APRIORI BASE EFFICIENT APPROACH FOR LARGE DATABASE COMPRESSED IN ASSOCIATION MINING

Arpana Jaiswal , Dr.Amit Khare , Rahul Moriwal
M.tech Student, Professor, Assistant Professor,
Department of Computer Science & Engineering ,
Acropolis Institute of Technology & Research, Indore, Madhya Pradesh, India.

ABSTRACT:
Data mining can be viewed as a result of the natural evolution of information technology. Large Amount of data is being using very rapidly in the world. It to be compressed takes much more time and takes lot of effort to process these data for knowledge discovery and decision making. Data compression technique is one of good solutions to be reduce size of data that can be save more time the time of discovering useful knowledge by using appropriate methods data mining approaches which is used to compress the original database into a smaller one and perform the data mining process for compressed transaction such as M2TQT, APRIORI algorithm, TM algorithm, AIS & SETM, CT-Apriori algorithm, CBMine, CT-ITL algorithm, FIUT- Tree. Among the various techniques M2TQT uses the relationship of transactions to merge related transactions and builds a quantification table to prune the candidate item sets which are impossible to become frequent in order to improve the performance of mining association rules. Thus M2TQT is observed to perform better than existing approaches.

Keyword: Quantification table, Association Mining, Marge Transitions.

I INTRODUCTION:

The current parallel and distributed algorithms are based on the serial algorithm Apriori. An excellent survey given in classifies the algorithms by load-balancing strategy, *data parallelism* and *task parallelism.* The two paradigms differ in whether the candidate set is distributed across the processors or not. In the data parallelism paradigm, each node counts the same set of candidates. May or may not be partitioned in either paradigm theoretically. In practice for more efficient I/O it is usually assumed the database is partitioned and distributed across the processors. In the data parallelism paradigm, the database is distributed across the processors. Each processor is responsible for computing the *local support counts* of all the candidates, which are the support counts in its database partition. All processors then compute the *global support counts* of the candidates, which are the total support counts of the candidates in the whole database, by exchanging the local support counts. Subsequently, large item sets are computed by each processor independently.

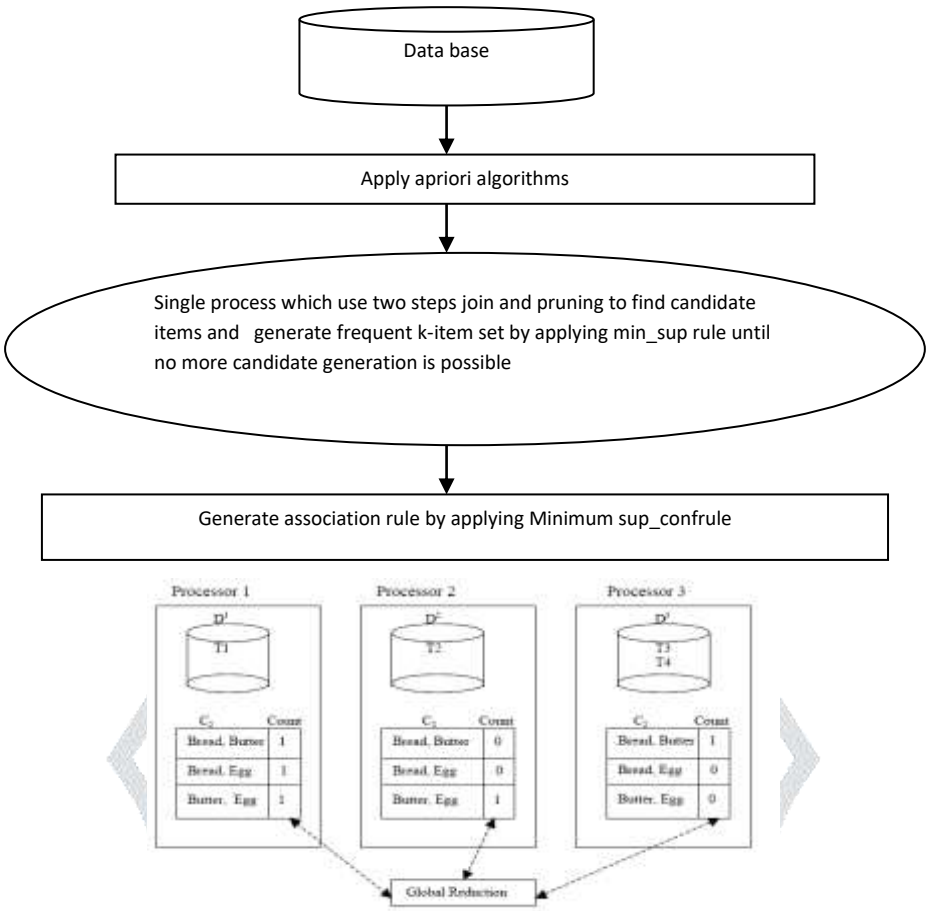

Figure 1.1 parallel and distributed

II MERGING PROCESS:

Since most data occupy a large amount of storage space, it is beneficial to reduce the data size which makes the data mining process more efficient with the same results. Compressing the transactions of databases is one way to solve the problem. . Figure 1.2 shows overview of the Merged Transaction Algorithm [2]. It is very effective to reduce the size of a transaction database. Their algorithm is divided into data preprocess and data mining.

