\text{'}] \end{aligned}$$

により表される．データベース中の最上位要素 “bib.xml” が図 1 に示す文書構造を持つとき，図

(注2)：差分導出の結果得られるシーケンスシグネチャは通常の演算の結果得られるシーケンスシグネチャと同一であるため省略する．

表 2　XQuery-Core 演算子に対する差分導出関数
Table 2　Differentiation functions for the XQuery-Core operators

$(E_1, \Delta E_1) \overset{o}{\uplus}^{\Delta} (E_2, \Delta E_2) = (\Delta S, \Delta T, \Delta L)$
$\Delta T = \pi_{\mathrm{T}} \Delta E_1 \uplus \pi_{\mathrm{T}} \Delta E_2$
$\Delta L = \mathrm{map}(\lambda x\,.\,(\pi_{\mathrm{ID}} x, \text{`0:'}\ \pi_{\mathrm{O}} x))\pi_{\mathrm{L}} \Delta E_1 \uplus \mathrm{map}(\lambda x\,.\,(\pi_{\mathrm{ID}} x, \text{`1:'}\ \pi_{\mathrm{O}} x))\pi_{\mathrm{L}} \Delta E_2$
$\beta^{\Delta}[(F, \Delta F)](E_2, \Delta E_2) = (\Delta S, \Delta T, \Delta L)$
$\Delta T = \mathrm{ext}(\lambda x\,.\,\pi_{\mathrm{T}} F^{\Delta}((\mathrm{Ssig}, \pi_{\mathrm{T}} E, \{\!\{x\}\!\}), (\mathrm{Ssig}, \pi_{\mathrm{T}} \Delta E, \emptyset)))\pi_2 E \uplus \mathrm{ext}(\lambda x\,.\,\pi_{\mathrm{T}} \Delta F(\mathrm{Ssig}, \pi_{\mathrm{T}} E \uplus \pi_{\mathrm{T}} \Delta E, \{\!\{x\}\!\}))\pi_{\mathrm{L}} E$
$\quad \uplus \mathrm{ext}(\lambda x\,.\,\pi_{\mathrm{T}} \Delta F(\mathrm{Ssig}, \pi_{\mathrm{T}} E \uplus \pi_{\mathrm{T}} \Delta E, \{\!\{x\}\!\}))\pi_{\mathrm{L}} \Delta E \uplus \mathrm{ext}(\lambda x\,.\,\pi_{\mathrm{T}} F(\mathrm{Ssig}, \pi_{\mathrm{T}} E \uplus \pi_{\mathrm{T}} \Delta E, \{\!\{x\}\!\}))\pi_{\mathrm{L}} \Delta E$
$\Delta L = \mathrm{ext}(\lambda x\,.\,\mathrm{map}(\lambda y\,.\,(\pi_{\mathrm{ID}} y\ ,\ \pi_{\mathrm{O}} x\ \text{`:'}\ \pi_{\mathrm{O}} y))\pi_{\mathrm{L}} F^{\Delta}((\mathrm{Ssig}, \pi_{\mathrm{T}} E, \{\!\{x\}\!\}), (\mathrm{Ssig}, \pi_{\mathrm{T}} \Delta E, \emptyset)))\pi_{\mathrm{L}} E$
$\quad \uplus \mathrm{ext}(\lambda x\,.\,\mathrm{map}(\lambda y\,.\,(\pi_{\mathrm{ID}} y\ ,\ \pi_{\mathrm{O}} x\ \text{`:'}\ \pi_{\mathrm{O}} y))\pi_{\mathrm{L}} \Delta F(\mathrm{Ssig}, \pi_{\mathrm{T}} E \uplus \pi_{\mathrm{T}} \Delta E, \{\!\{x\}\!\}))\pi_{\mathrm{L}} E$
$\quad \uplus \mathrm{ext}(\lambda x\,.\,\mathrm{map}(\lambda y\,.\,(\pi_{\mathrm{ID}} y\ ,\ \pi_{\mathrm{O}} x\ \text{`:'}\ \pi_{\mathrm{O}} y))\pi_{\mathrm{L}} \Delta F(\mathrm{Ssig}, \pi_{\mathrm{T}} E \uplus \pi_{\mathrm{T}} \Delta E, \{\!\{x\}\!\}))\pi_{\mathrm{L}} \Delta E$
