データベース・システム 82－6
（１９９１．３．１１）

拡張可能ＤＢＭＳ：ＣＯＭＭＯＮの格納構造と基本演算について

宝珍輝尚
ＮＴＴ情報通信処理研究所

拡張可能ＤＢＭＳ：ＣＯＭＭＯＮで採用するデータを表現するグラフ（データグラフ）のための基本演算を提案し、基本演算の実現方法について述べる。まず、データグラフを多種の意味的なデータモデル実現のプラットフォームとしたＣＯＭＭＯＮの概要を述べる。次に、データグラフのための基本演算について述べる。ここでは、従来の巡航的な基本演算を示し、手続き的に使用せずにデータグラフを操作可能な基本演算（集合，複写，部分グラフ化，統合，順序化，集約化および再帰に関する演算）を提案する。そして、ＣＯＭＭＯＮで選択可能とする３種の格納構造に基づいた基本演算の実現方法について述べる。

Storage Structure and Primitive Operations for Extensible DBMS:COMMON

Teruhisa HOUCHIN
NTT Communications and Information Processing Laboratories
1-2356 Take, Yokosuka-Shi, Kanagawa 238-03, Japan

Primitive operations and their implementations for data representing graphs(data graphs) in the extensible DBMS: COMMON are suggested. First, the COMMON's features are described. COMMON works on data graphs according to support several semantic data models. Next, operations for data graphs are described. They are devided into 2 groups: procedual and non-procedual ones. Set, copy, subgraph, unite, ordering, aggregating, and recursive operations are suggested as non-procedual ones. And next, implementations of non-procedual ones are studied on COMMON's selective storage stoructures.

1．はじめに

ＤＢＭＳの拡張性を高める研究が盛んに行われている[1-11]。この拡張可能ＤＢＭＳの研究アプローチは大別してフル機能ＤＢＭＳアプローチとＤＢＭＳgeneratorアプローチに分けられる。フル機能ＤＢＭＳアプローチは、完全な機能を持つＤＢＭＳに対してＤＢＭＳ応用分野で要求される機能追加を容易とするアプローチであり[2,3]、ＤＢＭＳgeneratorアプローチは、様々な部品を組み合わせてＤＢＭＳ応用分野に応じたＤＢＭＳを構築するというアプローチである[4-11]。後者のアプローチには多種の意味的なデータモデル実現の研究もある[7]。筆者は後者のアプローチを採り、多種の意味的なデータモデル実現等を目指し、ＣＯＭＭＯＮと名付けたＤＢＭＳgeneratorを試作中である。ＣＯＭＭＯＮでは、多種の意味的なデータモデル実現のために論理的な格納構造としてグラフ（データグラフ）を採用し、グラフの物理的な格納構造の提案を行っている[11]。この格納構造はグラフの枝または点に着目したものであり、ＣＯＭＭＯＮではこれらの格納構造と従来の格納構造を選択可能とする。

ここで、ＣＯＭＭＯＮを使用して構築するＤＢＭＳでも関係型ＤＢＭＳのような高水準なデータベース操作言語サポートの要求が考えられるが、このためには関係代数のような基本演算が必要である。

そこで、本稿ではデータグラフのための巡航的な基本演算の明確化，手続き的に使用せずにデータグラフを操作可能な基本演算の提案を行い、さらに、ＣＯＭＭＯＮで選択可能とする３種の格納構造に基づく基本演算の実現方法について述べる。

以下、２．でＣＯＭＭＯＮの概要について述べ、３．でデータのグラフ表現と格納構造について述べる。次に、４．でデータグラフに対する基本演算について述べ、５．で基本演算の実現方法について述べる。最後に、６．で関連する研究との比較を行う。

2．拡張可能ＤＢＭＳ：ＣＯＭＭＯＮ

2．1　ＣＯＭＭＯＮの概要

2．1．1　背景

ＣＯＭＭＯＮでも他の拡張可能ＤＢＭＳ[2-9]と同様、様々なＤＢＭＳ応用分野に対するＤＢＭＳの迅速な構築を目指している。このために、①ＤＢＩ（ＤＢインプリメンタ）の負荷軽減，②多種の意味的なデータモデル実現可能化を図ることが重要であると考えている。

ＤＢＩはＤＢＭＳという大規模なソフトウェアの設計・構築・維持管理を行わなければならず、その負荷が膨大となる。この負荷は、ＤＢＭＳgeneratorではフル機能ＤＢＭＳを拡張した拡張可能ＤＢＭＳよりも大きなものとなる。従って、実用的なＤＢＭＳgeneratorにはＤＢＩのための様々な負荷軽減策が必要である。現在、ＣＯＭＭＯＮではこの負荷軽減のために、ＤＢＭＳ処理データのＤＢ化[9]，柔軟なモジュール・インタフェース[9]を導入している。これらの概要を2.1.2で述べる。

多種の意味的なデータモデルの実現を目指す目的は、ＤＢＭＳ応用分野で必要とされる独自のデータモデルを容易に実現するためである。独自のデータモデルとは、地図分野において線分と地点との接続関係をプリミティブなデータモデルの構成要素としている[12]といったものである。このために、現在ＣＯＭＭＯＮは、多種の意味的なデータモデルのプラットフォームとしてのデータのグラフ表現，選択可能な格納構造を導入している。これらについては３．で詳しく述べる。

２．１．２　特徴

（１）ＤＢＭＳ処理データのＤＢ化[9]

