グラフに基づくデータベースに対する集合指向の統合演算

宝珍輝尚[†,*]

データがグラフ（データ表現グラフ）で表現されているデータベースに対する統合演算を提案する．提案する統合演算は，2つのデータ表現グラフをマージする集合指向の演算である．本演算は，あらかじめすべてのデータを1連結グラフで表現しにくい場合に，とりあえずn個の連結グラフとしてデータを格納し，検索時に動的に点を統合して操作する場合や，既存の点をマージしてあらかじめ設定された構造と異なる構造のグラフとしてデータを操作する場合に有効である．本論文では，まず，前提とするグラフに基づくデータモデル，ならびに，問合せ方式について述べる．次に，従来提案されているグラフに対する集合指向の演算について述べ，これらの演算ではアドホックに2つのデータ表現グラフをマージしようとすると手続き的になり使用に耐えないという問題点を示す．そして，この問題点を解決する統合演算を提案する．統合演算として，データ表現グラフの要素の一意性に基づく関係統合，制約統合，外統合を提案し，要素の値に基づいて統合を行うための複写指定と融合指定を提案する．最後に，提案した演算の宣言性を評価し，アドホック問合せにおいて有効であることを示す．

Set-Oriented Unite Operations on Graph-based Databases

TERUHISA HOCHIN[†,*]

Unite operations on graph-based databases are proposed. Data are represented with labeled directed graphs called *data representing graphs* in a graph-based database. The proposed unite operations are set-oriented ones, and merge two data representing graphs into a single one. The proposed operations are valuable for manipulating more than one data representing graph as a single data representing graph and for manipulating a data representing graph as the one having different structure from the original one by dynamically merging them. These operations are Relational Unite, Restrictive Unite, and Outer Unite. These are based on the identities of graph elements. Copy and Merge are the options to these unite operations in order to manipulate data representing graphs according to the values of their elements. First, assumptions on the data model and the querying method are mentioned. Second, set-oriented operations on graph-based databases are surveyed. After the problem in merging data representing graphs is mentioned, the unite operations are proposed in order to solve this problem. Lastly, the proposed operations are evaluated. These are declarative, and effective in making ad-hoc queries.

1. はじめに

データベース技術にはグラフを利用するものが数多くあり，スキーマの視覚的表現もその1つである[1),2)]．さらに進んで，グラフを直接利用するデータモデルの提案もなされている[3)~12)]．データがグラフで表現されているデータベースに対しては，グラフ中の経路を巡航したり，巡航経路を指定して問合せが行われる．これらの方法では，検索時に問合せ文を作成して，すなわちアドホックに，大量のデータから多くのデータを求めようとすると，いちいち巡航しなければならず手間がかかる．したがって，アドホックな問合せにおいては，リレーショナルモデルの関係演算のような集合指向の演算が必要となる．グラフに基づくデータモデルにおいても，グラフに存在するパターンを利用した集合指向の演算が提案されている[3),9)]．

グラフを扱おうとすると，2つのグラフの点をマージしたいという要求が出てくる．例えば，あらかじめすべてのデータを1連結グラフで表現しにくい場合に，とりあえず n 個の連結グラフとしてデータを格納し，検索時に動的に点をマージして操作したい場合や，既存の点をマージしてあらかじめ設定された構造と異なる構造のグラフとしてデータを操作したい場合である．しかしながら，2つのグラフの点をマージしようとすると，提案されている演算では処理を集合的に行うことができず，インスタンスごとにいちいち

† NTT 情報通信網研究所
NTT Network Information Systems Laboratories
* 現在 福井大学工学部情報工学科
Presently with Department of Information Science, Faculty of Engineering, Fukui University

マージしなければならない．このため，アドホックな問合せでは使用に耐えないという問題がある．

そこで本論文では，グラフに基づくデータベースに対するアドホックな問合せでの利用を目標として，データを表現するグラフ（データ表現グラフ）を動的にマージする演算の提案を行う．提案する演算は，一方のデータ表現グラフの点を他方のデータ表現グラフの点とする集合指向の演算である．本演算は，束縛変数を使用せずに多数の演算を逐次使用しないという意味で宣言的であり，アドホックな問合せに対して有効である．

2章では，本論文で前提とするグラフに基づくデータモデル，問合せ方式について述べる．3章では，従来から提案されている集合指向のグラフに対する演算について述べ，4章で，アドホック問合せにおいてグラフを連結して扱う場合の従来の演算の問題点を示す．そして，5章で，この問題を解決する統合演算を提案する．最後に，6章で評価を行い，提案する統合演算の有効性を示す．

