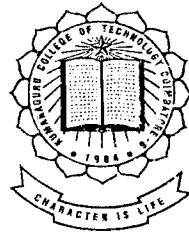




P-1443



# Efficient Data Mining Algorithms for use in Association Rule Mining

By

**Anand.M**

Reg. No. 71202621003

Of



**KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE**

**A PROJECT REPORT**

*Submitted to the*

**FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING**

*In partial fulfillment of the requirements  
for the award of the degree*

Of

**MASTER OF COMPUTER APPLICATION**

*June, 2005*

## BONAFIDE CERTIFICATE

Certified that this project report titled Efficient data mining algorithms for use in Association Rule Mining is the bonafide work of

Mr. M.Anand who carried out the research under my supervision.

Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.



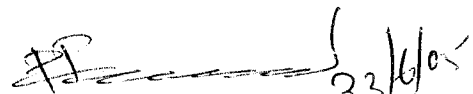
Head of the Department



Project Guide



Internal Examiner



External Examiner 23/6/05

## ABSTRACT

*“EFFICIENT DATA MINING ALGORITHMS FOR USE IN ASSOCIATION RULE MINING”* is an attempt to design new data mining algorithms, improve the existing ones for efficiency and also to adapt the existing algorithms to a different problem but within the domain of data mining. Since there is an increased availability of data due to switching over from manual to computer based systems, it has become a decreasingly viable option and possibility for humans to find information that is useful to the business or problem at hand using manual mechanisms. Therefore, use of efficient data mining algorithms becomes an absolute necessity.

Even though computer based information gathering systems tend to come up with a lot of results that are not relevant from the commercial point of view, their accuracy is evolving at a very fast clip. Also the very advantage of using such systems is that they may come up with results that the manual methods may fail to locate or pinpoint due to lack of insight and limited perception of the human brain. Data mining when used as a process of knowledge discovery helps very much in the decision making process.

The field of data mining although new, is very vast. Association Rule Mining which deals with techniques involved in finding interesting associations or correlation relationships among a large set of data items, is a subset of data mining and a separate field by itself. The field albeit a small part of data

mining is very interesting from the modern day business point of view. Almost all business decisions to be made depend on Association rule mining.

Efficiency of Data Mining algorithms used, in the purview of Association rule mining can be increased by design of new algorithms, data structures, using better data representation mechanisms, etc. This project focuses on single dimension association rules for the major part. Measuring the performance of the algorithms by itself is a task separate.

## ACKNOWLEDGEMENT

For any long endeavour to succeed in this modern world, requires a lot of support, encouragement, advice and the blessings of the almighty. I take this opportunity to express my gratitude and appreciation to all of those people out there who helped me when I was stuck.

I wish to thank Dr.S.Thangasamy, PhD, Professor and Head of the department, Computer Science and Engineering, for constantly encouraging me to pursue fresh ideas and set new goals for myself.

I wish to express my gratitude to my project guide Asst.Prof.A.Muthukumar, Course Coordinator, MCA. The very idea of the project as well as the material input and much needed support came from him.

## LIST OF TABLES

- 2.1 – Sample Database Table *Dt*
- 2.2 – A Sample Weblog File
- 2.3 – First Order Markov Model
- 2.4 – Second Order Markov Model
- 2.5 – Third Order Markov Model
- 2.6 – Database Sorted by Customer Id, Transaction time.
- 2.7 – Customer-Sequence Version of the Database
- 2.8 – The answer set sequential patterns.
- 2.9 – Diet habits of a specific ethnic groups.
- 2.10 – Diet habits of a specific ethnic groups.
- 3.1 – Node Count of Itemset tree and Corresponding execution times.
- 3.2 – Data count and Corresponding execution times.

## LIST OF FIGURES

- Figure 2.1 – Itemset tree construction(Step1-6)
- Figure 2.2 – Implementing the itemset tree
- Figure 2.3 – Rotated itemset tree
- Figure 3.1 – Node count Vs execution time
- Figure 3.2 – Data count and Corresponding execution time

# TABLE OF CONTENTS

Abstract	iii
List of Tables	Vi
List of Figures	vii

## **Chapter 1**

### **INTRODUCTION**

1.1 Project Outline	1
1.2 The world of Data Mining	1
1.2.2 Association Rule Mining	2
1.3 Choice of implementation Language	4

## **Chapter 2**

### **DESIGNING AND IMPLEMENTING ALGORITHMS**

2.1 Itemset tree data structure	5
2.2 Selective Markov Model	13
2.3 Mining sequential patterns	19
2.4 Multi-database Mining	23



**Chapter 3**

<b>EFFICIENCY DETERMINATION</b>	29
3.1 Code Profiling	29
3.1.1 Concept of clocks	30
3.2 Profiling tools	30
3.3 Results and Conclusions	34

**Appendices**

Appendix 1 – Glossary of terms	37
Appendix 2 – Source Code Sample	40
Appendix 3 – Sample Data	45
References	47

# CHAPTER 01

## INTRODUCTION

### 1.1 Project Outline

The focus of the project, "*EFFICIENT DATA MINING ALGORITHMS FOR USE IN ASSOCIATION RULE MINING*", is on choosing a set of mining algorithms that fit into the various stages of association rule mining. For instance, we start from the problem of choosing the right table from a database containing many tables. Then we proceed to problem of choosing the right attribute for use in association rule mining. Further we move to the problem efficiency of such chosen data mining algorithms, analyzing their performance and focus on optimizing them has also been considered.

### 1.2 The World Of Data Mining

Data mining also known by alternate terms such as knowledge discovery in databases is a set of techniques customized for the problem of uncovering useful patterns from large data sets. Data mining is primarily an offshoot of the database industry. An evolutionary path has been witnessed in database industry in development of various functionalities such as data collection and database creation, data management, data analysis, etc which affect the way data mining is applied to databases, since data mining is a higher level operation dependent upon the various above mentioned tasks.

The major reason the field of data mining has received so much attention in recent years, is due to the fact that the use of information technology has

resulted in vast amounts of accumulated data. Such accumulated data contains lots of hidden patterns and useful information that cannot be detected by manual inspection of the data due to the vast quantity of data, high cost of IT labour and the limited capacity of such human systems. Hence, data mining becomes inevitable. Data mining helps find information and knowledge that is useful in a variety of application domains such as business management, production control and market analysis to engineering design and science exploration.

It is simply not possible to apply data mining techniques to existing, in use databases, as such. Hence, the large data sets used for data mining are stored in data warehouses. Data warehouse is a database that contains historical information, and is a bit outdated. They are usually also stripped of all their constraints like locking mechanisms, integrity and security constraints. They are also de-normalised and certain fields are dropped because it is easier to carry out data mining tasks within a single table than across multiple ones.

When many data mining tasks are carried out simultaneously, we use a data warehouse server to take care of multiple requests for data. Data mining can also be carried out on data from relational, text, multimedia, transactional and other forms of database architectures. Data warehouse gathers data from various sources and locations which are often geographically apart. The gathered data is cleaned, transformed and integrated before being loaded into the warehouse. Data mining is a part of the process of knowledge discovery from databases.

Architecture of data mining systems have been classified into loose, semi-tight, tight and no coupling based on the amount of interaction and integration of the data mining system to an underlying database or data warehouse system. As the amount of coupling increases, the integration of the data mining system and data base increases. The utilization of third party utilities also increases.

