# Incremental Mining for Regular Frequent Patterns in Vertical Format

Vijay Kumar G.[#1], Valli Kumari V.*[2]
[#]School of Computing, K L University
Guntur 522502, India
[1] gvijay_73@yahoo.co.in
[*]Department of CS&SE, AU College of Engineering
Visakhapatnam 530003, India
[2] vallikumari@gmail.com

***Abstract*—In the real world database updates continuously in several online applications like super market, network monitoring, web administration, stock market etc. Frequent pattern mining is a fundamental and essential area in data mining research. Not only occurrence frequency of a pattern but also occurrence behaviour of a pattern may be treated as important criteria to measure the interestingness of a pattern. A frequent pattern is said to be regular frequent if the occurrence behaviour is less than or equal to the user given regularity threshold. In incremental transactional databases the occurrence frequency and the occurrence behaviour of a pattern changes whenever a small set of new transactions are added to the database. It is undesirable to mine regular frequent patterns from the scratch. Thus proposes a new algorithm called RFPID (Regular Frequent Pattern Mining in Incremental Databases) to mine regular frequent patterns in incremental transactional databases using vertical data format which requires only one database scan. The experimental results show our algorithm is efficient in both memory utilization and execution.**

**Keyword-Frequent patterns, Regular patterns, Transactional database, Incremental database, vertical data format.**

## I. Introduction

Among several interesting patterns, frequent pattern mining [1, 2, 3] is one of the active research area in data mining and knowledge discovery process. Mining frequent patterns basically depends upon the support count (number of times a pattern appears in the database). The occurrence frequency of a pattern may not always represent the significance of a pattern. The occurrence frequency along with occurrence behaviour may be treated as an important measure in several online applications. In a super market the user may be interested in frequently sold items which are sold at regular intervals. To improve web site design the website administrator may be interested in more often hit web pages at regular intervals. Also, in stock market there may be a special interest for stock brokers and traders where the set of high stocks indices rise at regular intervals. Such frequent pattern regularity may also be a useful measure among other applications like network monitoring, dna sequence, telecommunications or sensor networks. From the above examples we observed that the occurrence behaviour of a frequent pattern at regular intervals plays an important role in a wide variety of applications.

In incremental databases, new transactions will add continuously to the transaction database. So the regular frequent patterns change whenever database is updated. In order to make best decisions, users may be interested on getting the latest regular frequent patterns from the updated database. Therefore, it has been an important issue to search various efficient ways to find latest frequent patterns when the transactions are being updated the database. There are several tree based applications to mine frequent patterns [4, 5, 6, 7] in incremental databases. Similarly there are few tree based applications to mine regular patterns [8, 10] from incremental transactional databases. Most of the above approaches were based on FP tree which requires continual adjustments of tree nodes whenever the database updates. So in this paper we introduce RFPID-algorithm to mine regular frequent patterns using vertical data format [11, 12] which performs better with large number of transactions and long item sets with one database scan. The other advantage of vertical data format is, it uses simple operations like unions, intersections, deletions, simple arrays etc., and also it judges non frequent and non regular item sets before generating the candidate sets. To the best of our knowledge there is no algorithm that mines to find out regular frequent patterns in incremental transactional databases using vertical data format.

The rest of the paper is organized as follows. In Section 2 we discuss about the related work. In Section 3 we define the problem of regular frequent patterns in incremental transactional databases. In Section 4 we describe the process of mining regular frequent patterns with RFPID-algorithm. In Section 5 we show our experiment results and finally we conclude the paper in Section 6.

