単一化可能パタン検索の最適化

5L-6

大森 匡　田中 英彦

東京大学工学部

1.はじめに

演繹データベースシステムでは単一化可能パタンの検索は要求パタン・対象パタン共に複数個存在する場合，組合せが増大し処理の隘路となりやすい。本稿ではこの操作を単一化を伴う関係代数の結合演算とみなし，ハッシュとソートによって実行するアルゴリズムについて述べる。また，他方式との比較を行いその最適化アルゴリズムについても述べる。

2.単一化を伴う結合演算のアルゴリズム

以下では原子論理式を属性値に持つ関係A，B間の単一化を伴う結合演算を例にとる。簡単のために各関係はハッシング等によって述語，アリティ一定のタプル集合に予め分割されていると仮定する（Fig.1）。

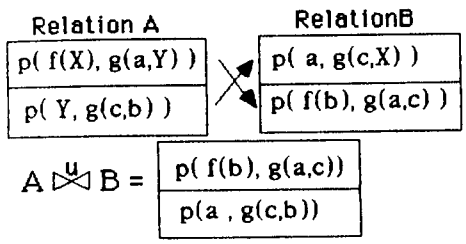

Fig.1 an example for join with unification

まずアルゴリズムの戦略について説明する。

与えられた述語のアリティに応じてFig.2のように木構造を設定し原子論理式からなるタプルt1,t2をこれにあわせて展開する。

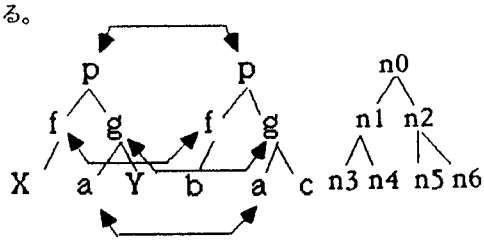

Fig.2 an example for unifiable atomic formulae

↔ : equal symbol

このとき二つのタプルt1,t2が単一化可能であればハッシュ木の各ノードniについて次の条件が成立する。

『niに対応するタプルt1,t2のシンボルをs1,s2とするとs1,s2がともに変数でなければこれらは一致しなければならない（Fig.2）。』

アルゴリズムは以下の通り（Fig.3）。

STEP1. ハッシング

関係A,Bの各タプルtについてハッシュ木のノードn0,n1,...に応じたシンボルs0,s1,....を取り，ベクタ形式のハッシュ値（以下ハッシュベクトル）〔h(s0),h(s1),....〕を求める。

(但し，h(変数シンボル)=0，h(その他)=1～N)

このハッシュベクトルに応じて関係A,Bをバケット集合$\{a_i\}$,$\{b_j\}$ (i,j=0,1,....)にクラスタリングする。

STEP2. ソーティング

任意のバケット組合せai,bjについてハッシュベクタH(ai)=〔p1,p2,..〕,H(bj)=〔q1,q2,..〕とする。このときもしpk・qk≠0,かつpk≠qkなるkが存在すれば，バケットai,bjの任意のタプル組合わせについてノードnkに対応する非変数シンボルは一致せず故に単一化可能なタプル組合せは存在しない。そうでなければpk=qk≠0なるkに対応するハッシュ木上のノード列をキーにしてバケットai,bjのタプルをマージソートする。その後，ソート列中のキーが等しいA,Bのタプル同士を総当たりで単一化試行。

$$p(f(X),g(a,Y)) \in a_i$$
$$p(f(b),g(a,c)) \in b_j$$

	n0	n1	n2	n3	n4	n5	n6
H(ai)= [	h(p),	h(f),	h(g),	0 ,	0,	h(a),	0]
H(bj)= [	h(p),	h(f),	h(g),	h(b),	0,	h(a),	h(c)]
	○	○	○	×	×	○	×

node sequence [n0,n1,n2,n5] makes key

Fig.3 an example for unifiable bucket

3. 他方式との比較

単一化を伴う結合演算のアルゴリズムはICOTからも提案されている。この方法（以下ICOT法）では原子論理式をbreadth-firstに展開し変数>定数の全順序をつけてマルチキーソートを行う。そのため優先順位の高いキーに変数シンボルがある原子論理式はソートストリームの先頭に現れそこからストリームの最後まで単一化試行を必要とし，単一化負荷の増大を招く。

An Optimization in Unifiable Pattern Search

Tadashi OHMORI , Hidehiko TANAKA

The University of Tokyo

これに対しhash&sort法は予め構造情報によるクラスタリングを行うことによって変数になりやすいノードがキーになることを回避し，また，『タプルを大量に発生する原因になる，定数になりやすいノード』をキーにして絞り込む。

関係A,B 間の単一化を伴う自然結合をhash&sort法，ICOT法で実行した（Table.1 ）。サンプルS1からS3に行くに従いbreadth-first で優先順位の高いノードの変数確率が高くなっている。単一化負荷，ソート負荷は各々｜A｜・｜B｜，｜A｜＋｜B｜に対する比率である。ICOT法が変数の出現確率に大きく影響されるのにたいしhash&sort法は常により高い絞り込み効果を達成できることがわかる。一方，hash&sort法のソート負荷はICOT法に対し2 ～6 倍となる。そのため単一化負荷，ソート負荷間のトレードオフを考慮し，タプルのパタン分布に応じたハッシュ木の最適化が必要となる。

