

**Finding Frequent Item Sets
Using
Incremental (Interval) Technique**

6376



Developed By

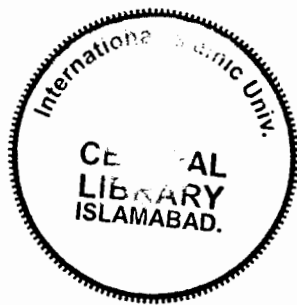
**Abdul Nasir
(392-FBAS/MSCS/F-07)**

Supervised By

Mr. Muhammad Imran Saeed

**Department of Computer Science
Faculty of Applied Sciences
International Islamic University, Islamabad
(2009)**

MS
620 0044
NAF



Accession No. TH-6376

Testing
Laboratories
Measurement

D-E
18-2-11

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

Date: 24-August-2009

Final Approval

This is to certify that we have read the thesis submitted by **Abdul Nasir** 392-MS (CS)-F07. It is our judgment that this thesis is of sufficient standard to warrant its acceptance by International Islamic University, Islamabad for the degree of MS in Computer Science.

Committee:

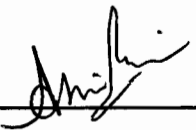
External Examiner

Dr. Nazir Ahmed Sangi
Associate Professor,
Chairman, DCS
Allama Iqbal Open University, Islamabad.



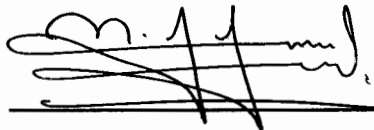
Internal Examiner

Mr. Asim Munir
Assistant Professor,
Department of Computer Science, FAS,
International Islamic University, Islamabad.



Supervisor

Mr. Muhammad Imran Saeed
Assistant Professor,
Department of Computer Science, FAS,
International Islamic University, Islamabad.



**A dissertation Submitted To
Department of Computer Science,
International Islamic University, Islamabad
As a Partial Fulfillment of the Requirement for the Award of the
Degree of MS in Computer Science.**

Dedicated To
The Most Beloved Hazrat Muhammad (S.A.W)
To
My Family
&
To
My honorable Teachers especially Muhammad Imran Saeed

Declaration

I hereby declare that this Research "**Finding Frequent Item Sets Using Incremental (Interval) Technique**" neither as a whole nor as a part has been copied out from any source. It is further declared that I have done this research with the accompanied report entirely on the basis of my personal efforts, under the proficient guidance of my teachers especially my supervisor **Mr. Muhammad Imran Saeed**. If any part of the system is proved to be copied out from any source or found to be reproduction of any project from any of the training institute or educational institutions, I shall stand by the consequences.

Abdul Nasir

392-MSCS/F07

Acknowledgement

First of all I am obliged to Allah Almighty the Merciful, the Beneficent and the source of all Knowledge, for granting us the courage and knowledge to complete this Project.

I am bound to thank **Mr. Muhammad Imran Saeed** for their kind behavior and guidance as supervisor. Beyond doubt, we would never have completed this task without his interest and strict standards of perfection.

I want to thank all the respected faculty members for teaching me in a very professional way.

I want to show my regards and greetings to all of my class fellows and friends with whom I spent a very nice and memorable time.

Above all I owe every thing to my beloved parents and my loving families for their love, guidance and their moral and financial support.

Abdul Nasir

392-MSCS/F07

Project In Brief

Project Title: Finding Frequent Item Sets Using Incremental (Interval) Technique

Undertaken By: Abdul Nasir

Supervised By: Muhammad Imran Saeed

Start Date: 28 February 2009

Completion Date: 24 August 2009

Tools & Technologies: Visual C #.Net And oracle 9i

Documentation Tools: Microsoft Office Word 2003

Operating System: Microsoft Windows XP Professional Version 2002 With Service Pack 2

System Used: Intel® Celeron Processor Pentium 4
CPU 1.80 GHz
RAM 512 MB

Abstract

In the recent years, as whole, data mining has attracted a vast deal of notice in society and in information industry, due to the wide availability of huge amount of data. And Secondly the need for turning such data into useful information and knowledge, which can be used for applications ranging from fraud diction, market analysis and customer's relations to science exploration and production control.

An efficient algorithm/m for association rule mining is one that can provide solution to some of the mentioned factors discussed in literature survey. As the algorithm should use small candidate item sets and generates frequent item sets efficiently, execution time should be minimized. Here, a new technique for finding frequent item sets is present. This is incremental based technique. In this technique generation of item sets are done on interval bases, and then integrate/merge the resultant item sets of different intervals. And at the end globally check the frequent item sets on the basis of support count. An efficient algorithm for association rule mining is one that can provide efficient solution and generates frequent item sets efficiently, execution time should be minimized. The incremental (interval) based technique is a new technique for finding frequent item sets. This technique can generate frequent item sets efficiently and also reuse the combinations of item sets that were created before. Ultimately, we got an efficient algorithm to find frequent item sets.

Table of Contents

Chapters	Page No
<hr/>	
CHAPTER # 1	INRODUCTION
1. Introduction	1
1.1. Data Warehousing	2
1.1.1. Data Warehouse Basic Architecture	2
1.2. Data Mining	5
1.2.1. The Foundation of Data Mining	6
1.2.2. Data, Information and Knowledge	7
1.2.3. Background	7
1.2.4. Classification of data mining	9
1.3. Use of data mining	9
1.4. Association rule mining	10
1.4.1. Frequent item set	10
1.4.2. Association rules	10
1.5. Existing techniques for finding frequent item sets	11
1.6. Scope of the project	13
CHAPTER # 2	LITERATURE SURVEY
2. Literature Surveys	14
2.1. Literature Surveys	15
2.1.1. AIS Algorithms	15
2.1.2. Apriori algorithm for Finding Frequent Itemsets	16
2.1.3. DCI Algorithm	17
2.1.4. Sampling Approach to find FIS	18
2.1.5. Partition Algorithm for finding FIS	18
2.1.6. Maxminer Algorithm	19
2.1.7. FP-Growth Algorithm for Finding Frequent Item Sets	20
2.1.8. Eclat algorithm for finding Frequent Item sets	20

CHAPTER # 3**PROBLEM STATEMENT AND PROPOSED SOLUTION**

3. INTRODUCTION	22
3.1. Problem Statement	22
3.2. Proposed Research	22
3.3. Design/modules	22
3.4. The Hypothesis	23
3.5. Proposed Research Approach	23
3.6. Delimitations and Assumptions of the research work	23

CHAPTER # 4**SYSTEM DESIGN**

4. System Design	25
4.1. Input	25
4.2. Software tool for development	25
4.3. Algorithm	26
4.4. Flow Chart	28
4.5. Design/modules	28
4.5.1. Database Acquisition	29
4.5.2. Combination Creation	29
4.5.3. Merging Candidate Item Sets of Different Intervals ...	31
4.5.4. Frequent Item Set Generation	34

CHAPTER # 5**IMPLEMENTATION**

5. Introduction	35
5.1. Main Form	35
5.2. Integration / Merging	58
5.3. Finding Frequent Item Sets	61

CHAPTER # 6**RESULTS AND CONCLUSION**

6. Results And Conclusion	65
--	-----------

Appendix A

Screen Shots	67
References	71

List of Figures & Tables

Figure/table No.	Caption	Page No
1.1	Data Warehouse Basic Architecture	3
1.2	DWH Architecture with Staging Area	4
1.3	DWH Architecture with Staging Area and Data Marts	5
1.4	Revolutionary process	6
2.1	Eclat Algorithm	21
4.1	Flow Chart of Incremental Algorithm	28
4.2	Database Appearance used for Incremental Algorithm	29
4.3	Sample Database	30
4.4	Database Table "A"	31
4.5	Database Table "B"	32
4.6	Resultant Table after Merging	33
4.7	Frequent Item set Table	34
A-1	Splash Screen	67
A-2	Load Data into the Database	68
A-3	Combinations of "Table A"	69
A-4	Combinations of "Table A" and "Table B"	70
A-5	Details of Possible itemsets	71
A-6	Merging / Integration	72
A-7	Frequent Item set	73

CHAPTER 1
INTRODUCTION

1. Introduction

The demands of the users of the computer systems, from the data they are storing, are increasing day by day, with the explosive growth in information technology. Their demands for the computers systems are increasing because their awareness about the importance of the data for the business and accounts has also increased now they want their computers to think for them assist them in different matters of decision making. For this purpose, several decision support systems are created. [1]

With the passage of time, people have found new ways to increase their business e.g. publicity, good environment, clearness etc. but they have also learnt the use of new technologies. Let us take the simple example of super store ,in past there was no new useful technique to increase their sales and incomes .they only used to make observations and then on those observations , they used to make strategies to develop their business and increase it further. But now many data mining techniques are available for decision making and to cater all these things properly, can also increase their sale much. [1]

In the recent years, as whole, data mining has attracted a vast deal of notice in society and in information industry, due to the wide availability of huge amount s of data and they great need for turning such data into useful information and knowledge, which can be used for applications ranging from fraud diction, market analysis and customers relations to science exploration and production control. [1]

In the Data mining we discover interesting knowledge from large amounts of data store in information repositories, databases or in. Data mining is particularly vulnerable to misuse; with its promise efficiently discover valuable, non obvious information from large data bases. Now, many of problems of the world are modeled as data mining problems. So, we are in need of efficient algorithms for data mining and frequent items sets algorithms are also capturing the interests of various researches. In this research, I also tried to focus on finding frequent items sets efficiently.

1.1.Data Ware housing

Data ware house is single storage repository in which data from all kinds of source is store together through data ware house, it becomes quite easier to analyze the information then the use of multiple data models for multiple sources.

Data ware house is relational databases which specially design for queries, and analysis rather then for transaction processing. It separate analysis workload from transaction work load and it also enables an organization to consolidate data from several sources. It normally contains historical data derived from transaction data, but can also include data from other sources.

Tools for extracting, transformation and loading data are also included in data warehousing [2].

1.1.1. Data Warehouse Architectures

Data warehouses and architecture of data warehouses vary depending upon the requirements and on organization's situation.

Three common architectures of Data Warehouse are:

- Data Warehouse Basic Architecture
- Data Warehouse Architecture with Staging Area
- Data Warehouse Architecture with Staging Area and Data Marts

- **Data Warehouse Basic Architecture**

Figure 1.1 describes the architecture for a data warehouse. Using the several source systems end users can directly access data, In Figure 1.1, in traditional OLTP system raw and metadata is present, as an additional type of data, Summary Data.

Example: Data warehouse query is to retrieve something like monthly sales.

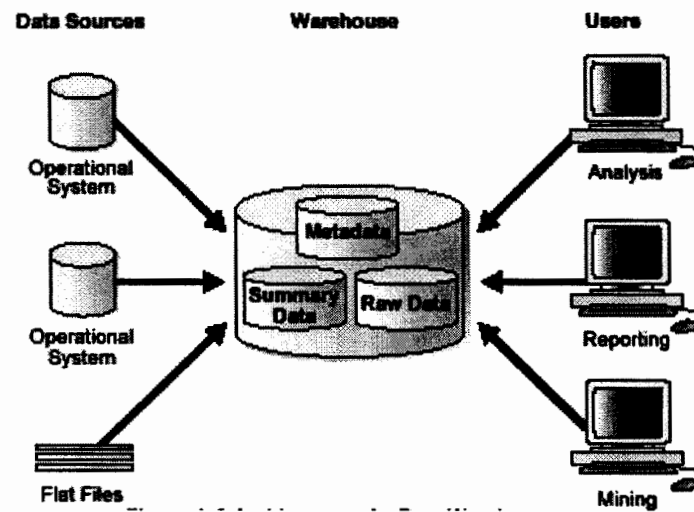


Figure 1.1: Data Warehouse Basic Architecture [3]

- **Data Warehouse Architecture with Staging Area**

In Figure 1.2, you need to clean and process your operational data before putting it into the warehouse. Through Programming you can also do the same thing but mostly use a staging area. In the staging area summaries are build and management of general warehouse is done.

Figure 1.2 shows the staging area architecture.

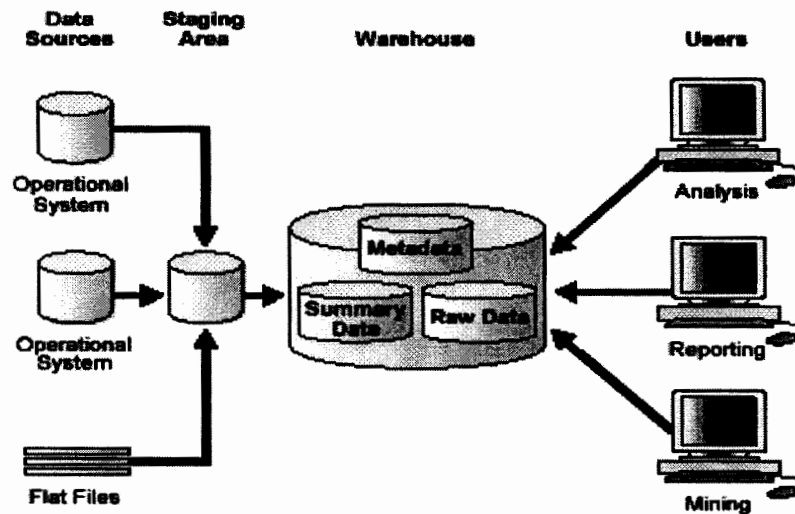


Figure 1.2: DWH Architecture with Staging Area [3]

- **Data Warehouse Architecture with Staging Area and Data Marts**

Although the architecture in Figure 1.3 is quite common, you may want to customize your warehouse's architecture for different groups within your organization. You can do this by using Data Marts, which are systems designed for a particular department or for a particular business function. Figure 1.3 demonstrates an example where Sales, Purchasing, and Inventories are separated. For example, a financial analyst might need historical data of purchase and sales for analysis.

