Kurdistan Region Government – Iraq
Ministry of Higher Education and Scientific Research
Salahaddin University – Erbil

Design and Implementation of a Proposed Technique for Association Rule Mining

A Thesis
Submitted to the College of Engineering in the University of Salahaddin – Hawler in Partial Fulfillment of the Requirements for the Degree of Master of Science in ICT Engineering

By
Polla Abdulhamid Fattah
B.Sc. of Software Engineering

Supervised by
Dr. Ibrahim I. Hamarash
Assist Prof. of Control Engineering

Erbil – 2008

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

مَنْ يَهْدِ اللَّهُ فَهُوَ الْمُهْتَدِ

وَمَنْ يُضْلِلْ فَلَنْ تَجِدَ لَهُ وَلِيًّا

مُرْشِدًا

صدق الله العظيم

الكهف -17

Supervisor Certification

I certify that this thesis “Design and Implementation of a Proposed Technique for Association Rule Mining” by Polla Abdulhamid Fattah was prepared under my supervision at Department of Electric, College of Engineering, Salahaddin University – Erbil, as a partial fulfillment of requirements for master degree in Information and Communication Technology ICT.

Signature:

Supervisor: Dr. Ibrahim I. Hamarash

Date: / / 2008

In view of the available recommendation I forward this project for debate by the examining committee.

Signature:

Supervisor: Dr. Ibrahim I. Hamarash

Head of the Department of Electric

Date: / / 2008

Examining Committee Certification

We certify that we have read this research “Design and Implementation of a Proposed Technique for Association Rule Mining” and as examiner committee examined the student “Polla Abdulhamid Fattah” in its content and what related to it, and that our opinion it meet the standards of a thesis for the degree of MSc in ICT Engineering.

Signature:
Dr. Saran Akram Chawshly
Date: / / 2008
Member

Signature:
Assist Prof. Amin Abbas
Date: / / 2008
Member

Signature:
Assist Prof. Dr. Ibrahim I. Hamarash
Date: / / 2008
Supervisor

Signature:
Assist Prof. Dr. Ahmad Tariq
Date: / / 2008
Chairman

Approved for the College Committee of Graduate Studies

Signature:
Assist Prof. Dr. Shawnim Rashid Jalal
Dean of the College of Engineering
Date: / / 2008

Dedication

To The Prophet Muhammad

To my father and his support

To my mother and her prays

To my wife for her faith in my success

To my brothers and my sister

To my little girl

To my supervisor

To Mr. Karim Zebary

Polla

Acknowledgements

First of all, Praise is to be for Allah Who has enlightened me and paved the way to accomplish this thesis.

After that, I would like to express my deepest gratitude and appreciation to my supervisor Dr. Ibrahim Ismail Hamarash, for his excellent advice, guidance and cooperation during the course of this work.

Special thanks to Dr. Saran for providing facilities, Dr. Rauf for providing some real data and resources, Dr. Samah for allowing me to use CISCO's network.

I would like to thank my family for their encouragement and support during my study.

Abstract

Modern humans found themselves living in an expanding universe of data in which there is too much data and too little information. The development of new techniques to find the required information from huge amount of data is one of the main challenges for software developers today. The process of knowledge discovering from data or databases is called Data Mining.

In this thesis, a new approach for association mining has been proposed, designed, implemented, coded, verified and tested on real data. The approach proposes splitting of the running mode of the data mining into two different parts, in order to optimize the time-memory domain.

The first part is responsible of finding all subsets (itemsets) in every transaction then stores and accumulates there frequencies (without any pruning) in a database for future use, this process leads to fetch each transaction only once which obviously reduces I/O for fetching transactions.

The second part uses the output of the first part this data consists of itemsets and their frequencies this reduces wait time for the users to produce rules and it also leads for flexibility of the output type, also using these data enables users to change their queries and criteria (minsup, minconf) without running all process from scratch just it require running second part of the system.

For each part, an algorithm has been developed and coded using JAVA/MySql. The system is verified and applied to a real data of a shopping basket in a Erbilian supermarket. Test results show significant improvement in the quality and time required for rule generation.

Contents

List of Algorithms

List of Tables

List of Figures

List of Abbreviations

DBMS	Database Management System
DM	Data mining
GUI	Graphical User Interface
KDD	Knowledge Discovery in Database
mincof	Minimum Confidence
Minsup	minimum Support
NP	None Polynomial
SQL	Structured Query Language
TID	Transaction IDentifier

1

Chapter One

Introduction

1.1 Introduction

Data mining is the process of discovering meaningful new correlations, patterns and trends by sifting through large amounts of data stored in repositories, using pattern recognition technologies as well as statistical and mathematical techniques [DANI05].

The amount of data stored in database continues to grow fast. Intuitively, this large amount of stored data contains valuable hidden knowledge which could be used to improve decision making process of an organization. For instance, data about previous sales might contain interesting relationship between products and customers. The discovery of such relationships can be very useful to increase the sales of a company. However, the number of human data analyst growth at much smaller rate than the amount of stored data. Thus, there is a clear need for automatic (or semi-automatic) methods for extracting knowledge from data. This need has led to the emergence of a field called data mining and knowledge discovery [WEIS98].

Modern humans find themselves living in an expanding universe of data in which there is too much data and too little information. The development of new techniques to find the required information from huge amount of data is one of the main challenges for software developers today.

The politicians, the managers, marketers, and any decision makers require the computer to deeply understand the data universe to extract the hidden and unintentionally constructed knowledge and information depending on new tools, especially after failure of conventional tools to mine and analyze such knowledge and information. Data mining is an approach to overcome these problems. It is the process of Knowledge Discovery from Databases (KDD).

The process of knowledge discovery (where data is transformed into knowledge for decision making) is data mining. The term is misnomer; mining gold from sands is referred to as gold mining and not sand mining. Thus, data mining should have been more

appropriately named “Knowledge Mining”, but such a misnomer that carries both “data” and “mining” become a popular choice in information systems literature [MICH04].

Over the last four decades, Database Management System’s processed data by using the database technology available to users that supports query languages. The problem with query languages they are structural languages which assume that the user is aware of database schema. For example the query language “SQL” supports operation of selection from table or join related information from tables based on common fields. Today’s database users need functionality of consolidation, aggregation, and summarization of data, which require viewing the information along multiple dimensions that is what SQL is unable to do. The automatic Knowledge discovery tools have emerged in order to overcome this difficulty, and have taken the attention of the researchers of database literature. Knowledge Discovery in Databases includes all pre-processing steps in stored data, discovering interesting patterns on the data are referred to as Data Mining; which refers to the discovery of interesting patterns from data in knowledge discovery process. These interesting patterns may be in the form of associations, deviations, regularities, etc [HUSS02].

1.2 Literature Survey

The term Knowledge Discovery in Database “KDD” and Data Mining was first formally put forward by Usama Fayaad, (who began working in the field in 1989 at NASA’s Jet propulsion Lab, compiling data on astronomical phenomena.) at the first International conference on Knowledge Discovery and Data Mining, held in Montern in 1995[PAOL03].

The problem of discovering co-occurrence of an item in small data is a very simple task. However, the large volume of data makes this problem massively difficult and efficient algorithms are needed [RAKE94].

The problem of discovering association rule is decomposed into two parts: i) discovering all frequent patterns (represented by large itemsets or frequent itemsets), in the database,

and ii) generating the association rules from those frequent itemsets. The second part is a straightforward problem and can be managed in polynomial time. On the other hand, the first task is very difficult especially for large databases. The typical example of association rule is the basket data analysis. In the basket database, all the records consist of two fields: Transaction ID (TID) and the item the customer bought in the transaction. Usually the transaction consists more than one item. An itemset is a set of items. It may be frequent, (large), or infrequent, (small). It is called frequent if the number of the occurrences of its items together in a database is greater then or equal to a use-defined threshold known as minimum support (minsup); otherwise it is called small or infrequent [RAKE94].