2. 前　　提

2.1 データグラフ

本論文では，データモデルとして，データグラフ[10),11)]をさらに拡張して使用する．データグラフとは，データ構造からデータ項目の意味内容を取り去り，リンク構造のみに着目して得られる有向グラフである．A. L. Rosenberg はデータグラフを強連結有向グラフと定義し[10)]，坂部らは弱連結有向グラフに拡張している[11)]．ここでは，さらに，非連結で，枝の始終点が点集合となりえるグラフに拡張したデータグラフをもとにする．本論文でのデータグラフの定義を以下に示す．

データグラフはデータを表現するラベル付き有向グラフである．グラフ要素をデータベース内で一意に識別する一意識別子 d_{ID}，同種のグラフ要素集合をデータグラフ内で一意に識別する要素名 N，データ d によって，ラベルは3つ組 (d_{ID}, N, d) で表される．点および枝にはこのラベルが付与される．枝は始点集合と終点集合を持つ．

以降（図を含む）では，簡単化のため一意識別子 d_{ID} を省略し「$N: d$」で要素を表現する．データ d は組（データ型，値）で表現される．データ表現グラフとは，点集合 V と枝集合 E からなる順序組 (V, E) で表現される（連結または非連結な）グラフ $g(V, E)$ である．データグラフは，データ表現グラフを成分にもつグラフである．データグラフの集合がデータベースである．データベース内でデータグラフを識別するために名前を付加し，この名前をデータグラフ名という．同一データグラフ名のデータグラフの要素名のみをラベルとした（同一要素名が2以上ある場合は統合して得られた）グラフを，そのデータグラフのスキーマグラフという．リレーショナルモデルと対比すると，データグラフはリレーションに，データ表現グラフはタプルに対応する．

スキーマグラフの例を**図1**（a）に示す．この例には2つのスキーマグラフ（Residence と CityInf）がある．Residence には2つの枝（PF, CC）がある．枝 CC の始点は点 Address，終点は点 City と点 County である．始終点がおのおの1の場合は，簡略して CityInf の例のようにも表記する．図1（b）はデータグラフの例である．ここでは，データグラフを一点鎖

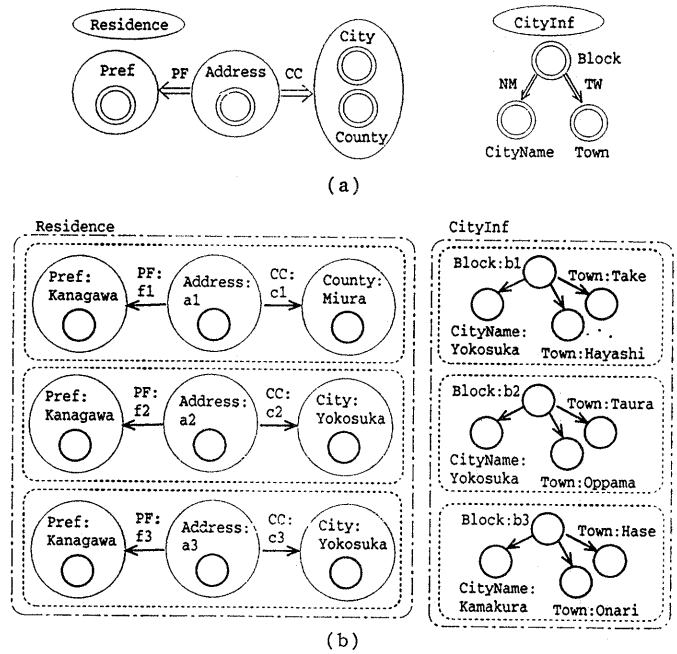

図1 スキーマグラフとデータグラフの例
（a）スキーマグラフ　（b）データグラフ

Fig. 1 Examples of data graphs and schema graphs.
（a）Schema Graphs　（b）Data Graphs

線で囲み，データ表現グラフを点線を囲んで表現している．例えば，データグラフ Residence には3つのデータ表現グラフが含まれていることを表している．点 Address : a1 は，点 Pref : Kanagawa, 点 County : Miura と連結している．なお，この例では枝の始終点は一点であるが，一般には枝の始終点は複数でありえる．スキーマグラフと同様に，データ表現グラフも簡単化のためにデータグラフ CityInf の例のようにも表記する．

2.2 問合せ方式

アドホック問合せでは，一時的なデータ定義（スキーマ）に従ってデータを操作したいことが良くある．リレーショナルモデルの関係演算の結合がその例である．結合結果のリレーションはデータ定義言語では直接定義されていない一時的なリレーションである．このように，問合せ中に限り一時的なスキーマを考慮する方法を本論文では前提とする．

