1D−3 データ縮退を用いた大規模リレーションにおけるグラフ算法の高速化

齋藤 勉　大森 匡*　田中 英彦
東京大学 工学部

1 背景と目的

データベースにおける再帰問合せは、データベースリレーションをグラフとみなした上でのグラフ算法を表現しているものが多い。例えば推移閉包演算や単一出発点最短経路演算はいずれも再帰問合せの処理に必要な演算である。従来の研究では、ごく一部の再帰問い合わせしか扱えず、一般的なグラフ算法は議論されてこなかった。本論文では代表的なグラフ算法として単一出発点最短経路問題を取り上げ、巨大データベースリレーション上での一般的なグラフ算法に適用可能な問い合わせ処理方式を提案する。

2 問題点

1. 一般にグラフ算法には、最短経路問題や二重連結成分展開といった、グラフ中の経路を探索し、探索途中の経路集合に対して何らかの処理をおこなう算法が多い。こうした算法では、算法自体の並列性は限定されていることが多い。
2. 算法実行中に必要な中間データが多いためにそれらのデータのバッファリングが全体の性能を支配する。
3. 単一出発点最短経路に限らず、できるだけ広いグラフ算法に適用可能な手法が望ましい。

3 データ縮退による問い合わせ処理

最短経路問題を解くとは、言い換えれば、データベース上の辺を表現しているタプルの集合から最短経路をなす辺のタプル集合を選び出すことである。(図 2. に示すように、元データとなるタプルは、[タプルid、出発点ノード名、到達点ノード名、辺の長さ、その他のデータ] から構成されている。) この時用いるアルゴリズムとして、ダイクストラの算法 (図 1.) を用いるが、この算法にはB,Dの二つの集合が必要となる。

中間集合Bをどのようなデータ型にするかで、次の3種の方法が考えられる。

NAIVE 辺を表すデータとして、データベース上のタプルをそのまま用いる方法。

CUT データベース上の元データからグラフ算法に必要なデータのみに射影 (projection) したものを用いる方法。射影は、グラフ算法実行中、データを2次記憶にアクセスした時に行なう。

REDUCED 元データからグラフ算法に必要なデータのみに射影し、さらに、属性変換テーブル (図3.) を用いて、ノード名を4バイトのid番号に変換したものを用いる方法。この場合、実際にグラフ算法を実行する前に、データベースを縮退データ (図 4.) に変換し、それを用いてグラフ算法を実行することになる。縮退操作は、ディスクの並列化によって容易に高速化が図れる。

REDUCED は、データ縮退を積極的に利用しようとする新しい方法であり、この方法を用いれば問い合わせ処理は、グラフ算法自体の並列度制約を受けず、また、データのバッファリングも不要となる。

どの方法についても、解答集合Dは、[ノード名またはid、そのノードを終点とした最短経路をなすタプルid、そこまでの最短距離] とし、グラフ算法の実行後データベース上の情報とつき合わせ、答え集合を構成するものとする。この復元操作も、ディスクの並列化によって簡単に性能向上が取り出せる。

解答集合D 最短経路が既に計算された頂点、及び最短経路をなす辺の集合
中間集合B 解答集合Dから直接のびている辺及びその先の頂点のうち、現在のところその頂点への最短経路をなすものの集合
手順 1 出発点 v_0 をBに加える。
手順 2 Bの中で v_0 からの距離が最短のノード v を選ぶ。
手順 3 v をDに加える。
手順 4 v を始点とする辺 (v,w) について、w がDにふくまれていないなら、w をBに加え、Bのデータを更新する。
手順 5 Bがなくなるまで手順2.〜4. を繰り返す。

図 1: ダイクストラのアルゴリズム

4 シミュレーションモデル

シミュレーションを行なってI/Oコストを評価した。その際、対象とするデータベースのファイル編成として、タプルidについてdense indexがあるとする。ファイル編成として次の3種を考える。

DENSE INDEX (DI) 頂点名についてdense indexがある場合。

CLUSTERED (CL) 頂点名についてB木などでクラスタ化されている場合。

NO INDEX (NI) インデクスがない場合。この場合、グラフ算法の実行の前に、頂点名についてクラスタ化したファイルをデータベースより作成し、それを利用する。

測定環境におけるパラメータを表1.に示す。ただし、インデクスアクセスのコストは無視している。

入力するグラフはノード数を1000、各頂点からの辺の数を10と一定にし、辺の先とコストはランダムに決定した。開始点から到達可能なノードは1000ノード中996ノードであった。

