入れ子型リレーションに対する結合演算問い合わせの最適化アルゴリズムとその評価

北川博之† 李亜新††

入れ子型リレーショナルモデルはリレーショナルモデルの拡張として提案され，より複雑な構造を持つデータを支援することが可能である．本論文では，入れ子型リレーショナルデータベースに対する結合演算問い合わせの最適化アルゴリズムを提示し，その有効性の評価を行う．本アルゴリズムは，入れ子型リレーションに対する結合演算の一種である入れ子型結合と埋込みの2種類の演算を対象とし，これらの演算からなる問い合わせに対する左連鎖線形処理木を導出する．この左連鎖線形処理木は，ある種の仮定の下で全左連鎖線形処理木のうちで最小コストのものとなる．また，シミュレーション実験により上記の仮定が厳密に成立しない状況下での本アルゴリズムの有効性を検証する．実験結果によれば，本アルゴリズムはこのような状況下でも全左連鎖線形処理木のうち最小コストの処理木，あるいは最小コストに近い低コストの処理木を導出する．また，本アルゴリズムが導出する処理木は，一般の線形処理木全体の中でも低コストのものであることが明らかとなった．

An Optimization Algorithm for Join-Type Queries on Nested Relations and Its Evaluation

Hiroyuki Kitagawa† and Yaxin Li††

Since the nested relational model is effective in supporting new emerging applications that involve complex data structures, intensive research efforts have been devoted to nested relations. One important research issue on nested relations is query processing, in particular query optimization. We focus on two join-type operations on nested relations: nested join and embed. In this paper, we present an algorithm to derive a left-deep linear processing tree for a given query consisting of nested joins and embeds. We show that the algorithm generates a cost optimal left-deep linear processing tree under some assumptions. Then, we present computer simulation results to evaluate effectiveness of the algorithm in more practical situations where the assumptions do not hold in a strict sense. The simulation results show that the algorithm generates a cost optimal or quasi-optimal left-deep linear processing tree. The cost of the derived left-deep linear processing tree is rather low even compared with other more general linear processing trees.

1. はじめに

近年の新しいデータベース応用においては，従来よりもより複雑な構造を持つデータの表現と操作の支援が，データベース管理システムに求められている[9),16),29)]．入れ子型リレーショナルモデル (nested relational model) は，このような要求に対応するためのリレーショナルモデルの拡張として提案され研究が行われている[2),18),19)]．リレーショナルモデルの第一正規形制約を緩和した入れ子型リレーションの概念は牧ノ内により提案され[17)]，リレーショナル代数を拡張した基本代数操作系が提案された[7),11)]．その後，入れ子の内部構造を直接操作可能とした代数の提案[3),12),21)]や，べき集合演算子の導入により記述力を高めた代数の提案[8)]等が行われた．また，入れ子型リレーションのデータ表現に意味的制約の一種を課した，分割標準形 (partitioned normal form)[18)] と呼ばれるサブクラスを対象とした代数操作系の研究も行われている[1),20)]．さらにまた，いくつかの入れ子型リレーショナルデータベース管理システムのプロトタイプ実装も報告されている[2),4)~6),23)]．

入れ子型リレーションに対する問い合わせ処理においては，リレーショナルデータベースにおける場合と

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

同様に，高コストの結合演算の実行順序に注意することが必要である[6),23)]．リレーショナルデータベースに対する結合演算問い合わせの最適化に関しては，動的計画法に基づく手法[24)]，アニーリング法等のランダム探索に基づく手法[26)]，木構造問い合わせグラフを対象とした IK 法[10)]，KBZ 法[13)]等の多項式オーダ最適化手法，ランダム探索と KBZ 法等の組合せによる手法[27),28)]等が，これまで研究されてきた．一方，入れ子型リレーションに対する問い合わせ処理に関しては，代数演算子の可換性に基づく手法[3),7),23)]，入れ子型リレーションの階層構造に従ってデータを格納したファイルのスキャンをベースとした手法[4),22),23)]，転置ファイルに基づく手法[5)]，結合演算の並列処理法[6)] 等が，これまで研究されてきた．しかし，入れ子型リレーションに対する複数の結合演算を含む問い合わせにおいて，その最適な実行順序を求める有効な手法は知られていない．

本論文では，リレーショナルデータベースに対する結合演算問い合わせの最適化手法である KBZ 法[13)]をベースとした，入れ子型リレーショナルデータベースに対する結合演算問い合わせの最適化アルゴリズムを提示し，その有効性の評価を行う．本アルゴリズムは，入れ子型リレーションに対する結合演算の一種として入れ子型結合と埋込み[6),12)]の2種類の演算を対象とし，これらの演算からなる問い合わせに対する左連鎖線形処理木を導出する．この左連鎖線形処理木は，ある種の仮定の下では全左連鎖線形処理木のうちで最小コストのものとなる．また，シミュレーション実験により上記の仮定が厳密に成立しない状況下での本アルゴリズムの有効性を検証する．まず，本アルゴリズムより導出される左連鎖線形処理木が，左連鎖線形処理木全体の中で，どの程度コストの最小性を満足するものかを評価する．次に，同様の評価をより一般の線形処理木全体の中で行う．これらのシミュレーション実験結果より，本アルゴリズムの有効性を示す．

本稿は，以下のように構成されている．第2章では，各種基本概念について説明する．第3章では，問い合わせの最適化アルゴリズムの前提となるコストモデルと最適化アルゴリズムを示す．第4章では，本アルゴリズムの評価のためのシミュレーションについて述べる．第5章では，本アルゴリズムの適用範囲の拡張の可能性について検討する．第6章では，本稿のまとめを行う．

2. 基本概念

2.1 入れ子型リレーション

入れ子型リレーションは，リレーショナルモデルの第一正規形制約を緩和し，その属性値として単純値のみでなく，リレーション値を許すものである．**図1**は，入れ子型リレーション R_1 の例である．R_1 のスキーマを $S_0(A, B, S_1(C, S_2(D)))$ で表す．属性 S_0, S_1 S_2 はリレーション値属性であり，属性 A, B, C, D は単純値属性である．以下では議論の単純化のため，すべての単純値属性の属性値の定義域は同一であるものとする．属性 A, B, S_1 は S_0 の子属性で，S_0 はこれらの親属性にあたる．同様に属性間の“子孫”や“先祖”等の関係が定義される．リレーション値属性 S_0 をリレーション R_1 のルート属性と呼ぶ．特に，ルート属性以外のリレーション値属性の属性値として現れるリレーションを部分リレーションと呼び，部分リレーションの組を部分組と呼ぶ．リレーション R_1 において，リレーション値属性 S_k の属性値として現れる各（部分）リレーション中の（部分）組の合計を $\gamma(R_1, S_k)$ で表す．また，S_k が R_1 のルート属性である場合，$\gamma(R_1, S_k)$ を単に $\gamma(R_1)$ で表す．図1では，$\gamma(R_1, S_0)=\gamma(R_1)=2$, $\gamma(R_1, S_1)=3$, $\gamma(R_1, S_2)=4$ である．

本稿での入れ子型リレーションの位置付けは，代表的入れ子型リレーショナルモデルの1つである NF^2 モデル[21)]等におけるものと同様であり，分割標準形のような表現上の制約は考慮しない．したがって，以下で述べる結合演算では，その適用対象および実行結果のリレーションにおいて，すべての単純値属性が同じでリレーション値属性の値のみが異なる（部分）組が同一の（部分）リレーション中に存在することを禁止しない．

2.2 入れ子型結合[6,12)]

入れ子型結合（nested join）演算は，リレーショナル代数における結合演算の拡張である．本論文では，入れ子型結合のうちの入れ子型自然結合のみを考え，

S0			
A	B	S1	
		C	S2
			D
X	Y	X	X
			Z
		Y	X
Y	Z	Z	Y

図1 入れ子型リレーション
Fig. 1 Nested relation.