The current association rule mining algorithms [RAKE94, RAKE96a, OGIH97a, SEGE97, ROBE99, CHAR98, OGIH00, BUND01, HIPP01, and GOUD01] are iterative and use repeated scans on the database causing massive I/O traffic. Most of them use complex data structures such as hashing trees. Also, most of them work to mine frequent itemsets, except the algorithms presented in [GOUD01 AND BUND01] which work to mine maximal itemsets. Maximal itemsets is a frequent itemset and not subsets of other frequent itemsets. Usually the number of frequent itemsets is very large and increases with the decreasing of minsup value. There for, the mining of them is computational intensive process and has proved as NP-complete problem [OGIH98]. The number of maximal itemset is very small relative to the cardinality of frequent itemsets. There for, the algorithm that mine maximal itemsets are fast algorithms. But these algorithms are doomed to fail in the second phase of rule mining because it requires the maximal itemsets to be degenerated to there subsets [HIPP01, MICH02].

Another important defect with the existent algorithms for association rules mining is their insensitivity to the knowledge changes that happen when the database is updated. These algorithms must be re-run to discover the withered and emerging itemsets of the updated database. In addition to these drawbacks mentioned the available association rules algorithms consist of steps that require programming languages statement and facilitates.

Such statements are not usually supported by DBMSs, Therefore the nature of algorithms is the cause of the bad integration of association rules mining and DBMSs[ABRA97, HIPP01, and RAME01].All the algorithms but DICT [SEGE97], can not be implemented by using DBMS tools such as SQL. DICT undergoes large and many problems with large and dense databases.

Recently data mining has been the province of high level specialist academic researchers and has featured complex custom programming and/or very high end software products, high cost and prolonged delivery schedules. Now, some front end tool venders are attempting to provide packaged commercial software products that combine simple capabilities with relatively easy to use GUI interfaces. These tools provide simpler capabilities instead of complex, esoteric ones. The idea is to get limited, but may be useful, data mining to a larger audience; the customer gets some of the benefits of data mining in a compressed time frame and at reduced cost [ODWH01]. Although many efforts have been put in the association area, and some commercial products, such as MineSet, intelligent miner, and Zhang tool have been developed. There are several limitations in these commercial products. Some of them are listed below:

- The existing miners manipulate data file in specific format, (usually from flat file) and they can not connect to multiple database management systems. For example the Intelligent Rule Miner can use the text file or the data of DB2 database as the data source [HUSS02].
- The available rule miners can only mine rules from one file or table. It is often required to combine data from more than one data source; therefore; it is important to develop miner having the ability to connect many files or tables to generate a suitable data set for mining [HUSS02].
- The available rule miners do not split the two phases: the frequent item generation and the rule generation and make one run for the mining. The separation of these two phases can reduce work time significantly by working at dead time in many cases.
- Because of the pruning process during the rule generation, there is no chance to find weak association rules instead of strong ones.

1.3 Aim of the Study

The aim of this study is to overcome some of the limitations and restrictions, which listed in the previous section, through proposing a new approach for mining association rules.

The proposed technique divides the running time of the association rule mining process into two parts first part runs at any time that there is no need for the server this part finalizes all process expensive tasks and other part runs at user query that not require many process time as a result users not wait too much for computer to answer queries. This is done by adding a layer between two processes which is output of the first process and second process uses this ready and easy data for finding association rules.

1.4 Thesis organization

This thesis is organized as follows: Chapter 1 of this thesis has been devoted to give a general definition to data mining and knowledge discovery with a brief historical survey. The rest of the thesis is organized as follows: Chapter 2 presents the fundamentals of data mining and association rules in databases. The main activities of data mining are explained and the principles and algorithms of association rules mining are presented. Chapter 3 describes the design of the Databot, which consists of the proposed algorithms and efficient methods to store and retrieve itemsets and their frequencies from database. Chapter 4 involves technical implementation by using java/MySql and algorithm verifications. Finally, conclusions and suggestions for future researches are given in chapter 5.

Chapter Two

Theory of Data Mining and Association Rules

2.1 Introduction

Data mining can be defined as "The process of selection, exploration and modeling of large quantities of data to discover regularities or relations that are at first unknown with the aim of obtaining clear and useful results for the owner of the database" [PAOL03].

This definition tries to explain nature of the data mining but there is other definition which they focus on other sides of the data mining. Here three more definitions for data mining:

- Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner [DAVI01].
- Data mining is an interdisciplinary field bringing together techniques from machine learning, pattern recognition, statistics, databases, and visualization to address the issue of information extraction from large data bases [PETE98].
- Data mining is the process of discovering interesting knowledge from large amount of data stored in databases, data warehouses or other information repositories [JIAW01].

There are many other definitions of data mining according to its application field and task show that data mining is an interdisciplinary field, using methods of several research areas to extract high level knowledge from real world data sets.

2.2 An Overview of Data Mining and Knowledge Discovery

In essence, data mining consists of the (semi-)automatic extractions of knowledge from data. This statement raises the question of what kind of knowledge we should try to discover. Although this is a subjective issue, we can mention three general properties that the discovered knowledge should satisfy; namely, it should be accurate, comprehensible, and interesting [PETE98].

In data mining people are often interested in discovering knowledge which has a certain predictive power. The basic idea is to predict the value that some attributes will take on "the future", based on previously observed data. The discovered knowledge should have a high predictive accuracy rate [PETE98].

It is also, important that the discovered knowledge should be comprehensible for the user. This is necessary whenever the discovered knowledge is to be used for supporting a decision to be made by a human being. If the discovered "knowledge" is just a black box, which makes predictions without explaining them, the user may not trust it [MICH94]. Knowledge comprehensibility can be achieved by using high-level knowledge representation. A popular one, in the context of data mining, is a set of IF-THEN (prediction) rules, where each rule is of the form:

IF<some_conditions_are_satisfied>
THEN<predict_some_value_for_an_attribute>

The third property, knowledge interestingness, is the most difficult one to define and quantify, since it is, to a large extend, subjective. However, there are some aspects of knowledge interestingness that can be defined in objective terms. Subjective methods are user-driven and domain-dependent. For example, a user may specify rule templates, indicating which combination of attributes must occure in the rule for it to be considered interesting [KLEM94].

By contrast, objective methods are data-driven and domain-independent. Some of these methods are based on idea of computing a discovered rule against other rules [GIOR94].

2.2.2 Data Mining Tasks

In this section we briefly review the major data mining tasks. Each task can be thought of as a particular kind of problem to be solved by a data mining algorithm.

1. Classification and Prediction
2. Dependence Modeling
3. Clustering
4. Association

The first three tasks are examples of Directed Data Mining (Predictive) [PANG06], Which uses some variables to predict unknown or future values of other variables and the last one is Undirected Data Mining (Descriptive) [PANG06], which finds human-interpretable patterns that describe data [MICH04, JIAW01].

I. Classification and Prediction

This is probably the more studied data mining task. It has been studied for many decades by the machine learning and statistics communities. In this task the goal is to predict the value (the class) of a user-specified goal attribute based on the values of other attributes, called the predicting attributes. For instance the goal attribute might be the Credit of a bank customer in a Banking database environment, taking on values (classes) "good" or "bad", while the predicting attributes might be the customer's Age, Salary, Current_account_balance, whether or not the customer has an Unpaid_Loan [HAND97].

Classification rules can be considered as a particular kind of prediction rules where the rule antecedent ("IF part") contains a combination of conditions on predicting attribute values, and the rule consequent ("THEN part") contains a predicated value for the goal attribute. Examples of classification rules are:

IF(Unpaid_Loan = "no") and (Current_account_balance > $3000)
THEN (Credit = "good")
IF(Unpaid_Loan = "yes") THEN (Credit = "bad")

In the classification task the data being mined is divided into two mutually exclusive and exhaustive data set, the training set and the test set. The data mining algorithm has access to the values of both the predicting attributes and the goal attributes of each example (record) in the training set [HAND97].

Once the training process is finished and the algorithm has found a set of classification rules, the predictive performance of these rules is evaluated on the test set, which was not seen during training [HAND97].

Actually, it is trivial to get 100% of predictive accuracy in the training set by completely sacrificing the predictive performance on the test set, which would be useless. To see this, suppose that for a training set with n example, such that for each "discovered" rule: (a) the rule antecedent contains conditions with exactly the same attribute-value pairs as the

corresponding training example; (b) the class predicted be the rule consequent is the same as actual class of the corresponding training example. In this case "discovered" rules would trivially achieve a 100% of predictive accuracy on the training set, but would be useless of predicting the class of examples unseen during training. In other words there would be no generalization, and "discovered" rules would be capturing only idiosyncrasies of the training set, or just "memorizing" the training data. In the parlance of machine learning and data mining, the rules would be over fitting the training data [HAND97].

