International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.1, Issue1, pp-108-113 ISSN: 2249-6645

Implementation of object oriented approach to Index Support for Item Set Mining (IMine)

R.SRIKANTH,
2/2 M.TECH CSE, DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
ADITYA INSTITUTE OF TECHNOLOGY AND MANAGEMENT, TEKKALI,
ANDHRA PRADESH, INDIA.

D.T.V.DHARMAJEE RAO
PROFESSOR & HOD,
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING,
ADITYA INSTITUTE OF TECHNOLOGY AND MANAGEMENT, TEKKALI,
ANDHRA PRADESH, INDIA.

Abstract— **The increase in huge amount of data is seen clearly in present days because of requirement for storing more information. To extract certain data from this large database is a very difficult task. This leads to the researchers to drag themselves for developing better technique to mine the required data. There are various techniques proposed by several researchers to deal with this difficulty. Among various available techniques, association rule mining for extract the required data from the database is found to be better. This paper presents the IMine index, a general and compact structure which provides tight integration of item set extraction in a relational DBMS Since no constraint is enforced during the index creation phase, IMine provides a complete representation of the original database. To reduce the I/O cost, data accessed together during the same extraction phase are clustered on the same disk block. Experiments, run for both sparse and dense data distributions, show the efficiency of the proposed index and its linear scalability also for large data sets.**

***Index Terms*— Object Oriented Approach; IMine; Index Support; Item Set Mining.**

I. INTRODUCTION

DATA mining is provoked by decision support difficulties featured by majority of business organizations and is illustrated as an significant field of research. One of the major difficulties in data mining is creating fast and efficient techniques that can deals with large volumes of data as majority mining techniques carry out computation over the complete database and frequently the databases are in huge size. Physical analysis of these huge amount of information stored in modern databases is very difficult. A recognized data mining technique is association rule mining. It is able to discover all interesting relationships which are called as associations in a database. Association rules are very efficient in revealing all the interesting relationships in a relatively large database with huge amount of data. The large quantity of information collected through the set of association rules can be used not only for illustrating the relation-ships in the database, but also used for differentiating between different kinds of classes in a database. But the major difficulty in association rule mining is its complexity.

Research activity usually focuses on defining efficient algorithms for item set extraction, which represents the most computationally intensive knowledge extraction task in association rule mining [1]. The data to be analyzed is usually stored into binary files, possibly extracted from a DBMS. Most algorithms [2, 3] exploit ad hoc main memory data structures to efficiently extract item sets from a flat file. Recently, disk-based extraction algorithms have been proposed to support the extraction from large data sets [4,5,6], but still dealing with data stored in flat files. To reduce the computational cost of item set extraction, different constraints maybe enforced [7,8,9], among which the most simple is the support constraint, which enforces a threshold on the minimum support of the extracted item sets.

Relational DBMSs exploit indices, which are ad hoc data structures, to enhance query performance and support the execution of complex queries. In this paper, we propose a similar approach to support data mining queries. The IMine index (Item set-Mine index) is a novel data structure that provides a compact and complete representation of transactional data supporting efficient item set extraction from a relational DBMS. It is characterized by the following properties:

1. It is a covering index. No constraint (e.g., support constraint) is enforced during the index creation phase. Hence, the extraction can be performed by means of the index alone, without

accessing the original database. The data representation is complete and allows reusing the index for mining item sets with any support threshold.

2. The IMine index is a general structure which can be efficiently exploited by various item set extraction algorithms. These algorithms can be characterized by different in-memory data representations (e.g., array list, prefix-tree) and techniques for visiting the search space. Data access functions have been devised for efficiently loading in memory the index data. Once in memory, data is available for item set extraction by means of the algorithm of choice.
3. The IMine physical organization supports efficient data access during item set extraction. Correlation analysis allows us to discover data accessed together during pattern extraction. To minimize the number of physical data blocks read during the mining process, correlated information is stored in the same block.
4. IMine supports item set extraction in large data sets. We exploit a direct writing technique to avoid representing in memory the entire large data set. Direct materialization has a limited impact on the final index size because it is applied only on a reduced portion of the data set.

II. Literature Survey

The goal of data mining is to discover important associations among items such that the presence of some items in a transaction will imply the presence of some other items. To achieve this purpose, many people propose different procedures; here we discuss some of them.