結合属性としては単純値属性のみを対象とする．

S_k を R_i のリレーション値属性とした時，入れ子型自然結合 $R_i\bowtie[S_k]R_j$ は，R_i の各 S_k（部分）リレーションをリレーショナル代数の自然結合と同様の様式で，R_j と結合する．S_k の子属性の集合を C_k，R_j のルート属性の子属性の集合を C_j，$C=C_k\cap C_j$ とする．この時，$R_i\bowtie[S_k]R_j$ の実行結果は，R_i における S_k の各（部分）リレーション O_k を，属性集合 $C_k\cup C_j$ に対して定義された以下の（部分）組 t の集合 O_k' で置き換えたものである．

$$O_k'=\{t|(\exists t_k)(\exists t_j)(t_k\in O_k\wedge t_j\in R_j \wedge t[C]=t_k[C]=t_j[C] \wedge t[C_k-C]=t_k[C_k-C] \wedge t[C_j-C]=t_j[C_j-C])\}.$$

例として，入れ子型自然結合 $R_1\bowtie[S_1]R_2$ を**図2**（a）に示す．

入れ子型自然結合 $R_i\bowtie[S_k]R_j$ の結合選択率 SJ_{ij} を，以下のように定義する．

$$SJ_{ij}=\frac{\gamma(R_i\bowtie[S_k]R_j,\ S_k)}{\gamma(R_i,\ S_k)*\gamma(R_j)}.$$

S_k が R_i のルート属性である場合，$R_i\bowtie[S_k]R_j$ を単に $R_i\bowtie R_j$ で表す．この場合，入れ子型自然結合は本質的にはリレーショナル代数の自然結合と等価となる．$R_i\bowtie[S_k]R_j$ において，S_k をターゲット属性と呼ぶ．以下では，特に混乱の恐れのない限り，入れ子型自然結合を単に結合と呼ぶ．

入れ子ループ結合（nested loop join）は，結合を実行するための最も基本的アルゴリズムであり，リレーショナル代数の結合アルゴリズムとして知られている同様のアルゴリズムを拡張したものである．$R_i\bowtie[S_k]R_j$ に対する入れ子ループ結合では，基本的には，R_i を外部リレーション（outer relation），R_j を内部リレーション（inner relation）とする．入れ子ループ結合では，外部リレーション R_i のターゲット属性 S_k の各（部分）組 t_k に対して，内部リレーション R_j のすべての組 t_j を順次突き合わせ上記の条件 $t_k[C]=t_j[C]$ を満たすかどうか判定することで，結果のリレーションを生成する．

S_k がルート属性の場合，結合 $R_i\bowtie[S_k]R_j$ は上に述べたようにリレーショナル代数の結合と等価であり，この場合 R_i を内部リレーション，R_j を外部リレーションとして入れ子ループ結合を実行することも可能である．

2.3 埋 込 み[12)]

埋込み（embed）は，2つの入れ子型リレーションを融合して結果のリレーションを導出する点では入れ子型結合と同様であるが，融合の方式が異なる．本論文では，埋込みのうち自然埋込みのみを考える．

S_k を R_i のリレーション値属性とした時，自然埋込み $R_i\varepsilon[S_k]R_j$ は，R_i 中の S_k の（部分）リレー

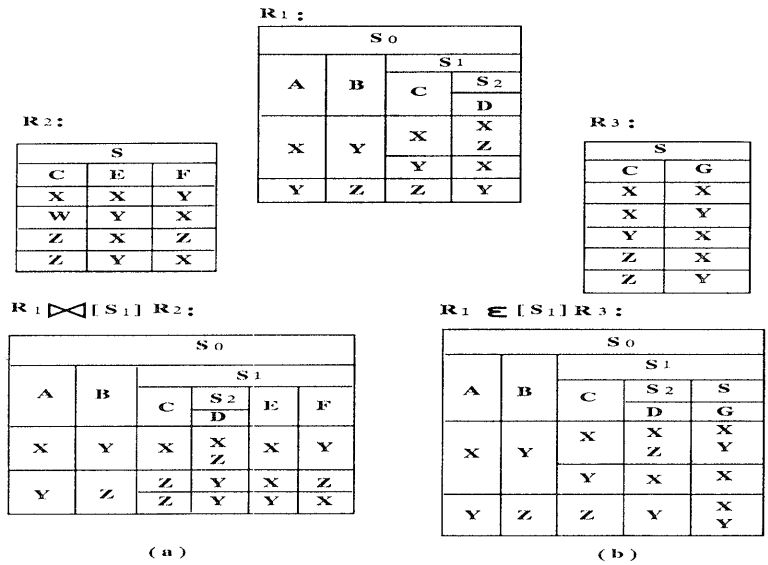

図 2 結合と埋込み
Fig. 2 Join and embed.

ションに出現する各（部分）組 t_k に対して，入れ子型自然結合と同様に対応する R_j の組を求める．自然埋込みでは，S_k の子属性として R_j のルート属性に対応した新たなリレーション値属性 S を生成し，各 t_k に対して対応する R_j の組の集合からなる部分リレーションを S の属性値として付加する．すなわち，$R_i \varepsilon[S_k]\, R_j$ の実行結果は，R_i における S_k の各（部分）組 t_k に対して，新たに付加されるリレーション値属性 S に対する属性値として，属性集合 $C_j - C$ に対して定義された以下の部分組 t の集合 O' を付加したものである．

$$O' = \{t \mid (\exists t_j)(t_j \in R_j \wedge t_k[C] = t_j[C] \wedge t = t_j[C_j - C])\}.$$

ただし，C_j，C の意味は自然結合に対する場合と同じである．例として，自然埋込み $R_1 \varepsilon[S_1]R_3$ を図2（b）に示す．

埋込み $R_i \varepsilon[S_k]R_j$ に対する埋込み選択率 SE_{ij} を，以下のように定義する．

$$SE_{ij} = \frac{\gamma(R_i \varepsilon[S_k]R_j, S)}{\gamma(R_i, S_k) * \gamma(R_j)}.$$

S_k が R_i のルート属性である場合，$R_i \varepsilon[S_k]R_j$ を単に $R_i \varepsilon R_j$ で表す．$R_i \varepsilon[S_k]R_j$ において，S_k をターゲット属性，S を付加属性と呼ぶ．以下では，自然埋込みを単に埋込みと呼ぶ，

入れ子ループ結合と同様，入れ子ループ埋込み（nested loop embed）は埋込み演算を実行するための基本的アルゴリズムである．$R_i \varepsilon[S_k]R_j$ に対する入れ子ループ埋込みでは，R_i を外部リレーションとし R_j を内部リレーションとする．外部リレーション R_i のターゲット属性 S_k の各（部分）組 t_k に対して，内部リレーション R_j のすべての組を順次突き合わせ，上記の条件 $t_k[R] = t_j[C]$ を満たす R_j の組 t_j に対応した付加属性 S の部分組 t を生成することで，結果のリレーションを生成する．

2.4 問い合わせグラフ

結合と埋込みからなる問い合わせを表現するのに，問い合わせグラフ（query graph）を用いる．問い合わせグラフ中で，ノードは問い合わせの対象となる基本リレーション（base relation）を表し，辺は2つのリレーション間の結合/埋込み演算を表す．辺は結合辺と埋込み辺に分類される．結合 $R_i \bowtie[S_k]R_j$ を表す結合辺は，R_i から R_j へのラベル S_k をもつ実線の有向辺である．埋込み $R_i \varepsilon[S_k]R_j$ を表す埋込み辺は，R_i から R_j へのラベル S_k をもつ点線の有向辺である．ただし，S_k が R_i のルート属性である場合には，結合辺あるいは埋込み辺のラベルを省略することができる．また，この場合結合は基本的には対称な演算となるので，結合辺は無向辺として表す．**図3**は問い合わせグラフの例である．結合辺しか含まない連結部分グラフを結合クラスタと呼ぶ．図3では，C_1〜C_5 が結合クラスタである．

