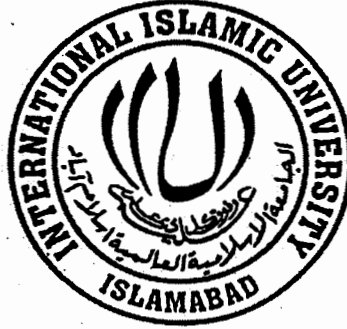


**Finding Frequent Item Sets Using  
Dissimilarity Based Measure**

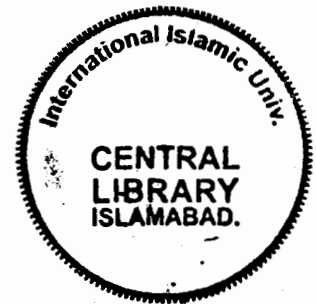


105240

*Developed By*  
**Saima Abdullah**  
325-FAS/MSCS/F06

*Supervised By*  
**Mr. Muhammad Imran Saeed**  
Asstt. Professor, DCS  
IIUI

*Co-Supervised By*  
**Ms. Saleha Jamshaid**  
Research Associate, IIUI



Department of Computer Science  
Faculty of Basic & Applied Sciences  
International Islamic University, Islamabad  
2008



*In the Name of*

*Allah*

*The Most Beneficent, The Most Merciful*

**A dissertation submitted to the  
Department of Computer Science,  
Faculty of Basic and Applied Sciences,  
International Islamic University, Islamabad, Pakistan  
as a partial fulfillment of the requirements  
for the award of the degree of  
Master of Science in Computer Sciences  
(MSCS)**

## **To my beloved parents**

Who always encourage me to accept challenges and guided me in every walk of life with kindness and affection, motivated me to work hard and always prayed for my success.

**Department of Computer Science  
Islamic International University, Islamabad**

**Date:**

**Final Approval**

It is certified that we have read project titled "Finding Frequent Itemsets using Dissimilarity Based Measures" submitted by Saima Abdullah D/O Muhammad Abdullah Reg. No. 325-FAS/MSCS/F06. It is our judgment that this project is of sufficient standard to warrant its acceptance by Islamic International University, Islamabad for the degree of Master of Science in Computer Sciences.

**Committee**

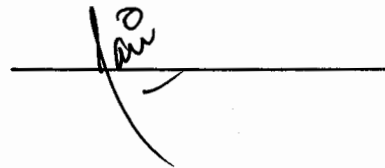
**External Examiner:**

Dr. Abdus Sattar,  
Ex-Director General  
Pakistan Computer Bureau, Islamabad.



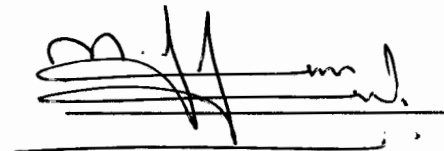
**Internal Examiner:**

Ms. Zakia Jalil  
Lecturer, DCS  
IIUI.



**Supervisor:**

Mr. Muhammad Imran Saeed  
Asstt. Professor, DCS  
IIUI.



**Co-Supervisor:**

Ms. Saleha Jamshaid  
IIUI.



## **DECLARATION**

I hereby declare that this research, neither as a whole nor as a part has been copied from any source. It is further declared that I have developed this research entirely based on my personal efforts under the able guidance of my Supervisor, Mr. Muhammad Imran Saeed. If any part of this report is proved to be copied or reported at any stage, I accept the responsibility to face the subsequent consequences. No part of this work inscribed in this report has either been submitted to any other University for the award of degree/ qualification.

Saima Abdullah

325-FAS/MSCS/F06

## ACKNOWLEDGEMENT

All praise to Allah, the Almighty, the Most Merciful and Compassionate, who gave me the ability to complete this research work.

I am thankful to my parents whose prayers and moral support has always been with me.

I express my profound gratitude to my project supervisor **Mr. Muhammad Imran Saeed**. His motivation guided me to this achievement and his suggestions and encouragement was always with me during the completion of this research work. I would also like to thank

**Ms. Saleha Jamshaid** for her help.

I am also thankful to **Mr. Saif-ur-Rehman** for his untiring help during the project efforts. He did a lot of efforts for my project.

I would like to thank my teachers, friends and all those persons who helped me during the project.

Saima Abdullah

325-FAS/MSCS/F06

## Project In Brief

**Project Title:** Finding Frequent Item sets using Dissimilarity Based Measures

**Objective:** The purpose of this research is to study distance based dissimilarity measure for mining Association Rules.

**Organization:** Department of Computer Science,  
International Islamic University, Islamabad.

**Undertaken By:** Saima Abdullah (325-FAS/MSCS/F06)

**Supervised By:** Mr. Muhammad Imran Saeed  
Assistant Professor,  
Department of Computer Science,  
International Islamic University, Islamabad.

**Tools Used:** MS- Access , Visual C#.Net

**Operating System:** Windows XP

**System Used:** Pentium IV

**Date Started:** September 2007

**Date Completed:** September 2008



## ABSTRACT

Association Rule mining is a two step process. First, finding frequent itemsets and second, generating association rules from those frequent item sets. First step is core of process. Many algorithms have been presented for this purpose. Here, a new technique is presented for finding frequent item sets based on clustering measures i.e Jacquard's dissimilarity measure. The proposed technique focuses on that clustering measures can be used for association rule mining. This provides the same results as any other Apriori based algorithm with less number of database scans and no candidate generation.

## Table of Contents

Chapter	Page No.
<b>1 Introduction.....</b>	<b>1</b>
1.1. Data Warehousing.....	1
1.1.1 Architecture of data warehouse.....	2
1.2. Data Mining.....	2
1.2.1. Background.....	3
1.2.2. Classification of Data Mining.....	4
1.3. Uses of Data Mining.....	5
1.4. Association Rule Mining.....	5
1.4.1. Frequent Item Sets.....	6
1.4.2. Association Rules.....	6
1.5. Existing Techniques for Finding Frequent Item Sets.....	7
1.6. Scope of the Project.....	8
<b>2. Literature Survey.....</b>	<b>10</b>
2.1. Literature Survey .....	12
2.1.1. AIS Algorithm.....	12
2.1.2. Apriori Algorithm for Finding FIS.....	13
2.1.3. Sampling Approach for Finding FIS.....	14
2.1.4. Partition Algorithm for Finding FIS.....	15
2.1.5. MAXMINER Algorithm.....	15
2.1.6. FP-Growth Algorithm for Finding FIS.....	16
2.1.7. Eclat Algorithm for Finding FIS.....	18
2.2. Problem Statement.....	19

<b>3. Problem Domain and Proposed Solution.....</b>	<b>21</b>
3.1. Large candidate set size.....	21
3.2. Large number of database scans.....	21
3.3. Algorithm execution time.....	22
3.4. Proposed Solution.....	22
3.4.1. Database scans.....	24
3.4.2. Large size of candidate sets.....	24
3.5. Dataset.....	24
<b>4. System Design.....</b>	<b>25</b>
4.1. Input.....	25
4.2. Software tools for development.....	25
4.3. End Users.....	25
4.4. Algorithm.....	26
4.5. Architectural diagram.....	27
4.6. Flow chart.....	28
4.7. Major Modules.....	28
4.7.1. Database acquisition.....	29
4.7.2. <i>Combination</i> creation.....	29
4.7.3. Combination-matrix creation.....	29
4.7.4. Frequent item sets creation.....	30
4.8. Software execution.....	31
4.8.1. Login.....	31
4.8.2. Load database.....	31
4.8.3. <i>Combination</i> .....	33
4.8.4. Combinations <i>matrix</i> .....	33
4.8.5. Frequent Item sets.....	35

<b>5. A Comparative Case Study.....</b>	
5.1. Sample database.....	<b>36</b>
5.2. Apriori algorithm Mining Process.....	36
5.3. DS-Miner Mining Process.....	36
	39
<b>6. Conclusion and Future Work.....</b>	
6.1. Introduction.....	<b>45</b>
6.2. Conclusion.....	45
6.3. Future Work.....	46
	46
<b>References &amp; Bibliography.....</b>	
	<b>47</b>

## List of Figures & Tables

Figure/Table No.	Caption	Page No.
1.1	Architecture of Data warehouse.....	2
2.1	Working of FP-Growth Algorithm.....	17
2.2	Eclat Algorithm.....	18
4.1	Architectural Diagram of DS-Miner Algorithm.....	27
4.2	Flow chart of DS-Miner algorithm.....	28
4.3	Database Appearance used for Algorithm.....	29
4.4	Combination Matrix.....	30
4.5	Login Form.....	31
4.6	Load Database.....	32
4.7	Database loaded.....	32
4.8	<del>Combination</del> .....	33
4.9	Combination Matrix.....	34
4.10	Combination Matrix with results.....	34
4.11	Frequent Item Sets .....	35
5.1	Sample database for comparative case study.....	36
6.1	Experimental Results of DS-Miner.....	45

# **CHAPTER 1**

## **INTRODUCTION**

## **1. Introduction:**

With the passage of time, people have found new ways to increase their business like publicity, good environment, cleanliness etc. But with all these things they have also learnt the use of new technologies. If we take the example of super store, then in past there was no new technology for store owners to increase their sales. They only used to make observations and on these they make strategies to develop their business But now with new techniques they can make decision like to place which things together so that customer for one thing also tempted to buy other and to put which thing on sale and other on high price etc. To cater these things properly can increase his sales very much. Most important of all these is to arrange items in shelves in a way that can enhance sales.

