Enhancing Query Performance Using Simultaneous Execution and Vertical Query Splitting

1st Rais Allauddin Mulla
Vasantdada Patil Pratishthan's College of Engineering & Visual Arts
Sion Mumbai, India
mtechraismulla@gmail.com

2nd Yogesh Mali
G.H.Raisoni College of Engineering & Management,Wagholi
Pune, India
yogeshmali3350@gmail.com

3rd Vijay U. Rathod
G.H.Raisoni College of Engineering & Management,Wagholi
Pune, India
vijay.rathod25bel@gmail.com

4th Ravindra S.Tambe
Dr. Vithalrao Vikhe Patil College of Engineering
Ahmednagar,India
ravindra.tambe01@gmail.com

5th Radha Shirbhate
G.H.Raisoni College of Engineering & Management,Wagholi
Pune, India
radha.shirbhate@gmail.com

6th Richa Agnihotri
G.H.Raisoni College of Engineering & Management,Wagholi
Pune, India
richa.agnihotri@gmail.com

Abstract— **in recent years, the significance of data mining algorithms has increased with the fast growth of data. Elements are often repeated when dealing with transitions that fetch data. Therefore, one of the most essential issues in data mining is mining common item collections. Here we tackle an algorithm that finds common elements appearing in transitions and tries to split the data accordingly into different tables, called vertical fragmentation. This gives you high time efficiency. Here we use FP growth for the mining phase. Another smart splitting method has been proposed that converts the database into separate tables.**

Keywords: Run-time Estimation Module, Vertical Fragmentation, Transaction Splitting, FP-Growth.

I. INTRODUCTION

In recent years, information technology is progressing in all companies. Various data management packages are available in the market, and they also have various techniques for retrieving data from them. Various techniques are used to achieve faster, better, and more accurate performance. Data sizes are also growing rapidly every day, driving interest in developing tools that can automatically, quickly, and accurately extract knowledge from data. In the era of data mining, ER research on big data became a focus. Data mining has new capabilities such as high volume, high speed, and high dimensionality. As a result of which query splitting has greater importance.

Elements are repeated many times in a transaction. Given this repetition, instead of retrieving these common items directly from the database, we can use vertical fragmentation to improve performance. From the transactions, we find the most often happening item sets and try to split the data set according to these item sets. Create different datasets for repeating item sets. Subsequent to parting the dataset into several parts, the transitions are also split into several parts and get the data accordingly. After the data is acquired, it is combined for the final output.

A. RELEVANCE

Data mining needs to find common sets of items to speed up queries if they are triggered frequently. To do this, you can use an algorithm developed by frequent Item sets mining.

This algorithm takes input as a query used by a group of people for transactions and produces a frequent item set as output. We used an a priori algorithm to find this objective, finding all frequently used item sets in a given set of queries. The basic idea behind Apriori algorithms is to iterate the query multiple times, like BFS (Broad First Search). The downside, however, is that the time required for execution is more than wasted time creating candidates at every point.

Another approach to the same task is the most popular item set mining called the FP growing algorithm. However, since FP tree generation and FP growth also rely on BFS, this also takes a significant amount of time to output. So, consider vertical fragmentation that can run on multiple CPUs using multithreading. To find common item sets, we try to use a weighted method that indicates how often a particular item appears in a query compared to others. The algorithm he consists of two phases.

II. LITERATURE REVIEW

A publication algorithm for frequent item sets that have been anonymised is described in this study. These two investigations, however, fail to meet the requirements for differential privacy and cannot effectively safeguard against an attacker with arbitrary previous knowledge [1].

The author's goal was a one-to-many element mapping-based encryption approach is security, and we also created an "audit environment" to ensure that the data mining findings are accurate [2].

In the context of distributed data mining, this research study by Author's focused on the safe mining of rules like association on horizontally partitioned data. Without revealing the details of individual transactions, this method enables you to learn the characteristics of transactions that were split between sites. This approach takes into account secure multi-party computation and uses cryptographic addresses that increase the mining task's efficiency while minimizing the amount of shared information [3].

The PrivBasis (PB) technique, which reduces high-dimensional input databases to lower dimensions to get over the high-dimensional problems that transactional databases face and enhance the quality of generic item sets, is described in this research paper. The input database is projected into lower dimensions via PrivBasis [4].

III. DESIGNING PHASE

There are two phases to the algorithm.

1. In the pre-processing stage, the original database is converted using the smart splitting approach after some statistics have been taken out of it.
2. In the phase of mining, a common set of items is

created privately for a certain threshold. To enhance the caliber of the outcomes, this phase employs dynamic reduction techniques and execution time estimation techniques.

