Mining Profile based sequential patterns in Progressive Databases

K.M.V. Madan Kumar[1*], B. Srinivasa Rao [2]

[1]Research Scholar, ANU, Guntur & Department of IT, TKR College of Engineering & Technology, Hyderabad.

[2]Research Guide, Acharya Nagarjuna University, Guntur.

ABSTRACT

Sequential Pattern Mining with incremental approach received a good attention in the recent past. Removing the obsolescence in patterns and reducing the domination of age-old data need to be focused for extraction of recent trends hidden in data. With userdefined 'Period of Interest', an approach for mining sequential patterns with progressive databases was proposed by J.-W. Huang et al.[8]. An extension of this approach is proposed in this paper, so as to allow the occurrence of repeated in a pattern may associate with different items at distinct positions within a pattern. Such distinct behavior exhibited by an item at different timestamps reveals new knowledge. Our algorithm also mines the profiles of the customers and produces a compact set of profile-based sequential pattern valid in any period of interest with the occurrence of repeated items.

Keywords:*sequential patterns, progressive databases, profile-based patterns*.

1. Introduction

The problem of discovering sequential patterns is to find inter-transactional patterns revealing the frequent occurrence of a set of items followed in sequence by zero or more sets of items mined from a time-stamp ordered transaction database. Although there have been many studies on the mining of sequential patterns in static and dynamic databases, they ignore the domination of age-old data in producing patterns. An approach that successfully mines sequential patterns valid in a period of interest was proposed by J. W. Huang et al.[8]. Their approach not only focuses on data in a period of interest, but also has the effect of literal removal of obsolete data and inclusion of incremental data. However, their approach limited to extraction of patterns containing non-repetitive items.

In this paper, we allow the items to be repeated in order to capture new knowledge that may be exhibited by any repeated item associating itself with different items and different time-stamps within a pattern. In addition, our algorithm considers the profiles of customers and mines the profiles together with their data of purchases in their sequential visits to a shop [10, 11]. The rest of the paper is organized as follows: the related work is presented in brief in section-2, our proposed algorithm P-Pisa is given in section-3 with a solution to a typical example, and conclusions in section 4.

2. Related works

Sequential pattern mining was first addressed in [1]. As the problem: " Given a sequence database, where each sequence consists of a list of ordered item setscontaining a set of different items, and a user defined minimum support threshold, sequential pattern mining is to find all subsequences whose occurrence frequencies are no less than the threshold from the set of sequences." The formal definition is given as follows.

Definition 1. Let I=$\{x_{1,} x_2, \dots \dots \dots\ x_n\}$ be a set of different items. An element e, denoted by $(x_i x_j \dots \dots)$, is a subset of items I which appear at the same time. A sequence s, denoted by<e_1, $e_{2,}$ .... e_m > is an ordered list of elements. A sequence database DB contains a set of sequences, and |DB| represents the number of sequences in DB. A sequence A= $\langle a_{1}, a_{2}, \dots a_n \rangle$ is a subsequence of another sequence B= $\langle b_{1,} b_{2,} \dots b_m \rangle$ If there exists a set of integers, $1 \leq i_1 < i_2 < \dots\ i_n \leq m$,such that a_1 bi1, $a_2 b_{i2}$ … and a_n b_{in}(where a is subset of b for the above statement). The sequential pattern mining can be defined as " Given a sequential database DB and a user-defined minimum support(min_sup), find the complete set of subsequences whose occurrence frequencies ≥ min_sup times of |DB|. " For ease of readability, we omit parentheses "( )" of an element containing only a single item.