一般の問い合わせグラフとしては種々のものが考えられるが，本論文の以下では，次の条件を満足する問い合わせグラフを対象とする．

1. すべての結合辺 $R_i \bowtie[S_k]R_j$ および埋込み辺 $R_i \varepsilon[S_k]R_j$ において，S_k は R_i のルート属性である．
2. 結合クラスタをノード，結合クラスタ間にまたがる埋込み辺を有向辺とみなしたグラフは，ルート付木構造をなす．
3. 各結合クラスタ内部の基本リレーションと結合辺からなる構造は木構造をなす．

上記2より，結合クラスタ間に親子関係が定義される．また，親結合クラスタがない結合クラスタをルート結合クラスタと呼ぶ．図3の問い合わせグラフは，上の条件をすべて満足し，C_1 がルート結合クラスタである．

2.5 処理木

問い合わせの実行プランは，処理木（processing tree）と呼ぶ二分木で表す．**図4**に処理木の例を示す．処理木の葉は基本リレーションを表し，中間ノードは結合演算および埋込み演算を表す．処理木では，リレーションを楕円，結合を矩形，埋込みを三角形で表す．中間ノードの左の子は，この中間ノードに対応する結合/埋込み演算実行時の外部リレーションを表し，右の子は内部リレーションを表す．もし，各中間ノードの2つの子の少なくともいずれか一方が基本リレーションの時，この処理木を線形処理木（linear processing tree, LPT）と呼ぶ．図4は図3の問い合わせに対する LPT の1つである．LPT においてすべて

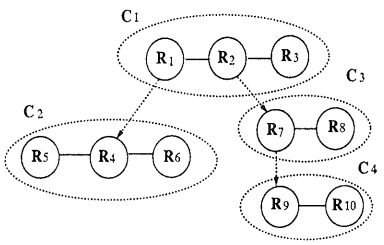

図3 問い合わせグラフ
Fig. 3 Query graph.

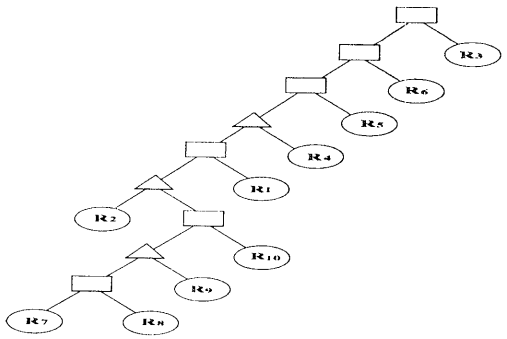

図 4 LPT
Fig. 4 LPT.

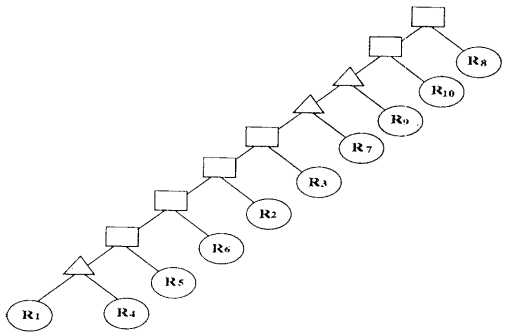

図 5 LD-LPT
Fig. 5 LD-LPT.

の中間ノードの右の子が基本リレーションの時，左連鎖線形処理木 (left-deep linear processing tree, LD-LPT) と呼ぶ．LD-LPT の最も左下のリレーションを先頭リレーションと呼ぶ．

本論文では，前節で述べた問い合わせグラフに対する条件，および入れ子ループ埋込み $R_i\varepsilon[S_k]R_j$ において常に R_i が外部リレーション，R_j が内部リレーションとなることより，LD-LPT の先頭リレーションは常に問い合わせグラフのルート結合クラスタ中の基本リレーションとなる．**図 5** は図 3 の問い合わせに対する LD-LPT の 1 つである．

以下の説明では，LD-LPT を先頭リレーションから始まるリレーション列として表現する．この方法による図 5 の LD-LPT の表現は，$R_1R_4R_5R_6R_2R_3R_7R_9R_{10}R_8$ となる．

3. 問い合わせ最適化処理

3.1 仮　　定

コストモデルと最適化アルゴリズムを構築する上で，以下の仮定を行う．

1. データベースは主記憶上に存在するものとし，主記憶中での（部分）組の比較回数に基づいて，結合と埋込みの処理コストを評価する．
2. 各結合の結合選択率および埋込みの埋込み選択率は，演算の実行順序によらずそれぞれ独立に決定される．すなわち，$R_i\bowtie[S_k]R_j$ に対する結合選択率 SJ_{ij} はこれらの 2 つのリレーションを最初に結合する場合のみでなく，例えば，R_i と他のリレーションを結合/埋込みした後にこの結合演算を実行する場合にも適用可能である．
3. すべての基本リレーションおよび中間リレーションにおいて，各リレーション値属性 S_k に対する部分組は均等に分布しており，S_k のある部分リレーションが多数の部分組を有し，S_k のある部分リレーションが少数の部分組しか有さないといったことはない．

3.2 コストモデル

処理木のコストは，その中に含まれるすべての結合と埋込み演算の処理コストの総和である．以下では，各演算の処理コストを示す．

入れ子ループ結合に基づく結合 $R_i\bowtie[S_k]R_j$ では，R_i のターゲット属性 S_k の各（部分）組と R_j の組を逐次比較することによって結果リレーションを生成する．前節の仮定 1 により，この組同士の比較回数が結合コストとなり，以下の式で表される．

$$Cost(R_i\bowtie[S_k]R_j)=\gamma(R_i,S_k)*\gamma(R_j)$$

同様に，入れ子ループ埋込みに基づく埋込み演算のコストは，以下の式で表される．

$$Cost(R_i\varepsilon[S_k]R_j)=\gamma(R_i,S_k)*\gamma(R_j)$$

上記のように，各演算の処理コストの算出においては，γ で表される当該（部分）組の総数を求める必要がある．これらは，各基本リレーションに関してはデータベースパラメタとして最初に与えられる．中間リレーションに関しては，以下の方法で導出する．

結合 $R_i\bowtie[S_k]R_j$ により生成される中間リレーションのリレーション値属性 S_m の（部分）組数 $\gamma(R_i\bowtie[S_k]R_j,S_m)$ は，前節の仮定 3 と結合演算の基本性質より以下のように計算される．

ケース 1：S_m が S_k 自身あるいは S_k の子孫である場合：

$$\gamma(R_i\bowtie[S_k]R_j,S_m)=SJ_{ij}*\gamma(R_i,S_m)*\gamma(R_j).$$

ケース 2：それ以外：

$$\gamma(R_i\bowtie[S_k]R_j,S_m)=\gamma(R_i,S_m).$$

結合の場合と同様，埋込みに対する $\gamma(R_i\varepsilon[S_k]R_j,S_m$

は，以下のように計算される．

ケース1：S_m が付加属性 S 自身あるいは S の子孫である場合：

$$\gamma(R_i\varepsilon[S_k]R_j, S_m)=SE_{ij}*\gamma(R_i, S_m)*\gamma(R_j).$$

ケース2：それ以外：

$$\gamma(R_i\varepsilon[S_k]R_j, S_m)=\gamma(R_i, S_m).$$

R_i あるいは R_j が基本リレーションでなく中間リレーションの場合は，仮定2により中間リレーションを導出するための各演算に対して逐次この計算を繰り返すことより，該当する（部分）組数を求めることができる．

3.3 問い合わせ最適化アルゴリズム

本節では，問い合わせ最適化アルゴリズムを示す．本アルゴリズムは，問い合わせグラフ，各基本リレーションの各（部分）組数，各演算の選択率が与えられた時，この問い合わせに対する LD-LPT を1つ導出する．