ＣＯＭＭＯＮは、ＤＢＭＳの処理に使用するデータ（ＤＢＭＳ処理データ）のデータベース化を実現するＤＢＭＳコアモジュール群とユーティリティから構成される（図１）。ＤＢＭＳ処理データとは、例えば、データベース操作言語の解析後に作成される構文木等のデータである。ＤＢＭＳコアモジュールでは、これらのＤＢＭＳ処理データの管理及び処理を行い、ＤＢＭＳ処理データのＤＢ化を実現する。

ＣＯＭＭＯＮを用いたＤＢＭＳの構築例を図１に示す。図１では、ＤＢＭＳ制御，構文解析，最適化，プラン実行というＤＢＭＳ実現モジュールをＤＢＩが作成している。ここで、ＤＢＭＳ実現モジュールとは、ＤＢＭＳコアモジュール群を用いてＤＢＭＳの処理を行うモジュールの総称である。

（２）柔軟なモジュール・インタフェース[9,10]

COMMONでは、DBMSを構成するモジュール間のインタフェースデータの授受を支援するインタフェースデータ管理部（DBI）を導入している。これにより、インタフェースデータ中の必要なデータ項目のみが選択的に操作でき、インタフェースデータを論理的に操作できる。

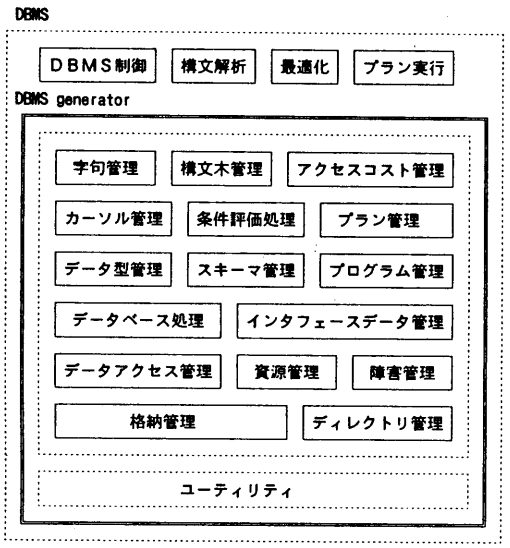


図1　DBMS構築システムを用いたDBMSの構成例

3．データのグラフ表現と格納構造

3．1　データグラフ

COMMONでは、多種の意味的なデータモデルのプラットフォームとして、データを表現するグラフ（データグラフ）を採用する。これは、グラフがデータ構造の中で最も一般的であり、さらに、様々なタイプ構成子およびタイプ間の関連を用いてデータの持つ意味を構造的に表現する意味的なデータモデル[13]の表現に適するためである。

3．1．1　定義

まず、グラフの構成要素の定義を行う前に、データの一意識別子とデータ型の定義を行う。

［定義1］（一意識別子）　関係R_{ID}が全順序であるような集合D_{ID}を考え、その要素d_{ID}を一意識別子といい、D_{ID}を一意識別子の定義域という。

［定義2］（データ型）　データ値を格納するデータフォーマットとこのデータフォーマット上のデータ値を処理する手続きの集合とデータに対する制約の集合をデータ型といい、データ型に付与された名前をデータ型名という。

［定義3］（データ）　組(DTN,dv)をデータという。ここで、DTNはデータ型名、dvはDTNの示すデータ型のフォーマット上に表現可能なデータである。

［定義4］（値の定義域）　n(n≧1)個のデータ型DT_iで表現可能な値の集合V_iの和集合$V(=V_1 \cup \cdots \cup V_n)$を値の定義域といい、その名前を定義域名という。

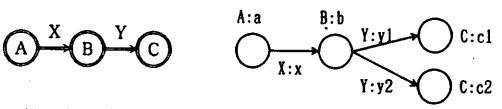
［定義5］（点，枝）　3つ組（d_{ID},DMN,d）を点といい、点を要素とする集合を点集合という。5つ組（d_{ID},DMN,d,v_i,v_t）を枝といい、枝を要素とする集合を枝集合という。ここで、d_{ID}は一意識別子、DMNは定義域名，dはデータ，v_iは枝の始点，v_tは枝の終点。

［定義6］（データグラフ）　点集合Vと枝集合Eからなる順序組（V，E）が連結グラフである場合、グラフg（V，E）をデータ表現グラフという。また、データ表現グラフを成分にもつ非連結グラフG（Ψ，Σ）をデータ群グラフという。データ表現グラフとデータ群グラフを総称してデータグラフという。

［定義7］（集合属性，単属性）　データ表現グラフ中に同一属性名の要素が2以上存在するとき、この属性を集合属性という。また、集合属性でない属性を集合属性と区別して単属性と呼ぶ。

データ表現グラフの定義と実体の例を図2に示す。図2（b）の属性名Cの点は複数であるので属性Cは集合属性である。

本稿で使用する表記を表1にまとめて示す。



(a) データ定義例　　(b) データ実体例

図2　データ表現グラフの例

3．1．2　等価性

（1）絶対等価性

［定義8］（絶対包含性）　データグラフPとQにおいて、Pの要素の一意識別子の集合がQの要素の一意識別子の集合を含むとき、QはPを絶対包含するといい、$P \subseteq_I Q$と表記する。

［定義９］（絶対等価性）　データグラフＰとＱにおいてＰ$\subseteq_I$ＱかつＱ$\subseteq_I$Ｐであるとき、２つのデータグラフＰ、Ｑは値等価であるといい、Ｐ#Ｑと表記する。

（２）全等価性

［定義１０］（全包含性）　グラフとしてデータグラフＰがＱのサブグラフであり対応する要素が値等価のとき、ＱはＰを全包含するといい、Ｐ$\subseteq_T$Ｑと表記する。