The sequential pattern mining with a static database finds the sequential patterns in the database in which data do not change over time [2,3, 4, 5, and 6]. On the other hand, the sequential pattern mining with an incremental database corresponds to the mining process where there are new data arriving as time goes by (i.e., the sequences database is incremental) [7]. As for the sequential pattern mining with a progressive database, new data are added into the database and obsolete data are removed simultaneously. Therefore one can find the most up-to-date sequential patterns without being influenced by obsolete data. It is noted that the sequential pattern mining with a static database and with an incremental database are both are special cases of the progressive sequential pattern mining. It is noted that users are usually more interested in the recent data than the old ones. However, if a certain sequence does not have any newly arriving elements, this sequence will still stay in the database and undesirably contribute to |DB|. Therefore, when new sequential patterns are generated, the new patterns which appear frequently in the recent sequences may not be considered as frequent sequential patterns because |DB| is never reduced. In view of this, the infrequent sequential patterns whose timestamps are obsolete should be removed. Sequential pattern mining with a progressive database is widely used in many fields. Hence an algorithm Pisa, which stands for Progressive mIning of Sequential pAtterns, corresponding to the mining in a progressive database is given in [8]. Pisa takes the concept of a period of interest (POI) into consideration. The definition can be illustrated as follows.

Definition 2: POI is a sliding window, whose length is a user-specified time interval, continuously advancing as the time goes by. The sequences having elements whose timestamps fall into this period, POI, contribute to the |DB| for current sequential patterns. On the other hand, the sequences having only elements with timestamps older than POI should be pruned away from the sequence database immediately and will not contribute to the |DB| thereafter.

Hence, users can identify the latest information without the influence of old data and recognize the most up-to-date sequential patterns. However, their approach of finding sequential patterns does not support the concept of a pattern which can contain more than one occurrence of a single item or item sets. It is discussed clearly in example1 given below. Also, the patterns are not up to the mark as observed with a database containing profile data. Discussed in example 2.

Example 1:

C-ID	SEQUENCE						
01	AB	C	B	E	B		F
02	B		C		C	B	
03	B	C	AB		E	B	F
04	C		AB	C		C	
05	B	AC	B	E	C		B

t_1 t_2 t_3 t_4 t_5 t_6 t_7

< --------------------Time -------------------→

Table 1. Database for Example 1

The database in table1gives BC, BE and BF as one of the frequent sequential pattern when Pisa [8] is applied, which states that C or E or F can occur after B. Let us assume heart attack causes death (suppose B=heart attack, F= death, C& E= any diseases) so it says that C or E (disease) or F (death) occurs after B (heart attack). The above statement does not reveal the exact interesting pattern as assumed above. But with repetition of items in a sequence gives BCBEBF as a frequent sequential pattern which gives very accurate observed output as C occurs after first B. E occurs after second B, and F occurs after third B. so this gives the very valuable result.

Example 2:

Given the database in table 2, whereM=MALE; F=FEMALE, G=GUJARAT; K=KERALA

By sequential pattern mining without profile based technique (without considering the profile of each customer) we get BCBE as one of the frequent sequential pattern (assuming min_sup as 2).

C-ID	SEX	CITY	SEQUENCE			
01	M	G	AB	C	B	E
02	F	K	B		C	
03	M	G	B	C	AB	
04	M	K	C		A	
05	F	K	B	AC	B	E

t_1 t_2 t_3 t_4 t_5 t_6

< --------------------Time -------------------→

Table 2. Database for Example 2

In the above example, if any retailer chain shop is opened once again in GUJARAT or KERALA, after the sequential purchase of BCB, the owner will invest in E. but investment in E is not required actually. The mined output of the progressive db given above having each transaction consisting of customer profile will be:

M G=> A, BCB,BB, CB, (AB)

F G=>; M K=>;FK=> BC

The above result justifies that there is no requirement for the investment on product E if opened both in GUJARAT and KERALA. Hence we consider only M in G and F in K.

3.Profile Based Progressive Mining of Sequential Patterns

Problem Definition:

"Given a progressive database of transactions together with profile data of customers, finding the complete set of profile-based frequent sequential patterns with repetitive items, which appears in data sequences whose count is atleastminsup times p where minsup is the user-defined min support and p is the number of data sequences in the predefined period of interest"

To solve the profile based progressive sequential pattern mining problem, we propose a progressive mining algorithm P-Pisa(Profile Pisa), an extension of Pisa given in [8]. P-Pisa maintains a PPS-tree (Profile Progressive Sequential tree) to keep the information of the progressive database (having profile information of the customers along with their purchase sequences) and up-to-date sequential patterns in each POI. We will first introduce PPS-tree in section 3.1. Next, the algorithm P-Pisa will be explained to demonstrate how P-Pisa works will be provided in section 3.3.

