IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-7, July 2018 E-ISSN: 2347-2693

# Analysis of Regular-Frequent Patterns in Large Transactional Databases

**S. Rana**

Dept. of Computer Science and Engineering, Pundra University of Science and Technology, Bogra, Bangladesh

*Corresponding Author: sohelranacse052@gmail.com

**Available online at: www.ijcseonline.org**

Accepted: 21/Jul/2018, Published: 31/July/2018

***Abstract*—** Regular-frequent patterns are an important type of regularities that exist in transactional, time-series and any other types of databases. A frequent pattern can be said regular-frequent if it appears at a regular interval given by the user specified threshold in the transactional database. The regularity calculation for every candidate pattern is a computationally expensive process, especially when there exist long patterns. Currently the FP-growth algorithm is one of the most popular and fastest approaches to mining periodic frequent item sets. Therefore, in this paper we introduce a novel concept of mining regular-frequent patterns (RFP) in transactional databases. We introduce two mining techniques based on transaction number and also based on products or itemsets on the vertical data format. The efficiency is achieved by eliminating aperiodic or irregular patterns during execution based on suboptimal solutions. Our tree based structure helps to captures the database contents in highly compact manner. Our experimental results are highly efficient and scalable as well as improve the overall response time.

***Keywords*—** Frequent patterns, regular patterns, transactional databases, vertical data format.

## I. INTRODUCTION

A huge number of possible sequential patterns are hidden in databases.The Apriori algorithm was the first frequent pattern mining algorithm that was introduce by Agrawal, Imielinski, and Swami [1] in 1993. It generate length (k+1) candidate itemsets from length k frequent itemsets so it requires large amount of memory which reduce the efficiency of the algorithm. Then the FP-growth algorithm and tree based structure (FP-tree) are introduced by Han, Pei and Yin [3] which is more memory efficient as well as in execution time because it needs only two scans to find frequent item sets. A pattern is called **regular-frequent**, if it satisfies the user specified minimum threshold value which represent the minimum number of support (*Min_Sup*) of an item in database and an another threshold value which represent the maximum periodicity(*Max_Per*) of that item. It analyzes how regularly the products are being purchased by the customers and how the ratio of selling a product or item. An example is as follows:

*{Computer, keyboard} [Support=12%, periodicity=3 hours]*.

The above example of pattern says that 12% of customers have purchased the products 'computer' and 'keyboard' at least once in every three hour. Here, 12% represent the frequent and three hours represent the periodicity or regularity. It analyses customer buying habits by finding associations between the different items that customers place in their "shopping baskets". However, if we remove the item which is not satisfies the user specified minimum threshold value during the execution of algorithm then the FP-growth algorithm will be more faster and memory efficient.

The rest of the paper is organized as follows: Section II deals with the Related Work. Section III describes the Problem Definitions. Section IV describes the way of finding Regular Periodic pattern based on transaction number, Section V deals with the Regular Periodic pattern based on product or itemset on vertical dataset, Section VI shows the experimental Results and Discussions followed by Conclusion with future research directions in Section VII.

## II. RELATED WORK

The constraint-based frequent patterns first introduced by Agrawal, Imielinski and Swami in 1993 [1]. The Periodic patterns related problems has been widely studied in [2],[3],[5],[6],[7],[8],[9],[10],[11]. The frequent pattern tree (FP-tree) and the FPgrowth algorithm are proposed by Han et al. [3] which is more memory and time efficient. A pattern-growth algorithm, PFP-growth, has been proposed by Uday and Reddy [2] to discover periodic-frequent patterns. The top-k periodic-frequent patterns have been focused on Amphawan, Lenca and Surarerks [4]. Uday and Reddy [5] have extended [9] to discover those frequent patterns that have exhibited partial periodic behavior within a database. Discover the maximum length frequent patterns are shown in [14]. These patterns play a key role in discovering associations, correlations, episodes, multi-dimensional

 

patterns, diverse patterns, emerging patterns, periodic frequent patterns and so on. The basic model use those studies, however, remains the same. Discover all periodic patterns that satisfy user define minimum support. However, in this paper we proposed a regular frequent pattern mining approach based on transaction number and also based on product or item. We confine our work to reduce the computational cost and execution time by removing irregular items during execution process.

## III. PROBLEM DEFINITION

In this section, we describe the conceptual framework, basic notations and definitions of regular-frequent pattern mining . Let I= {$i_1$, $i_2$, ……, $i_n$ } be the set of items, and $X \subseteq I$ be a **pattern**. A pattern containing β number of items is called a **β-pattern**. A **transactional database** TDB over I is a set of transections.

