

A
Dissertation
On

CUSTOMER RETENTION ANALYSIS

Submitted in Partial Fulfilment of the Requirement
for the Award of the Degree of

Master of Technology

In

Software Engineering

by

Neha

University Roll No. 2K12/SWE/15

Under the Esteemed Guidance of

Dr. Kapil Sharma

Associate Professor, Computer Engineering Department, DTU



2013-2014

**COMPUTER ENGINEERING DEPARTMENT
DELHI TECHNOLOGICAL UNIVERSITY
DELHI – 110042**

ABSTRACT

Mining frequent patterns in transaction databases, time-series databases, and many other kinds of databases has been studied popularly in data mining research. In our research work we have used FP-Tree based approach for mining single-level frequent patterns. We proposed a novel frequent-pattern tree (FP-tree) structure, which is an extended prefix-tree structure for storing compressed, crucial information about frequent patterns, and develop an efficient FP-tree based mining method, FP-growth, for mining the complete set of frequent patterns by pattern fragment growth. Methodology for Mining Multilevel frequent patterns is also used. Multilevel pattern carries more specific and concrete information than the single level.

We have used Graph based approach for extracting Multilevel frequent patterns. At each level it scans the datasets once and creates a directed graph, which is stored in form of an adjacency matrix and calculates all frequent patterns at the same level. Suppose database items are coded at three levels than this approach will need only three database scans. It does not require costly candidate generation method for creating new candidates. Another advantage of this approach is for less correlated databases it takes small memory space for storing graph at each level.

Keywords: Association Rules, Frequent Pattern Mining, Customer Churn, Apriori Algorithm, Retention, Frequent pattern Tree.

ACKNOWLEDGEMENT

First of all, I would like to express my deep sense of respect and gratitude to my thesis supervisor Dr. Kapil Sharma for providing the opportunity of carrying out this thesis and being the guiding force behind this research work. I am deeply indebted to him for the support, advice and encouragement he provided without which the thesis could not have been a success.

Secondly, I am grateful to Dr. O.P. Verma, HOD, Computer Engineering Department, DTU for his immense support. I would also like to acknowledge Delhi Technological University library and staff for providing the right academic resources and environment for this research work to be carried out.

Last but not the least I would like to express sincere gratitude to my parents and friends for constantly encouraging me during the completion of research work.

Neha

**University Roll no: 2K12/SWE/15
M.Tech (Software Engineering)
Department of Computer Engineering
Delhi Technological University
Delhi – 110042**

CERTIFICATE

This is to certify that the thesis titled “Customer Retention Analysis” is a bonafide record of work done by Neha, Roll No. 2K12/SWE/15 at Delhi Technological University for partial fulfilment of the requirements for the degree of Master of Technology in Software Engineering (Department of computer science). This thesis was carried out under my supervision and has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma to the best of my knowledge and belief.

Date: _ _ _ _

(Dr. Kapil Sharma)
Associate Professor & Project Guide
Department of Computer Engineering
Delhi Technological University

Declaration

I hereby declare that the thesis entitled “Customer Retention Analysis” which is being submitted to the Delhi Technological University, in partial fulfillment of the requirements for the award of degree of Master of Technology in Computer Science Engineering is an authentic work carried out by me. The material contained in this thesis has not been submitted to any university or institution for the award of any degree.

Neha

**University Roll no: 2K12/SWE/15
M.Tech (Software Engineering)
Department of Computer Engineering
Delhi Technological University
Delhi – 110042**

Contents

Abstract	ii
Acknowledgment	iii
Certificate	iv
Declaration	v
List of Figures	viii
List of Tables	ix
Abbreviations	x
Chapter One: Introduction.....	1
1.1 GENERAL	1
1.2 ROLE OF DATA MINING	2
1.3 MOTIVATION OF WORK.....	4
1.4 AIM OF WORK.....	4
1.5 ORGANISATION OF THESIS.....	6
Chapter Two: LITERATURE SURVEY	8
Chapter Three: FP-Tree Approach	16
3.1 OVERVIEW.....	16
3.2 FREQUENT-PATTERN TREE: DESIGN AND CONSTRUCTION.....	17
3.3 FP-GROWTH ALGORITHM	21
Chapter Four: Graph Based Approach.....	27
4.1 INTRODUCTION	27

4.2 CONCEPT OF MULTILEVEL PATTERN	27
4.3 ENCODED TRANSACTION TABLE	29
4.4 GRAPH BASED ALGORITHM	30
4.5 WORKING OF GRAPH BASED APPROACH	31
Chapter Five: Implementation Details and Results	38
5.1 RESULTS	39
5.1.1 Sample Dataset	39
5.2 OUTPUT	40
5.2.1 Screenshots	40
5.3 GRAPH BASED APPROACH	44
Chapter Six: Conclusion and Future scope	49
REFERENCES	50

LIST OF FIGURES

Figure 1 Market Basket Analysis	5
Figure 2 FP-tree for Table 1	20
Figure 3 FP- Growth Approach.....	23
Figure 4 Concept hierarchy (taxonomy).....	28
Figure 5 Directed Graph G for transaction Table 5	32
Figure 6Adjacency Matrix of Directed Graph G	32
Figure 7 FP Tree Generation	40
Figure 8 Prefix Subtree	42
Figure 9 Generation of Association Rules.....	43
Figure 10 Multilevel Pattern Mining	44
Figure 11 Filtered dataset	45
Figure 12 Filtered Dataset	46
Figure 13 Frequent Itemsets	47

LIST OF TABLES

Table 1:A transactional database as an example	18
Table 2 :Conditional FP-Tree	25
Table 3: Frequent pattern Generation	25
Table 4: Transactional Dataset D	29
Table 5 : Encoded transaction table T[1].....	30
Table:6 Level -1, minsup=4	33
Table 7 Filtered transaction table T [2]	33
Table 8 LEVEL-2 minsup=3	35
Table 9 Level-3 minsup = 3.....	36

Abbreviations

CET -	Closed Enumeration Tree
WSW-	Weighted Sliding Window
TID-	Transaction Identity
DFS-	Depth First Search
RELIM-	Recursive Elimination
CDS-	Compact Data Set
KDD-	knowledge Discovery in Database
MINSUP-	Minimum Support
DHP-	Direct Hashing and Pruning
FP-	Frequent Pattern
DB-	Database
MFI-	Most Frequent Itemset

Chapter One: Introduction

1.1 General

In area of data mining, the problem of deriving associations from data was first formulated by (Agrawal, R., Imieliński, T., & Swami, A. 1993), and is often referred to as the “market-basket” problem. In this famous problem, we are given a set of items (also called attributes) and a large collection of transactions which are treated as subsets (baskets) of these items. Association rule mining is about analysing and presenting strong rules discovered in databases using different measures of interestingness. Based on the concept of strong rules, association rules thus introduced, are used for discovering regularities between products in large scale transaction data. One such example is that of a supermarket. The items are products and the baskets are customer purchase entries. By finding associations between various products bought by customers, planning and marketing of products can be done better. (Kotsiantis, S., & Kanellopoulos, D. 2006) Many other applications with varied characteristics can also be used. For example, word counting in text documents, patient medical report and diagnosis and many more. Association rules (Agrawal, R., Imieliński, T., & Swami, A. 1993, June) are defined as statements of the form $\{X_1, X_2, \dots, X_n\} \rightarrow Y$ which means that Y may present in the transaction if X_1, X_2, \dots, X_n are all in the transaction. Note, that there can be a set of items, not just a single item. The probability of finding Y in a transaction with all X_1, X_2, \dots, X_n is called confidence. (Liming, W., & Hui, Z. 2007). The threshold (percentage) that a rule holds in all transactions is called support.

Support

The *support* (s) of an association rule is the ratio (in percent) of the records that contain X to the total number of records in the database. For e.g. if we say that the support of a rule is 5% then it means that 5% of the total records contain X .

Confidence

Confidence (\acute{a}) is the ratio (in percent) of the number of records that contain X & Y to the number of records that contain X . For e.g. if we say that a rule has a

confidence of 85%, it means that 85% of the records containing X also contain Y . The confidence of a rule indicates the degree of correlation in the dataset between X and Y (Han, J., Pei, J., Yin, Y., & Mao, R. 2004).

Today time series data analysis (Shih, M. J., Liu, D. R., & Hsu, M. L. 2010). is fundamental to engineering, scientific and business endeavours. Data is collected and analysed for frequent itemsets. For example, collecting data of various customer purchases from a supermarket at regular intervals of time might help us identify change in customer purchasing pattern. This gives us more realistic view of the system. Finding interest of the customer in purchasing helps planning and marketing of products in a better way.

1.2 Role of Data Mining

As database size has enlarged rapidly in recent years, this has led to development of tools, due to increasing interest which is capable of automatic withdrawal of knowledge from data (Papageorgiou, E. I. 2011).. In database the term knowledge discovery or data mining has been taken for a research field which deals with spontaneous discovery of knowledge or implicit information arising from the databases.

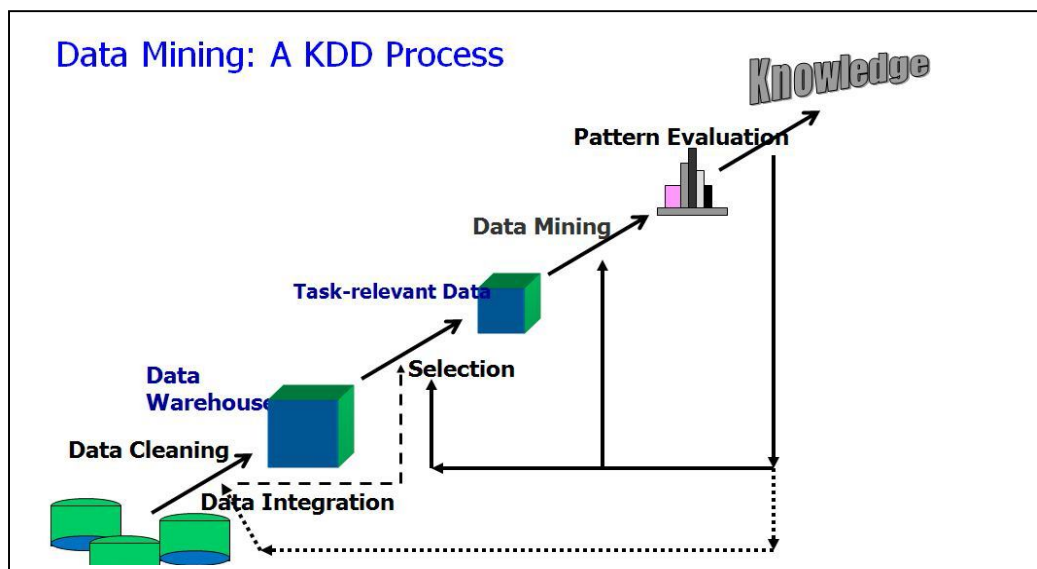


Figure 1 Data Mining Process

Databases which provides implicit information among different sets of objects, comes up with fascinating relationships that causes association rules and thus

discloses useful patterns for many applications like financial forecast, medical diagnosis, decision support and marketing policies etc. (Colantonio, A., Di Pietro, R., Ocello, A., & Verde, N. V. 2012). The Figure 1 depicts data mining process.

The term data mining is the process of identifying interesting patterns and knowledge from large amount of data (Slimani, T., & Lazzez, A. 2014). The data sources can include databases, data warehouses, the Web, other information repositories or data that are streamed into the system dynamically.

As IT technologies are getting advance, (Pyun, G., Yun, U., & Ryu, K. H. 2014) the number of data accumulated is also enhancing very frequently. Hence the role of data mining comes into the picture. The first algorithm proposed in this approach was Apriori Algorithm (Agrawal, R., & Srikant, R. 1994, September) With the time, several algorithms came up during the past several years which includes Apriori (Agrawal, R., & Srikant, R. 1994, September), (Relim Borgelt, C. 2005, August), ECLAT (Borgelt, C. 2003, November) etc.

As the name says, recurrent (frequent) patterns are patterns that are occurring recurrently in data (Sethi, N., & Sharma, P. 2013). Various types of recurrent patterns are sequential patterns (also known as frequent subsequences), frequent itemsets and frequent substructures. A frequent itemset implies to itemsets that often occur together in data transactions – for instance bread and butter, which many customers often purchase together in grocery stores. (Nebot, V., & Berlanga, R. 2012).

Association rule learning is an advance method of research for identifying relations among large databases. The definition of association rule mining is implied as: Consider $K = \{k_1, k_2 \dots k_n\}$ is itemset of n binary attributes. Let $D = \{S_1, S_2 \dots S_n\}$ be a transactions set known as database. Every transaction is associated with an identifier, called TID and in K it has a subset of the items. The definition of association rule is of the form $P \Rightarrow Q$ where $P, Q \subseteq K$ and $P \cap Q = \theta$. Here, P itemsets is called antecedent (LHS or left-hand-side) and Q is called consequent (RHS or right-hand-side) of the rule.

$$\text{Support } P \Rightarrow Q = P(P \cup Q)$$

$$\text{Confidence } P \Rightarrow Q = P(P | Q) = \text{Support}(P \cup Q) / \text{Support}(P)$$

Rules should satisfy minimum support and minimum confidence in order to determine recurrent itemsets (Han, J., & Kamber, M. 2006).

Recurrent pattern mining is a two step process:

- Find all frequent itemsets i.e. each of these item sets will occur at least as frequently as a predetermined minimum support count.
- Initiate strong association rule from frequent itemsets i.e. these rules must satisfy minimum support and minimum confidence.

1.3 Motivation of work

Motivation behind this work is to understand the dynamics of customer purchasing behavior by examining the recurring patterns that have taken place in transactions. This would not only help to cut down the customer churn rate but will also provide better customer satisfaction and good relationship. Identifying the interest of customer is essential for any service based business that has an ongoing relationship with their customers, that is of monetary value. So during an ongoing contract, we need to make sure that customers are happy with the service and they keep paying us. Otherwise loss of business, loss of revenue are not far away, when today's ongoing competition in the market in every field is considered.