## II. Related Work

Association rule mining was first introduced by Agrawal et al., [1, 2] to find frequent itemsets which satisfies the minimum support threshold to generate association rules from the frequent itemsets. The main disadvantage of Apriori algorithm is, it requires $k$ number of scans to generate $k$-itemset. Han et al., [3] introduced a highly compact data structure, FP-tree and FP-growth algorithm to mine frequent patterns in support descending order with only two database scans. Frequent pattern mining in incremental transactional databases have been studied widely over the last decade in data mining research and is based only on support threshold. Periodic patterns [13] are closely related to regular patterns but they differ with the type of data considered, time-series data or sequence data. Recently, Tanbeer et al., [8] introduced the problem of mining regular patterns in incremental transactional databases with a highly compact tree structure called IncRT-tree and a pattern growth approach based on the occurrence behaviour of a pattern. They also constructed an item header table called IncRT table consisting of five fields ($i$, $r$, $t_l$, $m$, $p$): item name ($i$), the regularity of $i$ ($r$), last *tid* where item $i$ occurred ($t_l$), a modification indicator of $i$ ($m$), and a pointer to the IncRT for $i$ ($p$). After inserting all transactions into the IncRT, $r$ for all items is calculated in the table by traversing the tree once. Whenever the database is updated, modification indicator $m$ will modify the one bit field and $t_l$ changes to the recent tid where item $i$ occurred. The node traversal pointers only visit each tail-node of the item and accumulate tids available in its tid-list in respective temporary arrays for every item from the tail node up to the root node. After traversal to the top-most item in the table, the complete list of *tids* for $i$ are obtained in their respective temporary arrays. Then the *periods* of $i$ are calculated to obtain regular itemsets.

Interestingness of a pattern may not always be measured only with support threshold or may not only with periodicity threshold. Therefore Tanbeer et al., [9] proposed a new approach to mine periodic-frequent patterns in transactional databases with the above two thresholds i.e., support and periodicity. In this paper PF-tree is constructed using ordinary node and a tail node. Each node in the PF-tree maintains parent, children and node traversal pointers. Irrespective to the type of node, they do not maintain the support count value in the PF-tree. The support and periodicity are maintained by PF-list consisting of three fields: item name ($i$), total support ($f$) and the periodicity of item $i$ ($p$). In order to become periodic-frequent, an itemset must satisfy both the following conditions (i) its support should not be less than a user given minimum support threshold value and (ii) its periodicity should not be greater than a user given maximum periodicity threshold value. Regular patterns as well as periodic frequent patterns satisfy the downward closure property i.e., if a pattern is found to be regular-frequent, then all of its non-empty subsets will be regular-frequent. So, in this paper we consider the above two papers [8, 9] to mine frequent patterns which occur at regular intervals in incremental transactional databases using vertical data format [11, 12] that satisfy the downward closure property.

## III. Problem Definition

In this section we describe the concepts of *regular frequent* pattern mining and define the basic definitions of the problem to obtain complete set of *regular frequent* patterns in incremental transaction databases.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items. A set $X = \{i_j, \ldots, i_k\} \subseteq I$, where $j \le k$ and $j, k \in [1, n]$ is called a pattern or an itemest. A transaction $t = (tid, Y)$ is a couple where *tid* is a transaction-id and $Y$ is a pattern. Let size ($t$) be the size of $t$, i.e., the number of items in $Y$. A transaction database $DB$ over $I$ is a set of transactions $T = \{t_1, \ldots, t_m\}$, $m = | DB |$ is the size of $DB$, i.e., the total number of transactions in $DB$. If $X \subseteq Y$, which means that $t$ contains $X$ or $X$ occurs in $t$ and denoted as $t_j^X$, $j \in [1, m]$. Therefore, $T^X = \{t_j^X, \ldots, t_k^X\}$, $j \le k$ and $j, k \in [1, m]$ is the set of all transactions where pattern $X$ occurs in $DB$.

### A. Definition 1 (frequent pattern X)

The total number of transactions in a $DB$ that contains pattern $X$ is called the support of $X$ i.e., $Sup(X)$. Hence $Sup(X) = | T^X |$, where $| T^X |$ is the size of $T^X$. The pattern $X$ is said to be frequent if its support is greater than or equal to user given minimum support threshold i.e., $Sup(X) \ge minsup(\delta)$.