本アルゴリズム中では，リレーションデータベースにおける結合演算問い合わせの最適化手法として知られている KBZ 法をその一部の手続きとして用いる．以下では，まず KBZ 法について述べた後，本アルゴリズムを示す．

3.3.1 KBZ 法

KBZ 法は Krishnamurthy, Boral, Zaniolo により提案された手法であり[13]，IK 法[10]を拡張したものである．KBZ 法は，リレーショナルデータベースに対する結合木で与えられる問い合わせグラフ，各基本リレーションの組数，各結合の選択率が与えられた時，最小コストの LD-LPT を導出する．KBZ 法では，データベースは主記憶上にあり，結合演算 $R_i\bowtie R_j$ の処理コストは $\gamma(R_i)*CJ(R_j)$ で計算されるものと仮定する．ここで，$CJ(R_j)$ は R_j のみに依存するパラメタで，R_i の1組当たりの結合コストを表す．以下では，$CJ(R_j)$ を単位コストと呼ぶ．入れ子ループ結合を用い，処理コストを組同士の比較総数とした場合には，$CJ(R_j)=\gamma(R_j)$ となる．

KBZ 法は，2つの手続き KBZ_1 と KBZ_2 から構成される．KBZ_1 は結合木とその中の1つのリレーション R を受け取り，R を先頭リレーションとする最小コストの LD-LPT を出力する．KBZ_1 の中では，R 以外の各リレーション R_i に対して，$(\gamma(R_i)*JS_i-1)/CJ(R_i)$ で表されるランク（rank）という指標を計算する．ただし，ここで JS_i は R をルートとした結合木における R_i とその親リレーションとの結合選択率であり，$CJ(R_i)$ は R_i の単位コストである．このランクに基づき，R をルートとする結合木の葉方向から順次リレーション列をマージしてゆくことにより，最終的に R を先頭リレーションとする最小コストの LD-LPT を導出する．

KBZ_2 は結合木中のすべてのリレーションに対して順次 KBZ_1 と呼び出し，得られた全 LD-LPT のうちで最小コストのものを，最終的な LD-LPT として出力する．結合木が N 個のリレーションを含む時，KBZ_1 の計算量は $O(N\log N)$ であり，KBZ 法全体では $O(N^2)$ で実行可能であることが示されている[13]．

3.3.2 最適化アルゴリズム

以下では，本論文の最適化アルゴリズムを示す．本アルゴリズムが導出する LD-LPT は，常に次の条件を満足する．すなわち，問い合わせグラフ中の任意の埋込み $R_i\varepsilon R_j$（R_j を含む結合クラスタを C_j とする）に対し，$R_i\varepsilon R_j$ の実行直後に C_i および C_j の

```
Procedure OPT
Input:  問い合わせグラフ: Q
Output: Q に対する最小コストの LD-LPT: P
begin
    C_R ← Q のルート結合クラスタ;
    Q' ← C_R 中の全ての基本リレーションと結合辺からなる問い合わせグラフ;
    for each (C_R の子結合クラスタ: C_j)
            /* C_R と C_j の間の埋込み辺を R_i ε R_j とする */
        begin
            P_j ← SUBOPT(C_j, R_j);
            リレーション R(C_j) と結合辺 R_i ⋈ R(C_j) を Q' に追加.
            ただし, R(C_j) の単位コスト CJ(R(C_j)) と結合辺
            R_i ⋈ R(C_j) の選択率 SJ は下記とする:
                    CJ(R(C_j)) = γ(R_j) + SE_ij * Cost(P_j)
                    /* SE_ij は埋込み R_i εR_j の選択率,
                    Cost(P_j) は処理木 P_j のコスト */
                    SJ = 1/γ(R(C_j));
        end
    for each (C_R 中の各基本リレーション: R_k)
        P_Rk ← KBZ(Q', R_k);
    P ← 最小コストの P_Rk;
    for each (R(C_j))
        P 中の R(C_j) との結合を R_i ε R_j および P_j に展開する;
    P を返す;
end
```

図 6 手続き OPT
Fig. 6 Procedure OPT.

子孫の結合クラスタに関するすべての結合と埋込みが，それ以外の結合と埋込みに先行して実行されるという条件を満足する．このような LD-LPT を，以下では DFE-LD-LPT (depth-first-execution left-deep linear processing tree) と呼ぶ．図5の LD-LPT は DFE-LD-LPT の条件を満足する．例えば，埋込み $R_1 \varepsilon R_4$ の実行直後に，結合クラスタ C_2 中の結合 $R_4 \bowtie R_5$ および $R_4 \bowtie R_6$ が実行されている．

本アルゴリズムの主手続きは**図6**に示す *OPT* である．*OPT* は問い合わせグラフ Q を入力し，Q に対する最小コストの LD-LPT P を導出する．*OPT* は，Q のルート結合クラスタ C_R の各結合クラスタ C_j に対して，**図7**に示す手続き *SUBOPT* を呼び出す．ここで，C_R と C_j の間の埋込み辺を $R_i \varepsilon R_j$ とし，Q 中で C_j をルート結合クラスタとする部分木で表される部分問い合わせグラフを $Q(C_j)$ とすると，*SUBOPT* は $Q(C_j)$ に対する R_j を先頭リレーションとする最小コストの LD-LPT P_j を導出する．*OPT* は，上に述べたように DFE-LD-LPT のみを導出する．*OPT* 中では，この条件に対応し，埋込み $R_i \varepsilon R_j$ および P_j で表される一連の演算列をマクロな演算として抽象化し，これらのマクロ演算およびルート結合クラスタ C_R にもともと含まれている結合の最小コスト実行順序を，手続き KBZ を用いて決定する．その際，マクロ演算は *OPT* のアルゴリズム中にあるような単位コストと結合選択率の決定により，コスト的にはある仮想的リレーション $R(C_j)$ と R_i との結合とみなすことができる．手続き *KBZ* は**図8**に示すとおりで，KBZ 法の手続き KBZ_1 に対応する．*OPT* の最後の部分では，処理木 P で $R(C_j)$ との結合と表現されている部分を，もともとの埋込み $R_i \varepsilon R_j$ および P_j の演算列に展開して，完全な LD-LPT を得る．*SUBOPT* の処理は *OPT* と本質的には変わらないが，導出する LD-LPT の先頭リレーションが初めから R に限定されている点が異なる．

例として，図3の問い合わせグラフ Q に手続き *OPT* が適用される場合の様子を示す．*OPT* では，ルート結合クラスタ C_1 の子結合クラスタである C_2 と C_3 に対して，それぞれ $SUBOPT(C_2, R_4)$ と $SUBOPT(C_3, R_7)$ を呼び出す．$SUBOPT(C_2, R_4)$ では，C_2 に子結合クラスタが存在しないので，**図9**(a)の Q_1 に対して $KBZ(Q_1, R_4)$ を呼び出す．この結果得られる R_4 を先頭リレーションとする LD-LPT を，仮に $R_4R_5R_6$ とする．$SUBOPT(C_3, R_7)$ では，C_3 が子結合クラスタ C_4 を持っているため，まず $SUBOPT(C_4, R_9)$ を実

```
Procedure SUBOPT
Input:  Q 中の結合クラスタ: C
        C 中の基本リレーション: R
Output: 部分問い合わせグラフ Q(C) に対する R を先頭リレーションとする
        最小コストの LD-LPT: P
begin
    Q' ← C 中の全ての基本リレーションと結合辺からなる問い合わせグラフ;
    for each (C の子結合クラスタ: C_j)
        /* C と C_j の間の埋込み辺を R_i εR_j とする */
        begin
            P_j ← SUBOPT(C_j, R_j);
            リレーション R(C_j) と結合辺 R_i ⋈ R(C_j) を Q' に追加.
            ただし, R(C_j) の単位コスト CJ(R(C_j)) と結合辺
            R_i ⋈ R(C_j) の選択率 SJ は下記とする:
                CJ(R(C_j)) = γ(R_j) + SE_ij * Cost(P_j)
                /* SE_ij は埋込み R_i εR_j の選択率,
                Cost(P_j) は処理木 P_j のコスト */
                SJ = 1/γ(R(C_j));
        end
    P ← KBZ(Q', R);
    for each (R(C_j))
        P 中の R(C_j) との結合を R_i ε R_j および P_j に展開する;
    P を返す;
end
```

図7 手続き SUBOPT
Fig. 7 Procedure SUBOPT.

```
Procedure KBZ
Input:  1つの結合クラスタのみからなる問い合わせグラフ: Q
        Q 中の基本リレーション: R
Output: Q に対する R を先頭リレーションとする最小コストの LD-LPT: P
begin
    for each (Q 中の R 以外のリレーション: R_i)
        ランク (γ(R_i) * JS_i − 1)/CJ(R_i) を計算;
        /*JS_i は R をルートとした結合木における R_i とその親リレーションと
        の結合選択率, CJ(R_i) は R_i の単位コスト */
    P ← ランクに基づき導出される, R を先頭リレーションとする最小コストの
        LD-LPT;
    P を返す;
end
```

図8 手続き KBZ
Fig. 8 Procedure KBZ.