1.4 Aim of work

The aim of my work is to study the customer needs, his interests, not only bounded to a single level but to detailed levels. That is the purchasing behavior of the customer is pruned to various levels of abstraction. For achieving this task we have used two algorithms. The first is the FP- Tree algorithm and the second one is a graph based approach for mining the interest of the customer to multiple levels of abstraction. For example, besides finding 80% of customer that purchase computer may also purchase printer, it is interesting to allow to users to "drill-down" and so that 75% of people buy color printer if they buy desktop computer. The latter statement carries more specific and concrete information than the former. In other words, buying habits of customer are analyzed by discovering the association between different products which are often put together by the customer in shopping baskets. Consider another example as shown in Figure 1, if customers tend to buy beer then

how likely they will also buy chips (and what type of chips) at the same time from the market?

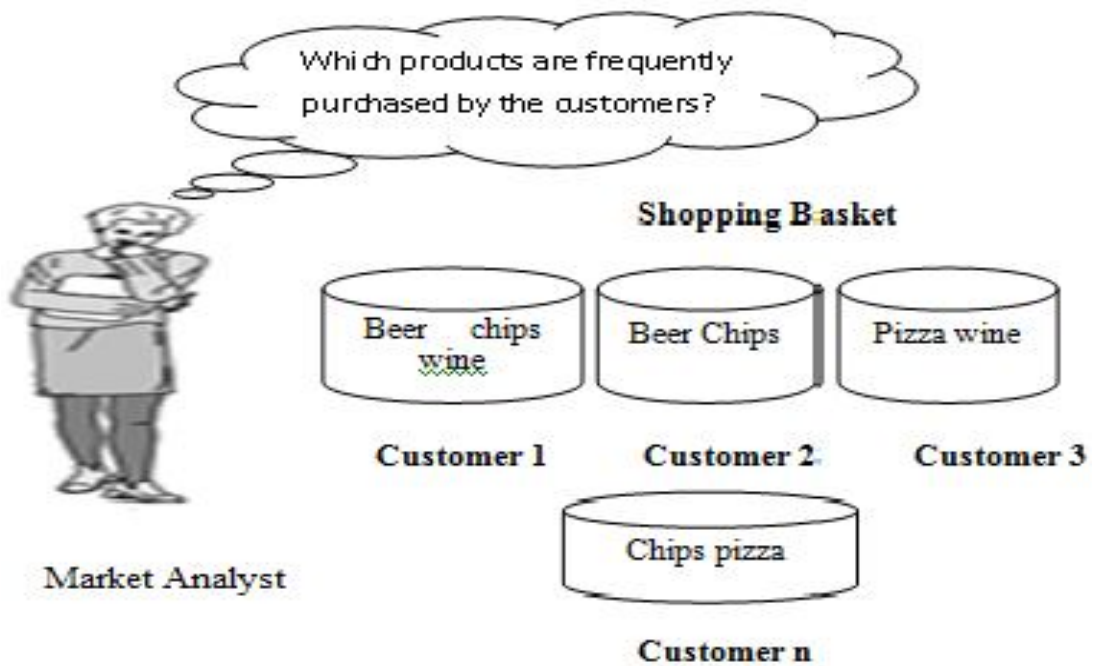


Figure 1 Market Basket Analysis

Therefore, a data mining system should provide efficient method for mining multiple-level interesting patterns.

1.5 Organisation of Thesis

The thesis is divided into 6 sections including Introduction.

The next chapter, Chapter-2 presents related literature available for our work. It covers literature related to association rule mining as well as discovery of frequent itemsets from large databases.

Chapter 3 presents FP-Tree and FP-Growth approach followed by an example to obtain results. The approach in solving the problem has been presented here.

Chapter-4 gives the Graph Based Approach for mining patterns at multiple level of abstraction.

Chapter 5 gives Implementation details and results obtained. The final chapter summarizes work done in the thesis and suggests future work that may be done in this area.

Chapter 2
LITERATURE SURVEY

Chapter Two: LITERATURE SURVEY

Normally, the process to analyze data from distinct point of view and summarize it into knowledgeable information i.e. information which can be used to increase income, reduce costs, or perform both is data mining (referred as knowledge discovery). Data mining, extracting hidden pattern or information from huge databases, is a dominant advance technology having great power in helping companies to focus on most significant information within data warehouses. From several different sources and angles or dimensions it allows many users to analyze data and then categorize it, and then relationships are identified. Professionally, the way of searching patterns or correlations among various fields in huge relational databases is data mining. Data mining perform prospective, automated analyses move which is more than examining past events provided by retroactive tools typical of decision support systems. The term data mining is the tool that answers business questions that takes too much time to resolve. Data mining scour the databases for finding hidden, predictive information that is beyond the expectation of experts.

Many research algorithms have been studied in this research work. The various algorithms that have been studied are Apriori (Agrawal, R., & Srikant, R. 1994, September), FP Growth (Han,J, Pei, J., Yin, Y., & Mao, R. 2004)., Compacting Data Sets (Han,J, Pei, J., Yin, Y., & Mao, R.2004)., Relim- Recursive Elimination (Borgelt, C. 2005)Direct Hashing and Pruning –DHP Park, (J. S., Chen, M. S., & 1995)., Multilevel Association Rule Mining Algorithm Based on Boolean Matrix (Chen, J., Lin, G., & Yang, Z.) etc. Brief discussion has been done on these algorithms.

Apriori: It is a seminal algorithm for recurrent itemset mining and learning association rule over transactional databases. This algorithm is suggested by(Agrawal, R., & Srikant, R. 1994, September). Apriori is based on the closure property which indicates that if an itemset is recurrent then all its subsets will also be recurrent. It operates on transaction databases where itemset refers to transaction. For a specified transaction threshold ζ ., itemsets which are subsets of at least ζ . transaction in the database are identified by the Apriori Algorithm. It is based on bottom up approach.

To count the candidate item sets in an efficient manner, Apriori make use of breadth first search technique and also hash tree structure. It follows the two steps:

- firstly, DB is scanned once in order to get frequent 1-itemset,
- From length k frequent itemsets generate length (k+1) itemset candidate itemsets, against DB test the candidate and finally when no frequent or candidate set generation is found terminate the process.

The applications of this algorithm are in Market Basket Analysis and potential user identification in telecom companies.

FP Growth: This algorithm is meant for discovering recurrent itemsets in a transactional database. proposed this algorithm. FP Growth is memory efficient and very fast algorithm. It uses FP-Tree which is a special internal structure. From the database it stores the transaction and each item is represented by linked list. FP tree consist of set of child nodes and root node and also frequent item header table. To find out the frequent pattern subsequently the node link structure and the insert-tree (P,N) subroutine is used. It makes use of horizontal and vertical database layout that stores the database in memory. (Han,J, Pei, J., Yin, Y., & Mao, R. 2004)

The principal objective of FP Growth was to eradicate the drawbacks of Apriori Algorithm in testing and generating the candidate set. Bottleneck of Apriori was handled with by introduction of FP tree which is a compact data structure (Han et al, 2004).

Compacting Data Sets (CDS): Apriori Algorithm is highly time consuming as it requires huge candidate generation, rigorous data scanning is performed and requires lot of input/output operations. To overcome these drawbacks CDS algorithm is used (Han,J, Pei, J., Yin, Y., & Mao, R. 2004). which is unique among all the algorithms as it not only eradicate redundant candidate generation but also eradicate transactions that are duplicate. It merges the duplicate transaction and intersection between the itemsets and then undesirable subsets are deleted repeatedly (Han,J, Pei, J., Yin, Y., & Mao, R. 2004).

CDS provides a new and efficient method for discovering frequent pattern; it compact data-set by removing items in infrequent 1-itemsets and duplicate transactions are merged repeatedly, and make use of the subset of transactions with

other transaction itemsets to perform pruning; along with the discovering process, with increasing number of deleted transactions, for calculating the subset the time needed will decrease rapidly. When data-set volume increases, its time & space cost drastically decrease, so its usability retains for MFI applications for high volume data-sets. In various aspects, CDS algorithm can be further optimized, such as keeping record of all resulting sub set to avoid duplicated generation .

Recursive Elimination (RELIM): The recursive elimination algorithm (Borgelt, C. 2005) is meant for searching frequent item sets. It has been adopted by FP-growth algorithm and resembles H-mine algorithm. Its main advantage is its simplicity of its structure. Without prefix tree it does its work. In singly linked lists, transactions are processed directly and then organized. The biggest benefit of this algorithm is simplicity of the data structures which are needed and re-representation of the transactions is not necessary and thus saves memory in the recursion. From transaction database with the frequencies are listed. Then, the reduced transactions (reducing by the user defined minimum support to transaction) database with items in the transaction database are sorted in ascending order with their frequencies. Then simple arrays of item identifiers represent each transaction. Then a recursive procedure is used to find out the frequent items.

Direct Hashing and Pruning (DHP): It is an improvement technique towards Apriori. (Park, J. S., Chen, M. S., & 1995). The Apriori algorithm can be improved using hashing technique to reduce the size of the candidate k-itemsets. The Hash table is based on the hash function $h(x, y) = ((orderofx) * 10 + (orderofy)) \% 7$. Then based on the hash value the items can be mapped to different buckets. Thus the candidate items are reduced. The performance is enhanced because DHP prunes both the number of transactions and number of items in each transaction during the iterations.

Multilevel Association Rule Mining Algorithm Based on Boolean Matrix: To find out the frequent itemsets Boolean matrix approach is used. The boolean matrix is of the form “0” and “1” and the “AND” operation is $0.0 = 0$, $1.0 = 0$, $1.1 = 1$, $0.1 = 0$. Now with minimum support matrix dimension is reduced. To generate 3 itemsets AND operation is performed. The algorithm (Chen, J., Lin, G., & Yang, Z. 2011, June). stops as soon as the maximum frequent itemset is found. Since

only once it scans the database and requires small memory for the operation, this is the biggest advantage (Chen, J., Lin, G., & Yang, Z. 2011, June)..

Frequent Pattern Algorithm using Dynamic Function - The entire database is scanned in this algorithm and transaction pairs are generated with longest common sequences and computes longest common sequences of item id for each previous transaction pair. Then the algorithm prunes the transaction pairs with empty longest common sequences. Using the dynamic function the longest common sequence is found. The support count is done for pruned subset patterns rather than the whole database. In the next operation again the pruned transaction pair with the least common sequences was observed. The advantage with this approach is that the database access is reduced and the subsequent iteration is faster than the previous iteration. (Jiawei, Hong, & Dong, 2007).

Partition –based Algorithm - Partition-based Algorithms (Aggarwal, C. C., Li, Y., Wang, J., & Wang, J. 2009, June) solves the problem of high number of database scans, associated with Apriori-based algorithm. It requires two complete data scan to mine frequent item sets. The Partition algorithm divides the dataset into many subsets so that each subset can be fitted into the main memory. The basic idea of the Partition-based algorithm is that a frequent item set must be frequent in at least one subset. Partition-based algorithm generates local frequent item sets for each partition during the first data scan. Since the whole partition can be fitted into the main memory, the complete local frequent item sets can be mined without any disk I/O operations. The local frequent item sets are added to the global candidate frequent item sets. In the second data scan, false candidates are removed from the global candidate frequent item sets. In a special case where each subset contains identical local frequent item sets, Partition algorithm can mine all frequent item sets with a single data scan. However, when the data is distributed unevenly across different partitions, this algorithm may generate a lot of false candidates from a small number of partitions. By employing the knowledge collected during the mining process, false global candidate frequent item sets are pruned when they are found that they cannot be frequent. In addition, those algorithms reduce the number of scans in the worse case to $(2b-1)/b$ where b is the number of partitions.

Moment Algorithm - It (Chi, Y., Wang, H., Yu, P. S., & Muntz, R. R. 2004), introduced one algorithm, Moment, to extract closed frequent itemset within sliding window. They designed, a prefix based data structure, CET (Closed Enumeration Tree), to maintain closed frequent itemset (Chi, Y., Wang, H., Yu, P. S., & Muntz, R. R. 2004). CET maintains the boundary between closed frequent itemset and rest of itemset, which makes the boundary relatively stable, whenever any itemset changes its state (frequent to non frequent vice-versa), ultimately reducing the updating cost. An efficient algorithm to incrementally update the CET, which update the CET when newly arrived transaction change the content of window or oldest transaction being deleted from the window. When a transaction arrives/expires, Moment traverse the part of CET related to that transaction.

The merit of Moment is that it computes the exact set of closed frequent itemset over a sliding window. Although an update to a node may result in a propagation of the node insertion and deletion in the CET, most of the nodes related to an incoming or expiring transaction do not change their type often. Therefore, the average update cost in the CET is small. Limitation of Moment algorithm is, it stores transaction using data structure, FP tree (Chi, Y., Wang, H., Yu, P. S., & Muntz, R. R. 2004), which require a considerable amount of memory. Secondly, if the size of window is too large, CET can huge.

WSW algorithm: (Tsai, P. S. 2009), proposed a new framework data stream mining, called weighted sliding window model, which allows user to specify the number of windows, weight of window and size of window. A single pass algorithm, called WSW, has been proposed to extract frequent itemset from data streams. The motivation for weighted sliding window model come from the fact, that the size of traditional sliding window model is fixed or defined by the number of transaction, say W . Though recent W transactions are considered, the time to cover these W transactions may be long or varies, which may effectively decrease the mining result. (Tsai, P. S. 2009) proposed weighted sliding window model that defines window size by time not by the number of transaction and user can specify the number of windows, with each window assign with different weight (sum of all window weight equals to 1). For example data may be more influential at current moment and hence,

should be assigned higher weight. The algorithm WSW scans the data once in each window, and calculates the support count for each item present in current window, to find out frequent k -itemset ($k=1$). Based on this information candidate $(k+1)$ -itemset are pointed out to find frequent $(k+1)$ -itemset. The process terminates when no further candidates itemset can be generated. If the number of windows increases the time to determine frequent itemset increases. To reduce this runtime they proposed an improvement of WSW algorithm called, WSW-imp, which further reduce the time of deciding whether a candidate itemset is frequent or non-frequent itemset.