3. 集合指向のグラフ演算

データベース中のデータを操作する演算は，一時に1インスタンスを扱うものと一時に多インスタンスを扱うものに分けて考えられる．ここでは，前者をインスタンス指向の演算，後者を集合指向の演算と呼ぶ．枝に沿って巡航を行う巡航演算はインスタンス指向であり，関係代数の射影演算は集合指向である．アドホック問合せでは，検索条件に合致したデータ集合を宣言的に求められる集合指向の演算が必要となる．

グラフに基づくデータモデルでも，集合指向の演算が提案されている[3),9)]．本論文では，一時的なデータ定義を利用して柔軟に問合せが可能な GOOD[3)] をもとに議論する．

GOOD では演算が変換言語（transformation language）として定義されている[3)]．この言語には，点の付加，点の削除，枝の付加，枝の削除，抽象化という5演算がある．これらの演算は，インスタンスを表現するグラフの一部であるパターンを探索し，そのパターンに対しておのおのの処理を行う．これらの演算は集合指向の演算である．GOOD ではこれらを問合せにおいて使用可能としている．抽象化演算は，同一ラベルを持つ1以上のグラフを1グラフとするという重複排除の演算である．抽象化以外の演算の意味は自明である．

さらに，GOOD では，使い勝手を良くするために付加を一般化したマクロ（一般化付加）が提案されている[4)]．一般化付加（generalized addition）とは，点の付加と枝の付加を同時に指定できる演算である．この演算では，まず点が付加され，その後に枝が付加される．

4. グラフの連結における問題点

4.1 グラフの連結

グラフを扱おうとすると問題になるのは複数のグラフを連結して利用する場合である．例えば，あらかじめすべてのデータを1連結グラフで表現しにくい場合に，とりあえず n 個の連結グラフとしてデータを格納し，検索時に動的に点をマージして操作したい場合や，新たな枝を追加してあらかじめ設定された構造と異なる構造のグラフとしてデータを操作したい場合である．ここでは，まず，問題点を明確化するためにグラフの連結方法を整理する．

P，Qを連結なデータ表現グラフとする．PとQから連結なデータ表現グラフを得る方法は以下の方法かまたはその組み合わせである．

（a） パスによるリンク付け　Pの点 A : a とQの点 B : b を1以上の枝と0以上の点を使用して結ぶ方法である．例えば図1（b）で，点 Address : a2 と点 Block : b1 を新たな枝 In : il で連結する場合である．

（b） 既存の枝の端点変更によるリンク付け　Pの枝 Z : z の端点をQの点 B : b に変更することによりPとQを結ぶ方法である．例えば図1（b）で，枝 CC : c2 の終点を点 Block : b1 に変更する場合である．

（c） 既存の点のマージによる統合　Pの点 A : a をQの点 B : b とすることでPとQを1データ表現グラフとする方法である．例えば図1（b）で，点 Block : b1 を枝 CC : c2 の終点ともする場合である．

方法（a）と方法（c）は，対象のデータグラフが多インスタンスの場合もそのまま適用できる．例えば，Residence の Address (a1, a2, a3) と CityInf の Block (b1, b2, b3) を，方法（a）でリンク付けの条件なしにリンク付けすると，a1 と b1 間，a1 と b2 間などを結ぶ9つのパスが生成される．また，方法（c）でマージすると，a1, a2, a3, b1, b2, b3 がマージされて1データ表現グラフとなる．

ところが，方法（b）では変更する端点をインスタンスごとに指定しなければならない．例えば，a1 を

b1, b2, b3 のすべてに変更することはできず，a1 を b1 にするというように変更前と変更後の点を必ず指定しなければならない．したがって，方法(b)はインスタンス指向の演算でなければ実現できない．ここでは，集合指向の演算を考えるため，以降では方法(b)は考えない．

4.2 アドホック問合せにおける問題点

方法(a)は3章で述べた GOOD の一般化付加で実現できる．しかし，方法(c)は従来のグラフ演算と1対1に対応せず，グラフ演算を組み合わせなければ実現できない．つまり，条件を満足するデータ表現グラフの組毎に枝に点を加えるという演算を行わなければならない．

ここで，簡単な問合せは宣言的に記述する方が手続き的に記述するよりも誤りの少ないことが，Welty と Stemple によって示されている[13]．ここで，宣言的とは束縛変数を使用せず多数の演算を逐次使用しないことであり，逆に手続き的とは，変数束縛を多数使用し多数の演算を逐次使用することである．アドホックな問合せはほとんどが簡単な問合せであると考えられるので，アドホックな問合せは宣言的に記述できる方が良いことになる．方法(a)は従来の演算により宣言的に記述できるのでアドホックな問合せでも使用に耐えうる．しかし，方法(c)は従来の演算のみでは手続き的にしか記述できず，アドホックな問合せには使用に耐えないという問題がある．