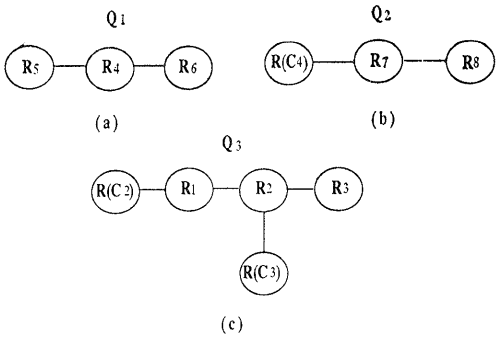

図 9 中間的な問い合わせグラフ
Fig. 9 Intermediate query graphs.

行する．$SUBOPT(C_4, R_9)$ の唯一の解は R_9R_{10} である．$SUBOPT(C_3, R_7)$ では，図9(b)の Q_2 に対して $KBZ(Q_2, R_7)$ を実行する．Q_2 において $R(C_4)$ は，上で述べた仮想的リレーションである．$KBZ(Q_2, R_7)$ の結果を仮に $R_7R(C_4)R_8$ とする．すると，$SUBOPT(C_3, R_7)$ では $R(C_4)$ のところを対応する演算列に展開して，LD-LPT $R_7R_9R_{10}R_8$ を OPT に返す．最後に OPT では，図9(c)の Q_3に対して $KBZ(Q_3, R_1)$, $KBZ(Q_3, R_2)$, $KBZ(Q_3, R_3)$ を実行する．ここで仮に，$KBZ(Q_3, R_1)$ が LD-LPT $R_1R(C_2)R_2R_3R(C_3)$ を返し，これが $KBZ(Q_3, R_2)$, $KBZ(Q_3, R_3)$ の返す LD-LPT よりも低コストであるとする．この時，OPT は次の LD-LPT を最終的な解として導出する：

$R_1R_4R_5R_6R_2R_3R_7R_9R_{10}R_8$.

3.4 アルゴリズムの基本的性質

以下では，上記アルゴリズムの基本的性質を示す．まず，本アルゴリズムが3.1節の仮定の下で最小コストの LD-LPT を導出することを示し，次に，本アルゴリズムの計算量を示す．なお，紙面の都合上以下の各定理の証明の詳細は文献 14) に示す．

定理1：手続き OPT は最小コストの DFE-LD-LPT を導出する．

[証明の概略] 手続き OPT が導出する処理木は DFE-LD-LPT であることは，その手順より明らかである．また，手続き OPT および $SUBOPT$ で用いる手続き KBZ が，3.2 節の仮定と同様の仮定の下で各（部分）問い合わせグラフに対する最小コストの DFE-LD-LPT を導出することより，本定理は成立する．■

定理2：与えられた問い合わせグラフに対する全 LD-LPT の中に，コスト最小の DFE-LD-LPT が必ず存在する．

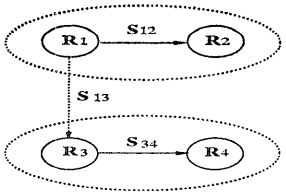

図 10 問い合わせグラフの例
Fig. 10 Sample query graph.

[証明の概略] 図 10 の問い合わせグラフを考える．この時，S_{12} が R_1 のルート属性である場合の結合 $R_1 \bowtie[S_{12}]R_2$ の対称性を除けば，以下の3種類の LD-LPT が存在し，このうち P_{1234} および P_{1342} が DFE-LD-LPT である．

$P_{1234}=R_1R_2R_3R_4$

$P_{1324}=R_1R_3R_2R_4$

$P_{1342}=R_1R_3R_4R_2$.

この時，（部分）組数や選択率等のパラメタの値に関わらず，

$Cost(P_{1234}) \leqq Cost(P_{1324}) \leqq Cost(P_{1342})$

または

$Cost(P_{1234}) > Cost(P_{1324}) > Cost(P_{1342})$

が常に成立することを示すことができる．すなわち，図 10 の問い合わせグラフに対して本定理は成立する．問い合わせグラフの他のいくつかの基本パターンについて同様の考察を行うことで，本定理は証明される．■

上記の定理より，本アルゴリズムの導出する最小コストの DFE-LD-LPT は，LD-LPT 全体の中でも最小コストであることが言える．

本アルゴリズムの計算量に関しては，以下の考察が成り立つ．すなわち，手続き OPT ではルート結合クラスタの各子結合クラスタ C_j に対する手続き $SUBOPT$ の実行と手続き KBZ の実行を行う．ここで，結合クラスタ C_j をルート結合クラスタとする部分問い合わせグラフ $Q(C_j)$ 中の基本リレーション数を N_j とする．3.3.1 の手続き KBZの計算量に関する議論より，OPT から起動された各 $SUBOPT$ の計算量は $O(N_j \log N_j)$ となる．一方，ルート結合クラスタ中の基本リレーション数を N_R とし，ルート結合クラスタの子結合クラスタの数を N_c とする．この時，OPT から直接手続き KBZ を呼び出す部分では，N_R+N_c 個のリレーションからなる問い合わせに対して KBZ 法全体を適用した場合に相当する計算量が必要であり，3.3.1 の議論より $O((N_R+N_c)^2)$ とな

る．したがって，問い合わせグラフ中の全基本リレーション数を N とすると，本アルゴリズムの計算量は一般に $O(N^2)$ となる．

4. シミュレーション

3.4 節に述べたとおり，本アルゴリズムは 3.1 節の仮定の下で最小コストの LD-LPT を導出する．しかし，3.1 節の仮定 2 および 3 は，現実には厳密な意味で成立することはない．本章では，サンプルデータベースを対象にしたシミュレーション実験結果を示し，本アルゴリズムの有効性を検証する．

4.1 実験方法

シミュレーションは，**図 11** に示す 3 種類の問い合わせグラフに対して行う．これら 3 種類の問い合わせグラフは，いずれも 8 つの基本リレーション，4 つの結合クラスタ，4 つの結合，3 つの埋込みから構成される．図 11 (a)(b)(c)の順で，結合クラスタが縦長の配置から横広がりの配置となる．

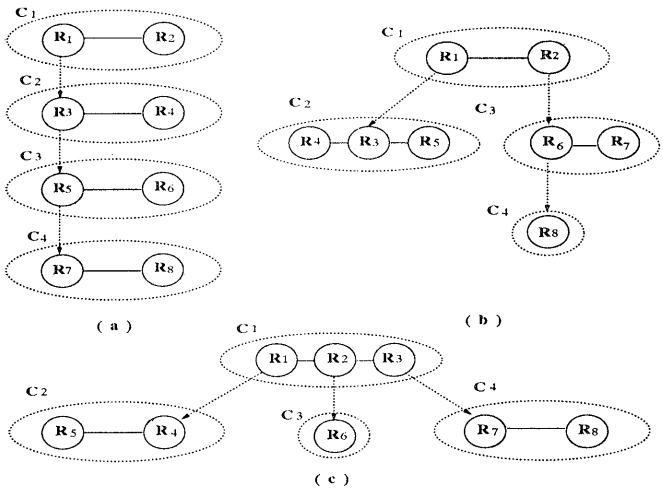

図 11 シミュレーション用問い合わせグラフ
Fig. 11 Query graphs for simulations.

