Counting Inference Approach to Discover Calendar based Temporal Association Rules

Anjana Pandey, Niket Bhargava and K.R.Pardasani

***Abstract*-- A temporal association rule is an association rule that holds during specific time interval. An example is that bread and butter are frequently sold together in morning hours. In real world data the knowledge used for mining rule is almost time varying. Most of the popular temporal association rule mining methods are having performance bottleneck for database with different characteristics. Method like Temporal-Apriori takes more time to generate frequent patterns in Association rule mining. This problem is mainly related to the number of operations required for counting pattern supports in the database. To remedy this, we propose an algorithm Temporal-PASCAL (TPASCAL) to discover calendar –based temporal association Rules.**

The TPASCAL uses Pattern counting inference that minimize as much as possible the number of pattern support counts performed when extracting frequent patterns. It is based on the notion of key patterns, where a key pattern is minimal pattern of equivalence class gathering all patterns common to the same objects of the database relation. With pattern counting inference, only the supports of the frequent key patterns (and some infrequent ones) are determined from the database, while supports of the frequent non key pattern are derived from those of the frequent key patterns. To restrict the time based assocationship calendar based pattern can be used. A calendar unit such as months and days, clock unit, such as hours and seconds & specialized units, such as business days and academic years, play a major role in a wide range of information system applications . This algorithm is based on the level wise extraction of frequent patterns. The performance is evaluated by comparing the efficiency of TPASCAL with the Temporal - Apriori algorithm.

Index Terms--Counting Inference, Key-Pattern, Support, Calender Schema, Apriori, Temporal Apriori, Frequent Pattern, Temporal Assocation Rule.

I. INTRODUCTION

The problem of the discovery of association rules (AR) comes from the need to discover patterns in transaction data in a supermarket. But transaction data are temporal. For example, when gathering data about products purchased in a supermarket, the time of the purchase is registered in the transaction. This is called transaction time, in temporal database jargon, which matches the valid time, corresponding to the time of the business transaction. The notion of association rule was proposed to capture the co occurrence of items in transaction [1]. Association Rule (AR) is the rule that expresses associative relationship between two item sets and denoted by X => Y (Bread => Jelly). Two important concepts in AR are support and confidence.

Definition 1.1. AR X => Y has support s %, if at least s % of transactions in the database (DB) contain X and Y.

Definition 1.2. AR X => Y has confidence c %, if at least c % of transactions in DB that contain X also contain Y.

Example1.1. A grocery store retailer is trying to decide whether to put bread on sale. To help determine the impact of this decision, the retailer generates association rules that show what other products are frequently purchased with bread. He finds that 60% of the time that bread is sold so is pretzels and that 70% of the time jelly is also sold, that may be denoted:

Bread => Pretzels (with confidence = 0.6) and Bread=> Jelly (with confidence = 0.7).

Based on these facts, he tries to capitalize on the association between bread, pretzels and jelly by placing some pretzels and jelly at the end of the aisle where the bread is placed. In addition, he decides not to place either of these items on sale at the same time

An interesting extension to association rules is to include a temporal dimension. For example, if we look at a database of transactions in a supermarket, we may find that sweets and crackers are seldom sold together. However, if we only look at the transactions in the week before diwali, we may discover that most transactions contain sweets and crackers, i.e., the association rule "sweets->*crackers*" has a high support and a high confidence in the transactions that happen in the week before diwali. The above suggests that we may discover different association rules if different time intervals are considered. Informally, we refer to the association rules along with their temporal intervals as temporal association rules. In this paper, we use *calendar schema[2]* as frameworks to discover temporal association rules. A hierarchy of calendar units determines a calendar schema. For example, a calendar schema can be (*year, month*, *day*). A calendar schema defines a set of *simple calendar-based patterns* (or *calendar patterns* for short). For example, given the above calendar schema, we will have calendar patterns such as *every day of January of*

This work was supported in part by the RGTU University and Bansal Group Bhaopal.es:

Niket Bhargava is with BIST College, District Bhopal, State MP, India (e-mail: nikresearch@rediffmail.com).