DFS and Hybrid Algorithm – (Eclat and Clique, 2002) combine both depth first search (DFS) and intersection counting. Since intersection counting is used, no complicated data structure is required. These hybrid algorithms reduces the memory requirement, since only the TID sets of the path item sets from the root to the leaves have to be kept in the memory simultaneously. Intersection of TID sets can be stopped as soon as the remaining length of the shortest TID set is shorter than the required support minus the counted support value. The intersection of TID sets of 1-item set to generate frequent 2 item sets is expensive. The maximal hyper graph clique clustering is applied to 2-frequent item sets to generate a refined set of maximal item sets.(Garg, D., & Sharma, H. 2011). at DFS cannot prune candidate k item sets by checking frequent $(k - 1)$ item sets, because DFS searches from the root to the leaves of the tree without using any subsets relationship. It is cheaper to use item set counting with BFS to determine the supports, when the number of candidate frequent item sets is small. When the number of candidate frequent item sets is relatively large, the hybrid algorithm switches to TID set intersection with DFS, since simple TID set intersection is more efficient than occurrence counting when the number of candidate frequent item sets is relatively large. This results in additional costs to generate TID sets. The authors proposed to use hash-tree-like structure to minimize the cost of transition. However, the authors do not provide an algorithm to determine the best condition to switch the strategy. Authors provide parameters to change in strategy. However, those parameters may not be generalized enough for all kinds of datasets. Incorrect timing of changing strategy may decrease the performance of hybrid algorithm.

Eclat algorithm – (Borgelt, C. 2003, November). It follows depth-first search using set intersection and uses a vertical database layout i.e. instead of explicitly listing all transactions; each item is stored together with its cover (also called tidlist) and uses the intersection based approach to compute the support of an itemset. The support of an itemset X can be easily computed by simply intersecting the covers of any two subsets $Y, Z \subseteq X$, such that $Y \cup Z = X$. It states that, when the database is stored in the vertical layout, the support of a set can be counted much easier by simply intersecting the covers of two of its subsets that together give the set itself. In this algorithm each frequent item is added in the output set. Then for every frequent item i , the i -projected database D_i is created. This is done by first finding every item j that frequently occurs together with i . The support of set $\{i, j\}$ is computed by intersecting the covers of both items. If $\{i, j\}$ is frequent, then j is inserted into D_i together with its cover. The reordering is performed at every recursion step of the algorithm. Algorithm recursively calls to find all frequent itemsets in the new database D_i . Candidate itemsets are generated using only the join step from Apriori. Again all items in the database are reordered in ascending order of support to reduce the number of candidate itemsets that are generated, and hence, reduce the number of intersections that need to be computed and the total size of the covers of all generated itemsets. Since the algorithm doesn't fully exploit the monotonicity property, but generates a candidate itemset based on only two of its subsets, the number of candidate itemsets that are generated is much larger as compared to a breadth-first approach such as Apriori. As a comparison, Eclat essentially generates candidate itemsets using only the join step from Apriori, since the itemsets necessary for the prune step are not available.

CHAPTER 3
FP-TREE APPROACH

Chapter Three: FP-Tree Approach

3.1 Overview

Frequent-pattern mining plays an essential role in mining associations, correlations, sequential patterns, multi-dimensional patterns, max-patterns, emerging patterns, and many other important data mining tasks. Most of the previous studies adopt an Apriori-like approach, which is based on the anti-monotone Apriori heuristic. If any length k pattern is not frequent in the database, its length $(k + 1)$ super-pattern can never be frequent. The essential idea is to iteratively generate the set of candidate patterns of length $(k+1)$ from the set of frequent-patterns of length k (for $k \geq 1$), and check their corresponding occurrence frequencies in the database.

The Apriori heuristic achieves good performance gained by (possibly significantly) reducing the size of candidate sets. However, in situations with a large number of frequent patterns, long patterns, or quite low minimum support thresholds, an Apriori-like algorithm may suffer from the following two nontrivial costs:

- It is costly to handle a huge number of candidate sets. For example, if there are 104 frequent 1-itemsets, the Apriori algorithm will need to generate more than 107 length-2 candidates and accumulate and test their occurrence frequencies. Moreover, to discover a frequent pattern of size 100, such as $\{a_1 \dots a_{100}\}$, it must generate approximately 1030 candidates. This is the inherent cost of candidate generation, no matter what implementation technique is applied.
- It is tedious to repeatedly scan the database and check a large set of candidates by pattern matching, which is especially true for mining long patterns.

A method that may avoid candidate generation-and-test and utilize some novel data structures to reduce the cost in frequent-pattern mining is as follows:-

First, a novel, compact data structure, called frequent-pattern tree, or FP-tree in short, is constructed, which is extended prefix-tree structure storing crucial, quantitative information about frequent patterns. To ensure that the tree structure is compact and informative, only frequent length-1 items will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of node sharing than less frequently occurring ones. Our

experiments show that such a tree is compact, and it is sometimes orders of magnitude smaller than original database. Subsequent frequent-pattern mining will only need to work on the FP-tree instead of the whole data set.

Second, an FP-tree-based pattern-fragment growth mining method is developed, which starts from a frequent length-1 pattern (as an initial suffix pattern), examines only its conditional-pattern base (a “sub-database” which consists of the set of frequent items co-occurring with the suffix pattern), constructs its (conditional) FP-tree, and performs mining recursively with such a tree. The pattern growth is achieved via concatenation of the suffix pattern with the new ones generated from a conditional FP-tree. Since the frequent itemset in any transaction is always encoded in the corresponding path of the frequent-pattern trees, pattern growth ensures the completeness of the results. In this context, our method is not Apriori-like restricted generation-and-test but restricted test only. The major operations of mining are count accumulation and prefix path count adjustment, which are usually much less costly than candidate generation and pattern matching operations performed in most Apriori-like algorithms.

3.2 Frequent-pattern tree: Design and construction

Let $I = \{a_1, a_2, \dots, a_m\}$ be a set of items, and a transaction database, $DB = \{T_1, T_2, \dots, T_n\}$ where $T_i = (i \in [1, \dots, n]) \frac{n!}{r!(n-r)!}$ is a transaction which contains a set of items in I . The *support* (or occurrence frequency) of a *pattern* A , where A is a set of items, is the number of transactions containing A in DB . A pattern A is *frequent* if A 's support is no less than a predefined *minimum support threshold*, ξ . Given a transaction database DB and a minimum support threshold ξ , the problem of *finding the complete set of frequent patterns* is called the frequent-pattern mining problem. *Frequent-pattern tree*: To design a compact data structure for efficient frequent-pattern mining, let's first examine an example.

For e.g. let the transaction database, DB , be the first two columns of Table 1:A transactional database as an example, and the minimum support threshold be 3 (i.e., $\xi = 3$).

A compact data structure can be designed based on the following observations:

- Since only the frequent items will play a role in the frequent-pattern mining, it is necessary to perform one scan of transaction database *DB* to identify the set of frequent items (with *frequency count* obtained as a by-product).
- If the *set* of frequent items of each transaction can be stored in some compact structure, it may be possible to avoid repeatedly scanning the original transaction database.
- If multiple transactions share a set of frequent items, it may be possible to merge the shared sets with the number of occurrences registered as *count*. It is easy to check whether two sets are identical if the frequent items in all of the transactions are listed according to a fixed order.

Table 1:A transactional database as an example

TID	Items Bought	(Ordered)Frequent Items
100	<i>f,a,c,d,g,i,m,p</i>	<i>f,c,a,m,p</i>
200	<i>a,b,c,f,l,m,o</i>	<i>f,c,a,b,m</i>
300	<i>b,f,h,j,o</i>	<i>f,b</i>
400	<i>b,c,k,s,p</i>	<i>c,b,p</i>
500	<i>a,f,c,e,l,p,m,n</i>	<i>f,c,a,m,p</i>

If two transactions share a common prefix, according to some sorted order of frequent items, the shared parts can be merged using one prefix structure as long as the *count* is registered properly. If the frequent items are sorted in their *frequency descending order*, there are better chances that more prefix strings can be shared.

With the above observations, one may construct a frequent-pattern tree as follows.

First, a scan of *DB* derives a *list* of frequent items, $\{(f: 4), (c: 4), (a: 3), (b: 3), (m: 3), (p: 3)\}$ (the number after “:” indicates the support), in which items are ordered in frequency descending order. This ordering is important since each path of a tree

will follow this order. For convenience of later discussions, the frequent items in each transaction are listed in this ordering in the rightmost column of Table 1:A transactional database as an example

Second, the root of a tree is created and labeled with “*null*”. The FP-tree is constructed as follows by scanning the transaction database *DB* second time.

- The scan of the first transaction leads to the construction of the first branch of the tree:
- $\{(f: 1), (c: 1), (a: 1), (m: 1), (p: 1)\}$. Note that frequent items in transaction are listed according to the order in the *list* of frequent items.
- For second transaction, since its (ordered) frequent item list (f, c, a, b, m) shares a common prefix (f, c, a) with the existing path (f, c, a, m, p) , count of each node along the prefix is incremented by 1, and one new node $(b: 1)$ is created and linked as a child of $(a: 2)$ and another new node $(m: 1)$ is created and linked as the child of $(b:1)$.
- For third transaction, since its frequent item list (f, b) shares only the node (f) with the f -prefix subtree, f 's count is incremented by 1, and a new node $(b: 1)$ is created and linked as a child of $(f: 3)$.
- The scan of fourth transaction leads to construction of second branch of the tree, $((c: 1), (b: 1), (p: 1))$.
- For last transaction, since its frequent item list (f, c, a, m, p) is identical to the first one, path is shared with the count of each node along the path incremented by 1.

To facilitate tree traversal, an item header table is built in which each item points to its first occurrence in the tree via a node-link. Nodes with the same item-name are linked in sequence via such *node-links*. After scanning all transactions, the tree, together with associated node-links, are shown in Figure 2.

Based on this example, a *frequent-pattern tree* can be designed as follows.

- It consists of one root labeled as “*null*”, a set of item-prefix subtrees as children of the root, and a frequent-item-header table.
 - Each node in item-prefix subtree consists of three fields: *item-name*, *count*, and *node-link*, where *item-name* registers which item this node represents, *count* registers the number of transactions represented by the portion of path reaching this node, and *node-link* links to the next node in the FP-tree carrying same item-name, or null if there is none.

➤ Each entry in frequent-item-header table consists of two fields, (1) *item-name* and (2) *head of node-link* (a pointer pointing to the first node in the FP-tree carrying the *item-name*).

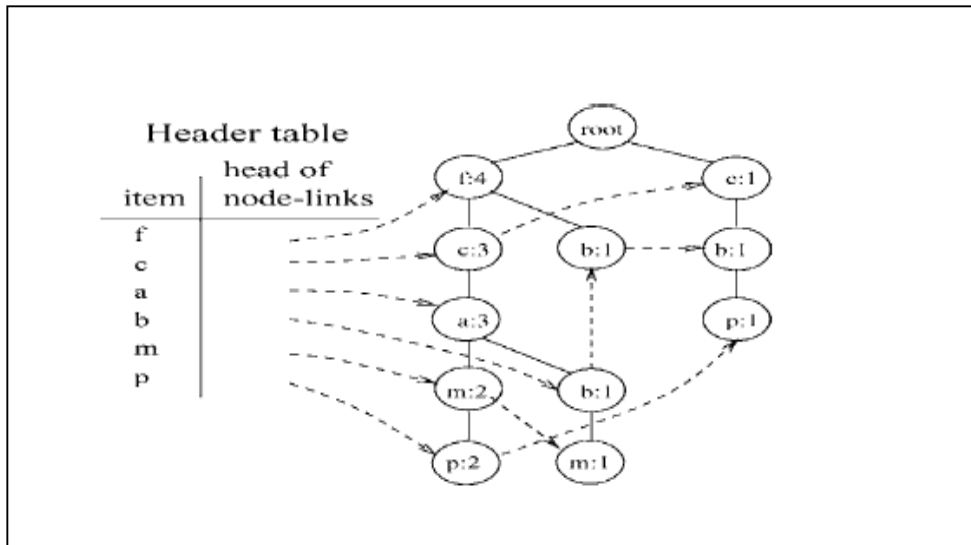


Figure 2 FP-tree for Table 1

Based on this definition, we have the following FP-tree construction algorithm.

Algorithm (FP-tree construction).

Input: A transaction database DB and a minimum support threshold ζ .

Output: FP-tree, the frequent-pattern tree of DB .

Method: The FP-tree is constructed as follows.

- Scan transaction database DB once. Collect F , sets of frequent items, and support of each frequent item. Sort F in support-descending order as $FList$, list of frequent items.
- Create the root of an FP-tree, T , and label it as “null”. For each transaction $Trans$ in DB do the following. Select frequent items in $Trans$ and sort them according to the order of $FList$. Let sorted frequent-item list in $Trans$ be $[p | P]$, where p is the first element and P is remaining list. Call $insert\ tree([p | P], T)$.

The function $insert\ tree([p | P], T)$ is performed if T has a child N such that $N.item-name = p.item-name$, then increment N 's count by 1; else create a new node N , with its count initialized to 1, its parent link linked to T , and its

node-link linked to nodes with the same *item-name* via node-link structure. If P is nonempty, call *insert tree*(P, N) recursively.