### **1.1 Data Ware Housing:**

Data warehouse is a single storage repository in which data from all kinds of sources is stored together. In this way it becomes easier to analyze the information than it would be when using multiple data models for multiple sources.

It is a relational database that is designed for query and analysis rather than for transaction processing. It usually contains historical data derived from transaction data, but can include data from other sources. Data warehouses separate analysis workload from transaction workload and enable an organization to consolidate data from several sources.

Tools for extracting, transformation and loading data are also included in data warehousing. [1]

### 1.1.1 Architecture of Data warehouse:

Architecture, basically shows how the data warehouse is built. Following figure shows a basic architecture of Data ware House

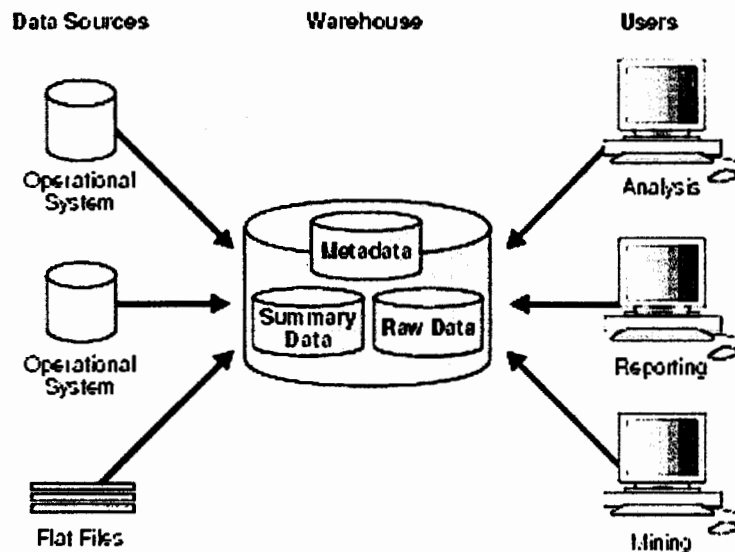


Fig 1.1 Architecture of data warehouse [1]

### 1.2 Data Mining:

Data mining is the process of analyzing large amount of data and then converting it into useful information that can be used to increase revenue and cut off the cost.

“Data mining is the process of sorting through large amounts of data and picking out relevant information. It is usually used by business intelligence organizations, and financial analysts, but is increasingly being used in the sciences to extract information from the enormous data sets generated by modern experimental and observational methods.” [2]



### 1.2.1 Background

“Traditionally, business analysts have performed the task of extracting useful information from recorded data, but the increasing volume of data in modern business and science calls for computer-based approaches. As data sets have grown in size and complexity, there has been a shift away from direct hands-on data analysis toward indirect, automatic data analysis using more complex and sophisticated tools. The modern technologies of computers, networks, and sensors have made data collection and organization much easier. However, the captured data needs to be converted into information and knowledge to become useful. Data mining is the entire process of applying computer-based methodology, including new techniques for knowledge discovery, to data.

The term data mining is often used to apply to the two separate processes of knowledge discovery and prediction. Knowledge discovery provides explicit information that has a readable form and can be understood by a user. Forecasting, or predictive modeling provides predictions of future events and may be transparent and readable in some approaches (e.g., rule-based systems) and opaque in others such as neural networks. Moreover, some data-mining systems such as neural networks are inherently geared towards prediction and pattern recognition, rather than knowledge discovery.” [2]

**Metadata**, or data about a given data set, are often expressed in a condensed *data-minable* format, or one that facilitates the practice of data mining. Common examples include executive summaries and scientific abstracts.

Data mining relies on the use of real world data. This data is extremely vulnerable to collinearity precisely because data from the real world may have unknown interrelations. An unavoidable weakness of data mining is that the critical data that may expose any relationship might have never been observed. Alternative approaches using an

experiment-based approach such as Choice Modelling for human-generated data may be used. Inherent correlations are either controlled for or removed altogether through the construction of an experimental design.

### 1.2.2 Classification of data mining:

The task of data mining can be divided into two terms:

- predictive mining
- descriptive mining

#### **Predictive mining:**

A predictive model can answer questions such as "Is this transaction fraudulent," "How much profit will this customer generate," or "Where in the body is the most likely location of this patient's primary tumor?" [3]

#### **Descriptive Mining:**

Descriptive models provide information about the relationships in the underlying data, generating information of the form "A customer who purchases diapers is 3 times more likely to also purchase beer," or "Weight and age, together, are the most important factors for predicting the presence of disease  $x$ ,"

**Clustering** is a descriptive technique that groups similar entities together and puts dissimilar entities in different groups. It can be used in marketing for finding customer affinity groups. [3]

Most but not all predictive models are also descriptive. An example is a decision tree. Some descriptive models cannot be used for prediction. An example is a model formed using a technique known as association. [3]

### 1.3 Use of Data Mining:

- **Market basket analysis**, is the most important use of data mining. If a super store records the purchases of customers, a data-mining system could identify those customers who favour milk over coke. Although some explanations of relationships may be difficult, taking advantage of it is easier. The example deals with association rules within transaction-based data. Not all data are transaction based and logical or inexact rules may also be present within a database. In a manufacturing application, an inexact rule may state that 73% of products which have a specific defect or problem will develop a secondary problem within the next six months.
- In recent years, data mining has been widely used in area of science and engineering, such as bioinformatics, genetics, medicine, education, and electrical power engineering.

### 1.4 Association rule mining:

In data mining most important concept is association rule mining. It comes in descriptive mining. Association rule mining finds interesting associations and/or correlation relationships among large set of data items. A typical and widely-used example of association rule mining is Market Basket Analysis. Association rule mining consists of two steps:

- Finding frequent/large itemsets
- Finding association rules from these large itemsets

### 1.4.1 Frequent Itemset:

Frequent itemset mining is the first step in association rule mining. It is the main task because once the frequent itemset has been found, it is straight forward to generate association rules.

In many (but not all) situations, we only care about association rules or causalities involving sets of items that appear frequently in baskets. For example, we cannot run a good marketing strategy involving items that no one buys anyway. Thus, much data mining starts with the assumption that we only care about sets of items with high support;

i.e., they appear together in many baskets. We then find association rules or causalities only involving a high-support set of items (i.e.,  $\{X_1; \dots; X_n; Y\}$ ) must appear in at least a certain percent of the baskets, called the support threshold.

### 1.4.2 Association Rules:

Association rules identify sets of data items that are statistically related in the underlying data.

For example, data are collected using bar-code scanners in supermarkets. Such 'market basket' databases consist of a large number of transaction records. Each record lists all items bought by a customer on a single purchase transaction. Managers would be interested to know if certain groups of items are consistently purchased together. They could use this data for adjusting store layouts (placing items optimally with respect to each other), for cross-selling, for promotions, for catalog design and to identify customer segments based on buying patterns.

Association rules provide information of this type in the form of "if-then" statements. These rules are computed from the data and, unlike the if-then rules of logic, association rules are probabilistic in nature.

### 1.5 Existing Techniques for Finding frequent itemsets:

As Frequent itemset mining is the main task in association rule mining, so many researchers have concentrated on this step most. Following are some techniques for finding frequent itemsets proposed by many researchers:

- **Apriori** is an extension of AIS algorithm proposed by Agarwal. It is support based algorithm. In it closure property is used that any subset of an frequent itemset must also be frequent. This is why it uses in each iteration only the itemsets that proved to be frequent in previous iteration. This algorithm finds frequent itemsets based on user specified min. support threshold. These frequent itemsets are used to generate a new candidate itemset. This procedure continues until no further items remain. In it there are large number of DB scans and also large number of candidate sets.
- **Partition algorithm** makes the partitions of database and for each item a tid-list is maintained . All local frequent itemsets are generated via tid-list intersection. These local frequent itemsets are merged and a second pass is made through all the partitions. The database is again converted into vertical layout and the global count for all chosen items is obtained. Partition algorithm involves only two DB scans. The key observation is that a globally frequent itemset in at least one partition must be locally frequent in at least one partition. But in this algorithm there can be large number of local frequent itemsets.

- **Eclat algorithm** is also support based and it is an extension of Apriori algorithm. It basically uses vertical database layout. Horizontal database is converted into vertical format for each item tid-list is stored. Support of itemset is determined by intersecting tid-list of two of its (k-1) subsets. But with this algorithm primary tid-list may become very large and it becomes difficult for memory to store all these ids.
- **Sampling approach** for finding frequent itemsets is proposed by Toivonen. With this approach, only a single sample is taken from the database from which a candidate set is derived for full database search. Apriori algorithm is applied on the sample. The problem in sampling is that the candidate set derived is necessarily a superset of the actual set of frequent sets and may contain many false positives.
- **FP-Growth algorithm** uses the FP-tree structure. FP-Growth tree is memory resident and requires additional storage in every node of the FP-tree. This method requires only two database scans for mining all frequent itemsets. In it divide and conquer methodology is used so mining task is divided into smaller parts.
- **MAXMINER** algorithm is used for finding maximal frequent itemsets. In it breadth-first traversal is used. It reduces database scans by employing Look-Ahead pruning strategy.