**Table 1.** Example: Transactional Database

| Ts | Items | Ts | Items |
|---|---|---|---|
| 1 | *a, b, d* | 6 | *c, d, f, g* |
| 2 | *a, c, f* | 7 | *b, d, e* |
| 3 | *b, c, d, g* | 8 | *a, b, f, g* |
| 4 | *d, e, h* | 9 | *b, c, d, g* |
| 5 | *a, b, e* | 10 | *c, e, h* |

**Definition 1. (Frequent pattern X)** The number of transactions containing X in TDB is called as the **support** of X, denoted by *Sup(X)*.That is, $Sup(X) = |Ts^X|$ where, $|Ts^X|$ represent the number of transactions that containing X. A pattern X is **frequent pattern** if $Sup(X) \geq Min\_Sup$, where *Min_Sup* is the user define minimum support threshold value.

**Example 1.** Table 1 consist the transactional database with the set of items I = {a, b, c, d, e, f, g, h}. Here, 'bd' is a pattern, is called 2-pattern because it contains only two items. The patern 'bd' appears at the trunsection number 1, 3, 7 and 9. So, the Total number of support of 'bd' is denoted by Sup(bd)= $|Ts^X|$=|1, 3, 7, 9|=5.

**Definition 2. (Regular-Frequent pattern X)** The period of X is define as the differences between $Ts^X_{k+1}$ *and* $Ts^X_k$ .That is $P^X = Ts^X_{k+1} - Ts^X_k$ . During the computation of periodicity, the first transaction $Ts_f$ is 0 and the last transaction is $Ts_l$ = n+1 where, n is the number of transactions. The **regularity** of X is denoted as Reg(X) = maximum ($P_1^X$, $P_2^X$, ……, $P_n^X$ ). A frequent pattern is called **regular-frequent** if Reg(X) ≤ Max_Per, where Max_Per is the user define maximum periodicity threshold value.

**Example 2.** In the TDB of Table 1 the patern 'bd' appears at the trunsection number 1, 3, 7 and 9. So, the Sup(bd) is 4. Therefore, the periods for the pattern 'bd' are 1( = 1 - $Ts_f$), 2(= 3 - 1), 4( = 7 – 3), 2( = 9 – 7), and 2($Ts_l$ – 9), where $Ts_f$ =0 and $Ts_l$ =11. So, the Reg(bd) = Max(1, 2, 4, 2, 2) is 4. If the user define Min_Sup is 3 and Max_Per is 5, then 'bd' is called **regular-frequent** pattern.

## IV. REGULAR-FREQUENT PATTERN-GROWTH: BASED ON TRANSACTION NUMBER

### A. *Construction of RFP-list*

The RFP-list consists of three fields: - product or item names (*i*), total support (*f*), and the periodicity of item (*p*) at each entry. The *ts* column represents the last occurring transactions of all items in the RFP-list.