3.3 FP-Growth Algorithm

Construction of a compact FP-tree ensures that subsequent mining can be performed with a rather compact data structure. In this section, we study how to explore the compact information stored in an FP-tree.

Let us examine the mining process based on the constructed *FP-tree* shown in Figure 3.

According to list of frequent items, $f-c-a-b-m-p$, all frequent patterns in the database can be divided into 6 subsets without overlap:

- patterns containing item p ;
- patterns containing item m but no item p ;
- patterns containing item b but no m nor p ;
- patterns containing item a but no b, m nor p ;
- patterns containing item c but no a, b, m nor p ; and
- patterns containing item f but no c, a, b, m nor p .

Algorithm (*FP-growth*: Mining frequent patterns with *FP-tree* by pattern fragment growth).

Input: A database DB , represented by *FP-tree* constructed and a minimum support threshold ζ .

Output: The complete set of frequent patterns.

Method: call *FP-growth*(*FP-tree*, *null*).

Procedure *FP-growth*(*Tree*; a)

{

(1) if *Tree* contains a single prefix path // Mining single prefix-path *FP-tree*

(2) then {

(3) let P be the single prefix-path part of *Tree*;

(4) let Q be the multiple-path part with the top branching node replaced by a *null* root;

(5) for each combination (denoted as β) of the nodes in the path P do

```

(6) generate pattern  $\beta \cup \alpha$  with  $support = \text{minimum support of nodes in } \beta$ ;
(7) let  $freq\ pattern\ set(P)$  be the set of patterns so generated; }
(8) else let  $Q$  be  $Tree$ ;
(9) for each item  $a_i$  in  $Q$  do { // Mining multiple-path  $FP$ -tree
(10) generate pattern  $\beta = a_i \cup \alpha$  with  $support = a_i.support$  ;
(11) construct  $\beta$ 's conditional pattern-base and then  $\beta$ 's conditional  $FP$ -tree  $Tree_\beta$ ;
(12) if  $Tree_\beta \neq \emptyset$ ;
(13) then call  $FP\text{-}growth(Tree_\beta, \beta)$ ;
(14) let  $freq\ pattern\ set(Q)$  be the set of patterns so generated; }
(15) return( $freq\ pattern\ set(P) \cup freq\ pattern\ set(Q) \cup (freq\ pattern\ set(P) \times freq\ pattern\ set(Q)$ ))
}

```

Let us mine these subsets one by one:

- We first mine patterns having item p . An immediate frequent pattern in this subset is $(p: 3)$.
- To find other patterns having item p , we need to access all frequent item projections containing item p . All projections can be collected by starting at p 's node-link head and following its node-links.
- Following p 's node-links, we can find that p has two paths in the FP -tree: $(f: 4; c: 3; a: 3; m: 2; p: 2)$ and $(c: 1; b: 1; p: 1)$. The first path indicates that string “ $(f; c; a; m; p)$ ” appears twice in the database. Notice the path also indicates that string $(f; c; a)$ appears three times and (f) itself appears even four times. However, they only appear twice *together* with p . Thus, to study which string appear together with p , only p 's prefix path $(f: 2; c: 2; a: 2; m: 2)$ (or simply, $(fcam: 2)$) counts. Similarly, second path indicates string “ $(c; b; p)$ ” appears once in the set of transactions in DB , or p 's prefix path is $(cb: 1)$. These two prefix paths of p , “ $\{(fcam: 2), (cb: 1)\}$ ”, form p 's subpatternbase, which is called p 's *conditional pattern-base* (i.e., the subpattern-base under the condition of p 's existence). Construction of an FP -tree on this conditional pattern-base (which is called p 's *conditional FP-tree*) leads to only one branch $(c: 3)$ as shown in Table 2. Hence, only one frequent pattern $(cp: 3)$ is derived. (Notice that a pattern is an itemset and is denoted by a string here.) The search for frequent patterns associated with p terminates.

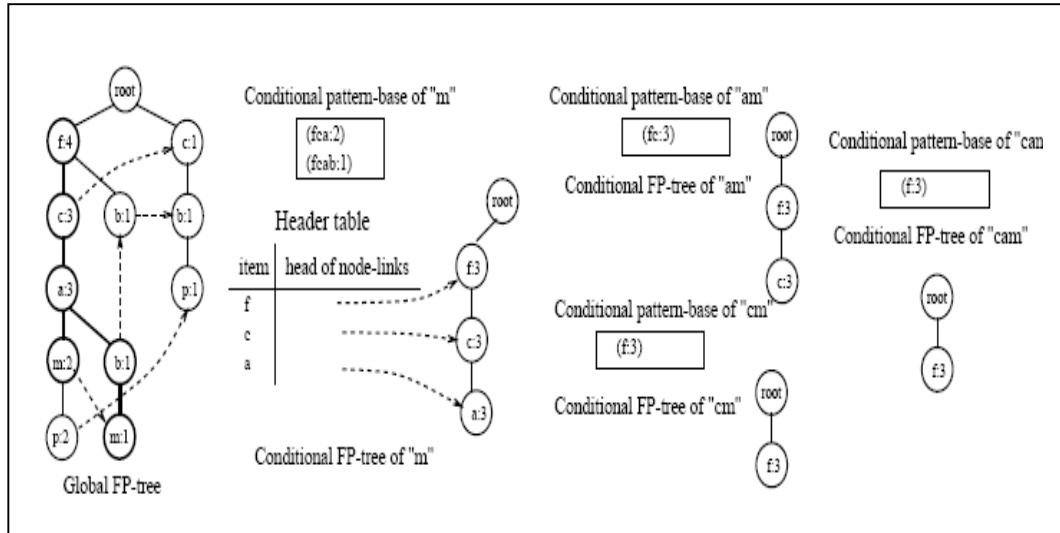


Figure 3 FP- Growth Approach

➤ Now, let us turn to patterns having item m but no item p. Immediately, we identify frequent pattern (m: 3). By following m's node-links, two paths in FP-tree, (f: 4; c: 3; a: 3;m: 2) and (f: 4, c: 3, a: 3, b: 1, m: 1) are found. Notice p appears together with m as well, however, there is no need to include p here in the analysis since any frequent patterns involving p has been analyzed in the previous examination of patterns having item p. Similar to the above analysis, m's conditional pattern-base is {(fca: 2), (fcab: 1)}. Constructing an FP-tree on it, we derive m's conditional FP-tree, (f: 3; c: 3; a: 3), a single frequent pattern path, as shown in above Figure 3. This conditional FP-tree is then mined recursively by calling mine ((f: 3; c: 3; a: 3)|m) as shown in Table 2.

➤ Figure 3 shows that "mine((f: 3; c: 3; a: 3)|m)" involves mining three items (a), (c), (f) in sequence. The first derives a frequent pattern (am: 3), a conditional pattern-base {(fc: 3)}, and then a call "mine((f: 3; c: 3)|am)"; second derives a frequent pattern (cm: 3), a conditional pattern-base {(f: 3)}, and then a call "mine((f: 3)|cm)"; and third derives only a frequent pattern (fm: 3). Further recursive call of "mine((f: 3; c: 3)|am)" derives (cam: 3), (fam: 3), a conditional pattern-base {(f: 3)}, and then a call "mine((f: 3)|cam)", which derives the longest pattern (fcam: 3). Similarly, call of "mine((f: 3)|cm)", derives one pattern (fcm: 3). Therefore, whole set of frequent patterns involving m is ((m: 3), (am: 3), (cm: 3), (fm: 3), (cam: 3), (fam: 3), (fcm: 3), (fcam: 3),

($fcm: 3$)). This indicates a single path FP-tree can be mined by outputting all the combinations of items in path.

- Similarly, we can mine patterns containing item b but no m nor p . Node b derives ($b:3$) and has three paths: ($f: 4; c: 3; a: 3; b: 1$), ($f: 4; b: 1$), and ($c: 1; b: 10$). Since b 's conditional pattern-base $\{(fca: 1), (f: 1), (c: 1)\}$ generates no frequent item, the mining for b terminates.
- For patterns having item a but no b, m nor p , node a derives one frequent pattern $\{(a: 3)\}$ and one subpattern base $\{(fc: 3)\}$, a single-path conditional *FP-tree*. Thus, its set of frequent patterns can be generated by taking their combinations. Concatenating them with ($a: 3$), we have $\{(fa: 3), (ca: 3), (fca: 3)\}$.
- Now, it is the turn to mine patterns having item c but no a, b, m nor p . Node c derives ($c: 4$) and one subpattern-base $\{(f: 3)\}$, and set of frequent patterns associated with ($c: 3$) is $\{(fc: 3)\}$.
- The last subset, i.e., pattern having item f but no any other items, is f itself and ($f: 4$) should be output. No conditional pattern-base need to be constructed as shown in last row of Table 2.

Table 2 :Conditional FP-Tree

Item	Conditional pattern-base	Conditional FP-tree
p	$\{(fcam: 2), (cb: 1)\}$	$\{(c: 3)\}p$
m	$\{(fca: 2), (fcab: 1)\}$	$\{(f: 3, c:3,a: 3)\}m$
b	$\{(fca: 1), (f: 1), (c: 1)\}$	$\{\}$
a	$\{(fc: 3)\}$	$\{(f: 3,c: 3)\}a$
c	$\{(f: 3)\}$	$\{(f: 3)\}c$
f	$\{\}$	$\{\}$

From above **Table 2** we generate following frequent patterns:

Table 3: Frequent pattern Generation

Items	Frequent Patterns Generated
P	$\{p c: 3\}$
M	$\{m f: 3, m c: 3, m a: 3, m f c: 3, m f a: 3, m c a: 3, m f c a: 3\}$
A	$\{a f: 3, a c: 3, a fc: 3\}$
C	$\{c f: 3\}$

Hence Frequent Patterns are successfully generated using FP-Tree based mining approach.

CHAPTER 4

Graph Based Approach

Chapter Four: **Graph Based Approach**

4.1 Introduction

The data mining is considered to be one of the promising field of Computer Science. Researches are based on its applications such as bioinformatics, web-usage mining, text and image recognition, remote sensing and biometric. Discovering association rules from database has been considered as one of difficult task. A key component of association rule mining is to find all frequent itemsets. The industries, which work on huge voluminous data needs to extract association relationship for improving their business and decision making aspect.

Most of the previous work has been focused on mining patterns at a single concept level. Many applications at multiple levels of abstractions require that mining be performed at multiple levels of abstraction. For example, besides finding 80% of customer that purchase computer may also purchase printer, it is interesting to allow to users to “drill-down” and so that 75% of people buy color printer if they buy desktop computer. The latter statement carries more specific and concrete information than the former. Therefore, a data mining system should provide efficient method for mining multiple-level interesting patterns.

4.2 Concept of Multilevel Pattern

The approach used for mining multilevel frequent pattern in our project is based on concept taxonomy or concept hierarchy. A concept hierarchy is defined as sequence of mapping from a set of low-level concept to higher-level, more general concept. For example we consider a concept hierarchy of computer items for the sales which is shown in Figure 5. It has four levels denoted with 0,1,2,3, level 0 for computer items i.e. most general abstraction level, level 1 includes Desktop computer, printer and other peripherals, level 2 includes Pentium Desktop computer, Celeron Desktop computer, Impact Printer etc. and level 3 includes HCL Pentium Desktop computer, WIPRO Impact Printer etc.

In concept hierarchy as shown in Figure 4 higher level shows more general information than the lower level i.e. lower level contains most specific information. In

Computer Items concept hierarchy Celeron desktop computer is more specific information than the Desktop computer, similarly HCL Celeron Desktop computer is more specific information than Celeron Desktop computer or Desktop Computer.

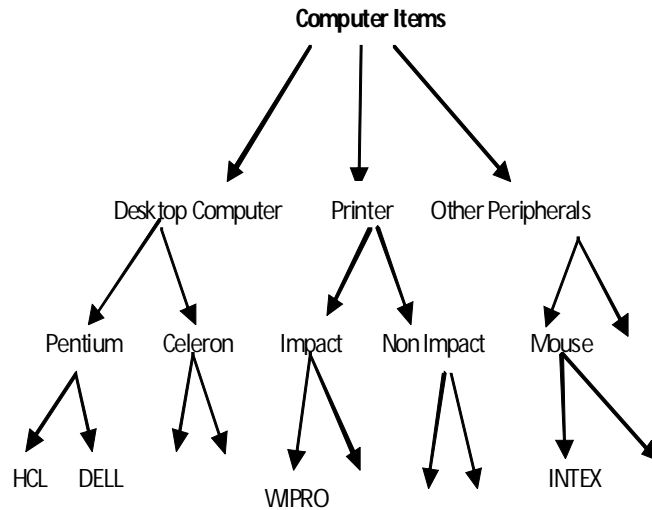


Figure 4 Concept hierarchy (taxonomy)

Based on the definition, the idea of mining multiple-level frequent patterns is illustrated below:

The taxonomy information is provided in Figure 5. Let category of items (such as “Printer”) represents the first-level concept, type (such as “Impact”) for the second level one, and brand (such as “WIPRO”) for the third level one. Using this taxonomy we can easily determine multilevel frequent patterns from transaction dataset. The frequent pattern mining process first discover frequent pattern at top-most level. Suppose frequent 1-itemsets are {Desktop computer}, {Printer}, similarly frequent 2-itemset is {Desktop computer, Printer} etc at level 1.

According to the definition of multilevel association rules, only the descendants of the frequent item at level-1 are considered as candidate in level-2 frequent 1-itemsets. Suppose {Other Peripherals} is a infrequent at level-1 than its descendants i.e. {mouse} etc will be not considered at level-2. Let the {Pentium Desktop computer},{Impact Printer},{Non Impact Printer}are frequent 1-itemsets, { Pentium Desktop computer , Impact Printer },{ Pentium Desktop computer, Non Impact Printer} are the frequent 2-itemsets etc.