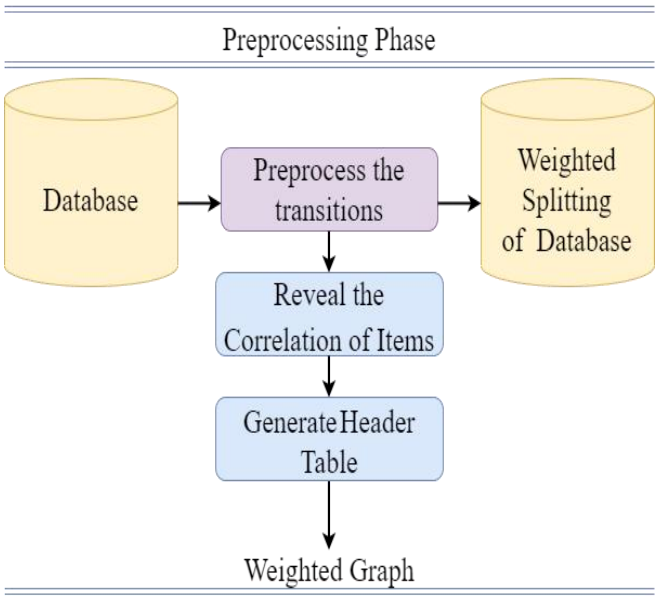

Fig. 1. Figure 1: Preprocessing Phase

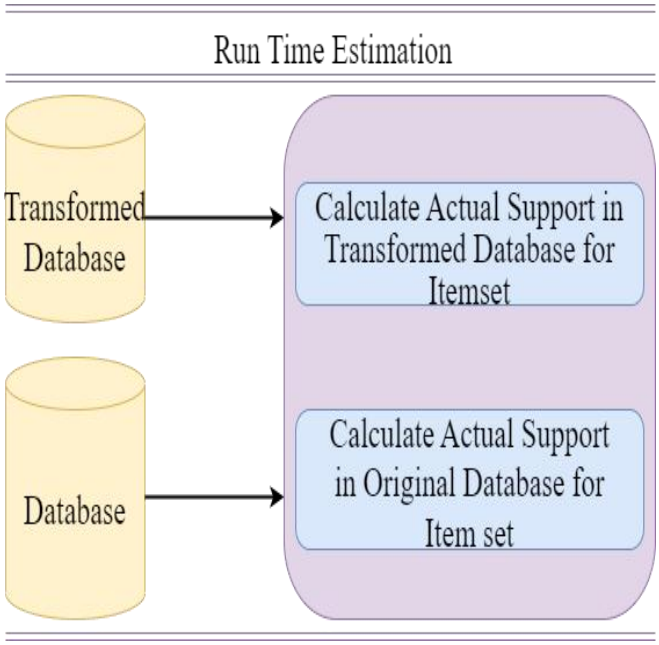

Fig. 2. Figure 2: Run Time Estimation

IV. METHODOLOGY

A. *Module1: Pre-processing Phase*

First, set the transaction length Lm. This is nothing but the number of items involved in the transaction. If the transition contains few elements, it is executed using the original data as is. If the number of items is greater than the Lm value, the transaction should be split. To split the transactions, I calculated the threshold λ (the number of times an item occurs in a transaction) for each item as a function of previous transactions and also created a header table. CR trees and undirected weighted graphs are generated from HT tables to help split transactions. Depending on the charts and HT tables, the database will be converted into multiple databases/tables.

TABLE I. TABLE I: CONSIDER THE FOLLOWING TRANSACTIONS

Transaction ID	Items into transaction
1001	a, b, c, f
1002	b, c, h
1003	a, b, c, e, f
1004	b, c, d, h
1005	a, g
1006	a, f, g

