データベースからの知識獲得を用いたデータベース圧縮システムREDUCE1

6AA－7　相坂一樹　呉乾禮　塚本昌彦　西尾章治郎
大阪大学大学院 工学研究科 情報システム工学専攻

1　はじめに

ここ数年ディスク価格は急激に下落しているが，データウェアハウスのような大規模データベースにおいては，蓄積コストはまだ主要なコストを占めている．この蓄積コストを下げるためには，データベースを圧縮することが有効であると考えられる．

データベースを圧縮することにより，単に必要な記憶容量を減らすだけでなく，バックアップデータの量も減らすことができる．これにより，バックアップ時間を短縮したり，データ回復において安全性を増すことができる．さらに，ネットワーク環境においても同じ記憶容量でたくさんのオリジナルデータのコピーを作ることができ，分散化に利用できる．また，データ転送においては，データ圧縮を利用して転送した方が，転送時間が短くなる．これは，文献[5, 6]で提案されているデータベース移動機構に特に有効である．

従来のデータ圧縮技術[9]を用いてデータベースを圧縮すると，蓄積容量が小さくなる代わりに，データベースにアクセスする場合には，圧縮したデータベース全体を展開しなければならないので，アクセス速度が非常に遅いという欠点がある．

これに対して，文献[4]では，データベースからの知識獲得[2]を用いてルールを発見し，そのルールをデータと置き換えることによって圧縮を行う方法が提案されている．この方法を用いれば，圧縮率は従来の方法に比べて下がる場合があるが，アクセス速度は速くなるものと考えられる．本稿では，このデータベースからの知識獲得を用いた圧縮機構をもつデータベースシステムREDUCE1(RElational and DedUctive database Compression Engine version 1)の設計と実装を行った．そして，システムの有効性を確かめるために，REDUCE1を用いて実際に大阪大学工学部の業績データベースを圧縮し，圧縮率と問合せ時間を計測した．

以下では，まず，2章でデータベースからの知識獲得を用いたデータベース圧縮方法の概要を述べる．次に，3章でREDUCE1の設計と実装について述べ，4章でREDUCE1を用いて業績データベースを圧縮した結果を示し，それに対する評価を行う．最後に，5章で結論と今後の課題を示す．

2　データベースからの知識獲得を用いたデータベース圧縮方法の概要

データベースからの知識獲得を用いたデータベース圧縮方法は，データベースから知識獲得を使ってルールを導出し，その導出されたルールを，該当するデータの代わりに演繹データベースのIDBに蓄えることによって，データ量を減らす．ルールで記述できなかったデータは，演繹データベースのEDBに蓄える．この圧縮方法の利点は，データベースを圧縮しても，データベースにアクセスする際，演繹に必要な部分だけにアクセスすればよいことである．

圧縮アルゴリズムには，文献[4]で提案されている2つのアルゴリズムのうち，アプリオリ・アルゴリズム[1]を選択した．圧縮の際には，ユーザの入力したしきい値以上同じパターンのデータがあるとルールに変換する．

REDUCE1 : Database Compression System using Knowledge Discovery in Databases
Kazuki AISAKA, Chien-le GOH, Masahiko TSUKAMOTO, and Shojiro NISHIO.
Department of Information Systems Engineering, Graduate school of Engineering, Osaka University.

3　REDUCE1の設計と実装

3.1　REDUCE1の構成

REDUCE1の演繹データベース管理システムには，米国ウィスコンシン大学マジソン校で開発されたCORAL[8]を用い，演繹データベースのEDBには，ORACLE社のORACLE7[7]を使用した．CORALはSun Microsystems社のマシン上で動作し，ORACLE7は，Silicon Graphics社のマシン上で動作する．そのため，実装においては，CORALとのインタフェース部分のみSun Microsystem社のマシン上で実装し，その他の部分はSilicon Graphics社のマシン上で実装した．互いの通信は，ソケット通信によって行うようにした．

3.1.1　モジュール

REDUCE1には6つのモジュールがある．本節では，その6つのモジュールについて順番に述べる．

ユーザインタフェース・モジュール: ユーザからのSQL文を解析し，圧縮を行う場合は知識獲得モジュールに，それ以外は問合せプリプロセッサ・モジュールにSQL文を渡す．

知識獲得モジュール: EDBインタフェース・モジュールに，知識獲得に必要なデータをSQL文の形式で要求し，ユーザの入力したしきい値を使ってアプリオリ・アルゴリズムを実行する．そして，発見されたルールの追加をDDBコミュニケーションモジュールに依頼する．また，ルールの検索を容易にするためルールはEDBにも蓄えるので，圧縮したデータの入力とともに，ルールの追加もEDBインタフェース・モジュールに依頼する．