［定義１１］（全等価性）　データグラフＰとＱがグラフとして枝の向きを含めて同形であり、対応する要素が値等価のとき、ＰとＱは全等価であるといい、Ｐ$\overset{T}{=}$Ｑと表記する。

表１　表記

表記	意　味
v[P]	データグラフＰ中の点集合
e[P]	データグラフＰ中の枝集合
z[P:C]	データグラフＰの属性名Ｃの示す要素
id[w]	要素（点または枝）ｗの一意識別子
k[w]	要素（点または枝）ｗの持つ値
i[e]	枝ｅの始点
t[e]	枝ｅの終点
c[v]	点ｖの接続枝（点ｖを始点または終点に持つ枝の集合）
c[V]	点の集合Ｖに属する点を始点または終点に持つ枝の集合
g∈G	データ表現グラフｇはデータ群グラフＧの成分
g∈P	データグラフＰがデータ群グラフのときｇ∈Ｐ，データグラフＰがデータ表現グラフのときｇ＝Ｐ

３．２　選択可能な格納構造

３．２．１　３種の格納構造

多種の意味的なデータモデル実現の目的は、様々なＤＢＭＳ応用分野にＣＯＭＭＯＮを適用するためである。ここで、ＤＢＭＳの意識する格納構造はＤＢＭＳの性能を決定する大きな要因であるが、該格納構造は固定的であり、ＤＢＭＳの適用分野に最適であるとは限らない。ＣＯＭＭＯＮでは、条件式評価性能が条件式中の属性のデータ配置上の位置に依存する（位置依存）という特性に着目し、位置依存する格納構造と位置独立な格納構造[11]を選択可能とする。これにより、ＤＢＩの負荷微増で格納構造の柔軟化を図る。

意味的なデータモデルを実現したＤＢＭＳの格納構造[20,22]は位置依存する格納構造であり、操作の起点となる（グラフの）ソースノードを中心としたデータ構造である。位置独立な格納構造はグラフの連結性に着目したデータ構造であり、連結なデータ表現グラフを単位とし、該グラフの各要素を指すノードを設けて該グラフを管理するものである。

（１）ソースノード中心のデータ構造

ソースノード中心のデータ構造では、点に接続する枝を該点で管理する。データ検索時は、ソースノードから該ノードに接続している要素を順次検索してゆくことになる。この概要を 図３(a)に示す。属性Ｂの点の値(ｂ)は属性Ａの点（ソースノード）から枝Ｘを経由し、該点に到達して初めて得られる。

（２）点中心のデータ構造

点中心のデータ構造の構成要素は top-n，点実体，枝実体，ポインタリストである。top-nは 点の定義数，点ポインタエントリを格納する。ｉ番目の点ポインタエントリは点の定義番号ｉの点実体の数と点実体へのポインタを格納する。ただし、点実体数が２以上の場合はポインタリストを指し、ポインタリストが各点実体を指す。点実体と枝実体は従来のデータ構造と同じデータ構造で関係付ける。図２のグラフの本データ構造での表現を 図３(b)に示す。本データ構造は、従来のグラフのデータ構造に各点への直接アクセスを可能とする top-nを付加したものといえる。

（３）枝中心のデータ構造

枝中心のデータ構造の構成要素は、top-e，枝実体，点実体，ポインタリストである。top-eは 枝の定義数，枝ポインタエントリ，孤立点の数，孤立点へのポインタリストへのポインタを格納する。枝ポインタエントリは(２)の点ポインタエントリと同様である。枝実体は、枝の値、始点と終点へのポインタを格納する。点実体は点の値のみを格納する。図２のグラフの本データ構造での表現を 図３(c)に示す。

点，枝中心のデータ構造では、定義要素数を既知と仮定した。本仮定は、ＤＢＭＳの意識する格納構造として本データ構造を使用する場合には常に満足される。

３．２．３　３格納構造の特性

３格納構造の長所，短所[11]を概説する。

（１）ソースノード中心のデータ構造

本データ構造は、枝中心のデータ構造と比較して、

巡航検索を検索手段とする場合に有効である。

（２）点中心のデータ構造

点中心のデータ構造は、枝中心のデータ構造と従来のデータ構造の利点を合わせ持つので、条件検索のみでなく巡航検索をも必要とする場合に有効である。

しかし、本格納構造は(１)，(３)と比較して格納効率ならびにデータ更新性能は良くない。また、本データ構造の特徴である第一要素（top-n）は、要素の定義数に比べデータ実体数が少なくなると格納効率が劣化する。この問題は現在検討中である。

（３）枝中心のデータ構造

本データ構造は、条件式評価性能が位置依存せず、定義上の要素数により条件式評価性能を見積ることができる。また、属性のデータ表現グラフ上の位置によっては(１)よりも高速である。

格納効率は(１)と同等である。ただし、top-eについてはtop-nと同様の問題がある。

４．グラフデータのための基本演算

ここでは、3.1で述べたデータグラフを非手続き的に操作可能な演算を提案する。これに先立ち、データグラフに対する既存の巡航的な演算を明確化する。

４．１　巡航的な基本演算

（１）現在指示子

［定義１２］（現在指示子）　データグラフ中の着目している要素を示す指示子を現在指示子という。

（２）点の移動

［定義１３］（点の移動）　点の移動とは、ある属性の点集合Ｖと Ｖ中の一点v_0とＶ中の要素の値の集合を半順序集合とする関係Ｒが与えられたときに、以下の規則に従って点集合Ｖから点を一つ選ぶ演算($m_v(v_0, V, R) = v_s \in \{v \mid v \in V \wedge \neg v \# v_0\}$) である。

