ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Association Rules Mining for Incremental Database

[I]Mehdi G. Duaimi, [II]Ahmed A. Salman
[I,II]Computer Science Dept., College of science, University of Baghdad, Baghdad, Iraq

Abstract
The association rules which are detected from a database only reflect the current state of the database. A great challenge in frequent itemsets mining from a huge data set is the fact that such mining occasionally generates a massive number of itemsets satisfying the (threshold) which is the minimum support. The main goal of the present work is solving the updating problem of association rules after a number of new records have been added to a database. The suggested approach towards incremental mining is to make use of previously mined knowledge and scan only incremented database. The presented algorithms try to reduce the number of scans of the database and maintain the association rules efficiently. A modified method named "Look Ahead for Promising items" (LAPI) is introduced for incremental mining. The proposed method uses information available from the TID Lists intersection to avoid the rescanning of the original database, it requires only the incremental database to be scanned.
Also, it prunes the supersets of a large itemset in DB as it is known to be small in the updated database. LAPI makes use of the negative border which is an indication of infrequent itemsets, whose subsets are all frequent. From the conducted results, the proposed algorithm has better pruning than that of Apriori algorithm. Besides, the techniques of Look Ahead strategy and TID intersection and negative border give a better performance for the LAPI technique.

Keywords
Data Mining, Incremental mining , Association rules, Databases, Apriori

I. Introduction

In this paper, we present a new idea to avoidrescanning the original database. Then we compute not only frequent itemset but also compute itemset that may be potentially large in an incremental database called "Promising frequent Itemset". Data mining is the process of extracting interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from large information repositories such as relational database, data warehouses, XML repository, Many people take data mining as a synonym for another general term, Knowledge Discovery in Database (KDD). Alternatively other people treat Data Mining as the core process of KDD [1]. One is called preprocessing, which is executed before data mining techniques are applied to the right data. After that comes another process called post processing, which evaluates the mining result according to user's requirements and domain knowledge. They actually processes work as follows [2]. The KDD methods are shown in Fig. 1. Usually there are three methods. One is called preprocessing, which is executed before data mining techniques are applied to the right data.

II. Association Rules Mining

Association rule mining searches for interesting relationships among items in a given data set. This section provides an introduction to association rule mining. Association rules are one of the promising aspects of data mining as knowledge discovery tool and have been widely explored to date, they allow to capture all possible rules that explain the presence of some attributes according to the presence of other attributes. Agrawal, Imielinski, and Swami first proposed association rule mining [3].

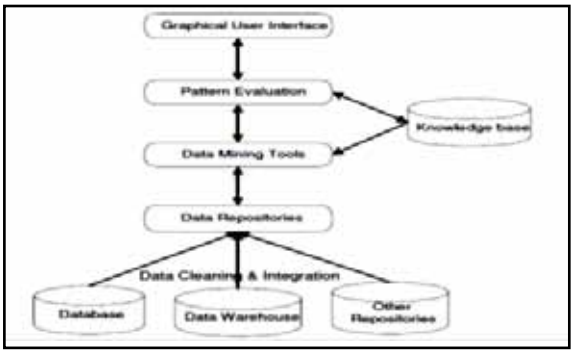

Fig 1. Knowledge Discovery in Database Processes

The problem of mining association rules can be divided into the following steps:
1. Find out all itemsets that have supports above the user-specified minimum support. Each such itemset is referred to as a large itemset. The set of all large itemsets in D is L, and Lk is the set of large k-itemsets.
2. Generate the association rules from the large itemsets with respect to another threshold, minimum confidence.
The second step is relatively straightforward.
Association rule mining is widely used in several business such as mobile data service environment [4], intelligent transportation system [5], market basket analysis [6, 7], hospital information system [8], etc.

III. Literature Review

1- Sanjeev Rao, Priyanka Gupta in 2012 [9] proposed a novel scheme for mining association rules pondering the number of database scans, memory consumption, the time and the interestingness of the rules. They removed the disadvantages of APRIORI algorithm by determining an FIS data extracting association algorithm which is proficient in terms of number of database scan and time. They eradicate the expensive step candidate generation and also avoid skimming the database over and again. Thus, they used Frequent Pattern (FP) Growth ARM algorithm that is the more effectual structure to extract patterns when database intensifies.