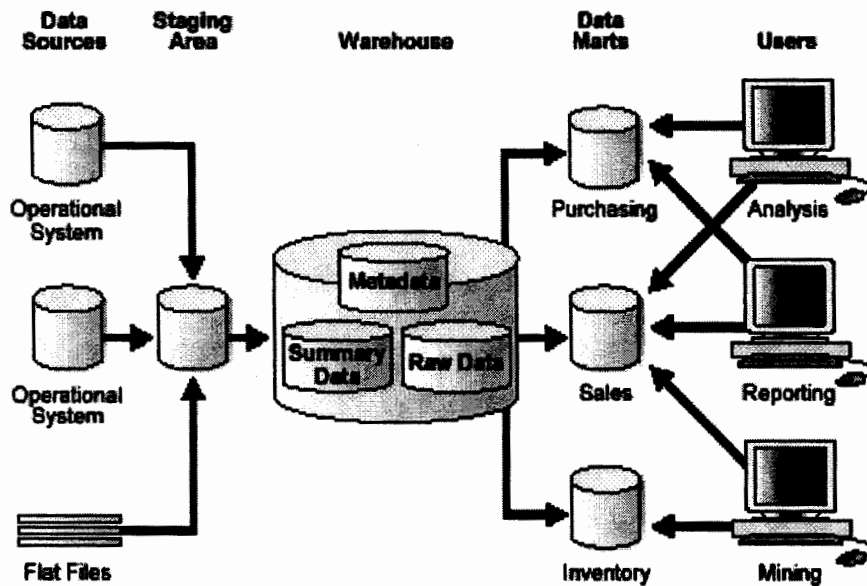


Fig 1.3: DWH Architecture with Staging Area and Data Marts [3]

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 sort large amount of data and gets your related information. This information normally used by business intelligence organizations, and financial analysts, but is increasingly being used in the science to extract information from the enormous data sets generated by modern experimental and observational methods. [4]

From the above definition, it is very obvious that data mining is the process of analyzing data from different perspective, and also summarizing it into useful information or knowledge that can be used to increase revenue, cuts costs and both.

For example, one grocery store used data mining tool and analyzed buying patterns .they pointed out through an experimental that when men bought diapers and also tended to buy beer on Thursday and Saturdays. Further analysis shows that these shoppers did their grocery shopping on Saturdays. On Thursday, they only bought a few items. The grocery claim could use this newly discovered information in many ways to revenue .e.g. They

could move the beer closer to the diaper display .and could make sure, beer and diapers were sold at full piece on Thursdays. They also could place chips/snacks between the diapers and beers, similarly they could place soda close to these items.

1.1.1. The Foundation of Data Mining

Data mining techniques are the results of long process of research and product development. Data mining takes this evolutionary process beyond retrospective data access and also navigation to perspective and proactive information delivery.

Each new steps, in the evolution from business data to business information, has built upon the previous one e.g. in data navigation applications, dynamic data access is critical for drill.

The four step revolutionary process is shown in table 1.

Evolutionary step	Enabling technologies	Characteristics
Data collection (1960s)	Computers, tapes, disks	Retrospective, static data delivery
Data access (1980s)	Relational databases(RDBMS),structures query language(SQL)	Retrospective, dynamic data delivery at record at record level
Data warehousing & Decision Support (1990s)	On-line analytic processing(OLAP),multidimensional databases, data warehouses	Retrospective, dynamic data delivery at multiple level
Data mining	Advanced algorithms, multiprocessor computers, massive databases	Prospective, proactive information delivery

Fig 1.4: Revolutionary process [1]

1.1.2. Data, Information and Knowledge

- **Data**

Data includes any facts, numbers, or text that can be processed by a computer, Data can be

- Operational or transactional data such as, sales, cost, inventory, payroll, and accounting
- Meta data – data about the data itself, such as logical database design or data dictionary definitions.

- **Information**

The patterns or relationships among all this data can provide information e.g. by analyzing sale transaction data, we yield all the information, on which and when products are selling.

- **Knowledge**

Information about historical patterns and future trends can be converted into knowledge e.g. Summary information on retails grocery shop sales can be analyzed to provide knowledge of consumer buying behavior. Hence a manufacturer or retailer could be determining which items require special offers and display etc.

Data mining tools are useful because they can predict future trends and behaviors, decisions. It can also answer those questions which were considered to be most difficult resolve in the past .data mining tools can also search databases for finding hidden patterns, predictive information that can be missed by the experts because it's behind the expectations.

1.1.3. Background

Traditionally, many useful tasks of extracting useful information from recorded data were performed by business analysts, but now the increasing data in modern business and science calls fro computers based approaches. There has been a shift away from direct hands on data analysis on data analysis toward indirect automate data analysis using

more complex and sophisticated tools , as data sets have grown in size complexity. Data collection and organization is made much easier due to the modern technologies of computers, networks, and sensors. However to make the captured data useful, it needs to be converted into information and knowledge. Thus, data mining is the entire process of applying computer based methodology, including new techniques for knowledge discovery, to data.

Data mining term is often in the two separate processes of prediction and knowledge discovery. Knowledge discovery provides explicit information that has readable form and can be understood by user. Neural networks and predictive modeling provides predications of the future events. Moreover, some data mining systems such as neural networks are geared towards prediction and pattern recognitions, rather than knowledge discovery. [4]

Metadata or data about a given dataset are usually expressed in a condensed data min able format or one that facilitates the practice of data mining. For example it concludes or includes executive summaries and scientific abstracts.

Data mining use real world data. It is extremely vulnerable to co linearity precisely because data from the real world may have unknown interrelation. The critical data that may expose any relationship might have never been observed through this data. So, it is a great weakness of this data mining. There are some of the alternative approaches, such as: experimental based approach, which may be used. Through the construction of an experimental design, inherent correlations are either controlled or removed altogether.

1.1.4. 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 the patient’s primary tumor?”[5]

- **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,”[5]

1.2. Use of data mining

Market basket analysis is the most important and well-known use of data mining.

If super store records the purchases of customer data mining system could identify those customers who favor milk over coke. The example used transaction based data. Not all data, are transaction based and logical or in exact rules within a database.

In recent years, data mining has been widely used in the areas of science and engineering, such as bioinformatics, genetics education and electrical power engineering.

1.3. Association rule mining

The most important concept in data mining is association rule mining. It comes under descriptive mining. It finds interesting association or correlations relationships among large set of data items. A widely used example of association rule mining is market basket analysis.

Association rule mining has two steps

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

1.3.1. Frequent itemset

It is the first step in association rule mining. It is the main task because once the frequent item set has been formed; it is straight to generate association rules.

In many cases, we only care about association rules or causalities involving sets of items that appear frequently in baskets. For examples: we can't run a good marketing strategy involving items that are not much profitable, thus, much data mining starts with this assumption that we only care about sets of items with high support set of items (i.e., $f(X_1, \dots, X_n; Y)$ must appear in at least a certain percent of baskets, called the support threshold.

1.3.2. Association rules

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

For Example, data are collected from bar code readers in supermarkets. Such 'market basket' databases contain large number of transactions. Each record lists the entire item bought by a customer on a single transaction. It's interesting news for the managers to know that if certain groups of items are consistently purchased together. They could use

this data for adjusting store layouts, for cross selling, for promotions, and for catalog designing.

Association rules gives information in the form “if-then” statements. These rules are computed from the data and, unlike if then rules are probabilistic in nature.

1.4.Existing techniques for finding frequent itemsets

As frequent item set mining is the main task in association rule mining. So many researchers have concentrated on this step most.

Following are some techniques proposed by many researchers finding frequent item sets.

- **Apriori** is an extension of AIS algorithm proposed by Agrawal et al [8]. It is support based algorithm. In it closure property is used that any subset of a 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 minimum support threshold. These frequent itemsets are used to generate a new candidate itemsets. This procedure continues until no further items remain
- **DCI Algorithm** this approach was proposed by Orlando et al [2] for solving the frequent set counting problem. Like as Apriori, DCI adopts a level-wise approach. DCI adopts a hybrid approach to compute the supports of itemsets, by exploiting a counting-based method during the first iterations, and an intersection-based method during the last ones. DCI enhances Apriori by using an innovative method for storing candidate itemsets and counting their support, and by exploiting effective pruning technique which reduce the size of the dataset as execution progresses. as soon as the pruned dataset becomes small enough to fit into the main memory, DCI builds on the fly vertical transaction database, and starts using an efficient intersection –based technique to determine the support of larger itemsets.

- **Partition** algorithm makes the partition 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 item sets.[11]
- **Eclat** algorithm is also support based and it as an extension of Apriori algorithm. It basically uses vertically database layout. Horizontal database is converted into vertical format for each item tid-list is stored. Support of itemset is determined by interesting 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.[15]
- **Sampling** approach for finding frequent itemsets is proposed by Toivonen et al [10]. 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** 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. Divide and conquer methodology is used so mining task is divided into smaller parts.[13]
- **Maxminer** algorithm is used for finding maximal frequent itemsets. In it breadth-first traversal is used. By employing look ahead pruning strategy, it reduces database scans. [12]

1.5. Scope of the project

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

- Finding frequent/large item sets
- Generating association rules from these frequent item sets

Main devotion and importance is given to it as finding item sets is the core of this. All previous techniques have following problems.

- 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.
- Execution time is large

In this research work, a new technique is introduced for finding frequent item sets that will use interval/incremental technique.

CHAPTER 2
LITERATURE SURVEY

2. Literature survey

Data mining is the process of extracting only useful data and important information from rocks of data. In the first step, data is transferred into patterns, and then rules are applied to extract information from these patterns. For decision making this information may be used.

Association rule mining is important and useful data mining model studied extensively by the database and data mining community. On the other hand, association rule analysis is used for market basket analysis, in which association rule mining 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 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 storing 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 has two basic 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 depicts that there are two main/ important factors in association rules that are confidence and support.

If association rules satisfy both a minimum confidence threshold and a minimum support threshold, then are considered interesting. Such threshold is set by user or domain experts.

2.1. Literature Surveys

Finding frequent item sets is the first step in association rule mining. Subsequent literature was reviewed.

1. AIS algorithm for finding FIS
2. Apriori algorithm for finding frequent item sets
3. DCI algorithm
4. Sampling approach for finding frequent item sets
5. Partition algorithm
6. Maxminer algorithm
7. FP-Growth algorithm
8. Eclat algorithm

There is a lot of material in books and internet which helped me in my work.

Following is the brief description of above algorithms.

2.1.1.AIS Algorithms

This algorithm was proposed by Agrawal et al [6]. It was first algorithm to find all frequent item sets in a transactional database [Agrawal 1993]. The AIS algorithm makes many scans of the database. Multiple passes through the entire database are done. 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 item sets of each pass are extended to generate candidate item sets. After scanning transactions, the common item sets between large item sets of the previous pass and items of this transaction are determined. These common item sets are extended with other items in the transaction to generate new candidate item sets. [7].

This process continues until no more candidates item set remains. As in this algorithm all frequent and candidate item sets 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 item sets that have never been extended and other approach is to delete candidate item sets that have maximal number of items and their siblings and store this parent item sets in the disk as seed for the next pass.

AIS algorithm has some disadvantages like:

- It generates too many candidate item sets which finally turned out to be infrequent/small. This requires more space and wastes much effort.
- Another problem is too many database scans.

2.1.2. Apriori algorithm for Finding Frequent Itemsets

This algorithm for finding frequent item sets (FIS) was proposed by Agrawal et al [8]. It is an extension of AIS algorithm for finding frequent item sets which requires many database scans, many candidate generations, and store counter of each candidate, Apriori algorithm is more efficient than AIS.

In Apriori algorithm first candidate 1-item sets is generated and then database is scanned to calculate their support count .large/frequent-1 item sets are generated and all item sets not satisfying minimum support threshold are pruned. Then candidate 2 item sets are generates by joining items in frequent-1 item set. It stores the candidate item sets in hash tree. Then database is again scanned to check support count of each item set in candidate 2 item set. This process continues until all items are checked.

In finding FIS, Apriori avoids the effort wastage of counting the candidate item sets that are known to be infrequent. The candidate sets are generated by joining the frequent item sets 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 has still the drawback that it scans whole database again and again during processing. Its main drawback lies 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 transactions, 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 for each of them whether it is a candidate set, can take a prohibitive amount of time.

Test 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 different from Apriori after first database scan. It uses pair of item set and its TIDs for counting their support after first scan.

