データベース・システム　72−13
（1989. 7. 21）

演繹データベースの再帰問合せ評価におけるページアクセスレベル最適化

Page Access Level Optimization for Recursive Query Evaluation in Deductive Databases

楠見　雄規
Yuki KUSUMI
松下電器産業株式会社
Matsushita Elect. Industrial Co., Ltd.

西尾　章治郎
Shojiro NISHIO
大阪大学基礎工学部情報工学科
Dept. of Info. & Comp. Sci., Osaka Univ.

長谷川　利治
Toshiharu HASEGAWA
京都大学工学部数理工学教室
Dept. of Appl. Math. & Phy., Kyoto Univ.

あらまし　演繹データベースの再帰問合せの評価に関して，従来の多くの研究がデータベースの論理的構造に関して行なわれてきたが，最近，物理的なアクセスレベルでの評価コストを減少させる一般的手段に関する研究が盛んになりつつある．そこで本稿では，再帰問合せの問題においてしばしば評価の対象となる推移的閉包の計算について，ファイルアクセスレベルでの効率の良い評価プランを立案する手法について検討する．特に，I/Oコストを物理的な入出力の単位である**ページI/O量**で測ることにし，最適な評価プランを求めるの手段として，ページレベルでの導出パスを記述する**ページアクセスグラフ**を導入する．結果として，このグラフの全体あるいは一部をメモリ上で処理できるという仮定のもとで，あるクラスのページアクセスグラフに対しては，最適な評価プランを立てることが可能であることを示す．

Abstract　In this paper, we study on file access strategies for the transitive closure computation which is a bottleneck of recursive query evaluation in deductive database systems. While most of researches in this field have focused their attention on the optimization with respect to logical structure of database, we will discuss the general strategy to reduce the cost for query evaluation in practical systems. The cost for evaluation is measured by means of page I/O, and the *page access graph* is introduced to obtain efficient access strategies. This graph represents derivation paths connecting pages in the database file for which a transitive closure is computed. Under the assumption that whole or a part of this graph is processed in the main memory, our algorithm provides the optimal file access scheduling for a certain class of page access graph.

1　はじめに

近年，**演繹データベース (DDBS)** に関しては，再帰的に定義された述語に対する問合せ評価のアルゴリズム [4] に関して活発に研究が行なわれている．しかし，これらの成果を実システム上に適用するに当たって，次に挙げる三つの観点からアルゴリズムを再検討する必要がある．

第一に，実際のファイルアクセスを考慮すると，アルゴリズムの性能は物理的なファイルアクセスの単位，すなわち，ページ(あるいはセグメント)を単位としてI/Oコストを測る必要がある．いくつかの文献(例えば [1][6])では，再帰問合せ評価アルゴリズムの性能を物理的なファイルアクセスに基づいて測っており，一旦読み込まれたページ内の全ての情報を利用するアルゴリズムも提案されている．しかし，それらは最小のページアクセス回数を得るためのアクセススケジュールを与えるものではない．

次に，与えられた問合せが束縛されているか否かに依存しない，より汎用の問合せ評価技法を開発する必要がある．従来，問合せの引数に最低1個の定数をもつ**束縛問合せ**の評価アルゴリズムと，引数に定数が現われない**非束縛問合せ**の評価アルゴリズムは別々に研究されてきた．しかし，演繹データベースにおけるファイルアクセスエンジンの構築を考えれば，それらに共通して適用できる手法を考案する必要がある．

第三の再検討事項として，充分な主記憶が用意されているようなシステムでは，関係ファイルや中間関係の全体をロードできない場合においても，ファイルの一部に対するキャッシングが性能改善のための強力な手法となり得る．したがって，実際のメモリが有限であることを考慮すれば，効率的なキャッシング手法の開発が非常に重要である．

以上の検討事項を念頭におきながら実システムにおける再帰問合せ評価アルゴリズムの効率的な適用を考慮し，本稿では，再帰問合せ評価にしばしば現われる推移的閉包の計算の性能向上に効果的なページアクセススケジューリングの手法を提案する．本稿では，推移的閉包の計算についてのみ議論を行うが，ここで提案する手法は，与えられた問題に対して適当な解析を行うことによって，より複雑な再帰構造をもつ問題に対しても適用可能であると考えられる．

以下，第2章で従来の研究の成果について簡単に言及した後，第3章では，与えられたファイルのページ構造を記述するための**ページアクセスグラフ**を導入する．このグラフは，第4章で，再帰問合せ評価における最適なページアクセススケジュールを考察するために非常に有用である．第5章以下では，実際の演繹データベースへの適用を考慮しながら，提案したアルゴリズムの特性を議論する．最後に，第8章で，本稿の考察結論を述べるとともに，今後の研究に対する問題を提起する．

2　過去の研究成果の概略

本稿においても従来の研究同様，演繹データベースは，**事実**の集合を格納する**外延データベース (EDB)** と，**導出ルール**の集合を格納する**内包データベース (IDB)** から構成されると仮定する．また，以下ではProlog流のルールの表記を用いる．このとき，事実は変数を含まない基礎節(即ち本体が空である節)で表される．また，EDB節とIDB節の両方の頭部に現れる述語は，適当な手続きによって取り除くことができるので，データベース中に現れる述語は**EDB述語**と**IDB述語**に分けることができる．特に，EDB述語は関係データベースにおける**関係**に対応する．したがって，以下では，EDB述語に対する事実の集合は**EDB関係**として格納されると仮定する．

上記のように定義される演繹データベースの特色は，柔軟な問合せ記述が可能になる点である．特に，通常の関係データベースシステムと比較して最も大きな特色は，再帰的に定義された述語に対する問合せが可能となる点である．

例1　EDBに与えられた親子関係に対して先祖関係を求めるため

のルール集合(即ち，**論理プログラム**)を示す．これは，**推移的閉包ルール**の典型的な例である．

$$anc(X,Y) :- par(X,Y).$$
$$anc(X,Y) :- anc(X,Z), par(Z,Y). \quad (1)$$

ただし，$par(X,Y)$ はEDB関係であり，「YがXの親である」ことを表す．また，$anc(X,Y)$ は，「YがXの先祖である」という意味である． □

このようなルールで定義された述語に対する問合せは，束縛問合せと非束縛問合せの二つのクラスに分類することができる．例えば，例1のルールに対し，a という特定の個人の先祖を求める問合せ，$?- anc(a,X)$ は**束縛問合せ**であり，引数に現れる a を**問合せ定数**とよぶ．これに対し，親子関係 parから導出される全ての先祖関係を求める問合せ，$?- anc(X,Y)$ は**非束縛問合せ**である．