2- Kamrul et al. in 2008 [10] presented a novel algorithm Reverse Apriori Frequent pattern mining, which is a new methodology for frequent pattern design production of association rule mining. This algorithm works proficiently, when the numerous items in the enormous frequent itemsets is near to the number of total attributes in the dataset, or if the number of items in the hefty frequent itemsets is predetermined.

3- M. Hahsler, C. Buchta and K. Hornik in 2007 gave a unique approach [11] based on the notion to firstly define a set of

"interesting" itemsets and then, selectively generate rules for only these itemsets. The major benefit of this idea over swelling thresholds or filtering rules is that the number of rules found is considerably reduced whereas, at the same time it is not obligatory to increase the support and confidence thresholds which may lead to omitting significant information in the database.

4- Yanbin Ye et al. in 2006 [12] have instigated a parallel Apriori algorithm based on Bodon"s work [13] and scrutinized its enactment on a parallel computer. They followed a partition based Apriori algorithm to partition a transaction data set. They revealed that by fitting every partition into inadequate main memory for fast access and permitting incremental generation of frequent itemsets enhanced the performance of frequent itemsets mining.

IV. Generating Association Rules from Frequent Itemsets

Once the frequent itemsets from transactions in the database DB have been found, it is straightforward to create strong association rules from them (where strong association rules satisfy both minimum support and minimum confidence). This can be made using Equation (1) for confidence, which we show again here for completeness:

$$\text{confidence}(A \rightarrow B) = P(B|A) = x = \frac{\text{support\_count}(A \cup B)}{\text{support\_count}(A)} \quad (1)$$

The conditional probability is expressed in terms of itemset support count, where Support_ count(A∪B) is the number of transactions containing the itemsets A∪B, and support _count(A) is the number of transactions containing the itemset A. Based on this equation, association rules can be generated as follows:

For each frequent itemset l, create all nonempty subsets of l item.

For every nonempty subset s of l, output the rule" s→(l-s)" if $\frac{\text{support\_count(l)}}{\text{support\_count(s)}} \geq$ min conf, where min conf is the minimum confidence threshold.

Because the rules are generated from frequent itemsets, each one automatically satisfies minimum support. Frequent itemsets can be stored ahead of time in hash tables along with their counts so that they can be accessed quickly.

V. Algorithms for Incremental Mining of Association Rules.

Due to the increasing use of large data with high computation required for various applications, the importance of data mining has grown rapidly. Thus, it is necessary to collect and analyze a sufficient data properly before making any decisions. Since the amount of data being processed is large, it is important for the mining algorithms to be very computationally efficient.

Recently many important applications have created the need of incremental mining. The aim of incremental mining techniques is to re-run the mining algorithm on the only updated database.

The overall process of incremental mining is summarized in Fig. 2 However; it is obviously less efficient since previous mining rules are not utilized for discovering new rules while the updated portion is usually small compared to the whole dataset. Consequently, the efficiency and the effectiveness of algorithms for incremental mining are both crucial issues.

VI. General Description of The Approach

The current association rule mining technique is based on counting occurrence of all possible combinations of items. The main idea of an association rule in active database refers to optimizations that are made along the mining computations over the updated dataset. In active databases, many incremental techniques are developed for mining association rules. Fig. 3 illustrates the proposed system diagram.

Apriori computes frequent itemsets in a large database through several iterations based on a prior knowledge. It is applied to find all possible frequent itemsets.

The algorithm can do the following steps to get association rules:

1. Finding all itemsets that have supports above the user-specified minimum Support (min-sup). Each such itemset is referred to as a large itemset.

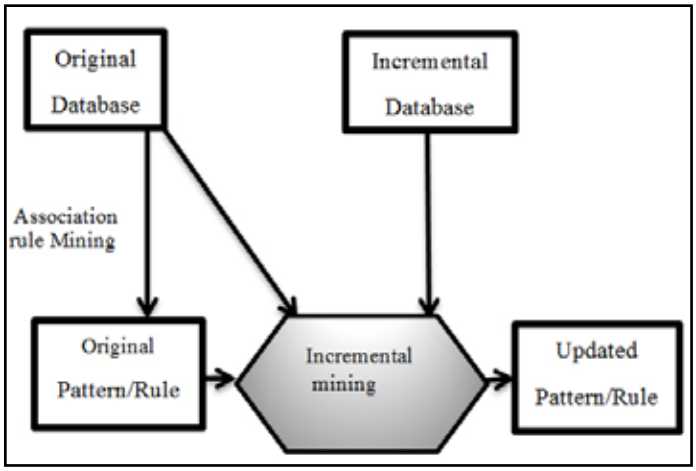

Fig. 2: Process of incremental mining

2. Generating the association rules from the large itemsets with respect to another threshold, minimum confidence (min-Conf).

