CHAPTER 1

INTRODUCTION

1.1 MARKET BASKET ANALYSIS (MBA)

Frequent itemset mining leads to the discovery of associations and correlations among items in large transactional or relational data sets. With massive amounts of data continuously being collected and stored, many industries are becoming interested in mining such patterns from their databases.

A typical example of frequent itemset mining is market basket analysis. This process analyzes customer buying habits by finding associations between the different items that customers place in their "shopping baskets" (Figure 1.1).

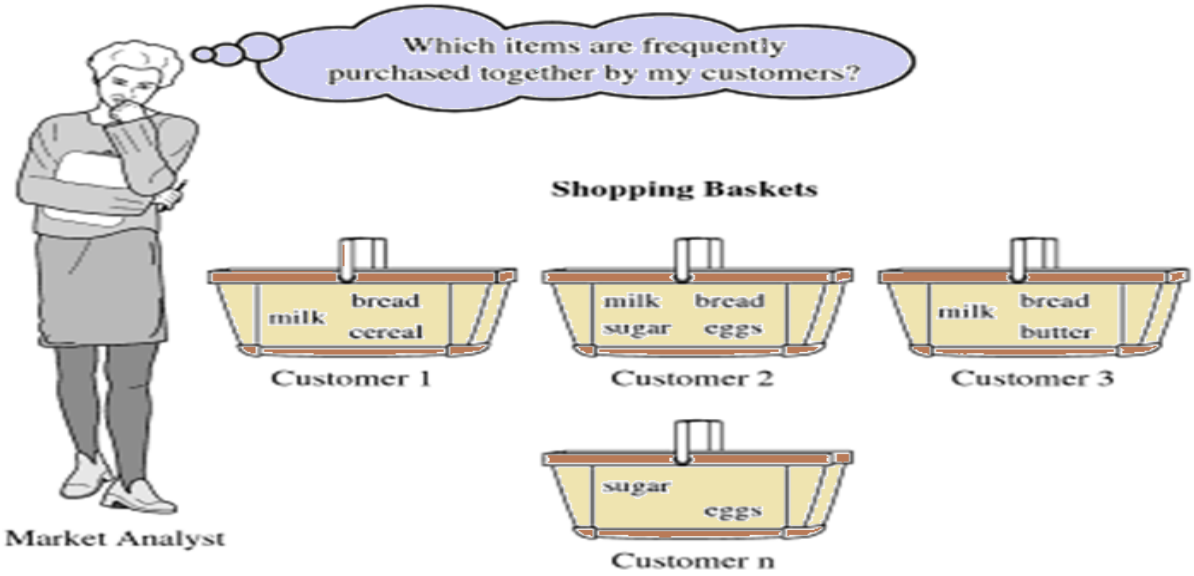


Figure 1.1 Market Basket Scenario

The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. For instance, if customers are buying milk, how likely are they to also buy bread (and what kind of bread) on the same trip to the supermarket? Such information can lead to increased sales by helping retailers do selective marketing and plan their shelf space.

1.2 PERTINANCE OF MBA

Increasingly, retailing is becoming a high-performance sport, and like athletes, retailers are seeking a competitive edge through technology. MBA, also known as affinity analysis, has emerged as the next step in the evolution of retail merchandizing and promotion. MBA allows leading retailers to quickly and easily look at the size, contents and value of their customer's market basket to understand the patterns in how products are purchased together, or basic product affinities. Advanced implementation s of MBA leverage near-instant results to encourage "*train-of-thought*" or interactive analysis, enabling retailers to drill-down into customer buying patterns over time to precisely target and understand specific combinations of products, departments, brands, categories and even time of day.

1.3 ASSOCIATION RULE MINING

Association Rule Mining (ARM) refers to the problem of identifying the relations between the elements of a given data set. This process usually involves two distinct stages: the first one involves determining all frequent itemsets present within the collection of transactions, while the second stage aims to find strong association rules based on the previously identified itemsets. The first stage is generally considered to be the most important one with respect to the overall efficiency of the ARM algorithms.

1.3.1 Apriori Algorithm

In computer science and data mining, Apriori is a classic algorithm for learning association rules. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation). As is common in association rule mining, given a set of itemsets (for instance, sets of retail transactions, each listing individual items purchased), the algorithm attempts to find subsets which are common to at least a minimum number C of the itemsets. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

The purpose of the Apriori Algorithm is to find associations between different sets of data. It is sometimes referred to as "Market Basket Analysis". Each set of data has a number of items and is called a transaction.

TID	List of ItemIDs
T100	milk,bread,butter
T200	bread,cereal
T300	bread,jam
T400	milk,bread,cereal
T500	milk,jam
T600	bread,jam
T700	milk,jam
T800	milk,bread,jam,butter
T900	milk,bread,jam

Input :Transaction Database D

Table 1.1 Sample Transaction Database

Frequent itemsets of market basket analysis done over the above given transactional database is shown below:

ItemSet	Supp_Count
{milk,bread,jam}	2
{milk,bread,butter}	2

Output : Frequent ItemSet

Table 1.2 Sample Frequent Itemsets

Here, the transaction database containing the various customer basket items is taken as the input. By subsequent join and prune steps the algorithm generates the frequent itemsets (i.e) the itemsets which have support count greater than minimum support value 2.Hence the output would be {milk,bread,jam} and {milk,bread,butter}.

1.3.2 FP-GROWTH ALGORITHM

Another remarkable algorithm that eliminates the generation of candidate itemsets is the Frequent Pattern (FP) Growth algorithm. The algorithm builds prefix trees from the target transaction database by performing exactly two scans of the database: one to find the frequent items and one to actually build the prefix tree. In order to determine all frequent itemsets, the prefix tree is traversed in a bottom-up manner.

Figure 1.2 shows the FP Tree for the given transaction database:

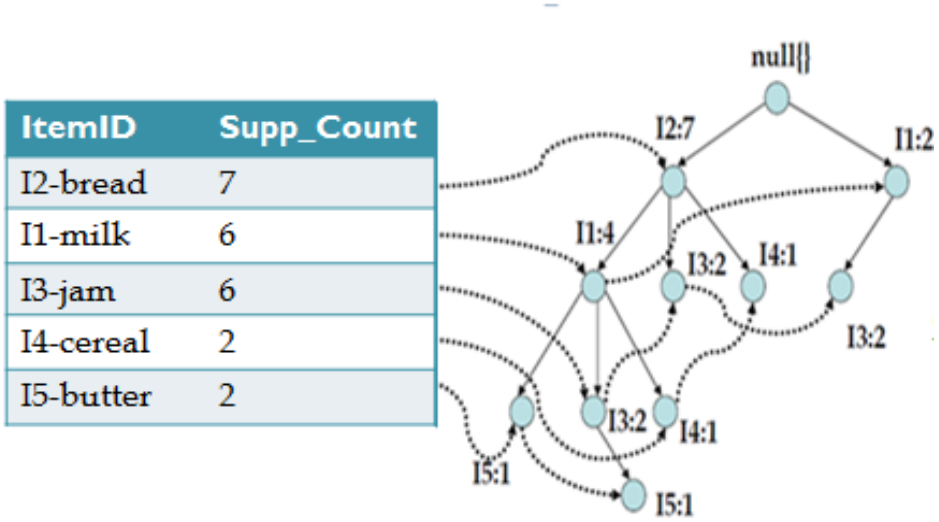


Figure 1.2 FP Growth Tree

Although FP-Growth substantially reduces the search cost, it suffers from the drawback of using a complex data structure (FP-Tree).With massive amounts of data, it is sometimes unrealistic to construct main memory based FP-Tree.

1.4 PURVIEW OF THE PROJECT

The discovery of interesting correlation relationships among huge amounts of business transaction records can help in many business decision-making processes. Its purview includes Market Basket Analysis and other similar applications such as detection of adverse drug reactions in health care data, credit card/banking system, business analysis, consumer preference analysis, web page frequentation analysis, store layout, product clustering, cross marketing and catalog design.

CHAPTER 2

LITERATURE REVIEW

Association Rule Mining (ARM) refers to the problem of identifying all relations between the elements of a given data set so that the presence of a given element/set of elements implies the presence of another element/set of elements.

The first notable algorithm to efficiently solve the issue of frequent itemset mining is the Apriori algorithm (1994) proposed by R. Agrawal and R. Srikant which follows candidate generation approach.

Ulterior attempts were made to develop better and faster algorithms. One such remarkable algorithm is FP-Growth which completely eliminates the candidate generation step.

[1] Alexander Archip and Mitica Craus ,'The Fast Itemset Miner: A detailed analysis of the Candidate Generation Stages' International Conference on System Theory Control and Computing (ICSTCC),Oct 2011.

This paper proposes a better and faster way of generating frequent itemsets by way of splitting the database into intervals and then mining the transactional database. This approach focuses on increasing size of the interval that candidates belong to, rather than increasing the size of the candidates by one in each iteration. Moreover, such an approach also favors a better support counting method, which greatly impacts on the overall time response of the FIM algorithm.

[2] J. Han and M. Kamber, Data Mining - Concepts and Techniques.

Morgan Kaufman Publications, 2006.

J.Han and M.Kamber outlines an approach for improving Apriori algorithm by way of '*Partitioning method*'. A partitioning technique, requires just two database scans to mine the frequent itemsets. It consists of two phases. In Phase I, the algorithm subdivides the transactions of *D* into *n* non-overlapping partitions. If the minimum support threshold for transactions in *D* is *min sup*, then the minimum support count for a partition is *min sup* , *the number of transactions in that partition*. For each partition, all frequent itemsets within the partition are found. These are referred to as local frequent itemsets. The procedure employs a special data structure that, for each itemset, records the TIDs of the transactions containing the items in the itemset. This allows it to find all of the local frequent *k*-itemsets, for *k* = 1, 2, : : : , in just one scan of the database.

[3] Wanjun Yu,Xiaochun Wang, Fangyi Wang, Erkang Wang, Bowen Chen,
'The research of improved apriori algorithm for mining association rules',11th IEEE International Conference on Communication Technology, ICCT 2008.

This paper proposes a novel algorithm called reduced apriori algorithm with tag (RAAT), which reduces one redundant pruning operations of C2. If the number of frequent 1-itemsets is n, then the number of connected candidate 2-itemsets is C_{n2}, while pruning operations C_{n2}. The novel algorithm decreases pruning operations of candidate 2-itemsets, thereby saving time and increasing efficiency. For the bottleneck: poor efficiency of counting support, RAAT optimizes subset operation, through the transaction tag to speed up support calculations.

CHAPTER 3

PROBLEM DEFINITION

The major objective of the project is to propose a new and efficient means of determining candidate itemsets using the algorithm – a Fast Itemset Miner (FIM). The project exploits both the Apriori Property and the particularities of FIM algorithm to avoid any unnecessary candidate validation steps and to avoid generating too many candidates. The FIM algorithm is proved to be a good enhancement over Apriori and FP growth algorithms.

Fast Itemset Miner (FIM) is developed to uncover interesting itemsets from a large database. It provides a novel approach for determining valid candidate itemsets. It favors a better support counting method which greatly reduces the time complexity required to mine the frequent itemsets. The efficiency is improved by not generating too many candidate itemsets and by way of partitioning the transactional database into several intervals. The time complexity of FIM algorithm is compared with that of Apriori algorithm. It shall be found that FIM outperforms Apriori in terms of time complexity.

CHAPTER 4

REQUIREMENTS SPECIFICATION

4.1 HARDWARE REQUIREMENTS

The project is intended to mine frequent itemsets in a faster way. Hence it is implemented as a desktop application. It requires a minimum of 1 GB RAM space with NTFS file system partition.

4.2 SOFTWARE REQUIREMENTS

4.2.1 Development Platform

The project is developed on Windows 7 platform. Windows 7 is one of the releases of Microsoft Windows, a series of operating systems produced by Microsoft for use on personal computers, including home and business desktops, laptops; netbooks, tablet PCs, and media center PCs.

4.2.1 Implementation Language

The project is implemented using **Java**. Java, being platform-independent the project is compatible with any other OS. Java is a general-purpose, concurrent, class-based, object-oriented language that is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that code that runs on one platform does not need to be recompiled to run on another.

4.2.2 Integrated Development Environment

NetBeans refers to both a platform framework for Java desktop applications, and an integrated development environment (IDE) for developing with Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala, Clojure and much more (for a complete overview visit the website of netbeans).