束縛問合せ評価のアルゴリズムにおける最適化の目標は，問合せ定数と関連のない事実(つまり，組)に対するアクセスを避けることである．この目標を達成する手法としては，問合せ定数から始まるトップダウン評価を行う方法 [5][6][7][8] や，トップダウンの束縛伝幡に基づきEDB関係のなかで問合せ定数に関連しない部分を取り除く方法 [3] などがある．

非束縛問合せの場合，計算途中に繰り返し現れるの大きな結合演算のコストを軽減しこと，または，同じ組に対するアクセスの重複を避けることが最適化の目標となる．この目標を達成ために，今までに多くのアルゴリズムが提案されている [1][2][9][10][11]．ボトムアップ探索を基本とする多くの手法は，結合演算の対象となる関係の大きさを小さくするために差分関係を導入している．一方 [9] では，推移的閉包計算の対象となる関係中のサイクルの扱いに工夫を凝らした上で，トップダウン探索を用いている．また，[1] では，問題を与えられたEDB関係から構成されるグラフの隣接行列上での推移的閉包計算として解析し，関係に対する適当なクラスタリングと行列上での計算順序の工夫によって，I/O量を効率良く減ずる方法が提案されている．

本稿の目的は，これらの問題を統一的に取り扱うことのできる効率的なファイルアクセス手法を提案することにある．

3 ページアクセスグラフ

前述のように，再帰問合せ評価の最適化の目的は，EDB関係と，計算途中で生じる中間関係に対するファイルアクセスのコストを軽減することである．特に，EDBが関係データベースとして構築されている場合，関係中の組は，ファイル中ではいくつかのページに分割して格納される．このようなページが物理的なファイルアクセスの単位となるので，I/Oコストは，アクセスされた総ページ数で測る必要がある．このような，ページI/Oコストに着目した最適化を，特に組単位のI/Oコストに着目した最適化と区別するために，**ページアクセスレベルの最適化**とよぶ．多くの研究成果によれば，中間関係に対するI/Oコストを軽減する効果的な方法は，結合演算をはじめとする諸演算のオペランドの関係から，関連しない，あるいは重複する組を取り去ることである．さらに，EDB関係のアクセスに対してページアクセスレベルの最適化を考慮に入れることによって，現実的により効率の良いファイルアクセスプランを立てることができる．

例 2 例1の先祖ルール(1)について再び考える．EDB関係とそのデータベースグラフが，図1に示すように与えられているとする．ただし，データベースグラフは，関係中の組 (a,b) に対し，有向枝 (a,b) をもつようなグラフである．なお，このデータベースの例は，本稿の残りの部分で必要に応じて参照される．問合せ $?- anc(a,X)$ に対し，効率よく最適化された**幅優先探索(BFS)**は，次のようなステップで答えを見いだす．

1. P_1を読み込み，a の親 $\{b\}$ を見つける．

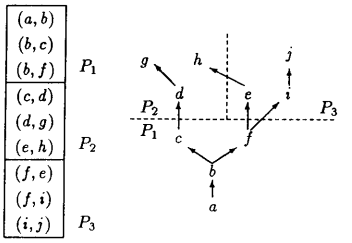


図 1: EDB関係の例と，対応するデータベースグラフ

2. P_1を読み込み，$\{c,f\}$ を見つける．
3. P_2と P_3を読み込み，$\{i,e,d\}$ を見つける．
4. P_2と P_3を読み込み，$\{g,h,j\}$ を見つける．

ここで，読み込みの際に使用可能なバッファが1ページ分であると仮定すれば，延べ4~5ページの読み込みが必要となる．しかし，これを次のように計算すれば3ページの読み込みで済む．

1. P_1を読み込み，$\{b,c,f\}$ を見つける．
2. P_3を読み込み，$\{e,i,j\}$ を見つける．
3. P_2を読み込み，$\{d,g,h\}$ を見つける．

ここでは，ステップ2で P_3を読み込んだが，先験的な情報が何もなければ，P_2と P_3のどちらを先に読むかは非決定的である． □

ここで，EDB関係に対するファイルアクセスの様子を形式化し，例2に見られるような非決定性を解消する上で役立つ，**ページアクセスグラフ**(以下，PAGと略)を導入する．

まず，EDB関係を次のようにモデル化する．

定義 1 $P_1, P_2, \ldots, P_n$をEDB関係 Pの任意の分割とする．このとき，各 P_iに含まれる組の数が高々 m 個であるとき，かつそのときのみ $P_1, P_2, \ldots, P_n$を**ページ**とよぶ． □

ページはファイルアクセスの単位となるのみならず，十分な主記憶容量が利用可能であるときにはキャッシングの単位ともなり得る．組指向で最適化されたアルゴリズムのほとんどは，さらに，ページアクセスレベルの最適化を施すことができる．次に，本稿の最適化手法の鍵となるページアクセスグラフを導入する．

定義 2 PをEDB関係，$P_1, P_2, \ldots, P_n$を P中のページとすると，Pの推移的閉包計算に対する**ページアクセスグラフ(PAG)** $G=(V,E)$ は，次のように定義される．

$$V = \{P_1, P_2, \ldots, P_n\}$$
$$E = \{(P_i, P_j) \mid \forall q \; s.t. \; tuple \; (p,q) \in P_i, \; (q,r) \in P_j, \; i \neq j\}$$

□

PAGは，関係に対するデータベースグラフを，第一属性の値から第2属性の値に向う有向枝で表現した際に，データベースグラフ中のパスが，ページ間でどの様につながっているかを表す．したがって，例1の先祖ルールで定義された述語に対し，問合せの第2引数が束縛されている場合は，上のように構築されたPAGの有向枝を逆向きに辿るか，あるいは，有向枝の向きを反対にする必要がある．非束縛問合せについては，明らかに枝の向きを問う必要はない．なお，1ページに一つの組が格納されるような特殊な状況では，PAGはジョインインデックス [12] と等価である．

次に，ページアクセススケジュールを考えるうえで有効な，PAG中の節点に対する番号付けの方法について説明する．

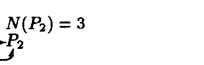


図 2: EDB の例に対する PAG と，各ノードの番号付け

定義 3 p_iと p_jを $PAG\ G$ の節点とする．このとき，**節点 p_iと p_jの距離**を，p_iから p_jに至る最長のパス長で定義し，$D(p_i, p_j)$ と表す．□

定義 4 $G = (V, E)$ をアサイクリックな PAG とし，$V = \{p_1, p_2, \ldots, p_n\}$ とすると，節点 p_iに対する**番号付け** $N(p_i)$ は次のように定義される．

1. p_iが先行する節点を持たないとき，$N(p_i) = 1$．
2. $(p_i, p_j) \in E$ならば，$N(p_j) = N(p_i) + D(p_i, p_j)$．