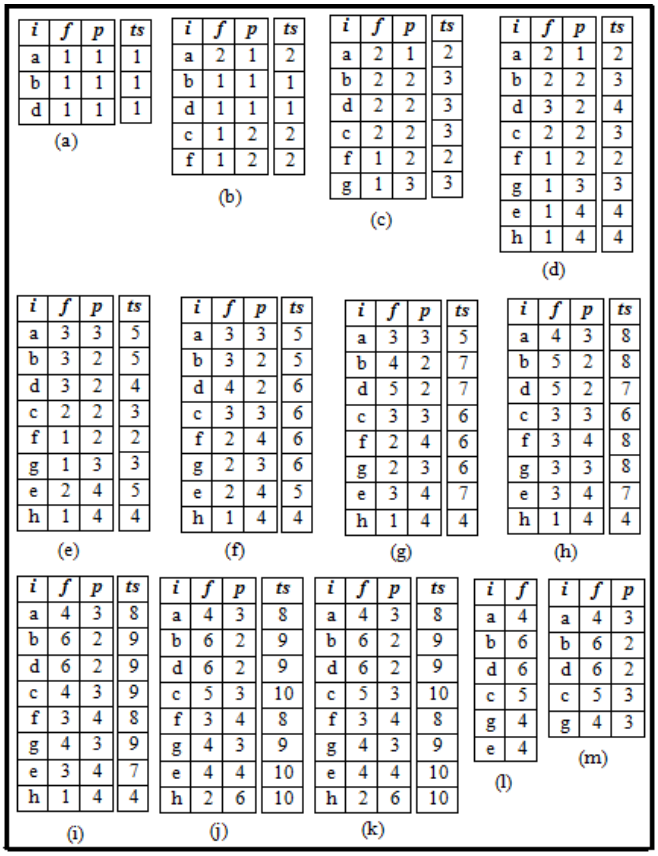


**Figure 1:** Construction of RFP-list. (a): After scanning first transaction, (b): After scanning second transaction. Similarly, (c) to (j) shows the views of after scanning 3rd to 10th transactions, (k): Updated periodicity of items, (l): Sorted list of frequent items for Min_Sup=4 and (m): Final list of regular-frequent items for Max_Per=3.

Figure 1 shows how the RFP-list is calculate for the TDB of Table 1. Figure 2(a) initialized the first transaction containing the items {a, b, d}. The second transaction contains the items {a, c, f}. Figure 2(b) is made by updating the raw that containing the item 'a' and adding two rows for new items 'c' and 'f'. Similarly, figure 2(c) to 2(j) represent the RFP-list after scanning third to last transaction respectively. 2(k) represents the updated periodicity by

 

comparing with last transaction. In figure 2(*l*), we remove the items that doesn't satisfies the user define Min_Sup threshold value 4. So, 2(*l*) is the final frequent pattern list. In figure 2(m), we remove the items which doesn't satisfies the user define Max_Per value 4. The figure 2(m) is the final regular-frequent list (RFP-list).

### *B. Construction of RFP-tree*

Only Regular-Frequent items will take part in the RFP-tree. The tree for the first transaction '1: a, b, d' according to the RFP-list order is shown in figure 2(a). The last-node d:1 carries the timestamp of this transaction. Similarly, after inserting second transaction '2: a, c, f' the tree is shown as 2(b) where, the item 'f' is not considered because 'f' is not in the final RFP-list. After inserting all element of TDB based on RFP-list, the final tree is shown in figure 2(c). Different pruning technique should be applied over RFP-tree.

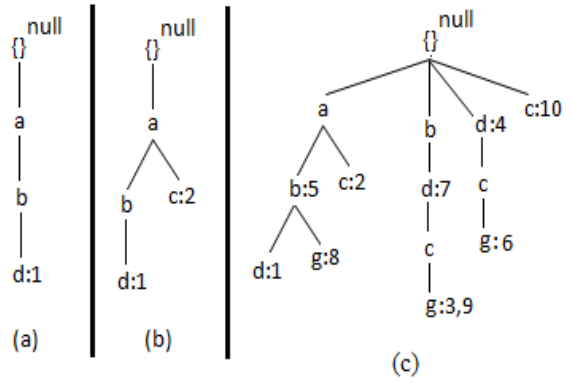


**Figure 2:** *Construction of RFP-tree for Table 1 based on RFP-list, (a) after scanning first transaction, (b) after scanning second transaction, (c) the final RFP-tree.*

### *C. Removing irregular(aperiodic) items during execution*

During execution, if we remove the item from the RFP-list when the periodicity is greater than the user defines maximum periodicity (Max_Per) (i.e. the items those not satisfies the condition of regularity) then the algorithm will be more memory efficient and faster. In figure 1(d) the periodicity of the item 'e' and 'h' is 4 but the Max_Per is 3. So, it doesn't support the condition of regularity. If we remove the item 'e' and 'h' from next steps, then the PFP mining algorithm will be faster and memory efficient, because we need not consider the irregular items in remaining steps. Similarly item 'f' is not considered at 3(f) to remaining steps. By applying this approach the RFP list mining steps are shown in figure 3 where the item 'e' and 'h' are not consider in 3(d) to remaining steps. The 3(l) represent the final regular-frequent list (RFP-list). Figure 3 is the just modification view of Figure 1 where, the irregular item sets are removed during construction of REF-list.