The task of finding frequent itemset patterns is simple: given specified threshold, find all itemset patterns that are frequent, and their frequencies, for all patterns. Mining association rules can be decomposed into two steps: the first is generating frequent itemsets. The second is generating association rules. The main challenge in association rule is to identify frequent itemsets. Finding frequent itemset is one important step in association rule mining. Because the solution of second subproblem was rather straightforward. The rules discovered from a database only reflect the current state of the database. The frequent itemsets determined by Apriori can be used to determine an association that highlight general trends in the database: this has applications in domains such as market basket analysis. Apriori Algorithm and one for updating the database by using efficient, incremental algorithm for the maintenance of the association rules LAPI (Look Ahead for Promising Items)

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

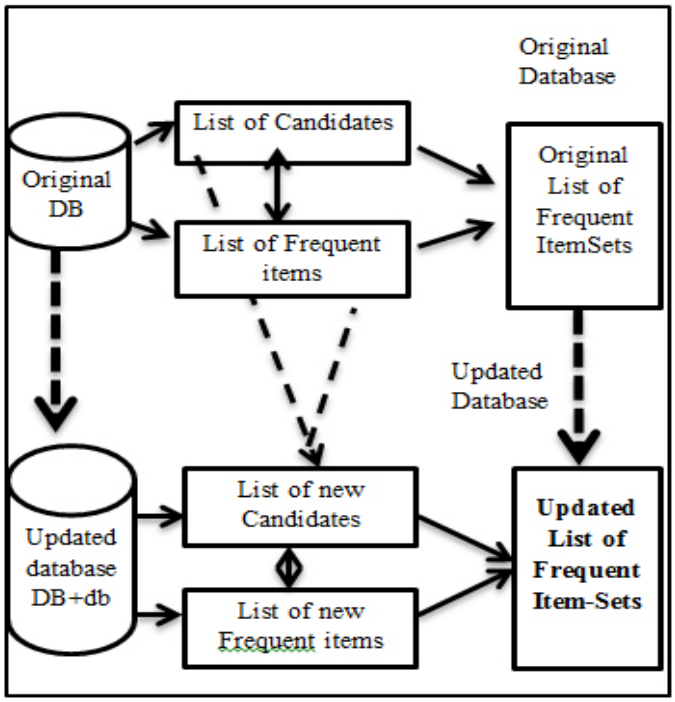

Fig. 3. Frequency updating for an incremental database

When updating the database, added items are calculated in the database for the elements of the updated account. Only the candidates for the next phase are done using an algorithm LAPI(Look Ahead for Promising Items) to decide to remove the less frequent items from the files that we have and check the upper superset in the Previous files of the database

A major challenge in mining frequent itemsets from a large data set is the fact that such mining often generates a huge number of itemsets satisfying the minimum support (min sup) threshold, especially when min sup is set low or when there are exist long patterns in the data set. Fig. 4 shows frequent items generation process.