3.1. Data Structure: PPS- Tree

The core of algorithm P-Pisa is PPS-tree. PPS- tree not only contains the information of all sequences in a progressive database with the profile but also helps P-Pisa to generate frequent sequential patterns in each POI. The nodes in PPS- tree can be divided into two different types. They are a root node and common nodes. The Root node is the root of PPS- tree containing nothing but a list of common nodes as its children. Each common node stores two items of namely information node label and a sequence list, as shown in Figure 1. The label is the same as the element in a sequence. The sequence list stores a list of sequence IDs to represent the sequences containing this element. Each sequence ID in the sequence list is marked by a corresponding profile and the timestamp.

3.2. Algorithm PPisa

The main concept of PPisa is to progressively update the information of each sequence and each candidate sequential pattern in the database. To achieve this goal, PPisa utilizes PPS- tree to store all sequences from one POI to another. When receiving the elements at the arriving timestamp, say *t+1*, PPisa traverse the original PPS-tree of timestamp t in post order (children first, then the node itself) and updates the PPS-tree of timestamp t. while traversing PPS-tree, PPisa 1) deletes the obsolete elements from, 2) updates current sequences in, and 3) inserts newly arriving elements into the PPS- tree of timestamp *t*. the detailed algorithm PPisa is shown in Figure 2. Firstly PPisa gets the profile variable. Then it gets the elements of all sequences at PPisa moves forward to the next timestamp until there is no newly arriving element in the progressive database. The main procedure, traverse, is used to traverse the PPS-tree of timestamp *t* and transform it into the new shown in figure 3, and an illustrative example is shown in Figure 4-7. The procedure traverse traverses PPS- tree in post order.

Consider the PPS-tree at time t_0. It consists solely of the root node. As seen in table 2, at time t_1 the sequences 01, 02,03,04,05 have elements AB, B, B, C, and B with corresponding profile information. Profile informationis extracted into eleProfile and elements are extracted into the eleSet in PPisa algorithm. The traverse algorithm inserts these elements and their corresponding profile information into the PPS tree. Since at time t_0 , the PPS consists of the root node, this is the only node which is traversed and processed. Lines 4 to 13 of traverse algorithm handle the processing of the root node and lines 14 to 30 deals with the processing of common node.

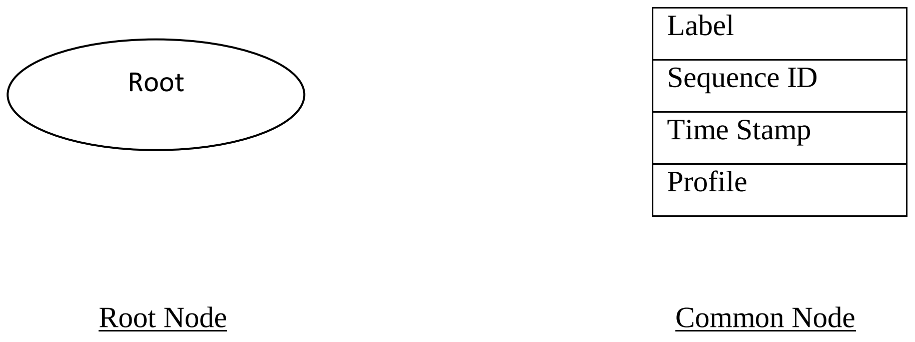

Fig.1. Data structure of Root node

For each element, the algorithm checks for all combinations of elements in the *eleSet*, i.e. for (AB) the procedure checks for A, B,(AB) to be the child of the root. While processing common nodes, every sequence in the sequence list of the node below it is searched to find any new data on that sequence. For example, at time t_2 for node labeled B (PPS tree at time t_1) the algorithm searches the sequence list to find for any new elements arriving on sequences 01, 02,03,04,05. Since this appears in sequence 05, it is inserted below node B in PPS-Tree as node A. Again now B comes in t_3 in PPS-Tree formed at time t_2 .This is the exception condition, i.e. repetition of items or item sets, which is implemented in our approach and shown in the traverse algorithm.But while processing common nodes if every sequence in the sequence list of the node below it contains the same data as it is there in this timestamp then it is inserted into the same below label node with corresponding Sequence ID, timestamp and their profile. If the combination is the label of one of the child nodes, and the sequence number of the element matches with any of the sequences in the child node's sequence list, the timestamp corresponding the matching sequence is updated, along with the T_0. Before inserting any new elements at a timestamp, obsolete sequences are deleted. A node whose sequence list contains no sequences is pruned off the tree. At last the traversing algorithm put out the frequent sequential patterns on the basis of the profiles of the customers satisfying the min_sup.

```
Algorithm P-Pisa (support, POI)
{
var PPS;        //PPS-Tree
var currentTime;        //timestamp now
var eleProfile=read all profile-info of the cust in DB
while (there is still new transaction)
        eleSet=read all ele at currentTime;
        traverse (currentTime,PPS);
        currentTime++;
}
```