Anjana Pandey is with RGTU University, District Bhopal, State MP, India (e-mail: apeeshukla@yahoo.com).

Kamal Raj Pardasani is with MANIT, District Bhopal, State MP, India (e-mail: kamalrajp@hotmail.com).

1999 and *every 16th day of January of every year*. Basically, a calendar pattern is formed for a calendar schema by fixing some of the calendar units to specific numbers while leaving other units "free" (so it's read as "every"). It is clear that each calendar pattern defines a set of time intervals. We assume that the transactions are time stamped so we can decide if a transaction happens during a specific time interval. Given a set of transactions and a calendar schema, our first interest is to discover all pairs of association rule and calendar pattern such that for each pair (r; e), the association rule r satisfies the minimum support and confidence constraint among all the transactions that happen during each time interval given by the calendar pattern e. For example, we may have an association rule sweets-> crackers along with the calendar pattern *every day in every November*. We call the resulting rules *temporal association rules w.r.t. Precise match[2]*. In some applications, the above temporal association rules may be too restrictive. Instead, we may require that the association rule hold during "enough" number of intervals given by the corresponding calendar pattern. For example, the association rule sweets-> crackers may not hold on every day of every November, but holds on more than 80% of November days. We call such rules *temporal association rules w.r.t. Fuzzy match[2]*.

The rest of the paper is organized as follows. The next section defines related work on discovery of association rules, temporal data mining in general and discovery of temporal rules in temporal database. In particular is given in section 2.Section 3 defines temporal association rule in terms of calendar patterns. The TPASCAL algorithm is described in section 4 and example is described in section 5.A summary of the paper and some perspectives are given in section 6.

II. Related Work

The concept of association rule was introduced as Apriori algorithm [3]. Three approaches have been proposed for mining frequent patterns: the first is traversing iteratively the set of all patterns in a levelwise manner [4]. The most prominent algorithm based on this approach is the Apriori algorithm [5], that uses identical property as the OCD algorithm [6] proposed currently. A variety of modification of this algorithm arose [7] in order to improve different efficiency aspects. However, all of these algorithms have to determine the supports of all frequent patterns and some infrequent ones from the database. The second approach is based on the extraction of maximal frequent patterns, from which all supersets are infrequent and all subsets are frequent.

The most prominent algorithm using this approach is Max-Miner [8]. The third approach, represented by the Close algorithm [9],is based on the theoretical framework introduced in [9] that uses the closure of the Galois connection[10].In paper [11] the omission of the time dimension in association rule was very clearly mentioned. A temporal aspect of association rule was given by Juan [12]. According to this transaction in the database are time stamped and time interval is specified by the user to divide the data into disjoint segments, like month, days and years. Further the cyclic association rule was introduced by ozden [13] with minimum support and high confidence. A nice bibliographic of temporal data mining can be found in the Roddick literature [14]. Rainsford & Roddick presented extension to association rules to accommodate temporal semantics. It can be used in point based and interval based model of time simultaneously [14].

III. Problem Formulation

A. .Key patterns and pattern Counting Inference

Definition 1. Let P be a finite set of items, O a finite set of objects (e.g., transaction ids) and $R \subseteq O \times P$ a binary relation (where (o, p) ε R may be read as "item p is included in transaction o").The triple D=(O, P, R) is called dataset [15].

Each subset p of P is called a pattern. We say that a Pattern P is included in an object o ε O if (o,p) ε R for all p ε P[].Let f be the function which assign to each pattern $p \subseteq P$ the set of all objects which include this pattern :f(P) = { o ε O| o includes P }. The support of a pattern P is given by supp (P) = card (f (P))/card (O). For a given threshold minsup ε[0,1], a pattern P is called frequent pattern if supp(P)$\geq$ minsup.