## 1.6 Scope of the project:

Association rule mining has been a very effective research area. This task basically consists of two steps:

- Finding frequent/large itemsets
- Generating association rules from these frequent itemsets

As finding frequent itemsets is the core of this, so main devotion has been given to it. All previous algorithms/ techniques that have been used for finding frequent itemsets have following problems:

- These all are support based, so in case of very low or very high support threshold , many frequent itemsets can be pruned/eliminated or included in FIS.
- Many algorithms require large number of database scans which require huge memory.
- Also apriori algorithm generates large candidate sets which are of no use in later processing and are pruned.

In this research work, a new technique is introduced for finding frequent itemsets that will use Jacquard's dissimilarity instead of support. In it focus is on that clustering can also be used for mining association rules.

## **CHAPTER 2**

# **LITERATURE SERVEY**



## 2. Literature Survey:

Data mining is the process of extracting only useful information from rocks of data. In first step existing data is transformed into patterns (data having same properties) then rules are applied to extract information from these patterns. This information can be used for decision making.

Association rule mining searches for interesting relationship among items in a given data set. It is an important data mining model studied extensively by the database and data mining community. Association rule analysis is used for Market Basket Analysis. In it association rule mining is used for

- Analyzing buying habits
- Help retailers for developing strategies

For example:

There is some store XYZ. The owner of store wants to increase his sales. He has good quality products, clean environment etc. But with all these, he also needs to use some other strategy. So association rule mining can help him so that he can find association between different items and place them together in shelves. If a customer comes to buy one thing, he is tempted to buy other thing also if it is placed near. In this way sales of both is increased.

e.g if Bread and Butter have strong association, then placing them properly can increase the sale of both.

So association rule mining has always been an interesting topic for researchers. Mining association rules can be divided into two steps:

- Find all frequent itemsets which have support greater than predetermined support threshold
- Generate all association rules , which have confidence greater than minimum predetermined confidence.

This shows that in association rules two factors are important that are support and confidence.

Association rules are considered interesting if they satisfy both a minimum support threshold and a minimum confidence threshold. Such threshold is set by user or domain experts.

## 2.3 Literature Survey:

Finding frequent itemsets is the first step in Association rule mining. Following literature was reviewed during thesis writing.

1. AIS Algorithm for finding FIS
2. Apriori Algorithm for finding Frequent Item Sets(FIS) :
3. Sampling approach for finding Frequent Item Sets
4. Partition Algorithm
5. Maxminer Algorithm
6. FP-Growth Algorithm
7. Eclat Algorithm

And also a lot of other material from internet and books to help in my work. Following is a brief description of all the above algorithms:

### 2.3.1 AIS Algorithm:

This algorithm was proposed by Agrawal in [4]. It was first algorithm to find all frequent itemsets in a transactional database [Agrawal 1993]. This technique is limited to only one item in the consequent. The AIS algorithm makes many scans of the database. The AIS algorithm makes multiple passes over the entire database. During each pass, it scans all transactions. In the first pass, it counts the support of individual items and determines which of them are large or frequent in the database. Large itemsets of each pass are extended to generate candidate itemsets. After scanning a transaction, the common itemsets between large itemsets of the previous pass and items of this transaction are determined. These common itemsets are extended with other items in the transaction to generate new candidate itemsets. [5].

This process continues until no more candidate itemset remains. As in this algorithm all frequent and candidate itemsets are assumed to be stored in the main memory, memory management is also proposed for it when memory is not enough. One approach is to delete candidate itemsets that have never been extended and other approach is to delete candidate itemsets that have maximal number of items and their siblings and store this parent itemset in the disk as a seed for the next pass.

AIS algorithm has some disadvantages like:

- It generates too many candidate itemsets which finally turned out to be infrequent/small. This requires more space and waste much effort.
- Other problem in it is too many whole database scans.

### **2.3.2 Apriori Algorithm for finding Frequent Item Sets(FIS)**

This algorithm for finding frequent itemsets was proposed by Agarawal in [6]. It is an extension of AIS algorithm for finding frequent itemsets which requires many database scans, many candidate generation, and store counter of each candidate. Apriori algorithm is more efficient than AIS.

In Apriori algorithm first candidate-1 itemset is generated and then database is scanned to calculate their support count. Large/frequent-1 itemsets are generated and all itemsets not satisfying minimum support threshold are pruned. Then candidate-2 itemset is generated by joining items in frequent/large-1 itemset. It stores the candidate itemsets in a Hash tree, a special data structure. Then database is again scanned to check support count of each itemset in candidate 2-itemset. This process continues until all items are checked.

In finding FIS, Apriori avoids the effort wastage of counting the candidate itemsets that are known to be infrequent. The candidate sets are generated by joining the frequent itemsets only level wise and also candidates are pruned according to the apriori property. This can reduce the computation, I/O cost and memory requirement.

But Apriori algorithm still has the drawback that it scans whole database again and again during processing. Its main drawback lies in its inefficient support counting mechanism. As already explained, for each transaction, we need to check for every candidate set whether it is included in that transaction, or otherwise, we need to check for every subset of that transaction whether it is in the set of candidate sets.

When transactions are large, generating all  $k$ -subsets of a transaction and testing for each of them whether it is a candidate set, can take a prohibitive amount of time.

Testing for each candidate set whether it is contained in the given transaction can also take too much time when the collection of candidate sets is large.

Another variation of Apriori algorithm is AprioriTid, which works differently from Apriori after first database scan. It uses pair of itemset and its TIDs for counting their support after first scan.

### **2.3.3 Sampling Approach to find FIS:**

This approach was proposed by Toivonen [7] for finding frequent itemsets. It is applied on large databases. In this approach a random sample is taken from the database and frequent itemsets are searched from that sample. In this approach only one database scan is required because the size of sample is taken that can be accommodated in main memory. This approach is beneficial when efficiency is of utmost importance.

The problem in partition/sampling is that the candidate set derived is necessarily a superset of the actual set of frequent sets and may contain many false positives.

### 2.3.4 Partition Algorithm for Finding FIS:

Partition algorithm is presented in [8]. This algorithm uses two database scans:

- In first scan, all potentially large itemsets are generated. This set of large itemsets is superset of all large itemsets.
- In second scan of database, counters are created for all these itemsets and their support is counted.

Basically in this algorithm, the database is divided into logical partitions. Each partition is read and vertical tid-lists are formed for each item. All locally frequent/large itemsets are generated from each partition separately one by one and at the end these frequent itemsets from all partitions are merged to make one set of all potentially frequent itemsets. Then actual support for these itemsets is counted to identify frequent/large itemsets by making second database pass.

The partition sizes are chosen so that each partition can be accommodated in main memory. There are only two database scans in whole process. The key observation used is that a globally frequent itemset in at least one partition must be locally frequent in at least one partition.

The performance can be decreased if database is heterogeneous because there are too many local frequent itemsets. On the other hand, if the entire database fits into main memory then dividing the transactions into many partitions is not necessary.

### 2.3.5 Maxminer Algorithm:

This algorithm is used to find maximal elements [9]. It employs the breadth-first traversal of the search space. It assumes that entire database fits into main memory.

Maximal Frequent itemset is one that has no frequent superset .

e.g

(Minimum support = 3 ):

$$D = \{\{1,2,3\}, \{1,2,3,4\}, \{1,4\}, \{1,2\}, \{3\}, \{1,2,3\}\}$$

$\{1,2\}$  is frequent.

$\{1,2,3\}$  is maximally frequent.

This algorithm reduces the search space by using a look-ahead mechanism for pruning itemsets. In it if a node with all its extensions can determine to be frequent , then there is no need to further process the node.

Maxminer algorithm is applied on the original horizontal database. In it number of database scans is same as in Apriori.

### 2.3.6 FP-Growth Algorithm for finding FIS:

It was proposed by JIAWEI HAN, JIAN PEI, YIWEN YIN . This algorithm uses support-based measure to find maximal frequent itemsets by using I-projected databases [10]. This algorithm Compress a large database into a compact, FP-tree structure.

#### FP-tree:

FP-growth tree is memory resident and requires additional storage <sup>for</sup> every node of FP-tree.

In it

Each node contains:

- Item label
- Count

Multiple instances of item may appear in tree  
 Built with ordering of items based on support

FP-Growth algorithm requires only two database scans for mining frequent/large itemsets:

- In first scan, it finds all frequent itemsets i.e item sets with support less than or equal to min. specified threshold.
- In second scan, it constructs 1<sup>st</sup> FP-tree that contains all frequency information of the given dataset.

The FP-tree stores a single item (attribute) at each node, and includes additional links to facilitate processing. These links start from a *header table* and link together all nodes in the FP-tree which store the same "label", i.e. item.