## 1.2.2 Association Rule Mining

The art or technique of detecting interesting correlation relationships amongst a large set of items is done using association rule mining. The discovery of association relationships among huge amounts of business data helps in business decision making processes.

Association rule mining is classified broadly into single and multidimensional association rule mining. Single dimensional association rule mining focuses on using just one facet of the given data at a time to produce relationships from the data. They are the most widely used and simplest of mechanisms. The main problem with these mechanisms is their tendency to come up with a lot of patterns which may not be very useful in the decision making process. Multi-dimensional association rule mining helps make complex decisions but are not widely suitable for all domains. Moreover the multidimensional method gives very specific patterns.

In many applications, it is very difficult to find strong associations among data items at low or primitive levels of abstraction due to the sparsity of data in multidimensional space. Strong associations discovered at high concept levels may represent common sense knowledge. However, what may represent common sense to one may not be so common to another. Therefore we use multilevel association rule mining to mine rules at various levels of abstraction. There are plenty of methods of mining multilevel association rules.

A factor contributing to the interestingness of a pattern is the pattern's overall simplicity for human comprehension. Support and confidence are the two most widely used measures for determining the interestingness of data. Support of an association pattern refers to the percentage of task related data tuples for which the pattern is true. Confidence is a certainty measure for association rules

Market basket analysis is a prominent application that is based on association rule mining.

### 1.3 Choice Of Implementation Language

C++ lends itself as a language of choice for implementing the algorithms. Although C seems to be the first choice because of its high speed and ease with which algorithms can be implemented in it, the lack of data security tends to be a major dis-advantage. In addition, as the size of the program increases lack of modularity is a problem in C. Exception handling in C++ helps us to trap errors in code where they appear. This helps us to trace and debug these errors.

It is easy to define shared aspects of the implementation that are helpful to all derived classes. So extending the algorithms and at the same time retaining the features of the original within the same code is very much possible in C++. Also, C++ provides a rich set of profiling tools to measure the efficiency and performance of our code.

## CHAPTER 02

# DESIGNING AND IMPLEMENTING ALGORITHMS

### 2.1 The Item-Set Tree: A Data Structure for Data Mining

Discovering knowledge is an expensive operation. It requires extensive access of secondary storage that can become a bottleneck for efficient processing. Running data mining algorithms from scratch, each time there is a change in data, is obviously not an efficient strategy. Building a structure to maintain knowledge discovered could solve many problems, that have faced data mining techniques for years, that is database updates, accuracy of data mining results, performance, and ad-hoc queries.

The need for an incremental data mining approach is very much needed. So a mining technique based on a data structure called the itemset tree has been used. Advancements in data capturing technology have lead to exponential growth in amounts of data being stored in information systems. The approach is effective for solving problems related to efficiency of handling data updates, accuracy of data mining results, processing input transactions, and answering user queries.

## Basic assumptions

Association mining helps us to discover dependencies among the values of an attribute. It has over the years, emerged as a prominent research area. The data structure has been created with the association-mining problem also referred to as the *market basket* problem in mind.

Let us consider a database table containing transactions. The transactions are purchases made by various customers who visit the store, over a period of time. We denote a transaction by  $S = \{s_1, s_2, \dots, s_n\}$  where, each  $s_i$  is a separate transaction and  $s_i \in S$ . Let  $I$  be a set of items such that  $I = \{i_1, i_2, \dots, i_n\}$ . Many algorithms have been proposed to generate association rules that satisfy certain measures. A close examination of those algorithms reveals that the spectrum of techniques that generate association rules has two extremes:

- A transaction data file is repeatedly scanned to generate large itemsets. The scanning process stops when there are no more itemsets to be generated.
- A transaction data file is scanned only once to build a complete transaction lattice.

Each node on the lattice represents a possible large itemset. A count is attached to each node to reflect the frequency of itemsets represented by nodes.

In the first case, since the transaction data file is traversed many times, the cost of generating large itemsets is high. In the later case, while the transaction data file is traversed only once, the maximum number of nodes in the transaction lattice is  $2^n$ ,  $n$  is the cardinality of  $I$ , the set of items. Maintaining such a structure is expensive.

Many knowledge discovery applications, such as on-line services and World Wide Web, require accurate mining information from data that changes on a regular basis. In World Wide Web, every day hundreds of remote sites are created and removed. In such an environment, frequent or occasional updates may change the status of some rules discovered earlier. Also, many data mining applications deal with itemsets that may not satisfy data mining rules. Users could be interested in finding correlation between itemsets, not necessarily satisfying the measures of the data mining rules.

Here a new approach is proposed, that represents a compromise between the two extremes of the association mining spectrum. In the context of the proposed approach two algorithms are introduced. The first algorithm builds an item-set tree by traversing the data file once, that is used to produce mining rules. While the second algorithm allows users to apply on-line ad hoc queries on the item-set tree.

## The Item-Set Tree

The item-set tree  $T$  is a graphical representation of the transaction data file  $DT$ . All transactions that are having the same itemset, belong to the same transaction group. Let us consider a database table,  $DT$  (Table 2.1), containing transactions. The transactions are purchases made by various customers who visit the store  $S$ , over a period of time. The variety of items in the store is set to eight for reason of simplicity in usage.



TID	Transaction items	Date
1	1,3,4,7	05-Jan-2003
2	2,3	06-Jan-2003
3	2,3,5	11-Feb-2003
4	6,7	23-Feb-2003
5	1,4	25-Feb-2003
6	2,3,4	03-Mar-2003

### 2.1 Sample Database Table DT

So in order to represent the following table DT as a tree, we have to take into consideration the following issues :

- \*\* There is a root node  $r$ , where  $r \in DT$ .  $r$  is the root of the tree.
- \*\*  $f(s)$  is the frequency of transaction  $s$ , ie the count of the occurrences of  $s$  in  $DT$ .

**Case 1:** All nodes  $S_j$  (children of  $r$ ) are such that these do share no leading elements in  $s$ . When a leaf node  $s$  is inserted as a son of  $r$ ,  $f(s)$  is initiated to 1.

**Case 2:**  $s = S_j$ , the node already exists.  $f(s_j)$  is incremented by 1.

**Case 3:**  $S \subset_e S_j$ ,  $s$  is an ordered subset of node  $S_j$ . A node  $s$ , representing  $s$ , is inserted as a child of  $r$  and as a parent of  $S_j$ .  $f(s) = f(S_j) + 1$ .

**Case 4:**  $S_j \subset_e S$ , node  $S_j$  is an ordered subset of  $s$ . The subtree, that has  $S_j$  as a root, is examined and the procedure starts over again

**Case 5:**  $S \cap_e S_j \neq \phi$ , there exists an ordered intersection between  $S$  and  $S_j$ . Two nodes are inserted. A node  $S_i$ ,  $S_i = S \cap_e S_j$ , is inserted between  $r$  and  $S_j$ , and a node  $s$  is inserted as a child of  $S_i$ .  $f(S_i) = f(S_j) + 1$ , and  $f(s)$  is initiated to 1.

Now let us look at the steps involved in adding each transaction in DT, to the representation of the table DT. The first step is to create the root node  $r$  (See Fig 2.1.1.)