5. 統合演算

4.2 節で述べた問題を解決するために統合演算を提案する．まず，要素の一意性に基づく統合演算を定義する．次に，値に基づく統合演算を定義する．

5.1 要素の一意性に基づく統合演算

まず，データグラフの和を定義する．2つのデータグラフの和とは，2つのデータグラフの要素である全データ表現グラフを要素とするデータグラフである．

[定義1](和)

$$F \cup G = \{h \mid h \in F \vee h \in G\} \qquad \square$$

図1(b)の Residence を枝 CC とその始終点からなる要素とした Residence' と CityInf を枝 NM とその始終点からなる要素とした CityInf' の和を**図2**に示す．なお，以降の演算結果の例でもこの Residence' と CityInf' を用いる．

次の定義に先立ち，いくつかの記法を示す．演算 drg(F) は，データグラフF中の全データ表現グラフを1データ表現グラフとする．V_f^X は，データ表現グラフf中の点名Xの点の集合を表す．さらに，V_F^X は，データグラフF中の全データ表現グラフの点名Xの点の集合を表す．すなわち，$F=\{f_1, f_2, \cdots, f_n\}$ とすると，$V_F^X = V_{f_1}^X \cup V_{f_2}^X \cup \ldots \cup V_{f_n}^X$.

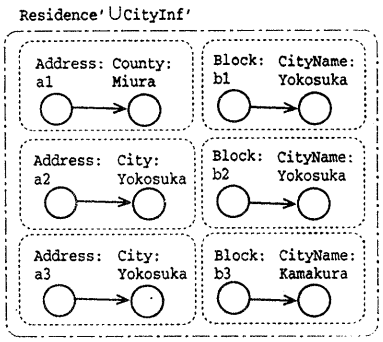

図 2 和の例
Fig. 2 An example of union.

次に，条件に関する記法について述べる．2つのデータ表現グラフf，gに対する条件を $\phi(f, g)$ とする．$\phi(f, g)$ は真偽値を返す．条件がない場合は常に真である．集合F中のすべての要素fに対して $\phi(f, g)$ が成立するとき $\phi(F, g)$ と書く．すなわち，$\phi(F, g) = \phi(f_1, g) \wedge \cdots \wedge \phi(f_m, g)$．同様に，$\phi(f, G) = \phi(f, g_1) \wedge \cdots \wedge \phi(f, g_n)$．さらに，$\phi(F, G) = \phi(f_1, g_1) \wedge \phi(f_1, g_2) \wedge \cdots \phi(f_m, g_n)$．また，集合F中のすべての要素fに対して $\phi(f, g)$ が成立しないとき $\bar{\phi}(F, g)$ と書く．すなわち，$\bar{\phi}(F, g) = \neg\phi(f_1, g) \wedge \cdots \wedge \neg\phi(f_m, g)$.

これらを用いて統合を定義する．統合は，2つのデータグラフFとGのデータ表現グラフの部分集合の和をデータ表現グラフとする．評価後変化しないデータ表現グラフを統合結果に含めるか否かによって，関係統合，制約統合，外統合に分けられる．関係統合は評価後変化しないデータ表現グラフを全く残さない．制約統合は評価後変化しないデータ表現グラフをすべて残し，外統合は，評価後変化しないデータ表現グラフの一方を残す．

[定義2](統合)

関係統合 $F[X1 \ll Y1, \cdots, Xn \ll Y_n[\phi]]G$
$= \{h \mid h = drg(F' \cup G'), V_h^{X1} = V_h^{Y1} = V_{F'}^{X1} \cup V_{G'}^{Y1}, \ldots, V_h^{Xn} = V_h^{Yn} = V_{F'}^{Xn} \cup V_{G'}^{Yn}, \phi(F', G'), \neg\phi(f, G'), \neg\phi(F', g), F' \subseteq F, G' \subseteq G, f \in F, f \notin F', g \in G, g \notin G'\}$.

制約統合 $F[X1 \ll Y1, \ldots, Xn \ll Yn(\phi)]G$
$= F[X1 \ll Y1, \ldots, Xn \ll Yn[\phi]]G$

$\cup \{f \mid \overline{\phi}(f, G), f \in F\} \cup \{g \mid \overline{\phi}(F, g), g \in G\}$.

外統合 $F[X1 \ll Y1, \ldots, Xn \ll Yn(\phi)]]G$

$= F[X1 \ll Y1, \ldots, Xn \ll Yn[\phi]]G \cup \{f \mid \overline{\phi}(f, G), f \in F\}$.