The NetBeans IDE is written in Java and runs everywhere where a JVM is installed, including Windows, Mac OS, Linux, and Solaris. A JDK is required for Java development. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans platform (including the NetBeans IDE) can be extended by third party developers.

4.2.3 Backend Used

Transactional Database that is used as the dataset for mining is stored in **MS Access**. Microsoft Office Access, previously known as Microsoft Access, is a database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools. It is a member of the Microsoft Office suite of applications, included in the Professional and higher editions.MS Access stores all database tables, queries, forms, reports, macros, and modules in the Access Jet database as a single file.

4.3 FUNCTIONAL REQUIREMENTS

REQ-1: The project is developed to mine the transactional database stored in MS Access.

REQ-2: Additional libraries such as Weka, jfreechart, jcommon and jmf jar files were required.

REQ-3: Frequent itemsets are to be generated using Apriori and FIM algorithm with FIM being more efficient.

4.4 PERFORMANCE REQUIREMENTS

PE-1: The application shall generate frequently purchased items based on the specified support count.

PE-2: The frequent itemsets are generated based on Apriori algorithm and FIM algorithm.

PE-3: FIM shall be more efficient than Apriori in terms of time complexity.

CHAPTER 5

DESIGN PROCESS

5.1 ARCHITECTURE DIAGRAM

The architecture proposed for implementing Fast Itemset Miner(FIM) is shown in Fig.5.1

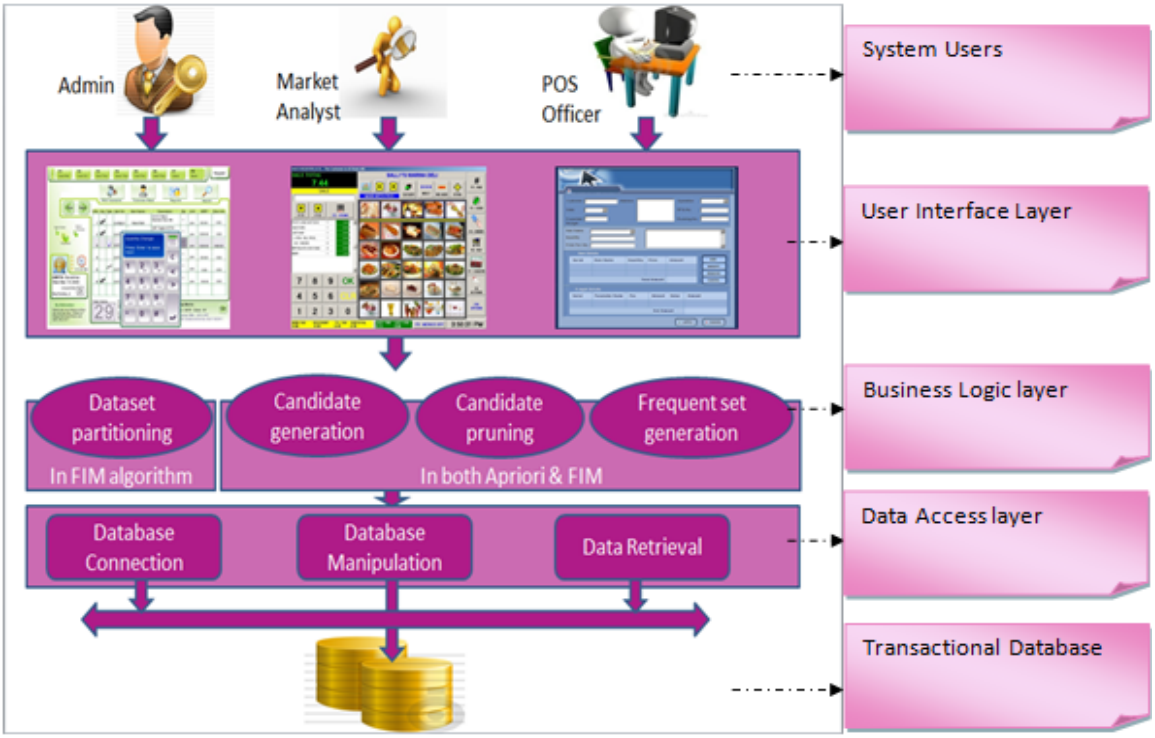


Figure 5.1 Overall Architecture

The architecture consists of 4 major layers namely User Interface, Business Logic, Data access and the Transaction database

5.5.1 User Interface Layer

The users of the system can be categorized into three groups such as administrators, Point Of Sale officers and market analysts. The administrators use the system to make necessary database modifications such as addition/deletion of transactional items. The

POS officers are allowed to enter the transactions into the database. The market analyst analyses the transactional database and discovers interesting patterns. The system provides a user-friendly interface in order that the analyst mines huge data tombs and extracts golden nuggets of knowledge.

5.5.2 Business Logic Layer

The frequent Itemset mining process consists of calculation of support count, generation of candidate itemsets, candidate pruning steps. The business logic module consists of implementation of the above steps.

5.5.3 Data Access Layer

Database access steps such as database connection/ manipulation/ retrieval are implemented here. The data access logic with the drivers needed for database connectivity is present here.

5.5.4 Transactional Database

The dataset that is the input for mining frequent itemsets is stored in the database. The dataset may consist of market basket data, health care data, traffic data etc.., based on the area of analysis.

5.2 PROJECT MODULES

5.2.1 Data Collection And Analysis

i)Input: Minimum of 1 lakh transactions in a supermarket with approx. 1000 items with timestamps.

ii)Output: Preprocessed dataset, ready for data mining.

A preprocessing stage is required to determine all frequent items, trim the transactions and sort the new transaction dataset descending with respect to an items support threshold.

5.2.2 Apriori Algorithm Implementation

i)Inputs:

- D - a database of transactions
- min support count - the minimum support count threshold.

ii)Output: L, frequent itemsets in D.

The first notable algorithm to efficiently solve the issue of frequent itemset mining is the Apriori algorithm. The algorithm performs multiple scans over the target transaction database. The first scan determines the frequent items with respect to the minimum support limit given by the user. All subsequent scans involve two stages:

- Join stage – All possible k-itemset candidates are identified based on the frequent itemsets found in the corresponding previous scan.
- Prune stage - The support for each candidate is then computed in the second stage and all candidates that do not meet the minimum support value are removed.

TID	*List of item_IDs*
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3

Table 5.1 Example Transaction Database

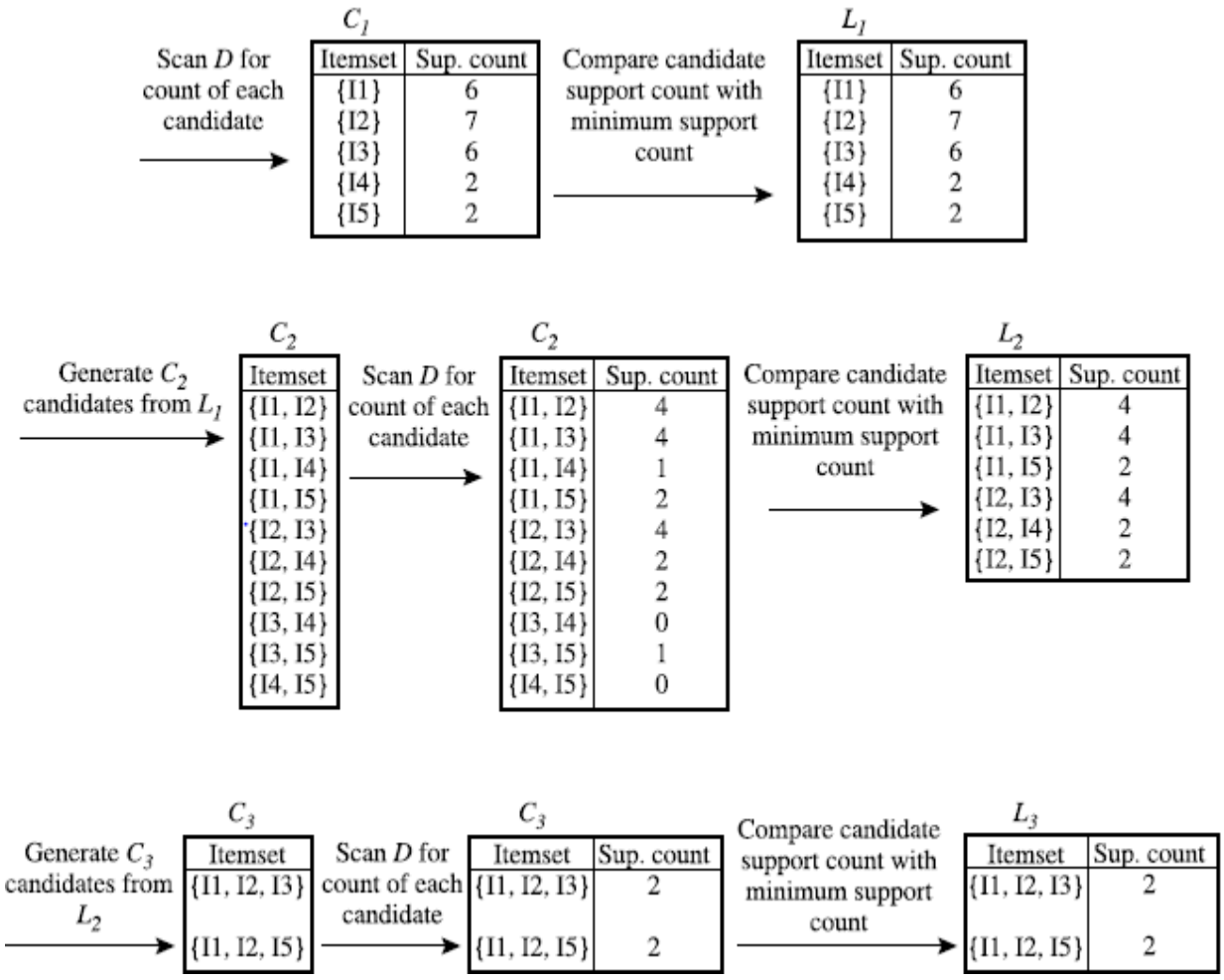


Figure 5.2 Iterative Levels of Apriori Algorithm

The key aspect of this algorithm is the efficient generation of candidates using the Apriori property - *'If an itemset is frequent then all of its subsets must also be frequent'*

5.2.3 FIM Algorithm Implementation

i)Input:

- D - a database of transactions
- min support count - the minimum support count threshold.

ii)Output: $F_{i,j}$, frequent itemsets in D.

Fast Itemset Miner (FIM) algorithm avoids any unnecessary candidate validation steps and also prevents generation of too many candidates. The following figure 5.3 shows the inputs, processing steps and the output of FIM algorithm.

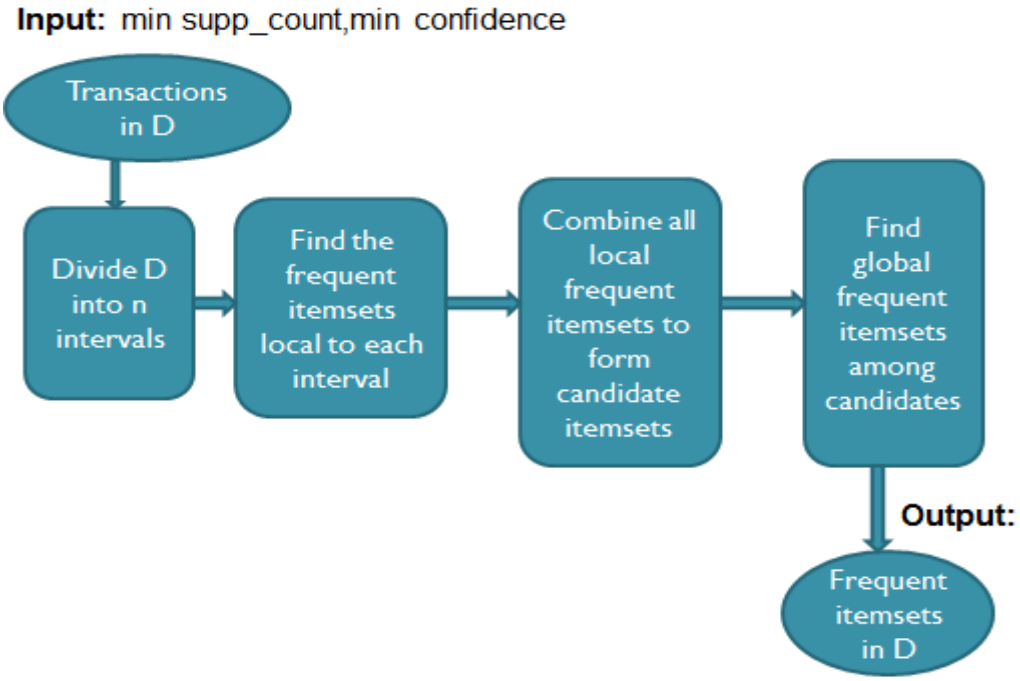


Figure 5.3 Process steps in FIM Algorithm

The algorithm subdivides the transactions in D into 'n' intervals. For each interval, all frequent itemsets within the partition are found. Global candidate itemsets can then be generated based on: Any itemset that is potentially frequent with respect to D must occur as a frequent itemset in at least one of the partitions. Now, the actual support count for each candidate is assessed to determine the global frequent itemsets.