The algorithm, FP-growth, for mining the FP-tree structure is a recursive procedure during which many sub FP-trees and header tables are created. Following figure shows the working of FP-Growth algorithm:

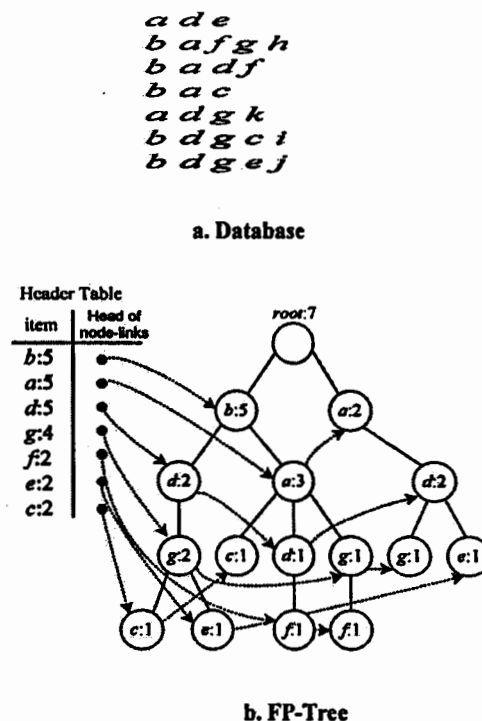


Fig 2.1 Working of FP-Growth Algorithm [11]



In this algorithm, the search technique employed in mining is a partitioning-based, divide-and conquer method rather than Apriori-like level-wise generation of the combinations of frequent itemsets. This dramatically reduces the size of conditional-pattern base generated at the subsequent level of search as well as the size of its corresponding conditional FP-tree. [12]

### 2.3.7 Eclat Algorithm for Finding FIS:

This algorithm is same like FP-Growth algorithm. It also uses support based measures to find Maximal Frequent itemsets. A difference between it and FP-Growth algorithm is that they use different data structures. It uses vertical layout and uses the intersection based approach to compute the support of an itemset. [13].

e.g

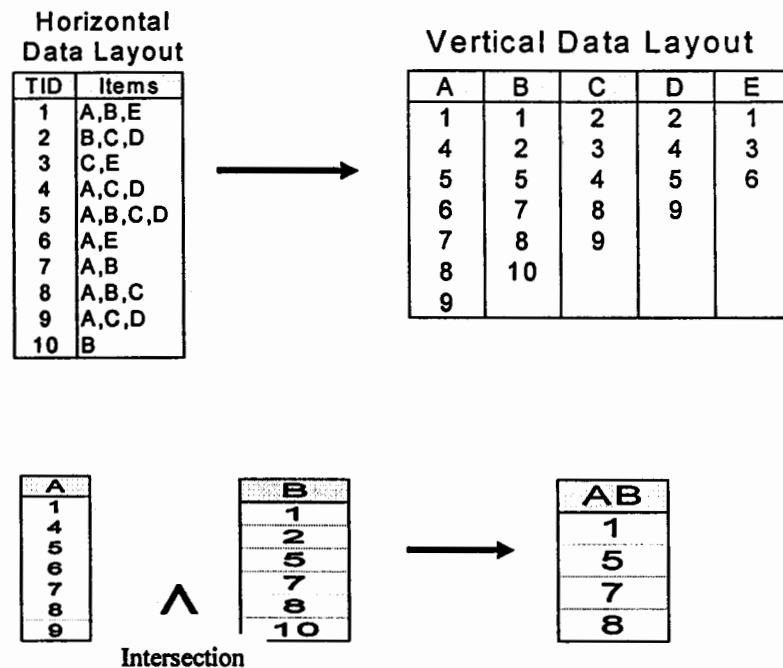


Fig 2.2 Eclat Algorithm [13]

Eclat is different from Apriori algorithm in pruning step. In it all items in database are reordered in support ascending order at every recursion step [14]. Eclat counts the support of all itemsets more efficiently than Apriori algorithm.

Disadvantage of this algorithm is that it generates too many candidate sets to derive frequent itemsets at each iteration of the algorithm. Another problem is in it Intermediate tid-lists may become too large for memory.

## 2.4 Problem Statement:

Mining Association Rules, is a two step process:

- All itemsets that have support greater than minimum specified threshold are found which are called frequent itemsets.
- Association rules are generated from these frequent itemsets.

The main task is finding frequent itemsets which is computationally difficult, so it dominates the efficiency of the algorithm. There are a number of techniques proposed by different researchers for finding frequent itemsets. All these techniques uses support based measures. All these techniques have problems like:

In AIS, there are too many DB scans and a large number of candidate itemsets that require more space and effort.

In Apriori algorithm also, database is scanned again and again and also in it we need to check for each candidate set, all transactions that whether it is present in it or not. This requires lot of time and space.

The performance of Apriori-Tid algorithm is not better than Apriori at initial stages.

Partition algorithm is not efficient if database can fit into main memory and also finding accurate number of partitions is a problem.

**Eclat** algorithm requires too many database scans and also it requires large memory space.

The literature survey shows that mining frequent itemsets is main problem in association rule mining. The support based strategies used for this purpose has many problems like: large number of candidate sets, huge number of DB scans, Poor support count method etc. So there is a need to introduce a new technique. For this purpose, I shall use Dissimilarity Based Measure. For it **Jacquard's distance** (measure dissimilarity) equation will be used that is:

$$D_{ij} = \frac{(q+r)}{(p+q+r)}$$

here, i and j are two items while

p = number of times both i and j are resent in database  
q = number of times i is present in database but j is not  
r = number of times j is present in database but i is not

## **CHAPTER 3**

# **PROBLEM DOMAIN AND PROPOSED SOLUTION**

### **3. Problem Statement and Proposed Solution**

Association Rule Mining basically considers the problem of market that retailers have to face. It finds interesting relationships between items. That's why the association model is often known as "market basket analysis". Initially association models were used only for basket data but now they are used for other applications also such that catalog-design, web page personalization and target marketing etc.

A lot of work has been done by different researchers in the field of association rule mining. There are some problems that must be considered while using a new algorithm for association rule mining because these can degrade the performance of the algorithm. These are:

#### **3.1 Large Candidate Set Size:**

For mining frequent itemsets, candidate sets are generated and frequent itemsets are generated from these. For example to find frequent 1-itemsets, we need to consider candidate 1-itemset. So if the size of candidate itemset is very large, it can degrade the performance very much because it consider false itemsets. So to improve the efficiency of the algorithm, candidate sets should be made smaller including only items that should be large items.

#### **3.2 Large Number of Database Scans:**

The other main issue in association rule mining is large number of database scans that are required during the execution of algorithms. For example, first individual items are examined to find frequent items by scanning database. Then frequent items are examined together to find out confidence between two frequent itemsets. For frequent 2-itemsets, confidence is measured and then frequent 3-itemsets are found. This process continues till frequent k-itemsets. During this process database is scanned many times.

As in data warehouse there is always large amount of data so again and again accessing database is not desirable because multiple disk I/Os are not appreciated specially with such large amount of data. So database scans should be minimized.

### **3.3 Algorithm Execution Time:**

In data ware house, there is large amount of data so often the queries take more time to execute than usual. Data mining queries in fact help in decision making. These decisions can help a lot to improve the business in question. So such delays in execution of queries are tolerable in business community.

Above are some major issues that must be addressed in association rule mining when designing a new algorithm for it.

### **3.4 Proposed Solution:**

An efficient algorithm for association rule mining is one that can provide solution to some of the above mentioned factors like the algorithm should create less number of database scans, small candidate itemsets, consider only true itemsets etc. All the algorithms that have been developed for association rule mining are support based. In these algorithms support of itemsets is counted and then it is checked whether it satisfies user defined minimum support threshold. If it does, itemset is Frequent or Large, if it does not then itemset is infrequent. In such approach, it is possible that many frequent itemsets are pruned because of very high support threshold or many infrequent itemsets are included in result because of very low support threshold.

Here, a new technique for finding frequent itemsets is to be used. This is **clustering based technique**. It is to use dissimilarity based measure to find FIS instead of support based measure. In it Jacquard's distance equation (measure dissimilarity) is used. This is first time that this equation is used for mining frequent itemsets.

Following is Jacquard's distance equation:

$$D_{ij} = \frac{(q+r)}{(p+q+r)}$$

Where:

p = Number of transactions in which both i and j are present

q = Number of transactions in which i is present but j is not

r = Number of transactions in which j is present but i is not

For n items it is:

$$d(i_1, i_2, i_3, \dots, i_n) = \sum_{i_1=0}^1 \sum_{i_2=0}^1 \sum_{i_3=0}^1, \dots, \sum_{i_n=0}^1 f(i_1, i_2, \dots, i_n) - \\ f(0,0, \dots, 0) - f(1,1, \dots, 1) + \sum_{i_1=0}^1 \sum_{i_2=0}^1 \sum_{i_3=0}^1, \dots, \sum_{i_n=0}^1 f(i_1, i_2, \dots, i_n) \\ - f(0,0, \dots, 0)$$

This equation is applied on the values from Combination-Maps (a sort of K-Map) which are created for each combination of items in database. The values in each cell of Combination-Maps are the number of times each corresponding combination exists in the database.

With this new technique, following overheads are minimized that were present in previous techniques:

#### **3.4.1 Database Scans:**

All previous techniques suffer from large number of database scans which result in multiple disk I/Os. As data mining includes large amount of data so this multiple disk I/Os decreases the performance of algorithm. The proposed algorithm has overcome this problem by scanning the Database only once in the process.

#### **3.4.2 Large Size of Candidate Sets:**

The size of candidate sets is also a problem in previous techniques. Large candidate sets often includes false itemsets which proved to be infrequent at the end. So these causes loss of memory and wastage of effort. Proposed technique does not include any type of candidate generation so it improves the efficiency of the algorithm.

#### **3.5 Dataset:**

The data used is synthetic data that is created exclusively for this study. Each tuple of the database is of the form <TID> <yes, no, yes> where TID is transaction identifier and <yes, no, yes> shows presence or absence of items. TID is primary key while for showing presence or absence of items , I have used yes/no data type.