ここで，$\ll$の前後の $X1$，$Y1$ 等を統合点と呼ぶ．

□

定義2の内容は次のとおりである．関係統合では，以降の条件 ϕ を満足する F' と G' の和をとって得られるデータ表現グラフの集合をデータグラフとする．また，そのデータ表現グラフ h の点 Xk と点 Yk はともに F′ の点 Xk と G′ の点 Yk の和である．F′ と G′ は，$\phi(f, G')$ を満たすFのすべての要素を F′ とし，$\phi(F', g)$ を満たすGのすべての要素を G′ とするように選ぶ．制約統合や外統合では，関係統合の結果と条件 ϕ を満足しないデータ表現グラフの和をとる．

Residence' と CityInf' を，関係統合，制約統合，外統合した例を，おのおの，**図3**の(a)，(b)，(c)に示す．外統合では，一番目に指定したデータグラフに属するデータ表現グラフが残る．

次に，統合演算の特殊な例をいくつか挙げる．まず，統合点の指定がない場合である．例として，Residence' と CityInf' を City==CityName という条件のみで制約統合した結果を**図4**(a)に示す．条件を満足するすべてのデータ表現グラフが1データ表現グラフとなる．

次に，統合の条件がない場合である．例えば，Residence' と CityInf' を単純に City と Block で関係統合した場合である．この結果を図4(b)に示す．統合の条件がないので全データ表現グラフが1データ表現グラフとなる．また，統合点は同一点集合に属することになる．

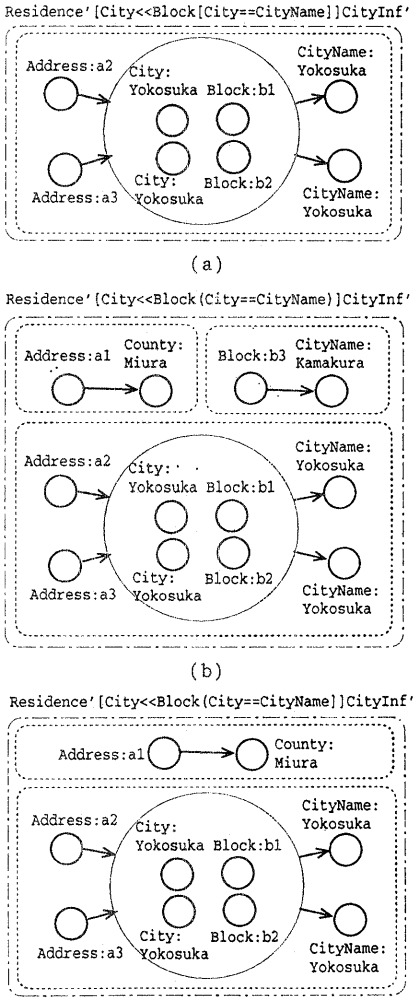

図3 統合演算の例

(a) 関係統合　(b) 制約統合　(c) 外統合

Fig. 3 Examples of unite operations.

(a) Relational Unite　(b) Restrictive Unite

(c) Outer Unite

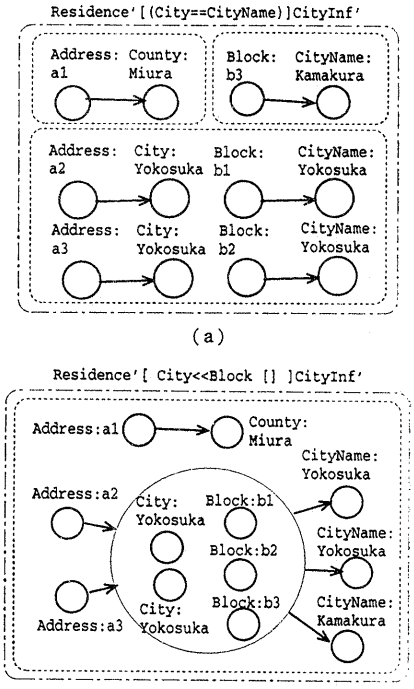

図4 統合の特殊な例

Fig. 4 Special examples of unite.

5.2 値に基づく統合演算

5.1 節で示した統合演算処理は要素の一意性を考慮している．すなわち，データ表現グラフの一意性は不変であり，統合される要素は共有される．ここでは，要素の一意性を考慮しない，値に基づく統合について述べる．これらは，一意性を考慮した演算のバリエーションと考えられる．

5.2.1 複写指定

複写指定された統合では，データ表現グラフの複写が行われ，要素は共有されない．複写されたデータ表現グラフの要素には新たな一意識別子が割り当てられる．ここでは，関係統合の複写指定を定義する．

[定義3]（複写指定）

片側複写 $F\{X1 \ll Y1[\phi]]G$

$= \{h \mid h = drg(copy(F') \cup G'), V_h^{X1} = V_h^{Y1} = V_{F'}^{X1} \cup V_{G'}^{Y1}, \phi(F', G'), \neg\phi(f, G'), \neg\phi(F', g), F' \subseteq F, G' \subseteq G, f \in F, f \notin F', g \in G, g \notin G'\}$.