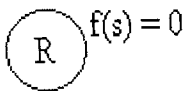


Figure 2.1.1

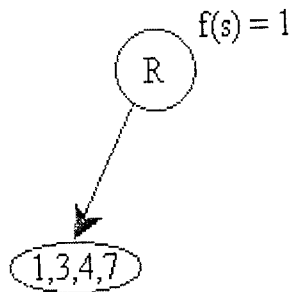


Figure 2.1.2

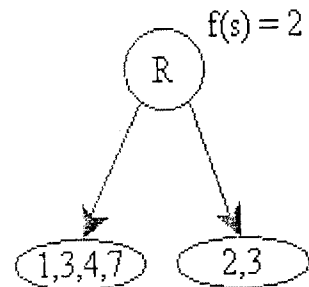


Figure 2.1.3

Each transaction is considered as a node and added to the tree.  $\{1,3,4,7\}$  is added as child of R (see Fig 2.1.1). Since elements  $\{1,3,4,7\}$  and  $\{2,3\}$  do not have any common first elements they are made as siblings (see Fig 2.1.1). Since  $\{2,3\}$  is common to both  $\{2,3\}$  and  $\{2,3,5\}$  the elements common to both, ie  $\{2,3\}$  is made as root and  $\{2,3,5\}$  is added as a child of it. The important thing is we do not take two copies of  $\{2,3\}$  but rather increment the frequency as needed (see Fig 2.1.4). Proceeding in the same manner  $\{6,7\}$  is added as child of root as it has no relation to any of the other nodes (see Fig 2.1.5).

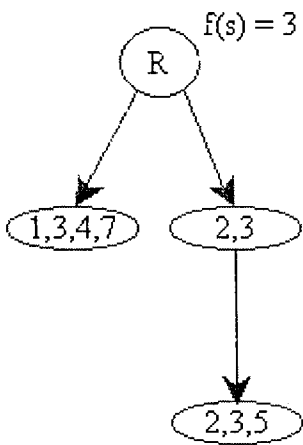


Figure 2.1.4

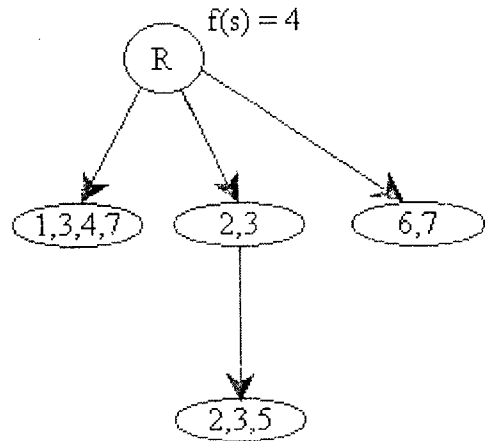


Figure 2.1.5

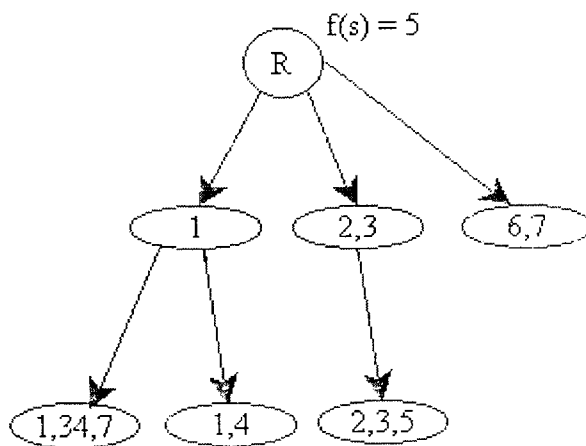
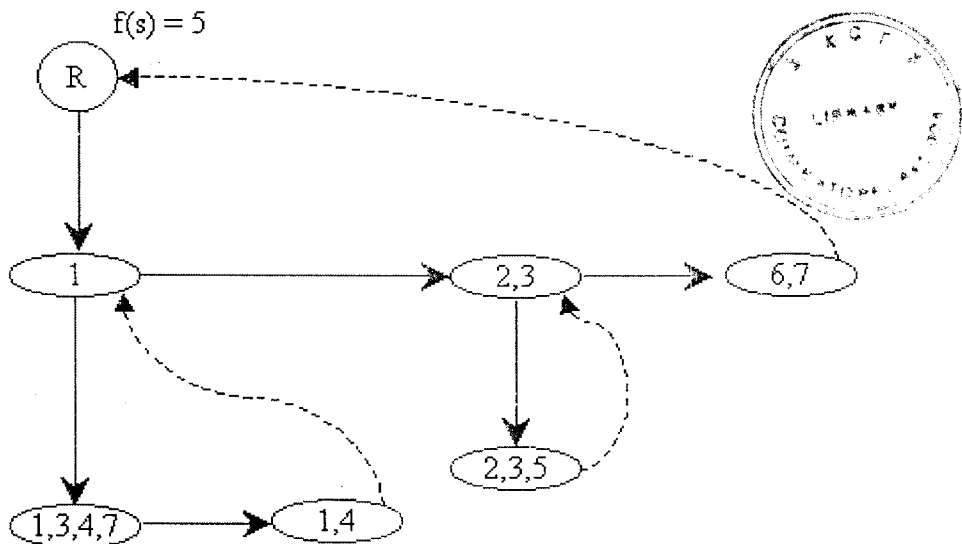


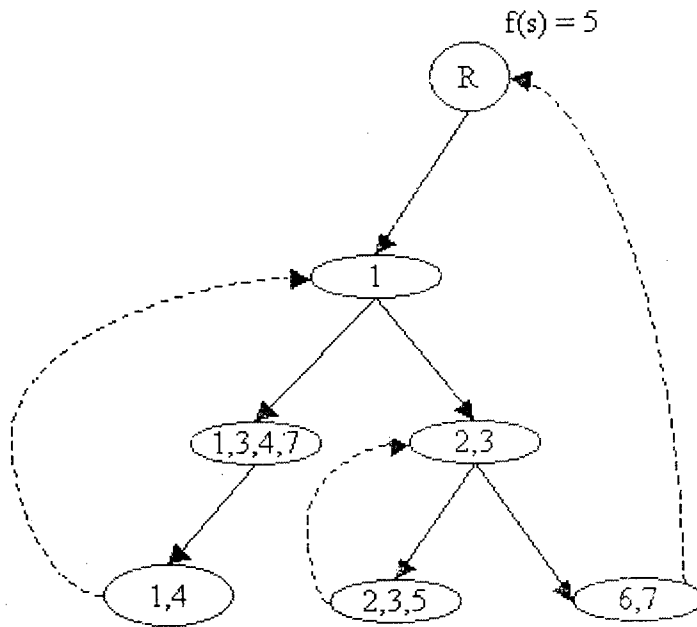
Figure 2.1.6 – Final Constructed tree

The problem now is how to represent a general tree in a program without loss of efficiency. Now let us consider the figure 2.1.6. In this tree we start from the root and to the left-most child L of the root, we attach to root using a down link. The rest of the nodes at that level i.e. the siblings of L are maintained as a linked list starting from L and proceeding to the last node linked by right pointers. The last node's right pointer points to the immediate parent i.e. root in this case. The constructed tree as per above logic is shown in Figure 2.2.



**Figure 2.2 – Implementing the itemset tree**

When the tree is rotated by an angle of  $45^\circ$ , we find that the general tree has been converted into an ordinary binary tree, without any loss in semanticity, data and efficiency. The rotated tree is shown in figure 2.3.