$R_1(A_1, A_{12}, A_{13})$, $R_2(A_2, A_{12})$, $R_3(A_3, A_{13}, A_{34}, A_{35})$, $R_4(A_4, A_{34})$, $R_5(A_5, A_{35}, A_{56}, A_{57})$, $R_6(A_6, A_{56})$, $R_7(A_7, A_{57}, A_{78})$, $R_8(A_8, A_{78})$.
(a)

$R_1(A_1, A_{12}, A_{13})$, $R_2(A_2, A_{12}, A_{26})$, $R_3(A_3, A_{13}, A_{34}, A_{35})$, $R_4(A_4, A_{34})$, $R_5(A_5, A_{35})$, $R_6(A_6, A_{26}, A_{67}, A_{68})$, $R_7(A_7, A_{67})$, $R_8(A_8, A_{68})$.
(b)

$R_1(A_1, A_{12}, A_{14})$, $R_2(A_2, A_{12}, A_{23}, A_{26})$, $R_3(A_3, A_{23}, A_{37})$, $R_4(A_4, A_{14}, A_{45})$, $R_5(A_5, A_{45})$, $R_6(A_6, A_{26})$, $R_7(A_7, A_{37}, A_{78})$, $R_8(A_8, A_{78})$.
(c)

図 12 サンプルデータベースのスキーマ
Fig. 12 Schema of sample databases.

これら 3 種類の問い合わせのそれぞれの対象となるサンプルデータベーススキーマを**図 12** (a)(b)(c)に示す．これらサンプルデータベースは，いずれも 8 つのフラットリレーションから構成される．各リレーションのスキーマは，以下にしたがって設定されている．

1. 各リレーション R_i は主キー属性 A_i をもつ．
2. リレーション R_i と R_j $(i<j)$ の対が結合の対象となる場合，両リレーションは結合属性 A_{ij} をもつ．埋込みに対しても同様の属性を設ける．

1 回の試行においては，各リレーション R_i の組数 $\gamma(R_i)$ は 50〜300 の範囲の乱数として決定する．各組の主キー属性 A_i は，0 から $\gamma(R_i)-1$ までの整数を順次割り当てる．また，それ以外の属性 A_{ij} の値は 50〜1000 の範囲のランダムな整数を割り当てる．

シミュレーションでは，3 種類の問い合わせグラフのそれぞれに対して，以下の 1)〜6) を 100 回ずつ繰り返して実行する．

1) 上記の方法でサンプルデータベースを生成する．

2) サンプルデータベースに対して，本アルゴリズムの適用に必要な，各基本リレーションの組数，結合選択率，埋込み選択率を計算する．

3) 2)のパラメタを用いて本アルゴリズムを実行し，LD-LPT（以下 P と記す）を求める．

4) 比較対象を LD-LPT 全体とし，以下を行う．

(1) P の実コスト $Cost(P)$ が LD-LPT 全体の中で真に最小かどうか判定する．

(2) 指標 $R_{min}=Cost$（最小コストの LD-LPT)$/Cost(P)$ を求める．

(3) 指標 $R_{max}=Cost$（最大コストの LD-LPT)$/Cost(P)$ を求める．

(4) 指標 $R_{ave}=$全 LD-LPT の平均コスト$/Cost(P)$ を求める．

5) 比較対象を LPT 全体に広べて，4)と同様の実験を行う．

6) 指標 RT$=Cost(P)/P$ の論理コストを求める．

ここで，実コストとはサンプルデータベースに対して該当する処理木を実行した時の実際の（部分）組比較回数の総数であり，論理コストとは3.2節のコストモデルに基づく見積り値である．

4.2 実験結果と評価

1. LD-LPT 全体における比較評価

上記実験手続きの 4)(1)～(4)に関する実験結果をまとめたのが，**表1～4** である．表1では，各問い合わせグラフに対する100回の試行のうち，実コストが真に最小コストの LD-LPT を本アルゴリズムが導出した回数が，成功回数欄に示してある．また，各問い合わせグラフに対する LD-LPT の総数および DFE-LD-LPT の総数も併せて示してある．本アルゴリズムは，各問い合わせグラフに対して 81%～93% の割合で最小コストの LD-LPT を導出しており，その平均は 87% である．

表2には，各問いわせグラフに対する100回の試行における指標 R_{min} の分布および平均値を示す．$R_{min}=1.00$ は，本アルゴリズムが最小コストの LD-LPT を導出した場合であり，R_{min} の値が小さいほどより不適切な LD-LPT を導出したことになる．本アルゴリズムの導出する LD-LPT のコストと真の最小コストとの差は，3種類の問い合わせグラフに対する平均では 0.3% という値となっている．

R_{max} に関する結果を示したのが表3である．いずれの問い合わせグラフに対しても，処理木の選択により大きく処理コストが異なるケースが多く発生することが分かる．特に，(a)や(c)においては，本アルゴリズムが導出した LD-LPT の5倍以上の最悪コストの処理木が存在する場合が少なからずあることが分かる．

表4は R_{ave} に関する結果のまとめである．R_{ave} は，最適化をせずに任意に1つの LD-LPT を選んだ場合のコストと，本アルゴリズムを用いた場合のコストの比ととらえることができる．これらより，本アルゴリズムによる最適化を行うことにより各問い合わせグラフに対する平均で 28%～45%，全体での平均では36%のコスト削減が図れていることが分かった．

2. LPT 全体における比較評価

同様に，上記実験手続きの 5)(1)～(4)に関する実験結果をまとめたのが，**表5～8** である．比較対象が LPT 全体に広がり，その中により低コストの処理木が存在する可能性が増加する．このため，表5に示すように，表1と比べて本アルゴリズムが導出した処理木が真に最小コストのものとなる割合は低下する．最小コストの処理木を導出する割合は各問い合わせグラフに対して 64～88% であり，その平均は 72% となる．最小コストの処理木を導出するのに失敗したケースでは，表6に示すように導出した処理木のコストの 30% 以上低コストの最小コストの処理木が存在

表1 LD-LPT 空間
Table 1 Case of LD-LPT space.

問い合わせグラフ	LD-LPT の総数	DFE-LD-LPT の総数	成功回数
(a)	120	12	81
(b)	280	24	91
(c)	308	60	93

表2 LD-LPT 空間における R_{min}
Table 2 R_{min} in case of LD-LPT space.

問い合わせグラフ	(a)	(b)	(c)	計
$R_{min}=1.00$	81	91	93	265
$0.90 \leqq R_{min} < 1.00$	18	9	7	34
$R_{min} < 0.90$	1	0	0	1
R_{min} の平均値	0.994	0.999	0.999	0.997

表3 LD-LPT 空間における R_{max}
Table 3 R_{max} in case of LD-LPT space.

問い合わせグラフ	(a)	(b)	(c)	計
$1.00 \leqq R_{max} < 3.00$	82	94	61	237
$3.00 \leqq R_{max} < 5.00$	9	6	31	46
$5.00 \leqq R_{max} < 7.00$	5	0	7	12
$7.00 \leqq R_{max}$	4	0	1	5
R_{max} の平均値	2.66	2.02	3.00	2.56

表4 LD-LPT 空間における R_{ave}
Table 4 R_{ave} in case of LD-LPT space.

