Partitioned Approach to Association Rule Mining over Multiple Databases*

Himavalli Kona and Sharma Chakravarthy

CSE Department, The University of Texas at Arlington
{hima,sharma}@cse.uta.edu

Abstract. Database mining is the process of extracting interesting and previously unknown patterns and correlations from data stored in Data Base Management Systems (DBMSs). Association rule mining is the process of discovering items, which tend to occur together in transactions. If the data to be mined were stored as relations in multiple databases, instead of moving data from one database to another, a partitioned approach would be appropriate. This paper addresses the partitioned approach to association rule mining for data stored in multiple Relational DBMSs. This paper proposes an approach that is very effective for partitioned databases as compared to the main memory partitioned approach. Our approach uses SQL-based K-way join algorithm and its optimizations. A second alternative that trades accuracy for performance is also presented. Our results indicate that beyond a certain size of data sets, the accuracy is preserved in addition to improving performance. Extensive experiments have been performed and results are presented for the two partitioned approaches using IBM DB2/UDB and Oracle 8i.

1 Introduction

Association rule mining [3, 4, 5] makes correlation between items that are grouped into transactions deducing rules that define relationships between itemsets. Here an attempt is made to identify if customer who buys item 'A' also buys item 'B'. Association rules are of the form A => B where A is the antecedent and B is the consequent. Several association rule algorithms have been proposed that work on data in a file [5, 6, 7]. Data base approach to association rules using SQL have been explored as well [8, 9, 11, 13, 14].

Parallel mining algorithms have been developed to overcome the limitations of main memory approaches by using the aggregate power and memory of many processors. A multi-database system [1, 2] is a federation of autonomous and heterogeneous database systems. Data is distributed over multiple databases (typically 2 or 3) in many organizations. Each of the databases may get updated frequently and independently. Most of the organizations today have multiple data sources distributed at differ-

* This work was supported, in part, by the Office of Naval Research, the SPAWAR System Center-San Diego & by the Rome Laboratory (grant F30602-02-2-0134), and by NSF (grants IIS-0123730 and ITR 0121297). We would like to acknowledge the contributions of Akshay Arora in improving the readability of this paper.

Y. Kambayashi et al. (Eds.): DaWaK 2004, LNCS 3181, pp. 320–330, 2004.

ent locations, which need to be analyzed to generate interesting patterns and rules. An effective way to deal with multiple data sources (where data to be mined is distributed among several relations on different DBMSs) is to mine the association rules at different sources and forward the rules to other systems rather than sending the data to be mined which is likely to be very large. Interactive mining has been proposed as a way to bring decision makers into the loop to enhance the utility of mining and to support goal oriented mining. Partitioned and incremental approaches can be applied to each of the data sources independently, which would require the transmission of intermediate results between the databases.

If the raw data from each of the local databases were sent to a common database for mining and generation of rules, certain useful rules, which would aid in making decisions about local datasets, would be lost. For example a rule such as "50% of the branches in the north saw a 10% increase in the purchase of printers when digital cameras and memory cards were purchased together" would not be generated if the raw data was transferred and processed as a whole. In such a case the organization may miss out certain rules that were prominent in certain branches and were not found in the other branches as in the above example. Generating such rules would aid in making decisions at each branch independently.

This paper addresses the problem of directly mining data stored in multiple relations or multiple databases. The main memory approaches are not applicable here unless data is siphoned out of each relation/database which is what we are trying to avoid in the first place. The cost models for data transfer in our case are very different from that of partitioned main memory algorithms. Furthermore, the utility of mining each data set independently as well as together is important for many applications. In this paper, we address the partitioned approaches to discover association rules on data residing in multiple databases. Although incremental approaches have been developed, we will not discuss them for lack of space. Please refer to [17] for details.

The rest of the paper is organized as follows: In Section 2 discusses some of the related work in this area. Section 3 discusses the inefficiency of the main memory partitioned approach when directly used for multiple databases. Section 4 presents our approaches and their performance evaluation. Section 5 has conclusions.