### B. Definition 2 (regularity of frequent pattern X)

Let $t^X_{j+1}$ and $t_j^X$, $j \in [1, (m - 1)]$ be two successive transactions where frequent pattern $X$ appears. The variation between these two successive transactions can be defined as a period of $X$, say $p^X$ (i.e., $p^X = t^X_{j+1} - t^X$, $j \in [1, (m - 1)]$). For ease, to calculate the period of a pattern, we consider the first transaction in the $DB$ as null i.e., $t_f = 0$ and the last transaction is the $m^{th}$ transaction i.e., $t_l = t_m$. Let for a $T^X$, $P^X$ be the set of all periods of $X$ i.e., $P^X = \{p_1^X, \ldots, p_r^X\}$, where $r$ is the total number of periods in $P^X$. Then the regularity of a frequent pattern $X$ can be denoted as $Reg(X) = max\{p_1^X, \ldots, p_r^X\}$. A frequent pattern $X$ is said to be regular frequent if its regularity is less than or equal to user given maximum regularity threshold i.e., $\lambda$.

### C. Definition 3 (Regularity of a frequent pattern X in incremental databases)

Let $db+$ denotes the set of newly added transactions to the database $DB$. The updated database is denoted as $UDB$ ($DB \cup db_i+$), $i$ be the number of newly added transactions to the $DB$. Whenever the database is updated with $i$ transactions the first transaction-id i.e., null transaction $t_f = 0$ will be replace to $t_f = t_f + i$ ($t_f = 0 + i$) at the front end of the transactional database and $t_l = t_l + i$ at the rare end of *DB*. For example, in *db*+ (Table 5) two transactions 10 and 11 are added to DB. So *i* value is 2, the null transaction for the updated *UDB* is $t_f = 2$ and the last transaction is $t_l = 11$. After replacing the values of first transaction ($t_f$) and last transaction ($t_l$) the process continues to find support and $P^X$ for every $T^X$ to obtain latest regular frequent patterns from the *UDB*.

## IV. Mining Regular Frequent Patterns

In this section we describe the mining process of regular frequent patterns in incremental transactional databases using vertical data format requires only one database scan. To generate length-1 itemset our algorithm constructs an item header table called RFPID-table consists of four fields (*Itemset*, *Tid*, *Sup*, *Reg*). *Itemset* is an item name, *Tid* is the transaction list where the item occurs in various transactions, *Sup* is the support of the itemset and *Reg* is the regularity of an itemset. Each itemset consists of its own array to accommodate *Tids* and other intermediate results.

Let Table 1 be the transactional database DB in horizontal format which is somewhat similar to the database in [9]. Convert the above horizontal database into vertical database with one database scan to store all length-1 items with respective *tids*, *support* and *regularity*. For example, Let us consider the minimum support threshold value, $\delta = 5$ and maximum

TABLE I
Transactional Database DB

| Tid | Transaction |
|---|---|
| 1 | a, d, e, c |
| 2 | d, e, f, a, c |
| 3 | a, e, c |
| 4 | d, e, c |
| 5 | e, c, a, f |
| 6 | b, f |
| 7 | d, c, e, b |
| 8 | b, c, d, e |
| 9 | a, d, c, b |

**RFPID-Algorithm**

```
Input : DB, λ, δ
Output : Complete set of regular frequent Patterns
  Procedure :
              Let $X_i \subseteq I$ be a k-itemset
              $P^X_i = 0$ for all $X_i$
        For each $X_i$
           Update Sup
              If $Sup(X_i) >= \delta$
                 Find the period of $X_i$
                    $P^X_i = P^X_{i+1} - P^X_i$
                    $reg(X_i) = max(P^X_i)$
                       If $reg(X_i) <= \lambda$
                          $X_i$ is a regular frequent itemset
                       Else
                          Delete $X_i$
              Else
                 Delete $X_i$
        Repeat
      Find if any db+ exist
           If db+ exist
           Repeat the procedure recursively
           Else
```

Increase the *k* value using ‘and operation’ until no candidate is generated.

regularity threshold value, λ = 4. Itemset {*a*} occurs in (1, 2, 3, 5, 9) transactions in the DB. Since item {*a*} occurs in five transactions so the support of item {a} i.e., *Sup*(*a*) = 5. RFPID-table is generated to store all length-1 itemsets, satisfies both support and regularity thresholds. From the above transactional database DB the length-1 itemsets {(*a*), (*c*), (*d*), (*e*)} satisfied both support and regularity threshold values. Itemset (*b*) did not satisfy support threshold as well as regularity threshold and item set (*f*) did not satisfy support threshold so the itemsets *b* and *f* will be deleted, see Table 2. Once the RFPID-table is generated we remove the itemsets from it that do not satisfy the user given support and regularity thresholds. The RFPID-algorithm will repeat the procedure recursively whenever the database is incremented, by replacing the new values into first transaction $t_f$ and last transaction $t_l$ to obtain the latest regular frequent patterns. In the process some of the irregular or infrequent patterns may become regular or frequent in updated database.