両側複写 $F\{X1 \ll Y1[\phi]\}G$

$= \{h \mid h = drg(copy(F') \cup copy(G')), V_h^{X1} = V_h^{Y1} = V_{F'}^{X1} \cup V_{G'}^{Y1}, \phi(F', G'), \neg\phi(f, G'), \neg\phi(F', g), F' \subseteq F, G' \subseteq G, f \in F, f \notin F', g \in G, g \notin G'\}$. □

関係統合の定義との違いは，drg( )の中で copy( ) を用いることのみである．

両側複写の例を**図5**に示す．Address：a2 が2つあることからわかるように，データ表現グラフが複写される．

制約統合，外統合に対する複写指定も，関係統合との対応により容易に定義が導出できる．

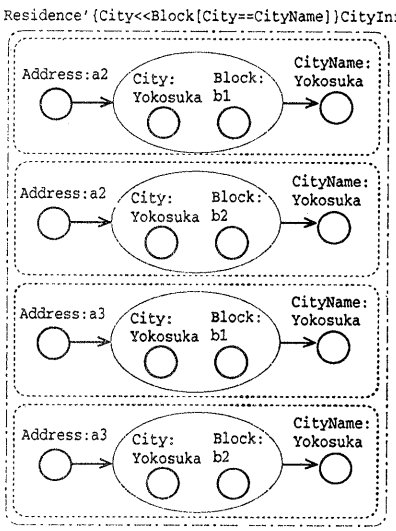

図5 複写指定の統合演算（両側複写）
Fig. 5 Unite operations with two-side-copy.

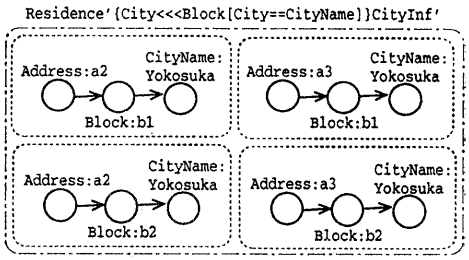

図6 融合指定の統合演算
Fig. 6 A unite operation with merge.

5.2.2 融合指定

値のみを考慮すると，統合結果の枝の始点（終点）を集合とするのではなく，一点とする場合も考えなければならない．例えば，通常のグラフ処理における融合[14]がこれに相当する．これを融合指定を行うことにより可能とする．融合指定の場合，複数の点を枝の始点（終点）とすることはできないので，残る点を指定しなければならない．また，残る点と指定された点は一点でなければならない．ここでは，両側複写指定された関係統合における融合指定を定義する．

[定義4]（融合指定）

$F\{X1 \lll Y1[\phi]\}G = \{h \mid h = drg(copy(F') \cup copy(G')), V_h^{X1} = V_{G'}^{Y1}, V_h^{Y1} = V_{G'}^{Y1}, \phi(F', G'), \neg\phi(f, G'), \neg\phi(F', g), F' \subseteq F, G' \subseteq G, f \in F, f \notin F', g \in G, g \notin G'\}$. □

定義からわかるように，新たなデータ表現グラフの点 X1 は点 Y1 で置き換えられる．

融合指定の統合演算の例を**図6**に示す．図6は，Residence' と CityInf' の両方を複写しながら関係統合し，Residence' の点 City を CityInf' の点 Block とした例である．

6. 評　価

提案した統合演算の，アドホックな問合せでの使用を念頭においた集合演算としての有効性を，問合せの宣言性により評価する．宣言性の評価には，Welty と Stemple により提案された手続き度[13]を使用する．

まず，手続き度について概説し，次に，統合演算を使用する場合としない場合の手続き度を比較する．

6.1 手続き度

Welty と Stemple によって提案された手続き度は言語のステップバイステップの度合を示すための測度である．手続き度 PM は下式によって求められる．

$$PM=\frac{N_v}{N_{vo}}+\frac{N_o}{N_{oo}} \quad (1)$$

ここで，N_v は変数束縛の数，N_{vo} は許される変数束縛順序の数，N_o は演算の数，N_{oo} は許される演算順序の数である．許される順序とは評価される文で示された順序または言語の構文上および意味上評価される文と同じとみなされる順序である．順序を変えた文と元の文の結果が同じであるからといって順序を変えた文が許される順序とは限らないことに注意しなければならない．例えば，以下の手続きを考えてみよう．

x=new node N;
y=select M;
copy value y to x;