Similarly at level-3 the frequent 1-itemsets are the {HCL Pentium Desktop computer}, {WIPRO Impact Printer}, {DELL Pentium Desktop computer} and frequent 2-itemsets are {HCL Pentium Desktop computer, WIPRO Impact Printer}, {DELL Pentium Desktop computer, WIPRO Impact Printer} etc.

The process mining frequent patterns at multiple concept level starts from top level and from next level, onwards the process repeats at even lower concept level until no large patterns can be found.

4.3 Encoded Transaction Table

Now suppose we have the original transactional datasets as shown in Table 4: Transactional Dataset D of Computer Items for the sales. Following table shows the items purchased for each transaction.

Table 4: Transactional Dataset D

TID	Items
T1	WIPRO Impact Printer, DELL Pentium Desktop Computer
T2	HP Non Impact Printer, Intel Mouse
T3	HCL Pentium Desktop Computer, HP Non Impact Printer
....,,.....,

Now we use an encoded transaction table instead of the original transaction table shown in Table 4: Transactional Dataset D .First we encode the original transaction table into encoded transaction.

with the help of concept taxonomy Figure 4.

As stated above, the taxonomy information for each (grouped) item in Figure 4 is encoded as a sequence of digits in the transaction Table 5 : Encoded transaction table T[1] For example, the item ‘HCL Pentium Desktop Computer’ is encoded as ‘311’ in which the digit, ‘3’, represents ‘Desktop Computer’ at level-1, the second digit, ‘1’, for ‘Pentium’ at level-2, and the third, ‘1’, for the company ‘HCL’ at level-3. Similar to [4], repeated items (i.e., items with the same encoding) at any level will be treated as one item in one transaction.

Table 5 : Encoded transaction table T[1]

TID	Items
T_1	{311, 321, 411, 421}
T_2	{311, 411, 422, 523}
T_3	{312, 322, 421, 611}
T_4	{311, 321}
T_5	{311, 322, 411, 421, 613}
T_6	{313, 523, 824}
T_7	{331, 431}
T_8	{523, 611, 713, 824}

4.4 Graph Based Algorithm

The unique feature of graph based algorithm is that it scans the entire database only once, during scanning the database it creates the directed graph, which is stored in memory in form of an Adjacency Matrix. The each vertex of graph represents to item of datasets and weight of each edge represent the occurrence of itemsets that means weight of edge between two vertices shows support of candidate 2-itemsets. The weight of vertex shows support of single itemsets (vertex). If any vertex has weight

less than the user define support then this vertex is deleted from the graph i.e. it is not participating for finding large frequent patterns. For mining large k-itemsets ($k \geq 3$) this algorithm uses relationship of vertices in a graph. This algorithm avoids the costly candidate generation process because large frequent patterns finding by logical *AND* operation between each row element of Adjacency matrix.

4.5 Working of Graph Based Approach

To explain the working of this approach, we use the encoded transaction Table 5 : Encoded transaction table T[1], which contains 8 transactions with itemsets $I = \{311, 312, 313, 321, 322, 331, 411, 421, 422, 431, 523, 611, 613, 713, 824\}$. Each item encoded into three digits. First higher most significant digit represents to a item at first level (higher level) i.e. 3**, 4** etc., first two higher most significant digits represents to a item at second level (subsequent lower level) i.e. 31*, 41* etc., similarly all three digits of a item represents to a item at lower level i.e. 311, 411 etc. Now here we will apply this graph based approach at the first level and suppose $min_sup = 4$ for this level.

In first step, it will scan the database i.e. transaction Table 5 : Encoded transaction table T[1] and create the directed graph G , which is shown in Figure 5. The items are stored in database in lexicographical order and thus a directed edge in between two vertices will be made in some spacial form. For example, if there is edge $A \rightarrow B$ i.e. in database transactions, occurrence of A will always precede B and if A and B are numeric number then, always, $A < B$. The edge (B, A) is assumed to be same as (A, B) . Thus we get symmetric matrix (only upper triangular matrix).

The directed graph G Figure 5 stored in memory in form of an adjacency matrix A Figure 7. The diagonal element of the matrix shows the occurrence of 1-itemsets and rest of elements shows occurrence of 2-itemsets.

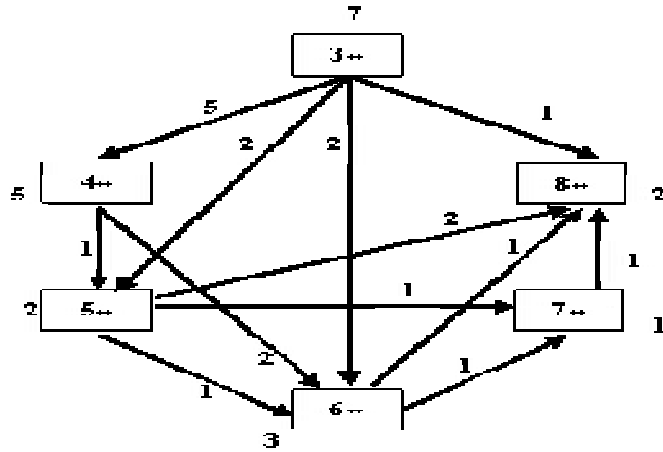


Figure 5 Directed Graph G for transaction Table 5

In second step, it checks count value of each element of the matrix A , if any diagonal element (for $i = j$) $A_{ij}count < \min_sup$ then, delete row and column of corresponding element from the matrix. Because, the super set of any infrequent itemset will never be frequent. Those elements for which $i \neq j$, $A_{ij}count < \min_sup$, assign zero value for them i.e. $A_{ij}count = 0$. After completion of second step we will gate filtered adjacency matrix.

$$\begin{matrix}
 & \begin{matrix} 3_{**} & 4_{**} & 5_{**} & 6_{**} & 7_{**} & 8_{**} \end{matrix} \\
 \begin{matrix} 3_{**} \\ 4_{**} \\ 5_{**} \\ 6_{**} \\ 7_{**} \\ 8_{**} \end{matrix} & = & \begin{bmatrix} A_{33} & A_{34} & A_{35} & A_{36} & - & A_{38} \\ & A_{44} & A_{45} & A_{46} & - & - \\ & & A_{55} & A_{56} & A_{57} & A_{58} \\ & & & A_{66} & A_{67} & A_{68} \\ & & & & A_{77} & A_{78} \\ & & & & & A_{88} \end{bmatrix}
 \end{matrix}$$

Figure 6Adjacency Matrix of Directed Graph G

In third step, we find all frequent patterns. All diagonal elements of filtered adjacency matrix show frequent 1-itemsets, and except diagonal element other element of adjaceny matrix (for which $count \neq 0$) shows frequent 2-itemsets. For extracting frequent k -itemsets($k > 2$), apply logical *AND* operation in between each row elements of the filtered adjacency matrix.

Finally following frequent itemsets will be generated from filtered adjacency matrix at concept level first.

Table:6 Level -1, minsup=4

Large 1- itemsets	Itemset	Support
	{3**}	7
	{4**}	5
Large 2- itemsets	{3**, 4**}	4

Similarly for getting frequent patterns from second, third and other lower levels, same procedure will followed as level first, i.e. at each concept level this approach scan datasets once and create a directed graph in the form of adjacency matrix, and generate all frequent patterns from given adjacency matrix. At each level we have used filtered transaction table e.g. for level-2, generate filtered transaction table T [2] by removing item which is infrequent and also removing the transaction in T [1] which contain only infrequent items.

Table 7 Filtered transaction table T [2]

TID	Items
T_1	{311, 321, 411, 421 }
T_2	{311, 411, 422 }
T_3	{312, 322, 421}
T_4	{311, 321 }
T_5	{311, 322, 411, 421 }
T_6	{313}

T_7	{331, 431}
-------	------------

The all possible frequent patterns mined from filtered transaction table T [2] at level second and level third (used T [3] which is generated by T [2]) are given in Table 8 and

Table 9 respectively.

Table 8 LEVEL-2 minsup=3

Large 1-itemsets	Itemset	Support
	{31*}	6
	{32*}	4
	{41*}	3
	{42*}	4
Large 2-itemsets	{31*, 32*}	4
	{31*, 41*}	3
	{31*, 42*}	4
	{32*, 42*}	3
	{41*, 42*}	3
Large 3-itemsets	{31*, 32*, 42*}	4

Table 9 Level-3 minsup = 3

Large 1-itemsets	Itemset	Support
	{311}	4
	{411}	4
	{421}	3
Large 2-itemsets	{311, 411}	3

CHAPTER 5

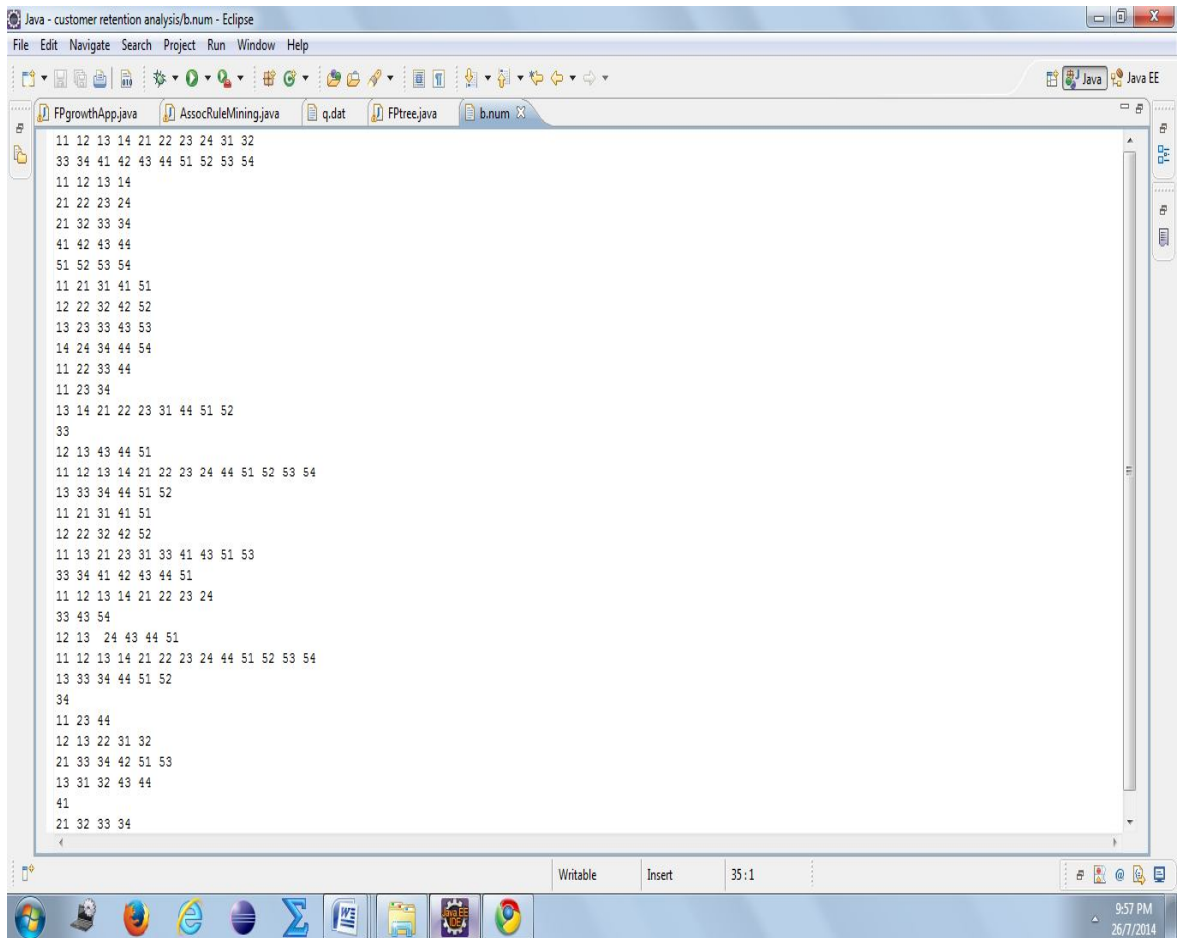
Implementation Details and Results

5.1 Implementation Details

- Development Language – Java
- Development Platform – Windows
- Test Framework – Java
- H/w & S/w Platform – Any on which Java Virtual Machine is supported