Fig. 2. Algorithm P-Pisa

Example 3:

Database used to show this example is the same as one given in Table 2.

Suppose min_sup is 0.4 and POI is 4 for this example.

SoSUPPORT_FREQUENT=0.4* NO_OF_DATA_SEQUENCE

= 0.4* 5

= 2

So inorder to get a frequent sequential pattern the support countof the item set should be greater than 2. PPS-tree at each timestamp t_1, t2, t3, t4 is given in figure 4 through 7 respectively. At time t0 there is the only Root node. At time t1 the item sets from sequence 01, 02,03,04,05 are added to the PPS-tree below the Rootnode having a field as a label. Transaction_id, timestamp and their corresponding profiles. This step is repeated until there is a transaction at the new timestamp. PPS-tree at each time stamp t_1, t2, t3, t4 is given in figure 4-7 respectively.Following are the frequent patterns being mined by applying ordinary algorithm (on progressive Db without profile information) and our P-Pisa algorithm (on progressive Db with profile information).

In Figure 4, Ordinary Algorithm:B

P-Pisa: M G=>B

F K=>B //all other combination null

In Figure 5, Ordinary Algorithm: A, BC, C

P-Pisa: MG=>BC, C //all other combination null

In Figure 6, Ordinary Algorithm: AB, BA, BCB, BB, CB, and (AB)

P-Pisa: MG=>BCB, BB,CB //all other combination null

In Figure 7, Ordinary Algorithm :AB,ABE,AE,BA,BCBE,BCE,BBE,BE,CB,CA,CE,(AB),E

P-Pisa: MG=>BCB,BB,CB ,

FG=>; MK=>; FK=>BC

```
1.  Procedure traverse(current Time,PPS)
2.  {
3.  for(each node of PPS in post order)
4.  if(node is Root)
5.  for(ele of every seq in eleSet)
6.  for(all combination of element in the ele)
7.  if(element==label of one of node.child)
8.  if(seq is in node.child.seq_list)
9.  update timestamp of seq to currentTime;
10. else
11. create a new sequence with currenttime and profile;
12. else      //create a child
13. create a new child with element,seq,currentTime and profile;
14. else //the node is a common node
15. for(every seq in the seq_list)
16. if(seq.timestamp<=currentTime-POI)
17. delete seq from seq_list and continue to the next seq;
18. if(there is new ele of the seq in eleSet)
19. for(all combinationof elements in the ele)
20. if(element==label of one of node.child)
21. if(seq is node.child.seq_list)
22. child.seq_list.seq.timestamp=seq.timestamp;
23. else
24. create a new sequence with seq.timestamp and profile
25. else      // create a child
26. create a new child with element,seq,profile and seq.timestamp;
27. if(seq_list. Size==0)
28. delete this node and all its children from its parent;
29. while(seq_list.eleProfile.match.size>=support*sequence number)
30. output the label of path from Root to this node as a SP along with its Profile
31. }
```

Hence the above results show that the ordinary sequential pattern algorithm gives BCBE as one of the frequent pattern even though it is not frequent in general if the profile is taken into consideration. In contrast to this our PPisa algorithm gives the interesting pattern as there is no need to invest on E even though BCB is purchased as mentioned above in example 2. Also in Pisa, the formation of the sequential pattern as BCBE is not possible but is possible in our P-Pisa.