2 Related Work

The partition algorithm for association rules presented in [7] makes at most two passes over the input database to generate the rules. In the first phase of the algorithm the database is divided into non-overlapping partitions and each of the partitions are mined individually to generate the local frequent itemsets. At the end of the first phase the local frequent itemsets are merged to generate the global candidate itemsets. In the second phase of the algorithm a TIDLIST is created and used to calculate the support of all the itemsets in the global candidate itemset to generate the global frequent itemsets. The algorithm could be executed in parallel to utilize the capacity of many processors with each processor generating the rules.

SQL-Based approaches map transactions into relations with (TID, Item) attribute format [3, 13]. The (TID, Itemlist) format was not chosen because the number of items for a particular transaction may exceed the number of attributes a DBMS can support. SQL-92 and SQL-OR approaches [13, 14, 16] were introduced for mining

data in relational DBMSs. The SQL-92 based approaches correspond to k-way join and its optimizations such as the Second Pass Optimization (SPO), Reuse of Item Combinations (RIC) and Prune the input table (PI). It has been shown [11, 16] that, of all the SQL-based approaches, K-way join and its optimizations have the best performance (as compared to query-sub query and group by approaches).

Most of the times data is already stored as relations in different databases belonging to the same organization. If one were to mine the data present in multiple databases, there are two options. [2] presents a weighted model for synthesizing high-frequency association rules from different sources. A high-frequency rule is the one that is supported by most of the data sources. The proposed model assigns a high-weight to a data source that supports/votes more high-frequency rules and a lower weight to a data source that supports/votes less high-frequency rules. [1] discusses a new multi-database mining process. The patterns in multi-databases are divided into the three classes as Local patterns, high-vote patterns and Exceptional patterns. The mining strategy identifies two types of patterns, high-vote patterns and exceptional patterns. The discovery of these patterns can capture certain distributions of local patterns and assist global decision-making within a large company.

3 Performance of the Partitioned Approach

The partition algorithm [7] is an efficient main memory algorithm for mining association rules in large data sets that could be partitioned as needed. This approach when used for multiple databases results in poor performance as will be shown in the paper. For its implementation, relational operations (using SQL) were used. There was no change in Phase I of the algorithm. Each database was considered an individual partition and the frequent itemsets were generated for each of the databases. In Phase II of the algorithm the frequent itemsets from each of the partitions were merged to form two sets of itemsets. The first set is the global frequent itemsets, which correspond to itemsets that are large in all the partitions (databases). The second set is the set of global candidate itemsets, which is the union of all the frequent itemsets from each of the partitions (not included in the global frequent itemsets). A TIDLIST is created for the entire database. As the data is assumed to be distributed over different databases, each partition needs to be shipped to a single database to create the TIDLIST. The TIDLIST was used for counting the support of the itemsets in the global candidate itemsets and the itemsets satisfying the user specified support were added to the set of global frequent itemsets.

3.1 Methodology for Experiments

The performance results presented in this paper are based on datasets generated synthetically using IBM's data-generator. The nomenclature of these datasets is of the form "TxxIyyDzzzK", where "xx" denotes the average number of items present per transaction, "yy" denotes the average support of each item in the dataset and "zzzK" denotes the total number of transactions in "K"(1000s). The experiments have been performed on Oracle 8i and IBM DB2 / UDB V7.2 (installed on a machine running Microsoft Windows 2000 Server with 512MB of RAM). Each experiment has been

performed 4 times. The values from the first run are ignored so as to avoid the effect of the previous experiments and other database setups. The average of the next 3 runs was used for analysis to avoid any false reporting of time due to system overload or other factors. For the purpose of reporting experimental results in this paper, due to space constraints, we have shown the results only for three datasets – T5I2D100K, T5I2D500K, and T5I2D1000K.

Figure 1. shows the performance of the TIDLIST approach for a T5I2D1000K dataset. The dataset is divided into two equal partitions each of size 500K. The analysis of the time taken for the different phases shows that the Phase II is the most time consuming. In Phase II, the TIDLIST is created for the whole dataset. The TIDLIST creation time increases exponentially as the size of the dataset increases. The partitioned approach although seems to work well for main memory databases, its performance for partitioned databases is not acceptable. The creation of the TIDLIST and the shipping of the partitions to a single database need to be avoided.