5.2.4 Time Complexity Comparison

i)Input: Time complexities of the algorithms.

ii)Output: Graphical comparison of the time complexities.

The Association Rule Mining algorithms such as Apriori and FIM are implemented and their time complexities are compared.

5.3 HIPO DIAGRAM

The Hierarchical Input Process Output diagram for the project is shown in the fig.5.4. HIPO diagram is a "popular systems analysis design aid and documentation technique" for representing the modules of a system as a hierarchy and for documenting each module. It is used to develop requirements, construct the design, and support implementation of an expert system to demonstrate automated rendezvous. The overall design of the system is documented using HIPO charts or structure charts.

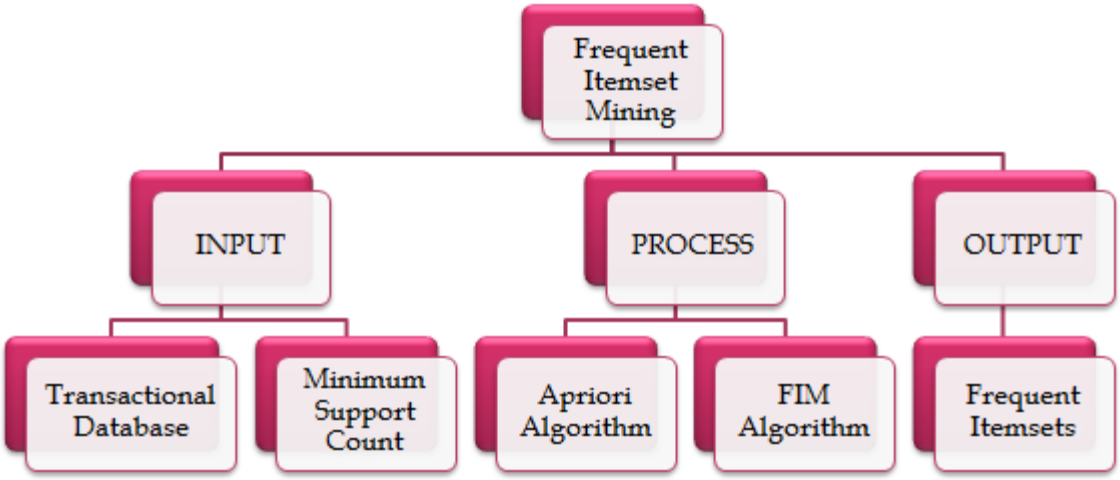


Figure 5.4 HIPO Diagram

As shown in figure 5.4, the input for the project constitutes the Transactional Database D and the minimum support count. The two major algorithms are implemented to generate frequent itemsets and are analyzed in terms of time complexity.

5.4 PROGRAM DESIGN LANGUAGE

5.4.1 Apriori Algorithm

Input:D, a database of transactions ; min sup, the minimum support count threshold.

Output:L, frequent itemsets in D.

Method:

(1) $L1$ = find frequent 1-itemsets(D);

(2) for ($k = 2$; $L_{k-1} \neq \phi$; k++) {

(3) C_k = apriori gen(L_{k-1});

(4) for each transaction $t \in D$ { // scan D for counts

(5) C_t = subset(C_k, t); // get the subsets of t that are candidates

(6) for each candidate $c \in C_t$

(7) c.count++;

(8) }

(9) $L_k = \{c \in C_k \mid c.count >= min\_sup\}$

(10) }

(11) return $L = \cup_k L_k$;

procedure **apriori_gen**(Lk-1:frequent (k-1)-itemsets)

(1) for each itemset $l1 \in L_{k-1}$

(2) for each itemset $l2 \in L_{k-1}$

(3) if ($l1[1] = l2[1]$)^($l1[2] = l2[2]$)^..^($l1[k-2] = l2[k-2]$)^$l1[k-1] < l2[k-1]$then{

(4) $c = l1 \bowtie l2$; // join step: generate candidates

(5) if has infrequent subset(c, L_{k-1}) then

(6) delete c; // prune step: remove unfruitful candidate

(7) else add c to C_k;

(8) }

(9) return C_k;

procedure **has_infrequent_subset**(c: candidate k-itemset;Lk-1 : frequent (k-1)-itemsets);

(1) for each (k-1)-subset s of c

(2) if s $\notin$ Lk-1 then

(3) return TRUE;

(4) return FALSE

Step 1 of Apriori algorithm finds the frequent 1-itemsets, L1. In steps 2 to 10, Lk-1 is used to generate candidates Ck in order to find Lk for k>=2. The apriori_gen procedure generates the candidates and then uses the Apriori property to eliminate those having a subset that is not frequent (step 3).Once all of the candidates have been generated, the database is scanned (step 4). For each transaction, a subset function is used to find all subsets of the transaction that are candidates (step 5), and the count for each of these candidates is accumulated (steps 6 and 7). Finally, all of those candidates satisfying minimum support (step 9) form the set of frequent itemsets, L (step 11).The apriori_gen procedure performs two kinds of actions, namely, join and prune. In the join component, Lk-1 is joined with Lk-1 to generate potential candidates (steps 1 to 4). The prune component (steps 5 to 7) employs the Apriori property to remove candidates that have a subset that is not frequent.

5.5.2 FIM Algorithm

Input: *D*, a database of transactions ; *min sup*, the minimum support count threshold.

Output: $F_{i,j}$, frequent itemsets in *D*.

Method:

(1) for k = 1 to n-1 do

(2) for all i, j : $i \in 1,....,n-k$; $j = i+k$ do

(3) $F_{i,j} \leftarrow F_{i,j-1} \cup F_{i+1,j}$

(4) $C_{i,j} = \{ X \cup Y \mid (X \in F_{i,j-1} \wedge i \in X) \wedge (Y \in F_{i+1,j} \wedge j \in X) \}$

(5)for all transactions $t \subset D_i$ do

(6) { D_i denotes the subset of transactions in D that include frequent item i }

(7) $C_t \leftarrow$ subset($C_{i,j}$, t)

{ determine all candidates in $C_{i,j}$ that are included in transaction t }

(8) for all candidates $c \in C_t$ do

(9) c.support++

(10) end for

(11)end for

(12) $F_{i,j} = F_{i,j} \cup \{c \in C_{i,j} \mid c.support >= min\_supp\}$

(13) $F_k \leftarrow F_k \cup F_{i,j}$

(14) end for

(15)end for

FIM algorithm works as follows:

Assuming n frequent items, in the first stage of the algorithm, new candidates will be generated based on the following subsets: {1,2},{2,3},…..,{n-1,n}. For the next stage, candidates will be determined from items belonging to {1,2,3},{2,3,4},…..,{n-2,n-1,n}. This same procedure continues until, finally, new candidate itemsets will include items from the interval {1,2,….,n}. One of the key advantages of this approach is that the FIM algorithm does not need to scan the whole transaction database in order to determine the support for any given valid candidate. In order to determine the support for all candidates included in $C_{i,j}$, the algorithm must scan only the transactions that include the item i.

CHAPTER 6

IMPLEMENTATION

The project is implemented using a database of 1 lakh of supermarket transactions and a minimum of 1000 items. The dataset was collected and cleaned to remove null values and duplicates. The preprocessed data was used to mine the frequent itemsets.

Apriori algorithm was implemented based on Apriori property. As it consists of join and prune steps, several iterative levels were implemented to reach at the frequent itemsets. Since a large database was to be mined, it took several minutes to execute for the specified support counts.

FIM algorithm on the other hand, outperformed Apriori algorithm and is proved to be faster on large databases. FIM algorithm which is based on candidate generation approach was implemented by partitioning the database into several intervals, generating local frequent itemsets and then finally combining local itemsets into global frequent itemsets.

On analyzing the time complexities using graphs for a given support count, FIM produced the results faster when compared to Apriori algorithm.

CHAPTER 7

INPUTS AND OUTCOMES

The various inputs and the corresponding outcomes of Apriori Algorithm are tabulated below:

SI.NO	INPUT: MINIMUM SUPPORT COUNT	OUTPUT: FREQUENT ITEMSETS
1	**3**	Marie gold,Miranda,} {Marie gold,Manna health mix,} {Marie gold,Good night liquid mosquito vapourizer,} {Marie gold,White horse chalkpiece,} {Marie gold,Modern Bread,} {Marie gold,Annapurna atta,} {Miranda,Manna health mix,} {Miranda,Good night liquid mosquito vapourizer,} {Miranda,White horse chalkpiece,} {Miranda,Modern Bread,} {Miranda,Annapurna atta,} {Manna health mix,Good night liquid mosquito vapourizer,} {Manna health mix,White horse chalkpiece,} {Manna health mix,Modern Bread,}

		{Manna health mix,Annapurna atta,} {Good night liquid mosquito vapourizer,White horse chalkpiece,} {Good night liquid mosquito vapourizer,Modern Bread,} {Good night liquid mosquito vapourizer,Annapurna atta,} {White horse chalkpiece,Modern Bread,} {White horse chalkpiece,Annapurna atta,} {Modern Bread,Annapurna atta,} {Marie gold,Miranda,Manna health mix,} {Marie gold,Miranda,Good night liquid mosquito vapourizer,} {Marie gold,Miranda,White horse chalkpiece,} {Marie gold,Miranda,Modern Bread,} {Marie gold,Miranda,Annapurna atta,} {Marie gold,Manna health mix,Good night liquid mosquito vapourizer,} {Marie gold,Manna health mix,White horse chalkpiece,} {Marie gold,Manna health mix,Modern Bread,} {Marie gold,Manna health mix,Annapurna atta,} {Marie gold,Good night liquid mosquito vapourizer,White horse chalkpiece,} {Marie gold,Good night liquid mosquito vapourizer,Modern Bread,} {Marie gold,Good night liquid mosquito vapourizer,Annapurna atta,}

		{Marie gold,White horse chalkpiece,Modern Bread,} {Marie gold,White horse chalkpiece,Annapurna atta,} {Marie gold,Modern Bread,Annapurna atta,} {Miranda,Manna health mix,Good night liquid mosquito vapourizer,} {Miranda,Manna health mix,White horse chalkpiece,} {Miranda,Manna health mix,Modern Bread,} {Miranda,Manna health mix,Annapurna atta,} {Miranda,Good night liquid mosquito vapourizer,White horse chalkpiece,} {Miranda,Good night liquid mosquito vapourizer,Modern Bread,} {Miranda,Good night liquid mosquito vapourizer,Annapurna atta,} {Miranda,White horse chalkpiece,Modern Bread,} {Miranda,White horse chalkpiece,Annapurna atta,} {Miranda,Modern Bread,Annapurna atta,} {Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,} {Manna health mix,Good night liquid mosquito vapourizer,Modern Bread,} {Manna health mix,Good night liquid mosquito vapourizer,Annapurna atta,} {Manna health mix,White horse chalkpiece,Modern Bread,}