4. Conclusion

An algorithm is proposed to mine sequential patterns from a transaction database that contains the profile of the customers and their purchases of in a sequence of visits to a shop. Extracted patterns reveal new knowledge as we allow the presence of repeated items.

The conjunction with the profiles of the customers results in a compact set of true patterns useful for business decisions. Contribution in this paper improves the quality of the patterns and reduces the irrelevant patterns.

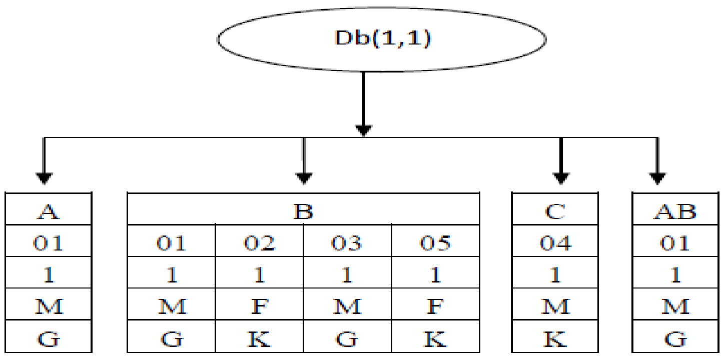

Fig. 4.Tree structure for timestamp t1

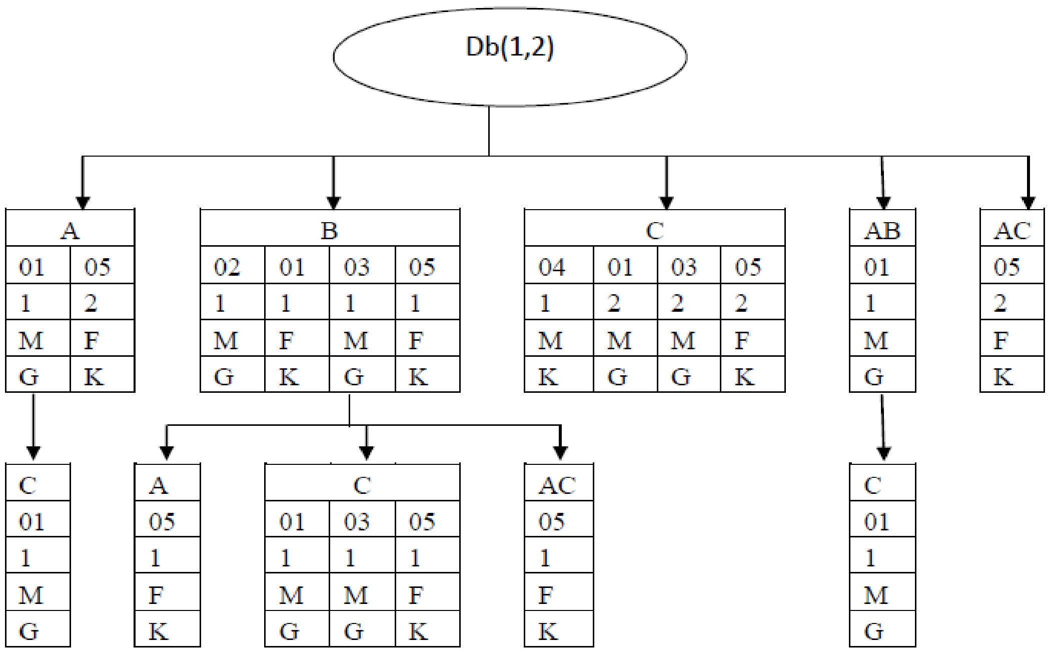
Db(1,2)
A 01 05 1 2 M F G K
B 02 01 03 05 1 1 1 1 M F M F G K G K
C 04 01 03 05 1 2 2 2 M M M F K G G K
AB 01 1 M G
AC 05 2 F K
C 01 1 M G
A 05 1 F K
C 01 03 05 1 1 1 M M F G G K
AC 05 1 F K
C 01 1 M G