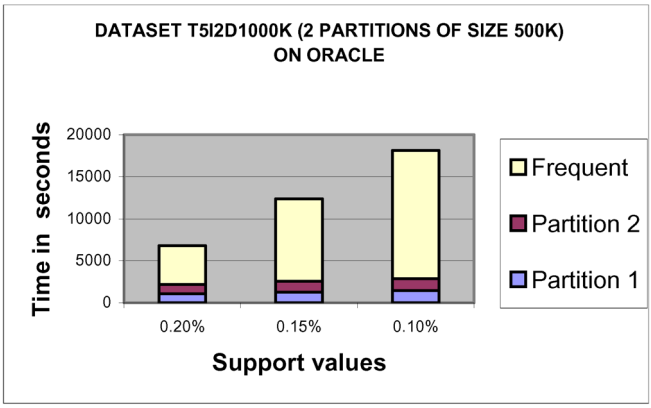

Fig. 1. Performance Of TIDLIST Approach On T5I2D1000K Dataset

4 Proposed Algorithms for Multiple Databases

This section discusses two partitioned approaches – DB-Partition I (or approach I in figures) and DB-partition II (or approach II in figures) – that have been developed and tested for multiple databases. Notations used in the remainder of this paper are shown in Table 1.

Table 1. Notation used for the Partitioned Approach

Notation	Meaning
C^P_K	Set of local candidate k-itemsets in partition P.
F^P_K	Set of local frequent k-itemsets in partition P.
C^G_K	Set of global candidate k-itemsets.
F^G_K	Set of global frequent k-itemsets.
$NBd(F^P_K)$	Set of local non-frequent k-itemsets in partition P.

The negative border of frequent k-itemsets corresponds to those itemsets that did not satisfy the support in pass k. That is, $NBd(F^P_K) = C^P_K - F^P_K$. Given a collection $F \subseteq P(R)$ of sets, closed with respect to set inclusion relation, the NBd(F) of F consists of the minimal itemsets $X \subseteq R$ not in F.

4.1 DB-Partition I

In the TIDLIST approach, the TIDLIST was created as a CLOB. In DB-partition I, the TIDLIST is not at all created and the k-way join approach is used instead. Some of the k-way join optimizations reported in [14, 16] have been used. The two k-way join optimizations used are: Second-pass Optimization (SPO) and Reuse of Item Combinations (RIC). In a multiple database scenario, each of the individual databases is considered as a partition and the merging is done by choosing one of the databases. The changes made to the partition algorithm are described below.

Phase I: In this phase the frequent itemsets F^P_K are generated for each of the partitions. Along with the frequent itemsets in each of the partitions, the negative border of the frequent 2-itemsets $NBd(F^P_2)$ is also retained. These itemsets are used for counting the support in the Phase II of the algorithm. Only the negative border of the 2-itemsets is retained because when the second pass optimization is used, the generation of the 2-itemsets is the first step in each partition. Since the 2-itmeset generation is the first pass, there is no loss of information and the negative border of the 2-itemsets will have all possible 2-itemsets, which did not satisfy the support.

After the frequent itemsets from all the partitions (databases) are generated, the frequent itemsets and the negative border of the frequent 2-itemsets from all the partitions are shipped to one of the databases to do the remaining computation. This step is shown as an edge with label "1" in Figure 2. Merging the frequent itemsets from all the partitions generates the global candidate itemsets $C^G_1, C^G_2, \ldots, C^G_K$.

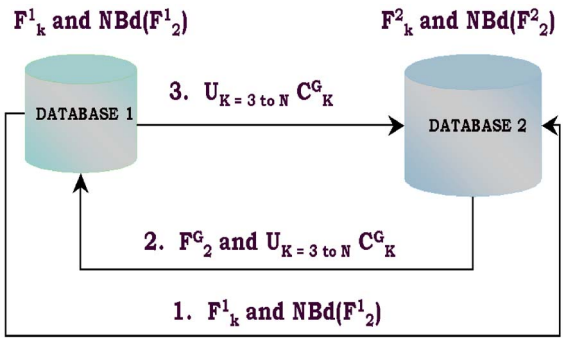

Fig. 2. Data Transfer Using DB-partition I

Phase II: In this phase, the global frequent itemsets – itemsets that are large in all the partitions – are generated. Merging the count obtained from the negative border and the frequent 2-itemsets from all the partitions generates the count for the remaining 2-itemsets in C^G_2. The itemsets satisfying the support are added to F^G_2. F^G_2 and

$\cup_{k=3 \text{ to } n} C^G_K$ are shipped to all the databases to generate the counts of the remaining candidate itemsets. This is shown as an edge with label "2" in Figure 2.