**Figure 2.3 – Rotated Itemset tree**

The size of the general tree can be expanded as much as needed. The only constraint is the availability of physical memory for storing the nodes of the tree.

## 2.2 SELECTIVE MARKOV MODEL

The problem of predicting a user's behaviour on a web site has gained importance due to the rapid growth of the world-wide-web and the need to personalize and influence a user's browsing experience. Markov models and their variations have been found well suited for addressing this problem. Higher order Markov models display high predictive accuracies. However, they are also very extremely complicated due to the large number of states that increases their space and run time requirements. Therefore, implementing these higher order Markov models is not very efficient. Here we explore the variations of Markov models that would be efficient as well as useful to the problem at hand.

The problem of modeling and predicting a user's surfing behaviour helps to improve search engines, web cache performance, understand and influence buying patterns and personalize the browsing experience.

In the world of mathematical probability, Markov models in their pure form have been used for studying and understanding stochastic processes. Therefore, the problem of predicting and modeling a user's surfing behaviour also being a variation of the stochastic process, Markov models tend to be useful for this.

The input for these problems is the sequence of web pages that were accessed by a user during a browsing session. Such sequences are traditionally stored in web log files on the internet server. The goal is to build Markov models that can be used to model and predict the web page that the user will most likely access next. For many applications first order Markov models are not very accurate in predicting the user's behaviour, since these models do not look far enough in to the past to correctly discriminate the different patterns. So, very often high order models become necessary. Unfortunately, these higher order

models though accurate, have a number of high state-space complexity, reduced coverage and sometimes even worse prediction accuracy.

As a solution to the problem at hand, the All- $k^{\text{th}}$  –order Markov model is used. Here we train varying order Markov models and use all of them during the prediction phase. As a result we have a solution that has low state complexity, improved prediction accuracy and retains the coverage.

### 2.2.2 An Illustration Of All- $K^{\text{th}}$ Order Markov Model

To make the understanding of Markov model easier, let us consider a practical working example. Given below is sample weblog file containing five browsing sessions of a user. Each session contains the various pages viewed by the user.

<b>SId</b>	<b>User Surfing Sequence *</b>
WS <sub>1</sub>	2,3,4,6,7,8
WS <sub>2</sub>	5,7,8
WS <sub>3</sub>	2,3,4,5,6,8
WS <sub>4</sub>	1,6,8
WS <sub>5</sub>	1,2,4,5,6,7,8

\* Web-page Identification number.

**TABLE 2.2 –A SAMPLE WEBLOG FILE**

Given the weblog file now we generate up to the third order Markov model for the All- $k^{\text{th}}$ -Order. It has been proved that Markov models are efficient up to only a certain order after which the space-time complexity increases.

## 2.2.2.2 First Order Markov Model

To generate the first order Markov model we first take into account the total unique pages from the web log file. Then we construct a Matrix with the first row and column being occupied by the unique elements identified. Then for every corresponding row/column value we update the value according to the pair's occurrence in the web log file.

For instance in Fig2.5, we have the value 2 against the row/column pair of {1,2}, which means in the log file the page numbered two was viewed twice by the user, immediately after viewing page numbered one. The row/column pair of {1,3}, {1,4}, {1,5} are null as not once in the log file are the pages 3 or 4 or 5, ever viewed contiguously after page one. Proceeding in a similar fashion we update the entire matrix by scanning the entire web log file once. The First order so generated serves as a reference when creating the second order.

<b>1<sup>st</sup> Order</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>
<b>1</b>	0	2	0	0	0	1	0	0
<b>2</b>	0	0	3	1	0	0	0	0
<b>3</b>	0	0	0	3	0	0	0	0
<b>4</b>	0	0	0	0	3	1	0	0
<b>5</b>	0	0	0	0	0	3	2	0
<b>6</b>	0	0	0	0	0	0	3	2
<b>7</b>	0	0	0	0	0	0	0	4
<b>8</b>	0	0	0	0	0	0	0	0

TABLE 2.3 – First Order Markov Model



### 2.2.2.3 Second Order Markov Model

The method to create the second order is to ensure that only those column values of the matrix in first order that have a non-null value to be considered. The second step is to generate two item pairs by parsing the first order matrix and combining the row/column values in first order matrix into a single item in an orderly manner. Then these combined items are used as the row values of second order's matrix. For instance, the values {1,2} and {2,3} are not null and hence they become the first two rows of the matrix. The next step is to parse the original web log file once again and search for values as was done for order one.

In Fig2.6, the values against the row/column value of {1,2}/3 is one, since in the original log file we have a single occurrence of the sequence {1,2,3}. Wherever there such sequences do not exist, the value is null as in first order's matrix.

2 <sup>nd</sup> order	1	2	3	4	5	6	7	8
{ 1,2 }	0	0	1	1	0	0	0	0
{ 1,6 }	0	0	0	0	0	0	0	1
{ 2,3 }	0	0	0	3	0	0	0	0
{ 2,4 }	0	0	0	0	1	0	0	0
{ 3,4 }	0	0	0	0	2	1	0	0
{ 4,5 }	0	0	0	0	0	3	0	0
{ 4,6 }	0	0	0	0	0	0	1	0
{ 5,6 }	0	0	0	0	0	0	2	1
{ 5,7 }	0	0	0	0	0	0	0	1
{ 6,7 }	0	0	0	0	0	0	0	3
{ 6,8 }	0	0	0	0	0	0	0	0
{ 7,8 }	0	0	0	0	0	0	0	0

TABLE 2.4 – SECOND ORDER MARKOV MODEL

#### 2.2.2.4 Third Order Markov Model

The generation of the third order is similar to the second order. The second order serves as the basis for the third order. Once we move higher up the order, only the original weblog file and the immediately preceding order serve as the input and basis for the generation of this order. The pure Markov model stops when the N<sup>th</sup> orders' matrix is a null matrix. But in this method we stop at the K<sup>th</sup> Order and take the results of all the orders to arrive at a suitable and effective solution. The important thing to note is that the diagonal values are null, since the transactions contain pages only till the recurrence of an already visited page. Using these orders, we can filter based on support or confidence to obtain results.

3 <sup>rd</sup> Order	1	2	3	4	5	6	7	8
{ 1,2,3 }	0	0	0	1	0	0	0	0
{ 1,2,4 }	0	0	0	0	1	0	0	0
{ 1,6,8 }	0	0	0	0	0	0	0	0
{ 2,3,4 }	0	0	0	0	2	1	0	0
{ 2,4,5 }	0	0	0	0	0	1	0	0
{ 3,4,5 }	0	0	0	0	0	2	0	0
{ 3,4,6 }	0	0	0	0	0	0	1	0
{ 4,5,6 }	0	0	0	0	0	0	2	1
{ 4,6,7 }	0	0	0	0	0	0	0	1
{ 5,6,7 }	0	0	0	0	0	0	0	2
{ 5,6,8 }	0	0	0	0	0	0	0	0
{ 5,7,8 }	0	0	0	0	0	0	0	0
{ 6,7,8 }	0	0	0	0	0	0	0	0

TABLE 2.5 – THIRD ORDER MARKOV MODEL

## 2.3 Mining Sequential patterns

Database mining is motivated by the decision support problem faced by most large retail organizations. Progress in bar-code technology has made it possible for retail organizations to collect and store massive amounts of sales data, referred to as the basket data. A record in such data typically consists of the transaction date and the items bought in the transaction. Very often, data records also contain customer-id, particularly when the purchase has been made using a credit card or a frequent-buyer card. Catalog companies also collect such data using the orders they receive.