問合せプリプロセッサ・モジュール: ユーザの入力したSQL文を解析し，演繹操作が必要であれば，EDBインタフェース・モジュールに演繹に必要なデータの取得を要求し，DDB通信モジュールに演繹を依頼する．演繹操作が必要でないならば，EDBインタフェース・モジュールにSQL文を渡す．

EDBインタフェース・モジュール: ORACLE7とのインタフェース部分であり，SQL文をORACLE7に渡し，結果を受け取る．

DDB通信モジュール: CORALインタフェースモジュールに，CORALへのルールの追加や削除，ルールの演繹などをソケットを通じて要求する．また，実際にはルールの検索を容易にするため，ルール表がEDBに格納してあるので，ルールの検索・追加・削除の際には，ルール表に対する操作もEDBインタフェース・モジュールに要求する．

CORALインタフェース・モジュール: DDB通信モジュールから，ソケットを通じて要求された命令に応じて，実際にCORALにアクセスしてルールの追加，削除またはルールの演繹を行い，処理が正常に終ったかどうかを通知する．

データベースに対して読み込み操作を行う部分には，EDBに問合せ表を作り，その問合せ表に対して問合せを行う方法と，問合せ表を作らずにSQL文から直接演繹するルールを決定し，演繹結果が問合せ結果となる方法の2種類の方法を実装した．直接演繹する方法は，SQL文を演繹問合せの形に変換し，演繹するルールを決定する機構が必要となる．この変換機構の処理を効率的に行うことができれば，従来の圧縮していないデータベースと同様の問合せ速度を実現できるものと考えられる．今回の実装では，簡単なSQL文のみに直接演繹する方法を用いている．

また，削除には，削除バッファに入れる方法と，実際のリレーションで削除を行い，ルールも削除してしまう方法の2

通りがある．今回実装したシステムでは，SQL 文の書き方によってどちらの方法で削除するかを決定することにしたが，削除するデータの量によって，2 通りの方法のうち，蓄積容量が少なくなる方を選ぶことが有効であると考えられる．

4 REDUCE1 の評価

本章では，REDUCE1 を用いて行った実験について述べる．実験には，大阪大学工学部の業績データベースを用いた．業績データベースには，属性として論文番号，著者名 (4 人まで)，掲載誌，発行年月，キーワード 3 つ，専門分野の識別番号，所属が入っていて，タップル数は約 2000 個である．

4.1 最適なしきい値の決定

最適なしきい値を求めるために，しきい値を 3 から 30 まで変化させ，圧縮率の違いを計測した結果が図 1である．圧縮率は，

圧縮率 = (圧縮後のデータサイズ)/(圧縮前のデータサイズ)

で計算した．図の縦軸と横軸には，それぞれ圧縮率としきい値を取った．

図 1より，最も良い圧縮率はしきい値 10 の時の 82.9%であった．この値は，従来のデータ圧縮手法と比較すると良い結果とは言えないが，他の知識獲得アルゴリズムと組み合わせることによって，さらに圧縮率は良くなるものと考えられる．

しきい値 10 の時，導出されたルールは 31 個で，新しくできた表の数も 31 個になった．そして，元の表には 17 個のタップルしか残っていなかった．31 個のルールの中には，あまり人間には有用と思われないルールもあるが，知識獲得を用いたデータベース圧縮方法では，そのようなルールでも圧縮に使うことができる．

4.2 問合せ時間の比較

圧縮を行っていないデータベースと，4.1節で求めた最適なしきい値を用いて圧縮したデータベースに対して読み込み操作を行い，その処理時間を比較した．REDUCE1 の読み込み操作には，問合せ表を EDB に作ってから問合せを実行する方法と問合せを演繹問合せに変換し，演繹結果を直接問合せ結果とする方法の 2 通りの方法があるが，実験には演繹接問合せに変換する方法を用いた．

結果を図 2に示す．図の横軸には，問合せの結果返ってきた答がデータベース全体に占める割合を取った．実験の際は，例えば，データベース全体の約 50%を占めるような問合せには，