		{Manna health mix,White horse chalkpiece,Annapurna atta,} {Manna health mix,Modern Bread,Annapurna atta,} {Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,} {Good night liquid mosquito vapourizer,White horse chalkpiece,Annapurna atta,} {Good night liquid mosquito vapourizer,Modern Bread,Annapurna atta,} {White horse chalkpiece,Modern Bread,Annapurna atta,} {Marie gold,Miranda,Manna health mix,Good night liquid mosquito vapourizer,} {Marie gold,Miranda,Manna health mix,White horse chalkpiece,} {Marie gold,Miranda,Manna health mix,Modern Bread,} {Marie gold,Miranda,Manna health mix,Annapurna atta,} {Marie gold,Miranda,Good night liquid mosquito vapourizer,White horse chalkpiece,} {Marie gold,Miranda,Good night liquid mosquito vapourizer,Modern Bread,} {Marie gold,Miranda,Good night liquid mosquito vapourizer,Annapurna atta,} {Marie gold,Miranda,White horse chalkpiece,Modern Bread,} {Marie gold,Miranda,White horse

		chalkpiece,Annapurna atta,} {Marie gold,Miranda,Modern Bread,Annapurna atta,} {Marie gold,Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,} {Marie gold,Manna health mix,Good night liquid mosquito vapourizer,Modern Bread,} {Marie gold,Manna health mix,Good night liquid mosquito vapourizer,Annapurna atta,} {Marie gold,Manna health mix,White horse chalkpiece,Modern Bread,} {Marie gold,Manna health mix,White horse chalkpiece,Annapurna atta,} {Marie gold,Manna health mix,Modern Bread,Annapurna atta,} {Marie gold,Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,} {Marie gold,Good night liquid mosquito vapourizer,White horse chalkpiece,Annapurna atta,} {Marie gold,Good night liquid mosquito vapourizer,Modern Bread,Annapurna atta,} {Marie gold,White horse chalkpiece,Modern Bread,Annapurna atta,} {Miranda,Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,} {Miranda,Manna health mix,Good night liquid mosquito vapourizer,Modern Bread,} {Miranda,Manna health mix,Good night liquid

		mosquito vapourizer,Annapurna atta,} {Miranda,Manna health mix,White horse chalkpiece,Modern Bread,} {Miranda,Manna health mix,White horse chalkpiece,Annapurna atta,} {Miranda,Manna health mix,Modern Bread,Annapurna atta,} {Miranda,Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,} {Miranda,Good night liquid mosquito vapourizer,White horse chalkpiece,Annapurna atta,} {Miranda,Good night liquid mosquito vapourizer,Modern Bread,Annapurna atta,} {Miranda,White horse chalkpiece,Modern Bread,Annapurna atta,} {Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,} {Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,Annapurna atta,} {Manna health mix,Good night liquid mosquito vapourizer,Modern Bread,Annapurna atta,} {Manna health mix,White horse chalkpiece,Modern Bread,Annapurna atta,} {Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,Annapurna atta,} {Marie gold,Miranda,Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,}

		{Marie gold,Miranda,Manna health mix,Good night liquid mosquito vapourizer,Modern Bread,} {Marie gold,Miranda,Manna health mix,Good night liquid mosquito vapourizer,Annapurna atta,} {Marie gold,Miranda,Manna health mix,White horse chalkpiece,Modern Bread,} {Marie gold,Miranda,Manna health mix,White horse chalkpiece,Annapurna atta,} {Marie gold,Miranda,Manna health mix,Modern Bread,Annapurna atta,} {Marie gold,Miranda,Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,} {Marie gold,Miranda,Good night liquid mosquito vapourizer,White horse chalkpiece,Annapurna atta,} {Marie gold,Miranda,Good night liquid mosquito vapourizer,Modern Bread,Annapurna atta,} {Marie gold,Miranda,White horse chalkpiece,Modern Bread,Annapurna atta,} {Marie gold,Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,} {Marie gold,Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,Annapurna atta,} {Marie gold,Manna health mix,Good night liquid mosquito vapourizer,Modern Bread,Annapurna atta,} {Marie gold,Manna health mix,White horse

		chalkpiece,Modern Bread,Annapurna atta,} {Marie gold,Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,Annapurna atta,} {Miranda,Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,} {Miranda,Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,Annapurna atta,} {Miranda,Manna health mix,Good night liquid mosquito vapourizer,Modern Bread,Annapurna atta,} {Miranda,Manna health mix,White horse chalkpiece,Modern Bread,Annapurna atta,} {Miranda,Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,Annapurna atta,} {Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,Annapurna atta,} {Marie gold,Miranda,Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,} {Marie gold,Miranda,Manna health mix,Good night liquid mosquito vapourizer,White horse chalkpiece,Annapurna atta,} {Marie gold,Miranda,Manna health mix,Good night

		liquid mosquito vapourizer,Modern Bread,Annapurna atta,} {Marie gold,Miranda,Manna health mix,White horse chalkpiece,Modern Bread,Annapurna atta,} {Marie gold,Miranda,Good night liquid mosquito vapourizer,White horse chalkpiece,Modern Bread,Annapurna atta,}
2	5	{Manna health mix,Lipton green tea bags,} {Manna health mix,Good night liquid mosquito vapourizer,} {Manna health mix,Sakthi garam masala,} {Lipton green tea bags,Good night liquid mosquito vapourizer,} {Lipton green tea bags,Sakthi garam masala,} {Good night liquid mosquito vapourizer,Sakthi garam masala,} {Manna health mix,Lipton green tea bags,Good night liquid mosquito vapourizer,} {Manna health mix,Lipton green tea bags,Sakthi garam masala,} {Manna health mix,Good night liquid mosquito vapourizer,Sakthi garam masala,} {Lipton green tea bags,Good night liquid mosquito vapourizer,Sakthi garam masala,} {Manna health mix,Lipton green tea bags,Good night liquid mosquito vapourizer,Sakthi garam masala,}
3	8	{Manna health mix,Meera Herbal Shampoo,}

		{Manna health mix,Good night liquid mosquito vapourizer,} {Meera Herbal Shampoo,Good night liquid mosquito vapourizer,} {Manna health mix,Meera Herbal Shampoo,Good night liquid mosquito vapourizer,}

Table 7.1 Sample Apriori outputs

The various inputs and the corresponding outcomes of FIM Algorithm are tabulated below

SI.NO	**INPUT: MINIMUM SUPPORT COUNT**	**OUTPUT: FREQUENT ITEMSETS**
1	1	{Perk,Coffy-bite,} {Coffy-bite,Digestive,} {Digestive,Time pass,} {Time pass,True nice,} {True nice,krackers,} {krackers,ripple top wafer,} {ripple top wafer,Hippo,} {Hippo,Haldirams bhel puri,} {Natura sugarfree,Lion oats,} {Cashew Nuts,Lizol,}

		{Babool,Deo fresh,} {Sesa ,Gopuram turmeric paste,} {Fenugreek,pineapple,} {Betel nut,Tutty fruity,} {Tutty fruity,Royal basmati rice,} {Aachi ragi flour,Aachi chemba idiyappam powder,} {Toblerone,Galaxy,} {Fruitella,Milky way,} {Milky way,Snickers,} {Perk glucose,Marie gold,} {Lays,Haldirams sev puri,} {Kissan jam,Sil jam,} {Garnier fructis,Comfort fabric conditioner,} {Nivea,Emami gold turmeric skin cream,} {TNPL A4 sheets,Globe lock and key,} {Globe lock and key,Chima Scribbling pad,} {Krack cream,Archies diary,} {Archies diary,Roots combs,} {Haldirams gulab jamuns,Haldirams rasagollas,} {TEAM Mineral Water,Dry chillis,} {Aachi ghee rice paste, Shakthi ragi flour,} {Eclairs,Naturo,} {Naturo,Marie gold,} {Amul basundi,Amul mozzarella cheese,} {Amul mozzarella cheese,Amul kool,} {Taj mahal tea,Loreal paris,} {Robin Blue Washing Powder,Cuticura,}

		{Spinz,Sesa ,} {ambipur,revlon,}{Lakme nailpolish and lipstick,Lifebuoy handwash,} {Lifebuoy handwash,vaseline,} {vaseline,Moserbaer cd,} {Moserbaer cd,Camlin crayons,} {Camlin crayons,capsicum,} {capsicum,Daurala sugar cubes,} {Daurala sugar cubes,Annapoorna salt,} {Annapoorna salt,Poorna refined ricebran oil,} {Poorna refined ricebran oil,Pillsbury atta,} {Pillsbury atta,Aachi curry,}
2	2	{Garlic,Lettuce,} {Lettuce,Fortune refined oil,} {Marie gold,Kellogs honey loops,} {Mithai Mate,Amul basundi,} {Lacto Calamine Face wash,Cuticura,} {Odonil,Good night liquid mosquito vapourizer,} {Good night liquid mosquito vapourizer,Lifebuoy sanitizer,} {Modern Bread,Tutty fruity,} {Tutty fruity,Annapurna atta,} {MTR butter chakli,Manna health mix,} {Kohinoor basmati rice,Ajinomoto,} {grapes,Pista,} {Pista,Poorna refined ricebran oil,} {Poorna refined ricebran oil,Sakthi garam masala,} {Garlic,Green Peas,}

		{Green Peas,Knorr soupy noodles,} {Knorr soupy noodles,Lipton green tea bags,} {Lipton green tea bags,Sagar skimmed milk powder,} {Robin Blue Washing Powder,Johnson&Johnson,} {Poorna refined ricebran oil,Aachi egg masala,} {Aachi egg masala,Aachi mutton masala,} {MTR bisebela bath,Aachi biriyani rice paste,} {Top-ramen,Peacock papad,} {Peacock papad,Sakthi garam masala,} {Milky way,Rajaram sesame balls,} {Fiama De Wills Shampoo,Fiama de wills soap,} {Meera Herbal Shampoo,Mr.muscle cleaning Liquid,} {Good night liquid mosquito vapourizer,ambipur,} {Rice flour,Peacock papad,} {Cashew Nuts,Tutty fruity,} {Perk glucose,Sil jam,} {Sil jam,Amul Cheese,}
3	3	{Marie gold, Good night liquid vapourizer}

Table 7.2 Sample FIM outputs

CHAPTER 8

PERFORMANCE AND CONTRIBUTIONS

8.1 METRIC USED-TIME COMPLEXITY

The project is measured against the time taken to generate frequent itemsets by Apriori algorithm and FIM algorithm separately. For various support counts, the time complexity to generate frequent itemsets are calculated and graphically represented.

MIN. SUPPORT COUNT (min. no. of transactions containing the itemsets)	APRIORI ALGORITHM TIME (ms)	FIM ALGORITHM TIME (ms)
1	6391	3937
2	5531	2352
3	4886	1647
4	3344	1190

Table 8.1 Time Complexity Comparison

8.2 GRAPHICAL REPRESENTATION

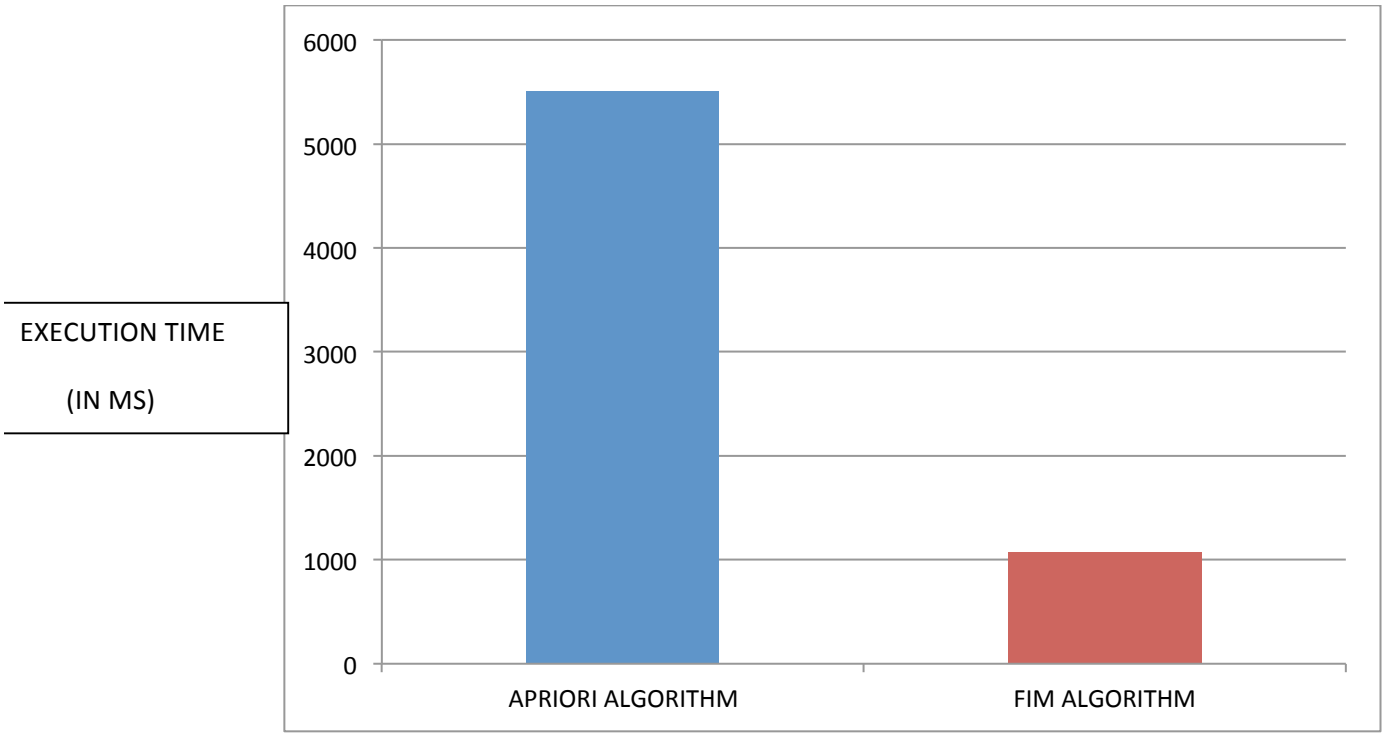


Figure 8.1 Graphical Representation of Time Complexity

8.3 CONTRIBUTION OF WORK

1. Our work can be used to generate frequent itemsets in case of supermarket data. It can also be extended to discover correlated data items in the case of health care data, traffic data, credit card usage analysis, web page frequentation analysis and telephone usage analysis.