［規則１］関係Ｒの順序関係に従ってv_0の次の点を選択する。関係Ｒに関して比較不能かまたは関係Ｒが与えられないときは規則２に従う。

［規則２］一意識別子の関係R_{ID}の順序関係に従ってv_0の次の点を選択する。

（３）枝にそった移動

［定義１４］（枝の前後への移動）　枝の前(後)への移動とは、属性，値，始点(終点)を一にする枝の集合

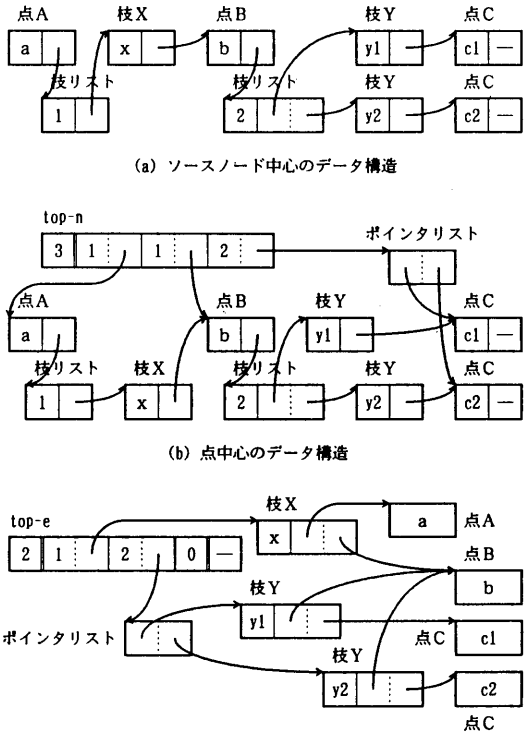


図３　選択可能とするデータ構造

ＥとＥの終点(始点)の一つv_0と終点(始点)の値の集合を半順序集合とする関係Ｒが与えられたときに、終点(始点)の集合に上記の規則１、２を適用して終点（始点）を一つ選ぶ演算である。枝の前への移動を$m_{ef}(v_0, E, R)$ ($= v_t \in \{t\{e\} \mid e \in E \wedge \neg t\{e\} \# v_{t0}\}$)，枝の後ろへの移動は$m_{eb}(v_{i0}, E)$ ($= v_i \in \{i\{e\} \mid e \in E \wedge \neg i\{e\} \# v_{i0}\}$) と表記する。

（４）付加，除去，複写

ここでは、更新系の演算について述べる。

［定義１５］（付加）　データグラフＰ（Ｖ，Ｅ）にグラフＱ（Ｖ'，Ｅ'）を付加するとは、点集合Ｖ∪Ｖ'および枝集合Ｅ∪Ｅ'を持つグラフ(Ｖ＋Ｖ'，Ｅ＋Ｅ')を作ることである。これをＰ＋Ｑで表す。従って、Ｐ＋Ｑ＝（Ｖ＋Ｖ'，Ｅ＋Ｅ'）。

［定義１６］（除去）　データグラフＰ(Ｖ,Ｅ)からグラフＱ(Ｖ'，Ｅ')を除去するとは、ＶからＶ'の点を除去し、除去される点に接続するＰの全ての枝とＥ'の和をＥから除去することである。これを、Ｐ－Ｑで表す。従って、Ｐ－Ｑ=(Ｖ-Ｖ'，Ｅ-(c{Ｖ,}∪Ｅ'))。ただし、

$V_r = \{v \mid v\#v' \wedge v \in V \wedge v' \in V'\}, V' \subseteq V,\ E' \subseteq E.$

4．2　非手続き的な基本演算

ここでは、非手続き的なデータグラフ演算を提案する。提案する演算は、集合演算，複写演算，部分グラフ化演算，統合演算，順序化，集約化および再帰に関する演算である。以下では特に断わらない限り、点または枝を要素と表記する。

4．2．1　集合演算

データグラフを構成する要素は一意識別子をもつので一意識別子を考慮した集合演算を定義する。

［定義１７］（共通部分，合併，差）　データグラフ$P_1(V_1, E_1)$とデータグラフ$P_2(V_2, E_2)$の共通部分$P = P_1 \cap P_2$，合併$P = P_1 \cup P_2$，差$P = P_1 - P_2$とは、それぞれ、点集合$V_1 \cap V_2$，枝集合$E_1 \cap E_2$を持つグラフ，点集合$V_1 \cup V_2$，枝集合$E_1 \cup E_2$を持つグラフ，V_2に属さないV_1の点集合とE_2に属さないE_1の枝集合を持つグラフである。

一意識別子を考慮せず要素の値のみに着目した集合演算は、値を一にする要素を同一要素とする必要があり、以降で述べる集合統合によって実現できる。

4．2．2　複写演算

さらに、複写演算を定義する。

［定義１８］（複写）　データグラフPの複写copyPは、データグラフPと全等価であり絶対等価でないデータグラフである。

4．2．3　部分グラフ化演算

データグラフの部分グラフを取り出す演算として射影と制約を定義する。

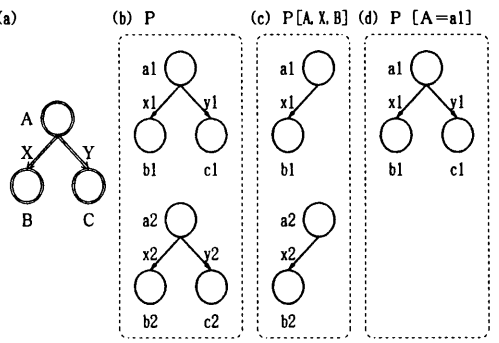
［定義１９］（射影）　データグラフPの点または枝の属性$C_1, \cdots, C_n$への射影$P[C_1, \cdots, C_n]$は、指定された属性の要素のみからなるグラフである。