□

このような番号付けは，通常の**深さ優先探索 (DFS)** によって実行することができる．

例 3 図 1の EDB の例について，PAG とその節点の番号付けは図 2のようになる． □

PAG を効率的に生成することができ，かつその PAG の全体，ないし一部を格納するのに十分な主記憶上の領域があれば，問合せ評価において，EDB 関係のページアクセスのスケジュールを立てる場合に利用することができる．

最初に，PAG を構成する手続きについて考える．この手続きは一見，関係中の任意の二つの組の組合せを全て探索する必要があるように思えるが，通常，関係と共に管理されているインデックスや，あるいはハッシュテーブルを利用することで，効率的に PAG を構成することができる．

アルゴリズム 1 (PAG の構成)

入力　EDB 関係 P
　　Pの第 1 属性に関するインデックス
　　(ただし，各キーが現れるページが全て記録されているものとする)
出力　Pの推移的閉包計算のための PAG
1.　**for** Pの第 2 属性に現れる各キー $q \in P$ **do**
1.1　　インデックスを利用して，qが第 1 属性に現れるページ $P_{j_1}, \ldots, P_{j_s}$ を全て求める
1.2　　$E := E \cup \{(P_i, P_{j_1}), \ldots, (P_i, P_{j_s})\}$
　　endfor

□

ここで，このアルゴリズムの計算量を検討する．インデックスとして B-木を使うとすれば，ステップ 1.1 で訪問される節点の総数は，Nを関係中の総組数とすると，高々$N(1+\log(N+1)/2)$である．また，ステップ 1.2 で生成される有向枝の総数は最悪の場合，m を 1 ページ中の最大組数，n を関係中のページ数とすると，$\max(n^2, mn(n-1))$ になる．しかし，後に第 5章で述べるように，第 1 属性 (または，第 2 属性) に同じ値をもつ全ての組が，同じページに格納されるようにソート (つまり，クラスタリング) されているという仮定の下では，有向枝の総数は高々mn で抑えられる．さらに，PAG を EDB の更新に併せて管理すれば (その方法については自明であろう)，問合せが来たときに PAG を構成する必要がなくなり，PAG の管理に必要なオーバーヘッドは無視できるようになる．

次に，PAG の物理的な大きさについて検討しよう．PAG を関係の形で格納すると仮定すると，PAG の格納に必要な領域は次のように見積られる．

上記のクラスタリングの仮定のもとでは PAG 中の有向枝数は mn であり，この数は，各ページが最大限に利用されたときの関係中の組数 Nに他ならない．B_{tuple}を EDB 関係中の一つの組を格納するのに必要なバイト数，B_{arc}を PAG の有向枝を格納するのに必要なバイト数とすると，PAG 全体を格納するのに必要なページ数は nB_{arc}/B_{tuple}である．

例 4 $B_{tuple} = 128bytes$，$B_{arc} = 4bytes$ とし，1 ページを $8192bytes$ とすると，2000 組 (つまり，32 ページ) の関係に対し，PAG の格納に必要となる領域は高々 1 ページである． □

ここでは，PAG が関係の形で格納されると仮定したが，実システムではブール行列の形で取り扱う方が有利であると予想される．この問題に関するさらに詳細な議論は，第 5章で行う．

4 推移的閉包計算における PAG の役割

本稿の以下の部分では，PAG の全体ないし一部分を，必要に応じて主記憶にロードすることができるものと仮定する．この仮定のもとでは，EDB に対する最適なページアクセスプランを立てることができる．なお，従来の EDB アクセスに対する組単位の最適プランは，必ずしも実システムの I/O コストに対する最適プランになるとは限らないが，再帰問合せ評価において複雑になりがちな EDB 関係に対するアクセスを形式化することには意味があると考えられる．本稿で提案するアルゴリズムは，従来より提案されている高性能なアルゴリズムをベースに，さらに現実的なアクセス単位 (つまり，ページレベルでのアクセス) を考慮した最適化を目指している．

なお，本章では，2 項関係 $P(X, Y)$ に対して IDB ルール (1) で定義される推移的閉包計算のみを扱うが，文献 [6] における議論と同様，本稿の成果はより複雑なルールで定義された IDB 述語に対する問合せに対しても適用できる．

4.1 束縛問合せの評価における PAG の利用

マジック集合法 (Magic Set Method)，**数え上げ法 (Counting Method)** [3]，**Henshen-Naqvi のアルゴリズム** [8]，**δ-ウェーブフロントアルゴリズム (δ-Wavefront Algorithm)**，**レベルサイクルマージ法** [6][7] 等は，束縛問合せを評価するためのよく知られたアルゴリズムであるが，これらはページアクセスレベルの最適化を行うものではない．

文献 [6][7] では，推移的閉包問合せのように繰り返しのレベルが重要でない場合や，正確なレベル情報が記録されるようなアルゴリズムを用いる場合には，厳密な幅優先探索は不要であるとの考えから，**レベル緩和δ-ウェーブフロントアルゴリズム (Level Relaxed δ-Wavefront Algorithm)** が提案されている．ここでは，まず，元々のδ-ウェーブフロントアルゴリズムと，そのレベル緩和版を，それぞれ，アルゴリズム 2とアルゴリズム 3に挙げる．

アルゴリズム 2 (δ-ウェーブフロントアルゴリズム) [7]

入力　EDB 関係 $P(X, Y)$
　　IDB ルール (先祖ルール (1))
　　問合せ $? - A(a, X)$　(ただし，a は問合せ定数)
出力　問合せに対する答えの集合
1.　$WAVE := \{a\}$; $ANSWER := \phi$;
2.　**while** $WAVE \neq \phi$ **do**
2.1　　$WAVE := \pi_2(WAVE \ltimes P)$;
2.2　　$WAVE := WAVE - ANSWER$;
2.3　　$ANSWER := ANSWER \cup WAVE$;
　　endwhile.