2. FIM is a novel approach in mining associations among large databases in a faster way. Hence time taken to mine the itemsets is greatly reduced.

CHAPTER 9

CONCLUSION

9.1 CONCLUSION & FUTURE WORK

Thus FIM is found to be a novel approach which addresses one of the important areas of data mining domain called '*Mining frequent patterns, associations and correlations*'. The project is implemented with 1 lakh supermarket transactions with a minimum of 1000 items. Frequent itemsets are generated successfully by using Apriori and FIM algorithms. It is evident that FIM surpasses Apriori in terms of time complexity.

Our immediate priority is to properly test this new approach, not only against Apriori, but also against faster algorithms such as FP-Growth. Favorable results would underline the fact that candidate based approaches should still be considered in addressing the frequent itemset mining problem. If properly handled, such approaches do have the advantage of eliminating unnecessary test cases. A second line of tests must address the parallel FIM algorithm. This new algorithm implies a simple, straight forward parallelization that may be easily adapted to either shared memory parallel machines, either message passing ones.

ANNEXURE

A.1 SOURCE CODE

//Candidates.java

```
public Candidates(int[] paramsupcount)
{
 supcount = Arrays.copyOf(paramsupcount, paramsupcount.length);
items=new int[supcount.length];
for(int i=0;i<supcount.length;i++)
{
items[i]=i;
}
}

public Candidates(int[] paramitems,int[] paramcounts)
{
items = Arrays.copyOf(paramitems, paramitems.length);
supcount = Arrays.copyOf(paramcounts, paramcounts.length);
}
public void printCandidates(){
System.out.println("Candidates:");
for(int i=0;i<supcount.length;i++)
{
System.out.println("Item:"+items[i]+"Support:"+supcount[i]);
}
}
}
```

//PruneItems.java

```
public PruneItemSets(Candidates candi,int min_supp)
{
candi.printCandidates();
System.out.println("Inside prune items consssssssssss");
candiinstance=new Candidates();
candiinstance=candi;
minsupinstance=min_supp;
}
public Candidates pruneItems()
{
int[] itemflag=new int[candiinstance.items.length];
int[] supflag=new int[candiinstance.supcount.length];
int[] pruneditems;
int[] prunedsupcount;
int count=0,h=0;
Candidates p=null;
for(int i=0;i<candiinstance.items.length;i++){
        if(candiinstance.supcount[i] < minsupinstance)
{
        //System.out.println("Failed"+candiinstance.items[i]);
}
else{
//System.out.println("Succeeeeeeeeeess:"+candiinstance.items[i]);
itemflag[i]=candiinstance.items[i];
supflag[i]=candiinstance.supcount[i];
}
}
```

```
for(int j=0;j<itemflag.length;j++)
{
if(itemflag[j]!=0 && supflag[j]!=0)
{
count++;
//System.out.println("ItemFlag:"+itemflag[j]+"FlagSup:"+supflag[j]);
}
}
pruneditems=new int[count];
prunedsupcount=new int[count];
for(int m=0;m<itemflag.length;m++){
if(itemflag[m]!=0 && supflag[m]!=0){
pruneditems[h]=itemflag[m];
prunedsupcount[h]=supflag[m];
h++;}
}
p=new Candidates(pruneditems,prunedsupcount);
return p;
}
}
```

//IntCombi.java

```
void comb(int... items) {
Arrays.sort(items);
for (int k = 1; k <= items.length; k++) {
kcomb(items, 0, k, new int[k]);
}
}
```

```
public void kcomb(int[] items, int n, int k, int[] arr) {
if (k == 0) {
fvcombi.addElement(Arrays.toString(arr));}
 else {
for (int i = n; i <= items.length - k; i++) {
arr[arr.length - k] = items[i];
kcomb(items, i + 1, k - 1, arr);
}
}
}
public void printcombi()
{
System.out.println("Combination Elements:");
for(Enumeration e=fvcombi.elements();e.hasMoreElements();){
System.out.println(e.nextElement().toString());
}
}
}
```

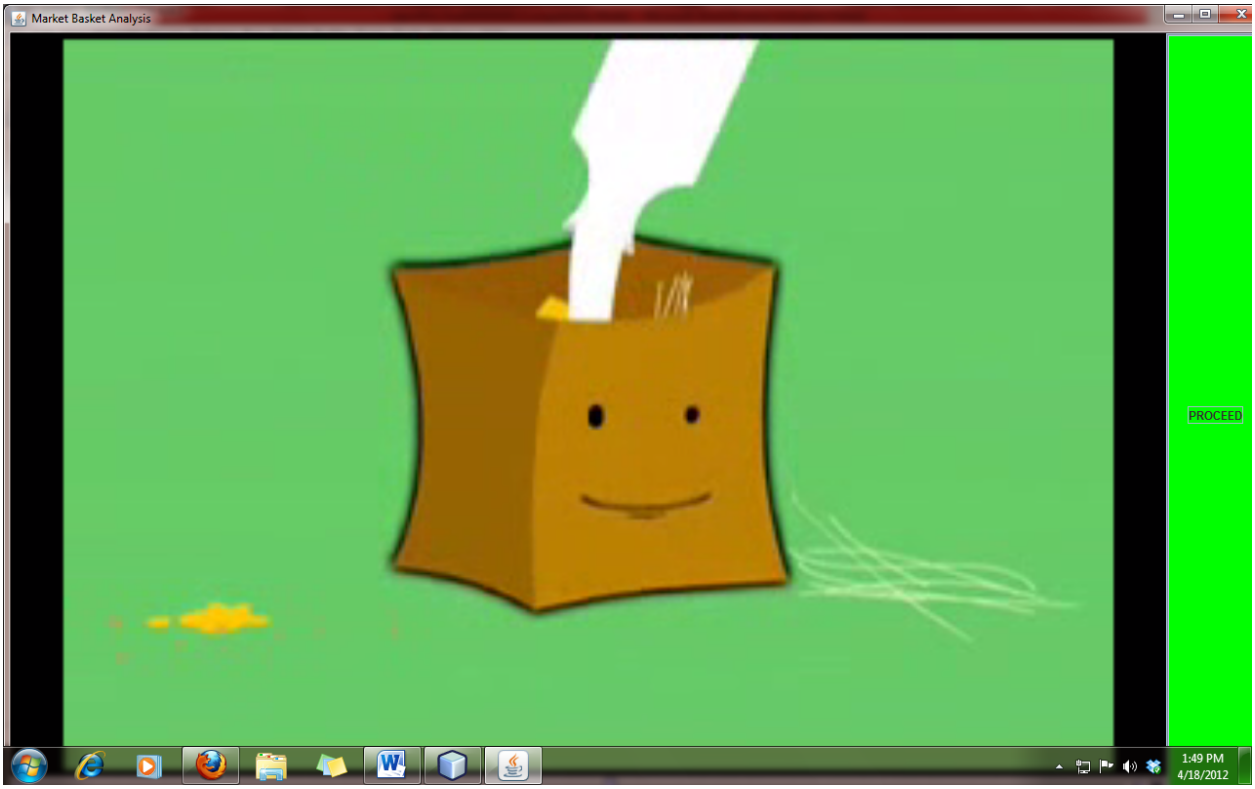
//Database partitioning in FIM.java

```
res = stm.executeQuery("select * from fimtrans where thours='"+strhrs+"'");
while(res.next())
{
id = res.getString("id");
trans = res.getString("trans");
time=res.getString("thours");
streachhourid[u][k]=new String(id);
System.out.println("u:"+u+"k:"+k+"streachhourid:"+streachhourid[u][k]);
Object translist=m.put(id,trans);
```

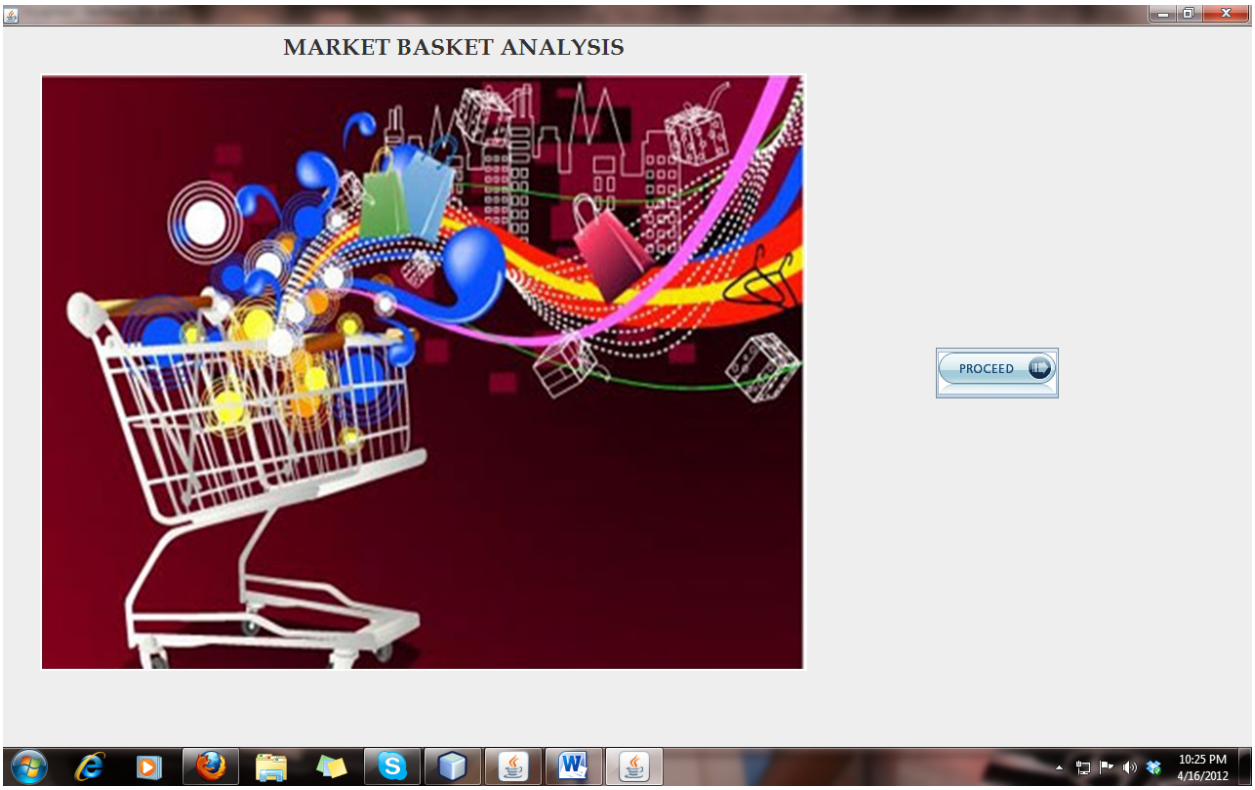
```
Object transtime=maptranstime.put(id,time);
transset=m.entrySet();
boolean a = transactions.add(m.values());
boolean o = translisttime.add(maptranstime.values());
k++;
}
arrcounter[u]=k;
k=0;u++;
ithours++;
}
} catch(Exception e) {
System.out.println("SQLerror "+ e);
}
lolmaps=new ArrayList<List>();
for(u=0;u<=13;u++)
{
for(k=0;k<=arrcounter[u];k++)
{
if(streachhourid[u][k]!=null)
{
System.out.println("u:" + u + "k:" + k + "val:" +
streachhourid[u][k]);//"+"stritems:"+stritemarray[itit]);
 Object put = maptranspartid.put(streachhourid[u][k], m.get(streachhourid[u][k]));
System.out.println("u:"+u+"maptranspartid:"+maptranspartid.values());
}
}
}
}
```