Definition 2.A K- pattern p is a subset of P with card (P) = K .A candidate K-pattern where all its proper sub-patterns are frequent. [15]

Definition 3.For patterns P, $Q \subseteq P$ let PθQ if and only if f (P) =f (Q) .the set of patterns which are equivalent to a pattern P is given by [P] = {$Q \subseteq P$ | Pθ Q}[15]

Definition 4 A pattern P is a key pattern if P ε min [P]. A candidate key pattern is a pattern where all its proper sub patterns are frequent key patterns.

Theorem 1(i) if Q is a key pattern and $P \subseteq Q$,then P is also a key pattern.(ii) if P is not a key pattern and ,$P \subseteq Q$ then Q is not a key pattern either.[15]

Theorem 2.Let P be a pattern.(i) Let pε P .then P ε [P \{p}] if and only if supp(P)= supp(P\ {p})

(ii) P is a key pattern if and only if supp(P) $\neq$ min $_{p\varepsilon P}$ (supp(P\{p})) [15]

Theorem 3 Let P is a non key pattern then supp(P)= min $_{p\varepsilon P}$ (supp(P\{p})) [15]

B. Simple Calendar-based Pattern

When temporal information is applied in terms of date, month, year & week form the term Calendar schema. it is introduced in temporal data mining. A calendar schema is a relational Schema R =(Gn : Dn,G_{n-1} : $D_{n-1,.....,}$ G1 : D1) together with a valid constraint[2]. Each attribute Gi is a granularity name like year, month and week. Each domain Di is a finite subset of the Positive integers. A calendar schema (year: {2007,2006….}, month: 1,2,3,4…12}, day{1,2,3…..31}) with the constraints is valid if that evaluates(yy,mm,dd) to true only If the combination gives a

valid date while <1996,2,31> is not .In calendar pattern, the branch e cover e' in the same Calendar schema if the time interval e' is the subset of e and they all follow the same pattern. If a calendar patterns $<d_n,d_{n-1},\ldots d_1>$ covers another pattern $<d'_n,d'_{n-1},\ldots d'_1>$ if and only if for each I,1<=I<=n or $d_i = d'_1$.Now our task is to mine frequent pattern over arbitrary time interval in terms of calendar pattern schema.

C. Temporal Association rule

Definition 1: The frequency of itemset over a time period T is the number of transactions in which it occurs divided by total number of transaction over a time period. In the same way, confidence of an item with another item is the transaction of both items over the period divided by first item of that period [9]. Support (A)= Frequency of occurrence of A in specified time interval / total no of tuples in Specified time interval.

Confidence (A=>B [Ts, Te]) = Support count (A ∪B) over interval /occurrence of A in interval. T_s indicates the valid start time & T_e indicate valid time according to temporal data

IV. Our Method

Mining temporal association rules can be decomposed into two steps: (1) finding all Frequent item sets for all star calendar patterns on the given calendar schema, and (2) generating Temporal association rules using the frequent item sets and their calendar pattern. The first step is the crux of the discovery of temporal association rule; in the following, we will focus on this problem.