We are given a database  $D$  of customer transactions. Each transaction consists of the following fields: customer-id, transaction-time, and the items purchased in the transaction. No customer has more than one transaction with the same transaction-time. We do not consider quantities of items bought in a transaction: each item is a binary variable representing whether an item was bought or not.

An *itemset* is a non-empty set of items. A *sequence* is an ordered list of itemsets. Without loss of generality, we assume that the set of items is mapped to a set of contiguous integers. Given a database  $D$  of customer transactions, the problem of mining sequential patterns is to find the maximal sequences among all sequences that have a certain user-specified minimum support. Each such maximal sequence represents a *sequential pattern*. We call a sequence satisfying the minimum support constraint a *large sequence*. Example Consider the database shown in Table 2.6. (This database has been sorted on customer-id and transaction-time.) Table 2.7 shows this database expressed as a set of customer sequences.

Customer ID	TransactionTime	Items Bought
1	June 25 '93	30
1	June 30 '93	90
2	June 10 '93	10, 20
2	June 15 '93	30
2	June 20 '93	40, 60, 70
3	June 25 '93	30, 50, 70
4	June 25 '93	30
4	June 30 '93	40, 70
4	July 25 '93	90
5	June 12 '93	90

TABLE 2.6 -Database Sorted by Customer Id, Transaction time.

Customer Id	Customer Sequence
1	((30), (90))
2	((10 20), (30), (40 60 70))
3	((30, 50,70))
4	((30) (40,70) (90))
5	((90))

TABLE 2.7 - Customer-Sequence Version of the Database

All the transactions of a customer can together be viewed as a sequence, where each transaction corresponds to a set of items, and the list of transactions, ordered by increasing transaction-time, corresponds to a sequence. We call such a sequence a *customer-sequence*. Formally, let the transactions of a customer, ordered by increasing transaction-time, be  $T_1, T_2, \dots, T_n$ . Let the set of items in  $T$  be denoted by  $\text{itemset}(T)$ .

With minimum support set to 25%, i.e., a minimum support, of 2 customers, two sequences:  $((30) (90))$  and  $((30) (40 70))$  are maximal among those satisfying the support constraint, and are the desired sequences. (Table 2.8).

<b>Sequential Patterns with support &gt; 25%</b>
$((30) (90))$
$((30) (40 70))$

TABLE 2.8: The answer set sequential patterns.

The sequential pattern  $((30) (90))$  is supported by customers 1 and 4. Customer 4 buys items (40 70) in between items 30 and 90, but supports the pattern  $((30) (90))$  since we are looking for patterns that are not necessarily contiguous. The sequential pattern  $((30) (40 70))$  is supported by customers 2 and 4. Customer 2 buys 60 along with 40 and 70, but supports this pattern since (40 70) is a subset of (40 60 70). An example of a sequence that does not have minimum support is the sequence  $((10 20) (30))$ , which is only supported by customer 2. The sequences  $((30)), ((40)), ((70)), ((90)), ((30) (40)), ((30) (70))$  and  $((40 70))$ , though having minimum support, are not in the answer because they are not maximal.

The problem of discovering what items are bought together in a transaction" over basket data was introduced. While related, the problem of

finding what items are bought together is concerned with finding intra-transaction patterns, whereas the problem of finding sequential patterns is concerned with inter-transaction patterns. A pattern in the first problem consists of an unordered set of items whereas a pattern in the latter case is an ordered list of sets of items.

## 2.4 Multidatabase Mining

Various tools and systems for knowledge discovery and data mining are developed and available for applications. However, when lots of databases exist, an immediate query is, where we start mining should be started from. It is not true that the more databases, the better for data mining. It is only true when databases involved are relevant to a task at hand. Breaking away from the conventional data mining assumption that many databases be joined into one, it is proved that the first step for multidatabase mining is to identify databases that are most likely relevant to an application; without doing so, the mining process can be lengthy, aimless and ineffective. A measure of relevance is thus proposed for mining tasks with an objective to find patterns or regularities about certain attributes. An efficient algorithm for identifying relevant databases is described. Experiments are conducted to verify the measure's performance and to exemplify its application.

With more and more databases created, an increasingly pressing issue is how to make efficient use of them. So lots of research interest on knowledge discovery and data mining has been created. Researchers are trying to develop efficient algorithms to cope with large volumes of data but little work has been devoted to the data aspect in the knowledge discovery process. In most organizations, data is rarely specially collected and stored for the purpose of mining knowledge, but usually as the byproducts of other tasks. Furthermore, with the development of technologies, it is not uncommon that an organization has a large number of database systems and diverse data sources.

Although most data mining algorithms assume a single data set, for real world applications, practitioners have to face the problem of discovering knowledge from multiple databases. In order to do so, one way is to employ a brute force approach to join the available tables into a single large table, upon which existing data mining techniques or tools can be applied. There are several



problems for this approach in real world applications. First, database integration itself is still a problematic area, especially where the source domains differ. Second, all tables with foreign key references need to be joined together to produce a single combined table. The size of the resulting table, in terms of both the number of records and the number of attributes, will be much larger than the original individual tables. The increase of data size not only prolongs the running time of mining algorithms, but also affects the behavior of mining algorithms. From the viewpoint of statistics, joining one relevant database with an irrelevant one will result in a more difficult task to find useful patterns as search space is enlarged by irrelevant attributes.

For this simplified analysis, we have not yet considered the factor of missing values due to joining. This factor will certainly increase the difficulty of data mining, too. Third, if databases are joined and data mining algorithms are applied, the users face the problem of identifying interesting patterns from a large number of discovered rules. In practice, it is too easy to discover a huge number of patterns in a database, however, it is difficult for users to search in all the discovered patterns for useful ones. The redundant, useless or uninteresting patterns can be even more easily generated when there are quite a number of databases irrelevant to the mining task. Therefore, as in any effective knowledge discovery process, the first important step in mining multiple databases is indeed to select those databases that are relevant to a specific mining task.

To understand the same we provide a motivating example to make the understanding much more easier. Two small databases are used to argue that for data mining tasks involving multiple databases, it is better to identify relevant database first before applying mining techniques. Databases are about people's diet habits. It is user's intention to discover some useful knowledge about Chinese diet habits from the databases. Here and in most KDD applications, being useful means something that should potentially lead to some useful actions by a user. Specific information is one type of usefulness.

In our example, the useful knowledge should be some diet habits specific to Chinese. Two tables (databases) with desired attributes are as shown in Tables 1 and 2. Let us look at Table 1 first. From the attribute values summarized in Table 3, we observe the following:

No person is on diet, regardless of his ethnic group; For both Chinese and Russian, 50% of records show that they take alcohol sometimes; Similarly, 50% of Chinese and Russian have snacks between regular meals; and It seems difficult to arrive at a convincing conclusion on favorite non-vegetable food since a very small number of tuples are available: there are five different foods listed and the largest group only has six records available.

From the above observations, we can see that, the database in fact does not contain information specific to Chinese. In such a case, it is fair to say that the database is irrelevant with respect to the query about Chinese. It is easy to see that database in Table 2.X is relevant to the query about Chinese since we can at least derive such a statement (rule).