5.1 RESULTS

5.1.1 Sample Dataset

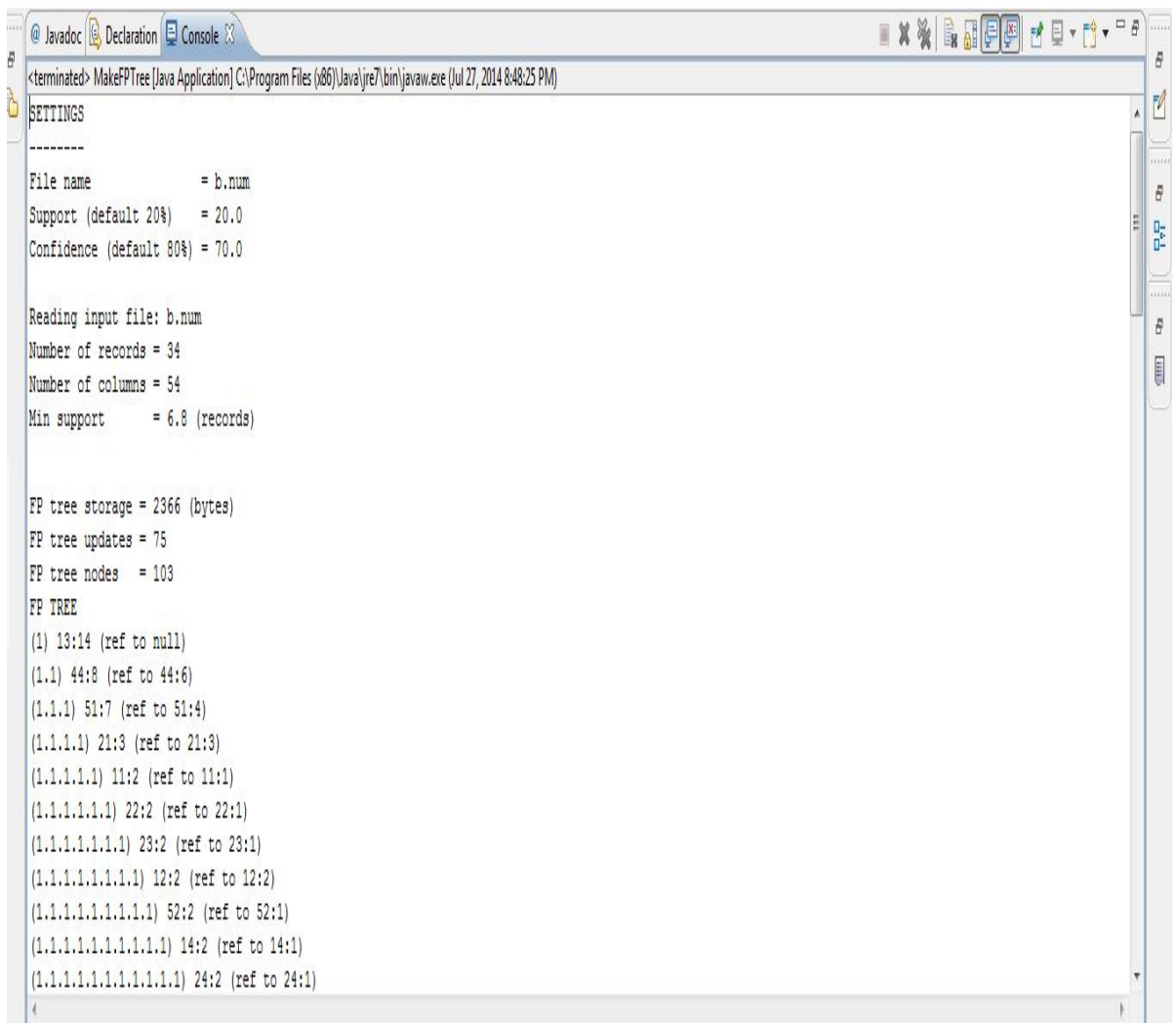


The screenshot shows the Eclipse IDE interface. The main editor window displays the content of the file 'b.num'. The text consists of a list of numbers, some on single lines and some grouped into lines. The numbers are: 11, 12, 13, 14, 21, 22, 23, 24, 31, 32, 33, 34, 41, 42, 43, 44, 51, 52, 53, 54. The numbers are arranged in a pattern that suggests they are being grouped into sets of four.

```
11 12 13 14 21 22 23 24 31 32
33 34 41 42 43 44 51 52 53 54
11 12 13 14
21 22 23 24
21 32 33 34
41 42 43 44
51 52 53 54
11 21 31 41 51
12 22 32 42 52
13 23 33 43 53
14 24 34 44 54
11 22 33 44
11 23 34
13 14 21 22 23 31 44 51 52
33
12 13 43 44 51
11 12 13 14 21 22 23 24 44 51 52 53 54
13 33 34 44 51 52
11 21 31 41 51
12 22 32 42 52
11 13 21 23 31 33 41 43 51 53
33 34 41 42 43 44 51
11 12 13 14 21 22 23 24
33 43 54
12 13 24 43 44 51
11 12 13 14 21 22 23 24 44 51 52 53 54
13 33 34 44 51 52
34
11 23 44
12 13 22 31 32
21 33 34 42 51 53
13 31 32 43 44
41
21 32 33 34
```

5.2 OUTPUT

5.2.1 Screenshots



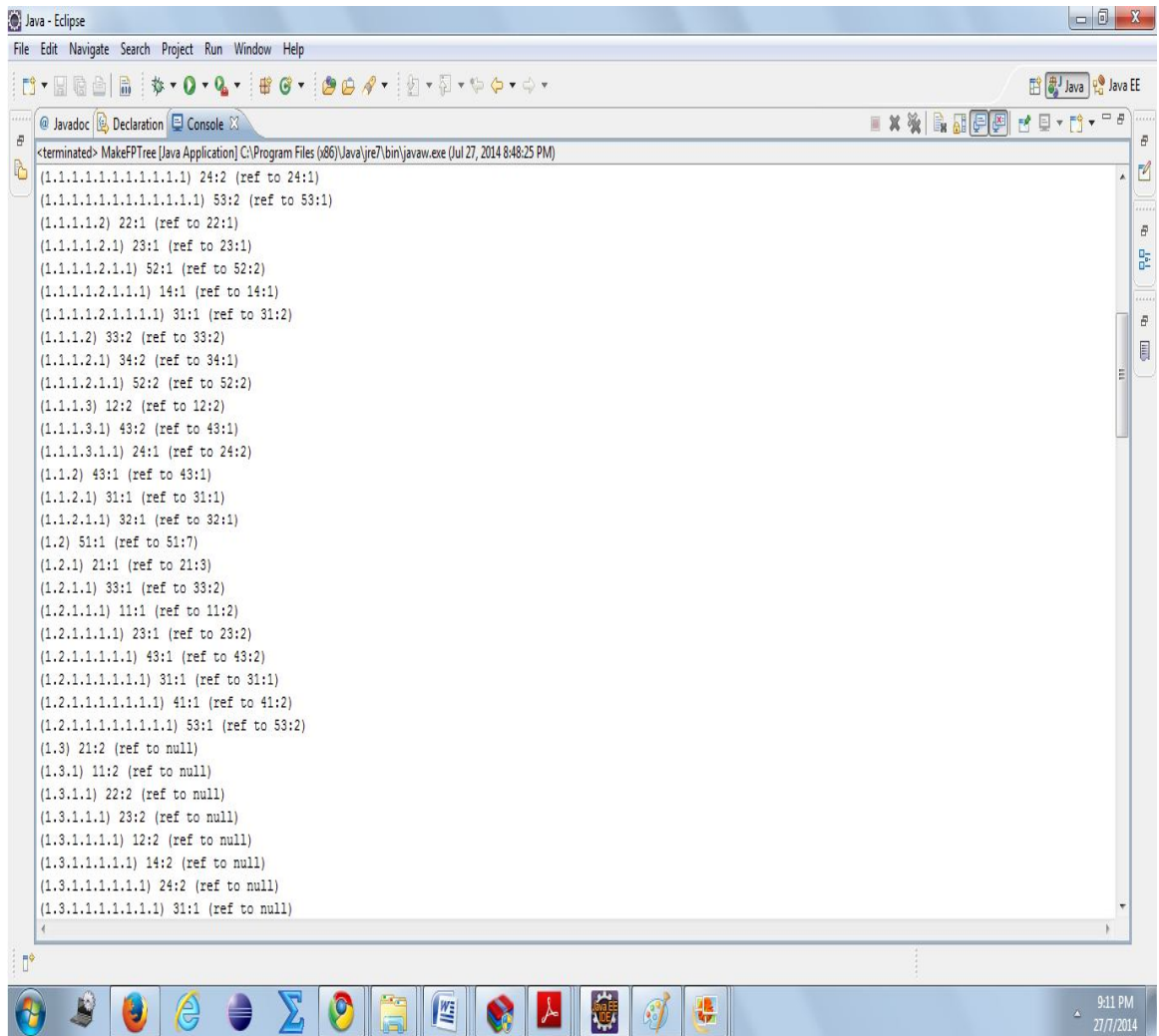
```
<terminated> MakeFPTree [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (Jul 27, 2014 8:48:25 PM)
SETTINGS
-----
File name           = b.num
Support (default 20%) = 20.0
Confidence (default 80%) = 70.0

Reading input file: b.num
Number of records = 34
Number of columns = 54
Min support       = 6.8 (records)

FP tree storage = 2366 (bytes)
FP tree updates = 75
FP tree nodes  = 103
FP TREE
(1) 13:14 (ref to null)
(1.1) 44:8 (ref to 44:6)
(1.1.1) 51:7 (ref to 51:4)
(1.1.1.1) 21:3 (ref to 21:3)
(1.1.1.1.1) 11:2 (ref to 11:1)
(1.1.1.1.1.1) 22:2 (ref to 22:1)
(1.1.1.1.1.1.1) 23:2 (ref to 23:1)
(1.1.1.1.1.1.1.1) 12:2 (ref to 12:2)
(1.1.1.1.1.1.1.1.1) 52:2 (ref to 52:1)
(1.1.1.1.1.1.1.1.1.1) 14:2 (ref to 14:1)
(1.1.1.1.1.1.1.1.1.1.1) 24:2 (ref to 24:1)
```

Figure 7 FP Tree Generation

Contd..