The pseudo-code is given in algorithm 1.we apply the theorems given in the last section into an algorithm. A list of notations is provided in table 1.we assume that P is linearly ordered e.g , P = {1,…..,n}.This will be used in TPASCAL – GEN.The algorithm starts with the empty set, which always has a support of 1 and which is (by definition) a key pattern (step 1 and 2). The algorithm work in passes. In each pass, the basic time intervals in the calendar schema are processed one by one. During the processing of basic time interval e_0 in pass k, the set of frequent K-itemset P_k (e_0) is first computed, and then P_k (e_0) is used to update the large k itemsets for all the calendar patterns that cover e_0 .In first pass (step 4)we compute the frequent 1-itemsets for each basic time interval. They are marked as key patterns unless their support is 1(steps 5-7).The main loop is similar to the one in Apriori.(step 10….) .In the subsequent passes, we divide the processing of each basic time interval into three phases. Phase 1,TPASCAL-GEN is called to compute the candidate patterns for the basic time interval. Phase II reads the transaction whose timestamps are covered by the basic time interval, updates the support of the candidate frequent Itemsets, and discover frequent itemsets for this basic time interval. The support of key ones is determined via a database pass (steps 15-20).Then (steps 21-27) the 'traditional pruning (step 22) is done. At the same time, for all remaining candidate key patterns, It is determined whether they are key or not(step 23 and 24). Now let us explain phase III. After the basic time interval e_0 is processed in pass k ,the large K-itemset for all the calendar patterns that cover e_0 are updated as follows. for precise match ,this is done by intersecting the set $P_k(e_0)$ of frequent k itemset for the calendar pattern e(i.e., $P_k(e) = P_k(e) \cap P_k(e_0)$). (Certainly, $P_k(e) = P_k(e_0)$ when $P_k(e)$ is updated for the first time.).it is easy to see that after all the basic time interval are processed, the set of frequent k-itemset for each calendar pattern consist of the k-itemset that are frequent for all basic time intervals covered by the pattern. The way TPASCAL-GEN operates is basically known from the generator function Apriori-Gen which was introduced in [15].when called at the Kth iteration ,it uses as input the set of frequent (K-1) patterns P_{K-1}. its output is the set of candidate K –patterns. in addition to Apriori-Gen's join and prune steps, TPASCAL-Gen makes the new candidate inherit the fact of being or not a candidate key pattern (step 9) by using Theorem 2; and it determines at the same time the support of all non key candidate patterns (step 12) by using theorem 4

TABLE 1
NOTATION USED IN TPASCAL

K	Is the counter which indicates the current iteration. In the Kth iteration all frequent K-patterns and all key patterns among them are determined.
P_k	Contains after the Kth iteration all frequent k patterns P together with their support P.supp, and a boolean variable P.key indicating if P is a (candidate) key pattern
C_k	Stores the candidate k patterns together with their support (if Known),the Boolean variable P.key and a counter P.pred_supp which stores the minimum of the supports of all (k-1) sub patterns of P

The proposed algorithm is as follows,

A. Algorithm TPASCAL:

1) ϕ.supp ← 1 ; ϕ.Key ← true ;
2) $P_0 \leftarrow \{\varnothing\}$
3) For all basic time interval e_0 do begin
4) $P_1(e_0) \leftarrow$ { frequent 1 pattern in $T[e_0]$}
5) For all $p \in P_1(e_0)$ do begin
6) p.pred_suppp ← 1 ;p.Key ← (p.supp ≠1) ;
7) end;

for all star pattern e that cover e_0 do

8) update P_1 (e) using $P(e_0)$;
9) end
10) for (k=2 ; ∃ a star calendar pattern e such that $P_{k-1}(e) \neq \Phi$,K++) do begin
11) for all basic time interval e_0 do begin
// Phase I: generate candidates

12) $C_k(e_0) \leftarrow$ TPASCAL-Gen($P_{k-1}(e_0)$)
// Phase II: Scan the transactions
13) For all transaction $T \in T[e_0]$ do
14) If $\exists C \in C_k(e_0)$ | C.key then
15) For all $o \in D$ do begin
16) $C_0 \leftarrow$ Subset ($C_k(e_0)$,O, T) // C.count ++ if
// $c \in C_k(e_0)$ is contained in T
17) for all $C \in C_0$ | C.Key do
18) c.supp ++
19) end ;
20) for all $c \in C_k(e_0)$ do
21) if c.supp ≥ minsup then begin
22) if c.key and c.supp = c.pred_supp then
23) c.key ← False ;
24) $P_k(e_0) \leftarrow P_k(e_0) \cup \{c\}$
25) End
// Phase III : update for star calendar patterns

26) for all star pattern e that cover e_0 do
27) update P_k (e) using $P_k(e_0)$
28) end
29) Output <P_k (e), e> for all star calendar pattern e
30) End

B. Algorithm 2 TPASCAL-GEN

Input : $P_{k-1}(e_0)$, the set of frequent (K-1) patterns p with their support p.supp and the p.key flag.