［定義２０］（制約）　データグラフPの定義域が等価な２つの属性C_pとC_qの間のθ制約$P[C_p \theta C_q]$は、C_pとC_qの少なくとも一組の要素の値がθ関係を満たすデータ表現グラフを成分とするグラフである。

［例１］　図４（ａ）の定義に従った図４（ｂ）のようなデータグラフPに対して、射影を行った結果を図４（ｃ）に、制約を行った結果を図４（ｄ）に示す。

4．2．4　統合演算

データグラフ間の統合に関する演算として、直統合，制約統合，関係統合，集合統合を定義する。



(a)データグラフPの定義　(b) データグラフP
(c)射影の例　(d) 制約の例

図4　部分グラフ化の例

（１）直統合　直統合は要素を無条件に統合する。

［定義２１］（直統合）　データグラフPの属性C_pへのデータグラフQの属性C_qの直統合$P[C_p \langle\langle C_q]Q$は、属性$C_q$の要素を属性$C_p$の要素としたグラフである。

（２）制約統合　制約統合は、要素の値のθ関係（θは＝，≠，＜，＞，≦，≧）を使用して要素を統合する。

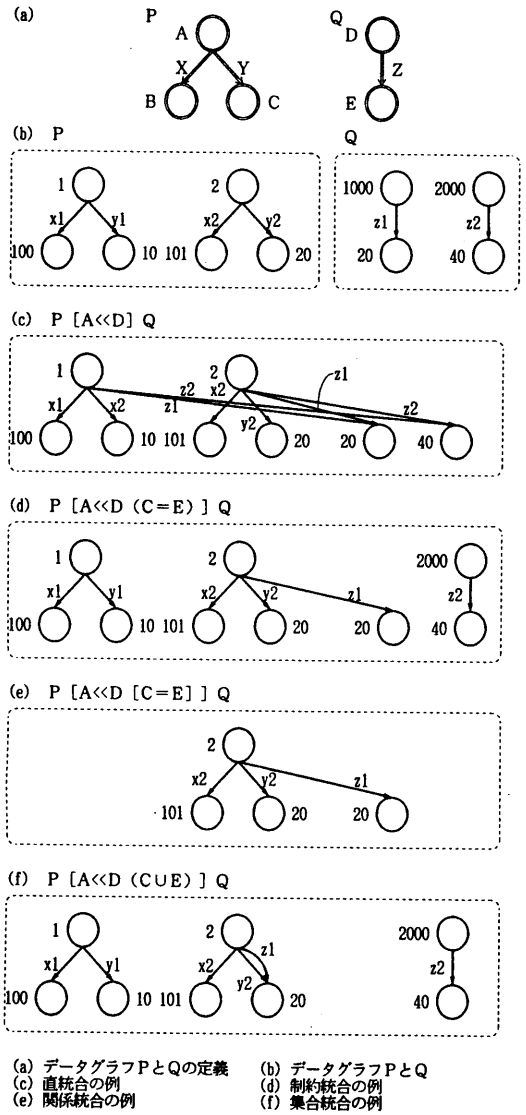
［定義２２］（制約統合）　データグラフPとその属性C_p，C_{pc}，データグラフQとその属性C_q，C_{qc}があるとき、Pの属性C_pへのQの属性C_qの$C_{pc} \theta C_{qc}$に基づく制約統合$P[C_p \langle\langle C_q(C_{pc} \theta C_{qc})]Q$は、属性$C_{qc}$の要素の値が属性$C_{pc}$の要素の値と$\theta$関係のときに限り$C_q$の要素を$C_p$の要素としたグラフである。

（３）関係統合　関係統合は、要素の値のθ関係（θは＝，≠，＜，＞，≦，≧）を使用し、θ関係を満たす要素のみを結果とする。

［定義２３］（関係統合）　データグラフPとその属性C_p，C_{pc}，データグラフQとその属性C_q，C_{qc}があるとき、Pの属性C_pへのQの属性C_qの$C_{pc} \theta C_{qc}$に基づく関係統合$P[C_p \langle\langle C_q[C_{pc} \theta C_{qc}]]Q$は、属性$C_{qc}$の要素の値が属性$C_{pc}$の要素の値と$\theta$関係のときに限り$C_q$の要素を$C_p$の要素とし、$\theta$関係にないデータグラフの要素を含まないグラフである。

（４）集合統合　集合統合は値の集合に集合演算ω（ωは共通部分($\cap$)，合併($\cup$)，差($-$)）を行って統合する。

［定義２４］（集合統合）　データグラフPとその属性C_p，C_{pc}、データグラフQとその属性C_q，C_{qc}があるとき、Pの属性C_pへのQの属性C_qの$C_{pc} \omega C_{qc}$に基づく集合統合$P[C_p \langle\langle C_q(C_{pc} \omega C_{qc})]Q$は、属



(a) データグラフPとQの定義　(b) データグラフPとQ
(c) 直結合の例　(d) 制約結合の例
(e) 関係結合の例　(f) 集合結合の例

図5　結合の例

性C_{QC}の要素の値の集合と属性C_{PC}の要素の値の集合に集合演算ωを施した結果の値の集合をもつようにC_Qの要素をC_Pの要素としたグラフである。

［例２］　図５(a)のような定義で図５(b)に示すデータグラフＰとＱに対して、直結合，制約結合，関係結合，集合結合を行った例を各々図５の(c)～(f)に示す。

４．２．５　順序化に関する演算

データベースから要素の値で順序付けして検索結果を得ることは利用者にとって不可欠である。このために、データベースへのデータ格納時に順序付けておくことも可能であるが、リレーショナル型のＤＢＭＳのように動的に順序化する演算も必要である。