注) (a) 単項関係 $WAVE$は，**ウェーブフロント集合**と呼ばれる．(b) $WAVE$中の各キーは，次の導出過程を導くことから，**ドライバ (driver)** と呼ばれる． □

このアルゴリズムは，まさに例2に示した効率良く最適化されたBFSそのものと考えることができる．

アルゴリズム 3 （レベル緩和δ-ウェーブフロントアルゴリズム）[7]文献[7]に示されるアルゴリズムとは異なり，ここでは簡単のため，EDBに対する1ページ分のみのバッファを仮定して，アルゴリズムを記述する．アルゴリズムに対する入出力は，アルゴリズム2と同様である．

1. $WAVE := \{a\}$; $ANSWER := \phi$;
2. **while** $WAVE$中のドライバから ドライブされるページが存在する **do**
2.1 ドライブ可能なページP_iを読み込む;
2.2 **while** $WAVE$中のドライバが主記憶上で 新たなキーを生成し得る **do**
2.2.1 $NEW\_ANS := \pi_2(WAVE \ltimes P_i)$;
2.2.2 $live\_driver := NEW\_ANS - ANSWER$;
2.2.3 $WAVE := WAVE \cup live\_drivers - drivers$;
2.2.4 $ANSWER := ANSWER \cup live\_drivers$;
endwhile
endwhile

注) (a) ステップ2.2.1は主記憶上で実現される．(b) ステップ2.2.3でアクセスされたドライバについては，同じ値のドライバは一つのページに格納される．すなわち，クラスタリングされているという仮定に基づき，$WAVE$から取り除かれる． □

このアルゴリズムは，例2の後半に示したような，巧妙なページアクセスによって答えの導出を行う．ところで，レベル緩和法は，δ-ウェーブフロントアルゴリズムに対する低レベルの最適化を実現しているが，例2と次に示す例を比較することにより，この方法には，まだ非決定性要素が存在することが分かる．したがって，レベル緩和法について，この非決定性を排除するためのさらなる最適化を行う必要がある．

例 5 先祖ルール(1)に対する問合せ? $-anc(a,X)$と，図1に示すEDBについて考える．例2で示したように，P_1内における導出が終了した直後の$WAVE$中のドライバは，$\{c,f\}$である．ここで，次に読み込むページとして，cに対するP_2を選択するとそれ以降の導出は次のように行われる．

1. P_2を読み込み，$\{d,g\}$を得る．
2. P_3を読み込み，$\{e,i,j\}$を得る．
3. P_2を読み込み，$\{d,g,h\}$を得る．

例2で示した最適な場合に読み込まれる延べページ数が3ページであるのに対し，P_3を選択した場合に，読み込まれる延べページ数は4ページになることが分かる． □

レベル緩和δ-ウェーブフロントアルゴリズムにおけるこの非決定性は，ページ間でのジョインセレクティヴィティに関する先験的な情報が与えられない限り，排除することは不可能である．しかし，あるクラスの問合せに対しては，PAGを利用してこの非決定性問題を解決することが可能である．

定理 1 二項関係Pのデータベースグラフに対して，問合せ定数aで始まる部分推移的閉包を求める束縛問合せを考える．もし，問合せ定数aが第1属性に現れるページから始まるPAGの強連結成分(**CCPAG**)にサイクルがなければ，高々1回，各ページを読み込むような最適なページアクセスプランが存在する．
証明: $CCPAG = (V_c, E_c)$がサイクルを含まないという仮定から，V_cの各ノードに定義4の番号付けを行うことが出来る．ここで，アクセスプランとして，アクセス要求されているページ(即ち，$WAVE$中のドライバに対して以後の導出が進み得るページ)のうち，最小の番号を持つものを読み込むようなプランを考えよう．このプランは，CCPAGにサイクルがないこと，および，PAGの定義から，現在アクセスしているページより小さい番号を持つページをアクセスすることはないことに注意する．結局，各ページは，PAG上の先行するページをすべて処理した後にアクセスされることになる．もしそうでなければ，小さな番号のページで未処理のものが生じることになり，矛盾を生じる．この議論は，導出の全ての段階について成り立つので，このプランによれば各ページが高々1回読み込まれることが分かる． □

定理1とその証明は，PAG上の各ノードをその番号順に読み込むプランは，EDBに対するページアクセスのコストを考える場合の，CCPAGにサイクルがないという条件の下での最適プランであることを意味する．しかし，一般的には，EDBに対するデータベースグラフにサイクルがない場合でも，PAGはサイクルを持ち得る．よって，このプランをより一般のPAGに対しても適用できるようにすることが重要である．

与えられたPAGにサイクルが存在する場合，CCPAG上の強連結成分(SCC)を一度に処理した方がよいと思われる．なぜなら，このようなプランは，サイクル中の全てのドライバを一度に計算してしまうので，不要な再読み込みを排除することが出来るからである．この考え方は，定理1におけるCCPAG中のノードに対する番号付けを，CCPAG中のSCCに対する番号付けと改めることによって実現できる．以上の考察から，一般のPAGに対して適用可能な，ページアクセスプランを考慮したレベル緩和δ-ウェーブフロントアルゴリズムは次のように記述される．

アルゴリズム 4 (PAGを用いたレベル緩和δ-ウェーブフロントアルゴリズム)

入力 EDB関係 $P(X,Y)$
IDB ルール(先祖ルール(1))
問合せ? $-A(a,X)$ (ただし，aは問合せ定数)
出力 問合せに対する答えの集合
1. 問合せ定数aに対する$CCPAG$を求める;
2. $CCPAG$をSCC，$V_1,\ldots,V_r$に分解する;
3. $CCPAG$の各SCCに番号付けを行う;
4. $WAVE := \{a\}$; $ANSWER := \phi$;
5. **while** $WAVE$中のドライバから ドライブされるページが存在する **do**
5.1 アクセス要求されているSCCのうち，**最小の番号を持つSCC** V_iを決定する;
5.2 通常のレベル緩和δ-ウェーブフロントアルゴリズムを用いて，V_iの中で$ANSWER$と$WAVE$を導出する
endwhile

□

例 6 例2の問合せを再び考える．問合せ定数aに対するCCPAGはPAGそのもの(図2参照)であり，サイクルを持たない．よって，各ノードに対する番号付けもまた，図2に示される通りである．ページP_1での導出が終了した後，我々のアルゴリズム4は，常に，P_2より先にP_3を読み込む．そして，このプランはこの問合せに対する最適アクセスプランである． □

このアルゴリズムの正当性は，このアルゴリズムがレベル緩和δ-ウェーブフロントアルゴリズムにページアクセスに対する順序付けを付加したものであることから明かである．しかし，ここで用いた番号付けは，中間関係の最小性を保証するものではない．中間関係の最小性を保証するための番号付けは今後の課題であり，(PAG上のアークの重み等の)他の先験的な情報を用いなければ，解決することは困難であると思われる．

4.2 非束縛問合せ評価におけるPAGの利用

束縛問合せの場合とは異なり，非束縛問合せを評価する際には答えと関連しない事実の集合というのは存在しない．換言すれば，「結合の前に選択を行う」という原則に従って最適化を行うことはでき