Some examples of classification task [MICH00]:

- Choosing content to be displayed on a Web page
- Determining which phone numbers correspond to fax machines
- Spotting fraudulent insurance claims
- Assigning industry codes and job designations on the basis of free-text job descriptions

II. Dependence Modeling

This task can be regarded as a generalization of the classification task. In the former task, we want to predict the value of several attributes rather than a single goal attribute, as in classification. We focus again on the discovery of prediction (IF-THEN) rules, since this is a high-level knowledge representation [HAND97].

It is most general form, any attribute can occur both in the antecedent ("IF part") of a rule and in the consequent ("THEN part") of another rule, but not in both the antecedent and the consequent of the same rule. For instance, and for the same Banking data environment, we might discover the following two rules:

IF (Current_account_balance > $3000) AND (Salary = “hight”)
THEN (Credit = “good”)
IF (Credit = “good”) AND (Age > 21) THEN (Grant_Loan? = “yes”)

In some cases we want to restrict the use of certain attributes to a given part (antecedent or consequent) of a rule. For instance, we might specify that the attribute Credit can occur only in the consequent of a rule, or that the attribute Age can occur only in the antecedent of a rule [HAND97].

III. Clustering

As mentioned above, in the classification task the class of a training example is given as input to the data mining algorithm, characterizing a form of supervised learning. In contrast, in the clustering task the data mining algorithm must, in some sense, “discover” classes by itself, by partitioning the examples into clusters, which is a form of unsupervised learning [FLOC95].

Examples that are similar to each other (i.e. examples with similar attribute values) tend to be assigned to the dame cluster, whereas examples different from each other tend to be assigned to distinct clusters. Note that, once the clusters are found, each cluster can be considered as a “class”, so that now we can run a classification algorithm on the clustered data, by using the cluster name as class label [PARK98].

IV. Association Rules

In the standard form of this task each data instance (or “record”) consists of a set of binary attributes called items. Each instance usually corresponds to a customer transaction, where a given item has a true or false value depending on the whether or not the corresponding customer bought that item in that transaction. An association rule is a relationship of the form IF X THEN Y, where X and Y are Sets of items and $X \cap Y = \emptyset$ [RAKE93, RAKE96b]. As an example for a supermarket database, the association rule is:

IF fried_potatoes THEN soft_drink, ketchup.

Although both classification and association rules have an IF-THEN structure, there are important difference between them. We briefly mention here two of the main differences. First association rules can have more than one item in the rule consequent, whereas classification rules always have one attribute (the goal one) in consequent. Second, unlike the association task, the classification task is asymmetric with respect to the predicting attributes and the goal attribute. Predicting attributes can occur only in the rule consequent [HAND97].

This research is devoted to association rules mining, so, this task is discussed in detail in the rest of this chapter.

2.2.3 Data Mining Architecture

Data mining systems contain some common components which they interact together to perform data mining task. Typical data mining system architecture is shown in figure 2.1[JIAW01].

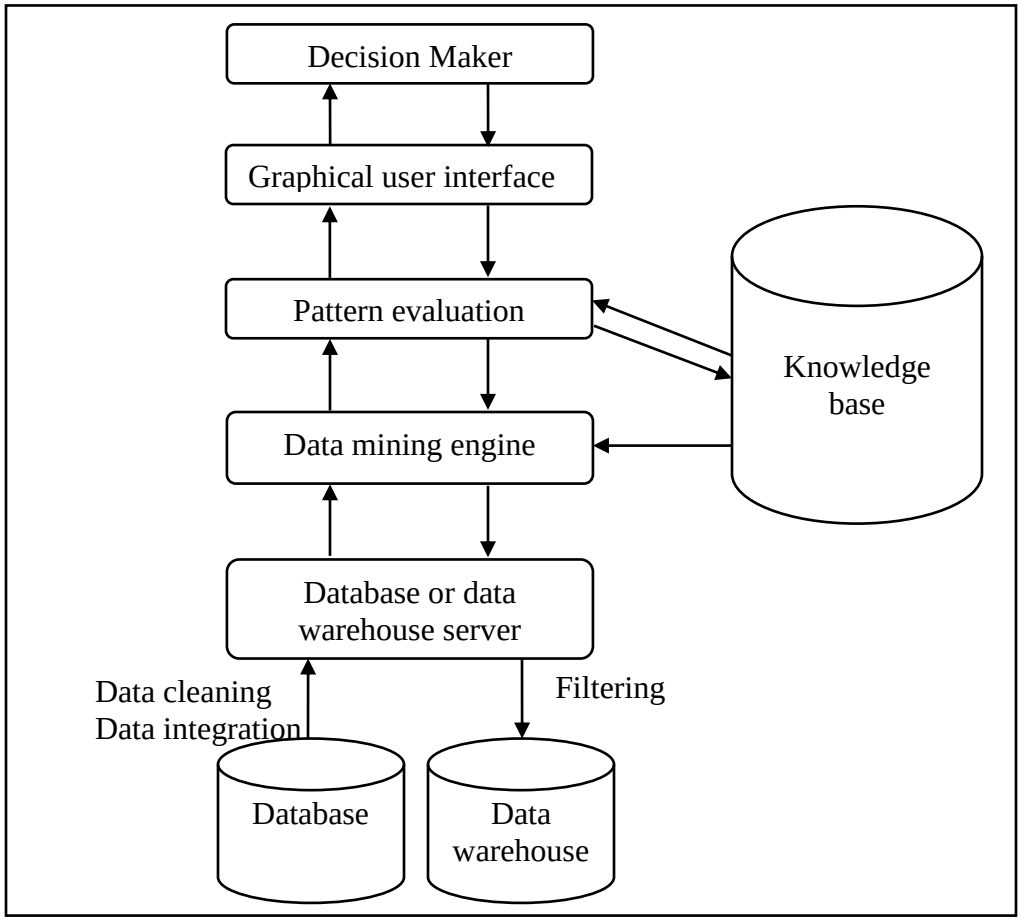


Figure 2.1 Architecture of a typical data mining system

The architecture contains the following components [JIAW01]:

- **Database or data warehouse server:** It is responsible for fetching the relevant data, based on the user's data mining request. "In the proposed system this module has been changed dramatically by saving frequencies of the data patterns instead of saving actual data or its variations".

- **Knowledge base:** this is the domain knowledge that is used to guide the search, or evaluate the interestingness of resulting patterns. Such knowledge can include concept hierarchies, used to organize attributes or attribute values in to deferent levels of abstraction knowledge. Another example of domain knowledge is deferential interestingness constraints or thresholds, and metadata (e.g., describing data from multiple heterogeneous sources).

- **Data mining engine:** this is essential to the data mining system and ideally consists of a set of functional modules for data mining tasks.

- **Pattern evaluation:** this concept typically employs interestingness measures and interacts with the data mining models so as to focus the search towards interesting patterns. It may use interestingness thresholds to filter out discovered patterns. Alternatively, the patterns evaluation module may be integrated with the mining module, depending on the implementation of the data mining method used.

- **Graphical user interface:** this module communicates between users and data mining system, allowing them to interact with the system by specifying a data mining query or task, providing information to help focus the search, and performing exploratory data mining based on the intermediate data mining results.

2.2.4 Data Mining Life Cycle

The Cross-Industry Standard Process for Data Mining (CRISP–DM) was developed in 1996 by analysts representing DaimlerChrysler, SPSS, and NCR [PETE00]. CRISP provides a nonproprietary and freely available standard process for fitting data mining into the general problem-solving strategy of a business or research unit [DANI05, PETE00].

According to CRISP–DM [PETE00], a data mining project has a life cycle consisting of six phases. The CRISP–DM life cycle is shown in Figure 2.2. The phase sequence is adaptive, that means the next phase in the sequence often depends on the outcomes associated with the preceding phase. The iterative nature of CRISP is symbolized by the outer circle in Figure 2.2.