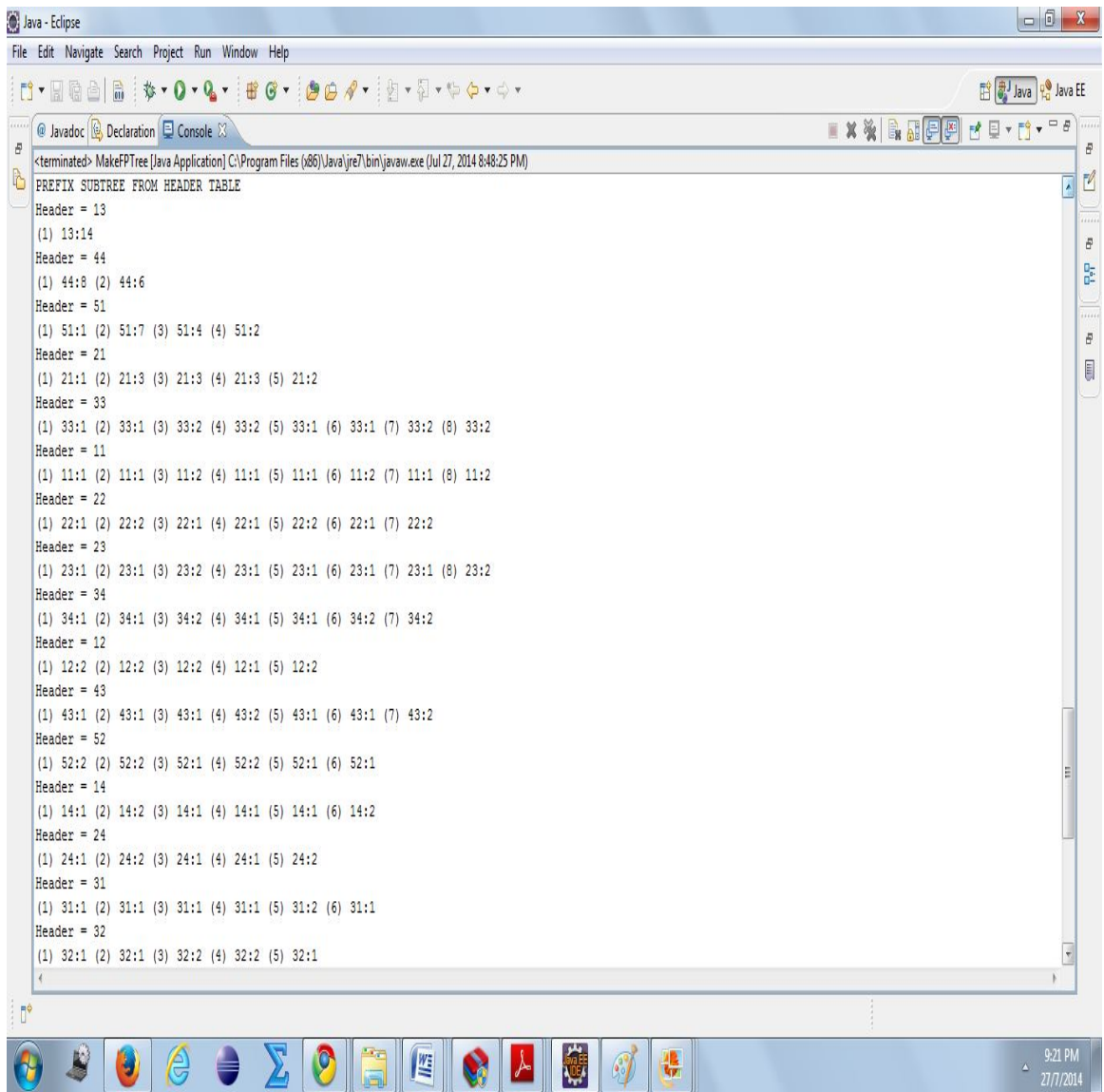


Figure 8 Prefix Subtree

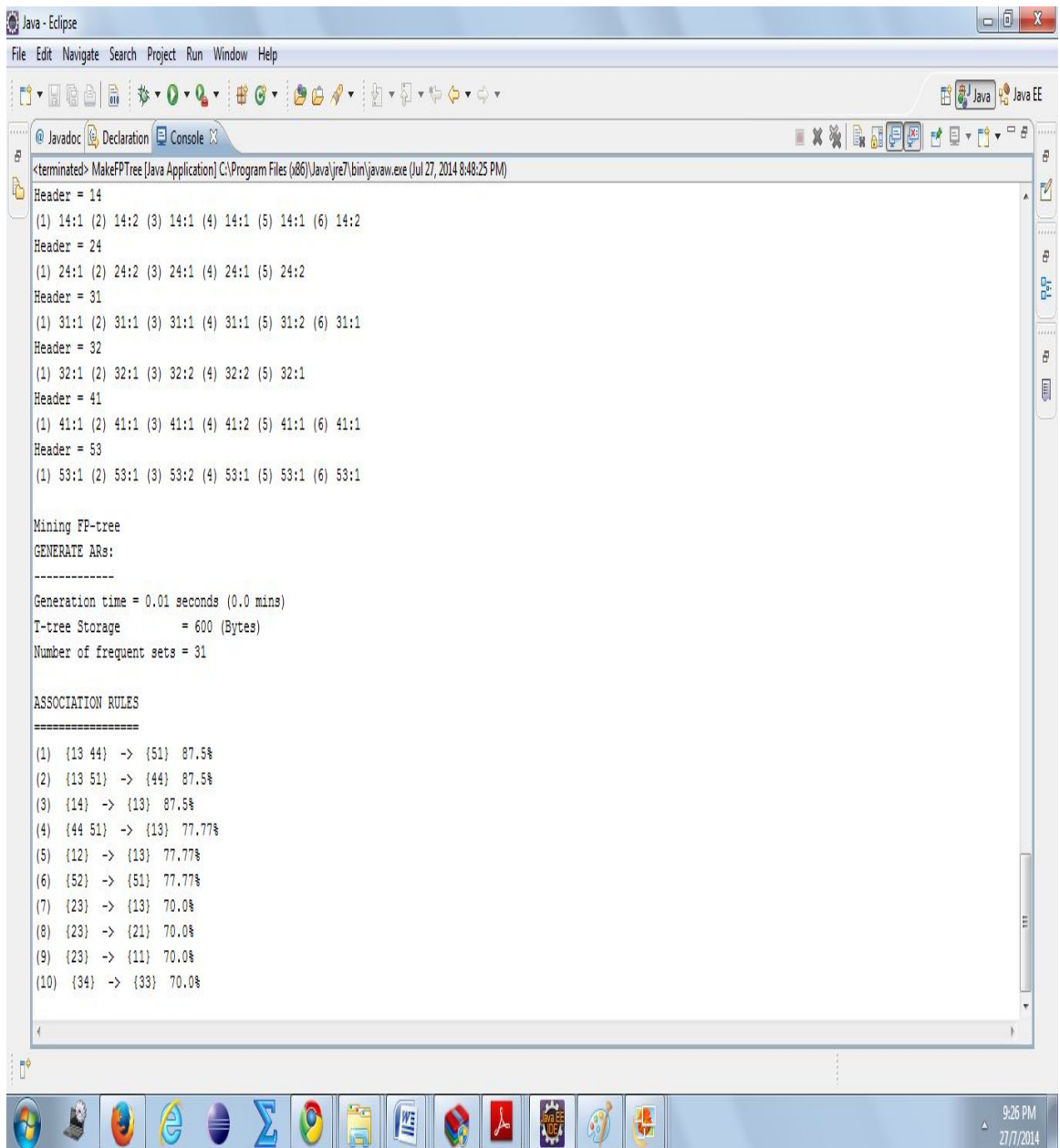


Figure 9 Generation of Association Rules

5.3 Graph Based Approach

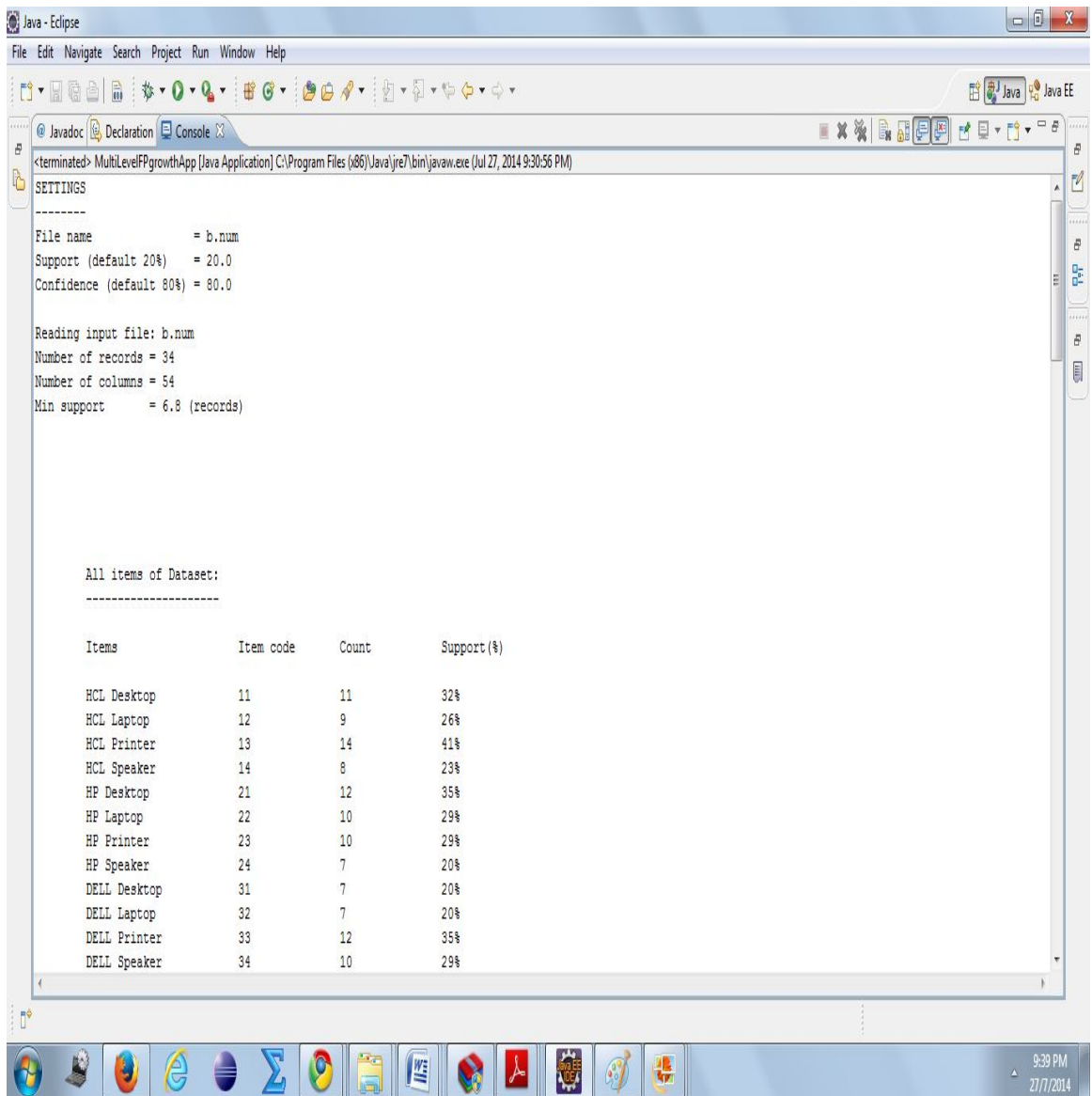


Figure 10 Multilevel Pattern Mining

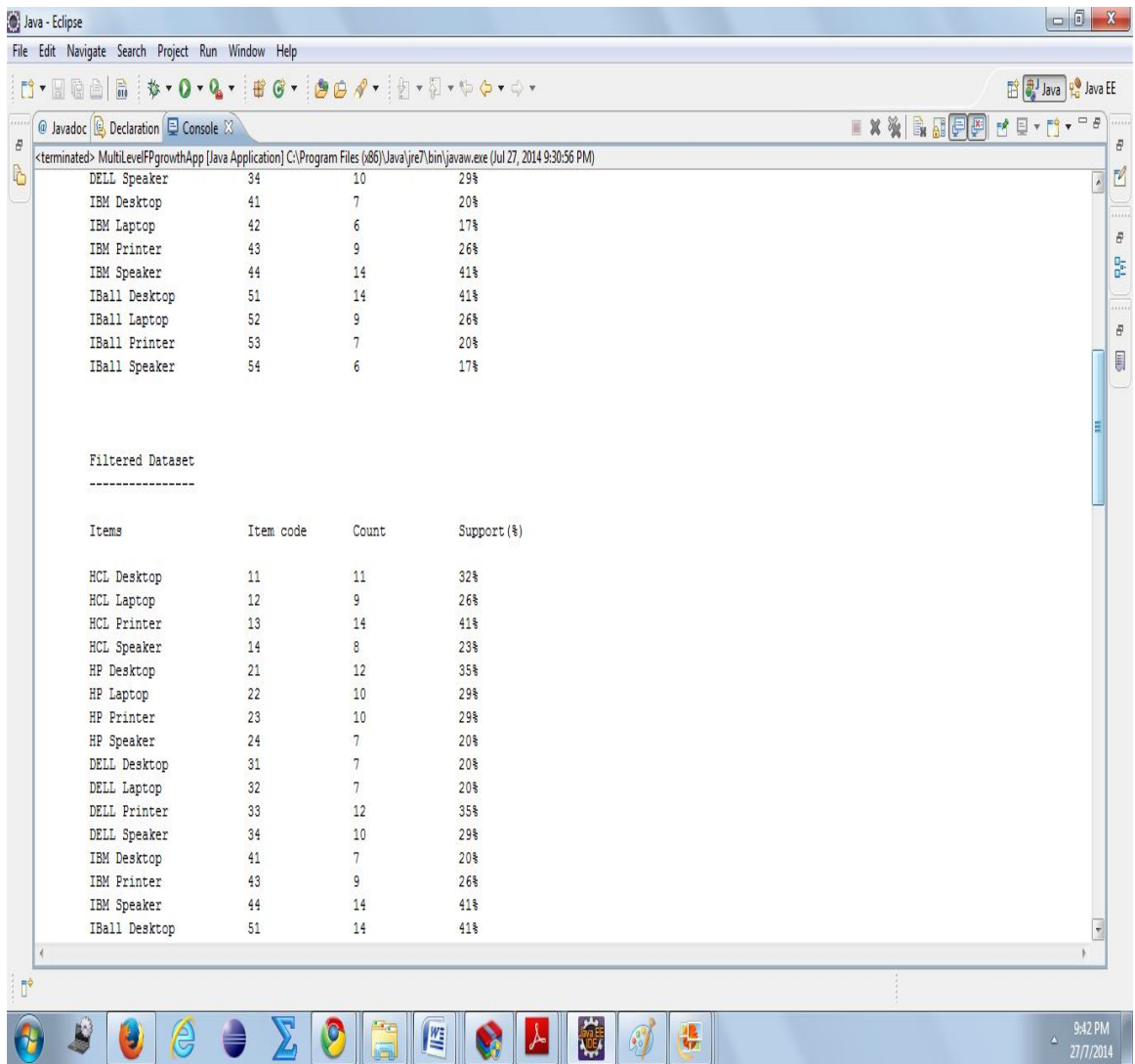


Figure 11 Filtered dataset

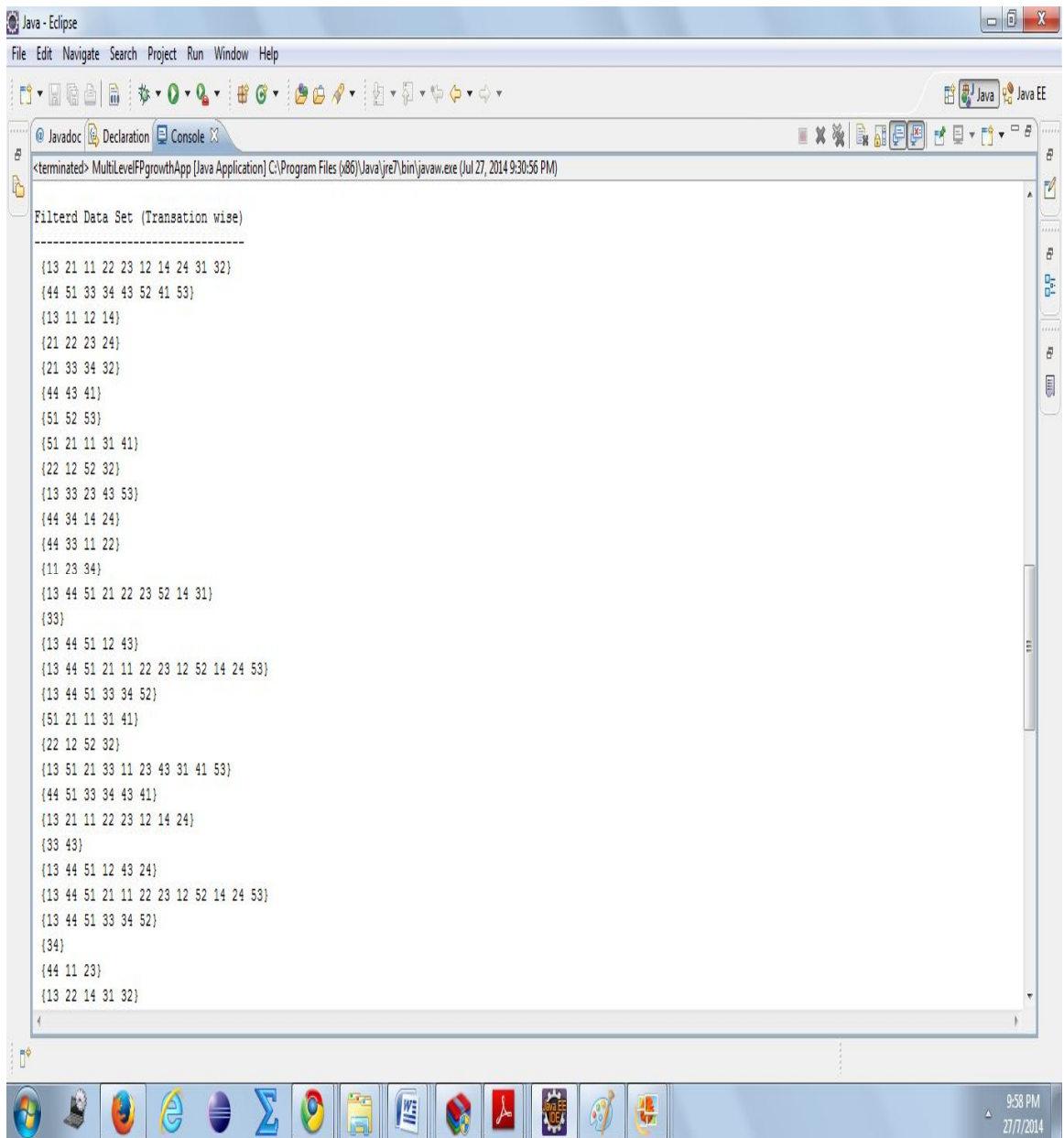


Figure 12 Filtered Dataset

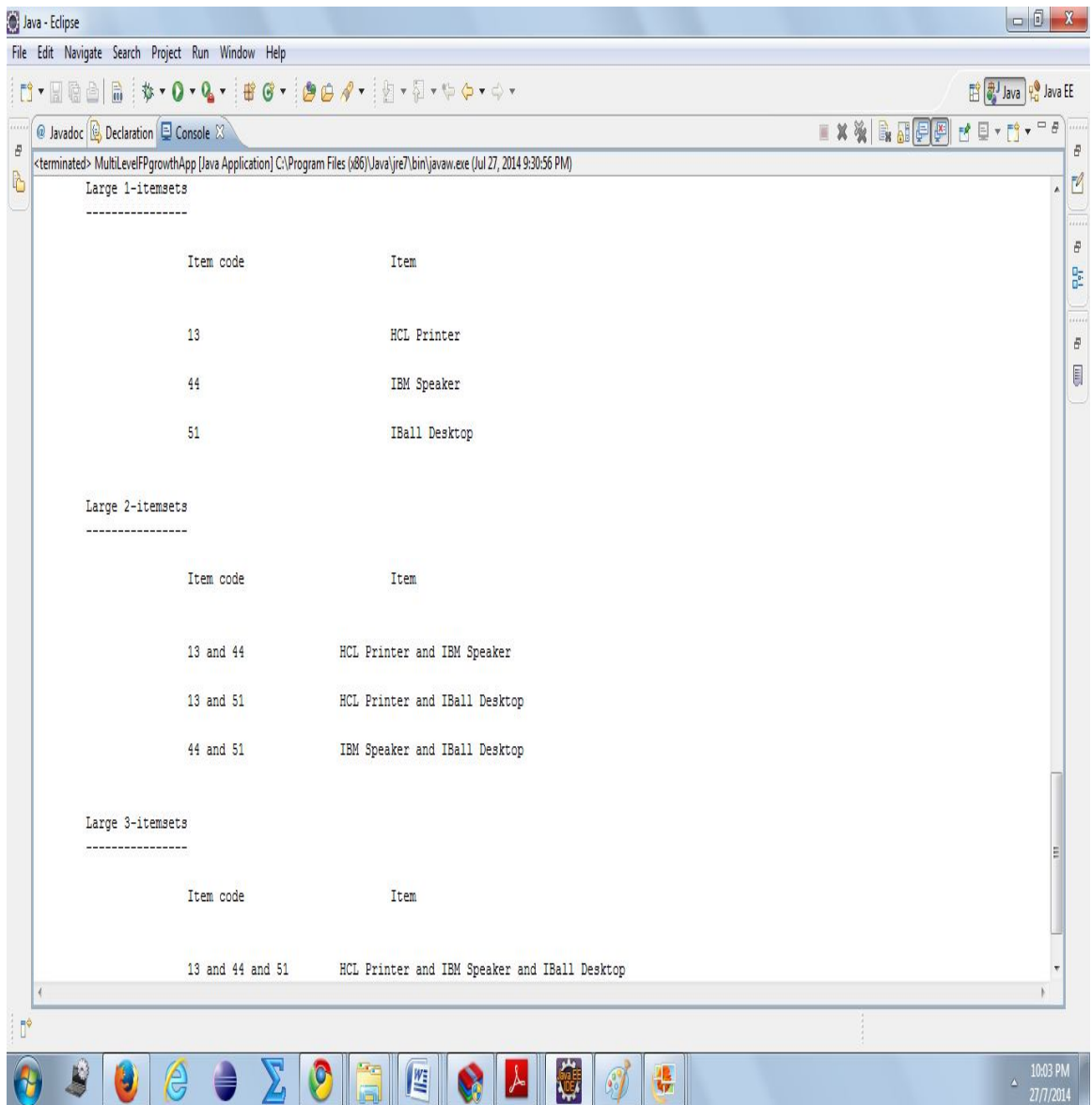


Figure 13 Frequent Itemsets

CHAPTER 6

CONCLUSION AND FUTURE SCOPE

Chapter Six: **Conclusion and Future scope**

From our research work and study it can be concluded that the recurrent pattern mining at multiple levels has major advantages over other single level frequent pattern mining approach. It carries more specific and concrete information than single level frequent pattern mining approach. We can obtain information at different hierarchal levels. The number of database scans for the candidate generation algorithm - Apriori algorithm increases with the dimension of the candidate itemsets, while the FP-growth algorithm needs at most two scans of database.

Thus, candidate generating algorithms like Apriori behave well only for small databases with a large support factor (at least 30%). In other cases the algorithms without candidate generation FP-growth behave much better and is memory saving also as compared to Apriori. Also with study of customer purchasing pattern at various abstract levels, it is easy for business people to target the customers with the most demanding products in a more efficient and lucrative way. This would not only result in more profits but would cut the customer churn to a great extent which would help in retention of the existing customers.

For future work, we would propose to incorporate other mining techniques to find multilevel changes by monitoring data over time through association rules. For example for census data, we may obtain associations which can help us figure out reasons why population in a certain age group say 60-70 is decreasing, why average income trends are shifting above 50000 per month etc. Moreover, qualitative assessment of data changes could also be studied.

REFERENCES

REFERENCES

- Agrawal, R., Imieliński, T., & Swami, A. (1993, June). Mining association rules between sets of items in large databases. In *ACM SIGMOD Record* (Vol. 22, No. 2, pp. 207-216). ACM
- Agrawal, R., & Srikant, R. (1994, September). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB* (Vol. 1215, pp. 487-499).
- Park, J. S., Chen, M. S., & Yu, P. S. (1995). *An effective hash-based algorithm for mining association rules* (Vol. 24, No. 2, pp. 175-186).
- Borgelt, C. (2003, November). Efficient implementations of apriori and eclat. In *FIMI'03: Proceedings of the IEEE ICDM workshop on frequent itemset mining implementations*.
- Wang, J., Han, J., & Pei, J. (2003, August). Closet+: Searching for the best strategies for mining frequent closed itemsets. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 236-245).
- Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1), 53-87.
- Chi, Y., Wang, H., Yu, P. S., & Muntz, R. R. (2004, November). Moment: Maintaining closed frequent itemsets over a stream sliding window. In *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on* (pp. 59-66). IEEE.
- Han, J., Pei, J., Yin, Y., & Mao, R. (2004). Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1), 53-87.
- Borgelt, C. (2005, August). Keeping things simple: Finding frequent item sets by recursive elimination. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations* (pp. 66-70). ACM.
- He, Z., Xu, X., Huang, Z. J., & Deng, S. (2005). Fp-outlier: frequent pattern based outlier detection. *Computer Science and Information Systems/ComSIS*, 2(1), 103-118.
- Kotsiantis, S., & Kanellopoulos, D. (2006). Association rules mining: A recent overview. *GESTS International Transactions on Computer Science and Engineering*, 32(1), 71-82.
- Han, J., & Kamber, M. (2006). *Data Mining, Southeast Asia Edition: Concepts and Techniques*. Morgan Kaufmann
- Thakur, R. S., Jain, R. C., & Pardasani, K. R. (2007). Fast Algorithm for Mining Multi-Level Association Rules in Large Databases. *Asian Journal of Information Management*, 1(1).
- Wu, S. T. (2007). Knowledge discovery using pattern taxonomy model in text mining.
- Liming, W., & Hui, Z. (2007). Algorithms of Mining Global Maximum Frequent Itemsets Based on FP-Tree [J]. *Journal of Computer Research and Development*, 3, 011.

- Peng, Y., Kou, G., Shi, Y., & Chen, Z. (2008). A descriptive framework for the field of data mining and knowledge discovery. *International Journal of Information Technology & Decision Making*, 7(04), 639-682.
- Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., & Steinberg, D. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1), 1-37
- Aggarwal, C. C., Li, Y., Wang, J., & Wang, J. (2009, June). Frequent pattern mining with uncertain data. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 29-38). ACM
- Li, H. F., & Lee, S. Y. (2009). Mining frequent itemsets over data streams using efficient window sliding techniques. *Expert Systems with Applications*, 36(2), 1466-1477.