この場合，束縛変数は x と y である．変数 x と y の順序を変えることはできないので N_{vo} は1である．3個の命令があるので演算数 (N_o) は3である．new node 文と select 文の順序は意味に影響しないので N_{oo} は2である．したがって，PM=2/1+3/2=3.5 である．

6.2 統合演算の手続き度

統合演算を使用する場合 (Case U) と使用しない場合 (Case C) の手続き度を考える．ここでは，基本的な制約統合について考える．その他（関係統合と外統合について，ならびに，複写指定・融合指定）については 6.3 節で言及する．

(1) 例による手続き度計算

ここでは，図1(b)のデータグラフから図3(b)のデータグラフを得ることを考えてみる．Case U では，2つの制約演算 (Residence から Residence', CityInf から CityInf') と統合演算を使用すれば実現できる (**図7**(a))．2つの制約演算の順序は変えられる．Residence' と CityInf' は一種の変数で，これらの順序は変えられない．したがって，PM=2/1+3/2=3.5 である．

```
(a)
  Residence' = Residence[Address,CC,City,County]
  CityInf'   = CityInf[Block,NM,CityName]
  Residence'[City<<Block[City==CityName]]CityInf'

(b)
  Residence' = Residence[Address,CC,City,County]
  CityInf'   = CityInf[Block,NM,CityName]
  for( f in Residence' )
    for( g in CityInf' )
      for( ef in f.CC )
        for( eg in g.NM )
          if ef.City == eg.CityName then
            union ef.City and eg.Block
```

図7 統合のための処理
(a) 統合演算使用　(b) 統合演算未使用
Fig. 7 Processings for unite.
(a) with unite operation
(b) without unite operation

一方，Case C では，図7(b)に示す処理を行わなければならない．変数は，Residence' と CityInf' 以外に，それらの中のデータ表現グラフを束縛する変数 (f, g) と City，Block に連結する枝 (CC, NM) を束縛する変数 (ef, eg) が必要である．これらの変数の順序は変えられない．演算は，上記の2つの制約演算以外に，点 City と点 Block の和をとる演算 (union) を用いる．枝 CC の終点集合と枝 NM の始点集合が，逐次，点 City と点 Block の和をとって得られる点集合となる．以上より，PM=6/1+3/(2*1)=7.5 である．

(2) 統合処理の手続き度

ここでは，統合に関する処理のみの手続き度を求める．一般に，Case U では1つの統合演算で束縛変数なしに操作できる．Case C の場合，統合に関与するデータ表現グラフを束縛する変数が必要である．また，定義上の要素に対して複数の要素が存在しうるので，要素を束縛する変数が必要となる．これらの変数の順序は変えられない．また，統合点の組ごとに1つの演算が必要で，これらの演算の順序は変えられる．

以上より，Case U，Case C の手続き度 (PM_u, PM_c) は，統合点の組の数を m とすると，下式で与えられる．

$$PM_u=\frac{0}{1}+\frac{1}{1}=1 \quad (2)$$

$$PM_c=\frac{2+2m}{1}+\frac{m}{m!}=2+2m+\frac{1}{(m-1)!} \quad (3)$$

(3) 全体の手続き度

ある処理を行うための演算列 $op_1, \cdots, op_n$ があり，統合処理の前に統合処理以外の処理が行われているとする．すなわち，ある k ($1 \leq k < n$) に対して，$op_1, \cdots, op_k$ は統合処理以外のための演算列，$op_{k+1}, \cdots, op_n$ は統合処理のための演算列である．ここでは，演算列全体 $op_1, \cdots, op_n$ の手続き度を考える．Case U，Case C の手続き度 (PM_{ut}, PM_{ct}) は，演算列 $op_1, \cdots, op_k$ の手続き度を $PM_p=n_v/n_{vo}+n_o/n_{oo}$ とすると，下式で与えられる．ただし，統合対象のデータグラフは統合処理以前の処理の中の変数で束縛されると仮定している．

$$PM_{ut}=\frac{n_v}{n_{vo}}+\frac{n_o+1}{n_{oo}}=PM_p+\frac{1}{n_{oo}} \quad (4)$$

$$PM_{ct}=\frac{n_v+2+2m}{n_{vo}}+\frac{n_o+m}{n_{oo}\cdot m!} \quad (5)$$

6.3 考 察

提案した統合演算を使用すると，式(2)からわかるように，手続き度を統合点の数に依存せず一定にできる．これは，アドホック問合せの記述において望ましい特性と考えられる．また，式(2)と式(3)からわかるように，統合点の数が多い場合には，統合演算の使用により手続き度を低くでき，有利である．これは，演算数ではなく変数束縛数の影響が大きい．

ここでは制約統合を考えた．関係統合や外統合では，条件を満足しないデータ表現グラフを結果としないようにするために順序の変えられない演算が必要であり，これらの PM_c は制約統合時の PM_c よりも高くなる．したがって，統合演算を使用する方が有利である．