Output: $C_k(e_0)$,the set of candidate k patterns c each with the flag c.key,the value c.pred_supp,and the support c.supp if c is not a key pattern

1) insert into $C_k(e_0)$ select $p.item_1$,$p.item_2$,……$p.item_{k-1}$,$q.item_{k-1}$
from P_{k-1} p, P_{k-1} q
Where $p.item_1 = q.item_1$, ……. $p.item_{k-2} = q.item_{k-2}$, $p.item_{k-1} < q.item_{k-1}$;
2) for all c ε $C_k(e_0)$ do begin
3) c.key ←true ; c.pred_supp ← +∞ ;
4) for all (k-1) subsets s of c do begin
5) if $s \notin P_{k-1}(e_0)$ then
6) delete c from $C_k(e_0)$;
7) else begin
8) c.pred_supp ← min(c.pred_supp,s.supp);
9) if not s.key then c.key ←false ;
10) end;
11) end;
12) if not c.key then c.supp ←c.pred_supp;
13) end;
14) return $C_k(e_0)$

V. THE RUNNING EXAMPLE

We illustrate the TPASCAL algorithm on the following dataset for minsup=2/5, and threshold temporal support=3 on Month attribute of the calendar supporting transaction Database. The sample dataset is see Table 2,

TABLE 2
SAMPLE DATASET

Timestamp(Month)	TransactionId	Items
1	Txnnb200719	A,C,D,F
2	Txnnb200720	B,C,E,F
3	Txnnb200721	A,B,C,E,F
4	Txnnb200722	B,E,F
5	Txnnb200723	A,B,C,E,F

The algorithm performs first one database pass to count the support of the 1-itemset pattern (key and nonkey both), than checked for lifespan temporal support. The candidate pattern {D} is pruned because it is infrequent. Also {F} is marked as nonkey because it has the same support as empty set: Now P1 is see Table 3,

TABLE 3
FREQUENT (KEY & NON-KEY) 1-ITEMSET WITH SUPPORT AND LIFESPAN

P1	supp	Key	LifeSpan
A	3/5	T	[1,5]
B	4/5	T	[2,5]
C	4/5	T	[1,5]
E	4/5	T	[2,5]
F	1	F	[1,5]

At the next iteration, all candidate 2-itemset patterns are created and stored in C2, key and nonkey elements are categorized and lifespan is evaluated. at the same time the support of pattern containing infrequent pattern {F} as sub-pattern is computed. Then a database pass is performed to determine the support of the remaining six candidate patterns and then the life span temporal support is validated against temporal support threshold hence P2 achieved: Now C2 is see Table 4

TABLE 4
CANDIDATE (KEY & NON-KEY) 2-ITEMSET WITH SUPPORT AND LIFESPAN

C2	pred_supp	Key	supp	LifeSpan
AB	3/5	T	?	[2,5]
AC	3/5	T	?	[1,5]
AE	4/5	T	?	[2,5]
AF	3/5	F	3/5	[1,5]
BC	4/5	T	?	[2,5]
BE	4/5	T	?	[2,5]
BF	4/5	F	?	[2,5]
CE	4/5	T	?	[2,5]
CF	4/5	F	?	[1,5]
EF	4/5	F	?	[2,5]