"If the group is Chinese, the main food is rice" since all records about Chinese show that the main food is rice whereas records about Russian and Indian do not indicate such a habit. We have seen that although both databases contain information about the diet habits of Chinese, one of them contains relevant information but the other does not.

ID	Group	Alcohol	On-Diet ?	Snack B/w meals	Favorite Non-veg
950351	Chinese	Never	No	Sometimes	Fish
950301	Chinese	Never	No	Seldom	Pork
950282	Chinese	Sometimes	No	Seldom	Fish
940112	Chinese	Often	No	Often	Chicken
940023	Chinese	Sometimes	No	Sometimes	Beef
938976	Chinese	Sometimes	No	Sometimes	Pork
950612	Russian	Never	No	Seldom	Beef
950122	Russian	Often	No	Sometimes	Beef
940227	Russian	Sometimes	No	Sometimes	Chicken
938567	Russian	Sometimes	No	Sometimes	Pork
950348	Indian	Often	No	Sometimes	Fish
950312	Indian	Sometimes	No	Seldom	Pork
950123	Indian	Never	No	Sometimes	Chicken
940247	Indian	Sometimes	No	Sometimes	Fish
940100	Indian	Never	No	Sometimes	Beef
950312	Indian	Sometimes	No	Seldom	Pork

TABLE 2.9 – Diet habits of a specific ethnic groups.

<b>ID</b>	<b>Group</b>	<b>Main food</b>	<b>Regular eating times</b>	<b>Drink</b>	<b>Vegetarian</b>
950578	Chinese	Rice	3	Tea	No
950351	Chinese	Rice	3	Tea	No
950301	Chinese	Rice	3	Tea	No
950282	Chinese	Rice	3	Tea	No
940226	Chinese	Rice	3	Cola	No
940112	Chinese	Rice	3	Tea	Yes
950612	Russian	Bread	3	Coffee	No
950122	Russian	Bread	3	Cola	No
940227	Russian	Rice	3	Cola	Yes
940121	Russian	Bread	3	Tea	No
950348	Indian	Bread	3	Coffee	No
950312	Indian	Bread	3	Coffee	Yes
950123	Indian	Rice	3	Cola	No
940247	Indian	Rice	3	Coffee	No
940109	Indian	Bread	3	Tea	No

TABLE 2.10 – Diet habits of a specific ethnic groups.

To measure the Relevance factor

$$RF(s,Q) = Pr(s|Q).Pr(Q).log [Pr(s|Q)/Pr(s)]$$

Where,

RF is Relevance Factor

S is a selector. eg : "drink = tea"

Q is Query Predicate. eg : "group = chinese".

Pr(Q) and Pr(s) are priors and estimated by the ratios how frequently they appear in a database; and Pr(s|Q) is the posterior about the frequency ratio of s appearing given that Q occurs.

Given below are the various possibilities and the resulting conclusions.

**Case 1** : If  $Pr(s|Q)/Pr(s)$  is close to 1, i.e.,  $Pr(s|Q) \approx Pr(s)$ , s is independent of Q;

**Case 2** : If  $Pr(s|Q)/Pr(s)$  is close to 0, i.e.,  $Pr(s|Q)$  is almost 0, s rarely occurs given Q;

**Case 3** : If  $Pr(s|Q)/Pr(s)$  is less than 1, s is not frequent enough using when Q is given;

**Case 4** : If  $Pr(s|Q)/Pr(s)$  is greater than 1, then s occurs more often given Q than without Q, hence s and Q are correlated.

Thus given a set of tables in a database, we can find the one that is most useful and relevant to the problem at hand.

## CHAPTER 03

### EFFICIENCY DETERMINATION

The quality of the application we develop is determined by various factors such as speed of execution, scalability, portability, errors per KLOC, etc. Of these factors determining the efficiency of the code that we have developed is absolutely necessary for evaluating the quality of the code. However, considering the speed of current day processors fitted on personal computers, the concept of speed alone is not the true measure of efficiency. Still it is considered a good pointer to the quality.

#### 3.1 Code Profiling

The art of measuring the time taken for completion of execution by an application or some subset of the application is called *Code Profiling*. The subset we mention is referred to as a profile. Code profiling records the clock cycles expended by the profiled section of code or function i.e. the number of CPU clock cycles taken by the code that we have profiled. By repeatedly profiling our code, we get a clear picture as to which functions or sections of code are causing bottlenecks or hot spots. After localizing or pinpointing the bottlenecks, it's then just a matter of applying the code optimization techniques, to lower the number of clock cycles used by the function or section of code.

By knowing how frequently a piece of code is used, you can more accurately gauge the importance of optimizing that piece of code. There are a number of good tools for profiling user space applications. Two useful ways of profiling code are: counters and lock profiling. Any changes made in order to allow code profiling should be done only during development. Since these are not the sort of changes that we want to release to end users.

### 3.1.1 Concept Of Clocks

Clocks or clock cycles are what the program instructions consume while executing tasks. Code written in a high level language is compiled by a compiler. Compiler outputs a list of instructions native to the target machine. Each instruction requires a specific number of clock cycles to execute to completion. For the same instruction, different CPU's may take a different number of clock cycles or different approach to execute.

Hypothetically speaking, assume the ADD instruction needs 4 cycles to execute, on a computer equipped with Pentium 4 rated at 2.2 GHz (2, 200, 000, 000 cycles per second) then the time taken by CPU to execute:

$$\frac{2200000000 \text{ cycles per second}}{4 \text{ clocks for ADD instruction}} = 550000000 \text{ adds per second.}$$

## 3.2 Profiling Tools

The Visual C++ language provides a rich set of code profiling tools that provide varying range of accuracies, security and sophistication. Other than the set of tools provided in Visual C++ third party products are also available, each offering different advanced facilities. Some of the tools and the drawbacks in these tools also have been given.

## **Clock Function**

The clock function has been provided with the C language as part of the standard library. It is very simple to use in that we call clock function twice once before the start of the profiled code section and once at the end. The difference in times between the two return values is the approximate time taken by the section to execute to completion. Clock traditionally measures the time consumed in milliseconds. Hence the ability of this function to determine speed on modern processors is limited to very large time consuming programs.

## **GetTickCount Class**

The GetTickCount class is provided with Microsoft Foundation Classes. It has a better accuracy than the clock function. It has some serious issues regarding security.

## **QueryPerformanceCounter Class**

If the system supports a high-resolution counter, one can use QueryPerformanceCounter and QueryPerformanceFrequency to do high-resolution timings. QueryPerformanceCounter changes value between successive API calls, indicating its usefulness in high-resolution timing. The resolution in this case is on the order of a microsecond. Because the resolution is system-dependent, there are no standard units that it measures. One has to divide the difference by the QueryPerformanceFrequency to determine the number of seconds elapsed.

## **QueryPerformanceCounter() vs. GetTickCount()**

QueryPerformanceCounter uses the PC's clock counter just as GetTickCount, but it reads the current value of the countdown register in the timer chip to gain more accuracy -- down to 1.193MHz (about 800ns). However, it takes 5 to 10us to call QueryPerformanceCounter, because it has to do several port I/O instructions to read this value. In a multiprocessor NT,



QueryPerformanceCounter uses the Pentium cycle counter. You should not be using the raw value of QueryPerformanceCounter; you should always divide by QueryPerformanceFrequency to convert to some known time base.