# **CHAPTER 4**

## **SYSTEM DESIGN**

## **4. System Design**

As mentioned in chapter 2, Association Rule Mining consists of two steps. Generating association rules is trivial after calculation of accurate large/frequent itemsets, so my research is confined to the first step.

The purpose of this study is to develop a data mining algorithm which will find FIS on the basis of Dissimilarity measures rather than on the basis of support measures (that is used in previous algorithms).

### **4.1 Input:**

Input is a large transactional database created in MS-Access. The database is the exclusively created synthetic database, which is used to represent the actual transaction database's items.

### **4.2 Software Tool for Development:**

The choice of software tool is important and depends on the problem in hand. This is because of the various facilities provided by various languages and packages. After devoting a lot of time, " C#. Net " is considered to be quite appropriate. Windows - XP operating system is used for development.

### **4.3 End Users:**

There are only two users of this application. One is Administrator and other is user.

#### 4.4 Algorithm:

Following is the pseudo code for finding frequent itemsets using dissimilarity based measure:

##### Input:

- D : Transactional database
- $\phi$  : User specified threshold

##### Output:

- F : Frequent itemsets

##### Steps:

1.  $\rho = \{i_1, i_2, i_3, \dots, i_n\}$  set of data items in transactional database.
2. Create combinations for all items in  $\rho$
3. Create Combination Map for all combinations.
4. Scan the transactional database and put the presence for every combination of data items in the corresponding Combination Maps for every combination of row.
5. for every combination of  $\rho$ 
  - Calculate dissimilarity using combination map constructed for every combination using the following jacquard's dissimilarity equation:

$$d(i_1, i_2, i_3, \dots, i_n) = \frac{\sum_{i_1=0}^1 \sum_{i_2=0}^1 \sum_{i_3=0}^1 \dots \sum_{i_n=0}^1 f(i_1, i_2, \dots, i_n) - f(0,0,\dots,0) - f(1,1,\dots,1)}{\sum_{i_1=0}^1 \sum_{i_2=0}^1 \sum_{i_3=0}^1 \dots \sum_{i_n=0}^1 f(i_1, i_2, \dots, i_n) - f(0,0,\dots,0)}$$

- if  $d < \phi$  then  $i_1, i_2, i_3, \dots, i_n$  are frequent

( $i_1, i_2, \dots, i_n$  is itemset with n items)

## 4.5 Architectural Diagram:

Following is architectural diagram:

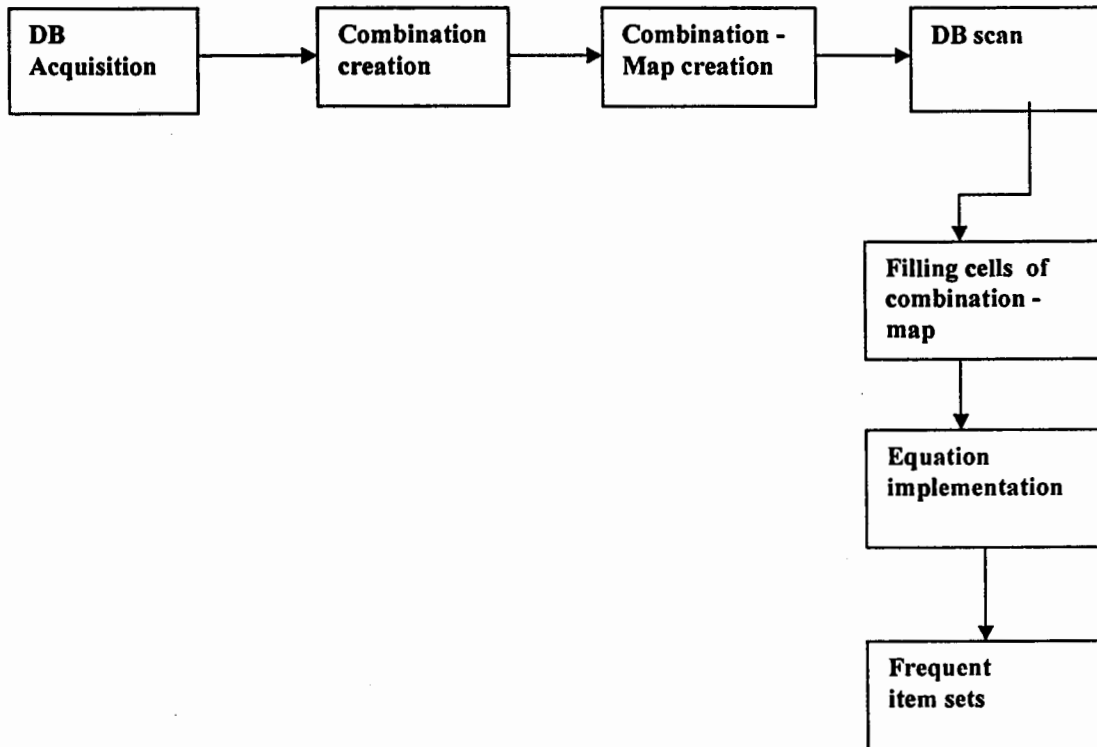


Fig 4.1 Architectural Diagram of DS-Miner Algorithm

## 4.5 Flow Chart

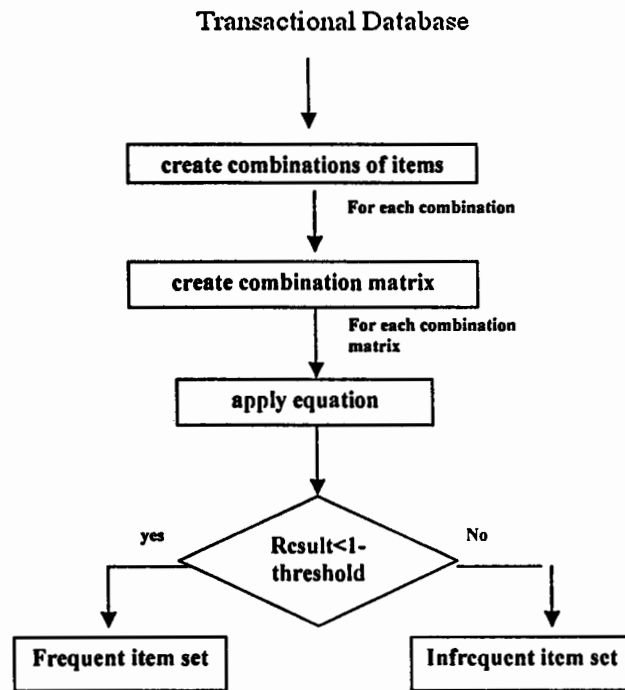


Fig. 4.2 Flow chart of DS-Miner algorithm

## 4.7 Major Modules:

The major modules of the algorithm are given below:

1. Database Acquisition
2. Combinations Creation
3. Combination-Matrix ( a sort of K-Map) Creation
4. Frequent Itemset Creation

### 4.7.1 Database Acquisition:

The input to this project is synthetic database which is created in MS Access exclusively for this project. The records are in the form <TID> <yes or no (showing item presence or absence)> as following:

Transaction id	Bread	Butta	Coke	Egg	Rice	Milk	Sugar	Burger	Tea Pack	Sandwich
1	Yes	Yes	No	Yes	Yes	Yes	No	No	No	No
2	No	No	Yes	Yes	No	Yes	No	Yes	No	No
3	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes
4	No	Yes	Yes	No	Yes	Yes	Yes	No	No	No

Fig 4.3 Database Appearance used for Algorithm

### 4.7.2 Combination creation:

All the items are taken from the database and then combinations are created for all these items. E.g if there are three items

(milk, sugar, rice) then combinations are:

(milk), (sugar), (rice), (milk, sugar), (milk, rice), (sugar, rice), (milk, sugar, rice)

### 4.7.3 Combination Matrix Creation:

Once the combinations for all items have been created then combination-matrix for all combinations are created. e.g for (milk, sugar) a separate combination matrix is created containing all possible combinations of milk and sugar like:

	sugar	— sugar
Milk	2	2
— Milk	0	0

Fig 4.4 Combination Matrix

Here milk and sugar are used to show absence of items. The values in the cells of combination-map are taken from the database showing number of transactions in which corresponding combination is present like:

Milk, sugar both are present in 2 transactions

Milk is present but sugar is not in 2 transactions etc.

Similarly for all combinations combination maps are created and are filled with values.

#### 4.7.4 Frequent Itemset Creation:

Once the combination-maps are created then it is straight forward to find frequent itemsets by applying **Jaccard's distance** on the values of matrix. The threshold is specified by the user that is the value **between 0 and 1**. If the result after applying jacquard's distance is **less than** the threshold, the itemset is **frequent or large** otherwise it is infrequent so is pruned.

## 4.8 Software Execution:

This algorithm is implemented using the Visual C#.Net. Modules are created and tested for their effectiveness.

Following are the snapshots of the different phases of execution of project:

### 4.8.1 Login:

When software is executed, a Login form appears which prompts the use to enter user name and password. Two types of access to software are allowed: one for administrator and other for user.