2.1.3.DCI Algorithm

This approach was proposed by Orlando et al [9] for solving the frequent set counting problem. Same as Apriori algorithm, DCI also use level-wise approach. DCI adopts a hybrid approach to compute the supports of itemsets, by exploiting a counting-based method during the first iterations, and an intersection-based method during the last ones. DCI enhances Apriori by using an innovative method for storing candidate itemsets and counting their support, and by exploiting effective pruning technique which reduce the size of the dataset as execution progresses. as soon as the pruned dataset becomes small enough to fit into the main memory, DCI builds on the fly vertical transaction database, and starts using an efficient intersection –based technique to determine the support of larger itemsets.

Unfortunately, for problems datasets from which long patterns can be mined, we have explosion of the number of candidate and frequent itemsets, since all the 2^L subsets of each long maximal frequent itemset of length L have to be produced. In this case,

counting-based approach becomes very expensive, since the supports of frequent itemsets become very large, and the various candidate turn out to be set-included in a lot of transaction.

2.1.4.Sampling Approach to find FIS

This approach was proposed by Toivonen et al [10] for finding frequent item sets. It is applied on large databases. In this approach a random sample is taken from the database and frequent item sets are searched from that sample. In this approach only one database scan is required because the size of sample is taken that can be accommodate in main memory. This approach is beneficial when efficiency is of utmost importance.

The problem in sampling is that the candidate set derived is necessarily a superset of the actual set of frequent set and may contain many false positives. And May not a representative sample.

2.1.5.Partition Algorithm for finding FIS

Partition algorithm is presented in [11].This algorithm uses two database scans:

- In first scan, all potentially frequent item sets are generated .this set of frequent item sets is superset of all frequent item sets.
- In the second scan of database, counters are created for all these item sets 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 item sets are generated from each partition separately one by one and at the end these frequent item sets from all partitions are merged to make one set of all potentially frequent item sets. Then actual support for these item sets is counted to identify frequent/large item sets by making second database pass.

The partitions sizes are chosen so that each partition can be accommodated in main memory. These are only two database scans in whole process. The key observation used is that a globally frequent item set 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 item sets. On the other hand, if the entire database fits into main memory then dividing the transactions into many partitions is not necessary.

2.1.6. Maxminer Algorithm

This algorithm is used to find maximal elements [12]. It employs the breadth-first traversal of the search space. It assumes that entire database fits into main memory. Maximal frequent item set is one that has no frequent superset.

e.g.

(Minimum support = 3)

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

$\{1, 2\}$ is frequent as it is present in four transaction i.e. larger than minimum support.

$\{1, 2, 3\}$ is maximally frequent as this set has no superset that satisfy the minimum support.

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

Maxminer algorithm is applied on the original horizontal database. Database scans are same as in Apriori.

2.1.7.FP-Growth Algorithm for Finding Frequent Item sets

It was proposed by J.HAN et al [13].This algorithm uses support-based measured to find maximal frequent item sets using I-projected databases. This algorithm compresses a large database into a compact, FP tree structure.

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

- In the first scan, it finds all frequent item sets.
- In second scan, it constructed 1st FP-tree that contains all frequent information of the given dataset.

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 combination of frequent item sets. This technique 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 [14].

2.1.8.Eclat algorithm for finding Frequent Item sets

This algorithm is same like FP-Growth algorithm .It uses support based measures to find maximal frequent item sets. Difference is that it uses different data structures, it uses vertical layout and uses the intersection based approach to compute the support of an item set. [15]

Horizontal Data Layout

TID	Items
1	A,B,E
2	B,C,D
3	C,E
4	A,C,D
5	A,B,C,D
6	A,E
7	A,B
8	A,B,C
9	A,C,D
10	B

Vertical Data Layout

A	B	C	D	E
1	1	2	2	1
4	2	3	4	3
5	5	4	5	6
6	7	8	9	
7	8	9		
8	10			
9				

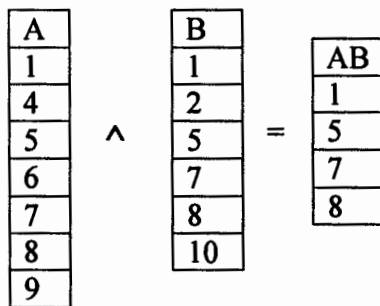


Fig 2.1: Éclat Algorithm [16]

Éclat is different from Apriori algorithm in pruning step. All the items in the database are recorded in ascending order with respect to the support count [16]. Eclat counts the support of all item sets more efficiently than Apriori algorithm.

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

CHAPTER 3
PROBLEM STATEMENT AND PROPOSED
SOLUTION

3. Introduction

After literature survey I identified gap in the previous algorithms or in the techniques.

3.1. Problem Statement

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:

The constantly increasing data size and time to time refreshment of the data repositories increases time-complexity of the Association Rule Mining process. So, when there is large set of candidates or if the size of the candidate item set is very large, it can degrade the performance. Size of the candidates directly affects the performance.

3.2. Proposed Research

An efficient algorithm for association rule mining is one that can provide solution to some of the above mentioned factors like the algorithm should use small candidate item sets and generates frequent item sets efficiently, execution time should be minimized.

Here, a new technique for finding frequent item sets is present. This is incremental based technique. In this technique generation of item sets are done on interval bases. And then merging the resulting item sets of intervals. And at the end globally check the frequent item sets on the basis of support count.

3.3. Design/modules

- Generation of candidate item sets locally (for each interval).
- Merging candidate item sets of different intervals.
- Frequent item set generation

3.4. The Hypothesis

“By using the incremental or interval base technique, it is more efficient to find Frequent Item sets in the Association Rule mining process.”

3.5. Proposed Research Approach

To conduct a research work a researcher can adopt various research methods like the Experiment, Survey, Simulations, Case-Study or the Ethnography depending on the nature of the research and suitability of the research method. In order to Find Frequent Item Sets in the Association Rule Mining through Incremental/interval base technique, it was required to conduct an experiment on the synthetic data.

As for as other research methods are concerned, they are not suitable for this specific problem. Like the ‘Survey’, was not applicable because according to my knowledge this approach is not adopted before this by any researcher or data mining expert, hence. As for as the method of ‘Case-Study’ is concerned, it was also not feasible currently because it is too earlier to implement it this technique in a specific domain before applying it to the synthetic data. Similarly the Ethnography was not applicable in this case.

3.6. Delimitations and Assumptions of the research work

Data Mining requires a single, separate, clean and consistent source of data. So to perform the data mining activities, it is assumed that clean and consistent data is available. It is assumed that transactions are provided in ordered binary format.

The data has to be extracted in a periodic manner not only once. And to keep up-to-date supply changed data to the data warehouse. It should be based upon the loading paradigm of the Data warehouse. Most data warehouses are loaded in a periodic manner. For example, may be weekly, monthly or may be on every night, new data is added into the data warehouse. And then it is used for finding frequent item set mining. We assume that

if we keep on mining the data as soon as it comes into the data warehouse then it will evaluate efficiently.

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 the support measures (that is used in previous algorithm).

4.1. Input

Input is a large transactional database. The database is the exclusively created synthetic database, which is used to represent the actual transaction database's items. Items of the transactions are in the form of "zero" or "one". "One" means that item is present in the transaction and "zero" means that item is not present in transaction.

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 careful to be relatively appropriate for front-end and Oracle 9i at the back-end. Windows-XP operating system is used for development.

4.3. Algorithm

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

Input

D: Transactional Database

@: User specified threshold

Output

F: Frequent itemsets

Steps

```

For ( i=1 to n partitions)
{
  L(i)=GenerateCandidate(D)
  If(i>1) then
    AG=Aggregate (AG,L(i))
}
For each itemset AG
{
  If(frequency<minthreshold)
  Delete c
}

```

```

Procedure GenerateCandidate(D)
{
  L1=FindFrequent1ItemSets(D)
  For (k=2 ; Lk-1 =  $\phi$  ; k++)
  {
    Ck=Generate(Lk-1)
    For each transaction t  $\in$  D
    {
      Ct=Subset (Ck,t)
      For each candidate C  $\in$  Ct
      C.Count ++
    }
    Lk={C  $\in$  Ck / C.Count >= 1}
  }
  Return L= UkLk
}

```

Procedure Generate (L_{k-1} : frequent (k-1)-itemsets)

```

{
  For each itemset  $l1 \in L_{k-1}$ 
    For each itemset  $l2 \in L_{k-1}$ 
      {
        If ( ( $l1[1]=l2[1]$ )  $\wedge$  ( $l1[2]=l2[2]$ )  $\wedge$  ...  $\wedge$  ( $l1[k-2]=l2[k-2]$ )  $\wedge$  ( $l1[k-1]=l2[k-1]$ ) )
          {
             $C = l1 \cup l2$ ; //join step
            Add C to  $C_k$ 
          }
      }
    }
  Return  $C_k$ 
}

```

Procedure Aggregate (AG,L)

```

{
  Merge into AG a
  Using L b
  On ( a.itemset=b.itemset)
  When matched then
    Update set a.frequency = (a.frequency + b.frequency)
  When not matched then
    Insert values ( b.itemset , b.frequency)

  Return AG
}

```

4.4. Flow Chart

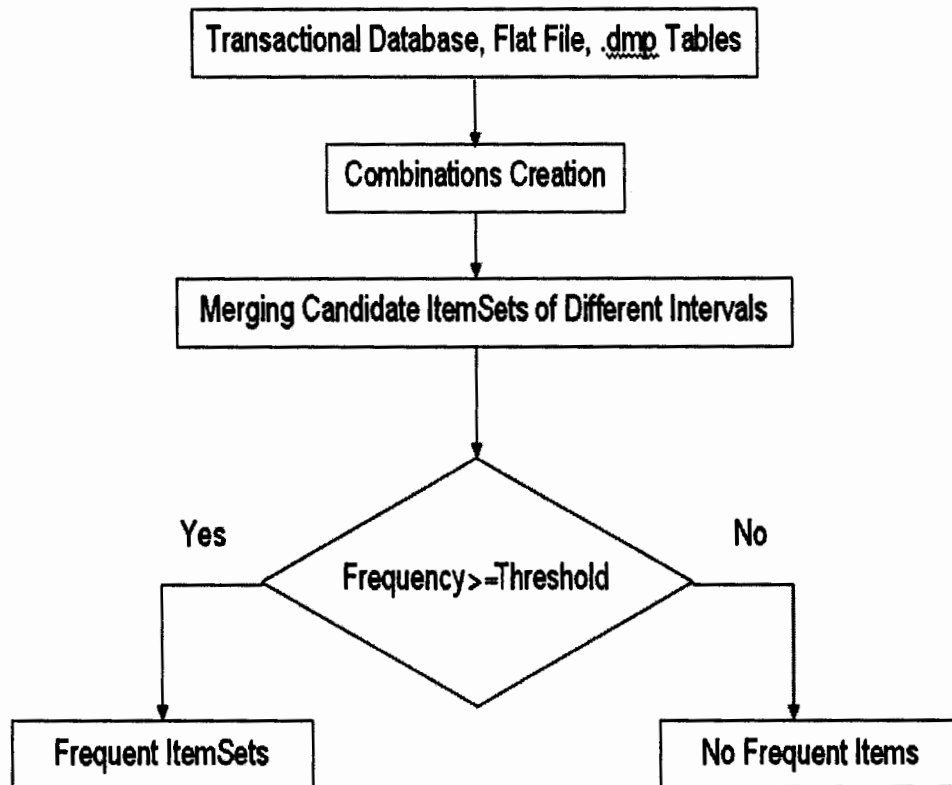


Fig 4.1: Flow Chart of Incremental Algorithm

4.5. Design/modules

- 1 Database Acquisition
- 2 Generation of candidate item sets locally (for each interval).
- 3 Merging candidate item sets of different intervals.
- 4 Frequent item set generation

4.5.1.Database Acquisition

The input to this project is synthetic database which is created in MS Access exclusive for this project. The records are in the form <TID> < 1 or 0 (showing item presence or absence)> as following

TID	MILK	BREAD	BUTTER	HONEY	EGGS	BISCUIT	MILKCREA M	YOGURT	COFFEE	TEA_BAGS
1	1	1	0	1	0	0	1	0	1	1
2	0	0	1	1	0	1	0	1	1	1
3	0	0	0	0	0	0	1	0	1	0
4	0	0	1	0	1	1	1	1	0	0
5	1	0	1	1	0	1	1	1	0	0
6	0	1	0	1	1	0	0	0	0	0
7	1	1	0	1	0	0	0	0	1	1
8	0	0	0	0	1	0	0	0	0	1
9	1	0	1	0	1	1	1	1	0	1
10	0	1	0	0	0	1	1	0	1	0

Fig 4.2: Database Appearance used for Incremental Algorithm

4.5.2.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, Butter, Bread) then Combinations are:

(Milk),(Butter),(Bread),(Milk, Butter),(Milk, Bread),(Bread, Butter),(Milk, Butter, Bread)

We consider only those combinations that have value one.