- Tsai, P. S. (2009). Mining frequent itemsets in data streams using the weighted sliding window model. *Expert Systems with Applications*, 36(9), 11617-11625
- Novak, P. K., Lavrač, N., & Webb, G. I. (2009). Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *The Journal of Machine Learning Research*, 10, 377-403.
- Loekito, E., Bailey, J., & Pei, J. (2010). A binary decision diagram based approach for mining frequent subsequences. *Knowledge and Information Systems*, 24(2), 235-268.
- Gaber, M. M., Zaslavsky, A., & Krishnaswamy, S. (2010). Data stream mining. In *Data Mining and Knowledge Discovery Handbook* (pp. 759-787). Springer US.
- Shih, M. J., Liu, D. R., & Hsu, M. L. (2010). Discovering competitive intelligence by mining changes in patent trends. *Expert Systems with Applications*, 37(4), 2882-2890.
- Ng, W., & Dash, M. (2010). Discovery of frequent patterns in transactional data streams. In *Transactions on large-scale data-and knowledge-centered systems II* (pp. 1-30). Springer Berlin Heidelberg
- Lin, W. Y., Wei, Y. E., & Chen, C. H. (2011). A generic approach for mining indirect association rules in data streams. In *Modern Approaches in Applied Intelligence* (pp. 95-104). Springer Berlin Heidelberg.
- Liu, H., Lin, Y., & Han, J. (2011). Methods for mining frequent items in data streams: an overview. *Knowledge and information systems*, 26(1), 1-30.
- Papageorgiou, E. I. (2011). A new methodology for decisions in medical informatics using fuzzy cognitive maps based on fuzzy rule-extraction techniques. *Applied Soft Computing*, 11(1), 500-513.
- Kuo, R. J., Chao, C. M., & Chiu, Y. T. (2011). Application of particle swarm optimization to association rule mining. *Applied Soft Computing*, 11(1), 326-336.
- Chen, J., Lin, G., & Yang, Z. (2011, June). Extracting spatial association rules from the maximum frequent itemsets based on Boolean matrix. In *Geoinformatics, 2011 19th International Conference on* (pp. 1-5). IEEE
- Bifet, A., Holmes, G., Pfahringer, B., & Gavaldà, R. (2011, August). Mining frequent closed graphs on evolving data streams. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 591-599). ACM.

- Gu, T., Wang, L., Wu, Z., Tao, X., & Lu, J. (2011). A pattern mining approach to sensor-based human activity recognition. *Knowledge and Data Engineering, IEEE Transactions on*, 23(9), 1359-1372.
- Deypir, M., & Sadreddini, M. H. (2012). A dynamic layout of sliding window for frequent itemset mining over data streams. *Journal of Systems and Software*, 85(3), 746-75 Li, Y.,
- Chen, M., Li, Q., & Zhang, W. (2012). Enabling multilevel trust in privacy preserving data mining. *Knowledge and Data Engineering, IEEE Transactions on*, 24(9), 1598-1612.
- Colantonio, A., Di Pietro, R., Ocello, A., & Verde, N. V. (2012). Visual role mining: A picture is worth a thousand roles. *Knowledge and Data Engineering, IEEE Transactions on*, 24(6), 1120-1133.
- Deypir, M., Sadreddini, M. H., & Hashemi, S. (2012). Towards a variable size sliding window model for frequent itemset mining over data streams. *Computers & Industrial Engineering*, 63(1), 161-172.
- Nebot, V., & Berlanga, R. (2012). Finding association rules in semantic web data. *Knowledge-Based Systems*, 25(1), 51-62.
- Chen, H., Shu, L., Xia, J., & Deng, Q. (2012). Mining frequent patterns in a varying-size sliding window of online transactional data streams. *Information Sciences*, 215, 15-36.
- Sethi, N., & Sharma, P. (2013). Mining Frequent Pattern from Large Dynamic Database Using Compacting Data Sets. *International Journal of Scientific Research in Computer Science and Engineering*, 1.
- Quadrana, M., Bifet, A., & Gavaldà, R. (2013, October). An Efficient Closed Frequent Itemset Miner for the MOA Stream Mining System. In *Artificial Intelligence Research and Development: Proceedings of the 16th International Conference of the Catalan Association for Artificial Intelligence* (Vol. 256, p. 203). IOS Press.
- Rao, C. S., Giri, D. R., Shankar, R. S., & Kumar, S. P. (2013). Index Support for Item Set Mining-A Case Study. *Indian Journal of Advances in Computer & Information Engineering*, 1(1), 37-42.
- Leung, C. K. S., & Hayduk, Y. (2013, January). Mining frequent patterns from uncertain data with mapReduce for big data analytics. In *Database Systems for Advanced Applications* (pp. 440-455). Springer Berlin Heidelberg.
- Mining N-most Interesting Multi-level Frequent Itemsets without Support Threshold. In *Recent Advances in Information and Communication Technology* (pp. 125-134). Springer International Publishing.
- Pyun, G., Yun, U., & Ryu, K. H. (2014). Efficient frequent pattern mining based on linear prefix tree. *Knowledge-Based Systems*, 55, 125-139.
- Masciari, E., Shi, G., & Zaniolo, C. (2014). Trajectory Data Pattern Mining. In *New Frontiers in Mining Complex Patterns* (pp. 51-66). Springer International Publishing.
- Slimani, T., & Lazzez, A. (2014). Efficient Analysis of Pattern and Association Rule Mining Approaches. *arXiv preprint arXiv:1402.2892*. Chompaisal, S., Amphawan, K., & Surarerks, A. (2014).