データベースシステム　122－44
（2000. 7. 27）

ＤＢ２－ＳＱＬによる分類階層相関ルールマイニング

吉澤　剛士☆★　　イコ　プラムディオノ★　　喜連川　優★

★東京大学生産技術研究所　〒 106-8558　東京都港区六本木 7-22-1　TEL:03-3402-6231
☆日本アイ・ビー・エム株式会社　〒261-8522 千葉県千葉市美浜区中瀬 1-1 TEL:043-297-5935
E-mail:{yoshi,iko,kitsure}@tkl.iis.u-tokyo.ac.jp

あらまし：蓄積されたデータベースが大きくなるにつれてそのデータベースから隠された付加価値を発掘するデータマイニングの重要性が広く認識されるようになった。一方現在のデータベースの主流は関係データベースシステムであるのでＳＱＬを用いたデータマイニングの実現が広く望まれるが、ＳＱＬベースのデータマイニングは専用プログラムや市販のマイニングツールに比べて性能の面では劣ることが知られている。本稿では商用データベース上でのＳＱＬベースのデータマイニングの評価を報告する。分類階層を考慮したＳＱＬベースの相関ルールマイニングが商用パラレルデータベース上でも効果があることを検証した。
キーワード：データマイニング、パラレルＳＱＬ、分類階層、商用パラレルデータベース

DB2-SQL Based Association Rule Mining with Taxonomy

Takeshi Yoshizawa☆★, Iko Pramudiono★, Masaru Kitsuregawa★

★Institute of Industrial Science, The University of Tokyo 7-22-1 Roppongi, Minato-ku, Tokyo 106, Japan
☆IBM Japan Co.,Ltd. 1-1, Nakase, Mihama-ku, Chiba-shi, Chiba 261-8522, Japan
{yoshi,iko,kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract. Data mining is becoming increasingly important since the size of databases grows even larger and the need to explore hidden rules from the databases becomes widely recognized. Currently database systems are dominated by relational database and the ability to perform data mining using standard SQL queries will definitely ease implementation of data mining. However the performance of SQL based data mining is known to fall behind specialized implementation and expensive mining tools being on sale. In this paper we present an evaluation of SQL based data mining on commercial RDBMS (IBM DB2 UDB EEE). We prove that SQL based association rule mining with taxonomy can achieve sufficient performance.
Keywords: data mining, parallel SQL, taxonomy, commercial parallel RDBMS

1．はじめに

集約されたものではなく、トランザクション単位に蓄積された膨大なデータから隠された付加価値となるような関係を抽出することは研究の世界のみならず、ビジネスの世界でも大いに注目を集め、実際にかなりの投資対効果も出てきている。その背景には近年増え続けるデータベースのデータ量と競争に勝ち抜くためにデータウェアハウスに代表されるような情報の活用が不可欠という認識がある。

このようなデータマイニングの処理にはマーケット・バスケット分析（ＭＢＡ）などに代表される相関ルールの抽出が挙げられる。但し処理負荷が非常に重いことから効率的な処理を目指していくつかのアルゴリズムが提案されてきており［１］［２］、実装方式としては専用のプログラムかもしくは高価な市販のマイニング・ツールに頼らざるを得ない状況となっている。

一方、解析対象データのほとんどは関係データベースシステムによって管理されていることから関係データベース自身でデータマイニングが行えることが期待される。即ち関係データベース問い合わせ言語ＳＱＬを用いた実装である。これはシステムとの親和性及びコストの面からそのメリットは大きいと考えられる。また柔軟性と移植のし易さという更なるメリットも秘めている。そのため最近ではＳＱＬ及びその拡張を用いたデータマイニングが研究されている[3]［４］。しかし、一般的にはパフォーマンスの面では専用プログラムや市販のマイニング・ツールにかなり劣っていることが知られている。

この課題を克服するために、以前幾つかのＳＱＬ問い合わせの改良を試み、商用パラレルデータベース（ＩＢＭのＤＢ２　ＵＤＢ　ＥＥＥ）上にＳＱＬ処理系を用いたデータマイニングを実装し、チューニングと性能評価を行った[5]。今回は更に商用パラレルデータベースにおける分類階層を用いたＳＱＬベース相関ルールマイニングの有効性を検証する。[8]にはＳＱＬを用いた同様な実験が報告されているが、並列化は行われなかった。