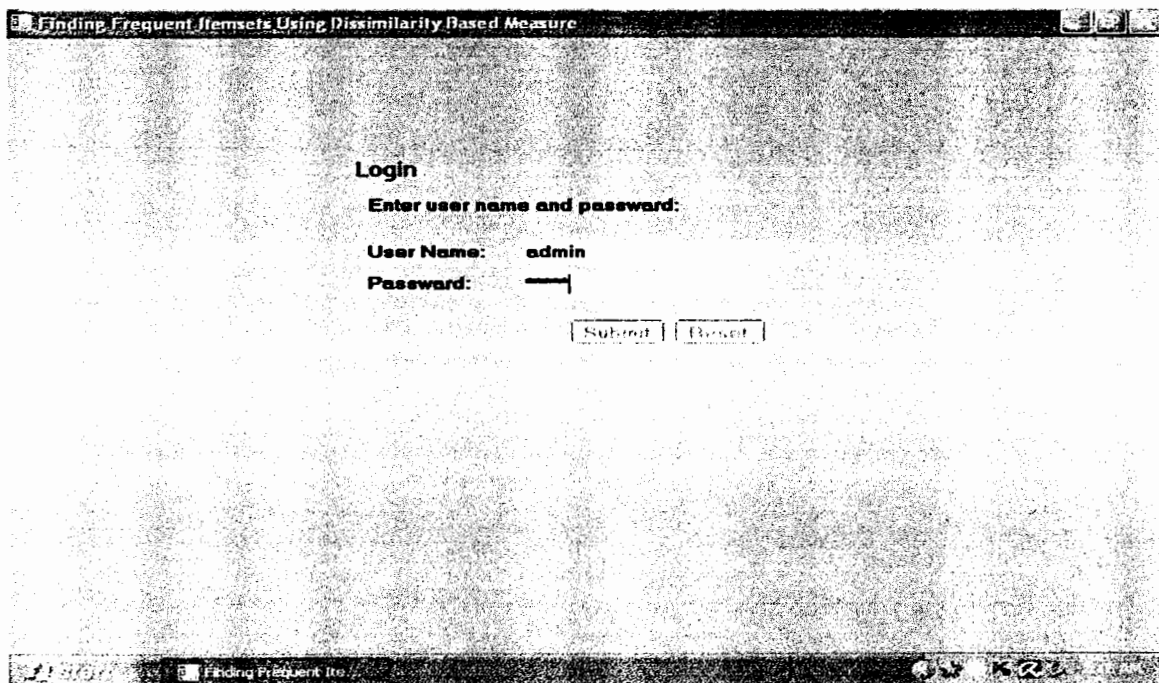


Fig 4.5 Login Form

### 4.8.2 Load Database

When admin/user submit the valid user name and password, a button labeling load database appears:



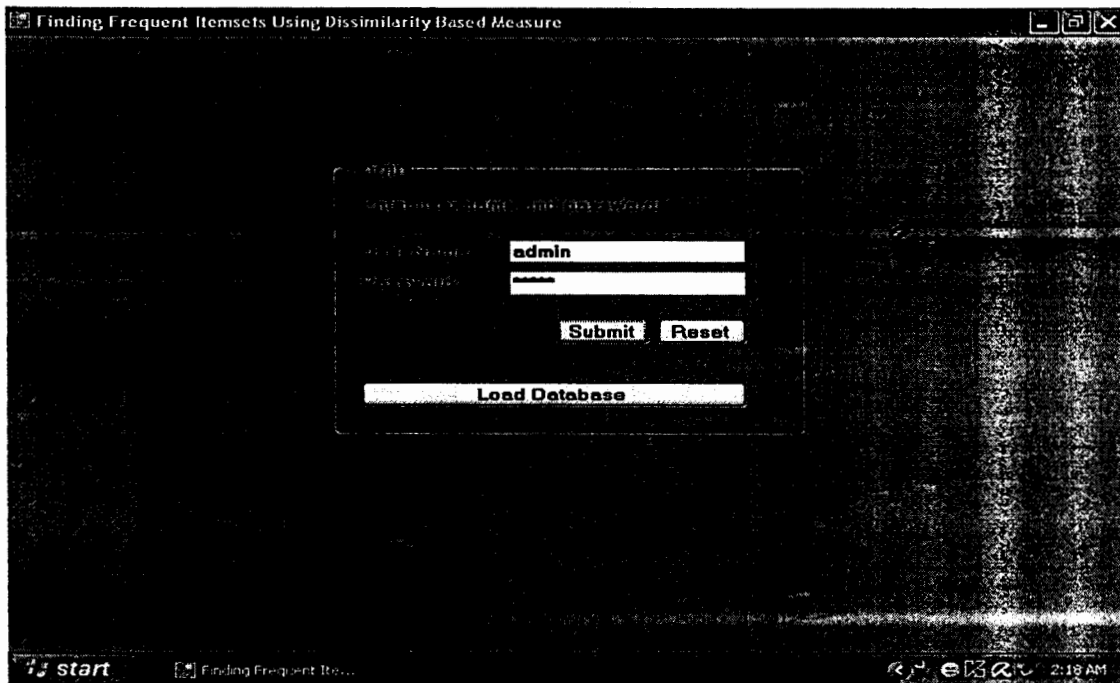


Fig 4.6 Load Database

On clicking it database containing the user transactions table is opened. Following figure shows this database:

Transaction Id	Bread	Egg	Butter	Coke	Milk	Sugar	Burger
1101	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1102	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1103	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1104	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
1105	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1106	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1107	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1108	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1109	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1110	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1111	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1112	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1113	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1114	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1115	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1116	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1117	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
1118	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Fig 4.7 Database loaded

### 4.8.3 Combinations:

Now if the user is administrator, transactions form open with two buttons labeling “Combinations” and “Frequent Item Sets”. Administrator can either select “frequent Item sets” to directly go to the frequent item sets or “combinations” button. On clicking “Combinations”, a form opens with all the combinations of items in the database.

No	Permutations
20	Egg-Burger
22	Butter-Coke
24	Butter-Sugar
26	Butter-Sandwich
28	Coke-Sugar
30	Coke-Sandwich
32	Milk-Burger
34	Sugar-Burger
36	Burger-Sandwich
38	Bread-Egg-Coke
40	Bread-Egg-Sugar
42	Bread-Egg-Sandwich
44	Bread-Butter-Milk
46	Bread-Butter-Burger
48	Bread-Coke-Milk

Total Permutations: 255

Fig 4.8 Combinations

### 4.8.4 Combination Matrix:

On double clicking any combination, its corresponding combination matrix is opened that contains all possible combinations (e.g (milk, butter) and (milk, butter) are two combinations of a combination (milk, butter)) of the combination. The values in the cells show the number of times corresponding combination appears in the database.

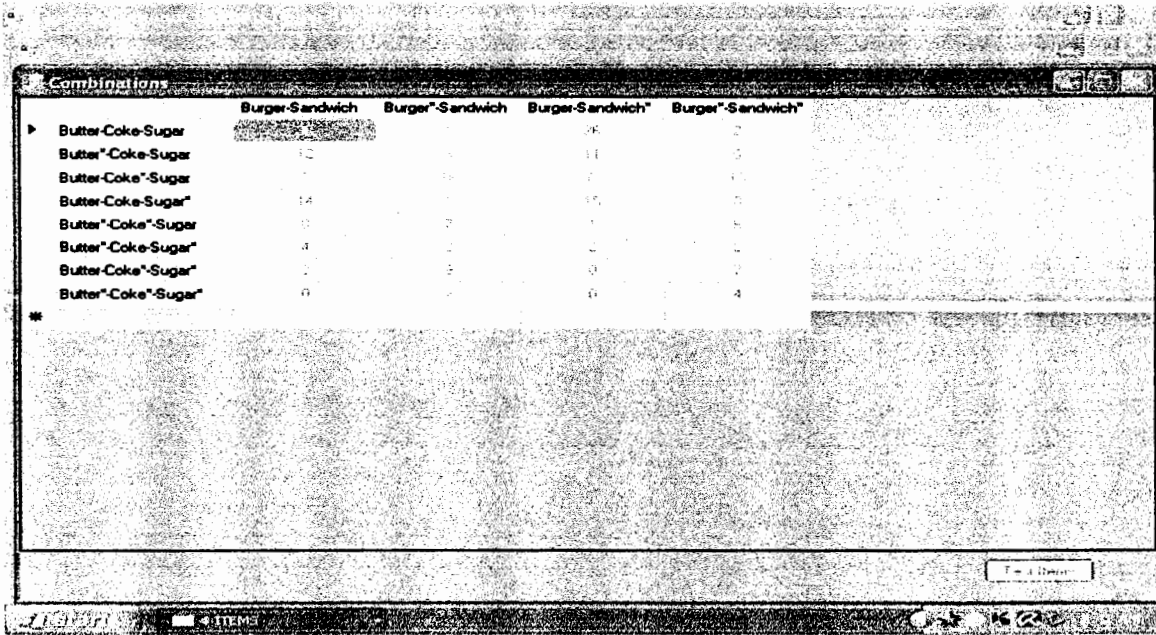


Fig 4.9 Combination Matrix

By clicking on “Test Items”, we get the result at the bottom of form after applying jacquard’s distance equation on the values of the cells and it also shows whether this combination is frequent or not. Combination is shown frequent if resulted value is less than threshold otherwise it is infrequent.