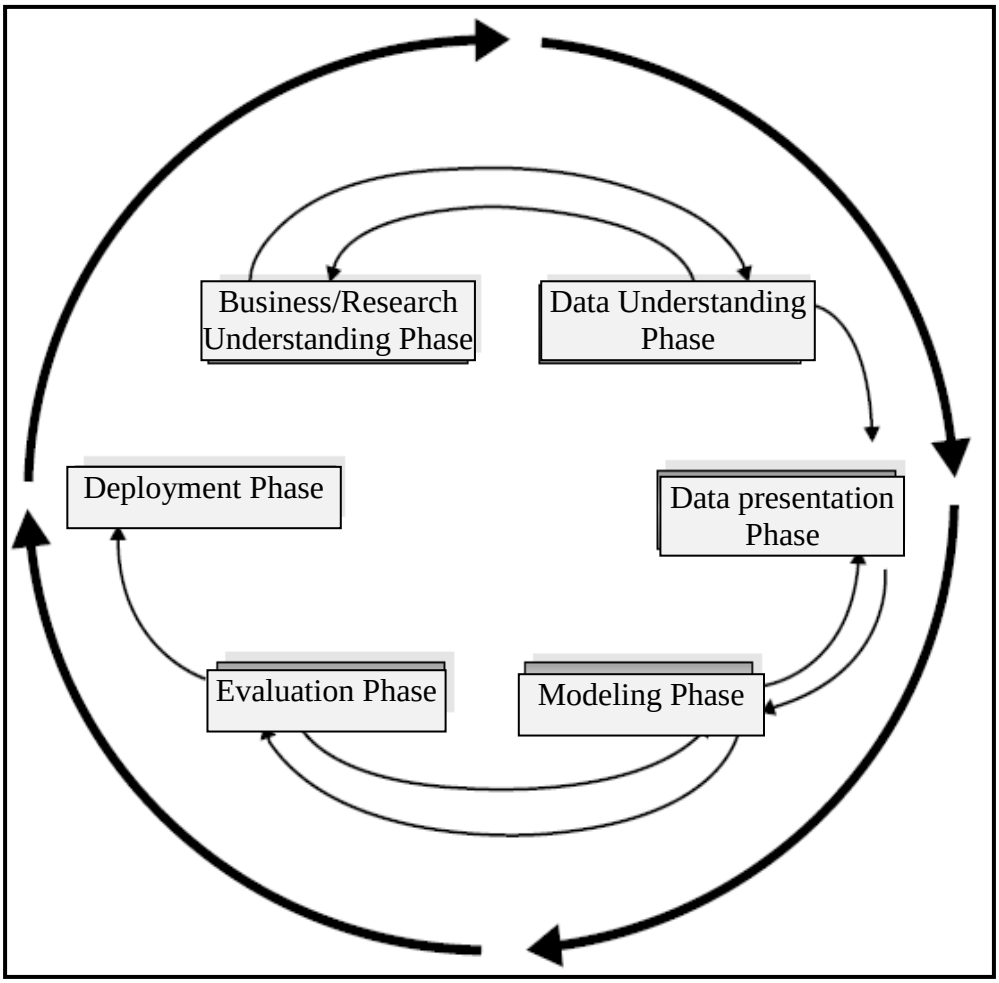


Figure 2.2 CRISP-DM life Cycle is an iterative, adaptive process

The Six Phases of CRISP–DM are [DAIN05]:

1. **Business understanding phase.** The first phase in the CRISP–DM standard process may also be termed the research understanding phase.
   a. Enunciate the project objectives and requirements clearly in terms of the business or research unit as a whole.
   b. Translate these goals and restrictions into the formulation of a data mining problem definition.
   c. Prepare a preliminary strategy for achieving these objectives.
2. **Data understanding phase**
   a. Collect the data.
   b. Use exploratory data analysis to familiarize yourself with the data and discover initial insights.
   c. Evaluate the quality of the data.
   d. If desired, select interesting subsets that may contain actionable patterns.
3. **Data preparation phase**
   a. Prepare from the initial raw data the final data set that is to be used for all subsequent phases. This phase is very labor intensive.
   b. Select the cases and variables you want to analyze and that are appropriate for your analysis.
   c. Perform transformations on certain variables, if needed.
   d. Clean the raw data so that it is ready for the modeling tools.
4. **Modeling phase**
   a. Select and apply appropriate modeling techniques.
   b. Calibrate model settings to optimize results.
   c. Remember that often, several different techniques may be used for the same data mining problem.

d. If necessary, loop back to the data preparation phase to bring the form of the data into line with the specific requirements of a particular data mining technique.

5. **Evaluation phase**

a. Evaluate the one or more models delivered in the modeling phase for quality and effectiveness before deploying them for use in the field.
b. Determine whether the model in fact achieves the objectives set for it in the first phase.
c. Establish whether some important facet of the business or research problem has not been accounted for sufficiently.
d. Come to a decision regarding use of the data mining results.

6. **Deployment phase**

a. Make use of the models created: Model creation does not signify the completion of a project.
b. Example of a simple deployment: Generate a report.
c. Example of a more complex deployment: Implement a parallel data mining process in another department.
d. For businesses, the customer often carries out the deployment based on your model.

2.3 Association Rules Mining

Association rules mining is one of the main aspects of data mining, and have been widely studied in recent years. Association rule mining finds interesting association relationships among a large set of data items. With massive amounts of data continuously being collected and stored in databases. Many industries are becoming interested in mining association rules from their databases. For example, the discovery of interesting association relationships among huge amounts of business transaction records can help catalog design, cross marketing, loss leader analysis, and other business decision-making processes.

An association rule X⟹Y | c, is a statement of the form "for a given set of items a particular value of itemset X determines the value of itemset Y as another particular value". Thus, association rules aim for discovering the patterns of co-occurrence of itemsets in a database. For instance, an association rule in a supermarket data may be "In 5% of transactions, 85% of the people buying bread and yogurt in the same transaction".

The problem of discovering association rules was first explored in [RAKE93] on supermarket basket data, that was a set of transactions that include items purchased by the customers. In the pioneering work, the data was considered as binary, which means every transaction is a binary array each element of this array is an item of the supermarket.

2.3.1 Definitions and General Terms

At 1994 Rakesh Agrawal, Tomasz Imielinski and Arun Swami defined association rules as: "Let $I = I_1, I_2,\ldots, I_m$ be a set of binary attributes, called items. Let T be a database of transactions. Each transaction t is represented as a binary vector, with t[k] = 1 if t bought the item I_k, and t[k] = 0 otherwise. There is one tuple for each transaction. Let X be a set of some items in I. we say that a transaction t satisfies X if for ass items I_k in X, t[k] = 1.

By an association rule, we mean an implication of the form $X \Longrightarrow I_j$, where X is a set of some items in I, and I_j is a single item that is not present in X. The rule $X \Longrightarrow I_j$ is satisfied in the set of transactions of T with the confidence factor $0 \leq c \leq 1$ Iff at least c% of the transactions in T that satisfy X also satisfies I_j." [RAKE93, RAKE96a, RAKE96b].

Definitions of the main terms of association rules are:

1. **Itemset**: is a subset of items for the original set of the problem. Any itemset is called k-itemset if it contains k items from the original set. In data mining literature Itemset is used instead of item set.
2. **Support Count**: For itemset X, is the number of transactions that contain X as a subset in the database.

3. **Support**: For itemset X, is the percentage ratio of support Count of X on the total number of transactions. in other word

   $$\text{support}(X) = \frac{Support\_Count(X)}{\sum Transactions} \quad (2.1)\ [JIAW01]$$

4. **Confidence**: for any A $\Longrightarrow$ B has confidence c in transaction set ***D*** c is the percentage of transactions in ***D*** contains A that also contains B. this is taken to be conditional probability

   $$p(B|A) = \frac{suppor\_count(A \cup B)}{suppor\_count(A)} \quad (2.2)\ [JIAW01]$$

5. **Frequent Itemset:** is an itemset whose support is greater than or equal to a minimum support threshold. In early works, this term is referred as large itemset. [JIAW01, RAKE93]
6. **Association Rule:** is an implication of the form $X \Longrightarrow Y$ where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. X is called the antecedent of the rule, and Y is called the consequence of the rule.

2.3.2 Association Rules Mining and Its Variations

Association rules mining is used in various areas in the real world from market basket analysis to finding patterns in galaxy images and finding relations between diseases and various human parameters like body mass, blood pressure temperature environment and so on. Because of its wide usage, there are many types of association rule mining, and it can be classified in various ways, based on the following criteria [JIAW01]:

- **Based on the types of the values handled in the rule**: If a rule concerns associations between presence or absence of items, then it is a **Boolean association rule**.