Fig. 5. Tree structure for timestamp t2

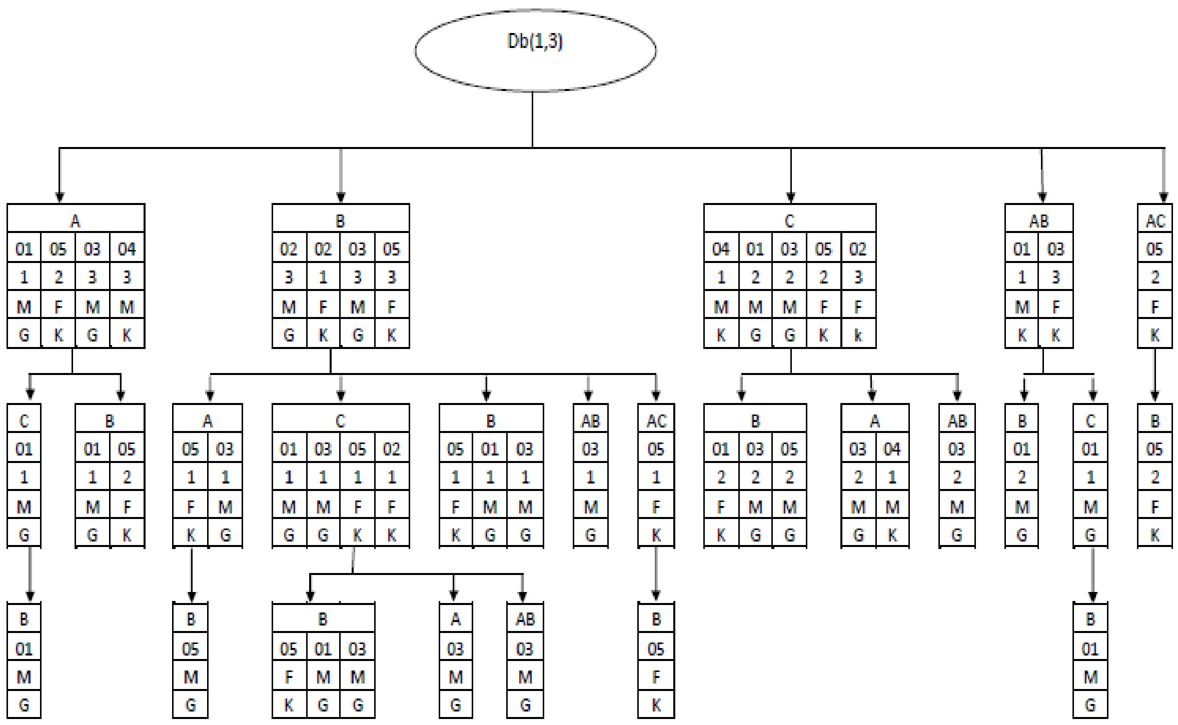

Fig. 6.Tree structure for timestamp t3

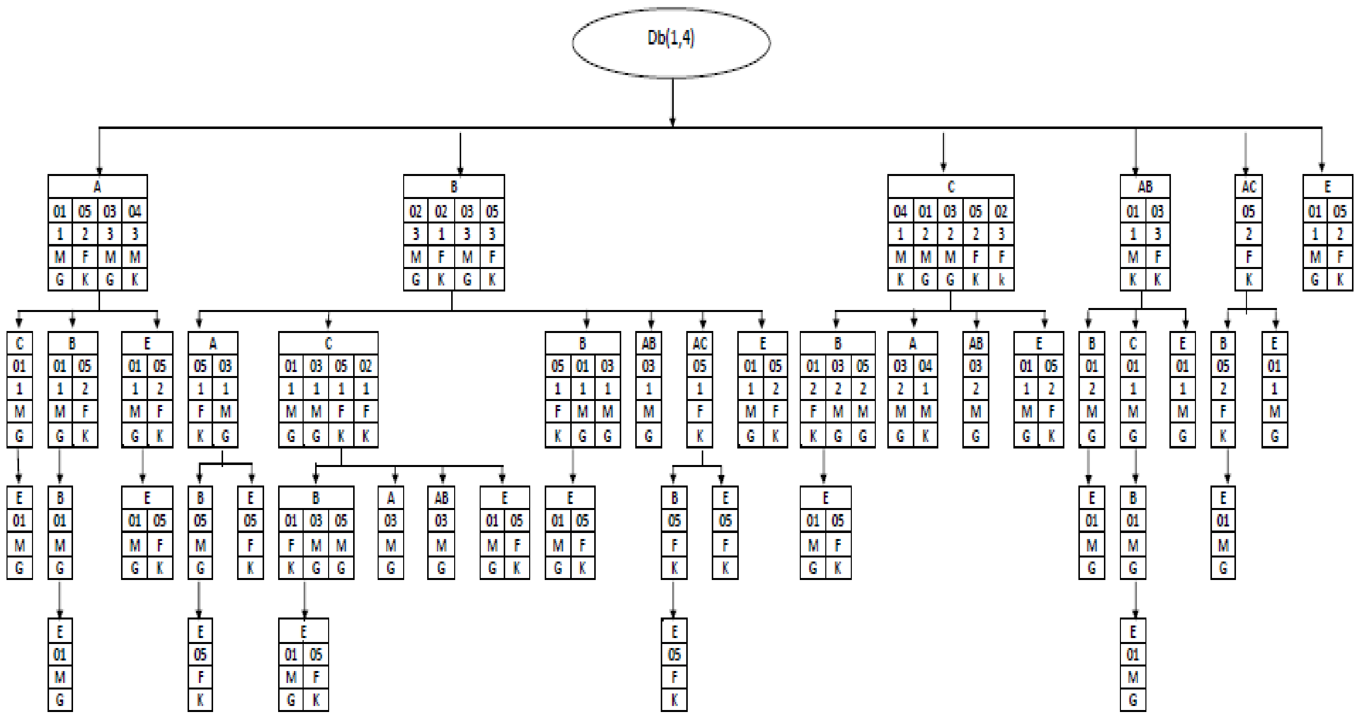

Fig. 7.Tree structure for timestamp t4

References

1. Rakesh Agarwal and RamaKrishna Srikant. “Mining Sequential Patterns,” Proc. 11th Int’l Conf. Data Engg.(ICDE ’95), pp 3-14, Feb 1995.
2. Ramakrishnan Srikant and Rakesh Agarwal “Mining Sequential Patterns: Generalizations and performance improvements,” Proc. Fifth Int’I conf. Extending Database Technology (EDBT’96), March 1996.
3. Mohammed J. Zaki SPADE “An Efficient Algorithm or Mining Frequent Sequences, “Machine Learning ACM’01 pp-31-60 vol. 42 issue no.1-2, 2001.
4. Ayres J. Gehrke, T.Yiu, and J. Flannick”Sequential pattern Mining Using a Bitmap Representation”, Proc. Eight ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (KDD’ 02),pp-429-435, July2002.
5. J. Han, J.Pei, B.Mortazavi-Asl, Q.Chen, U. Dayal, M.C. Hsu FREE-SPAN: “Frequent Pattern-Projected Sequential Pattern Mining” Proc. Sixth ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (KDD ’00), pp 355-359, 2000.