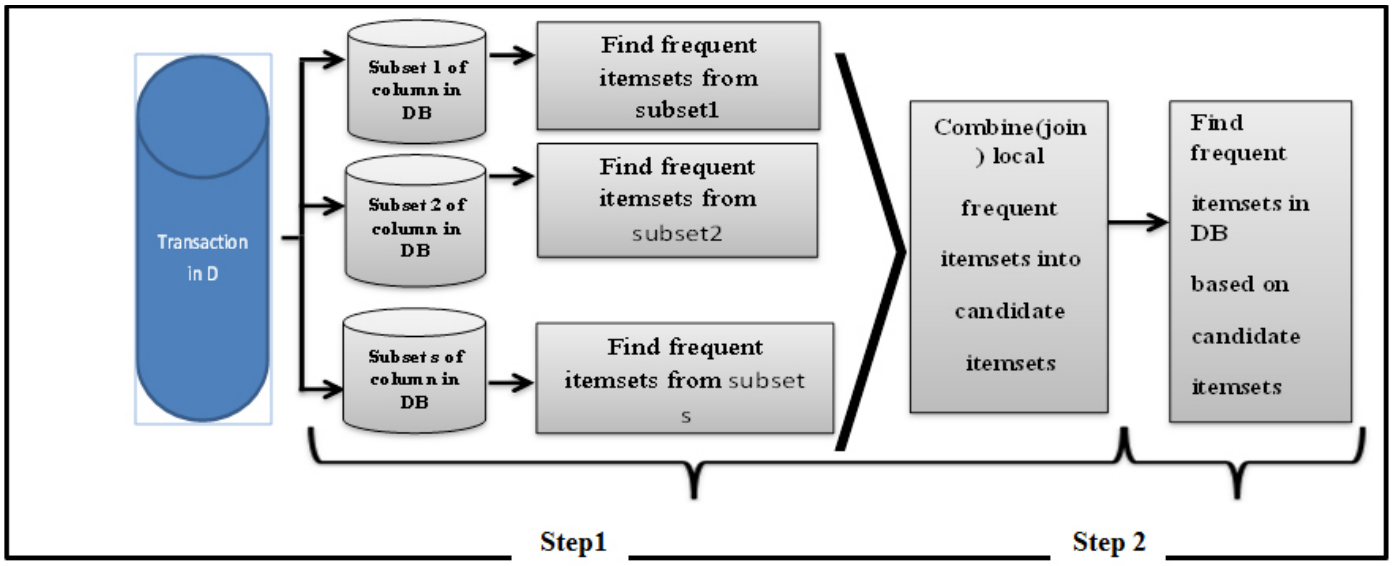

Fig. 4: Generation of frequent itemsets from a dataset.

VII. Candidates Items Filtering

Files are checked in order to be saved; they store the candidates of itemsets. The loop in this process continues wherever the read string is not empty. It reads a whole line each time. The new line is loaded into new string variable. A string array is used to split each reads whole string into its parts. The space is deleted by unique object. If the number of itemset is greater than threshold new file is opened and it stores the candidates. The file is called (CanFile), adding the strength by using items satisfied the constraints of Min_sup and Neg-sup are added to candidate files. Neg_sup is calculated by the following equations.

Neg_sup= Min_sup- J

Where J is a number specified by the user. It is like a threshold . If the number of itemset is greater than (Min _sup- J)and smaller than (Min _sup) new file is opened to include this itemset that satisfy this condition.

The read process will continue until reaching the end of the file to store the items that have the frequency larger or equal to threshold. In order to ensure that reading process continues line by line. It should not read an empty string. When it read an empty string the procedure exits and ends the reading process.

New variables are defined to handle split and compare processes. Checking whether the frequency of items is quite enough to stay as promised frequent items. If it is as mentioned it would be saved in a new file that include two items with their frequency. As shown in pseudo code lists (1).

VIII. Look Ahead for Promising Items (LAPI)

This technique uses Incremental efficient Algorithms LAPI (Look Ahead for Promising Items) to update large itemsets when new transactions are added to the set of old transactions. LAPI scans DB (old database) at most once and db (Incremental database)

exactly once. It is used to enhance the efficiency of Fast Update-based algorithms. That is the most important advantage of LAPI since it significantly reduces the number of candidates in db.
The LAPI algorithm is an FUP-based algorithm (FUP). It prunes the supersets of an originally frequent itemset in DB (old database) as soon as it becomes infrequent in the db (updated database) , rather than waiting until the k-th iteration. Only itemsets which are frequent in both db and DB+ (= DB ∪db) are taken to generate candidate itemsets to be further checked against db. if a k-itemset is frequent in db but infrequent in DB+. This can reduce the number of candidate itemsets in db. An additional set of unchecked itemsets has to be maintained during the mining process.