E. Baralis et al., [10] recommended itemset mining on indexed data blocks. Numerous attempts have been offered to combine data mining activities with relational DBMSs, but a correct incorporation into the relational DBMS kernel has been infrequently achieved. This paper suggested an innovative indexing method, which denotes the transactions in a succinct form, suitable for tightly incorporating frequent itemset mining in a relational DBMS. The data illustration is complete, i.e. no support threshold

Mining association rules from XML data with index table was suggested by Xin-Ye Li et al., [11]. Mining XML association rule is tackled with extra challenge because of the inherent flexibilities of XML in both arrangement and semantics. With the purpose of making mining XML association rule very efficient, this paper provides a new definition of transaction and item in XML environment, then construct transaction database depending on an index table. Based on the definition and the index table utilized for XML searching, it is easy to check the relation among the transaction and retrieve an item quickly. A high adaptive mining approach is also illustrated. By using this approach, mining rules can be processed with no assistance of interest associations specified by users and mining unknown rules. The effectiveness of these approaches is proved with the help of experiments on real-life data.

E.J. Keogh et al., [12] proposed an indexing scheme for fast similarity search in large time series databases. This paper addresses the trouble of similarity searching in huge time-series databases. The authors proposed an innovative indexing approach that permits quicker retrieval. The index is produced by generating bins that include time series subsequences of roughly the similar shape. For every bin, this proposed approach can rapidly compute a lower bound on the distance among a given query and the most similar element of the bin. This bound permits to search the bins in greatest-first order, and to prune some bins from the search space without verifying the contents. Further speedup can be achieved by optimizing the data inside the bins in such a way that ignores the process of comparing the query to every item in the bin.

L. Golab et al., [13] proposed indexing time method for evolving data with variable lifetimes. Numerous applications store data items for a pre-determined, fixed duration of time. Examples consist of sliding windows over online data streams, in which old data are thrown out as the window slides forward. Earlier researches on management of data with limited lifetimes have emphasized online query processed in main memory. In this approach, the authors concentrate on the difficulty of indexing time-developing data on disk for offline investigation. With the intention of decreasing the I/O costs of index updates, existing work separates the data chronologically. Thus, only the previous separation is examined for expirations, only the youngest separations acquire insertions, and the remaining partitions in the middle are not processed. On the other hand, this result is based upon the hypothesis that the order in which the data are introduced is equivalent to the termination order, which means that the lifetime of each data item is the similar. In order to break this hypothesis, the authors reveal that the existing solutions no longer be relevant, and suggested a new index partitioning strategies that provide low update costs and quick access times.

A new approach of modified transaction reduction algorithm for mining frequent itemset was proposed by R.E. Thevar et al., [14]. Association rule mining is to take out the interesting association and relation among the huge volumes of transactions. This procedure is segmented into two sub problem: first problem is to discover the frequent itemsets from the transaction and then the second problem is to build the rule from the mined frequent itemset [15]. Frequent itemsets creation is the necessary and most time huge procedure for association rule mining. Currently, most well-organized apriori-like algorithms rely deeply on the minimum support constraints to prune the enormous amount of non-candidate itemsets. These algorithms store numerous unnecessary itemsets and transactions. In this paper, the authors proposed an innovative frequent itemsets creation algorithm called MTR-FMA (modified transaction reduction depends on frequent itemset mining algorithm) that sustains its performance even at relative low supports. The experimental output also proves that proposed MTR-FMA algorithm on an outset is quicker than high efficient AprioriTid and other algorithms.

Lei Wen et al., [16] developed an efficient algorithm for mining frequent closed itemset. Association rule mining was a significant field of data mining investigation. Determining the potential frequent itemset was a vital step. The existed frequent itemset discovery algorithms could find out all the frequent itemset or maximal frequent itemset. N. Pasquier developed an innovative job of mining frequent closed itemset. The size of frequent closed itemset was much lesser than all the frequent itemsets and did not lose any information.

In this paper, we propose a new itemset approach depends on the index. This approach can discover all the frequent closed itemset powerfully by using indexing method.

III. Index Structure

Index structure for extracting item set as sequence of data blocks. The index supports user communication, where the user specifies many constraints for itemset extraction. It permits the mining of the complete set of itemsets which satisfy (a) time constraints and (b) support constraints. Since the index contains all feature potentially required during the mining task, the extraction can be carried out by means of the index, without accessing the database. The data representation is absolute, i.e., no support threshold is enforced throughout the index construction stage, to permit reusing the index for mining itemsets with any support threshold. Constraints like support and confidence is not enforced throughout the index creation stage. Therefore, the extraction can be carried out using the index alone, without accessing the original database. As the databases are necessary in almost all the retail stores, super markets, etc., it is necessary to develop an approach for item set mining with the help of index support. The structure of the IMine index is characterized by two components: the Item set-Tree and the Item-Btree. The two components provide two levels of indexing. The Item set-Tree (I-Tree) is a prefix-tree which represents relation R by means of a succinct and lossless compact structure.