ない．よって，導出途中で生じるEDBに対する再アクセスを避けることが最適化の目標となる．

セミナイーブ法(Semi-Naive Evaluation)[2]，**ブロック化ワーシャル**(Blocked Warshall)と**ブロック化ワレン(Blocked Wallen)**[1]，**ロガリズミックアルゴリズム**(Logarithmic Algorithm)[11]，**ハイブリッドアルゴリズム**(Hybrid Algorithm)[10]，**深さ優先推移的閉包アルゴリズム**(Depth First Transitive Closure Algorithms)[9]などのアルゴリズムは，非束縛問合せを処理する効率的なアルゴリズムとして知られている．これらは，サイクルや，イレギュラーパスに起因する不要な再計算を軽減することに主眼をおいて最適化されている．しかし，これらのアルゴリズムの性能は，EDB関係の性質や各組のページへの格納のされ方によって，激しく変動し得る[10]．このような性能の変動を抑える一つの方法として，PAGのような先験的な情報を用いてアルゴリズムをさらに最適化することが考えられる．本節では，簡単のため，セミナイーブ法にPAGによる最適化を適用する．ただし，ここで提案する手法は，一般的にセミナイーブ法を基本とする幅優先探索のアルゴリズムに適用可能である．まず最初に，セミナイーブ法のアルゴリズムを簡単に記述する．このアルゴリズムは，δ-ウェーブフロントアルゴリズムと非常に類似した制御構造をもっている点に着目すべきである．

アルゴリズム 5 （セミナイーブ法）[2].

入力 EDB関係 $P(X,Y)$
IDBルール（先祖ルール(1)）
問合せ $?-A(X,Y)$
出力 問合せに対する答えの関係 $A(X,Y)$

1. $A := P$; $\Delta A := P$;
2. **while** $\Delta A \neq \phi$ **do**
2.1 $\Delta A := \pi_{14}(\Delta A \bowtie_{2=3} P)$; $\Delta A := \Delta A - A$;
2.2 $A := A \cup \Delta A$;
endwhile

□

例 7 例2で与えられたルールとEDBに対して，問合せ$?-anc(X,Y)$を考える．セミナイーブアルゴリズムは，次のようなファイルアクセスの手順で答えを見つけることになる．

1. 初期化（$P_1,\ldots,P_3$を読み込む）

$$\Delta A = \{(a,b),(b,c),(b,f),(c,d),(d,g),(e,g),(f,e),(f,h),(h,i)\}$$
$$A = \{(a,b),(b,c),(b,f),(c,d),(d,g),(e,g),(f,e),(f,h),(h,i)\}$$

2. $P_1,\ldots,P_3$を読み込む

$$\Delta A = \{(a,c),(a,f),(b,d),(b,e),(b,h),(c,g),(f,g),(f,i)\}$$
$$A = \{(a,b),(b,c),(b,f),(c,d),(d,g),(e,g),(f,e),(f,h),(h,i),(a,c),(a,f),(b,d),(b,e),(b,h),(c,g),(f,g),(f,i)\}$$

3. P_2,P_3を読み込む

$$\Delta A = \{(a,d),(a,e),(a,h),(b,g),(b,i)\}$$
$$A = \{(a,b),(b,c),(b,f),(c,d),(d,g),(e,g),(f,e),(f,h),(h,i),(a,c),(a,f),(b,d),(b,e),(b,h),(c,g),(f,g),(f,i),(a,d),(a,e),(a,h),(b,g),(b,i)\}$$

4. P_2,P_3を読み込む

$$\Delta A = \{(a,g),(a,i)\}$$
$$A = \{(a,b),(b,c),(b,f),(c,d),(d,g),(e,g),(f,e),(f,h),(h,i),(a,c),(a,f),(b,d),(b,e),(b,h),(c,g),(f,g),(f,i),(a,d),(a,e),(a,h),(b,g),(b,i),(a,g),(a,i)\}$$

このとき，EDBに対して1ページ分のみのバッファが利用できるとすれば，最低で延べ7ページのファイルアクセスが生じる． □

EDB関係のPのPAGが疎であるとき，すなわち，PAG中のアークの数が充分小さいとき，前節の議論をPAG全体に適用することで，ページアクセスレベルでの最適化が可能になる．定理1は，推移的閉包全体の計算についても成立する．

系 1 EDB関係Pに対する推移的閉包P^+全体の計算を考える．もし，PAG Pにサイクルがなければ，Pの各ページP_iを高々1回読み込むようなアクセスプランが存在する．
証明: PAGの定義と，各ノードに対する番号付けより，PAGにサイクルがなければ，ページP_iの処理の際に，その第1属性と非零のジョインセレクティヴィティをもつようなページは，PAG中でノードP_iに先行する．したがって，定理1と同様，ノード上の番号の小さいものから順に読み込むようなプランは，同じページを2度アクセスすることはない． □

この系は，PAGにサイクルがない場合に，EDB関係に対する最適のアクセスプランで，推移的閉包を計算することが出来ることを保証するものである．しかし，PAGがサイクルを含むような一般の場合には，最小のアクセスを保証することは困難である．ここでは，アルゴリズム4と同様，各SCCに対する番号付けを行うことで，一般のPAGに対して適用可能なアルゴリズムを構成する．

アルゴリズム 6 （レベル緩和セミナイーブ法）

入力 EDB関係 $P(X,Y)$
Pに対するPAG，ただし，P中のSCC $V_1,\ldots,V_r$について，$N(V_1) \leq \cdots \leq N(V_r)$とする
IDBルール（先祖ルール(1)）
問合せ $?-A(X,Y)$
出力 問合せに対する答えの関係 $A(X,Y)$

1. $A := \phi$;
2. **for** $i := 1$ **to** r **do**
2.1 **solve** (
$A_i(X,Y) :- V_i(X,Y).$
$A_i(X,Y) :- A_i(X,Z), V_i(Z,Y).$;
) ;
2.2 **if** $\exists$succ(V_i) **then**
2.2.1a $\Delta A_i := A_i$;
2.2.2a **for** $\forall k$ s.t. pred$(V_i) = V_k$ **do**
$\Delta A_i := \Delta A_i \cup \pi_{14}(\Delta A_k \bowtie_{2=3} A_i)$
endfor ;
2.2.3a $\Delta A_i := \Delta A_i - A$;
2.2.4a $A := A \cup \Delta A_i$;
else
2.2.1b $A := A \cup A_i$;
2.2.2b **for** $\forall k$ s.t. pred$(V_i) = V_k$ **do**
$A := A \cup \pi_{14}(\Delta A_k \bowtie_{2=3} A_i)$
endfor
endif
endfor

注) このアルゴリズムは，ステップ2.1が主記憶上で処理される限り，各ページに高々1回アクセスする． □

例 8 例7の問合せを再考する．レベル緩和セミナイーブ法を用いて問合せ処理を行うと次のようになる．

1. P_1を読み込む

$$A_1 = \{(a,b),(b,c),(b,f),(a,c),(a,f)\}$$
$$\Delta A_1 = \{(a,b),(b,c),(b,f),(a,c),(a,f)\}$$
$$A = \{(a,b),(b,c),(b,f),(a,c),(a,f)\}$$