Pseudo code list(1) saving the concluded promised items from the join operation
Input: text files that contain candidate item sets
Output: new text file contain candidate item sets that support ≥ threshold
Begin:
Pass the path and name of file to the constructor to read the data of file
Read the first line of the file
While string is nothing do {Continue to read until you reach the end of the file}
Split the line that read into strings to check the frequent of items with threshold
Write the line that verify the condition in new file
Read new line
End while
Close the file that is already read
End

LAPI algorithm can do the following steps:
1. Scaning db and finding number of Occurrence for all 1-itemsets.
2. Promising – finding all itemsets in DB that are small in DB+ and removing the supersets of that itemset from DB.
3. Checking the large itemsets in DB whose items are absent in db support DB+ = support(DB + db)
4. Checking large itemsets in db to know if they are large in DB, those are large by definition.
5. Inspecting all large itemsets that are large in DB and have not been checked, to know if they are large in (DB + db).

IX. Update Problem Characteristics

Many algorithms have contributed to achieving incremental mining; however there are scopes to increase the efficiency of algorithms, development of new algorithms, and to reduce umber of scans of databases.
There are several important characteristics in the update problem.
1. The update problem is reduced to finding the new set of large itemsets. Then, a new association rules are computed from the new large itemsets.
2. An old large itemset has the potential to become small in the updated database.
3. An old small itemset could become large in the new database.
4. It finds the new large itemsets. All the records in the updated database, including those from the original database, have to be checked against every candidate set.

X. Calculation Of Frequent Items In Updated Database

Many incremental mining techniques have been proposed by different researchers in accordance with the need of applications which uses record based database and where database grows rapidly. The main goal of this method is to solve the updating problem of association rules after a number of new records have been added to a database. The overall approach towards incremental mining is to make use of previously mined knowledge and scan only incremented database. The algorithm tries to reduce the number of scans of the database and maintain the association rules efficiently.
The database is scanned to search about new transactions that are added to the database. Then it checked the updated database by using efficient incremental updating algorithm for mining association rules. An incremental association rule discovery technique called Look Ahead for Promising Items (LAPI) method is introduced. Mining of association rules can provide very valuable information, and improve the quality of business decisions. LAPI is proposed, which resolves the problem of updating the association rules when increasing transaction database without changing the minimum support and minimum confidence. Main features of this algorithm are those:
Frequent item sets of new transaction database are produced by Apriori TID List method, and candidate itemsets are pruned in effective ways. Hence the time for the database to be scanned is reduced. The efficiency of updating association rules is improved.
The first step is scanning database whether updated or not, if it is updated then the set of procedures will be applied on the updated part, as shown in pseudo code lists (2).

Pseudo code list(2) scan the database for new transactions
Input: database file
Output: new text file contain candidate of three item sets
Begin:
Connect the database with the program
con.Open () {open the connection of database}
sql = "SELECT all attribute" {to easy the fill data process}
da1.Fill (ds1, "db1") {to fill the dataset}
Count = ds1.Tables (0).Rows. Count
extract distinct attribute from tables where the database is updated
fill the dataset with data
review tables in data grid view through dataset
DataGridView1.DataSource = ds. Tables (0)
View the data in list box and items to it
First clear the list box from any previous items
First clear the list box from any previous items
Add the items to list box
offer the content of dataGridView into ListBox stored or execute the string then return the value of repeated element to concat with item
concat the name of field with number of repeat of each item
End

At this step, the updated frequencies are saved into a file. The new transactions need to be stored in a file in order to perform multi processes upon it, like extracting candidate for new

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

transactions. The process is shown in pseudo code list (3).

```
Pseudo code list(2) scan the database for new transactions
Input: database file
Output: new text file contain candidate of three item sets
 Begin:
      Connect the database with the program
     con.Open ()    {open the connection of database}
        sql = "SELECT all attribute"       {to easy the fill data
process}
     da1.Fill (ds1, "db1")                    {to fill the
dataset}
      Count = ds1.Tables (0).Rows. Count
      extract distinct attribute from  tables where the database
is  updated
      fill the dataset with data
      review tables in data grid view  through dataset
      DataGridView1.DataSource = ds. Tables (0)
      View the data in list box and items to it
      First clear the list box from any previous items
      First clear the list box from any previous items
     Add the items to list box
     offer the content of dataGridView into ListBox stored or
execute the string  then return the value of repeated element
to concat with item
     concat the name of field with number of repeat of each
item
  End
```

XI. Transaction Identifiers Lists Intersection (TID List intersection)

At this step, TID list method is adopted, for a single item and saved into a specific structure as list box. Then, it is saved in other specific structure as files. This method is very important and useful to reduce the number of scan of the database. Also it is precise for calculating the frequency of all items. It takes the index of items and retrieves the number of event for each item (the promised frequent item) in the whole database (the old and updated database).