Each of the databases generates a materialized table from the global frequent 2-itemsets using the *Reuse of item combination* optimization. The materialized table is used in the successive passes to generate the counts of the itemsets in the global candidate itemsets. Once the counts are generated in all the partitions they are shipped back to one database to do the final counting. This is shown as an edge with label "3" in Figure 2.

Figure 2 shows the data transferred in each of the steps. Database 1 and Database 2 are considered the 2 partitions. Database 2 is chosen for merging the frequent itemsets from all the partitions to global candidate itemset and for generating the final cumulative count of all frequent itemsets obtained from all the partitions in step "3".

Performance Analysis: Performance experiments were done on datasets of different sizes. Each data set divided into 2 or 3 non-overlapping partitions. Figure 3. shows the performance of a T5I2D1000K dataset divided into 2 equal sized partitions each of size 500K. It can be seen from the graph that the improvement in performance in DB-partition I compared to TID LIST approaches 58% for a support values of 0.20% and the improvement increases to about 78% for a support value of 0.10%. As the support value decreases the percentage improvement in the performance increases.

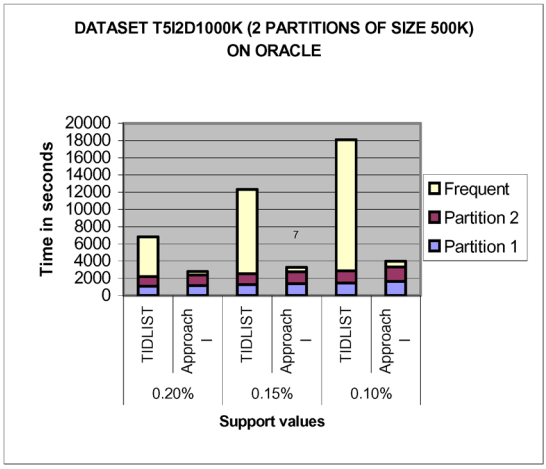

Fig. 3. Performance Comparison Of TIDLIST And DB-partition I

Figure 4. shows the data transfer when there are 3 partitions. At the end of each phase the intermediate results are transferred to one of the partitions to do the remaining computations.

The performance for T5I2D500K is shown in Figure 5. The dataset is divided into 3 partitions of size 200K, 200K and 100K. The performance is shown for Oracle and DB2. For DB2 the percentage improvement in performance decreases from 80% to 53% as the support value decreases from 0.30% to 0.20%. The performance in Oracle shows an increase from 18% to 75% as the support value decreases from 0.20% to 0.10%.