MILK	BREAD	BUTTER	HONEY	EGGS	BISCUIT	MILKCREA M	YOGURT	COFFEE	TEA_BAGS
1	0	1	1	0	1	0	0	0	0
0	0	0	1	1	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0
1	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	1	1	1	0
0	0	0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	1	1	1
0	1	0	0	0	0	0	0	0	1
0	0	0	0	1	1	0	1	1	0
1	0	1	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	1
0	1	0	1	1	0	0	0	0	0
0	0	1	1	0	0	1	0	0	0
1	1	0	0	0	0	0	1	1	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1
0	1	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0
0	0	0	1	0	0	0	1	0	1
0	1	0	0	1	0	1	0	0	0
1	0	0	1	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0	0	1
1	0	1	1	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	1
0	0	1	1	0	1	0	0	0	0
0	0	0	1	1	0	0	0	0	0
0	0	0	1	0	0	1	0	0	1
0	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	1	0
1	0	1	1	0	0	0	0	0	0

Fig 4.3: Sample Database

4.5.3. Merging Candidate Item Sets of Different Intervals

There are two Table "A" and "B"

E.g. we have combinations of table "A"

MILK	BREAD	BUTTER	HONEY	EGGS	BISCUIT	MILKCREAM	YOGURT	COFFEE	TEA_BAGS	TOTAL
1	0	1	1	0	1	0	0	0	0	7
0	0	0	1	1	0	0	0	1	0	14
0	0	0	0	0	0	0	1	0	0	2
1	0	0	0	1	0	0	0	0	0	1
0	0	0	1	0	0	1	1	1	0	9
0	0	0	0	1	0	0	0	1	0	11
0	1	0	0	0	1	0	1	1	1	4
0	1	0	0	0	0	0	0	0	1	3
0	0	0	0	1	1	0	1	1	0	8
1	0	1	1	0	0	0	0	0	0	9

Fig 4.4: Database Table "A"

E.g. we have combinations of table “B”

MILK	BREAD	BUTTER	HONEY	EGGS	BISCUIT	MILKCREAM	YOGURT	COFFEE	TEA_BAGS	TOTAL
0	0	0	1	0	0	1	1	1	0	9
0	0	0	0	1	0	0	0	1	0	11
0	1	0	0	0	1	0	1	1	1	4
0	1	0	0	0	0	0	0	0	1	3
0	0	0	0	1	1	0	1	1	0	8
1	0	1	1	0	0	0	0	0	0	9
1	0	0	0	0	0	0	0	1	0	10
0	0	1	1	0	0	0	0	0	0	15
0	0	0	0	1	0	1	0	0	1	4
0	1	0	1	1	0	0	0	0	0	6
0	0	1	1	0	0	1	0	0	0	11
1	1	0	0	0	0	0	1	1	0	8
0	0	1	1	0	0	0	0	0	0	4
0	0	0	0	1	0	0	0	0	1	2
0	1	0	0	0	0	0	0	1	0	10
1	0	1	1	0	1	0	0	0	0	7
0	0	0	1	1	0	0	0	1	0	14
0	0	0	0	0	0	0	1	0	0	2
1	0	0	0	1	0	0	0	0	0	1

Fig 4.5: Database Table “B”

After Merging Table "A" And "B"

MILK	BREAD	BUTTER	HONEY	EGGS	BISCUIT	MILKCREAM	YOGURT	COFFEE	TEA_BAGS	TOTAL
0	0	0	1	0	0	1	1	1	0	18
0	0	0	0	1	0	0	0	1	0	22
0	1	0	0	0	1	0	1	1	1	8
0	1	0	0	0	0	0	0	0	1	6
0	0	0	0	1	1	0	1	1	0	16
1	0	1	1	0	0	0	0	0	0	18
1	0	0	0	0	0	0	0	1	0	10
0	0	1	1	0	0	0	0	0	0	15
0	0	0	0	1	0	1	0	0	1	4
0	1	0	1	1	0	0	0	0	0	6
0	0	1	1	0	0	1	0	0	0	11
1	1	0	0	0	0	0	1	1	0	8
0	0	1	1	0	0	0	0	0	0	4
0	0	0	0	1	0	0	0	0	1	2
0	1	0	0	0	0	0	0	1	0	10
1	0	1	1	0	1	0	0	0	0	14
0	0	0	1	1	0	0	0	1	0	28
0	0	0	0	0	0	0	1	0	0	4
1	0	0	0	1	0	0	0	0	0	2

Fig 4.6: Resultant Table after Merging

4.5.4.Frequent Item Set Generation

After merging the candidates of different intervals, then it is straight forward to find frequent itemsets. The threshold is specified by the user .On the basis of threshold, pruned those itemsets that are less than threshold (infrequent itemsets).Remaining item sets are frequent.

For example we use 10 as threshold then frequent item sets are:

MILK	BREAD	BUTTER	HONEY	EGGS	BISCUIT	MILKCREAM	YOGURT	COFFEE	TEA_BAGS	TOTAL
0	0	0	1	0	0	1	1	1	0	18
0	0	0	0	1	0	0	0	1	0	22
0	0	0	0	1	1	0	1	1	0	16
1	0	1	1	0	0	0	0	0	0	18
1	0	0	0	0	0	0	0	1	0	10
0	0	1	1	0	0	0	0	0	0	15
0	0	1	1	0	0	1	0	0	0	11
0	1	0	0	0	0	0	0	1	0	10
1	0	1	1	0	1	0	0	0	0	14
0	0	0	1	1	0	0	0	1	0	28

Fig 4.7: Frequent Item set Table

CHAPTER 5
IMPLEMENTATION

5. Introduction

This algorithm is implemented using Visual C#.Net as front end and Oracle as Backend. I present the Implementation part of the algorithm in this phase.

Basically there are three steps

- Create combinations
- Integration /Merging
- Finding frequent Item Sets

5.1. Main Form

// In the main form we can perform two steps of the algorithm. In step 1 we can create the combinations of different partitions. And in the step 2 we can integrate or merge the combinations of different partitions.

```
using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;

namespace Mining
{
    public partial class Mining : Form
    {
        public Mining()
        {
            InitializeComponent();
        }
    }
}
```

// This is the connection string that connects the Visual C# application to the Oracle database.

```
string connectionString =
"Provider=MSDAORA.1;UserID=scott;password=TIGER";
public MDataSet my_dataset;
/// <summary>
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
```

```

private void button1_Click(object sender, EventArgs e)
{
    OpenFileDialog ofd = new OpenFileDialog();
    ofd.ShowDialog();
    textBox1.Text = ofd.FileName;
}

private void button2_Click(object sender, EventArgs e)
{
    my_dataset = new MDataSet(textBox1.Text);
    //my_dataset.create_similarity_matrix(1);
    Mommo.Data.ArrayDataView ar = new
    Mommo.Data.ArrayDataView(my_dataset.discrete_data);
    dataGridView1.DataSource = ar;
    //my_dataset.sort_similarity_matrix();
    //DataToImage d2i = new
    DataToImage(my_dataset.similarity_matrix);
    //pictureBox1.Image = d2i.my_image;
}

private void button3_Click(object sender, EventArgs e)
{
    //my_dataset.runAlgorithm();
}

// Below is code to display the combinations.

private void ShowComb_Click(object sender, EventArgs e)
{
    //string connectionString = "Provider=MSDAORA.1;User
    ID=scott;password=TIGER";
    OleDbConnection myOleDbConnection = new
    OleDbConnection(connectionString);
    OleDbCommand myOleDbCommand =
    myOleDbConnection.CreateCommand();
    myOleDbConnection.Open();
    //String insertSQL = "insert into nasir (empno,ename)" + "
    values ('" + textBox1.Text.ToString() + "','" +
    textBox2.Text.ToString() + "')";
    //OleDbCommand cmd = new OleDbCommand(insertSQL,
    myOleDbConnection);
    //cmd.ExecuteNonQuery();
    //////////////////////////////////////
    OleDbDataAdapter dAdapter;
    //SqlDataAdapter dAdapter;
    DataSet ds;
    //SqlConnection con = new SqlConnection(conn);

    String insertSQL2 = "select * from e order by total";
    //where " + supplierdroplist.Text + " like'" +
    suppliertxt.Text + "'";
}

```

```

        //dAdapter = new SqlDataAdapter(insertSQL,
        myOleDbConnection);
        dAdapter = new
        OleDbDataAdapter(insertSQL2,myOleDbConnection);
        ds = new DataSet();

        dAdapter.Fill(ds);
        dataGridView2.DataSource = ds.Tables[0];
        //dataGridView2.DataBind();
        //GridView2.DataSource = ds.Tables[0].DefaultView;
        //GridView2.DataBind();
        //ds.Tables[0].Clear();
        myOleDbConnection.Close();
    }

//This part of coding is used for creating combinations

private void createComb_Click(object sender, EventArgs e)
{
    /*if (progressBar1.Value == progressBar1.Maximum)
    {
        progressBar1.Value = progressBar1.Minimum;
    }
    for (int i = progressBar1.Minimum; i <=
    progressBar1.Maximum; i++)
    {
        progressBar1.PerformStep();
    }*/

    OleDbConnection myOleDbConnection = new
    OleDbConnection(connectionString);
    OleDbCommand myOleDbCommand =
    myOleDbConnection.CreateCommand();
    myOleDbConnection.Open();

    /*OleDbDataAdapter dAdapter;
    DataSet ds;
    String insertSQL = "select * from d"; //where " +
    supplierdroplist.Text + " like'" + suppliertxt.Text + "'";

    dAdapter = new OleDbDataAdapter(insertSQL,
    myOleDbConnection);
    ds = new DataSet();

    dAdapter.Fill(ds);
    dataGridView2.DataSource = ds.Tables[0];*/
    String insertSQL = "insert into e select
    MILK, BREAD, BUTTER, HONEY, EGGS, BISCUIT, MILKCREAM, YOGURT, COFFE
    E, TEA_BAGS, count(milk) Total from " + textBoxPart1.Text + "
    group by
    cube(MILK, BREAD, BUTTER, HONEY, EGGS, BISCUIT, MILKCREAM, YOGURT,
    COFFEE, TEA_BAGS) ";

```

```

OleDbCommand cmd = new
OleDbCommand(insertSQL,myOleDbConnection);
cmd.ExecuteNonQuery();
////////////////////////////////////
/*String insertSQL = "insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from '" + textBox4.Text +
"'group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd = new
OleDbCommand(insertSQL,myOleDbConnection);
cmd.ExecuteNonQuery();*/
////////////////////////////////////
String insertSQL2 = "delete e where (MILK=0 or BREAD=0 or
BUTTER=0 or HONEY=0 or EGGS=0 or BISCUIT=0 or MILKCREAM=0
or YOGURT=0 or COFFEE=0 or TEA_BAGS=0) or (MILK is null and
BREAD is null and BUTTER is null and HONEY is null and EGGS
is null and BISCUIT is null and MILKCREAM is null and
YOGURT is null and COFFEE is null and TEA_BAGS is null)";
OleDbCommand cmd2 = new
OleDbCommand(insertSQL2,myOleDbConnection);
cmd2.ExecuteNonQuery();

String insertSQL3 = "update e set milk=0 where milk is
null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd3 = new
OleDbCommand(insertSQL3,myOleDbConnection);
cmd3.ExecuteNonQuery();
////////////////////////////////////
String insertSQL4 = "update e set BREAD=0 where BREAD is
null";
//insert into e select
,,,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFEE,TEA_BAGS,count(mil
k) Total from b group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd4 = new
OleDbCommand(insertSQL4,myOleDbConnection);
cmd4.ExecuteNonQuery();

String insertSQL5 = "update e set BUTTER=0 where BUTTER is
null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd5 = new
OleDbCommand(insertSQL5,myOleDbConnection);
cmd5.ExecuteNonQuery();

```



```

String insertSQL6 = "update e set HONEY=0 where HONEY is
null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd6 = new
OleDbCommand(insertSQL6,myOleDbConnection);
cmd6.ExecuteNonQuery();

String insertSQL7 = "update e set EGGS=0 where EGGS is
null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd7 = new
OleDbCommand(insertSQL7,myOleDbConnection);
cmd7.ExecuteNonQuery();

String insertSQL8 = "update e set BISCUIT=0 where BISCUIT
is null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd8 = new
OleDbCommand(insertSQL8,myOleDbConnection);
cmd8.ExecuteNonQuery();

String insertSQL9 = "update e set MILKCREAM=0 where
MILKCREAM is null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd9 = new
OleDbCommand(insertSQL9,myOleDbConnection);
cmd9.ExecuteNonQuery();

String insertSQL11 = "update e set YOGURT=0 where YOGURT is
null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd11 = new OleDbCommand(insertSQL11,
myOleDbConnection);
cmd11.ExecuteNonQuery();

```

```

String insertSQL12 = "update e set COFFEE=0 where COFFEE is
null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd12 = new
OleDbCommand(insertSQL12,myOleDbConnection);
cmd12.ExecuteNonQuery();

String insertSQL13 = "update e set TEA_BAGS=0 where
TEA_BAGS is null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd13 = new
OleDbCommand(insertSQL13,myOleDbConnection);
cmd13.ExecuteNonQuery();

////////////////////////////////////
String insertSQL14 = "alter table e add (pkey
varchar(10))";
OleDbCommand cmd14 = new
OleDbCommand(insertSQL14,myOleDbConnection);
cmd14.ExecuteNonQuery();

String insertSQL15 = "update e set
pkey=MILK|BREAD|BUTTER|HONEY|EGGS|BISCUIT|MILKCREAM|
YOGURT|COFFEE|TEA_BAGS";
OleDbCommand cmd15 = new
OleDbCommand(insertSQL15,myOleDbConnection);
cmd15.ExecuteNonQuery();
////////////////////////////////////
//string connectionString = "Provider=MSDAORA.1;User
ID=scott;password=TIGER";
//OleDbConnection myOleDbConnection = new
OleDbConnection(connectionString);
//OleDbCommand myOleDbCommand =
myOleDbConnection.CreateCommand();
//myOleDbConnection.Open();
//String insertSQL = "insert into nasir (empno,ename)" + "
values ('" + textBox1.Text.ToString() + "','" +
textBox2.Text.ToString() + "')";
//OleDbCommand cmd = new OleDbCommand(insertSQL,
myOleDbConnection);
//cmd.ExecuteNonQuery();
////////////////////////////////////
OleDbDataAdapter dAdapter;
//SqlDataAdapter dAdapter;
DataSet ds1;
//SqlConnection con = new SqlConnection(conn);