TABLE II
RFPID-table with support ( δ = 5 ) and regularity ( λ = 4 ) for 1-itemset

| Itemset | Tid | Sup | Reg |
|---|---|---|---|
| a | 1, 2, 3, 5, 9 | 5 | 4 |
| ~~b~~ | ~~6, 7, 8, 9~~ | ~~4~~ | ~~6~~ |
| c | 1, 2, 3, 4, 5, 7, 8, 9 | 8 | 2 |
| d | 1, 2, 4, 7, 8, 9 | 6 | 3 |
| e | 1, 2, 3, 4, 5, 7, 8 | 7 | 2 |
| ~~f~~ | ~~2, 5, 6~~ | ~~3~~ | ~~3~~ |

TABLE III
Calculating Regularity of an Itemset

| Itemset | $P^X$ | Reg |
|---|---|---|
| a | 1, 1, 1, 2, 4 | 4 |
| c | 1, 1, 1, 1, 1, 2, 1, 1 | 2 |
| d | 1, 1, 2, 3, 1, 1 | 3 |
| e | 1, 1, 1, 1, 1, 2, 1, 1 | 2 |

TABLE IV
RFPID-table with support ( δ = 5 ) and regularity ( λ = 4 ) for 2-itemset

| Itemset | Tid | Sup | Reg |
|---|---|---|---|
| a, c | 1, 2, 3, 5, 9 | 5 | 4 |
| a, d | 1, 2, 9 | 3 | 7 |
| a, e | 1, 2, 3, 5 | 4 | 4 |
| c, d | 1, 2, 4, 7, 8, 9 | 6 | 3 |
| c, e | 1, 2, 3, 4, 5, 7, 8 | 7 | 2 |
| d, e | 1, 2, 4, 7, 8 | 5 | 3 |

TABLE V
Increment Database db+

| Tid | Transaction |
|---|---|
| 10 | a, d, c, b |
| 11 | d, e, f, c, b |

Table 3 shows how the regularity of a pattern is calculated. The itemsets (a), (c), (d), (e) are regular frequent itemsets and eligible to obtain length-2 itemsets which are shown in Table 4. The procedure continues for length-3 and so on until no candidates are generated for mining regular frequent patterns. As in our problem definition let us consider for ease the first transaction $t_f$ is a null transaction i.e., $t_f = 0$ and the last transaction is $t_l = 9$. Itemset {c, d, e} occurs in (1, 2, 4, 7, 8) transactions, so its support is 5 which satisfies the user given minimum support threshold. So, itemset {c, d, e} is frequent and it can be processed to find these frequent pattern are regular or not. In order to find out regularity we need to find the periods of this pattern. The periods of this pattern are (1 – $t_f$ =)1, (2 – 1 =)1, (4 – 2 =)2, (7 – 4 =)3, (8 – 7 =)1 and ($t_l$ – 8 =)1. The periods of the pattern {c, d, e} are (1, 1, 2, 3, 1, 1) and the regularity of this pattern is maximum (1, 1, 2, 3, 1, 1). Therefore the regularity of {c, d, e} is 3 which satisfies the user given regularity threshold. So itemset {c, d, e} is a length-3 regular frequent itemset. As our algorithm satisfies downward closure property, the itemsets which are not regular frequent will not update in the RFPID-table.