6. J.Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, M.C. Hsu PREFIX-SPAN:” Mining Sequential Patterns Efficiently ByPrefix-Projected Pattern Growth,” Proc. 17th Int’l Conf. Data Engg. (ICDE’01)2001
7. S. Parthatrathy, M.J.Zaki, M.Ogihara, S. Dwarkadas “Incrementaland Iterative Sequence Mining.” Proc. Eight ACM Int’l Conf. Information and Knowledge management (CIKM ’99), pp. 251-258, 1999
8. J.W. Huang, C.Y. Tseng, J.C.Ou, M.S. Chen “ A General Model For Sequential Pattern Mining With A Progressive Database, “ IEEE Transactions on Knowledge and Data Engineering, Vol. 20. Issue no. 9, September 2008.
9. Q. Yang and H.H. Zhang “Web-Log Mining for Predictive Web Caching,” IEEE Transactions on Knowledge and Data Engineering, Vol.15, issue no. 4, pp-1050-1053, July/Aug 2003.
10. Fast algorithms for online generation of profile association rules: CC Aggarwal, Zheng Sun, Yu.P.S.
11. Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen and Umeshwar Dayal “Multi-dimensional Sequential Pattern Mining,” Proc. Tenth Int’l Conf. on Information and Knowledge Management.(ACM’01) pp.81-88, 2001.