QueryPerformanceCounter doesn't run at the same rate on all machines.

## **RDTSC Instruction**

**RDTSC** - Read Time Stamp Counter - returns the number of clock cycles since the CPU was powered up or reset. Intel introduced this instruction as a means of benchmarking Pentium processors. RDTSC is a two byte instruction - 0F 31 - and it returns a 64-bit count in EDX:EAX. One problem with RDTSC is that both IntToStr and Format('%d') can only handle longInts, not comps. A comp value can be passed to one of these functions, it cannot be any larger than High(LongInt), or 2,147,483,647, which is only a little over 16 seconds of clock ticks to a 133 MHz Pentium. To compare two long running processes, the difference between the start ticks and the stop ticks can easily exceed High(LongInt). In these cases, we need to use the CompToStr function:

## **CProfile software**

It's a class that uses two member functions Start and Finish and outputs a log file to disk or debug output window. The major benefit to this class is that by using the two functions, one can be more precise and local to what we want to analyze; Total control can be maintained over what we profile. This class is very object oriented. Another feature is, instead of explicitly supplying the start and end of your profiled section of code, the constructor starts recording and destructor stops recording. This means we are less likely to get faulty profile timings because now we are guaranteed to have properly recorded start and stop times. If one forgets the stop () in the other method, then obviously we are going to have incorrect measurement of clocks expended. The drawback to this

method is that now you don't have direct control over the code section you want to profile. Profiling starts whenever the object is instantiated and stops whenever the object is destroyed or goes out of scope. A PROFILER macro can be plugged into every function in your code (at the very beginning of each function) and leave it there and it acts like an ASSERT or TRACE macro. It only compiles under debug mode, it's not included in release.

### 3.3 Results and Conclusions

All the tests were carried out on an Intel Pentium II 133mhz processor. The reason being that the difference in computation time is discernible only in these low speed processors. The higher speed machines are so fast and the compilers so efficient that there is no discernible speed, when lower amounts of data are used. The results have been tabulated and for better understanding represented pictorially as graphs. The times shown may vary with different machines and also with the type of compilers installed. Various compiler optimizations have to be turned on to really notice the efficiency level of the code.

When Celeron processors are used, the computation time increases rapidly after the hundred thousand data threshold. However, as these processors are not classified for efficiency tests, these cannot be taken as a platform for measurement.

#### Efficiency test of Itemset tree

In the itemset tree as the number of nodes increase, the

<b>Nodes</b>	<b>Time (s)</b>
100	0.21
1000	0.48
5000	1.10
10000	1.407
50000	2.1
100000	3.34

**TABLE 3.1 Node Count of Itemset tree and Corresponding execution times.**

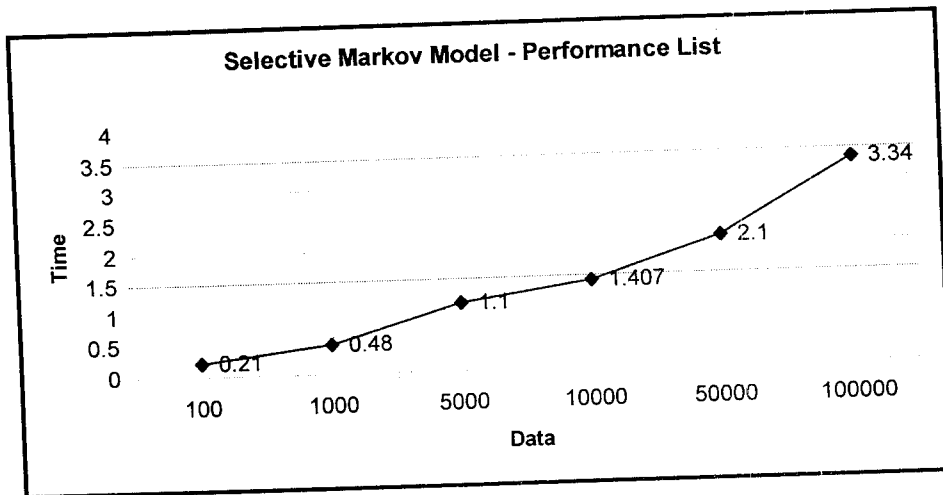


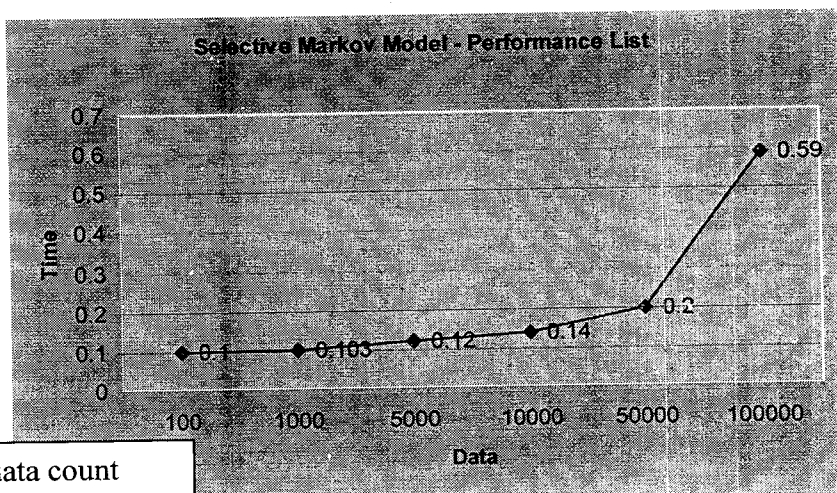
Figure 3.1 Node count Vs execution time

## Efficiency test of Selective Markov Model

Markov models show a linear increase in computation time as the amount of data increases. There is an even further steep increase in computation speed if the number of elements used is increased. With twenty elements and around seven hundred thousand data, the system crashes. At this stage a different data structure representation is needed, which at lower data values tends to be slow.

Data	Time (Sec)
100	0.1
1000	0.103
5000	0.12
10000	0.14
50000	0.20
100000	0.59

TABLE 3.2 Data count and Corresponding execution times.



x axis – data count  
y axis – execution

Figure 3.2 Data count and Corresponding execution time

# APPENDIX 01

## GLOSSARY OF TERMS USED

### **Association rule mining**

Association rule mining finds interesting association or correlation relationships among a large set of data items.

### **Classification**

It is the process of finding a set of models (or functions) that describe and distinguish data classes or concepts, for the purpose of being able to use the model to predict the class of object whose class label is unknown.

### **Clock**

The time taken by processor to complete one basic instruction unit. Varies for each different type and model of processors.

### **Code profiling**

The art of measuring the time taken for execution of a given piece of code.

### **Concept description**

Users like the ease and flexibility of having data sets described at different levels of granularity and from different angles. Such descriptive data mining is called concept description.

**CProfile**

A freeware utility class for code profiling that can be used for Visual C++. Has a very robust mechanism for profiling code without errors.

**Data cleaning**

Data cleaning routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data.

**Data cube**

A Data cube allows data to be modeled and viewed in multiple dimensions. It is defined by dimensions and facts.

**Data integration**

Merging of data from multiple data stores into a single entity. Helps to carryout higher level operations in a much more easier manner.