```

```

String insertSQL21 = "select * from e order by total";
//where " + supplierdroplist.Text + " like'" +
suppliertxt.Text + "'";
//dAdapter = new SqlDataAdapter(insertSQL,
myOleDbConnection);
dAdapter = new OleDbDataAdapter(insertSQL21,
myOleDbConnection);
ds1 = new DataSet();

dAdapter.Fill(ds1);
dataGridView2.DataSource = ds1.Tables[0];
//dataGridView2.DataBind();
//GridView2.DataSource = ds.Tables[0].DefaultView;
//GridView2.DataBind();
//ds.Tables[0].Clear();

////////////////////////////////////
//MessageBox.Show("Record Successfully Inserted");
//myOleDbConnection.Close();
////////////////////////////////////
////////////////////////////////////
myOleDbConnection.Close();
MessageBox.Show("Combinations Created Successfully");
}

```

//In this part we are dropping the table and then create the new table as well as table structure

```

private void dropCreate_Click(object sender, EventArgs e)
{
    OleDbConnection myOleDbConnection = new
    OleDbConnection(connectionString);
    OleDbCommand myOleDbCommand =
    myOleDbConnection.CreateCommand();
    myOleDbConnection.Open();

    /*OleDbDataAdapter dAdapter;
    DataSet ds;
    String insertSQL = "select * from d"; //where " +
    supplierdroplist.Text + " like'" + suppliertxt.Text + "'";

    dAdapter = new OleDbDataAdapter(insertSQL,
    myOleDbConnection);
    ds = new DataSet();

    dAdapter.Fill(ds);
    dataGridView2.DataSource = ds.Tables[0];*/
    String insertSQL = "drop table e";
    OleDbCommand cmd = new OleDbCommand(insertSQL,
    myOleDbConnection);
    cmd.ExecuteNonQuery();
}

```

```

String insertSQL2 = "create table e as select * from d
where l=2";
OleDbCommand cmd2 = new OleDbCommand(insertSQL2,
myOleDbConnection);
cmd2.ExecuteNonQuery();
myOleDbConnection.Close();
MessageBox.Show("After Dropping Table New Table is Created
Successfully");
}

//For integrating or Merging the combinations that were created in
different partition.

private void integrate_Click(object sender, EventArgs e)
{
OleDbConnection myOleDbConnection = new
OleDbConnection(connectionString);
OleDbCommand myOleDbCommand =
myOleDbConnection.CreateCommand();
myOleDbConnection.Open();

String insertSQL = "merge into e a using e2 b
on(a.pkey=b.pkey) when matched then update set
a.total=(a.total+b.total) when not matched then insert
values
(b.MILK,b.BREAD,b.BUTTER,b.HONEY,b.EGGS,b.BISCUIT,b.MILKCRE
AM,b.YOGURT,b.COFFEE,b.TEA_BAGS,b.total,b.pkey)";
OleDbCommand cmd = new
OleDbCommand(insertSQL,myOleDbConnection);
cmd.ExecuteNonQuery();

////////////////////////////////////
//string connectionString = "Provider=MSDAORA.1;User
ID=scott;password=TIGER";
//OleDbConnection myOleDbConnection = new
OleDbConnection(connectionString);
//OleDbCommand myOleDbCommand =
myOleDbConnection.CreateCommand();
// myOleDbConnection.Open();
//String insertSQL = "insert into nasir (empno,ename)" + "
values ('" + textBox1.Text.ToString() + "','" +
textBox2.Text.ToString() + "')";
//OleDbCommand cmd = new OleDbCommand(insertSQL,
myOleDbConnection);
//cmd.ExecuteNonQuery();
////////////////////////////////////
OleDbDataAdapter dAdapter;
//SqlDataAdapter dAdapter;
DataSet dsl;
//SqlConnection con = new SqlConnection(conn);

```

```

String insertSQL21 = "select * from e order by total";
//where " + supplierdroplist.Text + " like'" +
suppliertxt.Text + "'";
//dAdapter = new SqlDataAdapter(insertSQL,
myOleDbConnection);
dAdapter = new OleDbDataAdapter(insertSQL21,
myOleDbConnection);
ds1 = new DataSet();

dAdapter.Fill(ds1);
dataGridView4_form1.DataSource = ds1.Tables[0];

////////////////////////////////////
OleDbDataAdapter dAdapter22;
//SqlDataAdapter dAdapter;
DataSet ds;
//SqlConnection con = new SqlConnection(conn);

String insertSQL2 = "select * from e order by total";
//where " + supplierdroplist.Text + " like'" +
suppliertxt.Text + "'";
//dAdapter = new SqlDataAdapter(insertSQL,
myOleDbConnection);
dAdapter22 = new OleDbDataAdapter(insertSQL2,
myOleDbConnection);
ds = new DataSet();

dAdapter22.Fill(ds);
dataGridView2.DataSource = ds.Tables[0];
////////////////////////////////////
//string connectionString = "Provider=MSDAORA.1;User
ID=scott;password=TIGER";
//OleDbConnection myOleDbConnection = new
OleDbConnection(connectionString);
// OleDbCommand myOleDbCommand =
myOleDbConnection.CreateCommand();
//myOleDbConnection.Open();
//String insertSQL = "insert into nasir (empno,ename)" + "
values ('" + textBox1.Text.ToString() + "', '" +
textBox2.Text.ToString() + "')";
//OleDbCommand cmd = new OleDbCommand(insertSQL,
myOleDbConnection);
//cmd.ExecuteNonQuery();
////////////////////////////////////
Integrate test = new Integrate();
test.Show();

/*OleDbDataAdapter dAdapter00;

DataSet ds00;
String insertSQL200 = "select * from e order by total";
//where " + supplierdroplist.Text + " like'" +
suppliertxt.Text + "'";

```

```

dAdapter00 = new OleDbDataAdapter(insertSQL200,
myOleDbConnection);
ds00 = new DataSet();

dAdapter00.Fill(ds00);
dataGridView4.DataSource = ds00.Tables[0];*/

////////////////////////////////////
//MessageBox.Show("Record Successfully Inserted");
//myOleDbConnection.Close();
////////////////////////////////////
myOleDbConnection.Close();
MessageBox.Show("Integration Created Successfully");
}

//combinations are created for second partition

private void createcomb2_Click(object sender, EventArgs e)
{
    OleDbConnection myOleDbConnection = new
    OleDbConnection(connectionString);
    OleDbCommand myOleDbCommand =
    myOleDbConnection.CreateCommand();
    myOleDbConnection.Open();

    /*OleDbDataAdapter dAdapter;
    DataSet ds;
        String insertSQL = "select * from d"; //where "
+ supplierdroplist.Text + " like'" + suppliertxt.Text +
""";

    dAdapter = new OleDbDataAdapter(insertSQL,
myOleDbConnection);
    ds = new DataSet();

    dAdapter.Fill(ds);
    dataGridView2.DataSource = ds.Tables[0];*/
    /*String insertSQL = "insert into e2 select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b2 group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
    OleDbCommand cmd = new OleDbCommand(insertSQL,
myOleDbConnection);
    cmd.ExecuteNonQuery();*/
    //////////////////////////////////////
    //String insertSQL = "insert into e2 select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from " + textBoxPart2.Text + "
group by
cube (MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";

```

```

String insertSQL = "insert into e2 select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from " + comboBox1.Text + "
group by
cube(MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd = new OleDbCommand(insertSQL,
myOleDbConnection);
cmd.ExecuteNonQuery();
////////////////////////////////////

String insertSQL2 = "delete e2 where (MILK=0 or BREAD=0 or
BUTTER=0 or HONEY=0 or EGGS=0 or BISCUIT=0 or MILKCREAM=0
or YOGURT=0 or COFFEE=0 or TEA_BAGS=0) or (MILK is null and
BREAD is null and BUTTER is null and HONEY is null and EGGS
is null and BISCUIT is null and MILKCREAM is null and
YOGURT is null and COFFEE is null and TEA_BAGS is null)";
OleDbCommand cmd2 = new OleDbCommand(insertSQL2,
myOleDbConnection);
cmd2.ExecuteNonQuery();

String insertSQL3 = "update e2 set milk=0 where milk is
null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube(MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd3 = new OleDbCommand(insertSQL3,
myOleDbConnection);
cmd3.ExecuteNonQuery();
////////////////////////////////////
String insertSQL4 = "update e2 set BREAD=0 where BREAD is
null";
//insert into e select
,,,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFEE,TEA_BAGS,count(mil
k) Total from b group by
cube(MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd4 = new OleDbCommand(insertSQL4,
myOleDbConnection);
cmd4.ExecuteNonQuery();

String insertSQL5 = "update e2 set BUTTER=0 where BUTTER is
null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube(MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd5 = new OleDbCommand(insertSQL5,
myOleDbConnection);
cmd5.ExecuteNonQuery();

String insertSQL6 = "update e2 set HONEY=0 where HONEY is
null";

```

```

//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube(MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd6 = new OleDbCommand(insertSQL6,
myOleDbConnection);
cmd6.ExecuteNonQuery();

String insertSQL7 = "update e2 set EGGS=0 where EGGS is
null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube(MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd7 = new OleDbCommand(insertSQL7,
myOleDbConnection);
cmd7.ExecuteNonQuery();

String insertSQL8 = "update e2 set BISCUIT=0 where BISCUIT
is null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube(MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd8 = new OleDbCommand(insertSQL8,
myOleDbConnection);
cmd8.ExecuteNonQuery();

String insertSQL9 = "update e2 set MILKCREAM=0 where
MILKCREAM is null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube(MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd9 = new OleDbCommand(insertSQL9,
myOleDbConnection);
cmd9.ExecuteNonQuery();

String insertSQL11 = "update e2 set YOGURT=0 where YOGURT
is null";
//insert into e select
MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,COFFE
E,TEA_BAGS,count(milk) Total from b group by
cube(MILK,BREAD,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGURT,
COFFEE,TEA_BAGS)";
OleDbCommand cmd11 = new OleDbCommand(insertSQL11,
myOleDbConnection);
cmd11.ExecuteNonQuery();

String insertSQL12 = "update e2 set COFFEE=0 where COFFEE
is null";

```



```

//insert into e select
MILK, BREAD, BUTTER, HONEY, EGGS, BISCUIT, MILKCREAM, YOGURT, COFFE
E, TEA_BAGS, count(milk) Total from b group by
cube (MILK, BREAD, BUTTER, HONEY, EGGS, BISCUIT, MILKCREAM, YOGURT,
COFFEE, TEA_BAGS)";
OleDbCommand cmd12 = new OleDbCommand(insertSQL12,
myOleDbConnection);
cmd12.ExecuteNonQuery();

String insertSQL13 = "update e2 set TEA_BAGS=0 where
TEA_BAGS is null";
//insert into e select
MILK, BREAD, BUTTER, HONEY, EGGS, BISCUIT, MILKCREAM, YOGURT, COFFE
E, TEA_BAGS, count(milk) Total from b group by
cube (MILK, BREAD, BUTTER, HONEY, EGGS, BISCUIT, MILKCREAM, YOGURT,
COFFEE, TEA_BAGS)";
OleDbCommand cmd13 = new OleDbCommand(insertSQL13,
myOleDbConnection);
cmd13.ExecuteNonQuery();

////////////////////////////////////
String insertSQL14 = "alter table e2 add (pkey
varchar(10))";
OleDbCommand cmd14 = new OleDbCommand(insertSQL14,
myOleDbConnection);
cmd14.ExecuteNonQuery();

String insertSQL15 = "update e2 set
pkey=MILK|BREAD|BUTTER|HONEY|EGGS|BISCUIT|MILKCREAM|
YOGURT|COFFEE|TEA_BAGS";
OleDbCommand cmd15 = new OleDbCommand(insertSQL15,
myOleDbConnection);
cmd15.ExecuteNonQuery();
////////////////////////////////////

//string connectionString = "Provider=MSDAORA.1;User
ID=scott;password=TIGER";
//OleDbConnection myOleDbConnection = new
OleDbConnection(connectionString);
//OleDbCommand myOleDbCommand =
myOleDbConnection.CreateCommand();
//myOleDbConnection.Open();
//String insertSQL = "insert into nasir (empno,ename)" + "
values ('" + textBox1.Text.ToString() + "','" +
textBox2.Text.ToString() + "')";
//OleDbCommand cmd = new OleDbCommand(insertSQL,
myOleDbConnection);
//cmd.ExecuteNonQuery();
////////////////////////////////////
OleDbDataAdapter dAdapter;
//SqlDataAdapter dAdapter;
DataSet ds1;
//SqlConnection con = new SqlConnection(conn);