TABLE II. TABLE II: FOR ABOVE TRANSITION HEADER TABLE (HT) WILL BE

a	4
b	4
c	4
f	3
g	2
h	2

Undirected weighted graph will be like this

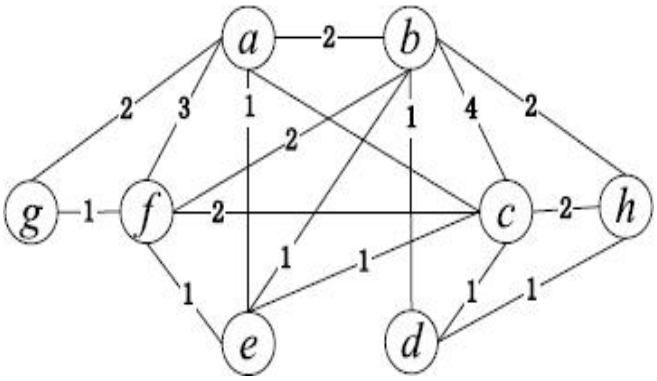

Fig. 3. Undirected Weighted Graph

B. *Module2: Run-time Estimation Module*

In the mining stage, we compute the maximum length Lm of frequent item sets based on the maximum support of i item sets provided with λ as a threshold. In addition, it computes execution time estimates to find loss information or data caused due to transaction splitting. To privately find common item sets, Palladian noise added to any item support, and the maximum and average support of the original database is calculated based on the noise support and run time estimates[6]. If this maximum support is more than a given threshold, insert item (c) into the header table (HT), and if estimated average support is far more than threshold, item c is considered a common item set. With HT, even with small key sizes, improves data security. More importantly, [9] HT consumes less computational power,

generates less heat, and delivers results very quickly compared to other algorithms. Increase Register the upper limit of the number of support calculations in an array. Then sort the elements of HT in descending order based on the maximum support value and build an FP tree. A conditional pattern base for each item is then generated in HT and a common set of items is searched based on the conditional pattern.

V. ALGORITHM

Input: Query (Q), Threshold (T), Query File (QF).

Output: Set of queries (SQ).

1. Separate the items from the query Q.
2. Find out maximum used item in previous all transactions using QF.
3. Find out count of remaining items with respect of item of 2nd step.
4. Arrange all the items on descending order of count.
5. Separate the items from query depending upon T
6. Make separated items from step 5 to form queries in SQ
7. Store query Q in QF for further use
8. Return SQ.

Step 1: We separate all the items from a query Q.
Step 2, 3: We process QF. In this processing we generate the set which indicates item and used count .i.e < item, count>. After which we get the item from query Q which is used frequently from the set derived in previous step.
Step 4: We get the count of all remaining items from Q which are used along with the item derived in previous step.
Step 5: We make the set of items from Q which are used to form query in next step. Every set contains number of items equal to threshold.
Step 6: Forms the query set SQ from a given query Q

VI. BENEFIT OF TRANSACTION SPLITTING

During the initial database scan, we often find a set of 1 item from the database transformed by the splitting method [2]. For each long transaction, split it into subsets using the smart truncation method recursively. The weights of the resulting subsets are evenly distributed. In addition, the mining process uses run-time estimation methods to quantify the data and information loss resulted due to splitting transactions. It can be seen that adopting the transaction splitting technique significantly better's the performance and accuracy of the transaction execution time. However, there are also security benefits where other attackers cannot externally capture the transaction record.

We tried the program for different number of attributes. We have run multi-threaded and single threaded programs. The execution time (maximum thread execution time for multi-threading) is recorded and the graph looks like this:

For the first time we considered 8 and 13 attributes. Series1 is for multi-threaded execution and Series2 is for single-threaded execution.

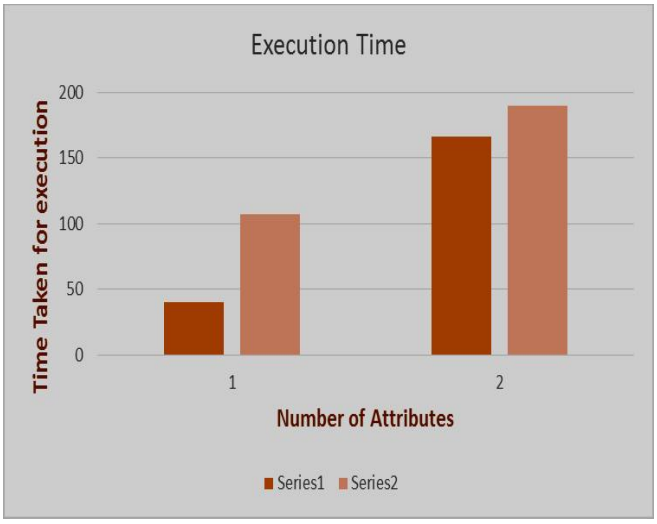

Fig. 4. Execution time for two files

The second time, we considered 16, 19 and 22 attributes. Series1 is for multi-threaded execution and Series2 is for single-threaded execution.