In the case of two items are in a set, it takes the intersection of them and calculates the number of those intersections because this represents the number of their occurrences together. Hence when checking the frequency of items, the process calculates the intersection times against a threshold. While for three items intersection, it calculates the intersection of two items alone. It extracts the effect of them and combines it with the result of one item to get the total frequency, as shown in pseudo code list (4).

```
Pseudo code list (4) TID list technique to calculate the
occurrences of items
Input: database file
Output: new text file contain one item sets with index for each
item
Class Record
Begin:
Construct new class by adding new class
For each element in class1 (old database) and class2 (updated
database)
If Not test Is Nothing Then
Add in the location1 id of element
Else
Location1of record in old database = New class (Of Integer)
Add id of element in Location1of record in old database
End If
Next
If File is Exists = True Then Delete the File
End If
Using stream writer to write this contain to file
For Each Record In class list
For Each item As Integer In record old database of class
Word = word & (number of index)
Next
For Each item As Integer In record of updated database
Word = word & (number of index)
Next
SW.WriteLine (value of items & "{" & index & "} " &
Count of record in old database + Count of record in updated
database)
Next
End
```

XII. Pruning from the old database

At this stage, a procedure of pruning is introduced. This is another method used to reduce the data that should be scanned in the old database.

This method deals with the 1-itemsets whose support is zero in the updated database (db), but large in the old database (DB). If the item was previously small in DB, then it is also small in DB+db. On the other hand, if item is large in DB, it should be checked whether supportDB (item)≥ minsup in |DB+db| or not. This process does not wait until the kth iteration in order to prune the supersets of an itemset in L_{DB} that are small in L_{DB+db}. The process is shown in pseudo code list (5).

```
Pseudo code list(5) initial pruning
Input: items from list box
Output: new text file containing prune set of items to delete
them items from other files (Old Database)
 Begin:
New class List (Of String)
 For Each element As Record In class list
   If a Count of element.in updated part = 0 And Count of
element.in old part > 0    Then
     Add the value of the element in list
  End If
 Next
 For Each element As String In temp
    remove from class list Value of element and return single
value
 Next
 If the File is Exist Then    {store items in files}
   Delete the file
 End If
       Using Stream Writer to write the items to file
 For Each item As String In temp
  Write in file (item)
 Next
      End Using
  ("The file pruneset-" & s & ".txt has been saved ")
End Sub
```

XIII. Effective Processes of LAPI

LAPI involves different effective steps contribution in improving its performance. All of the applied processes is stated subsequently.

1. Pruning Process

The major benefit of LAPI over the previously known update algorithms is by pruning the supersets of a large itemset in DB as it is known to be small in the updated database, without waiting until the K th iteration.

When new records are added to the set of old transactions. LAPI employs a dynamic Look Ahead approach in updating the existing large itemsets by inspecting and removing those that will no longer stay large after the insertion of the new set of transactions.

The advantages of LAPI are that it scans the old and new database only once time and it generates and counts the minimum number of candidate itemsets in order to determine the new set of large itemsets. Also, it prunes an itemset that will become small from the set of generated candidates earlier by a look-ahead pruning strategy.

The major advance in the pruning process is making the candidates less than of original database, and this is a big challenge to reduce the times of scanning and consequently reduce the entire process time.

2. TID Intersection process

The method for each candidate itemset defines a structure called TID list. This TID list includes the item name and all the indexes of occurrence of this item. This is useful for the intersection operation to find the frequency of each item. Fig. 5 shows a part of the items and their TID list. The frequency of 1-itemsets is counted. a TID list is created for each 1-itemset in db. A TID list for an itemset X is an ordered list of the (TID transaction identifiers) of the transactions in which the items are present. The frequency of an itemset X is the length of the corresponding TID list.

LAPI uses intersection operation of item's TID list to replace scanning database db and DB. Hence, the time of scanning database db and the number of candidate itemsets are reduced for candidate itemsets of both large old and incremental database. Therefore, the time of producing candidate itemsets is reduced.