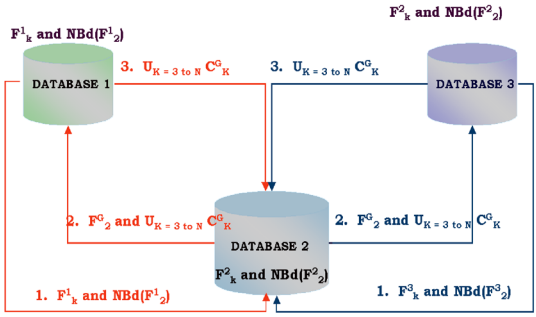

Fig. 4. Data Transfer Using DB-partition I For 3 Partitions

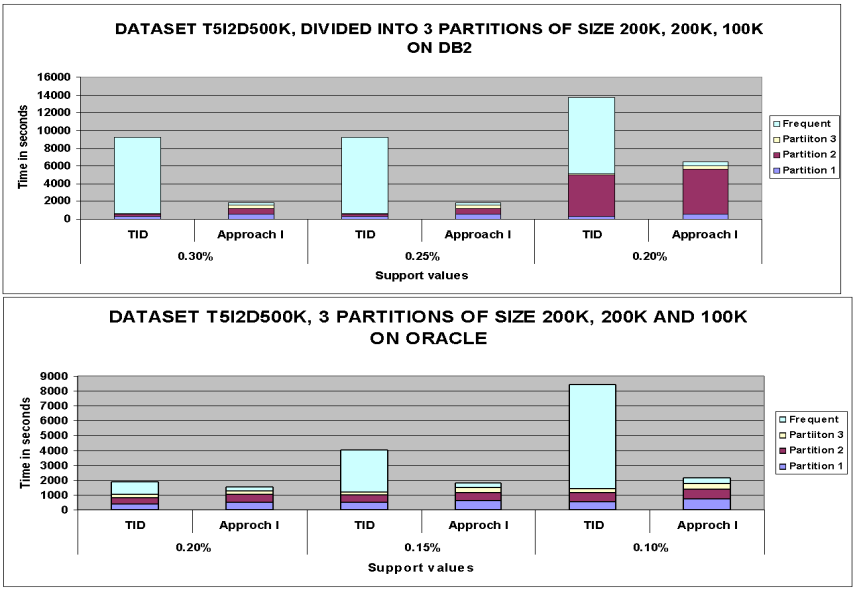

Fig. 5. Performance Comparison Of T5I2D500K For TIDLIST And DB-partition I

Data Transfer: Table 2 shows the number of records transferred between the databases in each step. The input data denotes the transactional data. It is assumed that the dataset is divided into 2 equal sized partitions. The numbers in the Table 2 indicate the number of records transferred. For example, for the T5I2D10K dataset the input data has 27000 records and the total records transferred using DB-partition I between the two databases is 51845. It is observed that transferring the intermediate results is better for the datasets, which have more than 100K transactions (which is typically the case).

In DB-partition I only the negative border of the frequent 2-itemsets was retained in all the partitions. In phase II, a materialized table was created to do the support counting. The time taken to create a materialized table increases as the size of the dataset increases. In this approach the data is transferred 3 times between the partitions. DB-partition II alternative was proposed to overcome the above drawbacks.

Table 2. Data Transferred Using DB_partition I

Dataset	Step 1 $[F^1_K + NBd(F^1_2)]$	Step 2 $[F^G_2 + U_{K=3 \text{ to } N} C^G_K]$	Step 3 $U_{K=3 \text{ to } N} C^G_K$	Total records
T5I2D10K	50360	1393	92	51845
T5I2D100K	199455	678	101	200234
T5I2D500K	319677	638	76	320391
T5I2D1000K	356696	632	75	357403

4.2 DB-Partition II

During the Phase I of this approach, the negative border of all the frequent itemsets in each of the partitions are retained as compared to the previous approach where only the negative border of the frequent 2-itemsets were retained. When the frequent itemsets are generated the data is transferred to one of the partitions (or databases) to form the global candidate itemsets and the global frequent itemsets. The global frequent k-itemsets are generated by merging the counts of the frequent k-itemsets and the negative border of the frequent k-itemsets. Figure 6 shows the data transfer in DB-partition II.

DB-partition II is different from DB-partition I with regard to the number of times data is transferred between the databases and the itemsets that are retained. In DB-partition I, only the negative border of the frequent 2-itemsets is retained. Since retaining the negative border does not require any additional computation, in DB-partition II, the negative border of all the frequent itemsets are retained for all the databases. In Phase II of DB-partition I, the global frequent 2-itemsets are generated using the local frequent 2-itemsets and their negative border from all the databases. The results have to be transferred to the individual databases to generate the remaining (3 to k) – itemsets, which requires the scanning the input data in each of the databases to generate the counts. But in DB-partition II, all the global frequent itemsets are generated using the local frequent itemsets and their negative border from all the databases. An additional scan of the database is **not** required and the intermediate results are transferred only once as compared to 3 times in DB-partitition I.