  For example

  $$\text{computer} \Longrightarrow \text{windows\_OS} \quad (2.3)$$

  Is a Boolean association rule. If a rule describes associations between quantitative items or attributes, then it is a quantitative association rule. In these rules,

quantitative values for items or attributes are partitioned in to intervals. The following rule is an example of a quantitative association rule, where X is a variable representing customer:

$$age(X, \text{“30…39”}) \wedge income(X, \text{“42k…48k”}) \Rightarrow buys(X, \text{high resolution TV}) \quad (2.4)$$

Note that quantitative attributes, age and income, have been discredited.

- **Based on the dimensions of data involved in the rule**: If the items or attributes in an association rule reference only one dimension, then it is a single dimensional association rule. Note that rule (2.3) could be rewritten as

  $$buys(X,\text{”computer”}) \Rightarrow buys(X,\text{”windows\_OS”}) \quad (2.5)$$

  This rule is a single dimensional association rule since it refers to only one dimension: buys. If a rule references tow or more dimensions, such as the dimensions buy, time_of_transaction, and customer_catagory, then it is a multidimensional association rule. Rule (2.4) is considered as a multidimensional associational rule since it involves three dimensions: age, income, and buys.

- **Based on the levels of abstractions involved in the rule set:** some method for association rule can find rules at differing levels of abstractions. For example, suppose that a set of association rules mined includes the following rules:

  $$age(X, \text{”30…39”}) \Rightarrow buys(X, \text{”laptop computer”}) \quad (2.6)$$

  $$age(X, \text{”30…39”}) \Rightarrow buys(X, \text{”computer”}) \quad (2.7)$$

  In rules above the items bought reference at deferent levels at different levels of abstraction. (e.g., “computer” is a higher-level abstraction of “laptop computer”).

  We refer to the rule set mined as consisting of multilevel association rule. If the rules within a given set do not reference items or attributes as deferent level of abstraction, then the set contains single-level association rules.

- **Based on various extensions to association mining:** association mining can be extended to correlation analysis, where the absence or presence of correlated items can be identified, can also be extended to mining maxpatterns (i.e., maximal frequent patterns) and frequent closed itemsets. Maxpattern is a frequent pattern, P such that any proper super pattern of P is not frequent. A frequent closed itemset is a frequent closed itemset where an itemset c is closed if there exists no proper superset of c, c', such that every transaction containing c also contain c'. maxpattern and frequent closed itemsets can be used to substantially reduce the number of frequent itemsets generated in mining.

Example 2.1

The above description of association is more clarified through this simple example. (This example is a Boolean, single-dimension, and single-level association rule mining is rule) Consider the example transaction database$_{ETDB}$ in Table 2.2. There are six transactions in the database with Transaction IDentifiers, TIDs, 1, 2, 3, 4, 5, and 6. The set items I= {A, B, C, D, E, F), each item is an abbreviation of an attribute of census data as shown in table 2.1. There are totally (26 - 1) = 63 nonempty itemsets (each nonempty subset of I is an itemset). (A) is a l-itemset and {AB} is a 2-itemset, and so on. Support$_{ETDB}$ (A) = 3 since three transaction include A in it. Let us assume that the minimum support (minsup) is two, (approximately taken as 33%). Then {A, B, C, D, E, AB, AC, AE, BC, BD, BE, CD, CE, DE, ABC, ABE, ACE, BCD, BCE. BDE, CDE, ABCE, BCDE} are the set of large itemsets, since their support is greater than or equal to 2, (33% x 6), and the remaining ones are small itemsets. There are two distinct itemsets, ABCF and BCDE, called maximal itemsets; all other large itemsets are subsets of one of them. Table 2.3 depicts all large itemsets with their supports. Let's assume that the minimum confidence (minconf) is set to 100%. Then, A $\Longrightarrow$ B is an association rule with respect to the specified minsup and minconf (its support is 3, and its confidence is

$\frac{\text{SupportETDB(AD)}}{\text{SupportETDB(A)}} \times 100 = 3/3 \times 100 = 100\%$). On the other hand, the rule B ⟹ A is not valid association rule since its confidence is 50% this shows that the association rule is not always interchangeable sides in real world applications. If the rule sides are interchangeable, the new rule rarely has the same confidence value. This property Increases the complexity of rule mining process. Table 2.4 depicts the association rule that can be mined from databaseEDTB according to 100% minconf value and 33% minsup value. It is obvious that the increment of minimum-support value reduces the number of large or frequent item set and vice versa. On the other hand, the increment of minimum-confidence value diminishes the number of valid association rules mined according to this value from the set of frequent itemsets, which they are extracted depending on the minsup value.

The rule ABE ⟹ C | (2/2 or 100%) means any person who has more than three children, serves in military, and can drive ⟹ he born in Iraq with 100%. [HUSS02].

Table 2.1 the items abbreviations of database$_{ETDB}$

Item	Attribute(1)	Possible non-attribute value
A	Has more than 3 children	3 or lee children
B	Veteran	Never served in military
C	Born in Iraq	Born abroad
D	Married	Single, divorced, widowed
E	Drive	Does not drive
F	House holder	Dependent, boarder, renter

Table 2.2 a transaction Database

TID (Person)	Items (attributes)
1	B, C, E
2	B, C, D, E
3	A, B, C, D, E
4	B, C, D
5	A, B, F
6	A, B, C, E

Table 2.3 frequent itemsets with minsup = 33% = 2

Support	Items	No.
6 = 100%	B	1
5 = 83%	C, BC	2
4 = 67%	E, BE, CE, BCE	4
3 = 50%	A, D, AB, BD, CD, BCD	6
2 = 33%	AC, AE, DE, ABC, ABE, ACE, BDE, CDE, ABCE, BCDE	10

Table 2.4 association rules

Association rules with minconf = 100%		
A ⟹ B (3/3)	AC ⟹ B (2/2)	AC ⟹ BE (2/2)
C ⟹ B (5/5)	AE ⟹ B (2/2)	AE ⟹ BC (2/2)
D ⟹ B (3/3)	AC ⟹ E (2/2)	DE ⟹ BC (2/2)
E ⟹ B (4/4)	AE ⟹ C (2/2)	ABC ⟹ E (2/2)
D ⟹ C (3/3)	DE ⟹ B (2/2)	ABE ⟹ C (2/2)
E ⟹ C (4/4)	DE ⟹ C (2/2)	ACE ⟹ B (2/2)

2.4 Apriori Algorithm

The Apriori algorithm is a state of the art algorithm and most of the association rule algorithms are somehow variation of this algorithm. Thus, it is necessary to mention Apriori in detail for an introduction to association rule algorithms, and we will content with brief analysis to another algorithms.

The ideal situation for Apriori algorithm is the problem of mining association rules in a set of transactions D to generate all association rules that have support and confidence greater than the user specified minsup and minconf respectively. Formally, the problem is generating all association rules $X \Longrightarrow Y$, where $support_D$ $(X \Longrightarrow Y)$ minsup $|\mathbf{D}|$ and $\frac{supportD(X \cup Y)}{supportD(X)}$ > minconf.s

As previously mentioned, the problem of finding association rules can be decomposed into two parts [RAKE93, RAKE94, RAKE96b]:

1. Generate all combinations of items with fractional transaction support ($\frac{SupportD(X)}{|D|}$) above a certain threshold, called minsup.
2. Use the frequent itemsets to generate association rules. For every frequent itemset l[1], find all non-empty subsets of l. For every such subset a, output a rule of the form a $\Longrightarrow$ (l - a) if the ratio of $support_D$(l) to $support_D$(a) is at least minconf if an itemset is found to be large in the first step, the support of that itemset should be maintained in order to compute the confidence of the rule in the second step.