**Data mining**

Data mining refers to extracting or “mining” knowledge from large amount of data.

**Data selection**

Data selection is where data relevant to the analysis task are retrieved from the database.

**Data transformation**

Where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, for instance

## **Data warehouse**

Data warehouse provides storage mechanisms, architectures and tools for business executives to systematically organize, understand and use their data to make strategic decisions.

## **Multi-dimension Association Rule Mining**

Association rules that take into consideration more than one dimension or attribute of the data to mine relationships.

## **Pattern evaluation**

To identify the truly interesting patterns representing knowledge based on some interestingness measures.

## **Prediction**

Prediction can be viewed as the construction and use of a model to assess the class of an unlabeled sample, or to assess the value or value ranges of an attribute that a given sample is likely to have.

## **Single dimension Association Rule Mining**

Association rules that contain a single predicate are referred to as single dimension association rules.

## **Support**

Support is a measure of what fraction of the population satisfies the both the antecedent and the consequent of the rule.

## **Vtune**

A third party tool for code profiling from Intel® that has very sophisticated and advanced features.



## APPENDIX 02

### SAMPLE SOURCE CODE

#### DATA GENERATOR SOURCE CODE

```
/*
 * Source code to generate random itemsets for use in
 * Association rule mining
 */

/*
 * randomDG.h
 */

#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
#include <time.h>

class Random
{
private:
    int i_List[100]; // to store items in each basket
    short int i_noi; // no of items per basket

    int i, j, tmp;
    ofstream out;

public:
    Random();
    void generate(int turn = 0);
```

```
void bubbleSort();
void display();
void sendToFile();
~Random();

};

/*
 * randomDG.cpp
 */

#include "RandomDG.h"

Random::Random()
{
    srand( (unsigned)time( NULL ) );
    i_noi = 0;
    out.open("Rand.txt");
}

void Random::generate(int turn)
{
    // Set number of items in this basket
    i_noi = turn;
    if(i_noi == 0)
        i_noi = 3;

    bool bflg = false;

    // generate the required no. of items
    for(i=0;i < i_noi;)
    {

        tmp = rand() % 10;
        if(tmp == 0)
            bflg = true;

        for(int x=0; x < i;x++)
            if(i_List[x] == tmp)
                bflg = true;
    }
}
```

```
        if(bflg == false)
        {

                i_List[i] = tmp;
                i++;

        }
        else
        {

                bflg = false;

        }
} // end of function

void Random::bubbleSort()
{

        for(int i=0;i<i_noi;i++)
                for(j=i+1;j<i_noi;j++)
                        if(i_List[i] > i_List[j] )
                        {

                                tmp = i_List[i];
                                i_List[i] = i_List[j];
                                i_List[j] = tmp;

                        }

}

void Random::display()
{

        for(i=0;i < i_noi;i++)
        {

                cout<<i_List[i];

                if(i+1 != i_noi)
                        cout<<",";

        }
        cout<<endl;
}
```

```

void Random::sendToFile()
{
    if(out.is_open())
    {
        out<<endl;
        for(i=0;i < i_noi;i++)
        {
            out<<i_List[i];
            if(i+1 != i_noi)
                out<<",";
        }
    }
}

Random::~~Random()
{
    out.close();
}

/*
 * Random number generator for Market Basket Analysis
 */

#include "Random.h"

void main()
{
    long l,l_nob;
    Random gen;
    char *strFileName = new char[25];

    cout<<"\n Enter the number of baskets : ";
    cin>>l_nob;

```

```
// we need 1000 items
for(l=0;l<l_nob;l++)
{
    gen.generate(l);
    gen.bubbleSort();
    gen.sendFile();
}

delete(strFileName);
}

/*
 * End of sample source
 */
```

## APPENDIX 03

### SAMPLE DATA

#### Data for itemset tree with 9 different items

1, 5, 6  
9  
2, 7  
1, 2, 9  
2, 3, 7, 9  
1, 3, 5, 8, 9  
1, 4, 5, 7, 8, 9  
8  
2, 4, 5, 7, 8, 9  
5, 7, 8  
1, 2, 3, 6  
1, 2, 5, 6  
3, 4, 6, 8, 9  
1, 4, 5, 7, 8, 9  
1, 2, 5, 7, 8, 9  
1, 2, 3, 4, 6, 7, 7  
2, 3, 4, 6, 8  
2, 6, 9  
1, 6, 7, 8  
4, 7  
1, 2, 3, 4, 5, 6, 7  
1, 2, 3, 5, 6, 8, 8, 7  
1, 2, 3, 4, 6, 8, 8, 7  
5, 7  
1, 6, 8  
5, 8, 9  
1, 4, 5, 6, 7, 7, 8, 7  
4, 7, 8  
5, 9  
3, 4  
1, 2, 5, 6, 7, 7, 8, 7

## Sample Data For Multidatabase Mining

ID \* Group \* Alcohol \* On-Diet? \* Snack\_B/w\_meals \* Fav\_Non-veg

---

```

950351 * C * 0 * 0 * 0 * F
950301 * C * 0 * 0 * 1 * P
950282 * C * 1 * 0 * 1 * F
940112 * C * 2 * 0 * 2 * C
940023 * C * 1 * 0 * 0 * B
938976 * C * 1 * 0 * 0 * E
950612 * R * 0 * 0 * 1 * B
950122 * R * 2 * 0 * 0 * B
940227 * R * 1 * 0 * 0 * C
938567 * R * 1 * 0 * 0 * P
950348 * I * 2 * 0 * 0 * F
950312 * I * 1 * 0 * 1 * P
950123 * I * 0 * 0 * 0 * C
940247 * I * 1 * 0 * 0 * F
940100 * I * 0 * 0 * 0 * B

```

## Sample Data For Selective Markov Model With maximum eight elements

---

```

2      3      4      6      7      8
5      7      8
2      3      4      5      6      8
1      6      8
1      2      4      5      6      7      8
1      2      3      4      5      6      7      8
5      7

```

## REFERENCES

### RESEARCH ARTICLES

- [1] R. Agrawal, T. Imilienski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," Proc. of the ACM SIGMOD Int'l Conf. On Management of data, May 1993.
- [2] R. Agrawal, and R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. Of the 20<sup>th</sup> VLDB Conference, Santiago, Chile, 1994.
- [3] H. Mannila, H. Toivonen, and A. Verkamo, "Efficient Algorithms for Discovering Association Rules," AAAI Workshop on Knowledge Discovery in databases (KDD-94) , July 1994.
- [4] Alaaeldin Hafez, Jitender Deogun, and Vijay V. Raghavan, "The Item-Set Tree: A Data Structure for Data Mining"
- [5] Ramakrishnan srikant and Ramesh Agrawal "Mining sequential patterns".
- [6] Ramakrishnan srikant and Ramesh Agrawal "Mining sequential patterns : Generalizations and Performance improvements".
- [7] Huan Liu, Hongjun Lu, Jun Yao, "Towards Multidatabase mining : Identifying relevant databases". IEEE TKDE13:(4) 541-553 ,July/August 2001.
- [8] H. Mannila, H. Toivonen, and A. Verkamo, "Efficient Algorithms for Discovering Association Rules," AAAI Workshop on Knowledge Discovery in databases (KDD-94) , July 1994.
- [9] Data mining concepts and techniques by Jiawei Han and Micheline Kamber