A Survey on ECLAT Based Algorithm

[1]Nisha Bali, [2]Dr. Kuldeep Singh and [3]Sunita Turan,
[1] University Institute of Engineering and Technology ,Kurukshetra University Kurukshetra, India
[2,3]Institute of Mass Communication and Media Technology (Multimedia), Kurukshetra University Kurukshetra, India

ABSTRACT: Eclat is a program for frequent item set mining, a data mining method that was originally developed for market basket analysis. Frequent item set mining aims at finding regularities in the shopping behavior of the customers of supermarkets, mail-order companies and online shops. In particular, it tries to identify sets of products that are frequently bought together. Once identified, such sets of associated products may be used to optimize the organization of the offered products on the shelves of a supermarket or the pages of a mail-order catalog or web shop, may give hints which products may conveniently be bundled, or may allow suggesting other products to customers. However, frequent item set mining may be used for a much wider variety of tasks, which share that one is interested in finding regularities between (nominal) variables in a given data set. For an overview of frequent item set mining in general and several specific algorithms (including Eclat)

Keywords *: Eclat , Frequent Pattern matching , DataMining*

I. INTRODUCTION

A. Frequent Pattern Mining

This is another important frequent pattern mining method, which generates frequent itemset without candidate Generation. It uses tree based structure. The problem of Apriori algorithm was dealt with, by introducing a novel, compact data structure, called frequent pattern tree, or FP-tree then based on this structure an FP-tree-based pattern fragment growth method was developed[5]. It constructs conditional frequent pattern tree and conditional pattern base from database which satisfy the minimum support [5].FP-growth traces the set of concurrent items [6].

B. Database layout/database format

A) **Frequent Itmeset**: The concept of frequent itemsets mining was introduced by Agrawal in 1993. It can be formally stated as: Let I = {i1, i2,…,in} be a set of n distinct items. Let D = {t1, t2,…, tm} be a set of m transactions, each transaction consists of a unique transaction identifier and a set of items, called itemset, which is a subset of set I. Definition 1: Let I1I, the support of I1 is the ratio of transactions which contain I 1. So, support(I1) = ||{tęD| I1 t}||||D||. Definition 2: If the support of I1 is larger than a minimum support threshold—min_sup which is set by user, then I1 is frequent itemset. The task of frequent Itemset mining is to find all the frequent itemsets from database. And it is the first and key step of association rule mining. With the help of frequent itemsets, we can generate rules to find useful information between items.

C. Database Layout:

There are two layout formats of the target dataset for association mining: the horizontal and the vertical layout. The horizontal layout database is also called the transactional database. It consists of a list of transactions, each transaction has an identifier (TID) followed by a list of items in that transaction. This format imposes some computation overhead during the support counting step [4]. In the vertical (or inverted) layout, database consists of a list of items, each item followed by the list of tids (also called tid-list). So, in the vertical layout database, it is quite easy to find frequent itemsets, because we just need to count the number of tids of each item.

D. ECLAT: description of ECLAT

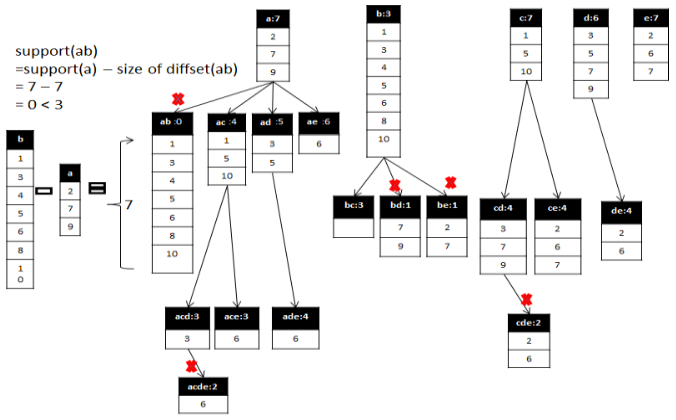
On an abstract level, the input to frequent item set mining consists of a bag or multiset of *transactions* that are defined over a set of *items*, sometimes called the *item base*. These items may be products, special equipment items, service options etc. For an abstract treatment it suffices that items have an identity, that is, it is possible to distinguish one item from another. The item base is the set of all considered items, for example, the set of all products that are sold by a given supermarket, mail-order company or online shop. Any subset of the item base is called an *item set*.

A transaction is simply an item set and it represents, for example, the set of products bought by a customer. Since two or more customers may, in principle, buy the exact same set of products, we cannot model the whole of all "shopping baskets" or "shopping carts" (bought, say, in a given week) as a *set* of transactions, since in a set each element is unique. There are several solutions to this problem: one may, for example, model the whole of all transactions as a bag or multiset (a generalization of a set, which allows for multiple occurrences of the same element) or as a vector (where elements at different positions may be the same, but are still distinguished by their position), or by extending each transaction with a unique *transaction identifier* (or *tid* for short; note that the position of a transaction in a vector representation is an implicit transaction identifier). Still another possibility consists in using a standard *set* of (unique) transactions and assigning to each of them an occurrence counter. Here I use the bag/multiset terminology, even though an occasional "set of transactions" may have slipped through. (This should always be read as "bag/multiset of transactions".)