［定義２５］（順序化）　データグラフＰの属性C_1，…，C_kの各属性の順序組$(C_1,\cdots,C_k)$において関係Ｒが半順序であるとき、属性C_1，…，C_kにおける関係Ｒに関する順序化Ｐ〔order$(C_1,\cdots,C_k)$，Ｒ〕とは、組$(a_{1i},\cdots,a_{1k})$を要素とする集合を関係Ｒに関して半順序集合とすることである。ただし、$a_{1i}=k\{z\{g_1:C_{1i}\}\}$，$g_1\in$Ｐである。

４．２．６　集約化に関する演算

データを集約したいという要求もデータベース利用者にはある。すなわち、データの総数、データの値の合計、平均、最大、最小を求めることである。ここでは、集約化ならびに集約演算を定義する。

（１）集約化

［定義２６］（集約化）　データグラフＰの属性C_1，…，C_kに関する集約化とは、各属性Ｃiの要素c_iからなる順序組$(c_1,\cdots,c_k)$が同一である場合に、同一とみなす属性の要素を統合することである。

集約化は、∪集合統合を用いて表現できる。例えば、$P\{C_1\langle\langle C_1,\cdots,C_k\langle\langle C_k((C_1,\cdots,C_k)\cup(C_1,\cdots,C_k))\}\}$Ｐや、$P\{C_0\langle\langle C_0,C_1\langle\langle C_1,\cdots,C_k\langle\langle C_k((C_1,\cdots,C_k)\cup(C_1,\cdots,C_k))\}\}$Ｐである。ただし、$(C_1,\cdots,C_k)$は順序組を表す。

集約化によって同一とみなされる１以上のデータ表現グラフは１データ表現グラフとなる。さらに、（２）で述べる集約演算を用いて集約化されなかった要素の集約値を得ることができる。

（２）集約演算

［定義２７］（点の縮約）　点集合Ｖの要素を単一の点とすることを点の縮約という。縮約される前の点を始終点とする枝は、点の縮約によって縮約後の点を始終点とする枝に変更される。

［定義２８］（枝の縮約）　始終点を一にする枝集合Ｅの要素を一つの枝とすることを枝の縮約という。

［定義２９］（縮約の表記と値）　要素集合Ｗの縮約をcompＷと表記し、Ｗの全要素が同一値ｄであるとき、compＷの値をｄとし、それ以外はNULL(値なし)とする。

点の縮約と枝の縮約の順序によって縮約結果が異なるので注意が必要である。

［定義３０］（カウント，合計，平均，最大，最小）

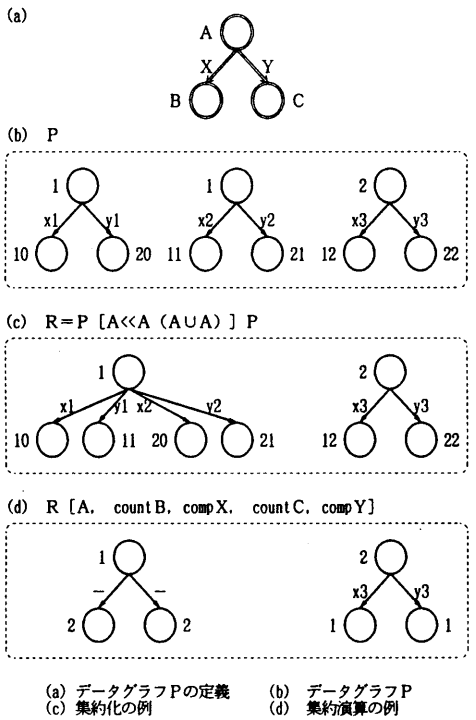
値が可算な要素集合Wに対して、要素集合Wの要素を、要素の数を値にもつ要素w_{count}に縮約する演算，Wの要素の値の総和を値にもつ要素w_{sum}に縮約する演算，Wの要素の値の平均値を値にもつ要素w_{avg}に縮約する演算，Wの要素の値の最大値を値にもつ要素w_{max}に縮約する演算，Wの要素の値の最小値を値にもつ要素w_{min}に縮約する演算を、それぞれカウント(countW)，合計(sumW)，平均(avgW)，最大(maxW)，最小(minW)という。

［定義３１］（集約演算）　データグラフPの属性Cについての集約演算P[go(C)]は、Pの成分であるデータ表現グラフ毎に属性Cの表す要素集合に集約演算goを施してできる以下のグラフである。ここで、集約演算goとは、count,sum,avg,max,min,compである。

(a)Cが点属性の場合、P[go(C)]= ({w_i}, {})。
ここで、w_i=goW_i，W_i={z{g_i:C} | g_i∈P}。

(b)Cが枝属性の場合、P[go(C)]= ({}, {w_i})。
ここで、w_i=goW_i，W_i={z{g_i:C} | g_i∈P}。

［例３］　集約化の例を図６に示す。データグラフPを(A,X,B,Y,C)に着目して集約化し（図６(c)）、さらに図６(d)では 属性B,X,C,Yの集約値を求めている。



(a) データグラフPの定義　(b) データグラフP
(c) 集約化の例　(d) 集約演算の例

図6　集約化の例

４．２．７　再帰に関する演算

データベース中のデータに対して演算を再帰的に行ってデータを検索したいということもしばしば起こる。ここでは、閉包，再帰について定義する。

［定義３２］（閉包）　データグラフ演算fがデータグラフPにおけるn項演算であり、p_i∈Pに対してf(p_1,…,p_n)∈Pが成立するとき、データグラフPはfのもとで閉じているという。