A.2 SCREEN SHOTS

MARKET BASKET ANALYSIS MAIN SCREEN

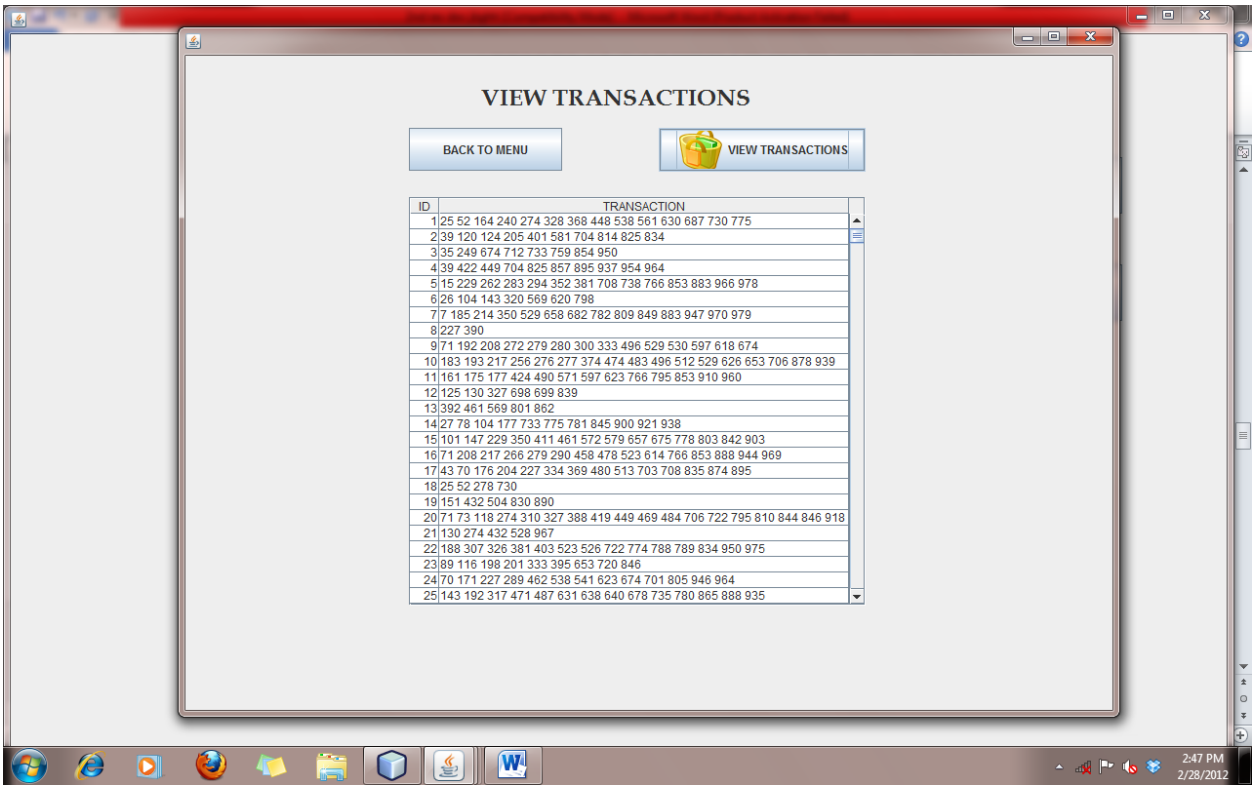


MARKET BASKET ANALYSIS MAIN SCREEN

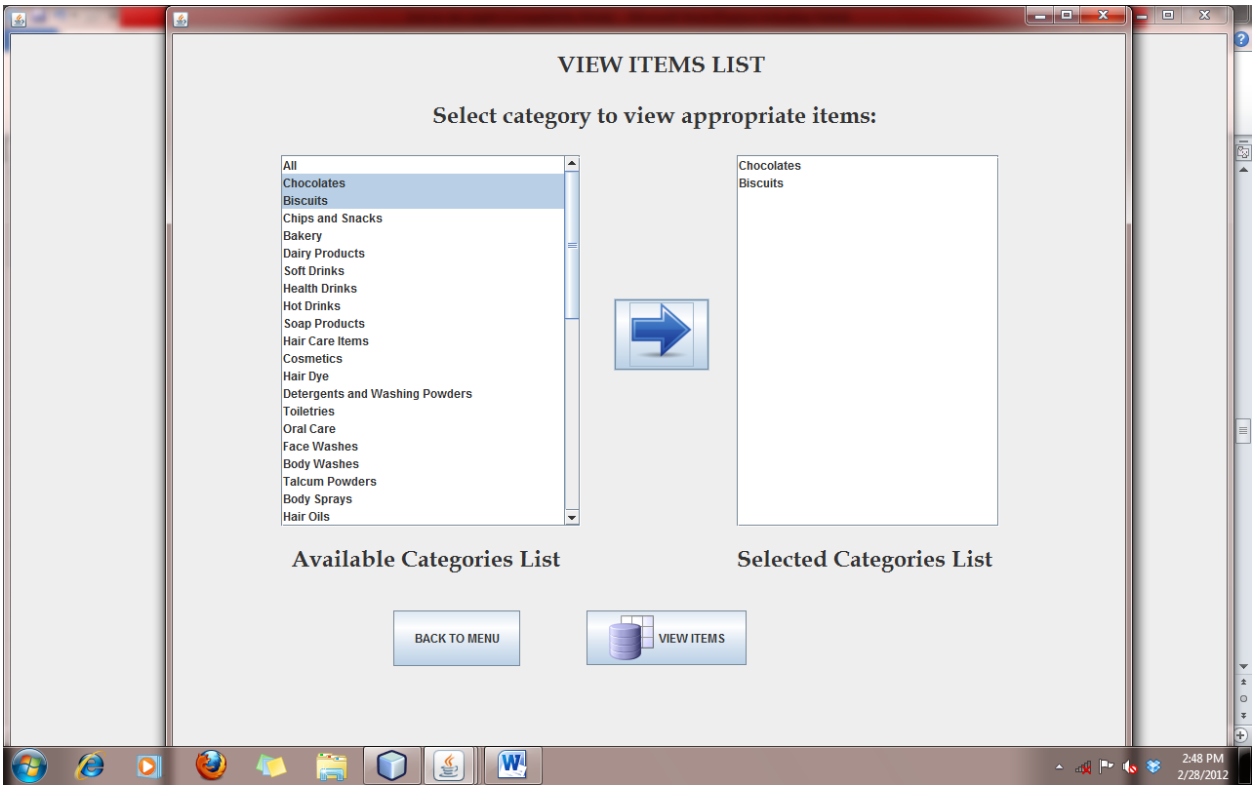


MENU OPTIONS SCREEN

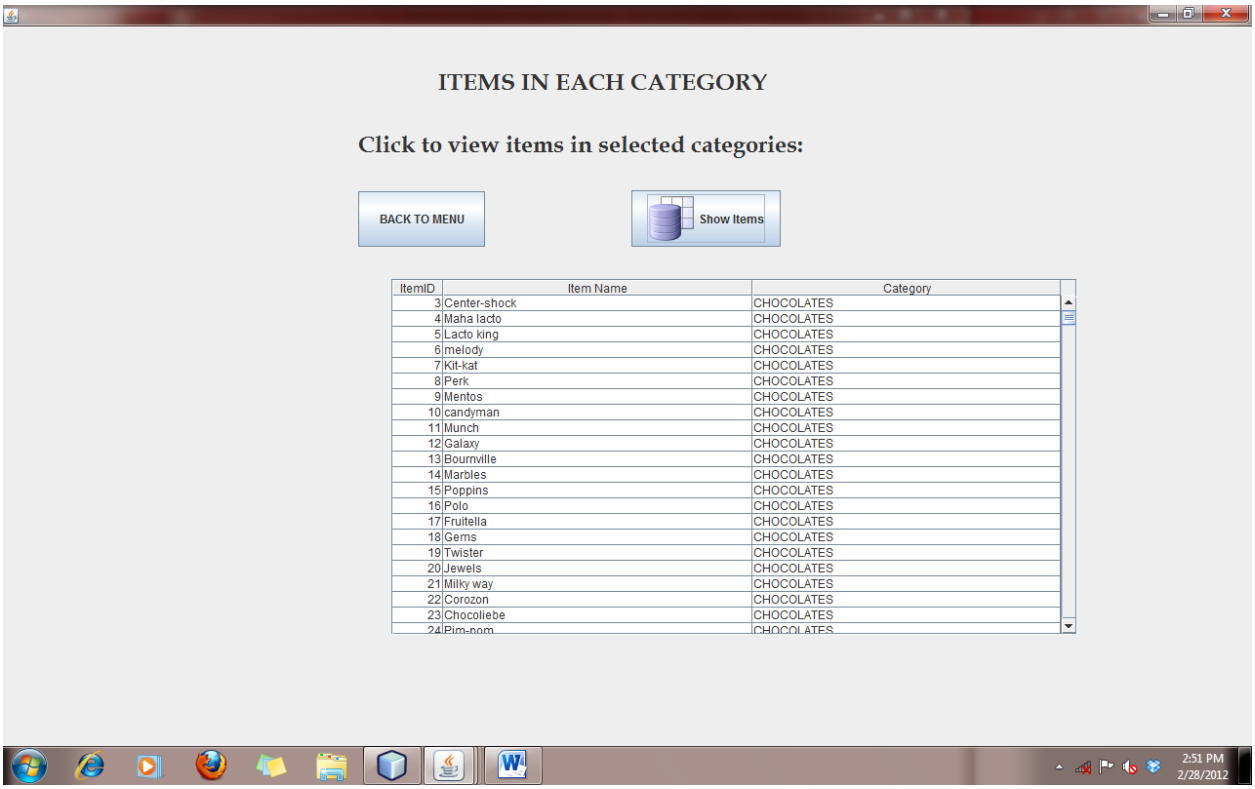
SCREEN TO VIEW SUPERMARKET TRANSACTIONS



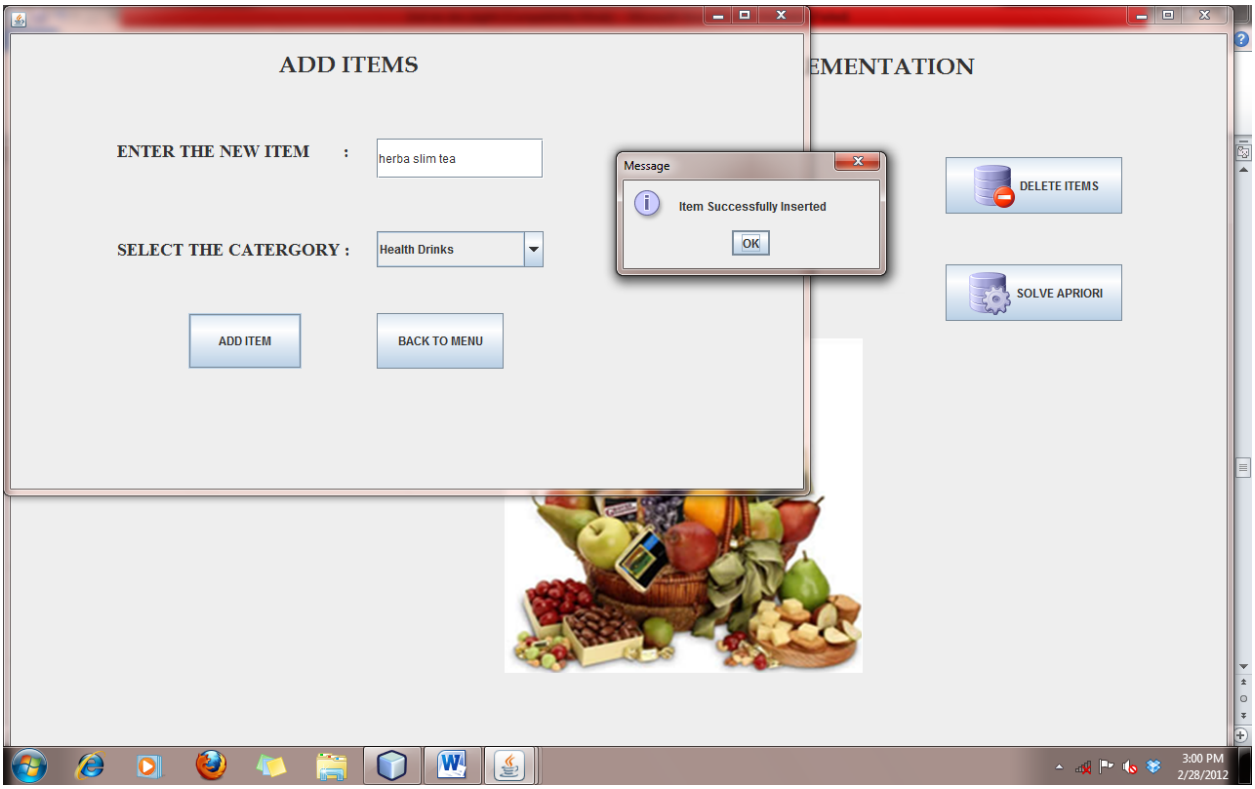
SCREEN TO SELECT CATEGORY TO VIEW ITEMS



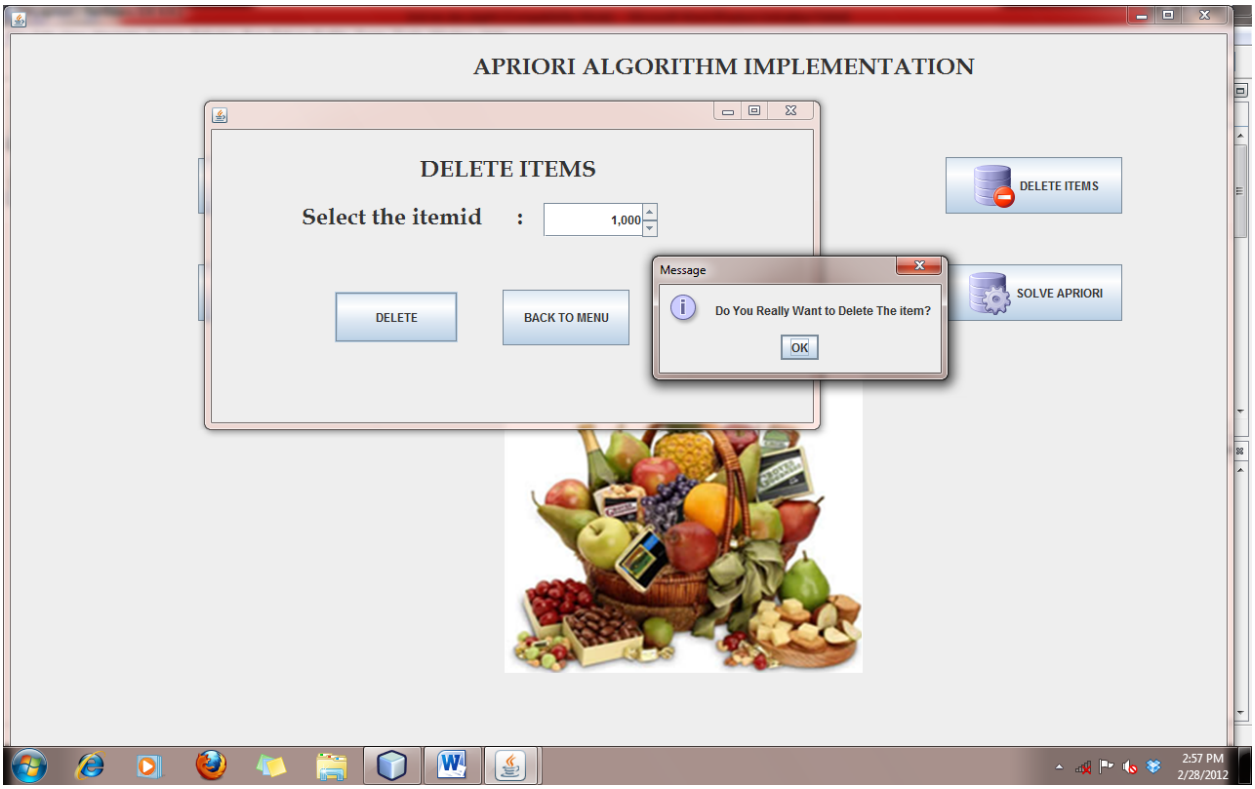
ITEMS LIST SCREEN



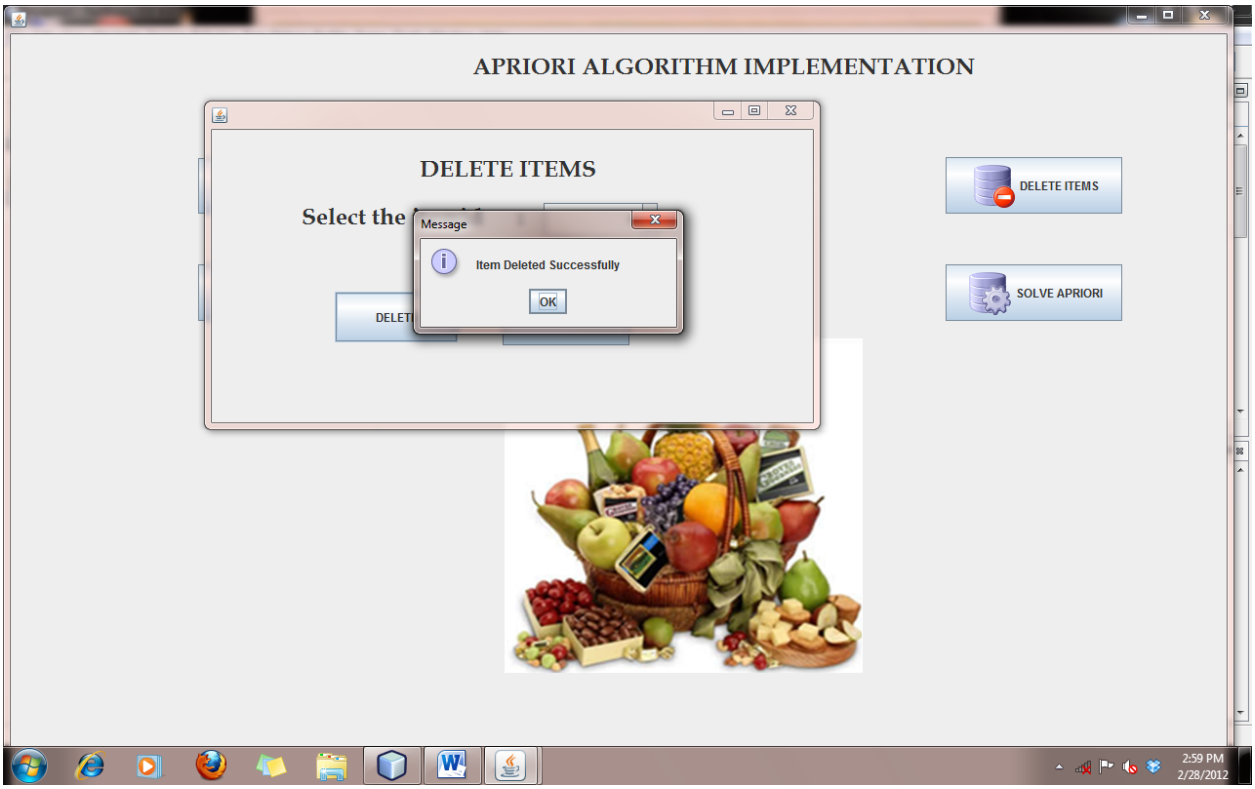