```

```

String insertSQL21 = "select * from e2 order by total";
//where " + supplierdroplist.Text + " like'" +
suppliertxt.Text + "'";
//dAdapter = new SqlDataAdapter(insertSQL,
myOleDbConnection);
dAdapter = new OleDbDataAdapter(insertSQL21,
myOleDbConnection);
ds1 = new DataSet();

dAdapter.Fill(ds1);
dataGridView3.DataSource = ds1.Tables[0];
//dataGridView2.DataBind();
//GridView2.DataSource = ds.Tables[0].DefaultView;
//GridView2.DataBind();
//ds.Tables[0].Clear();

myOleDbConnection.Close();
MessageBox.Show("Combinations Created Successfully");
}

//For dropping and creating table.

private void dropCreate2_Click(object sender, EventArgs e)
{
    OleDbConnection myOleDbConnection = new
    OleDbConnection(connectionString);
    OleDbCommand myOleDbCommand =
    myOleDbConnection.CreateCommand();
    myOleDbConnection.Open();

    /*OleDbDataAdapter dAdapter;
    DataSet ds;
    String insertSQL = "select * from d"; //where " +
    supplierdroplist.Text + " like'" + suppliertxt.Text + "'";
    dAdapter = new OleDbDataAdapter(insertSQL,
    myOleDbConnection);
    ds = new DataSet();

    dAdapter.Fill(ds);
    dataGridView2.DataSource = ds.Tables[0];*/
    String insertSQL = "drop table e2";
    OleDbCommand cmd = new OleDbCommand(insertSQL,
    myOleDbConnection);
    cmd.ExecuteNonQuery();

    String insertSQL2 = "create table e2 as select * from d
    where 1=2";//change table_d to table_d2
    OleDbCommand cmd2 = new OleDbCommand(insertSQL2,
    myOleDbConnection);
    cmd2.ExecuteNonQuery();

    myOleDbConnection.Close();
}

```

```

        MessageBox.Show("After Dropping Table New Table is Created
        Successfully");
    }

```

//If you want to display the combinations of the different partition

```

private void showComb2_Click(object sender, EventArgs e)
{
    //string connectionString = "Provider=MSDAORA.1;User
    ID=scott;password=TIGER";
    OleDbConnection myOleDbConnection = new
    OleDbConnection(connectionString);
    OleDbCommand myOleDbCommand =
    myOleDbConnection.CreateCommand();
    myOleDbConnection.Open();
    //String insertSQL = "insert into nasir (empno,ename)" + "
    values ('" + textBox1.Text.ToString() + "','" +
    textBox2.Text.ToString() + "')";
    //OleDbCommand cmd = new OleDbCommand(insertSQL,
    myOleDbConnection);
    //cmd.ExecuteNonQuery();
    //////////////////////////////////////
    OleDbDataAdapter dAdapter;
    //SqlDataAdapter dAdapter;
    DataSet ds;
    //SqlConnection con = new SqlConnection(conn);

    String insertSQL2 = "select * from e2 order by total";
    //where " + supplierdroplist.Text + " like'" +
    suppliertxt.Text + "'";
    //dAdapter = new SqlDataAdapter(insertSQL,
    myOleDbConnection);
    dAdapter = new OleDbDataAdapter(insertSQL2,
    myOleDbConnection);
    ds = new DataSet();

    dAdapter.Fill(ds);
    dataGridView3.DataSource = ds.Tables[0];
    //dataGridView2.DataBind();
    //GridView2.DataSource = ds.Tables[0].DefaultView;
    //GridView2.DataBind();
    //ds.Tables[0].Clear();

    //////////////////////////////////////
    //MessageBox.Show("Record Successfully Inserted");
    myOleDbConnection.Close();
}

private void Showcomb3_Click(object sender, EventArgs e)
{

```

```

//string connectionString = "Provider=MSDAORA.1;User
ID=scott;password=TIGER";
OleDbConnection myOleDbConnection = new
OleDbConnection(connectionString);
OleDbCommand myOleDbCommand =
myOleDbConnection.CreateCommand();
myOleDbConnection.Open();
//String insertSQL = "insert into nasir (empno,ename)" + "
values ('" + textBox1.Text.ToString() + "','" +
textBox2.Text.ToString() + "')";
//OleDbCommand cmd = new OleDbCommand(insertSQL,
myOleDbConnection);
//cmd.ExecuteNonQuery();
////////////////////////////////////
OleDbDataAdapter dAdapter;
//SqlDataAdapter dAdapter;
DataSet ds;
//SqlConnection con = new SqlConnection(conn);
String insertSQL2 = "select * from e order by total";
//where " + supplierdroplist.Text + " like" +
suppliertxt.Text + """;
//dAdapter = new SqlDataAdapter(insertSQL,
myOleDbConnection);
dAdapter = new OleDbDataAdapter(insertSQL2,
myOleDbConnection);
ds = new DataSet();

dAdapter.Fill(ds);
dataGridView4_for1.DataSource = ds.Tables[0];
//dataGridView2.DataBind();
//GridView2.DataSource = ds.Tables[0].DefaultView;
//GridView2.DataBind();
//ds.Tables[0].Clear();

////////////////////////////////////
//MessageBox.Show("Record Successfully Inserted");
myOleDbConnection.Close();
}

```

//For displaying Combinations

```

private void showComb4_Click(object sender, EventArgs e)
{
//string connectionString = "Provider=MSDAORA.1;User
ID=scott;password=TIGER";
OleDbConnection myOleDbConnection = new
OleDbConnection(connectionString);
OleDbCommand myOleDbCommand =
myOleDbConnection.CreateCommand();
myOleDbConnection.Open();
//String insertSQL = "insert into nasir (empno,ename)" + "
values ('" + textBox1.Text.ToString() + "','" +
textBox2.Text.ToString() + "')";

```

```

//OleDbCommand cmd = new OleDbCommand(insertSQL,
myOleDbConnection);
//cmd.ExecuteNonQuery();
////////////////////////////////////
OleDbDataAdapter dAdapter;
//SqlDataAdapter dAdapter;
DataSet ds;
//SqlConnection con = new SqlConnection(conn);

String insertSQL2 = "select * from e3 order by total";
//where " + supplierdroplist.Text + " like'" +
supplier.txt.Text + "'";
//dAdapter = new SqlDataAdapter(insertSQL,
myOleDbConnection);
dAdapter = new OleDbDataAdapter(insertSQL2,
myOleDbConnection);
ds = new DataSet();

dAdapter.Fill(ds);
dataGridView5.DataSource = ds.Tables[0];
myOleDbConnection.Close();
}

```

//After clicking on the "ind frequent itemset" it generates the frequent itemsets in the new form.

```

private void findFrequent_Click(object sender, EventArgs e)
{
OleDbConnection myOleDbConnection = new
OleDbConnection(connectionString);
OleDbCommand myOleDbCommand =
myOleDbConnection.CreateCommand();
myOleDbConnection.Open();
////////////////////////////////////
String insertSQL = "drop table e3";
OleDbCommand cmd = new OleDbCommand(insertSQL,
myOleDbConnection);
cmd.ExecuteNonQuery();
String insertSQL2 = "create table e3 as select * from e";
OleDbCommand cmd2 = new OleDbCommand(insertSQL2,
myOleDbConnection);
cmd2.ExecuteNonQuery();
////////////////////////////////////
String insertSQL3 = "delete e3 where total < '" +
threshold.Text + "'";
OleDbCommand cmd3 = new OleDbCommand(insertSQL3,
myOleDbConnection);
cmd3.ExecuteNonQuery();
////////////////////////////////////
//OleDbConnection myOleDbConnection = new
OleDbConnection(connectionString);

```

```

//OleDbCommand myOleDbCommand =
myOleDbConnection.CreateCommand();
// myOleDbConnection.Open();
//String insertSQL = "insert into nasir (empno,ename)" + "
values ('" + textBox1.Text.ToString() + "','" +
textBox2.Text.ToString() + "')";
//OleDbCommand cmd = new OleDbCommand(insertSQL,
myOleDbConnection);
//cmd.ExecuteNonQuery();
////////////////////////////////////
OleDbDataAdapter dAdapter22;
//SqlDataAdapter dAdapter;
DataSet ds;
//SqlConnection con = new SqlConnection(conn);

String insertSQL222 = "select * from e3 order by total";
//where " + supplierdroplist.Text + " like'" +
suppliertxt.Text + "'";
//dAdapter = new SqlDataAdapter(insertSQL,
myOleDbConnection);
dAdapter22 = new OleDbDataAdapter(insertSQL222,
myOleDbConnection);
ds = new DataSet();

dAdapter22.Fill(ds);
dataGridView5.DataSource = ds.Tables[0];
//dataGridView2.DataBind();
//GridView2.DataSource = ds.Tables[0].DefaultView;
//GridView2.DataBind();
//ds.Tables[0].Clear();

////////////////////////////////////
//MessageBox.Show("Record Successfully Inserted");
// myOleDbConnection.Close();
////////////////////////////////////
myOleDbConnection.Close();
MessageBox.Show("Frequent Item Sets Created Successfully");
}

private void Save_Click(object sender, EventArgs e)
{
OleDbConnection myOleDbConnection = new
OleDbConnection(connectionString);
OleDbCommand myOleDbCommand =
myOleDbConnection.CreateCommand();
myOleDbConnection.Open();
for (int count = 0; count < dataGridView1.Rows.Count - 1;
count++)
{
int rowCount = count + 1;
//Get first cell value
object milk = dataGridView1.Rows[count].Cells[0].Value;
if (milk == null)
{

```

```

        MessageBox.Show("Please enter prodcut name in Row
        number: " + rowCount);
        return;
    }
    object bread =
    dataGridView1.Rows[count].Cells[1].Value;
    if (bread == null)
    {
        MessageBox.Show("Please enter prodcut price in row
        number: " + rowCount);
        return;
    }
    //////////////////////////////////////

    object BUTTER =
    dataGridView1.Rows[count].Cells[2].Value;

    if (BUTTER == null)
    {
        MessageBox.Show("Please enter prodcut description in
        Row number: " + rowCount);
        return;
    }

    object HONEY =
    dataGridView1.Rows[count].Cells[2].Value;
    if (HONEY == null)
    {
        MessageBox.Show("Please enter prodcut description in
        Row number: " + rowCount);
        return;
    }

    object EGGS = dataGridView1.Rows[count].Cells[2].Value;
    if (EGGS == null)
    {
        MessageBox.Show("Please enter prodcut description in
        Row number: " + rowCount);
        return;
    }

    object BISCUIT =
    dataGridView1.Rows[count].Cells[2].Value;
    if (BISCUIT == null)
    {
        MessageBox.Show("Please enter prodcut description in
        Row number: " + rowCount);
        return;
    }

    object MILKCREAM =
    dataGridView1.Rows[count].Cells[2].Value;
    if (MILKCREAM == null)
    {

```

```

        MessageBox.Show("Please enter prodcut description in
        Row number: " + rowCount);
        return;
    }

    object YOGURT =
    dataGridView1.Rows[count].Cells[2].Value;
    if (YOGURT == null)
    {
        MessageBox.Show("Please enter prodcut description in
        Row number: " + rowCount);
        return;
    }

    object COFFEE =
    dataGridView1.Rows[count].Cells[2].Value;
    if (COFFEE == null)
    {
        MessageBox.Show("Please enter prodcut description in
        Row number: " + rowCount);
        return;
    }

    object TEA_BAGS =
    dataGridView1.Rows[count].Cells[2].Value;
    if (TEA_BAGS == null)
    {
        MessageBox.Show("Please enter prodcut description in
        Row number: " + rowCount);
        return;
    }
    ////////////////////////////////////////////////////
    string insertQry = "INSERT INTO data_fill
    (milk,bread,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGUR
    T,COFFEE,TEA_BAGS) VALUES ('" +
        milk + "','" + bread + "','" + BUTTER + "','" +
    HONEY + "','" + EGGS + "','" + BISCUIT + "','" +
    MILKCREAM + "','" + YOGURT + "','" + COFFEE + "','" +
    TEA_BAGS + "')";

    //OleDbConnection dbConnection = new
    OleDbConnection(conString);

    try
    {
        //myOleDbConnection.Open();
        OleDbCommand command = new OleDbCommand(insertQry,
        myOleDbConnection);
        command.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
    MessageBox.Show("inserted successfully");

```



```

        myOleDbConnection.Close();
    }

    private void browse2_Click(object sender, EventArgs e)
    {
        OpenFileDialog ofd = new OpenFileDialog();
        ofd.ShowDialog();
        textBox2.Text = ofd.FileName;
    }

    private void load2_Click(object sender, EventArgs e)
    {
        my_dataset = new MDataSet(textBox2.Text);
        //my_dataset.create_similarity_matrix(1);
        Mommo.Data.ArrayDataView ar = new
        Mommo.Data.ArrayDataView(my_dataset.discrete_data);
        dataGridView6.DataSource = ar;
        //my_dataset.sort_similarity_matrix();
        //pictureBox1.Image = d2i.my_image;
    }

    private void savebtn_Click(object sender, EventArgs e)
    {
        OleDbConnection myOleDbConnection = new
        OleDbConnection(connectionString);
        OleDbCommand myOleDbCommand =
        myOleDbConnection.CreateCommand();
        myOleDbConnection.Open();
        for (int count = 0; count < dataGridView1.Rows.Count - 1;
        count++)
        {
            int rowCount = count + 1;
            //Get first cell value
            object milk = dataGridView1.Rows[count].Cells[0].Value;
            if (milk == null)
            {
                MessageBox.Show("Please enter prodcut name in Row
                number: " + rowCount);
                return;
            }
            object bread =
            dataGridView1.Rows[count].Cells[1].Value;
            if (bread == null)
            {
                MessageBox.Show("Please enter prodcut price in row
                number: " + rowCount);
                return;
            }
            //////////////////////////////////////

            object BUTTER =
            dataGridView1.Rows[count].Cells[2].Value;
            if (BUTTER == null)
            {