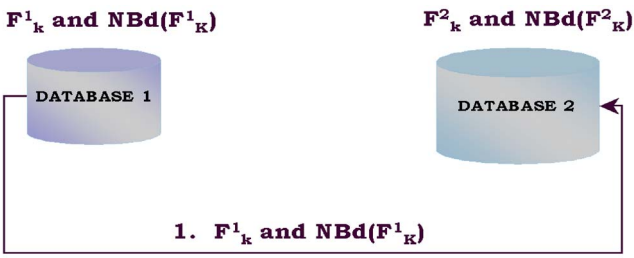

Fig. 6. Data Transfer In DB-partition II

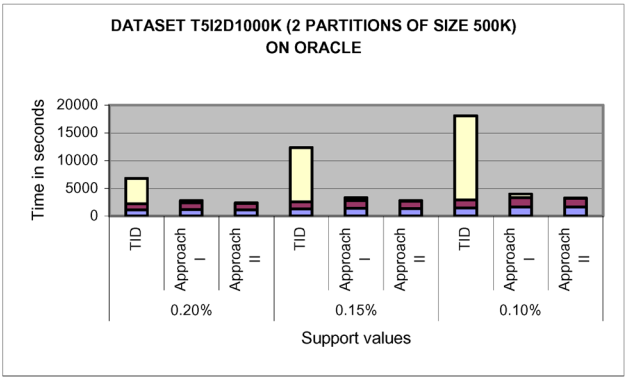

Fig. 7. Performance Comparison Of All The Approaches With 2 Partitions

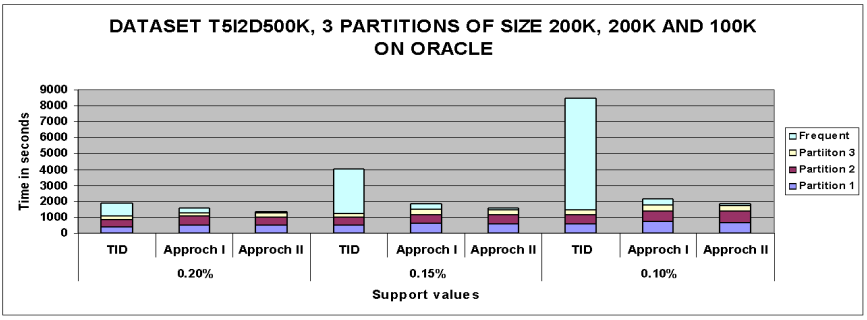

Fig. 8. Performance Comparison Of All The Approaches With 3 Partitions

Figure 7 shows the performance comparison of the TIDLIST, DB-partition I and DB-partition II. A T5I2D1000K dataset was divided into 2 partitions of size 500K each. From the graph it is noted that the performance of DB-partition II improved from 16% to 18% as the support value decreased from 0.20% to 0.10%. Figure 8 shows a similar performance graph for 3 partitions.

We compared the data transfer between DB-partition I and DB-partition II (Table 3) and they were not much different as the size of the data is large for the second pass (2-itemsets). It was noted that DB-partition II performed better than TIDLIST and DB-partition I for almost all the cases. This was because the creation of materialized table was eliminated and retaining the negative border does not require any additional computation. However there is a tradeoff associated with the DB-partition II. This approach may miss out the count of itemsets, which are globally large but locally small in a few partitions. The count of some k-itemsets whose subset did not appear either in the frequent itemsets or its negative border in the earlier passes may be missed. Our intuition was that this was the case only for small data sizes that too when the support is extremely low.

In order to verify this, we conducted experiments using DB-partition II for different data sizes. Figure 9 shows the error observed in the frequent itemsets generated. The frequent itemsets generated using the TIDLIST approach and DB-partition I was compared with the itemsets generated in DB-partition II. This approach showed some

error in the number of frequent itemsets generated in each pass. However, it was seen that there was some error only for the smaller datasets with lower support values. No error was noted beyond datasets of size T5I2D100K and above.