ADD ITEM SCREEN



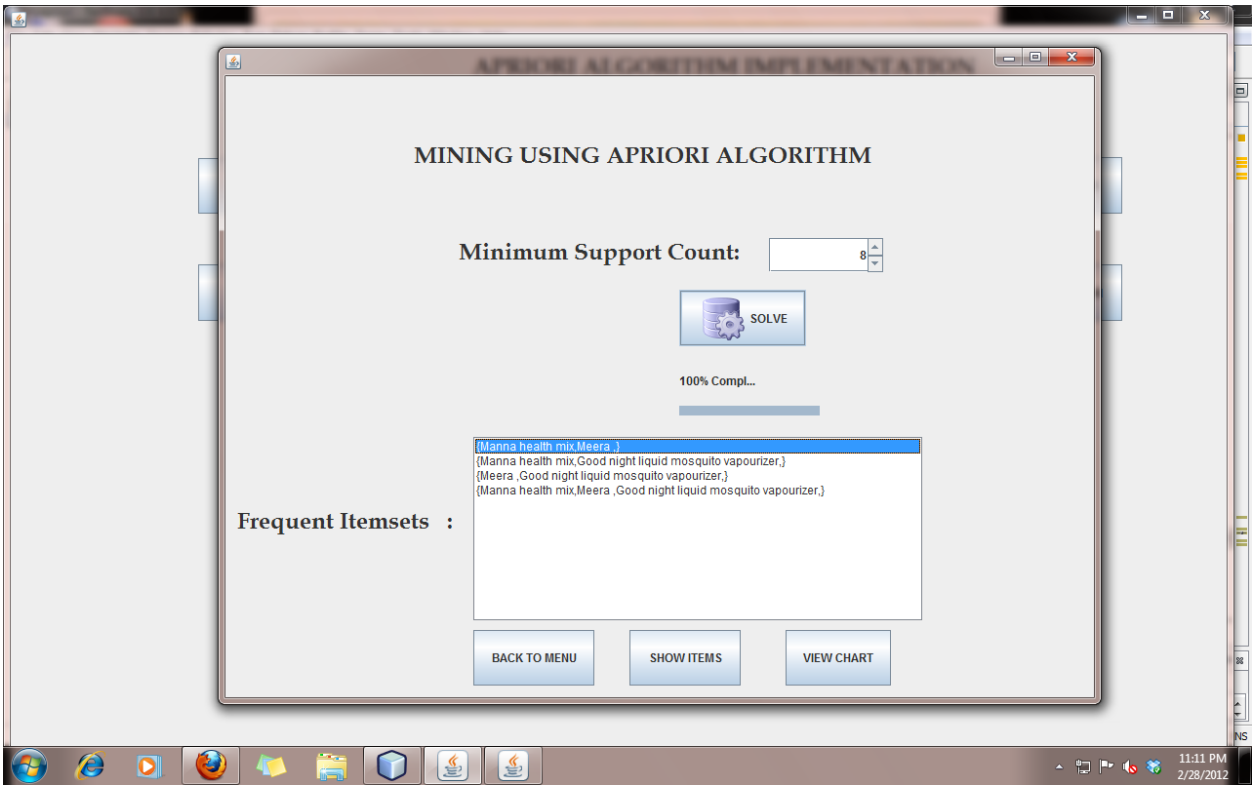
ITEM DELETION SCREEN



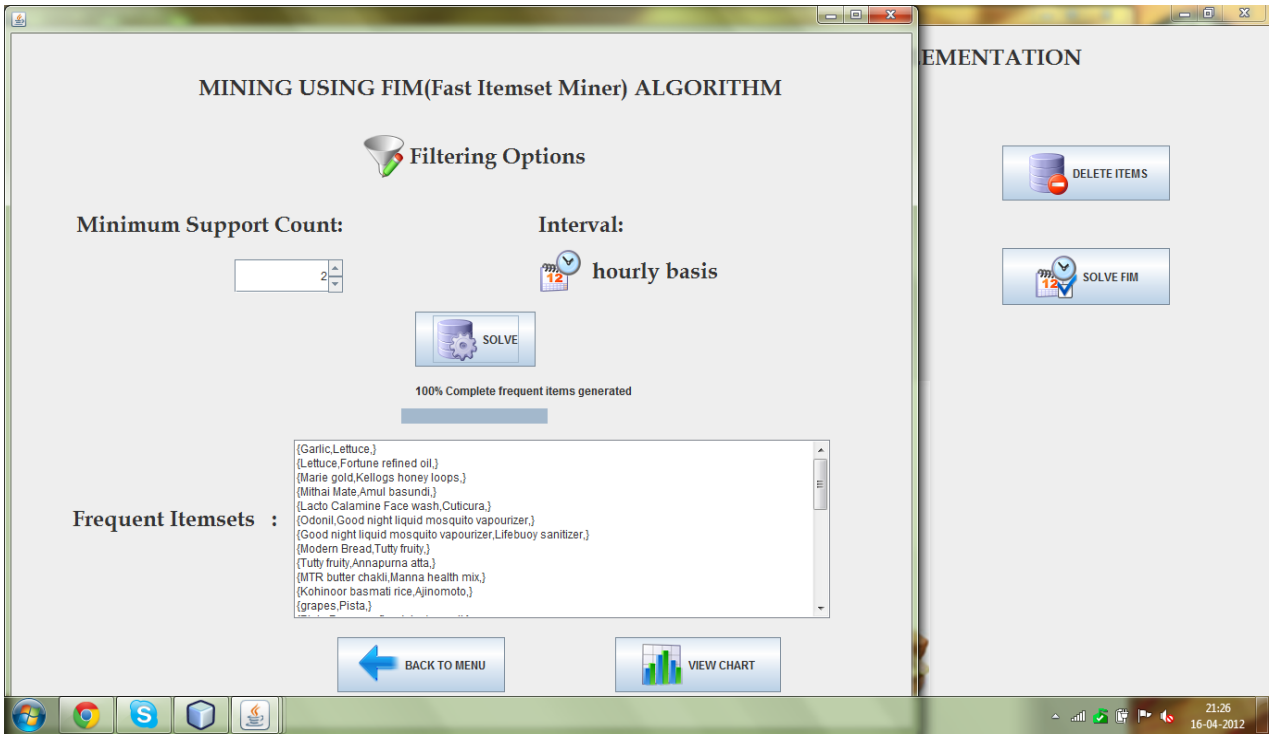
ITEM DELETED SCREEN



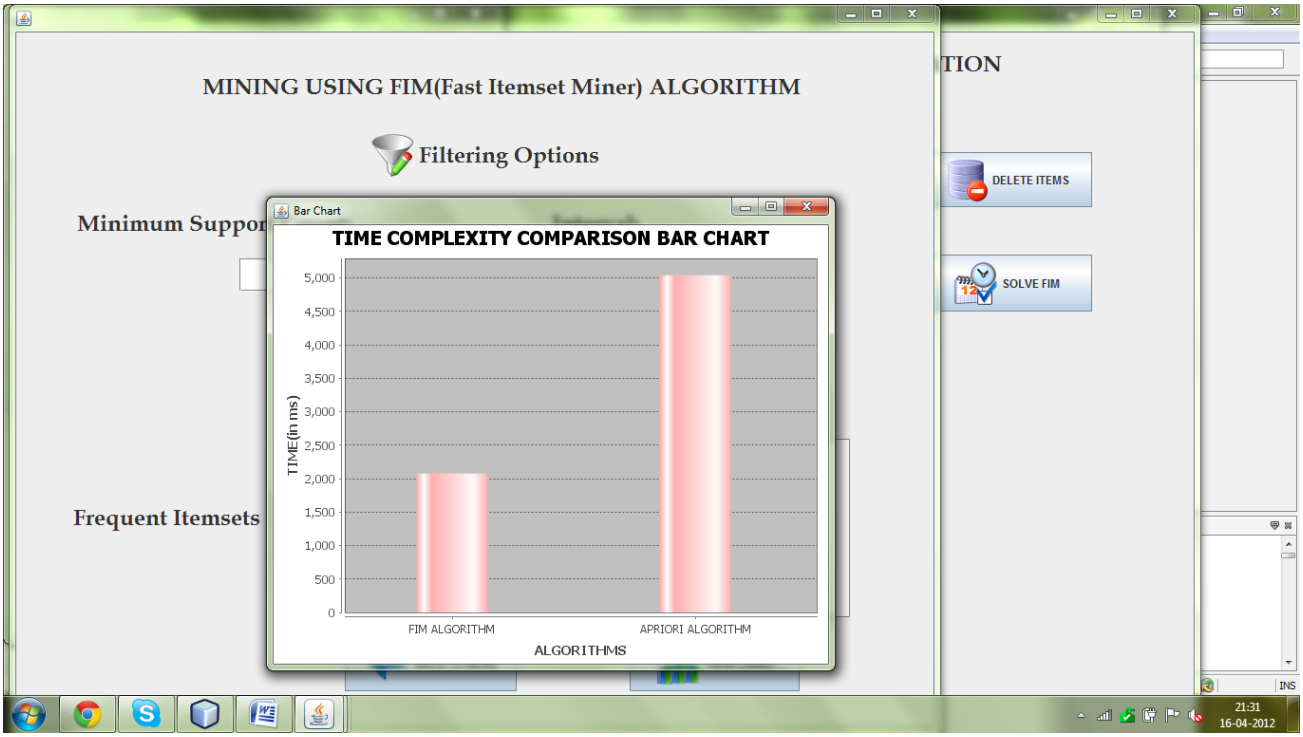
SCREEN FOR SOLVING APRIORI ALGORITHM

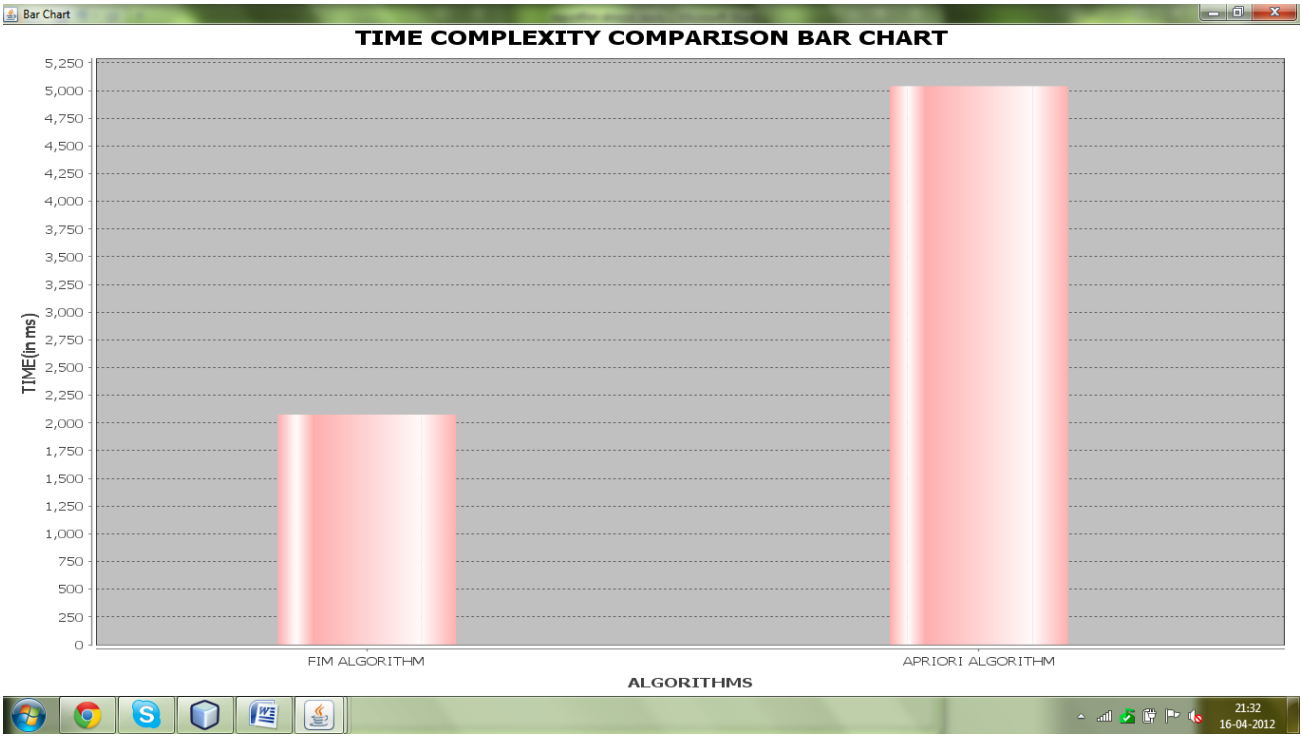


SCREEN FOR SOLVING FIM ALGORITHM

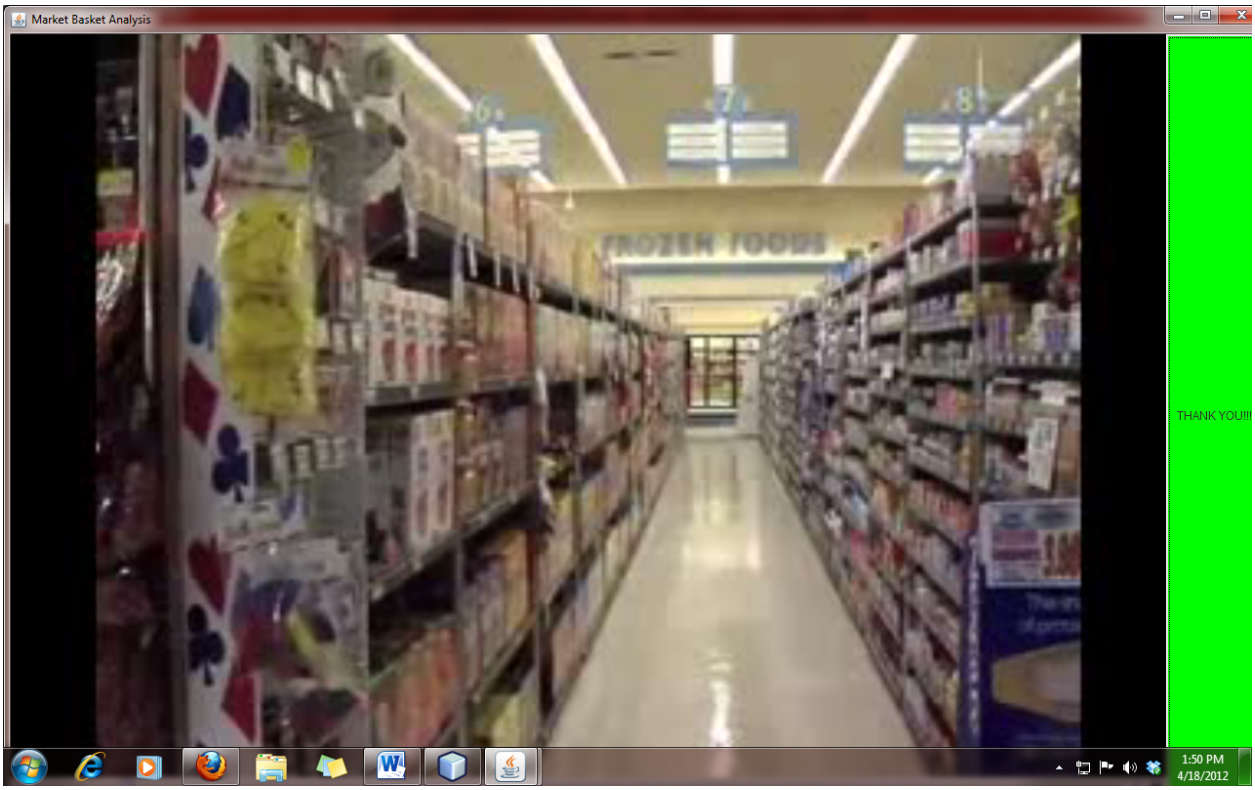


SCREEN FOR TIME COMPLEXITY COMPARISON





SCREEN FOR MAKET BASKET APPLICATIONS



REFERENCES

[1] Alexander Archip and Mitica Craus ,'The Fast Itemset Miner: A detailed analysis of the Candidate Generation Stages' International Conference on System Theory Control and Computing (ICSTCC),Oct 2011.

[2] J. Han and M. Kamber, Data Mining - Concepts and Techniques.
Morgan Kaufman Publications, 2006.

[3] Wanjun Yu,Xiaochun Wang, Fangyi Wang, Erkang Wang, Bowen Chen, 'The research of improved apriori algorithm for mining association rules',11th IEEE International Conference on Communication Technology,ICCT 2008.