```

```
        MessageBox.Show("Please enter prodcut description in  
        Row number: " + rowCount);  
        return;  
    }  
  
    object HONEY =  
    dataGridView1.Rows[count].Cells[2].Value;  
    if (HONEY == null)  
    {  
        MessageBox.Show("Please enter prodcut description in  
        Row number: " + rowCount);  
        return;  
    }  
  
    object EGGS = dataGridView1.Rows[count].Cells[2].Value;  
    if (EGGS == null)  
    {  
        MessageBox.Show("Please enter prodcut description in  
        Row number: " + rowCount);  
        return;  
    }  
  
    object BISCUIT =  
    dataGridView1.Rows[count].Cells[2].Value;  
    if (BISCUIT == null)  
    {  
        MessageBox.Show("Please enter prodcut description in  
        Row number: " + rowCount);  
        return;  
    }  
  
    object MILKCREAM =  
    dataGridView1.Rows[count].Cells[2].Value;  
    if (MILKCREAM == null)  
    {  
        MessageBox.Show("Please enter prodcut description in  
        Row number: " + rowCount);  
        return;  
    }  
  
    object YOGURT =  
    dataGridView1.Rows[count].Cells[2].Value;  
    if (YOGURT == null)  
    {  
        MessageBox.Show("Please enter prodcut description in  
        Row number: " + rowCount);  
        return;  
    }  
  
    object COFFEE =  
    dataGridView1.Rows[count].Cells[2].Value;  
    if (COFFEE == null)  
    {  
        MessageBox.Show("Please enter prodcut description in  
        Row number: " + rowCount);  
        return;  
    }  
}
```

```

    }

    object TEA_BAGS =
    dataGridView1.Rows[count].Cells[2].Value;
    if (TEA_BAGS == null)
    {
        MessageBox.Show("Please enter prodcut description in
        Row number: " + rowCount);
        return;
    }
    //////////////////////////////////////
    string insertQry = "INSERT INTO data_fill2
    (milk,bread,BUTTER,HONEY,EGGS,BISCUIT,MILKCREAM,YOGUR
    T,COFFEE,TEA_BAGS) VALUES('" +
    milk + "','" + bread + "','" + BUTTER + "','" + HONEY
    + "','" + EGGS + "','" + BISCUIT + "','" + MILKCREAM
    + "','" + YOGURT + "','" + COFFEE + "','" + TEA_BAGS
    + "')";

    try
    {
        //myOleDbConnection.Open();
        OleDbCommand command = new OleDbCommand(insertQry,
        myOleDbConnection);
        command.ExecuteNonQuery();
        //MessageBox.Show("Row: "+ rowCount+ " data saved in
        db");
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
    }

    MessageBox.Show("inserted successfully");
    myOleDbConnection.Close();
}

private void exitToolStripMenuItem_Click(object sender,
EventArgs e)
{
    Application.Exit();
}

private void loadDataToolStripMenuItem_Click(object sender,
EventArgs e)
{
    Load test = new Load();
    test.Show();
}

private void createCombinationsToolStripMenuItem1_Click(object
sender, EventArgs e)
{
    CreateCombination test = new CreateCombination();
    test.Show();
}

```

```

private void integrateMergeToolStripMenuItem_Click(object
sender, EventArgs e)
{
    Integrate test = new Integrate();
    test.Show();
}

private void findFrequentItemsToolStripMenuItem2_Click(object
sender, EventArgs e)
{
    FindFrequentItems test = new FindFrequentItems();
    test.Show();
}

private void button4_Click(object sender, EventArgs e)
{
    FindFrequentItems test = new FindFrequentItems();
    test.Show();
}
}
}

```

5.2. Integration / Merging

//Integration of combinations that are created in different partitions.

```

using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;

namespace Mining
{
    public partial class Integrate : Form
    {
        string connectionString = "Provider=MSDAORA.1;User
ID=scott;password=TIGER";
        public MDataSet my_dataset;
        public Integrate()
        {
            InitializeComponent();

```

```
//This is the function that can merge different different
partition(combinations).
```

```
private void button1_Click(object sender, EventArgs e)
{
    OleDbConnection myOleDbConnection = new
    OleDbConnection(connectionString);
    OleDbCommand myOleDbCommand =
    myOleDbConnection.CreateCommand();
    myOleDbConnection.Open();

    String insertSQL = "merge into e a using e2 b
on(a.pkey=b.pkey) when matched then update set
a.total=(a.total+b.total) when not matched then insert
values
(b.MILK,b.BREAD,b.BUTTER,b.HONEY,b.EGGS,b.BISCUIT,b.MILKCRE
AM,b.YOGURT,b.COFFEE,b.TEA_BAGS,b.total,b.pkey)";
    OleDbCommand cmd = new OleDbCommand(insertSQL,
myOleDbConnection);
    cmd.ExecuteNonQuery();
    myOleDbConnection.Close();
    MessageBox.Show("Integration Created Successfully");
}
}
```

```
//Display the resultant integration item sets
```

```
private void Showcomb3_Click(object sender, EventArgs e)
{
    //string connectionString = "Provider=MSDAORA.1;User
ID=scott;password=TIGER";
    OleDbConnection myOleDbConnection = new
    OleDbConnection(connectionString);
    OleDbCommand myOleDbCommand =
    myOleDbConnection.CreateCommand();
    myOleDbConnection.Open();
    //String insertSQL = "insert into nasir (empno,ename)" + "
values ('" + textBox1.Text.ToString() + "','" +
textBox2.Text.ToString() + "')";
    //OleDbCommand cmd = new OleDbCommand(insertSQL,
myOleDbConnection);
    //cmd.ExecuteNonQuery();
    //////////////////////////////////////
    OleDbDataAdapter dAdapter;
    //SqlDataAdapter dAdapter;
    DataSet ds;
    //SqlConnection con = new SqlConnection(conn);

    String insertSQL2 = "select * from e order by total";
    //where " + supplierdroplist.Text + " like'" +
suppliertxt.Text + "'";
    //dAdapter = new SqlDataAdapter(insertSQL,
myOleDbConnection);
}
```

```

        dAdapter = new OleDbDataAdapter(insertSQL2,
        myOleDbConnection);
        ds = new DataSet();

        dAdapter.Fill(ds);
        dataGridView4.DataSource = ds.Tables[0];
        //dataGridView2.DataBind();
        //GridView2.DataSource = ds.Tables[0].DefaultView;
        //GridView2.DataBind();
        //ds.Tables[0].Clear();

        //////////////////////////////////////
        //MessageBox.Show("Record Successfully Inserted");
        myOleDbConnection.Close();
    }

    private void exitToolStripMenuItem_Click(object sender,
    EventArgs e)
    {
        this.Close();
    }
}
}

```

5.3. Finding Frequent ItemSets

//At the end, user can specify minimum threshold value. And on the basis of that given threshold frequent itemsets are displayed.

```

using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;

namespace Mining
{
    public partial class FindFrequentItems : Form
    {
        public FindFrequentItems()
        {
            InitializeComponent();
        }

        string connectionString =
            "Provider=MSDAORA.1;UserID=scott;password=TIGER";
        public MDataSet my_dataset;
    }
}

```

//We can find Frequent Item Sets using the below code.

```
private void findFrequent_Click(object sender, EventArgs e)
{
    OleDbConnection myOleDbConnection = new
    OleDbConnection(connectionString);
    OleDbCommand myOleDbCommand =
    myOleDbConnection.CreateCommand();
    myOleDbConnection.Open();
    ///////////////////////////////////
    String insertSQL = "drop table e3";
    OleDbCommand cmd = new OleDbCommand(insertSQL,
    myOleDbConnection);
    cmd.ExecuteNonQuery();
    String insertSQL2 = "create table e3 as select * from e";
    OleDbCommand cmd2 = new OleDbCommand(insertSQL2,
    myOleDbConnection);
    cmd2.ExecuteNonQuery();
    ///////////////////////////////////
    String insertSQL3 = "delete e3 where total < '" +
    threshold.Text + "'";
    OleDbCommand cmd3 = new OleDbCommand(insertSQL3,
    myOleDbConnection);
    cmd3.ExecuteNonQuery();
    ///////////////////////////////////
    //string connectionString = "Provider=MSDAORA.1;User
    ID=scott;password=TIGER";
    //OleDbConnection myOleDbConnection = new
    OleDbConnection(connectionString);
    //OleDbCommand myOleDbCommand =
    myOleDbConnection.CreateCommand();
    //myOleDbConnection.Open();
    //String insertSQL = "insert into nasir (empno,ename)" + "
    values ('" + textBox1.Text.ToString() + "','" +
    textBox2.Text.ToString() + "')";
    //OleDbCommand cmd = new OleDbCommand(insertSQL,
    myOleDbConnection);
    //cmd.ExecuteNonQuery();
    ///////////////////////////////////
    OleDbDataAdapter dAdapter0;
    //SqlDataAdapter dAdapter;
    DataSet ds0;
    //SqlConnection con = new SqlConnection(conn);

    String insertSQL20 = "select * from e3 order by total";
    //where " + supplierdroplist.Text + " like'" +
    suppliertxt.Text + "'";
    //dAdapter = new SqlDataAdapter(insertSQL,
    myOleDbConnection);
    dAdapter0 = new OleDbDataAdapter(insertSQL20,
    myOleDbConnection);
    ds0 = new DataSet();

    dAdapter0.Fill(ds0);
}
```

```
private void exitToolStripMenuItem_Click(object sender,
EventArgs e)
{
    this.Close();
}
}
```



```

dataGridView5.DataSource = ds0.Tables[0];
//dataGridView2.DataBind();
//GridView2.DataSource = ds.Tables[0].DefaultView;
//GridView2.DataBind();
//ds.Tables[0].Clear();

////////////////////////////////////
//MessageBox.Show("Record Successfully Inserted");
//myOleDbConnection.Close();
////////////////////////////////////
myOleDbConnection.Close();
MessageBox.Show("Frequent Item Sets Created Successfully");
}

//For displaying the frequent itemsets

private void showComb4_Click(object sender, EventArgs e)
{
    //string connectionString = "Provider=MSDAORA.1;User
    ID=scott;password=TIGER";
    OleDbConnection myOleDbConnection = new
    OleDbConnection(connectionString);
    OleDbCommand myOleDbCommand =
    myOleDbConnection.CreateCommand();
    myOleDbConnection.Open();
    //String insertSQL = "insert into nasir (empno,ename)" + "
    values ('" + textBox1.Text.ToString() + "','" +
    textBox2.Text.ToString() + "')";
    //OleDbCommand cmd = new OleDbCommand(insertSQL,
    myOleDbConnection);
    //cmd.ExecuteNonQuery();
    //////////////////////////////////////
    OleDbDataAdapter dAdapter;
    //SqlDataAdapter dAdapter;
    DataSet ds;
    //SqlConnection con = new SqlConnection(conn);

    String insertSQL2 = "select * from e3 order by total";
    //where " + supplierdroplist.Text + " like'" +

```

supplier.txt.Text + "'";

OleDbDataAdapter(insertSQL,

CHAPTER 6
RESULTS AND CONCLUSION

6. Results and Conclusion

Efficiency of this algorithm is dependent on the number of partition in which data comes into the data warehouse. As well as number of partitions increases, the efficiency of this algorithm will increase as compare to the other algorithms. E.g. we have a dataset that contains 10 Lac transactions. And if it takes one second to resolve two transactions then totally it takes 5 Lac seconds in the case of other algorithms. But in our algorithm if there are 10 partitions then it takes only 50 thousand seconds with some integration/merging time. And if same data comes in 20 partitions then efficiency of this algorithm also increases. It takes only 25 thousand seconds with little bit integration/merging time. In the case of 10 partitions we save 4 Lac and 50 thousand seconds. And in the case of 20 partitions we save 4 Lac and 75 thousand seconds. And also saves a little bit time of privileges and syntax checking that is stored in shared pool buffer. So at the end as the number of partitions increases the efficiency will increase. So the efficiency of this algorithm is directly proportion to the number of partition.

Now we take a dataset that have 10 million transactions. If data comes in different number of partitions then we check the efficiency of this algorithm with graph.