The input to the apriori algorithm is a horizontal database of two fields TID (Transaction IDentifier) and its items (Purchased Items). The apriori algorithm works iteratively, it first finds the set of frequent 1-itemsets, and then set of 2-itemsets, and so on. The number of scans over the transaction database is as many as the length of the maximal itemset. Apriori is based on the following fact: “All subsets of a frequent itemset are also

[1] Because of the early works which refer to the frequent itemset as large itemset we denote frequent itemset as *l*

frequent" [RAKE96b]. This simple but powerful observation leads to the generation of a smaller candidate set using the set of frequent itemsets found in the previous iteration. The Apriori algorithm presented in [RAKE96b] is given in Algorithm 2.1.

```
Procedure Apriori ()
   L1 – {large 1-itemsets}
   k = 2
   while lk-1 ≠ 0 do
        begin
              Ck = apriori_gen(Lk -1) //Calling for apriori_gen
              for all transactions t in D do
              begin
              Ct = subset(Ck , t)
              for all candidates c ∈ Ct do
              c.count = c.ount + 1
              end
              Lk = {c ∈ Ck | c.count ≥ minsup}
   k = k +1
   end
```

Algorithm 2.1 Apriori Algorithm

```
apriori_gen (Lk-l);
   Ck = Ø
   for all itemsets X ∈ Lk-l d o
        if X1 = Y1 ^ … ^ Xk-2 = Y k-2 ^ Xk-1 < Yk-l then begin
              C = X1X2 … Xk-1Yk-1
              add C to Ck
   end
  delete candidate itemsets in Ck whose any subset is not in Lk-1
```

Algorithm 2.2 Candidate Generation Algorithm

Apriori first scans the transaction databases **D** in order to count the support of each item i in I, and determines the set of large I-itemsets. Then, one iteration is performed for each of the computation of the set of 2-itemsets, 3-itemsets, and so on. The k^{th} iteration Consists of two steps [RAKE94]:

1. Generate the candidate set C_k from the set of large (k- l)-itemsets, L_{k-1}.
2. Scan the database in order to compute the support of each candidate itemset in C_k.

The candidate generation procedure computes the set of potentially large k-itemsets from the set of large (k-1)-itemsets. A new candidate k-itemset is generated from two large (k-l)-itemsets if their first (k-2) items are the same (The new itemset contains the items in those two frequent itemsets in order).

In fact, the candidate set C_k is a superset of large k-itemsets. The candidate set is guaranteed to include all possible frequent k-itemsets because of the fact that all subsets of a frequent itemset are also frequent. Since all frequent itemsets in L_{k-1} are checked for contribution to a candidate itemset, the candidate set C_K is certainly a superset of large k-itemsets. The pruning step in apriori_gen function is necessary to reduce the size of the candidate set. For example, if L_{k-1} includes AB, AC, then a candidate ABC is generated in the join step of apriori_gen. However, it can not be a frequent itemset if Lk-1 does not include BC, so it can be pruned from the candidate set. For efficiently finding whether a subset of a frequent itemset is small or not, a hash table is used for storing the frequent itemsets [RAKE94].

After the candidates are generated, their counts must be computed in order to determine which of them are frequent. The counting step of an association rule algorithm is very crucial in the efficiency of the algorithm, because the set of candidate itemsets may be possibly huge. Apriori handles this problem by employing a hash tree for storing the

candidates. The subset function in apriori_gen is used to find the candidate itemsets contained in a transaction using this hash tree structure. For each transaction t in the transaction database D, the candidates contained in t are found using the hash tree, and then their counts are incremented. After examining all transactions in D, the set of candidate itemsets are checked to eliminate the non-frequent itemsets, and the ones that are frequent are inserted into L_k [RAKE96b].

The second sub problem is straightforward, and an efficient algorithm for extracting association rules from the set of frequent itemsets is presented in. The algorithm uses some heuristics as follows [RAKE96b]:

1. If a $\Longrightarrow$ (l - a) does not satisfy the minimum confidence condition, then for all non-empty subsets b of a, the rule b $\Longrightarrow$ (l - b) does not satisfied the minimum confidence, either because, the support of a is less than or equal to the support of any subset b of b
2. If (l - a) $\Longrightarrow$ a satisfies the minimum confidence, then all rules of the form of (l - b) $\Longrightarrow$ b must have confidence above the minimum confidence.

The rule generation algorithm is given in algorithm 2.1. Firstly, for each frequent itemset 1, all rules with one item in the consequent are generated. Then, the consequents of these rules are used to generate all possible rules with two items in the consequent, etc. The apriori_gen function in Algorithm 2.2 is used for this purpose [RAKE94].

On the other hand, discovering frequent itemsets is a non-trivial issue. The efficiency of an algorithm strongly depends on the size of the candidate set. The smaller the number of candidate itemsets is, the faster the algorithm will be. As the minimum support threshold decreases, the execution times of these algorithms increase because the algorithm needs to examine a larger number of candidates and larger number of itemsets [HUSS02].

```
Generate_rule(L);
 1  for all large K-itemset l_k k ≥ 2, in L do
 2  begin
 3     H_1 ={consequents of rules from l_k with one item
 4              in the subsequent}
 5     ap_genrules(l_k, H_1)
 6  end

 7  ap_genrules (l_k, H_m)
 8  if  k > m +1 then
 9  begin
 10    H_m+1 = apriori_gen(H_m)
 11    for all h_m+l ∈ H_m+1  do
 12    begin
 13         conf = support_D (l_k) / support_D (l_k – h_m+l)
 14         if conf >= minconf then add (l_k- h_m+l)  ⇒  h_m+l to the rule set
 16         else  delete h_m+l from H_m+l
 18   end
 19   ap_genrules(l_k, H_m+l)
 20 end
```

Algorithm 2.3 Rule Generation Algorithm

Example 2.2

To describe the Apriori algorithm, the same transaction database of example 2.1 (Table 2.1) is used. Suppose that the minimum support is set to 50%, i.e., 3 transactions. In the first pass. L1 = {A, B, C, D, E}. The apriori_gen function computes C_2 = {AB, AC, AD, AE, BC, BD, BE, CD, CE, DE}. The database is scanned to find which of them are frequent, and it is found that L_2 = {AB, BC, BD, BE, CD, CE}. This set is used to compute C_3. In the join step BCD, BCE, BDE and CDE are inserted into C3. However, BDE and CDE can not be frequent because DE is not an element of L2. Thus, BDE and CDE are pruned from the set of candidate itemsets. The database is scanned and it is found that L3 = {BCD, BCE}. C4 is found to be {BCDE). However, BCDE cannot be frequent because CDE is not an item of L3, and the algorithm terminates. Figure 2.3 shows the overall process of extracting frequent itemsets depending on Apriori algorithms [HUSS02].

L1	C2	L2	C3	L3	C4	L4
A	AB	AB	BCD	BCD	BCDE	{}
B	AC	BC	BCE	BCE		
C	AD	BD	BDE			
D	AE	BE	CDE			
E	BC	CD				
	BD	CE				
	BE					
	CD					
	CE					
	DE					

Figure 2.3 the overall process of Apriori algorithm

2.5 Variations of Apriori Algorithm

The major drawback of the Apriori is the number of scans over the database. Especially for the huge databases, the I/O overhead incurred reduces the performance of the algorithm. Apriori also works iteratively and it makes as many scans as the length of maximal itemset over the database. The candidate k-itemsets are generated from the Sets of frequent (k-1)-itemsets by means of join and pruning operations. Then the itemsets in the candidate set are counted by scanning the database. Apriori forms the foundation of the later algorithms on association rules. In [RAKE96b], two variations of Apriori were also presented to overcome this I/O cost.

2.5.1 Apriori_TID and Apriori Hybrid Algorithms

The Apriori_TID algorithm constructs an encoding of the candidate itemsets and uses this structure to count the support of itemsets instead of scanning the database. This encoded structure consists of elements of the form < TID, $\{X_k\}$ > where each X_k is a frequent k-itemset. In other words, the original database is converted into a new table where each row is formed of a transaction identifier and the frequent itemsets contained in that transaction. The counting step is over this structure instead of the database. After identifying new frequent k-itemsets, a new encoded structure is constructed. In subsequent passes, the size of each entry decreases with respect to the original transactions and the size of the total database decreases with respect to the original database. Apriori_TID is very efficient in the later iterations but the new encoded structure may require more space than the original database in the first two iterations.

To increase the performance of Apriori_TID a new algorithm, Namely Apriori Hybrid, was proposed in [RAKE96b]. This algorithm uses Apriori in the initial passes, and then swhiches to Apriori_TID when the size of the encoded structure fits into main memory. In this sense, it takes benefits of both Apriori and Apriori_TID to efficiently mine association rules.

The three algorithms mentioned above scale linearly with the number of transactions and the average transaction size. Apriori _TID and Apriori _Hybrid [RAKE94, RAKE96b] have the similar ideas in Apriori. The former uses an encoded structure which stores the itemsets that exist in each transaction. In other words, the items in the transaction are converted to an itemset representation. The candidates are generated as in Apriori but they are counted over the constructed encoding. The latter algorithm tries to get benefits of both Apriori and Apriori_TID by using Apriori in the initial passes and swhiching to the other in later iterations. Both algorithms make as many passes as the length of maximal itemset [HUSS02].