The Item-Btree (I-Btree) is a B+Tree structure which allows reading selected I-Tree portions during the extraction task. For each item, it stores the physical locations of all item occurrences in the I-Tree. Thus, it supports efficiently loading from the I-Tree the transactions in R including the item. In the following, we describe in more detail the I-Tree and the I-Btree structures. Fig. 1 a and b shows the complete structure of the corresponding IMine index. In the I-Tree paths (Fig. 1 a), nodes are sorted by decreasing support of the corresponding items. In the case of items with the same support, nodes are sorted by item lexicographical order. In the I-Tree, the common prefix of two transactions is represented by a single path.

A. I-Tree

The I-Tree associated to relation R is actually a forest of prefix-trees, where each tree represents a group of transactions all sharing one or more items. Each node in the I-Tree corresponds to an item in R. Each path in the I-Tree is an ordered sequence of nodes and represents one or more transactions in R. Each item in relation R is associated to one or more I-Tree nodes and each transaction in R is represented by a unique I-Tree path. Each I-Tree node is associated with a node support value, representing the number of transactions which contain (without any different interleaved item) all the items in the sub path reaching the node.

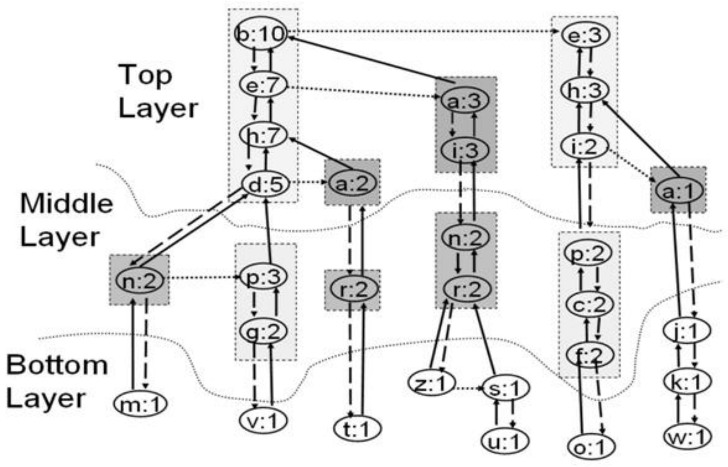


Fig 1 (a) I Tree for the example data set

B. *I-Btree*

The I-Btree allows selectively accessing the I-Tree disk blocks during the extraction process. It is based on a B+Tree structure [21]. Fig. 1 b shows the I-Btree for the example data set and a portion of the pointed I-Tree. For each item i in relation R, there is one entry in the I-Btree. In particular, the I-Btree leaf associated to i contains i's item support and pointers to all nodes in the I-Tree associated to item i. Each pointer stores the physical location of the record in table TI-Tree storing the node. Fig. 1 b shows the pointers to the I-Tree nodes associated to item r.

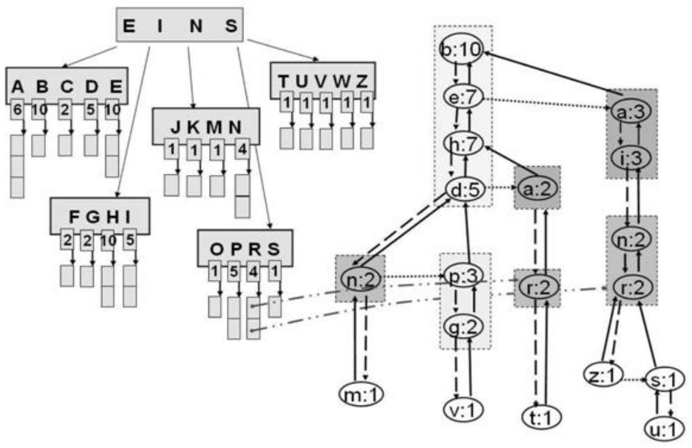


Fig 1 (b) I BTree for the example data set

IV. Item Set Mining

Item set mining are two sequential steps: 1) the needed index data is loaded and 2) item set extraction takes place on loaded data.