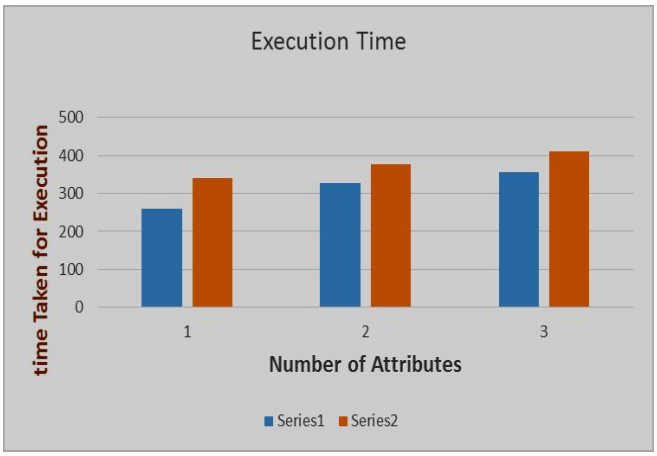

Fig. 5. Execution time for three files

The diagram above clearly shows that splitting the query into subparts and running it in multiple threads takes less time than running a single query for all attributes.

VII. CONCLUSION

This document shows how splitting a query can help you get results faster than working directly with the entire query at once. Therefore, splitting the query as vertical fragmentation and running these fragments as multithreads on a multiprocessor system has proven to be faster than running a single complete query at once. So the divide-and-conquer strategy works very well for such heavy or large query-related operations.

REFERENCES

[1] Maurizio Atzori, F. Bonchi, F. Giannotti, and D. Pedreschi, "Anonymity preserving pattern discovery," VLDB J., vol. 17, no. 4, pp. 703–727, 2018.

[2] M. Kantarcioglu and C. Clifton, "Privacy-preserving distributed mining of association rules on horizontally partitioned data,"IEEE

Trans. W. K. Wong, D. W. Cheung, E. Hung, B. Kao, and N. Mamoulis, “Security in outsourcing of association rule mining,” in Proc. 33rd Int. Conf. Very Large Data Bases, 2021, pp. 111–122

[3] J. Vaidya and C. Clifton, “Privacy preserving association rule mining in vertically partitioned data,” in Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2022, pp. 639–644

[4] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta, “Discovering frequent patterns in sensitive data,” in Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2022, pp. 503–512

[5] N. Li, W. Qardaji, D. Su, and J. Cao, “Privbasis: Frequent itemset mining with differential privacy,” Proc. VLDB Endowment, vol. 5, no. 11, pp. 1340–1351, 2023.

[6] F. McSherry and K. Talwar, “Mechanism design via differential privacy,” in Proc. 48th Annu. IEEE Symp. Found. Comput. Sci., 2022,pp. 94–103

[7] T. S. Ruprah, V. S. Kore and Y. K. Mali, "Secure data transfer in android using elliptical curve cryptography," 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), Chennai, India, 2017, pp. 1-4, doi: 10.1109/ICAMMAET.2017.8186639.

[8] Y. K. Mali and A. Mohanpurkar, "Advanced pin entry method by resisting shoulder surfing attacks," 2015 International Conference on Information Processing (ICIP), Pune, India, 2015, pp. 37-42, doi: 10.1109/INFOP.2015.7489347.

[9] M. K. Mali et al., "Evaluation and Segregation of Fruit Quality using Machine and Deep Learning Techniques," 2022 International Conference on Futuristic Technologies (INCOFT), Belgaum, India, 2022, pp. 1-8, doi: 10.1109/INCOFT55651.2022.10094447.

[10] N. P. Sable, V. U. Rathod, R. Sable and G. R. Shinde, "The Secure E-Wallet Powered by Blockchain and Distributed Ledger Technology," 2022 IEEE Pune Section International Conference (PuneCon), Pune, India, 2022, pp. 1-5, doi: 10.1109/PuneCon55413.2022.10014893.

[11] V. U. Rathod and S. V. Gumaste, "Role of Routing Protocol in Mobile Ad-Hoc Network for Performance of Mobility Models," 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), Lonavla, India, 2023, pp. 1-6, doi: 10.1109/I2CT57861.2023.10126390.

[12] N. P. Sable, V. U. Rathod, P. N. Mahalle and D. R. Birari, "A Multiple Stage Deep Learning Model for NID in MANETs," 2022 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 2022, pp. 1-6, doi: 10.1109/ESCI53509.2022.9758191.

[13] N. P. Sable, M. D. Salunke, V. U. Rathod and P. Dhotre, "Network for Cross-Disease Attention to the Severity of Diabetic Macular Edema and Joint Retinopathy," 2022 International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON), Bangalore, India, 2022, pp. 1-7, doi: 10.1109/SMARTGENCON56628.2022.10083936.