---

*'90年6月現在 三菱電機(株) に所属

Data-Size Reduction: Supporting Graph Algorithms on Large Database Relations
Tsutomu SAITOH,Tadashi OHMORI,Hidehiko TANAKA
the University of Tokyo

表 1: 測定環境におけるパラメータ

	測定環境
タプル数	10k 件
ノード数	1k 件
データベースのページ数	2500 pages
1 タプルのバイト数	256 bytes
1 頂点情報のバイト数	32 bytes
1 ページのバイト数	1k bytes
1 ページあたりのタプル数	4 個
メインメモリのバイト数	40k ～ 200k bytes
メインメモリのページ数	40 ～ 260 pages
ディスクユニット	10 台

表 2: 各方法における実行時間 (IO COUNT)

memory page	60	140	180	220
REDUCED	3473	1174	766	367
CUT, DI	12908	9861	9713	9576
CUT, CL	7098	2988	2976	2971
CUT, NI	9214	4764	4656	4543
NAIVE, DI	16652	13739	12444	11249
NAIVE, CL	12844	8578	6643	4853
NAIVE, NI	14865	10586	8647	6853

5 性能評価

表 2. に実行結果を示す。(データベースは 2500 ページ) 一次記憶量が 200 ページ付近で、REDUCED は NAIVE でクラスタ化インデクスを用いた時に較べて、10 分の 1 の実行時間短縮が達成できた。

REDUCED の場合、一次記憶が 180 ページ以上あるときは、縮退データ上のグラフ算法自体は一次記憶上のみで出来、ディスクアクセスはその前後のスキャン 1 回づつ計 2 回で済むことになる。従って REDUCED は、他の方法に比べて、ディスクユニットの並列化の効果を十分に引き出すことができる。

200 ページ以下での REDUCED の性能の悪化は、主に、データ縮退時の縮退データのメモリ溢れによるものである。

6 結論

巨大なデータベースリレーションのグラフ算法は、中間データが大きいためバッファリングが難しく、また並列化による性能向上が困難であるという欠点を持っていた。これらを解決するため、グラフ算法に必要な情報のみにデータを縮退して計算を行ない、終了後に復元操作を行なうといった手法 REDUCED を提案した。

この手法を用いてシミュレーションを行なった結果、一次記憶比で約 10 倍のデータベースにおいて、1 次インデクスをもつファイル上で算法を実行するときにくらべ、10 倍の速度向上が得られた。

今後の研究としては、REDUCED において、縮退データの生成時に、Hybrid Hash 等のステージング手法を使用して、一次記憶比で 100 倍程度のデータベースについても同様の性能を引き出したいと考えている。

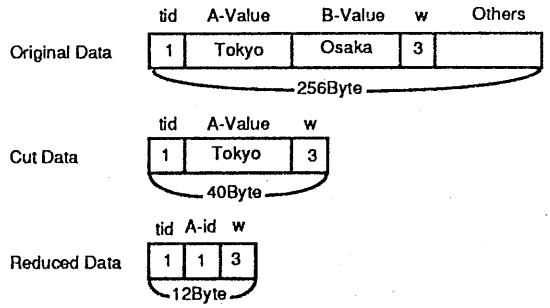

図 2: 辺を表すデータ

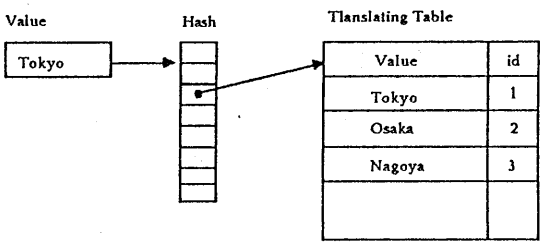

図 3: 属性変換テーブル

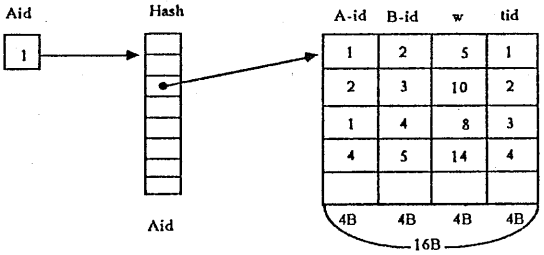

図 4: 縮退されたデータ

参考文献

[1] P.-A. Larson and V. Deshpande. A file structure supporting traversal recursion. In *Proc. ACM-SIGMOD Int'l Conf. on Management of Data '89*, pp. 243–252, 1989.

[2] 津高新一郎. データベースマシンにおける巨大リレーションのグラフ算法, 1989. 東京大学工学部電子工学科 卒業論文.