TABLE VI
Updated Database UDB (DB ∪ db+)

| Tid | Transaction |
|---|---|
| 3 | a, e, c |
| 4 | d, e, c |
| 5 | e, c, a, f |
| 6 | b, f |
| 7 | d, c, e, b |
| 8 | b, c, d, e |
| 9 | a, d, c, b |
| 10 | a, d, c, b |
| 11 | d, e, f, c, b |

TABLE VII
RFPID-table with support ( $\delta = 5$ ) and regularity ( $\lambda = 4$ ) for 1-itemset ($t_f = 2$ and $t_l = 11$)

| Itemset | Tid | Sup | Reg |
|---|---|---|---|
| ~~a~~ | ~~3, 5, 9, 10~~ | ~~4~~ | ~~4~~ |
| b | 6, 7, 8, 9, 10, 11 | 6 | 4 |
| c | 3, 4, 5, 7, 8, 9, 10, 11 | 8 | 2 |
| d | 4, 7, 8, 9, 10, 11 | 6 | 3 |
| e | 3, 4, 5, 7, 8, 11 | 6 | 2 |
| ~~f~~ | ~~5, 6, 11~~ | ~~3~~ | ~~5~~ |

Whenever the database is updated RFPID-algorithm updates the support count and regularity of an itemset. Let Table 5 is the increment database (*db+*) to the transactional database *DB* containing two transactions 10 and 11. From these two transactions itemset {b} turn into frequent itemset which is infrequent and irregular in the previous mining process. By changing the values of first transaction and last transaction we can obtain latest regular patterns from the incremental transactional databases. Our RFPID-table maintains all the transactions i.e., from transaction-id one to n$^{th}$ transaction. For ease we are showing the transactions from 3 to 11 in Table 6. The above Table 7 showing {b, c, d, e} are the most recent length-1 regular frequent patterns obtain from *UDB*. The mining process repeats the procedure recursively whenever the database is incremented.

## V. Experiment Results

We performed our experimental results over synthetic dataset (T1014D100K) and real datasets (mushroom and Kosorak) which are often used in frequent pattern mining and other interesting measures which are developed at IBM Almaden Quest research group and obtained from http://cvs.buu.ac.th/mining/Datasets/synthesis_data/. We compare the results of RFPID's with the existing PF-tree [9]. RFPID-algorithm requires only one database scan compared to PF-tree basically requires two database scans to construct PF-tree and PF-list. We also did experiments on updating database with different sizes. All experiments are done in java on windows XP contains 2.66 GHz with 2 GB of main memory.

In the first experiment we compared the results with PF-tree on T1014D100K which contain 100K transactions, 870 number of items and an average transaction length is 10.10 and on mushroom dataset containing 8124 transactions, 119 items and an average and maximum transaction length is 23. From the graphs in Fig 1 we can notice that RFPID-algorithm relatively takes equal time for low value thresholds and less time