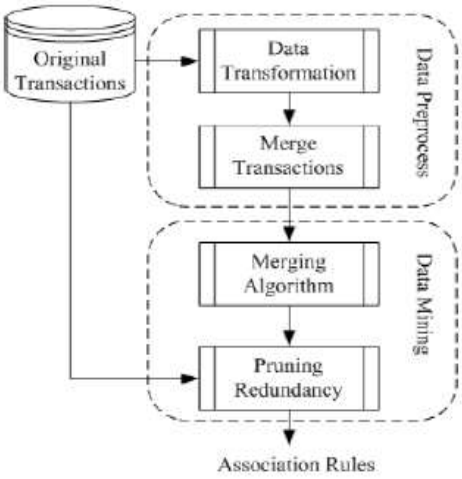

Figure 1.2 Merged Transaction Algorithms

There are two sub-processes in the data preprocess. One sub-process transforms the original database into a new data representation. It uses lexical symbols to represent raw data. Here, it's assumed that items in a transaction are sorted in lexicographic order. Another sub-process is sorting all the transactions to various groups of transactions and then merges each group into a new transaction.

III MINING MERGED TRANSACTIONS WITH THE QUANTIFICATION TABLE APPROACH:

First, Mining Merged Transactions with the Quantification Table uses the transaction relation distance to merge the relevant transactions. The definition of the transaction relation distance is defined D introduce how to build a quantification table. Then, it illustrates the process of compressing a database. Next shows how to compute support of item sets from minimum-frequency function. Finally, it explains how to recover data from the compressed database.

A .Transaction Relation Distance

Based on the relation distance between transactions one can merge transactions with closer relationship to generate a better compressed database. Here the transaction relation and transaction relation distance are defined as follows:

Definition:

(1) Transaction Relation: The relation between two different transactions T1 and T2 is that T1 is either a subset or a superset of T2.

(2) Transaction Relation Distance: Distance is the number of different items between two transactions.

B. Quantification Table

To reduce the number of candidate item sets to be generated, additional information is required to help prune non-frequent item sets.

TID	Transaction
100	ABCDE
200	CDE
300	ACD

Table 1.1: An Example Database

For instance, after reading the transaction {ABCDE} of TID 100, it knows the transaction length n is 5. For the prefix-item A, the counters under L5 to L1 are all increased by one from the initial value of zero. That is, A1 appears in each Li, where i = 5 to 1. For the prefix-item B, the counters under L4 to L1 are all increased by one as well. That is, B1 appears in each Li, where i = 4 to 1. The same process is performed for items C, D, and E. Similarly, after reading TID 200 {CDE}, the table has C2 in L3, L2, and L1; D2 in L2 and L1; E2 in L1. Finally, with the last transaction {ACD}, it will increase the counters by one from A1 to A2 in L3, L2, and L1; C2 to C3 in L2 and L1; D2 to D3 in L1. Table 2 shows the result of building the quantification table. With this table, we can easily prune the candidate item sets whose counters are smaller than the minimum support.

IV PROPOSED METHOD:

This section focuses on compressing related transactions and building a quantification table for pruning candidate itemsets that are impossible to become frequent itemsets. Algorithms like compress transactions to reduce the size of a transaction database. Then, they use Apriori-like algorithms to mine the compressed database. Whereas the two phase's approach of compression and data mining are used, they suffer the following problems:

(1) In the data compression phase, the original database cannot be recovered to support transaction updates.

(2) In the data mining phase, a lot of candidate item sets could be generated in a large transaction database.

Since both need to scan the database more than once, they have a much higher process cost. The first problem is due to the lack of rule or constraint in the process of merging transactions in the data compression phase. Therefore, the compressed database cannot be decompressed to its original form In addition, they don't use user-defined threshold to filter infrequent 1-itemsets from the original database.

Another problem is that Apriori-like algorithms generate a lot of candidate itemsets and need to check the candidate itemsets by scanning the database. It is very time-consuming.

In order to provide a better performance, we limit the number of database scan to be one in the data compression phase and build a quantification table. In the data mining phase, we use the same approach of Apriori algorithm to generate candidate itemsets and reduce the number of candidate itemsets by using the quantification table. We also reduce the time of scanning the database.

We called our approach the Mining Merged Transactions with the Quantification Table (MMTQT) which has three phases:

(1)Merge related transactions to generate a compressed database

(2) Build a quantification table

(3) Discover frequent itemsets

MMTQT Approach

First, MMTQT uses the transaction relation distance to merge the relevant transactions. The definition of the transaction relation distance is defined D introduce how to build a quantification table. Then, it illustrates the process of compressing a database. Finally, it explains how to recover data from the compressed database.