A. *Frequent Item Set Extraction*

This section describes how frequent item set extraction takes place on the IMine index. We present two approaches, denoted as FP-based and LCM-based algorithms, which are an adaptation of the FP-Growth algorithm [17] and LCM v.2algorithm [18], respectively.

B. *FP-based algorithm:*

The FP-growth algorithm stores the data in a prefix-tree structure called FP-tree. First, it computes item support. Then, for each transaction, it stores in the FP-tree its subset including frequent items. Items are considered one by one. For each item, extraction takes place on the frequent-item projected database, which is generated from the original FP-tree and represented in a FP-tree based structure.

C. *LCM-based algorithm:*

The LCM v.2 algorithm loads in memory the support based projection of the original database. First, it reads the transactions to count item support. Then, for each transaction, it loads the subset including frequent items. Data are represented in memory by means of an array based data structure, on which the extraction takes place.

V. Experimental Results

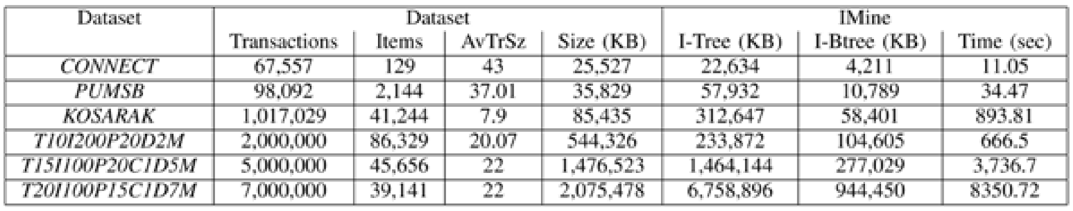
We validated our approach by means of a large set of experiments addressing the following issues:

- Performance of the IMine index creation, in terms of both creation time and index size,

- Performance of frequent item set extraction, in terms of execution time, memory usage, and I/O access time,
- Effect of the DBMS buffer cache size on hit rate,
- Effect of the index layered organization,
- Effect of direct writing, and
- Scalability of the approach.

We ran the experiments for both dense and sparse data distributions. We report experiments on six representative data sets whose characteristics (i.e., transaction and item cardinality, average transaction size (AvgTrSz), and data set size) are in Table 1. Connect and Pumsb [19] are dense and medium-size data sets. Kosarak [19] is a large and sparse data set including click-stream data. T10I200P20D2M is a dense and large synthetic data set, while T15I100P20C1D5M and T20I100P15C1D7M are quite sparse and large synthetic data sets. Synthetic data sets are generated by means of the IBM generator [20]. For all data sets, the index has been generated without enforcing any support threshold.

Table 1. Data Set Characteristics and Corresponding Indices.



Dataset	Dataset				IMine		
	Transactions	Items	AvTrSz	Size (KB)	I-Tree (KB)	I-Btree (KB)	Time (sec)
CONNECT	67,557	129	43	25,527	22,634	4,211	11.05
PUMSB	98,092	2,144	37.01	35,829	57,932	10,789	34.47
KOSARAK	1,017,029	41,244	7.9	85,435	312,647	58,401	893.81
T10I200P20D2M	2,000,000	86,329	20.07	544,326	233,872	104,605	666.5
T15I100P20C1D5M	5,000,000	45,656	22	1,476,523	1,464,144	277,029	3,736.7
T20I100P15C1D7M	7,000,000	39,141	22	2,075,478	6,758,896	944,450	8350.72

Table 1 reports both I-Tree and I-Btree size for the six data sets. The overall IMine index size is obtained by summing both contributions. The IMine indices have been created with the default value K_{avg}=1.2. Furthermore, the Connect, Pumsb, Kosarak, and T10I200P20D2M data sets have been created with K_{sup}=0, while large synthetic data sets with K_{sup} =0.05. The adopted I-Tree representation is more suitable for dense data distributions, for which it provides good data compression. In dense data sets (e.g., T10I200P20D2M) where data are highly correlated, the I-Tree structure is more compact. In sparse data sets (e.g., Kosarak), where data are weakly correlated, data compression is low and storing the I-Tree requires more disk blocks.

VI. CONCLUSION AND FUTURE WORK

Due to the world wide increase in the available data, it is very difficult for obtaining the related data with better accuracy. Therefore the available techniques for data mining will not be able to extract the relevant data. This leads to the requirement for developing a better data mining technique which suits all situations. Association rule mining technique is one of the better techniques among the existing techniques. Even, this technique possesses various difficulties when large database used. Later, a new technique called indexing is introduced to solve those problems. This indexing will carry the necessary parameters to classify the required data from the large database. This paper provides object oriented approach to IMine in a complete and compact representation of transactional data. It is a general structure that efficiently supports different algorithmic approaches to item set extraction. Experimental results show that, Selective access of the physical index blocks significantly reduces the I/O costs and efficiently exploits DBMS buffer management strategies.