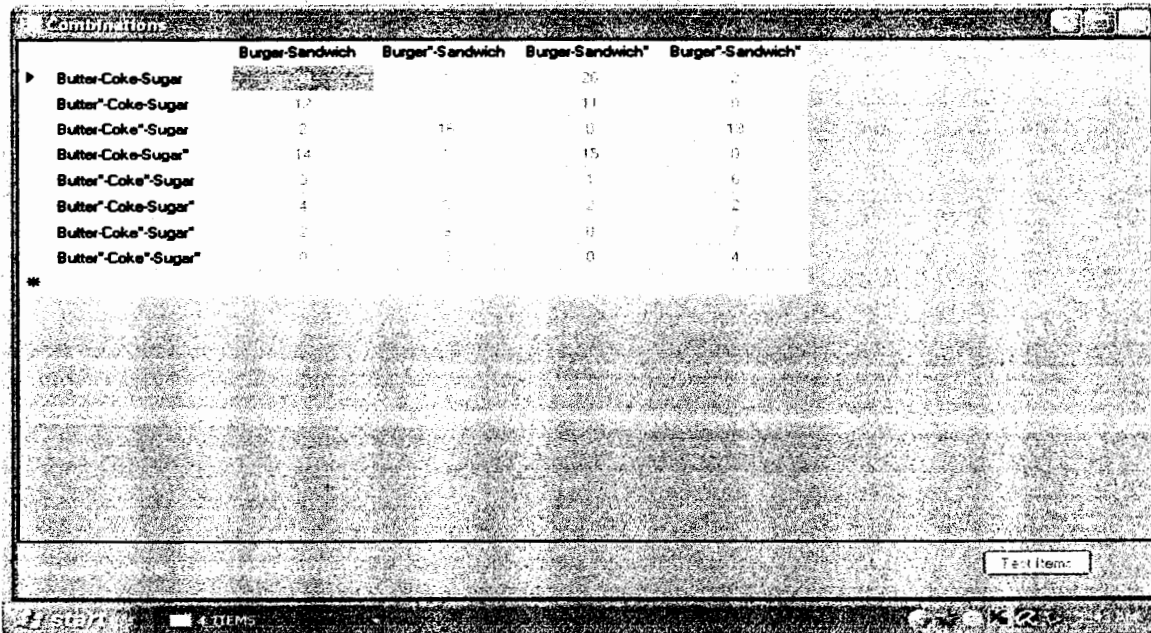


Fig 4.10 Combination Matrix with results

### 4.8.5 Frequent Item Sets:

If the user is not administrator then when database is loaded after logging in, the transactions form appears with only one button labeling "Frequent Item Sets". User clicks that button and gets a list of all frequent item sets which are mined after applying DS-Miner algorithm. Users other than administrator are not allowed to see the complicated working of the algorithm.

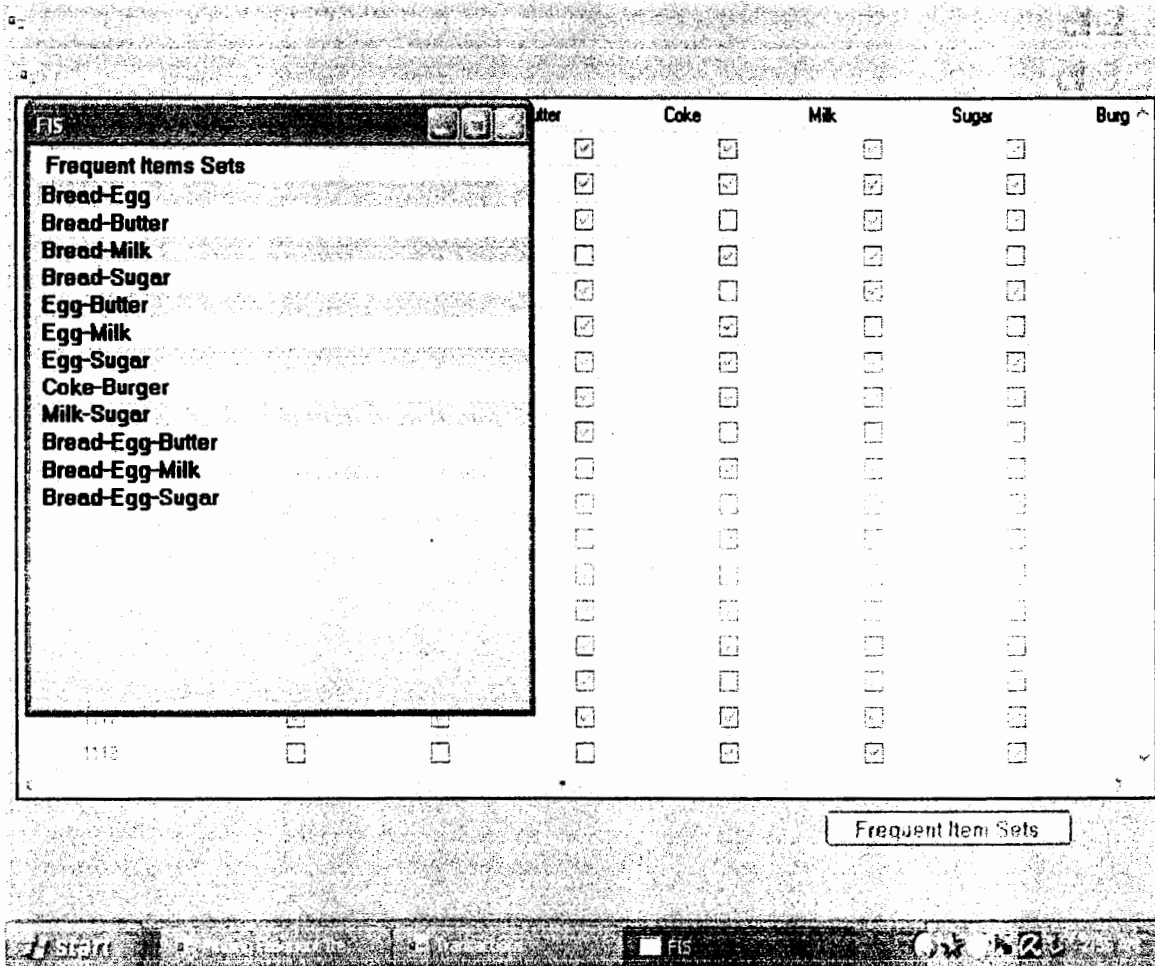


Fig. 4.11 Frequent Item Sets

So, after testing the algorithm with different datasets, we get the results that were required. This proves the effectiveness of the algorithm.

**CHAPTER 5**  
**A COMPARATIVE CASE STUDY**

## 5. A Comparative Case Study

Mining Association Rules is one of the most interesting area for research and as already described, many researchers have worked in this area. Finding frequent itemsets is core of this process. Many techniques presented for this purpose. Most of these are support based and use Apriori algorithm as their base.

To show the efficiency of my proposed technique, i shall provide here a comparative case study. As Apriori algorithm is the base of most techniques, so I shall compare my algorithm with it. I shall take a sample database and apply both algorithms on the same database one by one.

### 5.1 Sample Database:

Transaction_id	Bread	Butter	Coke	Egg
40	Yes	No	Yes	Yes
41	Yes	Yes	Yes	No
42	Yes	No	No	Yes
43	No	Yes	Yes	No
44	Yes	Yes	Yes	No
45	Yes	Yes	No	Yes
46	No	Yes	No	Yes
47	Yes	No	Yes	Yes
48	Yes	Yes	No	Yes
49	Yes	No	Yes	Yes

Fig 5.1 Sample database for comparative case study

### 5.2 Apriori Algorithm Mining Process ( Min. Support = 60% )

First Apriori algorithm is checked on this database. Following are steps for it:

1. **DB is scanned** to count support of each item and we get **candidate 1-set**:

Item set	Support
Bread	8
Butter	6
Coke	6
Egg	7

2. As min. support threshold is 60% so after counting the support of each item, we prune the item with less support and other items make frequent 1-itemset. Here, no item is pruned as no is infrequent. Now we have frequent/ large 1- itemset

Item set	Support
Bread	8
Butter	6
Coke	6
Egg	7

3. Now we concatenate the items in frequent 1-itemset to generate candidate 2-set.

Now we have:

Item set
{Bread, Butter}
{Bread, Coke}
{Bread, Egg}
{Butter, Coke}
{Butter, Egg}
{Coke, Egg}

**DB is again Scanned** to count the support of itemsets in Candidate 2-set.

Item set	Support
{Bread, Butter}	4
{Bread, Coke}	4
<b>{Bread, Egg}</b>	<b>6</b>
{Butter, Coke}	4
{Butter, Egg}	4
{Coke, Egg}	4

4. Now five itemsets will be pruned that have support count less than the min. specified threshold. These are {Bread, Butter} {Bread, Coke} {Butter, Coke} {Butter, Egg} and {Coke, Egg}. After pruning these infrequent itemsets, we have frequent/large 2-itemsets that is:

Item set	Support
<b>{Bread, Egg}</b>	<b>6</b>

Now we have no further candidate itemset. It means, we have frequent itemset as above whose subsets are also frequent.

Database is scanned many times for mining frequent itemsets.