$\quad \uplus \mathrm{ext}(\lambda x\,.\,\mathrm{map}(\lambda y\,.\,(\pi_{\mathrm{ID}} y\ ,\ \pi_{\mathrm{O}} x\ \text{`:'}\ \pi_{\mathrm{O}} y))\pi_{\mathrm{L}} F(\mathrm{Ssig}, \pi_{\mathrm{T}} E \uplus \pi_{\mathrm{T}} \Delta E, \{\!\{x\}\!\}))\pi_{\mathrm{L}} \Delta E$
$\rho^{\Delta}[F](E, \Delta E) = (\Delta S\ ,\ \Delta T\ ,\ \Delta L)$
$\Delta T = \pi_{\mathrm{T}} \Delta E$
$\Delta L = \mathrm{ext}(\lambda x\,.\,\mathrm{ext}(\lambda y\,.\,\mathrm{ext}(\lambda z.\ \mathrm{if}\ (\pi_{\mathrm{ID}} y = \pi_{\mathrm{ID}} z)\ \mathrm{then}\ \{\!\{(\pi_{\mathrm{ID}} x, \mathrm{enc}(\pi_{\mathrm{V}} y))\}\!\}\ \mathrm{else}\ \emptyset$
$\qquad)\pi_{\mathrm{L}} F^{\Delta}((\mathrm{Ssig}, \pi_{\mathrm{T}} E, \{\!\{x\}\!\}), (\mathrm{Ssig}, \pi_{\mathrm{T}} \Delta E, \emptyset)))\pi_{\mathrm{T}} F(\mathrm{Ssig}, \pi_{\mathrm{T}} E, \{\!\{x\}\!\}))\pi_{\mathrm{L}} E$
$\quad \uplus \mathrm{ext}(\lambda x\,.\,\mathrm{ext}(\lambda y\,.\,\mathrm{ext}(\lambda z.\ \mathrm{if}\ (\pi_{\mathrm{ID}} y = \pi_{\mathrm{ID}} z)\ \mathrm{then}\ \{\!\{(\pi_{\mathrm{ID}} x, \mathrm{enc}(\pi_{\mathrm{V}} y))\}\!\}\ \mathrm{else}\ \emptyset$
$\qquad)\pi_{\mathrm{L}} F(\mathrm{Ssig}, \pi_{\mathrm{T}} E \uplus \pi_{\mathrm{T}} \Delta E, \{\!\{x\}\!\}))\pi_{\mathrm{T}} F^{\Delta}((\mathrm{Ssig}, \pi_{\mathrm{T}} E, \{\!\{x\}\!\}), (\mathrm{Ssig}, \pi_{\mathrm{T}} \Delta E, \emptyset)))\pi_{\mathrm{L}} E$
$\quad \uplus \mathrm{ext}(\lambda x\,.\,\mathrm{ext}(\lambda y\,.\,\mathrm{ext}(\lambda z.\ \mathrm{if}\ (\pi_{\mathrm{ID}} y = \pi_{\mathrm{ID}} z)\ \mathrm{then}\ \{\!\{(\pi_{\mathrm{ID}} x, \mathrm{enc}(\pi_{\mathrm{V}} y))\}\!\}\ \mathrm{else}\ \emptyset$
$\qquad)\pi_{\mathrm{L}} F(\mathrm{Ssig}, \pi_{\mathrm{T}} E \uplus \pi_{\mathrm{T}} \Delta E, \{\!\{x\}\!\}))\pi_{\mathrm{T}} F(\mathrm{Ssig}, \pi_{\mathrm{T}} E \uplus \pi_{\mathrm{T}} \Delta E, \{\!\{x\}\!\}))\pi_{\mathrm{L}} \Delta E$
$\phi^{\Delta}[a](E, \Delta E) = (\Delta S\ ,\ \Delta T\ ,\ \Delta L)$
$\Delta T = \pi_{\mathrm{T}} \Delta E$
$\Delta L = \mathrm{ext}(\lambda x\,.\,\mathrm{ext}(\lambda y.\ \mathrm{if}\ a(\pi_{\mathrm{ID}} y, \pi_{\mathrm{ID}} x)\ \mathrm{then}\ \{\!\{(\pi_{\mathrm{ID}} y, \pi_{\mathrm{O}} x\ \text{`:'}\ \pi_{\mathrm{ID}} y)\}\!\}\ \mathrm{else}\ \emptyset)\pi_{\mathrm{T}} \Delta E)\pi_{\mathrm{L}} E$
$\quad \uplus \mathrm{ext}(\lambda x\,.\,\mathrm{ext}(\lambda y.\ \mathrm{if}\ a(\pi_{\mathrm{ID}} y, \pi_{\mathrm{ID}} x)\ \mathrm{then}\ \{\!\{(\pi_{\mathrm{ID}} y, \pi_{\mathrm{O}} x\ \text{`:'}\ \pi_{\mathrm{ID}} y)\}\!\}\ \mathrm{else}\ \emptyset)(\pi_{\mathrm{T}} E \uplus \pi_{\mathrm{T}} \Delta E))\pi_{\mathrm{L}} \Delta E$