Computing maximal frequent itemsets in the updated database is made to avoid the generation of many unnecessary candidates. The efficiency of updating association rules is improved. This method improves the performance of the LAPI algorithm.

The efficiency of method lies in taking advantage of becoming minor candidate for election the item gather or less time scanning database.

item=M.S.N. {207,209,214,218,220,225,226,227,228,229,230,232,235,237,239,243} NO. of records =16
item=D.S.N. {213,244,246,775,777,778,779,780,781,782,783,784,785,786,787,788,789,790,791,792,793,794,795,796,797,798,799,806,808,809,811,812} NO. of records =32

Fig. 5. TID list items of High Degree column

3. Negative Borders Process

A negative border is an indication of infrequent itemsets, whose subsets are all frequent. The database is Scanned to count support for frequent itemsets, and for itemsets in a negative border. If no itemset in the negative border is frequent, then no more passes over the database are needed. Otherwise, the database is scanned to count support for candidate itemsets generated from the negative border.

A majority of those itemsets would have been present in the original negative border or frequent itemset. Only those itemsets which were not covered with the negative border need to be checked against the whole TID Lists. As a result, the size of the candidate set in the final scan could potentially be much smaller as compared to that of Apriori algorithm.

Assuming that the two thresholds, minimum support and confidence, do not change, LAPI algorithm can guarantee to discover frequent itemsets. From the experiment, The current algorithm has better running time than that of Apriori algorithm.

XIV. LAPI AND APRIORI COMPARISON

From the results, the proposed algorithm has better running time than that of Apriori algorithm.

In order to evaluate the performance of A LAPI algorithm, experiments are conducted using an original database of 1000 records. LAPI and Apriori techniques are implemented using Visual Basic.net. Those algorithms are performed on Windows 7 Professional platform, and processor intel(R) Core(TM) i3-2350M CPU 2.30GHz, RAM 4.00 GB. Table 1 shows the accuracy for both LAPI and Apriori techniques. While, Table 2 and Fig. 6 represent LAPI algorithm and A Priori algorithm execution time ratio. The result shows that the performance of LAPI algorithm is better than Apriori algorithm. Moreover, the better the performance efficiency of LAPI algorithm is, the greater the number of transactions in new database db is. Because it takes larger time for scanning new database db with Apriori algorithm. But the number of times for scanning new database db with LAPI algorithm is once. Besides, the techniques of Look Ahead strategy with TID intersection and negative border give a better performance for the LAPI method.

Table 1 : LAPI And Apriori Accuracy Measure

Item sets	Accuracy	
	LAPI	Apriori
1-item	100%	100%
2-items	100%	100%
3-items	100%	100%

Table 2 : LAPI And Apriori Execution Time

Incremental size	Time(sec)	
	LAPI	Apriori
10%	2	19
25%	3	20
50%	5	24

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

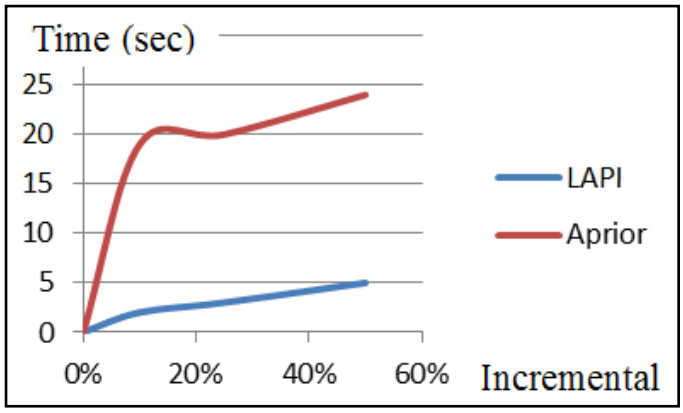

Fig. 6: Computed time for LAPI and Apriori for each incremental update size

XV. Conclusions

Extracting association rules is an important class of data mining, and associ¬ation rules have a wide area of usage. Although many efficient algorithms have been proposed up to now, discovering and generating association rules is still a computationally expensive operation in large databases.