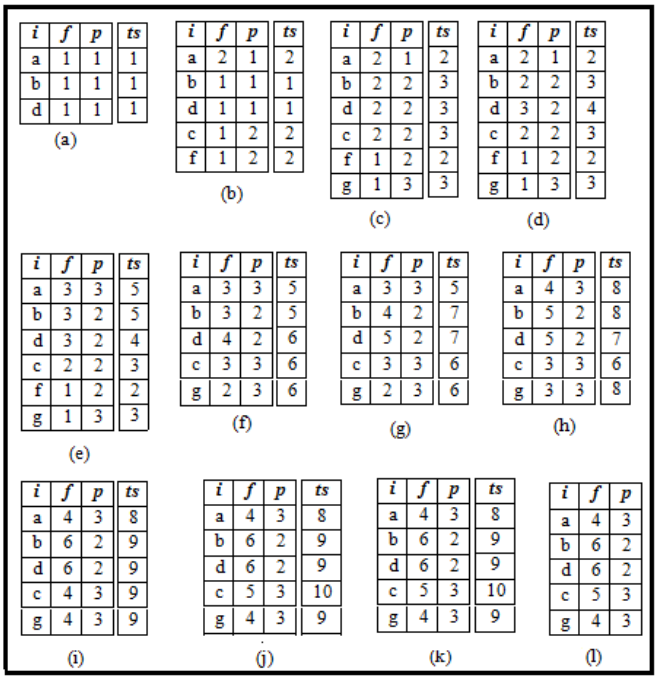


**Figure 3:** *Construction of RFP list (modification of Figure 1.) by removing aperiodic (irregular) items during execution.*

## V. REGULAR-FREQUENT PATTERN MINING: BASED ON ITEM OR PRODUCT

In this section we describe another proposed model for finding regular-frequent patterns based on items. The vertical view of the Table 1(original database) is shown in Table 2.

**Table 2.** Vertical Database (item based order of Table 1.)

| Item | Ts_ids | Item | Ts_ids |
|---|---|---|---|
| *a* | *1, 2, 5, 8* | *f* | *2, 6, 8* |
| *b* | *1, 3, 5, 7, 8, 9* | *g* | *3, 6, 8, 9* |
| *d* | *1, 3, 4, 6, 7, 9* | *e* | *4, 5, 7, 10* |
| *c* | *2, 3, 6, 9, 10* | *h* | *4, 10* |

Here, the product 'a' contains the transaction number 1, 2, 5 and 8. So, total support of 'a' is 4. Therefore, the periods for the pattern 'a' are 1( = 1 - $Ts_f$), 1(= 2 - 1), 3( = 5 – 2), 3( = 8 – 5), and 3($Ts_l$ – 8), where $Ts_f$ =0 and $Ts_l$ =n+1=11. So, the Reg(a) = Max(1, 1, 3, 3, 3) is 3. Similarly, Reg(b)=Max(1, 2, 2, 2, 1, 1, 2) is 2. The Table 3 shows the regularity and total support for each item of vertical database.

Now in Table 4 we remove the irregular and non-frequent items based on user define minimum support and maximum periodic value. If the user define Min_Sup is 4 and Max_Per is 3, then the item 'f' and 'h' are non-frequent because they support 3 and 2 times. So, we remove the item 'f' and 'h' from RFP-list. The Reg(X) of item 'e' is 4 but Max_Per is 3 so, we remove the item 'e' from the RFP-list shown in Table 4.

 

**Table 3.** Regularity and Support Calculation of Table 2.

| Item | $P^X$ | Reg(X) | Sup(X) |
|---|---|---|---|
| *a* | *1, 1, 3, 3, 3* | *3* | *4* |
| *b* | *1, 2, 2, 2, 1, 1, 2* | *2* | *6* |
| *d* | *1, 2, 1, 2, 1, 2, 2* | *2* | *6* |
| *c* | *2, 1, 3, 3, 1, 1* | *3* | *5* |
| *f* | *2, 4, 2, 3* | *4* | *3* |
| *g* | *3, 3, 2, 1, 2* | *3* | *4* |
| *e* | *4, 1, 2, 3,1* | *4* | *5* |
| *h* | *4, 6, 1* | *6* | *2* |

**Table 4.** Removing irregular and non-frequent items

| Item | Reg(X) | Sup(X) | Item | Reg(X) | Sup(X) |
|---|---|---|---|---|---|
| *a* | *3* | *4* | ~~*f*~~ | ~~*4*~~ | ~~*3*~~ |
| *b* | *2* | *6* | *g* | *3* | *4* |
| *d* | *2* | *6* | ~~*e*~~ | ~~*4*~~ | *5* |
| *c* | *3* | *5* | ~~*h*~~ | ~~*6*~~ | ~~*2*~~ |

Then, after removing non-frequent and irregular item set the final RFP-list is shown in Table 5.