2. P_3を読み込む

$$
\begin{aligned}
A_3 &= \{(f,e),(f,h),(h,i),(f,i)\}\\
\Delta A_3 &= \{(f,e),(f,h),(h,i),(f,i),(b,e),(b,h),(b,i),(a,e),(a,h),\\
&\quad (a,i)\}\\
A &= \{(a,b),(b,c),(b,f),(a,c),(a,f),(f,e),(f,h),(h,i),(f,i),\\
&\quad (b,e),(b,h),(b,i),(a,e),(a,h),(a,i)\}
\end{aligned}
$$

3. P_2を読み込む

$$
\begin{aligned}
A_2 &= \{(c,d),(d,g),(e,g),(c,g)\}\\
A &= \{(a,b),(b,c),(b,f),(a,c),(a,f),(f,e),(f,h),(h,i),(f,i),\\
&\quad (b,e),(b,h),(b,i),(a,e),(a,h),(a,i),(c,d),(d,g),(e,g),\\
&\quad (c,g),(a,d),(b,d),(a,g),(b,g),(f,g)\}
\end{aligned}
$$

この例では，データベースグラフ中の最長パスの長さに関係なく，アクセスされる延べページ数は3ページであり，これは，EDB関係のアクセスに関する限り，明らかに最小である．一方，中間関係については，レベル緩和版ではPAG上で後続するノードをもたないようなページに関する中間関係を生成しないので，レベル緩和アルゴリズムにおけるΔA_iの大きさの合計は，元のアルゴリズムにおけるΔAの大きさの合計より僅かに小さくなる．結論として，PAGにサイクルがなく，各ΔP_iが主記憶上で実現される限り，あるいは，アルゴリズム6のステップ2.1が主記憶上で処理される限り，レベル緩和セミナイーブアルゴリズムは，通常のセミナイーブ法より性能が良いと言える． □

さて，ここでアルゴリズム6の正当性を証明しよう．

補題 1 $V_1,\ldots,V_r$をEDB関係Pに対するPAGのSCCとし，$N(V_1)\le\cdots\le N(V_r)$を満たすと仮定する．また，$A_i$をルール，

$$
\begin{aligned}
A_i(X,Y) &:- V_i(X,Y).\\
A_i(X,Y) &:- A_i(X,Z),V_i(Z,Y).
\end{aligned}
$$

で定義される関係とする．このとき，手続き，

```
A := φ
for i := 1 to r do
    ΔA := A_i ∪ π14(A ⋈2=3 A_i) ;
    A := A ∪ ΔA
endfor
```

によって，Pの全閉包Aが正しく計算される．

証明: rに関する帰納法を用いる．$r=1$のときは明か．$r-1$のとき，この補題が成立していると仮定すると，$r-1$回目の繰り返しが終了した時点で，Aは$P-V_r$の推移的閉包になっている．よって，r回目の繰り返しで，

$$
\begin{aligned}
\Delta A &:= A_r \cup \pi_{14}(A \bowtie_{2=3} A_r) \cup \pi_{14}(A_r \bowtie_{2=3} A)\\
A &:= A \cup \Delta A
\end{aligned}
$$

を計算すればAはPの推移的閉包になることは明か．ところが，ΔAの式の右辺第3項は空になる．もしそうでなければ，V_rが後続するノードをもつことになり，$N(V_1)\le\cdots\le N(V_r)$という仮定に矛盾するからである．よって，補題の手続きによって，Pの推移的閉包が正しく計算できることが分かる． □

定理 2 アルゴリズム6はEDB関係Pの推移的閉包を正しく計算する．

証明: (概略) 補題1より，i回目の繰り返しで計算されるΔA_iが十分な集合であることを示せばよい．このことは，PAGの定義より容易に示すことができる． □

レベル緩和の考え方は，アルゴリズム中のステップ2.1で，各SCCの推移的閉包を求める際にも適用することが可能である．すなわち，あるページから評価を開始し，ページ単位でDFSを適用してゆく．このとき，未訪問のページに対して処理を進めるときには，不動点演算を行う必要はない．

最後に，アルゴリズム6のステップ2.1を主記憶上で実現するための方法について述べる．SCC V_i中のキーの数をm，一つの組を格納するのに必要なバイト数をB_{tuple}とする．このとき，A_iの結果の大きさは，最悪の場合m^2B_{tuple} (bytes)となる．したがって，このステップは，組指向のアルゴリズムで処理するより，隣接行列を用いて処理した方が有利であると考えられる．この場合，各キーが行列上の何行(列)目に対応するかを記憶するインデックスが必要になるが，これの大きさは$m\le 65535$のとき，$m(B_{tuple}/2+2))$ (bytes)になる．また，隣接行列の大きさは，一つのブール変数に1バイトの領域を確保するものとしても，m^2(byte)である．

例 9

$$
\begin{aligned}
T_{tuple} &= m^2B_{tuple}\\
T_{matrix} &= m(B_{tuple}/2+2)+m^2
\end{aligned}
$$

をいくつかのmについて計算してみよう．

m	T_{tuple}	T_{matrix}
10	10^4	620
100	10^6	15200
1000	10^8	1052000

ただし，B_{tuple}は100バイトと仮定した．この結果は，各V_i中のノードの数がPAGのノード数に比べて小さいときにステップ2.1を主記憶上で実現することが可能であることを示すと同時に，たとえ隣接行列を用いても，EDB全体に対する閉包を主記憶上だけで計算するには非常に大きな記憶容量が必要となることを示すものである． □

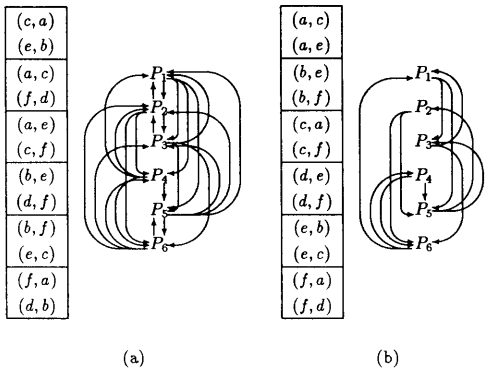
5 効率的なPAGを得るためのEDBのソーティング

PAGの大きさや，前節で述べたアルゴリズムの性能は，明らかに，関係ファイル中での各組のページへの格納のされ方，すなわち，EDBの**ページ構造**に大きく依存する．PAGが密になる要因としては，大きく分けて次の二つが考えられる．