2.5.2 Partition Algorithm

The Partition algorithm [ASHO95, RAME01] is another algorithm to mine association rules. Indeed, it is parallel Apriori algorithm. The major advantage of Partition algorithm is scanning the database exactly twice to compute the frequent itemsets by means of constructing a transaction list for each frequent itemset. Initially, the database is partitioned into n overlapping partitions, such that each partition fits into main memory. By scanning the database once, all locally frequent itemsets are found in each partition, i.e., itemsets that are large in that partition. Before the second scan, all locally frequent itemsets are combined to form a global candidate set. In the second scan of the database, each global candidate itemset is counted in each partition and the global support (support in the whole database) of each candidate is computed. Those that are found to be frequent are inserted into the set of frequent itemsets [HUSS02].

The correctness of the Partition algorithm is based on the following fact: “A large itemset must be large in at least one of the partitions”. Two scans over the database are sufficient in Partition. This is due to the creation of tidlist structures while determining frequent 1-itemsets. A tidlist for an item X is an array of transaction identifiers in which the item is present. For each item, a tidlist is constructed in the first iteration of the algorithm, and the support of an itemset is simply the length of its tidlist. The support of longer itemsets

is computed by intersecting the tidlist s of the items contained in the itemset. Moreover, the support of a candidate k-itemset can be obtained by intersecting the tidlists of the large (k-1)-itemsets that were used to generate that candidate itemset. Since the transactions are assumed to be sorted, and the database is scanned sequentially, the intersection operation may be performed efficiently by a sort-merge join algorithm [HUSS02].

For higher minimum supports, Apriori performs better than Partition because of the extra cost of creating tidlists. On the other hand, when the minimum support is set to low values and the number of candidate and frequent itemsets tend to be huge, Partition performs much better than Apriori. This is due to the techniques in counting the support of itemsets and fewer numbers of scans over the database. One final remark is that the performance of the Partition algorithm strongly depends on the size of partitions, and the distribution of transactions in each partition. If the set of global candidate itemsets tends to be very huge, the performance may degrade [HUSS02].

In [OGIH97a] an algorithm was produced, which makes only one pass over the database. It uses one of the itemset clustering schemes to generate potential maximal frequent itemsets (maximal candidates). Each cluster induces a sub-structure and this structure is traversed bottom-up or hybrid top-down/bottom-up to generate all frequent itemsets and all maximal frequent itemsets, respectively. Clusters are processed one by one; the tidlist structure in Partition is employed in this algorithm. It has low memory utilization since only frequent k-itemsets in the processed cluster must be kept in main memory at that time [HUSS02].

Chapter Three

Design of Proposed Technique

3.1 Introduction

Association rule mining, as has been defined in chapter two, is “finding interesting association relationships among a large set of data items”. All association rule mining algorithms, with all their variances, consist of two main phases: first finding frequent itemsets according to the given minsup value, and second, extracting rules from these frequent itemsets. As it is usual, the first phase of every association rule algorithm is time consuming and requires more processes due to multiple database scans which cause I/O bottleneck and finding subsets of every itemset is a time consuming task. All known algorithms use some sort of pruning according to the minsup at the first phase which requires re-run the program for every change in database and/or minsup value.

This chapter introduces algorithms that separates running time of the two phases, by adding an inter level consists of a database that contain frequencies of itemsets (support count). Its goal is to fetch data from multiple databases to a main server and store it in a suitable format for generating rules. The first phase can be run at dead time (after hour time for example) which accumulates new itemset frequencies on the old ones and run Rule-finder programs to extract rules at work time.

By saving itemset frequencies in mass storage hard discs (logically in databases) require new algorithms or twisting existing ones to reduce I/O time between server and its hard discs otherwise we will face a problem of delaying rule generation at the second phase. By removing pruning at the first phase and saving all itemset frequencies there are opportunities to find targeted items relation or finding weak relation between items which is impossible by existence of pruning for non-frequent.

Because of the algorithm is targeted to the multiple databases, there may be inconsistent coding for data in different databases. To overcome this problem we introduce a reference table that codes each item or gives multiple items the same code which simplifies inconsistency problem and also it helps to provide an abstraction level of items.

3.2 The Proposed System

The proposed technique is an association rule mining system that aimed to work on the multiple databases on a network. It has been proposed to be mainly consist of three parts named:

a. Data-collector

b. Frequency-base and ReferenceTable

c. Rule-finder

The Data-collector is responsible for fetching and collecting data from multiple databases on the network and substitute replace items with ReferenceTable codes and then finding power set for every item fetched and send their frequencies to a Frequency-base.

The second part of the system is a combination of Frequency-base and ReferenceTable. Frequency-base consists of tables for each k itemset. Each table contains coded itemsets and their frequencies. The ReferenceTable provides codes and meanings for each item.

The Rule-finder is mainly responsible for finding association rules according to the users' demand of specific minimum support and minimum confidence values. Figure 3.1 shows the proposed approach architecture. The algorithms for these phases and sub-steps are detailed in the following sections.

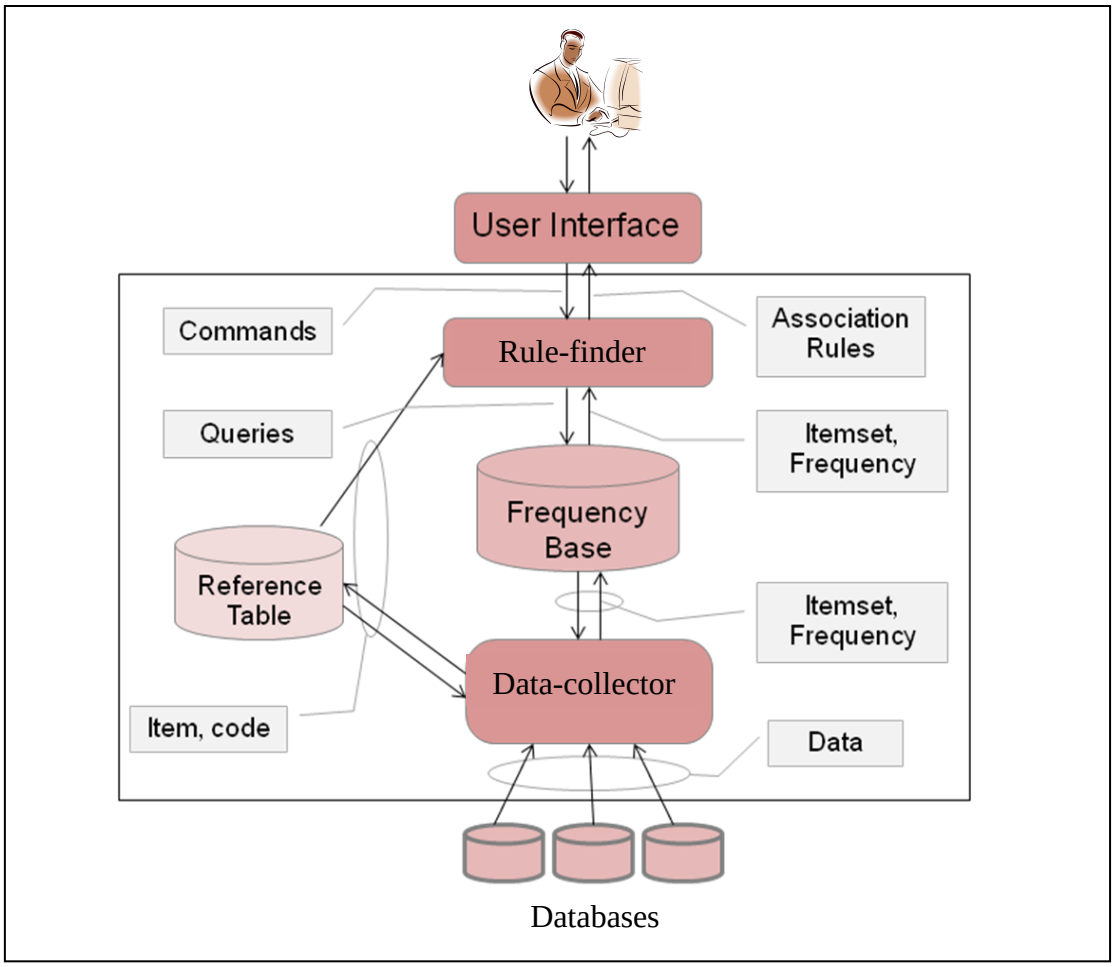


Figure 3.1 The proposed technique architecture

3.3 Target Database

Almost all algorithms of data mining specially association rules needs some sort of preprocessing on the dataset that to be mined. The final layout is important because it represents the input to the mining algorithm and every algorithm expects specific layout. There are two possible layouts of the target dataset for association mining: horizontal and vertical layouts. Each layout has its advantage and disadvantages [MANE99, HUSS02b].