**Table 5.** Final RFP-list

| Item | Reg(X) | Sup(X) |
|---|---|---|
| *a* | *3* | *4* |
| *b* | *2* | *6* |
| *d* | *2* | *6* |
| *c* | *3* | *5* |
| *g* | *3* | *4* |

## VI. EXPERIMANTAL RESULTS

In this section we are going to produce our experimental results. We compares the performance of RFP-growth using the transaction numbers based process shown in section IV(A) and RFP-growth by removing irregular item set during execution shown in section IV(C).

Both of those algorithms are written in GNU C++ and run with Ubuntu 14.4 on a 2.50 GHz machine with 4 GB of memory.

In Figure 4, The X-axis represents the maximum periodicity used to discover regular-frequent patterns and Y-axis represents the time to calculate the regular-periodic frequent patterns by removing irregular item sets after the execution of RFP-growth algorithm and removing the irregular item sets during the execution. We can see that, removing irregular items during execution is the more time efficient and scalable as well as. The execution time are calculated with different Max_Per value and fixed minimum support value =0.01%. Time is measured in second.

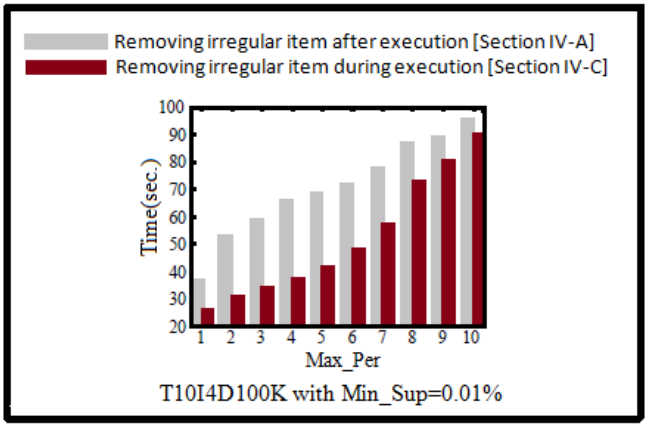


**Figure 4:** *Runtime comparison of RFP-growth algorithm by removing irregular itemset after executin [method discussed in section IV-A] and removing irregular itemset during execution [Section IV-C].*

The Figure 5 shows the number of regular-frequent patterns discovered at different Min_Sub (0.02%, 0.06%, 0.15%) and Max_Per(1-10) value. All experiment is performed on synthetic dataset T10I4D100K generated by IBM data generator which is often used for frequent pattern mining experiments. The X-axis represents the maximum periodicity used to discover regular-frequent patterns and Y-axis represents the number of regular-frequent patterns discovered at a given Max_Per value. The number of patterns are represents in thousands scale.

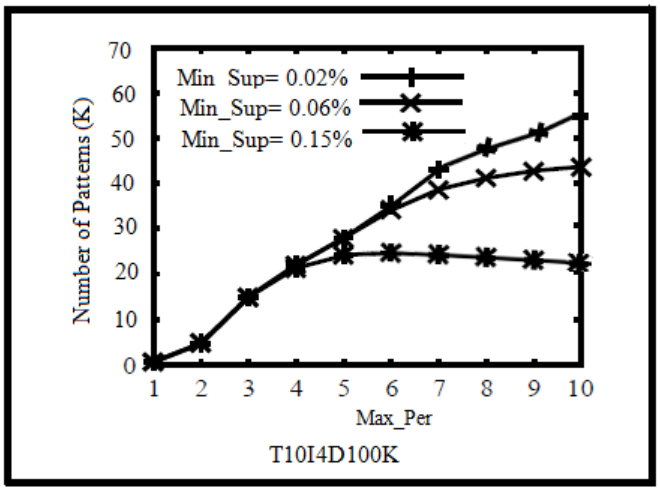


**Figure 5:** *Number of regular-frequent patterns discovered at different Min_Sup and Max_Per values.*

## VII. CONCLUSION AND FUTURE RESEARCH DIRECTIONS

Determining the regular-frequent pattern of an itemset is a computationally expensive process. In this paper, we have proposed a novel approach to reduce the computational cost

 

by saving memory space and execution time of mining regular-frequent patterns. Our system provide the way of mining regular-frequent pattern based on transaction number and also based on product or item on vertical dataset. We also generate the RFP-tree, a highly compact tree structure to capture the content of RFP-list. By removing irregular item set during execution of FP-growth algorithm, we make the system is more runtime efficient and scalable as well. Moreover, it is highly scalable in terms of runtime and memory consumption.