［定義３３］（再帰）　あるデータグラフ演算f(P_R, P_1,…,P_n)を再帰的に行うf*(P_R,P_1,…,P_n)とは、データグラフ $\bigcup_{i=1}^{\infty} Q_i$ を作ることである。ここで、

$$Q_i = \begin{cases} f(P_R, P_1, \cdots, P_n) & (i=1), \\ f(Q_{i-1}, P_1, \cdots, P_n) & (i \geqq 2). \end{cases}$$

このとき、P_Rを被作用データグラフ，P_1,…,P_nを作用データグラフと呼ぶ。

４．基本演算の実現方法

ここでは、３種の格納構造に基づく射影，制約，統合演算の実現方法について述べる。再帰については現在検討中であり、その他の演算は容易に実現できる。

実現方法の検討にあたり、関係ＤＢのＯＰＥＮ処理を想定した。この結果、表２に示す４種の処理で実現できることが分かった。表２の各処理（処理１～処理４）の概要を各々図７～図１０に示した。点中心の格納構造と枝中心の格納構造は処理の流れは同じである。また、統合は、関係代数の結合と同じ処理で実現できる。従って、格納構造に依存するのは統合処理中の部分グラフ化演算（射影，制約）の処理である。

表２　基本演算の実現方法

基本演算	ｿｰｽﾉｰﾄﾞ中心	点／枝中心
射影	処理１	処理２
制約	処理１	処理３
統合	処理４	処理４

６．関連する研究

（１）データのグラフ表現

データを関連というノードを使用せずにグラフで表現するモデルにはＳＢＤＭ[18]，関数型データモデル[19,20]，ＳＤＭ[21]，DBGraph[23]等がある。

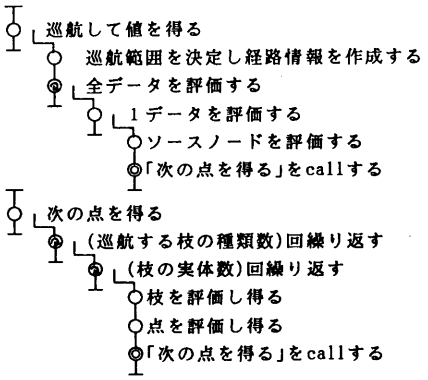


図7　処理1の概要

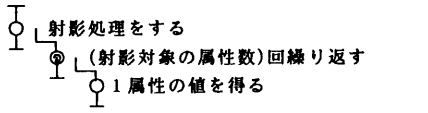


図8　処理2の概要

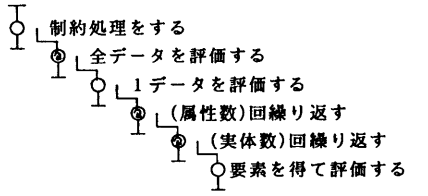


図9　処理3の概要

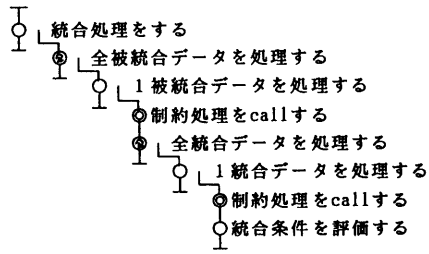


図10　処理4の概要

ＳＢＤＭでは、二項関連の順方向，逆方向のラベル付けによってデータ間の意味を表現する。また、二項関連が集合の意識を持つ。すなわち、二項関連の端点の値が一意か順序付けするかで二項関連は4分類され、集合の定義および操作を可能とする。本稿のデータグラフではこのような特徴はない。本データグラフではデータをグラフとして表現するのみにとどめ、種々の意味的なデータモデルの特徴をこのデータグラフに追加する形態で実現することを目標としている。

関数型データモデル，ＳＤＭで表現されたスキーマ，データ実体もグラフとなる。関数型データモデルではスキーマが一つのグラフで表現される。すなわち、問い合わせに出現するエンティティはあらかじめエンティティ間の関係(関数)が定義されていると考えている。従って、非連結な2グラフをスキーマとするデータを一問い合わせで処理することはない[20]。データグラフでは、データ群グラフを導入することで非連結なスキーマの表現を考慮している。

DBGraph[23]では、エンティティを表現する点とデータ値を表現する点で構成される2部グラフによりデータを表現する。DBGraphでは 同種の点同志を結ぶ枝は考慮されていない。また、点に接続する枝は枝の終点を表現するのみであり、データ構造としては本質的に従来のソースノード中心のデータ構造である。

（２）基本演算

データを表現するグラフに対する従来からの基本的な演算は、グラフ中の経路を巡航する[24]かまたは巡航経路を指定する[25]ものである。前者には利用者が検索経路を指定して手続き的に検索しなければならないという欠点がある。後者には操作対象のデータはすべて関連付けられているという前提があり、非連結なデータを表現するグラフを簡単に操作できないという欠点がある。本稿で提案した演算を使用すれば、非手続き的な検索，非連結な操作対象の操作が可能である。

意味的なデータモデルであるＳＤＭに基づくＤＢＭＳ[22]やオブジェクト指向データモデルの問い合わせ言語[16,17]では関係モデルの結合と同様な演算の提案があるが、これらは関係モデルの結合を単に流用している程度である。これに対し提案した統合演算はデータのグラフ化を基に導いた一般的な演算である。