SELECT * FROM 業績表 WHERE 論文番号 > 1000

のような SQL 文を用いた．また，図の縦軸には CPU 時間を取った．

図 2より，圧縮を行ったデータベースでの問合せは，圧縮を行なっていないデータベースより時間がかかる．しかし，その差はわずかであるので，SQL 文を効率的に演繹問合せに変換できるようになれば，ほとんど同じ時間で問合せを行うことができるものと考えられる．

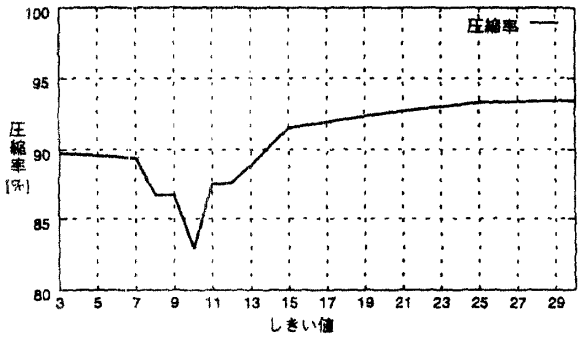

図 1: 圧縮率

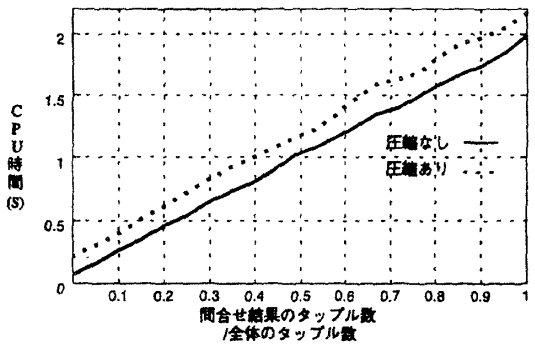

図 2: 問合せ時間

5 おわりに

本稿では，文献[4] で提案されているデータベースからの知識獲得を用いた圧縮機構をもつデータベースシステム REDUCE1 の設計，実装を行った．そして，REDUCE1 を用いて，実験を行った結果，効率的に問合せを行うことができることがわかった．

今後のシステムの改良点として，他の知識獲得アルゴリズムと組み合わせて，より圧縮率を高めることなどがあげられる．また，より高速なシステムを構築するために文献[3] で提案されているアルゴリズムを用いて，知識獲得を圧縮した状態から行うことなどを検討する必要がある．

謝辞

末筆ながら，本研究において貴重な御助言を頂いた，滋賀大学 谷口伸一助教授に感謝の意を表す．また，有益な御助言を頂いた春本要助手を始め，西尾研究室の諸氏に感謝の意を表す．

参考文献

[1] R.Agrawal and R.Srilant: "Fast Algorithms for Mining Association Rules," in *Proc. of the 20th VLDB Conference*, pp.487-499 (1994).

[2] U.M.Fayyad, G.Piatetsky-Shapiro, R.Uthurusamy: "Advances in Knowledge Discovery and Data Mining," AAAI Press / The MIT Press (1996).

[3] C.-L.Goh, M.Tsukamoto, and S.Nishio: "Knowledge Discovery in Deductive Databases with Large Deduction Results: The First Step," in *IEEE Trans. on Knowledge and Data Engineering, Special Issue on Database Mining*, Vol.8, No.6, pp.952-956 (1996).

[4] 呉乾禮, 塚本昌彦, 谷口伸一, 西尾章治郎: "データベースからの知識獲得を用いたデータベース圧縮について," 人工知能学会 知識ベースシステム研究会 (第 35 回), pp.1-6 (1996).

[5] T.Hara, K.Harumoto, M.Tsukamoto, and S.Nishio: "Database Migration for Transaction Processing in ATM Networks," in *Proc. of the International Conference on Information Networking (ICOIN-11)*, Vol.1, pp.1B-4.1-1B-4.10 (1997).

[6] T.Hara, K.Harumoto, M.Tsukamoto, and S.Nishio: "Location Management Methods of Migratory Data Resources in ATM Networks," in *Proc. of the ACM Symposium on Applied Computing (ACM SAC'97)*, pp.123-130 (1997).

[7] ORACLE 社: "Pro*C/C++プリコンパイラ・プログラマーズ・ガイド" (1995).

[8] R.Ramakrishnan, P.Seshadri, D.Srivastava and S.Sudarshan: "THE CORAL USER MANUAL: A Tutorial Introduction to CORAL," http://www.cs.indiana.edu/database/Coral/Coral.html

[9] M.A.Roth and S.J.V.Horn: "Database Compression," in *ACM SIGMOD RECORD*, Vol.22, No.3, pp.31-39 (1993).