From this we have Now P2 is see Table 5,

TABLE 5
FREQUENT (KEY & NON-KEY) 2-ITEMSET WITH SUPPORT AND LFESPAN

P2	supp	LifeSpan	Key
AB	2/5	[2,5]	T
AC	3/5	[1,5]	F
AE	2/5	[2,5]	T
AF	3/5	[1,5]	F
BC	3/5	[2,5]	T
BE	4/5	[2,5]	F
BF	4/5	[2,5]	F
CE	3/5	[2,5]	T
CF	4/5	[1,5]	F
EF	4/5	[2,5]	F

Repeating again we have Now C3 is see Table 6,

TABLE 6
CANDIDATE (KEY & NON-KEY) 3-ITEMSET WITH SUPPORT AND LIFESPAN

C3	pred_supp	Key	supp	LifeSpan
ABF	2/5	F	2/5	[2,5]
ABC	2/5	F	2/5	[2,5]
ABE	2/5	F	2/5	[2,5]
ACE	2/5	F	2/5	[2,5]
ACF	3/5	F	3/5	[1,5]
AEF	2/5	F	2/5	[2,5]
BCE	3/5	F	3/5	[2,5]
BCF	3/5	F	3/5	[2,5]
BEF	4/5	F	4/5	[2,5]
CEF	3/5	F	3/5	[2,5]

And Now P3 is evaluated as see Table 7,

TABLE 7
FREQUENT (KEY & NON-KEY) 3-ITEMSET WITH SUPPORT AND LIFESPAN

P3	Supp	Key	LifeSpan
ABF	2/5	F	[2,5]
ABC	2/5	F	[2,5]
ABE	2/5	F	[2,5]
ACE	2/5	F	[2,5]
ACF	3/5	F	[1,5]
AEF	2/5	F	[2,5]
BCE	3/5	F	[2,5]
BCF	3/5	F	[2,5]
BEF	4/5	F	[2,5]
CEF	3/5	F	[2,5]

So now in 4th and 5th iteration no database scan is required as we have no key candidate to calculate support in database. Now C4 is see Table 8,

TABLE 8
CANDIDATE (KEY & NON-KEY) 4-ITEMSET WITH SUPPORT AND LIFESPAN

C4	pred_supp	Key	supp	LifeSpan
ABCE	2/5	F	2/5	[2,5]
ABCF	2/5	F	2/5	[2,5]
ABEF	2/5	F	2/5	[2,5]
ACEF	2/5	F	2/5	[2,5]
BCEF	3/5	F	3/5	[2,5]

Now P4 is see Table 9,

TABLE 9
FREQUENT (KEY & NON-KEY) 4-ITEMSET WITH SUPPORT AND LIFESPAN

P4	supp	Key	LifeSpan
ABCE	2/5	F	[2,5]
ABCF	2/5	F	[2,5]
ABEF	2/5	F	[2,5]
ACEF	2/5	F	[2,5]
BCEF	3/5	F	[2,5]

We have Now C5 is see Table 10,

TABLE 10
CANDIDATE (KEY & NON-KEY) 5-ITEMSET WITH SUPPORT AND LIFESPAN

C5	pred_supp	Key	supp	LifeSpan
ABCEF	2/5	F	2/5	[2,5]

Now P5 is see Table 11,

TABLE 11
FREQUENT (KEY & NON-KEY) 5-ITEMSET WITH SUPPORT AND LIFESPAN

P5	pred_supp	Key	supp	LifeSpan
ABCEF	2/5	F	2/5	[2,5]

In next iteration algorithm generate no new 6-itemset candidate pattern so it stops here. Hence TPASCAL needs two database passes in which the algorithm counted the support of 6+6=12 patterns. Temporal-Apriori would have needed five database passes for counting supports of 6+6+10+10+5+1=32 patterns for the same dataset.

VI. CONCLUSION

We presented a new algorithm, called TPASCAL,for efficiently extracting frequent pattern in large database with time constraints such as calendar pattern .We are comparing the performance of our method with that of the "Temporal Apriori" algorithm proposed in [9] by running it manually. We will evaluate and compare the performance of this combination of algorithm on several four synthetic datasets, T20I6D100K, T25I10D10K and T25I20D100K that mimic market basket data. Next we are planning to develop an algorithm that can also incorporate confidence.