$\nu^{\Delta}[n](E, \Delta E) = (\Delta S\ ,\ \Delta T\ ,\ \Delta L)$
$\Delta T = \mathrm{ext}(\lambda x\,.\,\mathrm{ext}(\lambda y.\ \mathrm{if}\ \mathrm{match}(\text{`\^{}'}\ \pi_{\mathrm{ID}} x\ \text{`/'}, \pi_{\mathrm{ID}} y)\ \mathrm{then}\ \{\!\{(\mathrm{replace}(\pi_{\mathrm{ID}} x, \mathrm{enc}(\text{`}\boldsymbol{\nu}\text{['}\ \text{“}n\text{”}\ \text{`]'}\ \pi_{\mathrm{S}} E)\ \text{`/'}\ \mathrm{enc}(\pi_{\mathrm{O}} x), \pi_{\mathrm{ID}} y), \pi_{\mathrm{V}} y)\}\!\}\ \mathrm{else}\ \emptyset$
$\qquad)\pi_{\mathrm{T}} \Delta E)\pi_{\mathrm{L}} E$
$\quad \uplus \mathrm{ext}(\lambda x\,.\,\mathrm{ext}(\lambda y.\ \mathrm{if}\ \mathrm{match}(\text{`\^{}'}\ \pi_{\mathrm{ID}} x\ \text{`/'}, \pi_{\mathrm{ID}} y)\ \mathrm{then}\ \{\!\{(\mathrm{replace}(\pi_{\mathrm{ID}} x, \mathrm{enc}(\text{`}\boldsymbol{\nu}\text{['}\ \text{“}n\text{”}\ \text{`]'}\ \pi_{\mathrm{S}} E)\ \text{`/'}\ \mathrm{enc}(\pi_{\mathrm{O}} x), \pi_{\mathrm{ID}} y), \pi_{\mathrm{V}} y)\}\!\}\ \mathrm{else}\ \emptyset$
$\qquad)(\pi_{\mathrm{T}} E \uplus \pi_{\mathrm{T}} \Delta E))\pi_{\mathrm{L}} \Delta E$
$\Delta L = \emptyset$
$\mathrm{IF}^{\Delta}\ ((E_1\ ,\ \Delta E_1)\ ,\ (E_2\ ,\ \Delta E_2)\ ,\ (E_3\ ,\ \Delta E_3)) = (\Delta S\ ,\ \Delta T\ ,\ \Delta L)$
$\Delta T = \mathrm{if}\ \pi_{\mathrm{L}} E_1\ \mathrm{then}\ (\mathrm{if}\ (\pi_{\mathrm{L}} E_1 \uplus \pi_{\mathrm{L}} \Delta E_1)\ \mathrm{then}\ (\pi_{\mathrm{T}} E_2)\ \mathrm{else}\ (\pi_{\mathrm{T}} E_3 \uplus \pi_{\mathrm{T}} \Delta E_3 \uplus \dot{-}\pi_{\mathrm{T}} E_2))$
$\qquad \mathrm{else}\ (\mathrm{if}\ (\pi_{\mathrm{L}} E_1 \uplus \pi_{\mathrm{L}} \Delta E_1)\ \mathrm{then}\ (\pi_{\mathrm{T}} E_2 \uplus \pi_{\mathrm{T}} \Delta E_2 \uplus \dot{-}\pi_{\mathrm{T}} E_3)\ \mathrm{else}\ (\pi_{\mathrm{T}} E_3))$
$\Delta L = \mathrm{if}\ \pi_{\mathrm{L}} E_1\ \mathrm{then}\ (\mathrm{if}\ (\pi_{\mathrm{L}} E_1 \uplus \pi_{\mathrm{L}} \Delta E_1)\ \mathrm{then}\ (\pi_{\mathrm{L}} E_2)\ \mathrm{else}\ (\pi_{\mathrm{L}} E_3 \uplus \pi_{\mathrm{L}} \Delta E_3 \uplus \dot{-}\pi_{\mathrm{L}} E_2))$
$\qquad \mathrm{else}\ (\mathrm{if}\ (\pi_{\mathrm{L}} E_1 \uplus \pi_{\mathrm{L}} \Delta E_1)\ \mathrm{then}\ (\pi_{\mathrm{L}} E_2 \uplus \pi_{\mathrm{L}} \Delta E_2 \uplus \dot{-}\pi_{\mathrm{L}} E_3)\ \mathrm{else}\ (\pi_{\mathrm{L}} E_3))$