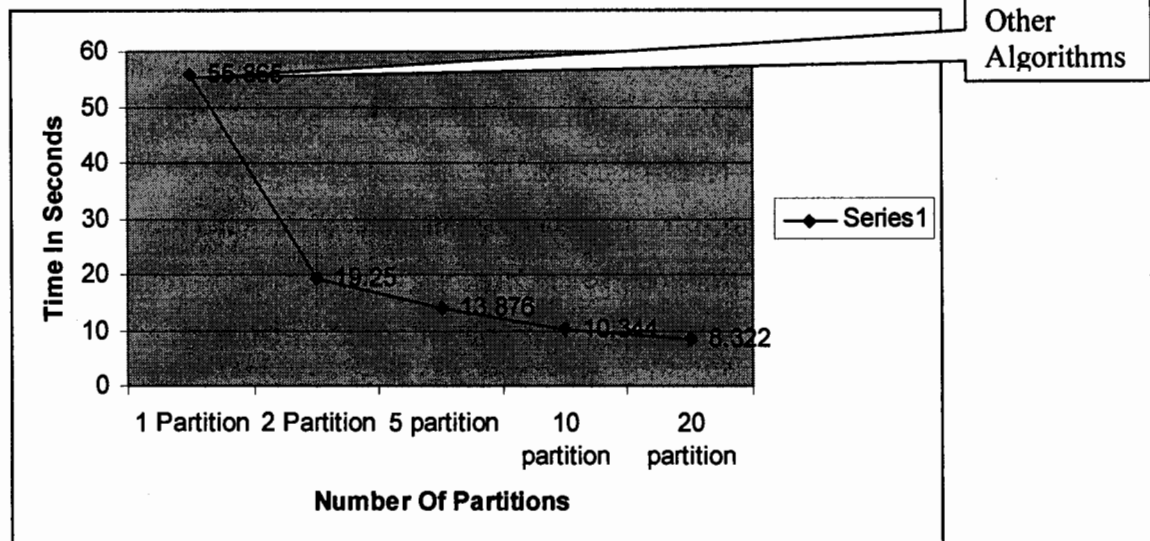


Fig 6.1 Comparison Graph

An efficient algorithm for association rule mining is one that can provide solution to some of the mentioned factors discussed in literature survey. As the algorithm should use small candidate item sets and generates frequent item sets efficiently, execution time should be minimized. Here, a new technique for finding frequent item sets is present. This is incremental based technique. In this technique generation of item sets are done on interval bases, and then integrate/merge the resultant item sets of different intervals. And at the end globally check the frequent item sets on the basis of support count. An efficient algorithm for association rule mining is one that can provide efficient solution and generates frequent item sets efficiently, execution time should be minimized. The incremental (interval) based technique is a new technique for finding frequent item sets. This technique can generate frequent item sets efficiently and also reuse the combinations of item sets that were created before. Ultimately, we got an efficient algorithm to find frequent item sets.

APPENDIX A

Screen Shots

This algorithm is implemented using Visual C#.Net as front end and Oracle as Backend.

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

- Splash Screen

When the software is executed this screen appears.

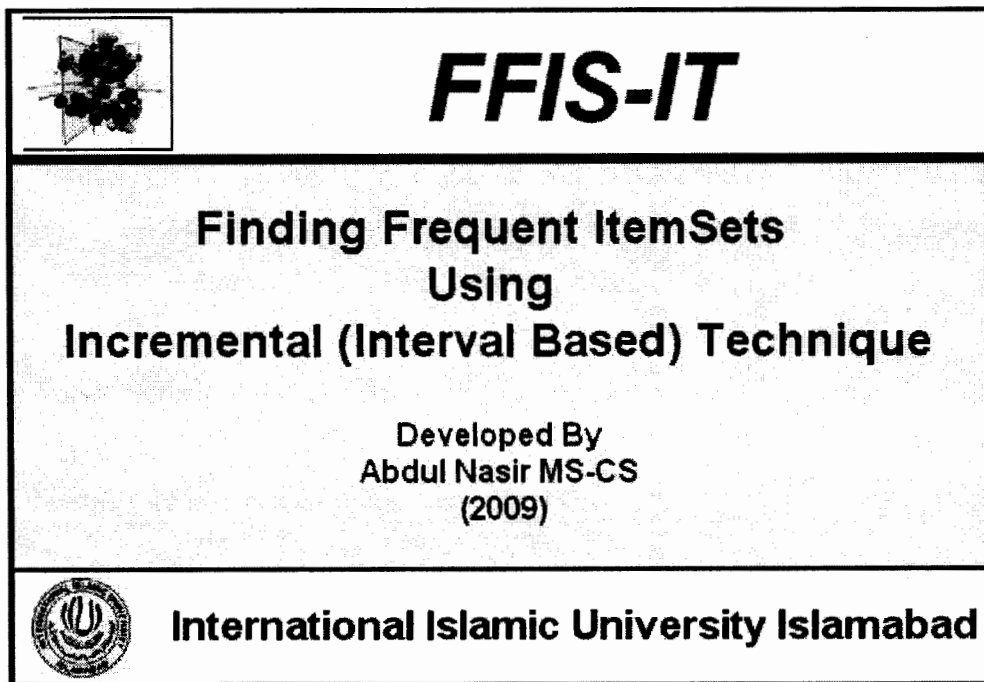


Fig A-1 Splash Screen

Appendix A

Loading data into the database. Data may load from flat file, dump table or may be from transactional database.

Step 1: **Load Data Into the Database**

	1	0	1	0	1	1	1	1	0
0	0	0	1	0	1	1	0	1	0
1	0	1	1	0	1	1	0	0	0
0	0	1	1	1	0	1	0	1	0
1	0	0	0	0	0	1	1	1	1
1	0	1	1	1	0	1	0	0	0
1	0	1	1	0	0	1	1	1	1
0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	0	1	1	1
1	1	0	0	0	1	1	0	0	1
0	0	0	0	1	1	1	1	1	0
1	1	1	1	1	0	1	0	1	0
1	0	0	0	0	1	1	0	1	1
1	0	1	1	0	0	1	0	1	0
0	0	1	1	1	0	1	1	0	1
0	1	1	1	1	1	1	0	0	0
0	0	1	0	1	1	1	1	0	0
1	1	1	0	0	0	0	1	1	0

Flat File Path: c:\data_load.txt **Load Data From Flat File** .dmp File Path: C:\data\dmp **Load Data From *.dmp File** **Clear**

Step 2: **Counting Frequency Of Possible Item Sets**

Fig A-2: Load Data into the Database

Appendix A

Possible itemsets combinations are created of Table "A".

Step 2: Counting Frequency of the Possible ItemSets

Table A Start Time 22:52:23:109 End Time 22:52:33:648

	0	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	1
0	1	1	1	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	1	1	1	0	0	1	1	1
1	0	1	1	1	0	1	1	1	1	1

Enter Table Name DATA_10_lac1

Combinations Created Successfully

Table B

Enter Table Name DATA_10_lac2

Step 3: Integrate Table A AND Table B Step 4: Find Frequent ItemSets

Finding Frequent ItemSets Using Incremental (Interval) Based Technique

Fig A-3: Combinations of "Table A"

Appendix A

Possible itemsets combinations are created of Table "A" and Table "B".

Step 2: Counting Frequency of the Possible ItemSets

Table A Start Time 22:53:48:406 End Time 22:53:58:609

	0	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	1
0	1	1	1	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	1	1	1	0	0	1	1	1
1	0	1	1	1	0	1	1	1	1	1

Enter Table Name DATA_10_lac1 Create Combinations Details of ItemSets

Combinations Created Successfully OK

Table B

	0	1	1				1	1	1	1
0	1	1	0	1	1	1	1	1	1	1
0	1	1	1	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	1	1	1	0	0	1	1	1
1	0	1	1	1	0	1	1	1	1	1

Enter Table Name DATA_part1 Drop And Create Create Combinations Details of ItemSets

Step 3: Integrate "Table A" AND "Table B" Step 4: Find Frequent ItemSets

Finding Frequent ItemSets Using Incremental (Interval) Based Technique

Fig A-4: Combinations of "Table A" and "Table B"

Appendix A

Details of possible itemsets

The screenshot shows a software window with the title "Finding Frequent ItemSets Using Incremental (Interval) Based Technique". The window contains a table titled "Counting Frequency of the Possible ItemSets". The table has 11 columns labeled "1 ItemSet" through "10 ItemSet" and 11 rows. The first row has a value of 1 in the 10th column and 4218 in the 11th column. The second row has a value of 1 in the 9th column and 4202 in the 11th column. The third row has a value of 1 in the 8th column and 4237 in the 11th column. The fourth row has a value of 1 in the 7th column and 6139 in the 11th column. The fifth row has a value of 1 in the 6th column and 4241 in the 11th column. The sixth row has a value of 1 in the 5th column and 4236 in the 11th column. The seventh row has a value of 1 in the 4th column and 4130 in the 11th column. The eighth row has a value of 1 in the 3rd column and 6182 in the 11th column. The ninth row has a value of 1 in the 2nd column and 6177 in the 11th column. The tenth row has a value of 1 in the 1st column and 4309 in the 11th column. The rest of the cells in the table are empty.

	1 ItemSet	2 ItemSet	3 ItemSet	4 ItemSet	5 ItemSet	6 ItemSet	7 ItemSet	8 ItemSet	9 ItemSet	10 ItemSet	
										1	4218
0									1	0	4202
0								1	0	0	4237
0							1	0	0	0	6139
0						1	0	0	0	0	4241
0					1	0	0	0	0	0	4236
0				1	0	0	0	0	0	0	4130
0			1	0	0	0	0	0	0	0	6182
0		1	0	0	0	0	0	0	0	0	6177
1		0	0	0	0	0	0	0	0	0	4309

Fig A-5: Details of possible itemsets

Appendix A

At the end user can find frequent item sets.

The screenshot displays a software application window titled "Finding Frequent ItemSets Using Incremental (Interval) Based Technique". The window is in "Step 4: Frequent ItemSets". It features a data table with 11 columns and 20 rows. The first column contains item IDs, and the remaining 10 columns contain binary values (0 or 1). A dialog box in the center of the table displays the message "Frequent Item Sets Created Successfully" with an "OK" button. At the bottom of the window, there is a field labeled "Enter Threshold" with the value "5000" and a "Find Frequent Items" button. The Windows taskbar is visible at the bottom of the screen.

	1	0	0	0	0	0	0	0	1	5000
0	1	0	0	0	0	0	0	0	1	5002
0	0	1	0	0	0	0	0	0	1	5012
0	0	1	0	1	0	0	0	0	0	5016
0	1	0	0	0	0	0	0	1	0	5042
1	1	0	0	0	0	0	0	0	0	5052
0	0	0	0	1	0	1	0	0	0	5056
0	0	0	0	0	1	1	0	0	0	5064
0	0	1	0				1	0	0	5076
0	0	1	0				0	0	0	5076
0	0	0	0				1	0	0	5080
0	1	0	0				0	0	0	5096
0	0	0	0				0	0	1	5104
0	1	0	0	1	0	0	0	0	0	5106
1	0	1	0	0	0	0	0	0	0	5134
1	0	0	0	0	0	1	0	0	0	5176
0	0	1	0	0	0	0	1	0	0	6742
0	1	0	0	0	0	0	1	0	0	6744
0	1	1	0	0	0	0	0	0	0	6822
0	0	0	1	0	0	0	0	0	0	8260
0	0	0	0	0	0	0	0	1	0	8404
0	0	0	0	0	0	0	0	0	1	8436
0	0	0	0	1	0	0	0	0	0	8472

Enter Threshold Find Frequent Items

Fig A-7: Frequent Itemsets

REFERENCES

References

1. J. Hen, M. Kamber. Data Mining Concepts and Techniques
2. Oracle® Database Data Warehousing Guide 10g Release 1 (10.1)
Part Number B10736-01
3. oracle 9i Data Warehousing Guide, Release 2(9.2)
4. J. Wiley .Data mining: concepts, Models, Methods, and Algorithms. 2003
5. E.Brand and R.Gerritsen. Data Mining and Knowledge Discovery DBMS, Data Mining Solutions Supplement
6. R. Agrawal,T. Imielinski and A. Swami.Mining Association Rules Between Sets of Items In large Databases. Proceedings of the ACM international conference on Management of data, May 1993
7. M.H. Dunham,Y. Xiao,L. Gruenwald,and Z. Hossain. A Survey of Association Rules
8. R. Agrawal, R. Srikant .Fast algorithm for mining association rules. VLDB conference,1994
9. S. Orlando,P. Palmerini, and R. Perego.DCI: a Hybrid Algorithm for Frequent Set Counting.
10. H. Toivon.Sampling large databases for association rules. Proc.22nd VLDB conference .bombay ,India, 1996
11. A. Savasere ,E. Omiecinski and S. Navathe. An efficient algorithm for mining association rules in large databases.Proc,of the 21st International Conference on very Databases.Zurich ,Switzerland,1995
12. R. J,and B. Jr.Efficiently mining long patterns from databases. Proc ACM SIGMOD international conference on management of data,1998
13. J. HAN, J. Pie,Y. Tin and R. Mao. Mining frequent patterns without candidate generation:A frequent pattern tree approach.Data mining and knowledge Discovery,2004
14. F. Coenen.The LUCS-KDD implementation of the FP-Growth algorithm.htm
15. M.J.Zaki,S. Parthasarathy ,M.Ogihara,and W. Li.New algorithm for fast discovery of association rules. Aug ,1997
16. B. Goethals .Frequent pattern Mining (survey).Department of Computer Science,University of Helsinki ,2003