2．分類階層構造を考慮した相関ルール

分類階層構造とはアイテム及びカテゴリー間の階層を定義するものである[7]。カテゴリーは通常グループ化されたフィールド値を表すが、１つの値だけで構成されることもある。相関ルールマイニングにおいて分類階層構造を使用するとカテゴリー間、フィールド値間、及びカテゴリーとフィールド値間の関係を検索することになる。カテゴリーをより多く定義するほど相関ルールマイニングの結果はより詳細となりより興味深いルールを発見することができる。

今回は簡便のため階層構造で上位アイテムに位置するものを祖先（ＡＮＣ）、下位アイテムに位置するものを子孫（ＤＥＳＣ）と呼ぶことにする。分類階層テーブルＴＡＸＯＮＯＭＹは（ＤＥＳＣ、ＡＮＣ）という形を持つ。

3．ＳＱＬによる分類階層相関ルールマイニング

分類階層構造を考慮した相関ルールマイニングのアルゴリズムは基本的に通常の相関ルールマイニングと変わりはないが、全てのアイテムの祖先も加えなければならないため、処理時間は一般的には増大する。しかし分類階層の性質を利用して処理量を減らすことも可能である。[6]ではＴＨ－ＳＱＬが提案され、以下の最適化を図っている。

1．　祖先と子孫の両方を含むアイテム集合は取り除く。祖先と子孫を含むルールは常に正しいので冗長である。これによりトランザクションデータの絞り込みが可能となる。TH--SQL ではこの最適化はパス２の候補アイテムセットＣ２の作成に行われる。

2．　候補アイテム集合Ｃ_ｋを作成し、Ａｐｒｉｏｒｉ[2]のような候補絞込みを施す。[7]はこの絞込みによって分類階層テーブルの参照がパス２だけで行えることを示した。

分類階層構造を考慮する相関ルールマイニングのＳＱＬ問い合わせを図１に示す。パス１で分類階層テーブルを用いてそれぞれのアイテムの祖先を全てトランザクションデータに加え、Ｒ１テーブルに収める。Ｒ１テーブル内のアイテム集合を数え上げ、最小支持度を満たすものはラージアイテム集合Ｆ１に入れる。