### 5.3 DS-Miner Mining Process:

Now proposed algorithm is applied on the same database. Following are steps for it:

I. First of all this algorithm creates combinations of the items in the database. These are:

{Bread} {Butter} {Egg} {Coke}

{Bread, Butter} {Bread, Egg} {Bread, Coke}

{Butter, Egg} {Butter, Coke}

{Egg, Coke}

{Bread, Butter, Egg} {Bread, Butter, Coke}

{Bread, Egg, Coke} {Butter, Egg, Coke} {Bread, Butter, Egg, Coke}

II. Then for all combinations **Combination-Maps** are created. The values in the cells are the number of transactions that contains corresponding combinations. As following:

	Bread	<u>Bread</u>
Butter	4	2
Butter	4	0

	Bread	<u>Bread</u>
Egg	6	1
Egg	2	1

	Bread	<u>Bread</u>
<u>Coke</u>	5	1
Coke	3	1

	Butter	Butter
Egg	3	4
Egg	3	0
	Egg	Egg
Coke	4	2
Coke	4	0

	Butter	Butter
Coke	3	3
Coke	3	1

	Bread, Butter	Bread, Butter	Bread, Butter	Bread, Butter
Egg	2	1	4	0
Egg	2	1	0	0

	Bread, Butter	Bread, Butter	Bread, Butter	Bread, Butter
Coke	2	1	3	0
Coke	2	1	1	0

	<b>Bread, Egg</b>	<b>Bread, Egg</b>	<b>Bread, Egg</b>	<b>Bread, Egg</b>
<b>Coke</b>	3	0	2	1
<b>Coke</b>	3	1	0	0

	<b>Butter, Egg</b>	<b>Butter, Egg</b>	<b>Butter, Egg</b>	<b>Butter, Egg</b>
<b>Coke</b>	0	3	3	0
<b>Coke</b>	3	1	0	0

	<b>Bread, Butter</b>	<b>Bread, Butter</b>	<b>Bread, Butter</b>	<b>Bread, Butter</b>
<b>Egg, Coke</b>	0	0	3	0
<b>Egg, Coke</b>	2	1	0	0
<b>Egg, Coke</b>	2	1	1	0
<b>Egg, Coke</b>	0	0	0	0

- III. Once all Combination-Maps are created, we can find the frequent itemsets by applying the Jacquard's distance equation on each itemset by taking values from the cells. The equation for two items is :

$$D_{ij} = \frac{(q+r)}{(p+q+r)}$$

And its generalized form for n items is:

$$d(i_1, i_2, i_3, \dots, i_n) = \frac{\sum_{i_1=0}^1 \sum_{i_2=0}^1 \sum_{i_3=0}^1 \dots \sum_{i_n=0}^1 f(i_1, i_2, \dots, i_n) - f(0,0,\dots,0) - f(1,1,\dots,1)}{\sum_{i_1=0}^1 \sum_{i_2=0}^1 \sum_{i_3=0}^1 \dots \sum_{i_n=0}^1 f(i_1, i_2, \dots, i_n)}$$

So now this equation is applied on all combination matrices. **Threshold is 60% ( 0.6)**. It means **dissimilarity is  $\geq 1-0.6 = 0.4$** . So now itemset with result  $< 0.4$  will be frequent otherwise it will be infrequent.

- I. {Bread, Butter}

$$(q+r) / (p+q+r)$$

$$(2+4) / (4+2+4) = 6 / 10 = 0.6$$

As result is greater than dissimilarity threshold that is 0.4, so this itemset is

**Infrequent.**

- II. {Bread, Egg}

$$(2+1) / (6+2+1) = 3 / 9 = 0.33$$

As result is less than dissimilarity threshold, so this is

**Frequent**

III. {Bread, Coke}

$$(1 + 3) / (5 + 1 + 3) = 4 / 9 = 0.44$$

**Infrequent**

IV. {Butter, Egg}

$$(4 + 3) / (3 + 4 + 3) = 7 / 10 = 0.7$$

**Infrequent**

V. {Butter, Coke}

$$(3 + 3) / (3 + 3 + 3) = 6 / 9 = 0.667$$

**Infrequent**

VI. {Egg, Coke}

$$(2 + 4) / (4 + 2 + 4) = 6 / 10 = 0.6$$

**Infrequent**

VII. {Bread, Butter, Egg}

$$(1 + 4 + 0 + 2 + 1 + 0) / (2 + 1 + 4 + 0 + 2 + 1 + 0) \\ = 8 / 10 = 0.8$$

**Infrequent**

VIII. {Bread, Butter, Coke}

$$(1 + 3 + 0 + 2 + 1 + 1) / (2 + 1 + 3 + 0 + 2 + 1 + 1) \\ = 8 / 10 = 0.8$$

**Infrequent**

IX. {Bread, Egg, Coke}

$$(0 + 2 + 1 + 3 + 1 + 0) / (3 + 0 + 2 + 1 + 3 + 1 + 0) \\ = 7 / 10 = 0.7$$

**Infrequent**

X. {Butter, Egg, Coke}

$$(3 + 3 + 0 + 3 + 1 + 0) / (0 + 3 + 3 + 0 + 3 + 1 + 0) \\ = 10 / 10 = 1$$

**Infrequent**

XI. {Bread, Butter, Egg, Coke}

$$(0 + 3 + 0 + 2 + 1 + 0 + 0 + 2 + 1 + 1 + 0 + 0 + 0) / (0 + 0 + 3 + 0 + 2 + 1 + 0 + 0 + 2 + 1 + 1 + 0 + 0 + 0) \\ = 10 / 10 = 1$$

**Infrequent**

Result shows that there is only one frequent itemset in the given sample database which is {Bread, Egg}. This is same result as given by Apriori process. But here number of database scans is less than Apriori. Here database is scanned only once to get the result but in Apriori database is scanned again and again to get the results. So this fulfills the purpose of my research that is to show DS-Miner as effective as Apriori or any other algorithm for Association Rule Mining because it gives same results.

## **CHAPTER 6**

# **CONCLUSION AND FUTURE WORK**

## 6. Conclusion and Future Work

The Dissimilarity Based Mining algorithm introduces a new technique for frequent item set mining. It basically focuses on that clustering measures can be used for association rule mining. Jacquard's Dissimilarity measure is used for generating frequent itemsets in this research. In it proving the efficiency of this algorithm as compare to others is not the task but still it has some advantage.

### 6.1 Introduction:

Database is generated in MS-Access which is used only for this research. DS-Miner is implemented in C#.Net language and is experimented with the database with different number of transactions. Pentium machine with 512 MB RAM and Windows XP operating system are used with user specified threshold value of 0.6 to generate Frequent item sets. If it is to be used with server machine then better results can be obtained.

Following are the results of implementation of DS-Miner on different datasets:

Database	No. of Transactions	No. of Items	FIS
1.	100	8	15
2.	200	8	12
3.	500	8	6

Table 6.1 Experimental Results of DS-Miner



## 6.2 Conclusion:

In previous techniques database was scanned many times for mining frequent itemsets. As in data warehouse huge data is used, so scanning whole database again and again is not efficient. It results in high CPU cost and large number of I/Os. But with the proposed technique, database scanning is reduced and database is scanned only once during whole process. One other advantage of this technique is that it is not support based so it avoids the problems like very high support threshold can prune frequent itemsets and very low support threshold can include infrequent itemsets.

In order to verify the correctness of the algorithm, it is compared with the Apriori algorithm on same dataset. The result of both algorithms was same which shows the effectiveness of the algorithm.

## 6.3 Future Work:

As DS-Miner gives same results as Apriori which is base of many support based algorithms, so it proves that this algorithm is as effective as Apriori. In future I will prove the efficiency of this technique by comparing it with other algorithms for association rule mining.

**REFERENCES  
&  
BIBLIOGRAPHY**

---

## References & Bibliography

1. Oracle® Database Data Warehousing Guide 10g Release 1 (10.1)  
Part Number B10736-01
2. **“Data Mining: Concepts, Models, Methods, and Algorithms”**.  
By Kantardzic, Mehmed , 2003. Published by: John Wiley & Sons.
3. **“Data mining and Knowledge Discovery DBMS, Data Mining Solutions Supplement”**, by Estelle Brand & Rob Gerritsen
4. **“Mining Association Rules Between Sets of Items in Large Databases”**  
by Rakesh Agarawal, Tomasz, Imielinski and Arun Swami. Proceedings of  
the ACM international conference on Management of data, May 1993
5. **“A SURVEY OF ASSOCIATION RULES”**  
By Margaret H. Dunham, Yongqiao Xiao Le Gruenwald, Zahid Hossain
6. **“Fast algorithm for mining Association Rules”**by Agarawal R. & Srikant R.  
VLDB Conference, 1994
7. **“Sampling large databases for association rules”**. By Toivon, In Proc.22<sup>nd</sup>  
VLDB Conference, Bombay, India, 1996
8. **“An efficient algorithm for mining association rules in large databases”**  
By Ashoke Savasere, Edward Omiecinski and Shamkant Navathe.  
Proc. of the 21<sup>nd</sup> International Conference on Very Large Databases,  
Zurich, Switzerland, 1995

9. **“Efficiently mining long patterns from databases”**. By Bayadro, R, J  
In Proc. ACM SIGMOD International Conference on management of data,  
1998
10. **“Mining frequent patterns without candidate generation: A frequent  
pattern tree approach”** . By J. Han, J. Pei, Y. Yin and R. Mao  
Data Mining and Knowledge Discovery, 2004
11. **“Fast Algorithms for Frequent Itemset Mining Using FP-Trees”**  
By Go`sta Grahne, Member, IEEE, and Jianfei Zhu, Student Member,  
IEEE
12. The LUCS-KDD Implementation of the FP-growth algorithm.htm
13. **“ New algorithm for fast discovery of association rules”** by M. J. Zaki,  
S Parthasarathy, M. Ogihara,, Aug, 1997
14. **“ Frequent Pattern Mining (Survey)”**, by Goethals B.  
Department of Computer Science, University of Helsinki. 2003