まず第一に，ファイルがクラスタリングされていないとき，すなわち，組$(q,r_1),\ldots,(q,r_k)$が複数のページに分かれて格納されているようなとき，組(p,q)を含むページから複数のアークがでることになり，PAGのアーク数を増やす原因となる．具体的には，1ページ中の組数をm，関係ファイル中のページ数をnとすると，mはページや組の占有するバイト数に依存する定数であることを考慮すれば，一般性を失うことなく$m<n$と仮定できる．クラスタリングされていないファイルに対するPAGでは1ページから出るアークの数は，最悪，$\min(n-1,m(n-1))=n-1$となるので，PAG中のアークの総数は最悪の場合$n(n-1)$になる．一方，ファイルがクラスタリングされていて，第1属性(または第2属性)に同じキーをもつ組が同じページに格納されていれば，1ページから出るアークの数は高々$\min(n-1,m)=m$であるから，PAGのアークの総数はmnで抑えられ，ページ数に対して線形になる．

例 10 図3に示されるような，$n=6$，$m=2$の論理的には等価な二つのEDB関係のページ構造を考える．図中，(a)はクラスタリングされておらず，(b)はクラスタリングされている．(a)のページ構造に対するPAGのアーク数は24であるのに対し，(b)では12である． □

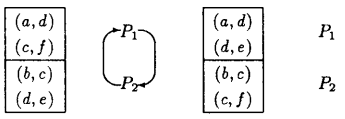
PAGが密になるもう一つの要因は，データベースグラフ中のパスが複数のページにまたがることによる．このことによって，単にPAGのアーク数が増えるのみならず，SCCが生じる原因ともなり得る．このような複数のページにまたがる導出パス(「橋」と呼ぶ)を完全に排除することは困難であり，もし，出来る限り排除しようとすれば，推移的閉包全体を計算するのと同等以上のコストを必要とするであろう．



(a) (b)

図 3: EDB のクラスタリングによる PAG の改善

例 11 図 4の (a) のページ構造に対する PAG のアーク数は 2 であるのに対し，論理的に等価な (b) では PAG にアークは無い．(a) の PAG のアークは，パス $a \to d \to e$ と $b \to c \to f$ が共に 2 ページに渡って格納されたことによって生じたものである． □



(a) (b)

図 4: ページ間の「橋」に起因する PAG 中のパスの増加

PAG が密になることによって，EDB の各ページに対して 2 回以上のアクセスが生じ得るのみならず，PAG の処理自体にかかるコストも増大する．EDB ファイルに対する最適なアクセスプランを得るには，PAG のサイクルを排除し，各ノードの次数を減少させねばならないが，良い PAG を得ようとすればするほど，EDB のソーティングに要するコストも増大する．よって，ページ構造をどの程度最適化するかは，EDB の性質や，システムの運用上生じる更新と問合せの割合などに依存する．

6 主記憶の効果的な利用法

主記憶上に，データを処理するための充分な領域を利用できる場合，主記憶と外部記憶の間で生じるスワッピングの回数を軽減することで，処理の高速化を図ることが出来る．本稿で提案したアルゴリズムは，EDB 関係のページアクセス回数を削減するものであるが，PAG にサイクルが存在する場合には再読み込みが生じ得る．しかし，その場合でも，どのページに対して再読み込みが生じるかを，PAG を利用して予測することができる．したがって，外部記憶に対して充分なキャッシュメモリが利用できる場合には，これらのページをキャッシュに残しておくことで不要な再アクセスを避けることが可能である．

さらに，より効果的な方法として，中間関係 (の一部) をキャッシュメモリに残す方法が考えられる．δ-ウェーブフロントアルゴリズムでは，ウェーブフロント集合は主記憶上で処理されることが仮定されている．しかし，非束縛問合せを処理するアルゴリズムでは，中間関係が非常に大きくなると考えられるので，このような仮定を置くことは妥当でない．ところが，本稿で提案したレベル緩和セミナイーブアルゴリズムでは，PAG の情報を活かしてキャッシングの戦略を立てることが可能である．具体的には，PAG 上で未処理の後続ノードをもつ P_i に対する ΔA_i をキャッシュメモリに残すことがキャッシングの戦略となる．このようなプランによって，問合せ評価の上で最適なアクセスプランと最適なメモリ効率を両立させるためには，PAG の各ノードに対して，より厳密な番号付けが要求される可能性がある．この問題は，今後の課題である．

以上のようなページ間のジョインセレクティヴィティを考慮したキャッシングの戦略は，通常の LRU(Least Recently Used) 方式などに比べてより効率的なメモリの管理が可能となると考えられる．

7 いくつかの例題に基づく性能比較

第 4章の議論によれば，各アルゴリズムの相対的な性能は，束縛問合せについては，

レベル緩和δ-ウェーブフロント
≥ PAG を利用したレベル緩和δ-ウェーブフロント

非束縛問合せについては，

セミナイーブ法
≥ PAG を利用したレベル緩和セミナイーブ法

が成立すると予想される．ただし，$A \geq B$は，Bが A より優れていることを示す．しかし，[7][10] にあるように，この種の最適化に対して，定量的な評価によって差を見いだすことは困難である．

特に，問合せの束縛の有無に関わらず，例えば，PAG 全体が強連結であるような場合には，明らかに PAG による最適化は単にオーバーヘッドになるのみである．このような場合，EDB のページ構造の最適化をどの程度行うか，あるいは，PAG を用いた最適化を行うか否かさえ，実システム上での実現の問題に強く依存すると考えられる．したがって，ここでは，PAG による最適化が効果をもつ典型的な例を挙げて，大まかな比較をするに留める．以下の比較では，PAG 自体を処理するのにかかるコストは，問合せ評価にかかるコストに比べて十分小さいと仮定する．

7.1 レベル緩和δ-ウェーブフロントアルゴリズムに関する PAG の利用による効用

束縛問合せ評価アルゴリズムについては，$CCPAG = (V_c, E_c)$ が，

$$
\begin{aligned}
V_c &= \{P_1, P_2, \ldots, P_n\} \\
E_c &\ni (P_{i-1}, P_i) \qquad (2 \leq i \leq n) \\
E_c &\ni (P_1, P_i) \qquad (3 \leq i \leq n)
\end{aligned}
$$

のように与えられる場合に，PAG の効用が最も顕著になる．

このような CCPAG(図 5参照) の V_c中のノードに対する番号付けは，$N(P_i) = i$ となる．E_c中の全てのアークが答えに寄与するとし，さらに，異なるパスから得られる答えの値はすべて異なると仮定する．このとき，中間関係や答えの処理に必要なコストはアクセスプランに関わらず一定と見なせるので，EDB の読み込みのコストだけを問題にする．