REFERENCES

[1] R. Agrawal and R. Srikant, “Fast Algorithm for Mining Association Rules,” Proc. 20th Int’l Conf. Very Large Data Bases (VLDB ’94), Sept. 1994.
[2] J. Han, J. Pei, and Y. Yin, “Mining Frequent Patterns without Candidate Generation,” Proc. ACM SIGMOD, 2000.
[3] H. Toivonen, “Sampling Large Databases for Association Rules,” Proc. 22nd Int’l Conf. Very Large Data Bases (VLDB ’96), pp. 134-145,1996.
[4] M. El-Hajj and O.R. Zaiane, “Inverted Matrix: Efficient Discovery of Frequent Items in Large Datasets in the Context of Interactive Mining,” Proc. Ninth ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (SIGKDD), 2003.
[5] G. Grahne and J. Zhu, “Mining Frequent Itemsets from Secondary Memory,” Proc. IEEE Int’l Conf. Data Mining (ICDM ’04), pp. 91-98, 2004.
[6] G. Ramesh, W. Maniatty, and M. Zaki, “Indexing and Data Access Methods for Database Mining,” Proc. ACM SIGMOD Workshop Data Mining and Knowledge Discovery (DMKD), 2002.
[7] Y.-L. Cheung, “Mining Frequent Itemsets without Support Threshold: With and without Item Constraints,” IEEE Trans. Knowledge and Data Eng., vol. 16, no. 9, pp. 1052-1069, Sept. 2004.
[8] G. Cong and B. Liu, “Speed-Up Iterative Frequent Itemset Mining with Constraint Changes,” Proc. IEEE Int’l Conf. Data Mining (ICDM ’02), pp. 107-114, 2002.
[9] C.K.-S. Leung, L.V.S. Lakshmanan, and R.T. Ng, “Exploiting Succinct Constraints Using FP-Trees,” SIGKDD Explorations Newsletter, vol. 4, no. 1, pp. 40-49, 2002.
[10] E. Baralis, T. Cerquitelli, and S. Chiusano, “Index Support for Frequent Itemset Min-ing in a Relational DBMS,” Proceedings 21st International Conference on Data En-gineering (ICDE), pp. 754 - 765, 2005.
[11] Xin-Ye Li, Jin-Sha Yuan and Ying-Hui Kong, “Mining Association Rules from XML Data with Index Table,” International Conference on Machine Learning and Cy-bernetics, Vol. 7, pp. 3905 – 3910, 2007.

[12] E.J. Keogh and M.J. Pazzani, “An index-ing scheme for fast similarity search in large time series databases,” Eleventh Interna-tional Conference on Scientific and Statis-tical Database Management, pp. 56 – 67, 1999.

[13] L. Golab, P. Prahladka and M.T. Ozsu, “Indexing Time-Evolving Data With Varia-ble Lifetimes,” 18th International Confe-rence on Scientific and Statistical Database Management, pp. 265 – 274, 2006.

[14] R.E. Thevar and R. Krishnamoorthy, “A new approach of modified transaction re-duction algorithm for mining frequent itemset,” 11th International Conference on Computer and Information Technology (ICCIT 2008), pp. 1 – 6, 2008.

[15] Jianyong Wang, J. Han, Y. Lu and P. Tzvetkov, “TFP: an efficient algorithm for mining top-k frequent closed itemsets” IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 5, pp. 652 – 663, 2005.

[16] Lei Wen, “An efficient algorithm for min-ing frequent closed itemset,” Fifth World Congress on Intelligent Control and Auto-mation (WCICA 2004), Vol. 5, pp. 4296 – 4299, 2004.

[17] J. Han, J. Pei, and Y. Yin, “Mining Frequent Patterns without Candidate Generation,” Proc. ACM SIGMOD, 2000.

[18] T. Uno, M. Kiyomi, and H. Arimura, “LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets,”Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations (FIMI), 2004.

[19] FIMI, http://fimi.cs.helsinki.fi/, 2008.

[20] N. Agrawal, T. Imielinski, and A. Swami, “Database Mining: A Performance Perspective,” IEEE Trans. Knowledge and Data Eng., vol. 5, no. 6, Dec. 1993.