Table 3. Data Transfer For DB-partition II

Dataset	Input data Records	Step 1 $[F^1_K + NBd(F^1_K)]$ records	DB-partition II
T5I2D10K	27000	50481	50481
T5I2D100K	273000	199521	199521
T5I2D500K	1368500	319740	319740
T5I2D1000K	2736000	356761	356761

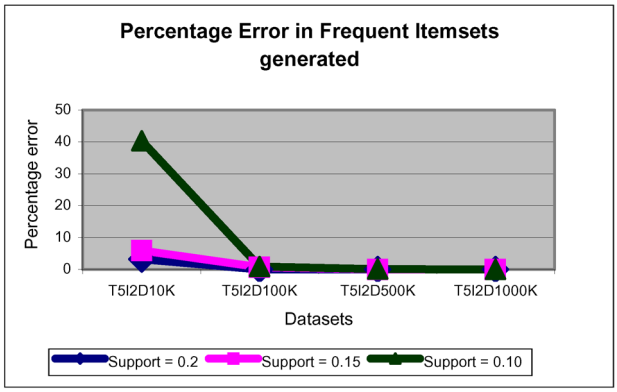

Fig. 9. Error Analysis

5 Conclusions

In this paper, we have focused on the partitioned approach to association rule mining that can be used for multiple databases. The partitioned approach to association rule mining is appropriate for mining data stored in multiple DBMSs. The partition algorithm proposed in this paper provides an efficient way of discovering association rules in large multi-database. This paper presented two approaches – DB-partition I and DB-partition II using the negative border concept which possesses slightly different performance characteristics. Extensive experiments have been performed for the partitioned approach on Oracle 8i and IBM DB2.

References

1. Zhang, S., X. Wu, and C. Zhang, *Multi-Database Mining.* IEEE Computational Intelligence Bulletin, Vol. 2, No. 1, 2003: p. 5-13.
2. Wu, X. and S. Zhang. *Synthesizing High-Frequency Rules from Different Data Sources*. in *IEEE TKDE* 2003.

3. Thomas, S., *Architectures and optimizations for integrating Data Mining algorithms with Database Systems*, Ph.D Thesis, *CISE*. Department 1998, University of Florida, Gainesville.
4. Thuraisingham, B., *A Primer for Understanding and Applying Data Mining.* IEEE, 2000. Vol. 2, No.1: p. 28-31.
5. Agrawal, R., T. Imielinski, and A. Swami. *Mining Association Rules between sets of items in large databases*. in Proc. of *ACM SIGMOD* 1993.
6. Agrawal, R. and R. Srikant. *Fast Algorithms for mining association rules*. in *20th Int'l Conference on Very Large Databases,*. 1994.
7. Savasere, A., E. Omiecinsky, and S. Navathe. *An efficient algorithm for mining association rules in large databases*. in *21st Int'l Cong. on Very Large Databases (VLDB).* 1995. Zurich, Switzerland.
8. Houtsma, M. and A. Swami. *Set-Oriented Mining for Association Rules in Relational Databases*. in Proc. of ICDE , 1995.
9. Meo, R., G. Psaila, and S. Ceri. *A New SQL-like Operator for Mining Association Rules*. in *Proc. of VLDB* 1996. Mumbai, India.
10. Agrawal, R. and K. Shim, *Developing tightly-coupled Data Mining Applications on a Relational Database System*. 1995, IBM Almaden Research Center: San Jose, California.
11. Sarawagi, S., S. Thomas, and R. Agrawal. *Integrating Association Rule Mining with Relational Database System: Alternatives and Implications*. in *ACM SIGMOD* 1998. Seattle, Washington.
12. Hongen, Z., *Mining and Visualization of Association Rules over Relational DBMSs*, MS Thesis *CISE Department* 2000, UFL, Gainesville.
13. Dudgikar, M., *A Layered Optimizer or Mining Association Rules over RDBMS*, MS Thesis, *CISE Dept*. 2000, University of Florida, Gainesville.
14. Mishra, P. and S. Chakravarthy, *Performance Evaluation and Analysis of SQL-92 Approaches for Association Rule Mining* in *Proc.* of *DAWAK* 2003.
15. Toivonen, H. *Sampling Large Databases for Association Rules*. *In Proc. of VLDB*. 1996.
16. Mishra, P. and S. Chakravarthy, *Evaluation of K-way Join and its variants for Association Rule Mining in Proc. of BNCOD 2002*.
17. Kona, H, *Association Rule Mining Over Multiple Databases: Partitioned and Incremental Approaches*, MS Thesis, UTA, Fall 2003.
http://www.cse.uta.edu/Research/Publications/Downloads/CSE-2003-40.pdf