中の破線で示されるデータ挿入に伴う差分導出関数 E^{Δ} の値は以下のように評価される．

$$
\begin{aligned}
E^{\Delta} = (&\text{“}E\text{”}\ ,\ \{\!\{(\text{“}E\text{”}\text{`}\texttt{/b\#1;b\#0;a\#1;b\#0;a\#0;n\#1;}\text{'}\\
&\text{`}\texttt{b\#0;a\#0;n\#0;z}\text{'}, \text{`}\texttt{author}\text{'})\ ,\\
&(\text{“}E\text{”}\text{`}\texttt{/b\#1;b\#0;a\#1;b\#0;a\#0;n\#1;}\text{'}\\
&\text{`}\texttt{b\#0;a\#0;n\#0;z/d}\text{'}, \text{`}\texttt{notoya}\text{'})\}\!\}\ ,\ \emptyset)
\end{aligned}
$$

4. 差分更新の一括実行

一般に実体化ビューはビューを利用した問い合わせ処理のために利用されるため，実体化ビューの内容は問い合わせによって参照されない限り更新を受ける必要は無い．そのため，実体化ビューの更新手法には，データベースの状態変化の発生時点で状態変化に対応した実体化ビュー更新操作を実行する即時更新方式

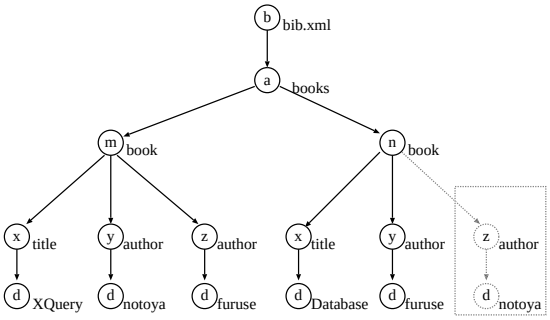

図 1　XML 文書構造
Fig. 1　An XML document structure

と，実体化ビューの内容に対する参照要求の発生時点で実体化ビュー更新操作を実行する一括実行方式が存在する．実体化ビュー更新の一括実行方式においては，個々のデータベース状態変化に対応する更新操作を繰返し実行するかわりに，前回実体化ビュー更新を行った時点以降のデータベース状態変化全体に相当する単一の更新操作を行うことにより，更新操作に必要とされる計算コストの削減が可能であることが知られている [5].