他のパスでは前パスのラージアイテム集合からトランザクションデータをフィルタリングし、ＲＴＭＰ_k テーブルに入れる。候補アイテム集合ＣＣ_ｋも前パスのラージアイテム集合 F_(k-1)を自分自身にジョインさせて作成されるが、さらにＡｐｒｉｏｒｉのように候補アイテム集合の全てのサブセットを F_(k-1)を用いてチェックし、F_(k-1)にないサブセットを持つ候補アイテム集合を排除する。またパス２では候補アイテム集合Ｃ２を作成する際、第１最適化を施す。最後に候補アイテム集合に含まれるアイテム集合をテーブルＲ_ｋに入れ、最小支持度を満たすものはパスｋのラージアイテム集合Ｆ_ｋに収める。候補アイテム集合がなくなるまで繰り返される。

```
CREATE TABLE SALES (id int, item int);
CREATE TABLE TAXONOMY (desc int, anc int);

--PASS 1
CREATE TABLE F_1 (item_1 int, cnt int);
CREATE TABLE R_1 (id int, item_1 int);
INSERT INTO R_1
    (SELECT   p.id, p.item
    FROM      SALES p)  UNION
    (SELECT DISTINCT p.id, p.anc
    FROM      SALES p, TAXONOMY t
    WHERE     p.item = t.desc);
INSERT INTO F_1
    SELECT    item AS item_1, COUNT(*)
    FROM      R_1
    GROUP BY item
    HAVING COUNT(*) >= :min_support;

INSERT INTO C_2
  (  SELECT   p.item1 AS item1, q.item1 AS item2
     FROM     F_1 p, F_1 q
     WHERE    p.item1 < q.item1 )
   EXCEPT
  (SELECT anc, desc FROM TAXONOMY
      UNION SELECT desc, anc FROM TAXONOMY);
-PASS k
CREATE TABLE RTMP_k (id int, item_1 int,
    item_2 int, ..., item_(k-1) int);
CREATE TABLE C_k (item_1 int, item_2 int, ... ,item_k int);
CREATE TABLE F_k (item_1 int, item_2 int, ... ,item_k int, cnt int);
CREATE TABLE R_k (id int, item_1 int, item_2 int, ... , item_k int);

INSERT INTO RTMP_k
    SELECT   p.id, p.item_1, p.item_2, ... , p.item_(k-1)
    FROM     R_(k-1) p, F_(k-1) c
    WHERE    p.item_1 = c.item_1
```

```
    AND    p.item_2 = c.item_2
                    .
                    .
    AND    p.item_(k-1) = c.item_(k-1);

INSERT INTO C_k    --for k > 2
    SELECT i1.item1, i1. item2, ..., i1.item_(k-1), i2.item_(k-1)
    FROM F_(k-1) i1, F_(k-1) i2, ..., F_(k-1) I_k
    WHERE i1.item1 = i2.item1
      AND i1.item2 = i2.item2
                    .
                    .
      AND i1.item_(k-1) < i2.item_(k-1)
-- pruning candidates
      AND i1.item2 = i3.item1   -- skip item1
                    .
                    .
      AND i1.item_(k-1) = i3.item_(k-2)
      AND i2.item_(k-1) = i3.item_(k-1)
                    .
                    .
      AND i1.item1 = I_k.item1   -- skip item_(k-2)
                    .
                    .
      AND i1.item_(k-3) = I_k.item_(k-3)
      AND i1.item_(k-1) = I_k.item_(k-2)
      AND i2.item_(k-1) = I_k.item_(k-1)
INSERT INTO R_k
    SELECT   p.id , p.item_1 , p.item_2, ... , p.item_k-1, q.item_k-1
    FROM     RTMP_k p, RTMP_k q, C_k c
    WHERE    p.id    = q.id
      AND    p.item_1 = q.item_1
      AND    p.item_1 = c.item_1
                    .
                    .
      AND    p.item_k-1 =c.item_k-1
      AND    q.item_k-1 =c.item_k
      AND    p.item_k-1 < q.item_k-1;
INSERT INTO F_k
    SELECT item_1, item_2, ... , item_k,
                COUNT(*)
    FROM    R_k
    GROUP BY item_1, item_2, ... , item_k
    HAVING COUNT(*) >= :min-support;
DROP TABLE R_k-1;
DROP TABLE RTMP_k;
```

図１：TH-SQL による分類階層相関ﾙｰﾙﾏｲﾆﾝｸﾞ

４．実行環境

使用したマシンはＩＢＭの RS/6000ＳＰ２（１２ノード）で、この上にＩＢＭ製パラレルデータベース（ＤＢ２　ＵＤＢ　ＥＥＥ　Ｖ５．２）を乗せて実験を行った。構成を表１に示す。

CPU	POWER2 77MHz
Main Memory	256MB
OS	AIX V4.3.3
Disk	SCSI hard disk 4.4GB
Network	HPS with 100MB/s
DB	IBM UDB EEE V5.2

表１：マシン構成表

５．データセット

この実験で用いたデータは Apriori アルゴリズムと共に報告されたプログラムで作成された[２]。そのパラメータは表２の通りである。トランザクションデータ及び分類階層テーブルは共にハッシングのロジックにより各ノードに分割される。

Number of records in transaction table	109322
Number of transactions	20000
Average transaction length	5
Number of records in taxonomy table	141704
Number of items	30000
Number of roots	50
Number of levels	4
Average fanout	5

表２：データセットパラメータ

６．性能評価

図２は並列化処理ノードの数を変化させていった場合の実行処理時間を示す。分類階層テーブルを処理するための並列化オーバーヘッドは予想していた程効率への影響は出ていない様子である。逆に今回分類階層テーブルのオーバーヘッドを減らす目的で分類階層テーブルを１ノードにのみに作成し、実行時に全ノードへブロードキャストさせるアクセスプランと分類階層テーブルを各ノードに複製を保持させる UDB の REPLICATED 機能を用いてみたが、結果は分類階層テーブルをハッシュにより各ノードに分散させ実行処理時に必要なノードへ DISTRIBUTE させるやり方と殆どどちらも処理時間に違いは出なかった。これは今回の分類階層テーブルのサイズ及びＣ２を作成するまでしか用いない SQL ロジックに起因しているものと思われる。ＴＡＸＯＮＯＭＹテーブルとジョインさせるＦ１テーブルが小さいため、Ｃ２テーブル作成のジョイン処理はＴＡＸＯＮＯＭＹテーブルの配置にあまり影響されないと思われる。また高速のスイッチ経由でノード間のデータを受け渡せるネットワーク構成の影響も大きいと思われる。[6]ではＴＴＲ－ＳＱＬと呼ばれるＳＱＬクエリーの場合、ＴＡＸＯＮＯＭＹテーブルから作成された分類階層テーブルが各パスｋのＲ_ｋテーブルに読み込まれるため、配置によって処理時間が変わる可能性があり、これから検討する課題である。