問い合わせグラフ	(a)	(b)	(c)	計
$1.00 \leqq R_{ave} < 2.00$	72	100	91	263
$2.00 \leqq R_{ave} < 3.00$	23	0	9	32
$3.00 \leqq R_{ave} < 4.00$	3	0	0	3
$4.00 \leqq R_{ave}$	2	0	0	2
R_{ave} の平均値	1.78	1.24	1.61	1.54

表5 LPT 空間
Table 5 Case of LPT space.

問い合わせグラフ	LPT の総数	DFE-LPT の総数	成功回数
(a)	960	12	65
(b)	496	24	64
(c)	392	60	88

表 6 LPT 空間における R_{min}
Table 6 R_{min} in case of LPT space.

問い合わせグラフ	(a)	(b)	(c)	計
$R_{min}=1.00$	65	64	88	217
$0.90 \leqq R_{min} < 1.00$	20	19	9	48
$0.80 \leqq R_{min} < 0.90$	9	8	3	20
$0.70 \leqq R_{min} < 0.80$	3	4	0	7
$R_{min} < 0.70$	3	5	0	8
R_{min} の平均値	0.964	0.952	0.993	0.970

表 7 LPT 空間における R_{max}
Table 7 R_{max} in case of LPT space.

問い合わせグラフ	(a)	(b)	(c)	計
$1.00 \leqq R_{max} < 3.00$	25	78	51	154
$3.00 \leqq R_{max} < 5.00$	44	20	40	104
$5.00 \leqq R_{max} < 7.00$	19	2	8	29
$7.00 \leqq R_{max}$	12	0	1	13
R_{max} の平均値	4.73	2.47	3.18	3.46

表 8 LPT 空間における R_{ave}
Table 8 R_{ave} in case of LPT space.

問い合わせグラフ	(a)	(b)	(c)	計
$1.00 \leqq R_{ave} < 2.00$	33	100	87	220
$2.00 \leqq R_{ave} < 3.00$	34	0	13	47
$3.00 \leqq R_{ave} < 4.00$	19	0	0	19
$4.00 \leqq R_{ave} < 5.00$	10	0	0	10
$5.00 \leqq R_{ave}$	4	0	0	4
R_{ave} の平均値	2.68	1.37	1.57	1.87

表 9 RT
Table 9 RT.

問い合わせグラフ	RT の平均値
(a)	1.006
(b)	0.997
(c)	0.998
平均	1.000

するケースも発生しているが，平均的にはコスト差は3% 程度となっている．表7に示すように，LPT 全体ではその中に極めて高コストの処理木が含まれる可能性も増加する．表8に示すように，比較対象を LPT 全体にした場合では，本アルゴリズムによる最適化を行うことにより各問い合わせグラフに対する平均で 31%～63%，全体での平均では 44% のコスト削減が図れていることが分かった．

3. コストモデルの評価

上記実験手続きの 6) で求める指標 RT の各問い合わせグラフごとの平均値を示したのが，**表9**である．表9より，本アルゴリズムが導出した処理木の処理コストの見積り値は，実コストとかなりの精度で一致していることが分かる．

5. 適用範囲の拡張に関する検討

本章では，本最適化アルゴリズムの適用範囲の拡張に関する検討を行う．

1. 結合アルゴリズムの一般化

上記の議論では，結合アルゴリズムとして入れ子ループ結合と入れ子ループ埋込みのみを対象としてきた．しかし，本アルゴリズムは一般に，3.2 節で述べた結合コストおよび埋込みコストが，それぞれ

$$Cost(R_i \bowtie [S_k] R_j) = \gamma(R_i, S_k) * CJ(R_j)$$

および

$$Cost(R_i \varepsilon [S_k] R_j) = \gamma(R_i, S_k) * CE(R_j)$$

で表される場合に拡張して適用可能である．ただし，$CJ(R_j)$ および $CE(R_j)$ は結合および埋込みの単位コストである．したがって，コストがこの形式で記述可能なハッシングや索引をベースとしたアルゴリズムに対しては，本アルゴリズムは適用可能である．また，ソートマージ結合も近似的にはこの形式でコスト見積りが可能であることが示されており[28]，近似解を求める範囲内では本アルゴリズグは適用可能である．

2. ディスクベースモデルへの拡張

本論文におけるコストモデルでは，データベースは主記憶上に存在するものとし，組の比較回数をコストとした．データベースがディスク上に存在する場合には，通常，ディスクブロックのアクセス回数が主要コストとなる．この場合，固定長レコードから成るヒープファイルを格納に用いた場合のように，近似的には組数と格納ブロック数が比例するような状況下では，1で述べたようなより一般化した形式でのコスト式が導出可能であり，本アルゴリズムは適用可能となる．しかし，入れ子型リレーションの物理的格納方式としてはこれまで数種類の方法が提案されており[19]，そのうちいくつかについてはこのような性質は満たされない．したがって，そのような場合の本アルゴリズムの適用可能性については，別途より詳細な検討が必要である．またディスクベースモデルにおいては，バッファサイズとその管理方式がブロックアクセス数に大きく影響してくるので，これを考慮に入れたコストモデルの構築が必要となる．

3. 結合/埋込み以外の演算子への対応

入れ子型リレーショナル代数には，結合，埋込み以

外に各種の演算子が含まれる．これらのうち，選択(selection)，射影（projection）等は，可能な限り結合演算に先行して実行するという戦略が，リレーショナルデータベースにおいてはとられることが多い．同様の戦略を入れ子型リレーショナルデータベースにおいて用いた場合，これらの各基本リレーションにローカルな演算子を実行した後の結合/埋込みの適用順序の決定には，本アルゴリズムは適用可能である．しかし，リレーションに比べて複雑な内部構造をもつ入れ子型リレーションにおいては，例えば，選択演算の選択条件の評価に要するコストが無視できない大きな値になることも考えられる．筆者らは，抽象データ型（ADT）を導入したリレーショナルデータベースを対象に，このような高コストの選択演算と結合演算を含む問い合わせの最適化アルゴリズムを提案している[30]．したがって，選択演算子を始めとするいくつかの基本演算子については，本アルゴリズムを拡張してその枠組みの中に取り込むことが可能であると思われる．

4. 問い合わせグラフの制約の緩和

本アルゴリズムが対象とする問い合わせグラフは，2.4 節で示した3つの制約を満たすものである．このうち線形処理木のみを導出対象とした場合に本質的な制約となるのは，制約条件1および3である．制約条件1に関しては，本アルゴリズムのコスト式の扱いを拡張することにより，より弱い制約に緩和することが可能であるとの見通しを得ている．一方，制約条件3に関しては，KBZ 法において，閉路を含む問い合わせグラフを処理するためのヒューリスティックスが提案されている[13]．これは，結合選択率に基づく最小極大木（minimal spanning tree）からなる木構造問い合わせグラフとみなして，処理木を導出するというものである．制約条件3を除去した場合，本アルゴリズムに同様のヒューリスティックスを導入することは容易であるが，その有効性に関してはシミュレーション等による評価が必要である．

5. オブジェクト指向データベースへの転用

オブジェクト指向データベースにおいても，複合オブジェクトの表現上の構造として，しばしば入れ子型リレーションに類似したデータ構造が用いられている．また，オブジェクト指向データベースを対象とした結合演算を含む代数の提案[25]や問い合わせ処理最適化法の提案[15]も行われている．本アルゴリズムは，オブジェクト指向データベースにおける結合問い合わせ処理を検討する上でも，1つの枠組みを提供するものと考えられる．その場合，ポインタに基づく結合と述語に基づく結合をコスト的に区別する必要がある[15]点や，本章の3で述べたようにメソッド呼び出しを含む述語の評価コストが高コストとなり得る点等を考慮して，検討を行うことが必要である．