VII. REFERENCE

[1]]M.H.Dunham. Data Mining: Introductory and Advanced Topics. Prentice Hall, 2003.

[2] Y Li, P Ning, XS Wang, S Jajodia " Discovering calendar- based temporal association rule "Data & Knowledge Engineering, 2003 - Elsevier

[3] Jian Pei ,jiawei Han,Yiwen Yin and Running Mao R :Mining frequent pattern without Candidate Generation Kluwer online Academy 2004

[4] Banu ozden ,Sridhar Ramaswamy ,Avi sliberschatz R:"cyclic assocation rule " In Proc. Of forteenth International conference on Data Engineering 1998 pp 412-425

[5] Claudio Bettini ,Xsean Wang R " Time Granularies in database,data mining and temporal reasoning 2000 pp 230 ISBN 3-540-66997-3 Springer –verlang july 2000 230 pages Monograph

[6] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994.

[7] Jiawai Han,Micheline Kamber, Book:"Data mining Concept & techniques " 2001
[8] J **Pei**, J Han, H Lu, S Nishio, S Tang, D Yang "H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases "Proc. of the 2001 IEEE ICDM Conf
[9] JM Ale, GH Rossi "AN APPROACH TO DISCOVERING TEMPORAL ASSOCIATION RULES Proceedings of the 2000 ACM symposium on Applied computing- 2000
[10] Ramaswamy, S Mahajan, A Silberschatz" On the Discovery of Interesting Patterns in Association Rules" Proceedings of the 24rd International Conference on Very …, 1998
[11] CP Rainsford, JF Roddick" Adding Temporal Semantics to Association Rules**"** Proceedings of the 3rd European Conference on Principles of – Springer
[12] C Giannella, J Han, J Pei, X Yan, PS Yu" mining frequent pattern in data stream at multiple time granularities" Next Generation Data Mining, 2003
[13] Banu ozden ,Sridhar Ramaswamy ,Avi sliberschatz R:"cyclic assocation rule " In Proc. Of forteenth International conference on Data Engineering 1998 pp 412-425
[14] JF Roddick, K Hornsby, M Spiliopoulou" An Updated Bibliography of Temporal, Spatial, and Spatio-temporal Data Mining Research" First …, 2001 - books.google.com
[15] Bastide, R Taouil, N Pasquier, G Stumme, "Levelwise Search of Frequent Patterns with Counting Inference "ACM SIGKDD Explorations Newsletter, 2000

VIII. BIOGRAPHIES

Niket Bhargava, born in Bohani, state is MP in the India, on July 6, 1978. He graduated from the Oriental Institute of Science And Technology, Bhopal, and completed his Master in Technology from the RGTU- the technical university of MP, India. MP stands for Madhya Pradesh. His employment experience included the 4 year as Teacher in Top Most Ranking Institutions of state of MP, India. His special fields of interest included analysis and application of algorithms and Intelligent Expert System for Medical Treatment. Presently working as Assistant Professor in Department of CSE at Bansal Institute of Science and Technology, Bhopal. He is also in-charge of MTech in CSE.Bhaargava presented 3 papers in national and international conferences.

Anjana Pandey, born on Dec'18, 1978. She completed her Master in Computer Application. Her special fields of interest included Datamining. Presently working in Department of CSE at UIT-RGTU, Bhopal. Presently she is persuing PhD. In CSE from MANIT, Bhopal.

K..R.Pardasani was born on 13th September 1960 at Mathure, India. He completed his graduation, post graduation and PhD (mathematics) from Jiwaji university gwaliar India. his employment experience includes Jiwaji university Gwaliar MDI, Gurgaon and MANIT Bhopal India. Presently he is professor & Head of mathematics at MANIT, Bhopal and his current interest are data mining and computational biology.