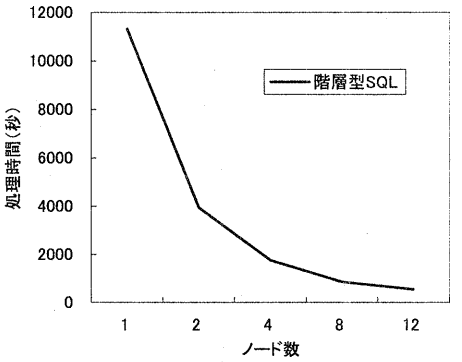

図２：処理ノード数による実行時間

図３は各 PASS 毎の処理時間の結果を示す。以前性能評価をした場合[5]と同様に、候補アイテム数が最も多いパス２の処理が実行時間の大半を占めることが伺えた。

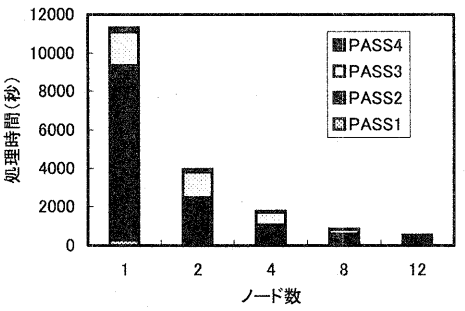

図３：各 PASS 毎の処理時間

7．まとめ

今回分類階層を考慮した相関ルールマイニングのＳＱＬについて商用パラレルデータベースを用いた場合の処理時間を確認した。この結果は処理時間そのものは以前 Kitsuregawa[６]らが独自に開発したＰＣクラスタ上で行った実験結果や相関ルールマイニング専用のＣ実装プログラムに比べれば劣るものの、パラレル処理におけるスケーラビリティー効果は商用データベースにおいても有効である結果となった。これにより商用パラレルデータベースのオプティマイザーは通常の相関ルールマイニング SQL と同様に分類階層を考慮した相関ルールマイニングにも対応していることを確認することが出来た。但し処理の詳細については今後の検討課題とする。

更に商用パラレルデータベースの主流になりつつあるイントラパラレル処理を用いた SMP マシンにおける処理時間及びスケーラビリティーの検証も今後行なって行きたい。

参考文献

[1] R.Agrawal,T.Imielinski,A.Swami.Mining Asociation Rules between Sets of Items in Large Databases. In Proc. Of the ACM SIGMOD Conference on Management of Data,1993.

[2] R.Agrawal,R.Srikant. Fast Algorithms for Mining Association Rules. In Proc. of the VLDB conference,1994.

[3] M.Houtsma, A.Swami. Set-oriented Mining of Association Rules. In Proc. of International Conference on Data Engineering, 1995.

[4] S.Sarawagi,S.Thomas, R.Agrawal. Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications. In Proc. of the ACM SIGMOD Conference on Management of Data, 1998.

[5] Takeshi Yoshizawa, Iko Pramudiono, Masaru Kitsuregawa. SQL Based Association Rule Mining using Commercial RDBMS (IBM DB2 UDB EEE). In proc. of the Data Warehousing and Knowledge Discovery, 2000（掲載予定）

[6] Iko Pramudiono, Takahiko Shintani, Takayuki Tamura, Masaru Kitsuregawa. Parallel SQL Based Association Rule Mining on Large Scale PC Cluster: Performance Comparison with Directly Coded C Implementation. In Proc. of First International Conference on Data Warehousing and Knowledge Discovery (DAWAK99), 1999

[7] R.Srikant, R.Agrawal. Mining Generalized Association Rules. In Proc. of the VLDB conference,1995.

[8] S.Thomas, S.Sarawagi. Mining Generalized Association Rules and Sequential Patterns Using SQL Queries.In Proc. of KDD, 1998.