ECLAT based algorithms-vertical layout based algorithms

The initial transaction database in vertical layout is firstly converted to format in which items are sets of whose transactions do not contain items. This is deduced from the definition, the initial transaction database in vertical layout is an equivalence with the prefix *P= {}*, so the transaction of *P* includes all transaction, all transactions contain *P*, and the different of an item $d(i) = t(P) - t(i)$, this is a set of transaction whose transactions do not contain *i*. From this initial equivalence class, we could generate all item sets with their different confidence and supports.

When Éclat uses the confidence and support format, it is called Éclat algorithm. And Éclat is different from Éclat in the step 5, instead of generating a new transaction, a new is generated.Furthermore, we also need to store support of item sets in equivalence class to facilitate calculating supports of new item sets.



II. LITRATURE SURVEY

On various algorithms which is used in "VERTICAL LAYOUT"

Since its introduction by Agrawal et al[1], it has received a great deal of attention and various efficient and sophisticated algorithms have been proposed to do frequent itemset mining. Among the best-known algorithms are Apriori, Eclat and FP-Growth.

The Apriori algorithm [2]uses a breadth-first search and the downward closure property, in which any superset of an infrequent itemset is infrequent, to prune the search tree. Apriori usually adopts a horizontal layout to represent the transaction database and the frequency of an itemset is computed by counting its occurrence in each transaction.

FP-Growth [3] employs a divide-and-conquer strategy and a FP-tree data structure to achieve a condensed representation of the transaction database. It is currently one of the fastest algorithms for frequent pattern mining.

Eclat[4] takes a depth-first search and adopts a vertical layout to represent databases, in which each item is represented by a set of transaction IDs (called a tidset) whose transactions contain the item. Tidset of an itemset is generated by intersecting tidsets of its items. Because of the depth-first search, it is difficult to utilize the downward closure property like in Apriori. However, using tidsets has an advantage that there is no need for counting support, the support of an itemset is the size of the tidset representing it. The main operation of Eclat is intersecting tidsets, thus the size of tidsets is one of main factors affecting the running time and memory usage of Eclat. The bigger tidsets are, the more time and memory are needed.

Zaki and Gouda[5] proposed a new vertical data representation, called Diffset, and introduced dEclat, an Eclat-based algorithm using diffset. Instead of using tidsets, they use the difference of tidsets (called diffsets). Using diffsets has reduced drastically the set size representing item sets and thus operations on sets are much faster. Eclat had been shown to achieve significant improvements in performance as well as memory usage over Eclat, especially on dense databases[5]. However, when the dataset is sparse, diffset loses its advantage over tidset. Therefore, Zaki and Gouda suggested using tidset format at the start for sparse databases and then switching to diffset format later when a switching condition is met.

CONCLUSION

To count supports of candidates, we need to go through transactions in the transaction database and check if transactions contain candidates. Since the transaction database is usually very large, it is not always possible to store them into main memory. Furthermore, to check if a transaction containing anitemset is also a non-trivial task. So an important consideration in frequent item set mining algorithms is the representation of the transaction database to facilitate the process of counting support. There are two layouts that algorithms usually employ to represent transaction databases: horizontal and vertical layout

References

[1] R. Agrawal, T. Imielinski, and A.N. Swami, "Mining association rules between sets of items in large databases," in *ACM SIGMOD International Conference on Management of Data*, Washington, 1993.

[2] R. Agrawal, and R. Srikant, "Fast algorithms for mining association rules," in *20th International Conference on Very Large Data Bases*, Washington, 1994.

[3] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM SIGMOD International Conference on Management of Data*, Texas, 2000.

[4] M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New algorithms for fast discovery of association rules," in *Third International Conference on Knowledge Discovery and Data Mining*, 1997.

[5] a. K. G. M.J. Zaki, "Fast vertical mining using diffsets," in *The nineth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[6] Paul W. Purdom, Dirk Van Gucht, and Dennis P. Groth, "Average case performance of the apriori algorithm," vol. 33, p. 1223–1260, 2004.

[7] S. Orlando, P. Palmerini, R. Perego, and F. Silvestri, "Adaptive and resource-aware mining of frequent sets," in *Proceedings of the 2002 IEEE International Conference on Data Mining*, 2002.

[8] P. Shenoy, J.R. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa, and D. Shah, "Turbo-charging vertical mining of large databases," in *ACM SIGMOD International Conference on Management of Data*, 2000.

[9] B. Goethals, "Survey on frequent pattern mining," 2002.

[10] Yan Zhang, Fan Zhang, Jason Bakos, "Frequent Itemset Mining on Large-Scale Shared Memory Machines," 2011.