与えられた CCPAG に対する最良のアクセスプランは，$P_1 \to P_2 \to \cdots \to P_n$ であり，このとき読み込まれる延べページ数は n である．逆に，最悪のアクセスプランは，$P_1 \to P_n \to P_{n-1} \to P_n \to P_{n-2} \to P_{n-1} \to P_n \cdots P_2 \to P_3 \to \cdots \to P_n$であり，EDB ファイルに対して用意されるバッファの大きさが 1 ページ分であると仮定すると，読み込まれる延べページ数は $(1 + n(n-1)/2)$ となる．結局，通常のδ-ウェーブフロントアルゴリズムを用いると，読み込まれるページ数は n から $(1 + n(n-1)/2)$ の間で変動するのに対し，PAG を用いて最適化を行えば，常に n ページのアクセスで全処理を行うことができる．

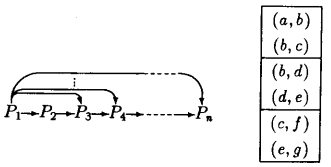


図 5: 束縛問合せに対する PAG の例

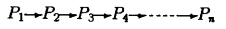


図 6: 非束縛問合せに対する PAG の例

7.2 セミナイーブ法における PAG を利用したレベル緩和の効用

つぎに，通常のセミナイーブアルゴリズムと PAG を利用したレベル緩和セミナイーブアルゴリズムを比較する．レベル緩和セミナイーブアルゴリズムで，A_iを主記憶上で計算するものと仮定すれば，中間関係の大きさに関する比較は，レベル緩和アルゴリズムのΔA_iの大きさの総計ともとのアルゴリズムのΔAを比べれば充分である．例 8での議論によれば，

$$\bigcup \Delta A_i \text{ in Level Relaxed Algorithm}$$
$$\subseteq \bigcup \Delta A \text{ in each iteration of Semi Naive}$$

が成立する．したがって，レベル緩和アルゴリズムによる問合せ処理において中間関係と答えの関係の処理に必要なコストは，元のアルゴリズムより大きくなることはないと考えられる．

以上のような考察から，以下では前節同様 EDB の読み込みに必要なコストについて考える．特に，単純であるにも拘らず両者の差が出るような例としては，次のような $PAG = (V, E)$(図 6参照) が考えられる．

$$V = \{P_1, P_2, \ldots, P_n\}$$
$$E \ni (P_{i-1}, P_i) \qquad (2 \leq i \leq n).$$

セミナイーブ法については，アルゴリズムの繰り返し回数がデータベースグラフ中の最長パスの長さ h に依存し，読み込まれる延べページ数は $(h+1)n$ と見積ることができる（ただし，この値は実現されるジョインアルゴリズムによっても大きく変化する）．一方，本稿でのレベル緩和アルゴリズムは，この PAG に対して高々 n ページのアクセスで済む．以上の議論から，例示した PAG に対して最適化の効果があることが分かる．

8 むすび

本稿では，演繹データベースにおける推移的閉包問題の問合せ評価に関して，EDB のファイル中のページに対するアクセス回数を最小化するアルゴリズムを示した．最適なアクセスプランを立てるための有効な手段として，ページ間のジョインセレクティヴィティの有無を表現するページアクセスグラフ (PAG) を導入し，PAG がサイクルを持たない場合には，本稿で提案したアルゴリズムが最適なページアクセスプランを与えることを示した．

この結果，本稿のアルゴリズムは，PAG の全体，ないし一部を主記憶上で処理できるという条件の下で，再帰問合せ評価アルゴリズムの性能を向上させるのに大いに効果のあることが分かった．本稿の結果より，演繹データベースマシンなどのシステムを構築する際に，このアルゴリズムをハードウェアレベルで実現することにより，再帰問合せを高速に処理することが可能になると考えられる．

最後に，今後の研究課題を挙げておく．

1. 本稿では，簡単のため PAG 中の SCC の処理に関する詳細な議論を避けた．実際に性能のよいアルゴリズムを得るためには，SCC の処理を詳細に記述する必要がある．
2. 実際の使用に耐える PAG を得るために，EDB を効率的にソートするアルゴリズムが必要である．
3. PAG の考え方を通常のジョインアルゴリズムに適用することも興味深い課題である．この場合のアルゴリズムは，ハッシュジョインに類似したものになると予想されるが，詳細な性能解析を行うことが重要であると考えられる．

謝辞

末筆ながら，本研究に関して有益な御助言を頂いた京都大学大学院生河野浩之氏，三浦史光氏に感謝の意を表する．

参考文献

[1] R.Agrawal and H.V.Jagadish, "Direct Transitive Closure Algorithms: Design and Performance Evaluation", *Technical Memorandum, AT&T Bell Laboratories,* Murray Hill, New Jersey, 1987.

[2] F.Bancilhon, "Naive Evaluation of Recursively Defined Relations", *On Knowledgebase Management Systems (M.Brodie and J.Mylopoulos eds.),* Springer-Verlag, New York, 1986.

[3] F.Bancilhon, D.Maier, Y.Sagiv and J.Ullman, "Magic Set and Other Strange Ways to Implement Logic Programs", *Proc. 5th ACM SIGMOD-SIGACT Symposium on Principles of Database Systems,* Cambridge, Massachusetts, 1986, pp.1-15.

[4] F.Bancilhon and R.Ramakrishnan, "An Amateur's Introduction to Recursive Query Processing Strategies", *Proc. ACM-SIGMOD Int'l Conf. on Management of Data,* Washington D.C., 1986, pp.16-52.

[5] J.Han and L.J.Henschen, "Processing Linear Recursive Database Queries by Level and Cycle Marging", *Tech. Rep. 87-05-DBM-01,* North Western Univ., Chicago, 1987.

[6] J.Han and L.J.Henschen, "Handling Redundancy in the Processing of Database Queries", *Proc. of ACM SIGMOD Int'l Conf. on Management of Data,* San Francisco, California, 1987, pp.73-81.

[7] J.Han, G.Qadah and C.Chaou, "The Processing and Evaluation of Transitive Closure Queries", *LCCR Tech. Rep. 88-1,* Simon Fraser Univ., Canada, 1988.

[8] L.J.Henschen and S.Naqvi, "On Compiling Queries in Recursive First-Order Databases", *J.ACM, 31(1),* 1984, pp.47-85.

[9] Y.E.Ioannidis and R.Ramakrishnan, "Efficient Transitive Closure Algorithms", *Proc. 14th Int'l Conf. Very Large Data Bases,* Los Angels, California, 1988, pp.382-394.

[10] H.Lu, "New Strategies for Computing the Transitive Closure of Database Relation", *Proc. 13th Int'l Conf. on Very Large Data Bases,* Brighton, England, 1987, pp.255-266.

[11] P.Valduriez and H.Boral, "Evaluation of Recursive Queries Using Join Indices", *Proc. the 1st Int'l Conf. on Expert Database Systems,* Charleston, South Carolina, 1986, pp.197-208.

[12] P.Valduriez, "Join Indices", *ACM Transactions on Database Systems, 12(2),* 1987, pp.218-246.