The final task is to conclude points through comparison between the proposed algorithms and more general algorithm, Apriori, to determine the strong and weak points in proposed algorithms. The results show that:

1- Firstly the proposed LAPI algorithm gives high power to Apriori and when it is used because it reduces scan time, and memory size that Apriori needs.
2- LAPI uses several techniques like TID list , TID intersection to calculate the frequency of items , and prune method to reduce the candidates of supersets of itemsets, and it depends on the concept of joining operation.
3- It builds and generates strong association rules faster than Apriori, while the accuracy stills as Apriori algorithm.
4- There is no candidate to be generated from non-promising itemsets, because all reduced transactions are used form k-itemsset table, and it ignores any superset in k-itemset table that has a number less than that of the threshold.
5- It uses TID intersection between attributes of a table to calculate the occurrence of more than 1-itemset in the same transaction in old and incremental database, and it uses TID list to find the frequency of 1-itemset in the incremental database instead the scan operation.
6- Using pruning method to reduce the time needed to be scanned, and reduce itemsets where that lead to enhance the performance.
7- From the above characteristics of proposed algorithm, it is conducted that : TID list and TID intersection with pruning give fast processing since they reduce the execution time while the accuracy stays same as in Apriori.

References

[1] *Chen, M.-S., Han, J., and Yu, P. S. Data mining: an overview from a database perspective. Ieee Trans. On Knowledge And Data Engineering 8, 866- 883. 1996.*
[2] *Han, J. and Kamber, M. Data Mining Concepts and Techniques, Morgan Kanufmann. 2000.*
[3] *Charu C. Agrawal and philip S. Ya, "Mining large itemesets for association rules", Bulletin of the IEEE computer society Technical Committe on Data Engineering, 21(1) March 1998.*
[4] *P. Pawar, A. K. Aggarwal, "Associative Rule Mining of Mobile Data Services Usage for Preference Analysis, Personalization & Promotion", in proc. of WSEAS International Conference on Simulation, Modeling and Optimization, Izmir, Turkey, Sept. 2004.*
[5] *Juan X., Feng Y., Zhiyong Z.: "Association Rule Mining and Application in Intelligent Transportation System"; Preceedings of the 27th Chinese Control Conference, 538-540, China (2008).*
[6] *Brin.S., motwani,R ,and silverstein,C. beyond market: Generalizing Association Rules to Correlations. In processing of the ACM SIGMOD, 265-267, Conference (1997).*
[7] *Bing Liu , Wynne Hsn , and Yiming Ma , "Mining association rules with multiple minimum supports ", proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, P 337-341. August 15-18, 1999.*
[8] *Elfangary, L. and Atteya, W.A ,"mining medical Databases Using Proposed Incremental Association Rules Algorithm(PIA), Digital Society, page(s): 88-92, second International Conference 2008.*
[9] *S. Rao, and P. Gupta "Implementing improved algorithm over Apriori data mining association rule algorithm", IJCST, vol. 3, pp.489-493, ISSN: 2229-4333., 2012.*
[10] *Kamrul, Shah, Mohammad, Khandakar, Hasnain, and Abu, "Reverse Apriori Algorithm for Frequent Pattern Mining", Asian Journal of Information Technology, pp.524-530, ISSN: 1682-3915., 2008.*
[11] *M. Hahsler, C. Buchta and K. Hornik, "Selective Association Rule Generation", in Proceedings on Computational Statics, Springer., 2007.*
[12] *Yanbin Ye and Chia-Chu Chiang, "A Parallel Apriori Algorithm for Frequent Itemsets Mining", Fourth International Conference on Software Engineering Research, Management and Applications, pp. 87- 94., 2006.*
[13] *F. Bodon, "A Fast Apriori Implementation", In Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, vol. 90. 2003.*

Mehdi G. Duaimi was born in Babylon, Iraq, 1968. He received his B.Sc., M.Sc., and Ph.D. degrees, all in computer sciences from Nahrain University, Baghdad, Iraq at 1992, 1995, and 2007 respectively. In 2009 he joined the University of Baghdad, where he is now an instructor in the Department of Computer Sciences. During the 1999 – 2009 years, he was at the Iraqi commission for computers and informatics - Baghdad where he worked as a database designer and as an instructor. He has some publications related to data mining and information retrieval. His current research interests include areas like data mining, databases and artificial intelligence.