As a part of future work, we would like to extend our mining techniques to determine regular-frequent pattern, partial and semi-periodic frequent pattern in transactional, time series and any other types of databases. There are still many interesting research issues to be examined. The context based mining, sequential and structured patterns, constraint based classification, clustering in database are interesting topics for future work.

## REFERENCES


[1] R. Agrawal, T. Imielinski, A.N. Swami "*Mining Association Rules Between Sets of Items in Large Databases*", International Conference on Management of Data(ACM SIGMOD), pp. 207–216, 1993.

[2] R.U. Kiran, P.K. Reddy *"Towards efficient mining of periodic frequent patterns in transactional databases*", DEXA, pp. 194-208, 2010.

[3] J. Han, J. Pei, Y.Yin "*Mining Frequent Patterns without Candidate Generation"*, International Conference on Management of Data(ACM SIGMOD), pp. 1–12, 2000.

[4] K. Amphawan, P. Lenca, A. Surarerks *"Mining top-k periodicfrequent pattern from transactional databases without support threshold",* Advances in Information Technology, pp. 18–29, 2009.

[5] J.N. Venkatesh, R.U. Kiran, K, Reddy, M. Kitsuregawa *"Discovering Periodic-Frequent Patterns in Transactional Databases Using All-Confidence and Periodic-All-Confidence",* Springer International Publishing,Switzerland, pp. 55-70, 2016.

[6] S.K. Tanbeer, C.F. Ahmed, B.-S. Jeong, Y.-K. Lee, *"Discovering periodic frequent patterns in transactional databases*" Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) , Springer, Heidelberg, PAKDD, LNCS, Vol. 5476, pp. 242–253,2009.

[7] J. Pei, J. Han "*Constrained Frequent Pattern Mining: A Pattern-Growth View**", SIGKDD Explorations, Vol.4, Issue.1, pp.31-39.

[8] R.U. Kiran, M. Kitsuregawa, P.K. Reddy, *"Efficient Discovery of Periodic-Frequent Patterns in Very Large Databases"* Journal of Systems and Software, 2015.

[9] A. Surana, R.U. Kiran, P.K. Reddy *"An efficient approach to mine periodic frequent patterns in transactional databases"* Cao, L., Huang, J.Z., Bailey, J., Koh, Y.S., Luo, J. (eds.) PAKDD Workshops 2011, Springer, Heidelberg (2012), LNCS, Vol.7104, pp.254–266, 2012.

[10] V.M. Nofong *"Discovering Productive Periodic Frequent Patterns in Transactional Databases",* Springer-Verlag Berlin Heidelberg, 2016.

[11] R.U. Kiran, M. Kitsuregawa *"Discovering quasi-periodic-frequent patterns in transactional databases"* Bhatnagar V, Srinivasa S (eds) BDA 2013, LNCS, Springer International Publishing, Heidelberg, pp97–115, 2013.

[12] V. Kumar, V. Kumari "*Incremental mining for regular frequent patterns in vertical format"* International Journal of Engineering Technology, Vol.5, Issue.2, pp.1506–1511, 2013.

[13] M.J. Zaki *"Parallel and distributed association mining: A survey"* IEEE concurrency, pp. 14-25, 1999.

[14] T. Hu, S.Y. Sung, H. Xiong, Q. Fu *"Discovery of Maximum Length Frequent Itemsets"* Information Sciences, pp.69–87, 2008.

[15] N.Sethi, P.Sharma *"Mining Frequent Pattern from lLarge Dynamic Database Using Compacting Data Sets"* International Journal of Scientific Research in Computer Science and Engineering, Vol.1, Issue.3, pp.31–34, 2013.


## Authors Profile

*Mr. Sohel Rana* pursed Bachelor of Science in Engineering degree in Department of Computer Science and Engineering from Bangabandhu Sheikh Mujibur Rahman Science and Technology University, Gopalganj, Bangladesh. He is currently pursuing MSc. degree in Computer Science and Engineering at Rajshahi University of Engineering Technology (RUET), Bangladesh. He is currently working as Faculty Member in Department of Computer Science and Engineering, Pundra University of Science and Technology, Bogra, Bangladesh. He has published two research papers in reputed international journals. His main research work focuses on Data Mining, Pattern Recognition, Big data analysis and Artificial Intelligence based education. He has 2 years of teaching experience and 2 years of Research Experience.

 