Architecture

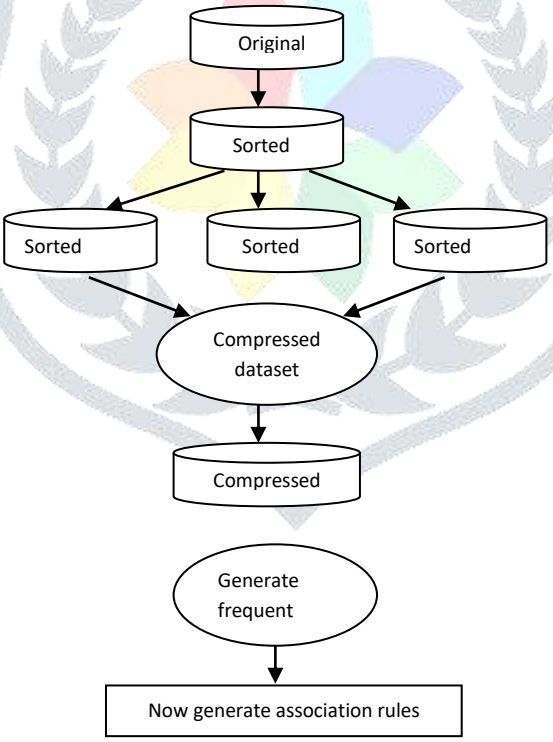

Figure 1.3 Compressed data set apriori algorithms

V CONCLUSION:

Simplify the description items in each transaction are presented in a lexicographical order. Algorithms like compress transactions to reduce the size of a transaction database. They use Apriori-like algorithms to mine the compressed database.

REFRENCES:

[1]. Jia -Yu Dai, Don – Lin Yang, Jungpin Wu and Ming-Chuan Hung," Data Mining Approach on Compressed Transactions Database" in PWASET Volume 30, pp 522-529, 2010.

[2]. M. C. Hung, S. Q. Weng, J. Wu, and D. L. Yang, "Efficient Mining of Association Rules Using Merged Transactions," in WSEAS Transactions on Computers, Issue 5, Vol. 5, pp. 916-923, 2008.
[3]. D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, and T. Yiu, "MAFIA: A maximal frequent itemset algorithm," IEEE Transactions on Knowledge and Data Engineering, Vol. 17, pp. 1490-1504, 2008.
[4]. D. Xin, J. Han, X. Yan, and H. Cheng, "Mining Compressed Frequent-Pattern Sets," in Proceedings of the 31st international conference on Very Large Data Bases, pp. 709-720, 2007.
[5]. G. Grahne and J. Zhu, "Fast algorithms for frequent itemset mining using FP-trees," IEEE Transactions on Knowledge and Data Engineering, Vol. 17, pp. 1347-1362, 2005.
[6]. M. Z. Ashrafi, D. Taniar, and K. Smith, "A Compress-Based Association Mining Algorithm for Large Dataset," in Proceedings of International Conference on Computational Science, pp. 978-987, 2003.
Ashrafi and K. Smith, “Data Compression-Based Mining Algorithm for Large Dataset," in Proceedings of International Conference on Computational Science, 2003.
[7]. D. I. Lin and Z. M. Kedem, "Pincer-search: an efficient algorithm for discovering the maximum frequent set," IEEE Transactions on Knowledge and Data Engineering, Vol. 14, pp. 553-566, 2002.
[8]. E. Hullermeier, "Possibilistic Induction in Decision-Tree Learning," in Proceedings of the 13th European Conference on Machine Learning, pp. 173-184, 2002.
[9]. Cheung, W., "Frequent Pattern Mining without Candidate generation or Support Constraint." Master’s Thesis, University of Alberta, 2002.
[10]. Huang, H., Wu, X., and Relue, R. Association Analysis with One Scan of Databases. Proceedings of the 2002 IEEE International Conference on Data Mining. 2002.
[11]. Beil, F., Ester, M., Xu, X., Frequent Term-Based Text Clustering, ACM SIGKDD, 2002
[12]. Zaki, M. J., Parthsarathy, S., Ogihara, M., and Li, W. New Algorithms for Fast Discovery of Association Rules. KDD, 283-286. 1997. Agarwal, R., Aggarwal, C., and Prasad, V.V.V. 2001.
[13]. Goulbourne, G., Coenen, F., and Leng, P. H. Computing association rule using partial totals. In Proceedings of the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases, 54-66. 2001.
[14]. Pei, J., Han, J., Nishio, S., Tang, S., and Yang, D. H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases. Proc.2001 Int.Conf.on Data Mining. 2001.
[15]. Orlando, S., Palmerini, P., and Perego, R. Enhancing the Apriori Algorithm for Frequent Set Counting. Proceedings of 3rd International Conference on Data Warehousing and Knowledge Discovery. 2001.
[16]. Grahne, G., Lakshmanan, L., and Wang, X. 2000. Efficient mining of constrained correlated sets. In Proc. 2000. Int. Conf. Data Engineering (ICDE’00), San Diego, CA, pp. 512–521.
[17]. Dong, G. and Li, J. 1999. Efficient mining of emerging patterns: Discovering trends and differences. In Proc. 1999 Int. Conf. Knowledge Discovery and Data Mining (KDD’99), San Diego, CA, pp. 43–52