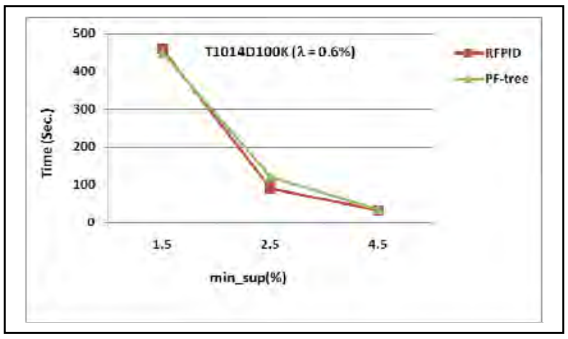

Fig. 1. (a) Execution time over T1014D100K

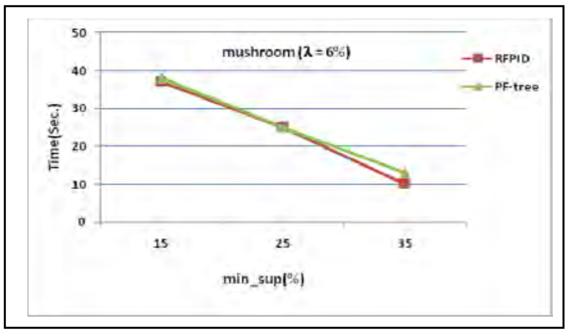

Fig. 1. (b) Execution time over mushroom

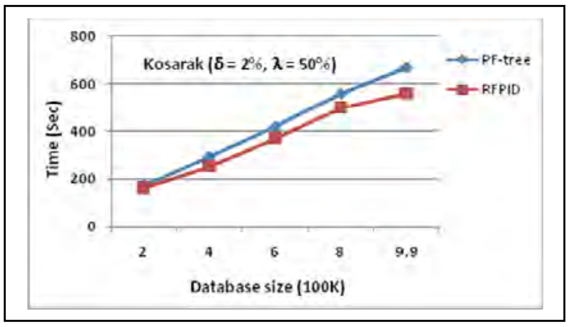

Fig. 2. (a) Execution time over Kosarak on different DB sizes

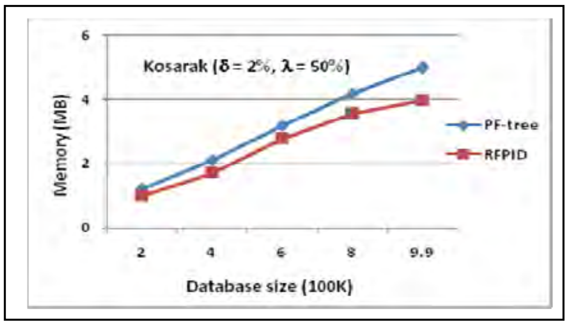

Fig. 2. (b) Memory usage over Kosorak on different DB sizes

for high value thresholds compared to PF-tree because of its RFPID-table compactness. The time and memory specify the total execution time and required memory with the increase in database size. Overall scalability of our algorithm is shown in Fig 2 out performs the scalability of PF-tree with different sizes of database. The performance of our algorithm is more efficient when the database is incremented.

## VI. Conclusion

In this paper, we propose a new algorithm RFPID which mines to obtain the complete set of latest regular frequent patterns in incremental transactional databases using vertical data format requires only one database scan. It is efficient over PF-tree because it is very simple and easy when compared with PF-tree. Our algorithm uses the advantages of vertical data format that needs simple calculations. Our experimental results show the out performance of our algorithm.

## References

[1] R. Agrawal, T. Imielinski, A. Swamy. Mining association rules between sets of items in large databases. In ACM SIGMOD Int. Conference on Management of Data, pp. 207 – 216 (1993).

[2] R. Agrawal, R. Srikanth. Fast algorithms for mining association rules. In Proceedings (1994) International conference on very large databases. (VLDB'94). pp. 487-499.

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

[4] Y. Chen, J. Guo, Y. Wang, Y. Xiong, Y. Zhu. Incremental Mining of Sequential Patterns Using Prefix Tree. In proc of PAKDD 2007, Springer. pp. 433-440 (2007).

[5] Md. M. Rashid, Md. R. Karim, B. S. Jeong, H. J. Chai. Efficient Mining Regularly Frequent Patterns in Transactional Databases. Springer Lecture Notes in Computer Science, vol 7238, pp. 258-271 (2012).

[6] S. J. Yen, Y. S. Lee, Y. T. Guo, J. Y. Gu. International Conference on Machine Learning and Cybernetics. ICMLC'2011. pp. 73-79 (2011).

[7] F. A. Anour et al., IMTAR: Incremental Mining of General Temporal Association Rules. Journal of Information Processing Systems. Vol 6 no.2 pp. 163-176. (2010).

[8] S. K. Tanbeer et al., Mining Regular Patterns in Incremental Transactional Databases. 12th International Asia-Pacific web conference, (2010) IEEE, DOI 10.1109/APWeb.2010.68, pp. 375-377.

[9] S. K. Tanbeer et al. Discovering Periodic-Frequent Patterns in Transactional Databases. Springer PAKDD, pp. 242-253 (2009).

[10] M. Y. Eltabakh et al. Incremental Mining for Frequent Patterns in Evolving Time Series Databases. Prudue University, prudue-e-pubs, Computer Science Technical Reports. pp-1-37 (2008).

[11] G. Y. Ming, W. Zhi-jun. A Vertical format algorithm for mining frequent itemsets. IEEE Transactions, pp. 11-13 (2010).

[12] M. J. Zaki, G. Karam. Fast Vertical Mining Using Diffsets, ACM SIGKDD. pp. 24-27 (2003).

[13] M. G. Elfeky, W. G. Aref, A. K. Elmagarmid. Periodicity Detection in Time Series Databases. IEEE Transactions on Knowledge and Data Engineering 17(7), pp. 875-887 (2005).