6. おわりに

本論文では，入れ子型リレーションに対する結合演算問い合わせの最適化アルゴリズムを提示し，その有効性の評価を行った．本アルゴリズムは，具体的には結合，埋込みの2種類の演算を対象とし，2.4 節で述べた問い合わせグラフに対して，左連鎖線形処理木(LD-LPT) を導出する．3.1 節の仮定の下で，本アルゴリズムは最小コストの LD-LPT を導出する．また，上記の仮定が厳密には成立しないサンプルデータベースを対象にしたシミュレーション実験による評価を行った．シミュレーション結果によれば，本アルゴリズムは平均 87% の割合で最小コストの LD-LPT を導出し，最適化によるコスト削減は平均 36% であった．また，処理木の候補を LPT 全体に拡大した場合は，本アルゴリズムが最小コストの処理木を導出する割合は平均 72% に低下するものの，最適化によるコスト削減は平均 44% との値を得た．これらの結果より，本アルゴリズムの有効性の確認と，そのコスト削減率の定量的な把握を行うことができた．さらにまた，本アルゴリズムの適用範囲の拡張の可能性に関する検討を行った．

今後の課題としては，出現データ値の種類や組数に大きな偏りがあるような場合における本アルゴリズムの有効性評価がある．また，第5章で述べた各種拡張の可能性とその場合の有効性について，より詳細な検討が必要である．

謝辞 本稿に対して有益なコメントをいただいた査読者の方々に感謝いたします．また，本研究も含め種々の研究課題に関して熱心に御討論いただいている，筑波大学電子・情報工学系 藤原 譲教授，鈴木 功教授，ならびに大保 信夫助教授に御礼申し上げます．

参考文献

1) Abiteboul, S. and Bidoit, N.: Non First Normal Form Relations to Represent Hierachically Organized Data, *Proc. 3rd ACM PODS*, pp. 191-200 (1984).
2) Abiteboul, S., Fischer, P. C. and Schek, H. J.

(eds.) : *Nested Relations and Complex Objects in Databases*, Lecture Notes in Computer Science 361, Springer-Verlag (1989).
3) Colby, L. S. : A Recursive Algebra and Query Optimization for Nested Relations, *Proc. ACM SIGMOD Conf.*, Portland, pp. 273-283 (1989).
4) Dadam, P., et al. : A DBMS Prototype to Support Extended NF^2 Relations : An Integrated View on Flat Tables and Hierarchies, *Proc. ACM SIGMOD Conf.*, Washington, D. C., pp. 356-367 (1986).
5) Deshpande, A. and Van Gucht, D. : An Implementation for Nested Relational Databases *Proc. 14th VLDB Conf.*, Los Angeles, pp. 76-87 (1988).
6) Deshpande, V. and Larson, P. A. : The Design and Implementation of a Parallel Join Algorithm for Nested Relation on Shared-Memory Multiprocessors, *Proc. 8th International Conf. on Data Engineering*, Phoenix, pp. 68-77 (1992).
7) Fischer, P. C. and Thomas, S. J. : Operators for Non-First-Normal-Form Relations, *Proc. IEEE COMPSAC 83*, Chicago, pp. 464-475 (1983).
8) Gyssens, M. and Van Gucht, D. : The Powerset Algebra as a Result of Adding Programming Constructs to the Nested Relational Algebra, *Proc. ACM SIGMOD Conf.*, Chicago, pp. 225-232 (1988).
9) Haskin, R. L. and Lorie, R. A. : On Extending the Functions of a Relational Database System, *Proc. ACM SIGMOD Conf.*, pp. 207-212 (1982).
10) Ibaraki, T. and Kameda, T. : On the Optimal Nesting Order for Computing N-Relational Joins, *ACM Trans. Database Syst.*, Vol. 9, No. 3, pp. 482-502 (1984).
11) Jaeschke, G. and Schek, H. J. : Remarks on the Algebra of Non First Normal Form Relations, *Proc. ACM PODS*, pp. 124-138 (1982).
12) Kitagawa, H. and Kunii, T. L. : *The Unnormalized Relational Data Model--For Office Form Processor Design—*, Springer-Verlag (1989).
13) Krishnamurthy, R., Boral, H. and Zaniolo, C. : Optimization of Nonrecursive Queries, *Proc. 12th VLDB Conf.*, Kyoto, pp. 128-137 (1986).
14) Li, Y., Kitagawa, H. and Ohbo, N. : Optimization of Join-Type Queries in Nested Relational Databases, Technical Roport, IISE, Univ. of Tsukuba (1994).
15) Lanzelotte, R., et al. : Optimization of Nonrecursive Queries in OODBs, *Proc. 2nd DOOD*, pp. 1-21 (1991).
16) Lorie, R. and Plouffe, W. : Complex Objects and Their Use in Design Transactions, *Proc. ACM SIGMOD Conf.*, pp. 115-121 (1983).
17) Makinouchi, A. : A Consideration on Normal Form of Not-Necessarily-Normalized Relation in the Relational Data Model, *Proc. 3rd VLDB Conf.*, Tokyo, pp. 447-453 (1977).
18) 三浦, 有澤 : 非正規関係データベース, コンピュータソフトウェア, Vol. 7, No. 1, pp. 72-80 (1990).
19) Ozsoyoglu, Z. M. (ed.) : *Special Issue on Nested Relations, IEEE Data Engineering*, Vol. 11, No. 3 (1988).
20) Roth, M. A., Korth, H. F. and Silberschatz, A. : Extended Algebra and Calculus for Nested Relational Databasess, *ACM Trans. Database Syst.*, Vol. 13, No. 4, pp. 389-417 (1988).
21) Schek, H. J. and Scholl, M. H. : The Relational Model with Relation-Valued Attributes, *Inf. Syst.*, Vol. 11, No. 2, pp. 137-147 (1986).
22) Scholl, M. H. : Thoretical Foundation of Algebraic Optimization Utilizing Unnormalized Relations, Technical Report, Technical Univ. of Darmstadt (1986).
23) Scholl, M. H., Paul, H. B. and Schek, H. J. : Supporting Flat Relations by a Nested Relational Kernel, *Proc. 13th VLDB Conf.*, Brighton, pp. 137-146 (1987).
24) Selinger, P. G., et al. : Access Path Selection in a Relational Database Management System, *Proc. ACM SIGMOD Conf.*, pp. 23-34 (1979).
25) Shaw, G. M. and Zdonik, S. B. : A Query Algebra for Object-Oriented Databases, *Proc. 6th International Conf. on Data Engineering*, Los Angeles, pp. 154-162 (1990).
26) Swami, A. and Gupta, A. : Optimazation of Large Join Queries, *Proc. ACM SIGMOD Conf.*, Chicago, pp. 8-17 (1988).
27) Swami, A. : Optimization of Large Join Queries : Combining Heuristics and Combinatorial Techniques, *Proc. ACM SIGMOD Conf.*, Portland, pp. 367-376 (1989).
28) Swami, A. and Iyer, B. R. : A Polynomial Time Algorithm for Optimizing Join Queries, *Proc. 9th International Conf. on Data Engineering*, Vienna, pp. 345-354 (1993).
29) The Committee for Advanced DBMS Function : The Third-Generation Database System Manifests, *ACM SIGMOD Record*, Vol. 19, No. 3, pp. 31-44 (1990).
30) Yajima, K., et al. : Optimization of Queries Including ADT Functions, *Proc. 2nd DASFAA*, Tokyo, pp. 366-373 (1991).

(平成5年11月11日受付)
(平成6年 6月20日採録)

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 年同大学院理学系研究科修士課程修了．日本電気(株)勤務を経て，1988 年筑波大学電子・情報工学系講師．現在，同学系助教授．理学博士（東京大学）．データベースシステム構成法，時間データ管理，エンジニアリングデータ管理，ソフトウェア開発支援システムなどに興味を持つ．著書「The Unnormalized Relational Data Model」（共著，Springer-Verlag）．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．

李 亜新（正会員）

1960 年生．1982 年中国国立ハルビン工業大学計算機科学系卒業．1985 年同大学同専攻修士課程修了．現在筑波大学工学研究科博士課程在学中．研究テーマ：データベースシステム．