本研究の手法では，差分適用関数は多重集合の加法的合併を用いて定義されているため差分適用関数は結合的である．

$$\begin{aligned} E_{(s'')} &= (E_{(s)} \Leftarrow \Delta E_{(s,s')}) \Leftarrow \Delta E_{(s',s'')} \\ &= E_{(s)} \Leftarrow (\Delta E_{(s,s')} \Leftarrow \Delta E_{(s',s'')}) \end{aligned}$$

これを利用して複数の差分値を単一の差分値に結合することにより，差分更新の一括実行が適用可能である．

また，差分更新の一括実行の効率的な運用のためには，更新後のデータベースの状態から更新を受ける前のデータベースの状態を求める式が小さなコストで評価可能であることが重要である．本研究の手法においては，更新前状態は式

$$\delta[r]_{(s)} = \delta[r]_{(s')} \Leftarrow (`\boldsymbol{\delta}[\text{'} \; r \; \text{`}]', (\dot{-}\blacktriangle T) \uplus \blacktriangledown T, \emptyset)$$

によって容易に導出可能であり，この点からも差分更新の一括実行に適する方式であると考えられる．

5. ま と め

本稿では，XQuery-Core のノードシーケンス演算に対し，多重集合代数を利用した実現を与え，それを利用した XQuery ビューに対する差分導出関数，差分適用関数の構成を示した．本研究では XML データベースへの問い合わせ代数として XQuery Formal Semantics を利用し，その実現を与えるための理論的基盤として多重集合代数および FlexKey エンコーディングを利用した．このように既存の手法を組み合わせて利用しているため，本手法は既存の XML データベースシステムや関係データベースシステムを利用した実体化ビュー管理機構の構築に有用であると考えられる．

文　献

[1] A. Gupta and I. S. Mumick: “Maintenance of materialized views: Problems, techniques and applications”, IEEE Quarterly Bulletin on Data Engineering; Special Issue on Materialized Views and Data Warehousing, **18**, 2, pp. 3–18 (1995).
[2] W3C: “XQuery 1.0: An XML Query Language”, W3C working draft 12 november 2003 edition (2003).
[3] K. Dimitrova, M. El-Sayed and E. A. Rundensteiner: “Order-sensitive view maintenance of materialized XQuery views”, In Proc. Lecture Notes in Computer Science 2813, Conceptual Modeling - ER 2003, 22nd International Conference on Conceptual Modeling, Springer (2003).
[4] W3C: “XQuery 1.0 and XPath 2.0 Formal Semantics”, W3C working draft 20 february 2004 edition (2003).
[5] L. S. Colby, T. Griffin, L. Libkin, I. S. Mumick and H. Trickey: “Algorithms for deferred view maintenance”, Proc. of ACM SIGMOD International Conference on Management of Data, ACM Press, pp. 469–480 (1996).
[6] S. Grumbach and T. Milo: “Towards tractable algebras for bags”, In Proc. 12th ACM Symp. on Principles of Database Systems, pp. 49–58 (1993).
[7] D. Suciu: “Bounded fixpoints for complex objects”, Theoretical Computer Science, **176**, 1–2, pp. 283–328 (1997).
[8] P. L. Wadler: “Comprehending monads”, Proceedings of the 1990 ACM Conference on LISP and Functional Programming, Nice, New York, NY, ACM, pp. 61–78 (1990).
[9] M. El-Sayed, K. Dimitrova and E. A. Rundensteiner: “Efficiently supporting order in XML query processing”, In Proc. WIDM’03 (2003).
[10] W3C: “XQuery 1.0 and XPath 2.0 Functions and Operators”, W3C working draft 12 november 2003 edition (2003).