Table 1. results of join with unification.
value = ‹ICOT› / ‹hash&sort›
( ) is default success ratio.

sample	load of unification	success ratio	load of sorting	|A|•|B|
S1	0.3 / 0.12	0.3 / 0.64 (0.08)	1.0 / 5.6	10^4
S2	0.63/0.42	0.61/0.82 (0.37)	1.0 / 5.3	10^3
S3	1.0 / 0.27	0.22/0.81 (0.22)	1.0/1.9	10^3

4. hash &sort法の最適化アルゴリズム

hash&sort法では変数になる確率が低いノードを使ってクラスタリングをしても単一化負荷はさほど軽減されずソート負荷の増大によって却って全体のコストが増大する場合がある。そうしたノード列はICOT法のように全順序をつけ，hash&sort法によるキー（以下主キー）の後ろに補助キーとして追加すれば良い（Fig.4 ）。ソート列中主キーが一致し，補助キーがICOT法同様全順序を保つタプル組合わせのみ単一化を試行する。

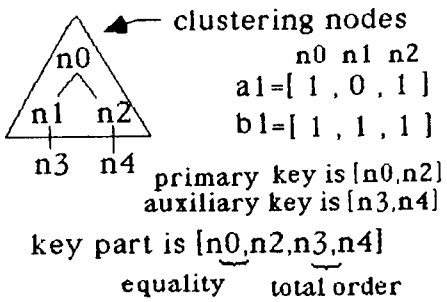

Fig.4 an optimized key

従って関係A,B のタプルのパタン分布に応じたハッシュ木の設定，つまり変数／非変数によってクラスタリングを行うノード（以下クラスタリングノード）の決定，および上述した補助キーにするノードの決定が必要となる。

以下はそのノード列生成アルゴリズムである。

今，ノード列n_{i_1} ,.... , $n_{i_{k-1}}$をクラスタリングノードとするバケットai,bj を取る。ai,bj をさらにノードn_{i_k}の変数／非変数に応じてａi0，ａi1 ，ｂj0，ｂj1に分割するとこれによる単一化負荷の増分 ΔU，ソート負荷の増分 ΔSは以下のようになる（Fig.5 ）。

$$\Delta S = (|ai0| + |bj0|) + (|ai0| + |bj1|) + (|ai1| + |bj0|) + (|ai1| + |bj1|) - (|ai| + |bj|) = |ai| + |bj|$$

ΔU＝ノードｎｋ をキーにすることで省かれる単一化試行数
＝Fig.5 の▨部にあたる負の値

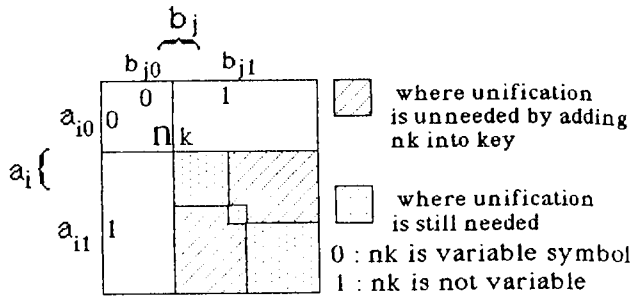

Fig.5 an example for cost-increment

故にコスト増分 ΔC＝Ks ・ΔS ＋Ku ・ΔU
（Ks ：単位ソート負荷，　Ku ：単位単一化負荷）
とすると ΔC< 0ならノードn_{i_k}をクラスタリングノードに追加する。ΔC > 0ならまだクラスタリングノードに含まれていないノードの内，変数になる確率が低いものから優先順位を高くして補助キーとする。関係A,B の直積空間をこの方法で述語シンボルから始めて再帰的にブロックに分割し，各ブロック毎にその条件を満たすA,B のバケット集合を取る。そしてそのブロックの指定ノードをFig.4 に従ったキーとしてソートを行う（Fig.6 ）。

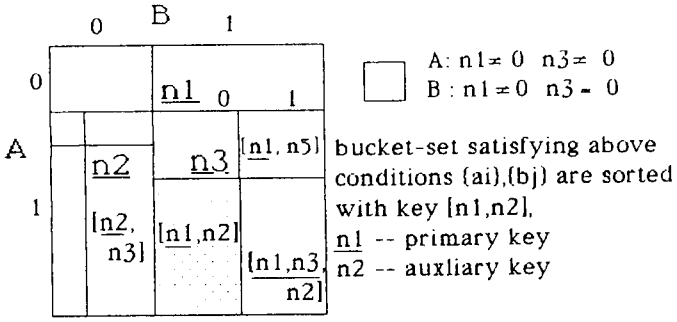

Fig.6 an example for optimized buckets

5.おわりに

本稿では単一化を伴う結合演算をハッシュとソートによって実行するアルゴリズムについて述べ，他方式と比較しその特性に基づいた最適化法を提案した。我々は，多くのアプリケーションではクラスタリングノードは少数で充分と考えている。今後はhash&sort法の最適化アルゴリズムの有効性を確認していく予定である。

〔参考文献〕

大森　他　”推論機能と関係データベースの融合方式”
電子通信学会技術研究報告AI-86-21