7．おわりに

本稿では、拡張可能ＤＢＭＳ：ＣＯＭＭＯＮで採用するデータを表現するグラフのための基本演算の提案と基本演算の実現方法について述べた。

まず、拡張可能ＤＢＭＳ：ＣＯＭＭＯＮの概要について述べた。ＣＯＭＭＯＮは、ＤＢＩの負荷軽減と多種のデータモデル実現を目標としたＤＢＭＳgeneratorである。ＣＯＭＭＯＮでは、データを表現するグラフ（データグラフ）を多種の意味的なデータモデル実現の

プラットフォームとし、データグラフのための３種の格納構造を選択可能としている。

次に、データグラフのための基本演算について述べた。従来の巡航的な基本演算を示し、手続き的に使用せずにデータグラフを操作可能な基本演算を提案した。これは、集合，複写，部分グラフ化，統合，順序化，集約化および再帰に関する演算である。

さらに、ＣＯＭＭＯＮで選択可能とする３種の格納構造に基づいて基本演算の実現方法を示した。部分グラフ化演算処理は格納構造に依存する。

最後に、データグラフ，基本演算について関連する研究と比較した。データグラフは、単にデータをグラフで表現するもので、多種の意味的なデータモデルのプラットフォームに適した表現である。また、提案した非手続き的な演算はデータのグラフ化に基づく一般的な演算である。

今後は、点，枝に着目した格納構造の格納効率向上の検討，データグラフの表現力評価，柔軟な最適化方式の検討，データベース言語処理の一貫性の検討，ＣＯＭＭＯＮの拡張性評価が課題である。

謝辞　本研究を行うにあたり、日頃から議論して頂く田中豪主幹員、田中一敏主幹員をはじめとするＮＴＴ情報通信処理研究所の皆様に感謝致します。

文　献

(1) D. Batory and M. Mannino :" Panel on Extensible Database Systems", Proc. of ACM SIGMOD'86, pp.187-190, 1986.

(2) M. Stonebraker, L. A. Rowe and M. Hirohama :"The implementation of POSTGRES", IEEE trans. of knowledge and data eng., Vol.2, No.1, pp.125-142, 1990.

(3) L. M. Haas et al:"Starburst mid-flight: As the dust clears", IEEE trans. of knowledge and data eng., Vol.2,No.1,pp.125-142,1990.

(4) D. S. Batory et al:"Implementation concepts for an extensible data model and data language", ACM Trans. Database Sys., Vol.13, No.3, pp.231-262, 1988.

(5) M. J. Carey et al:" Object and File Management in the EXODUS extensible database system", Proc. of 12th VLDB, pp.91-100, 1986.

(6) M. J. Carey et al:"A Data Model and Query Language for EXODUS", Proc. of ACM SIGMOD '88, pp.413-423, 1988.

(7) S. Hong and F. Maryanski:" Using a meta model to represent object-oriented data models", Proc. of 6th Intl. Conf. on DATA Engineering, pp.11-19, 1990.

(8) 古瀬 他:"拡張可能ＤＢＭＳ ＭＯＤＵＳの設計と実現技術", Proc. of ADDS'90, pp.139-148, 1990.

(9) 宝珍:"拡張可能データベース管理システム構築についての一考察",情処第40回全大5H-1,1990.

(10) 宝珍:"拡張可能ＤＢＭＳ:ＣＯＭＭＯＮの基礎的性能評価", 情処第41回全大1D-8, 1990.

(11) 宝珍:"拡張可能ＤＢＭＳ：ＣＯＭＭＯＮの格納構造について", 情処第42回全大 1L-1, 1991.

(12) 佐藤 他:"ＣＡＤ技術を応用した地図処理システム", 情処ﾃﾞｰﾀﾍﾞｰｽｼｽﾃﾑ研究会52-6, 1986.

(13) R. Hull and R. King:"Semantic database modeling: Survey, applications, and research issues", ACM Computing Survey, Vol.19, No.3, pp.201-260, 1987.

(14) 小野寺:グラフ理論の基礎,森北出版,1968.

(15) 植村：データベースシステムの基礎,オーム社, 1979.

(16) W. Kim :"A model of queries for object-oriented databases", Proc. of 15th VLDB, pp.423-432, 1989.

(17) A. M. Alashqur et al:"OQL:A query language for manipulating object-oriented databases ", Proc. of 15th VLDB, pp.433-442, 1989.

(18) J. R. Abrial:"Data Semantics",in Data Base Management(J. W. Klimbie and K.L. Koffeman eds),pp.1-59,North-Holland,Amsterdam,1974.

(19) E.H. Sibley and L. Kerschberg:"Data architecture and data model considerations", proc. of AFIPS Nat. Computer Conf., Vol.46, pp.85-96, 1977.

(20) A. Chan, S. Danberg, S. Fox, W. K. Lin, A. Nori and D. Ries:"Storage and access structures to support a semantic data model",Proc. of 8th VLDB,pp.122-130, 1982.

(21) M. Hammer and D. McLeod: " Database Description with SDM: A semantic database model", ACM Trans. Database Sys., vol.6, No.3, pp.351-386, 1981.

(22) D. Jagannathan et al:" SIM: A database system based on the semantic data model", Proc. of ACM SIGMOD'88, pp.46-55, 1988.

(23) P. Pucheral et al:"Efficient Main Memory Data Management Using the DBGraph Storage Model", Proc.of 16th VLDB,pp.683-695,1990.

(24) 日本規格協会，日本工業規格データベース言語ＮＤＬ，JIS X 3004, 1987.

(25) D. M. Campbell, D. W. Embley and B. Czejo: "A relationally complete query language for an entity-relationship model", Proc. of 4th Intl. Conf. on Entity-Relationship Approach, pp.90-97, 1985.