また，ここでは要素の一意性に基づく統合演算について考えた．複写指定や融合指定をした場合も，上記の議論はそのままあてはまる．

7. おわりに

本論文では，グラフ表現されたデータの集まりであるデータベースに対する統合演算を提案した．提案した統合演算は，2つのデータ表現グラフをマージする集合指向の演算である．統合演算として，関係統合，制約統合，外統合を提案し，オプションとして複写指定と融合指定を提案した．この統合演算により，アドホック問合せにおいて，手続き的にしか記述できなかったグラフのマージ処理が宣言的に記述可能となった．提案した演算は，従来の演算のみを使用する場合と比較して問合せにおける手続き度を低くでき，アドホック問合せに適することを明らかにした．したがって，本演算は，あらかじめすべてのデータを1連結グラフで表現しにくい場合に，とりあえず n 個の連結グラフとしてデータを格納し，検索時に動的に点を統合して操作する場合や，既存の点をマージしてあらかじめ設定された構造と異なる構造のグラフとしてデータを操作する場合に有効である．

今後は，グラフの分離を行うグラフ演算の検討やグラフィカルな問合せ言語での実装が課題である．

謝辞 本論文執筆にあたり有益なコメントをいただいた福井大学工学部情報工学科都司達夫教授ならびに論文査読委員の皆様に感謝いたします．本研究を行うにあたり日頃から熱心に議論していただいた，井上潮主幹研究員をはじめとする NTT 情報通信網研究所の皆様に感謝いたします．

参考文献

1) Hull, R. and King, R.: Semantic Database Modeling: Survey, Applications, and Research Issues, *ACM Comput. Surv.*, Vol. 19, No. 3, pp. 201-260 (1987).
2) Peckham, J. and Maryanski, F.: Semantic Data Models, *Comput. Surv.*, Vol. 20, No. 3, pp. 153-189 (1988).
3) Gyssens, M., Paredaens, J. and Gucht, D. V.: A Graph-Oriented Object Database Model, *Proc. of 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 417-424 (1990).
4) Andries, M. and Paradaens, J.: Macro's for the GOOD-Transformation Language, Technical Report 91-36, University of Antwerp (UIA) (1991).
5) Kunii, H. S.: *Graph Data Model and Its Data Language*, Springer-Verlag (1990).
6) 関 義長，遠山元道，浦 昭二：グラフデータを扱うデータベース・システム（グラフデータ操作言語 GOLI），第38回情報処理学会全国大会論文集，4R-6, pp. 1060-1061 (1989).
7) Curz, I. F., Mendelzon, A. O. and Wood, P. T.: G^+: Recursive Queries without Recursion, *Proc. of 12th International Conference on Expert Database Systems*, pp. 355-368 (1988).
8) Consens, M. P. and Mendelzon, A. O.: GraphLog: a Visual Formalism for Real Life Recursion, *Proc. of 9th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp. 404-416 (1990).
9) Guo, M., Su, S. Y. W. and Lam, H.: An Association Algebra for Processing Object-Oriented Databases, *Proc. of International Conference on Data Engineering*, pp. 23-32 (1991).
10) Rosenberg, A. L.: Data Graphs and Addressing Schemes, *Journal of Computer and System Sciences*, Vol. 5, pp. 193-238 (1971).
11) 坂部俊樹，稲垣康善，福村晃夫：データグラフの代数的ならびにグラフ的一様性の無限階層構造，信学論 D, Vol. J60-D, No. 2, pp. 122-128 (1977).
12) 宝珍輝尚：拡張可能 DBMS: COMMON の格納構造と基本演算について，情報処理学会データベース・システム研究会報告，82-6 (1991).
13) Welty, C. and Stemple, D. W.: Human Factors Comparison of a Procedural and a Nonprocedural Query Language, *ACM Trans. Database Syst.*, Vol. 6, No. 4, pp. 626-649 (1981).
14) 田口克則，池田 満，田中栄一：節点の分離・融合操作に基づく木の距離について，信学論 D-I, Vol. J75-D-I, No. 3, pp. 191-195 (1992).

(平成5年2月8日受付)
(平成5年12月9日採録)

宝珍　輝尚（正会員）

昭和34年生．昭和57年名古屋工業大学電気工学科卒業．昭和59年同大大学院修士課程修了．同年，日本電信電話公社入社．NTT情報通信網研究所を経て，平成5年7月より福井大学工学部情報工学科助手，現在に至る．マルチメディアデータベース管理システム，拡張可能データベース管理システム，グラフィカルなデータベース問合せの研究に従事．電子情報通信学会，IEEE，ACM各会員．