The Proposed Technique algorithms do not use preprocessing as a separate step and do not use intermediate files or databases between transactional database and finding support count process to store dataset. It simply converts each transaction immediately before using it, this is possible because Proposed Approach just scans database only once.

3.3.1 Horizontal layout

Horizontal layout is the standard format used by many researchers [OGIH98]. Here dataset consists of a list of transactions. Each transaction has a Transaction IDentifier TID followed by a list of items in that transaction. There are two types of horizontal layouts: Binary-horizontal layout and itemized-horizontal layout, see figure 3.2.

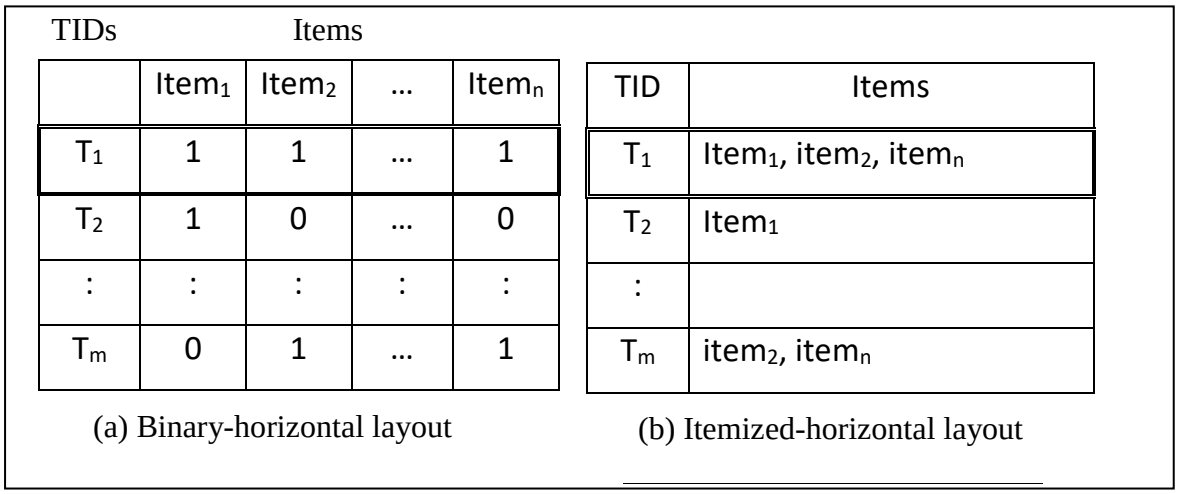


Figure 3.2 Horizontal layouts

Binary-horizontal layout is rarely used when the database contains a large number of items. In this case the database shapes a sparse matrix.

The Proposed association mining approach algorithm uses itemized-horizontal layout virtually by converting every transaction at scanning time to a sequence of coded items. This process by passes the need for other secondary programs to convert transactional database to a suitable format. That means the new approach reads from transactional database directly. This is especially important to fully automating process of support count for items at out-work time by servers.

3.3.2 Vertical layout

In the vertical layout (also called the decomposed strong structure [HIPP01]). The dataset consists of a list of items. Each item is followed by its tid-list. Vertical layout is not used as final representation for dataset before rule mining, but it is used as internal representation in Apriori-TID algorithm [RAKE96A].

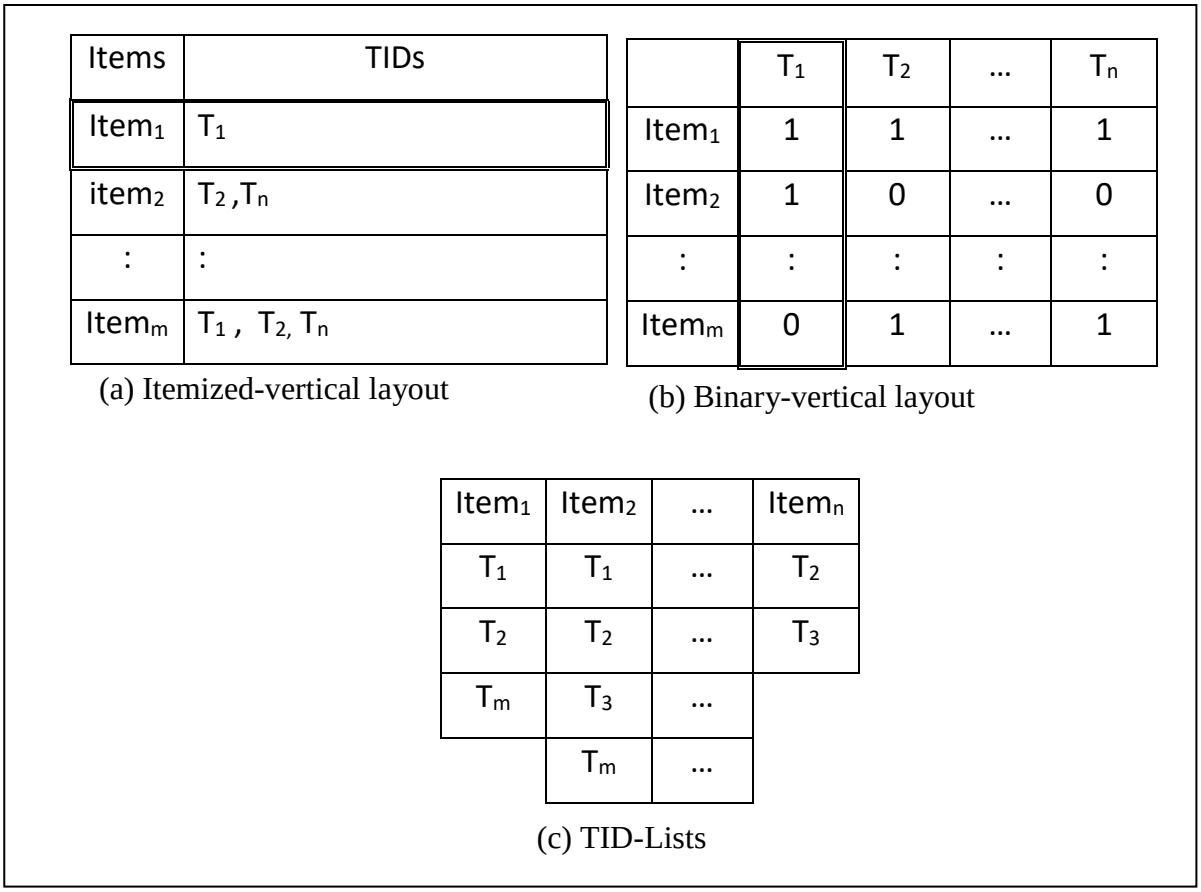


Figure 3.3 Vertical layouts

3.4 Finding Frequencies for Itemsets

The number of occurrences of an itemset in the transactional database called itemset frequency (or itemset support count). As mentioned in the previous chapter almost all algorithms uses some pruning on the itemsets to find frequent itemset according to a user provided value called minsup. Pruning is essential to minimize the cost of the main memory requirements for large databases and also to find interesting rules. In this

approach the algorithm tries to not prune itemsets instead, it stores itemsets and their frequencies in databases this eliminates the need of a huge main memory. The pruning is replaced by a step in the Rule-finder. The Rule-finder uses filtering just before rule generating for existing frequencies by minsup value.

3.5 Designing Data-collector

Data-collector is the part that is executed automatically over a network to fetch itemsets in a multiple databases and store there frequencies in Frequency-base. To simplify Data-collector design and operation we divide it into three layers. Each layer will prepare data to the next one. The layers are Scanner, ItemsetGenerator and the Frequency-base Creator.

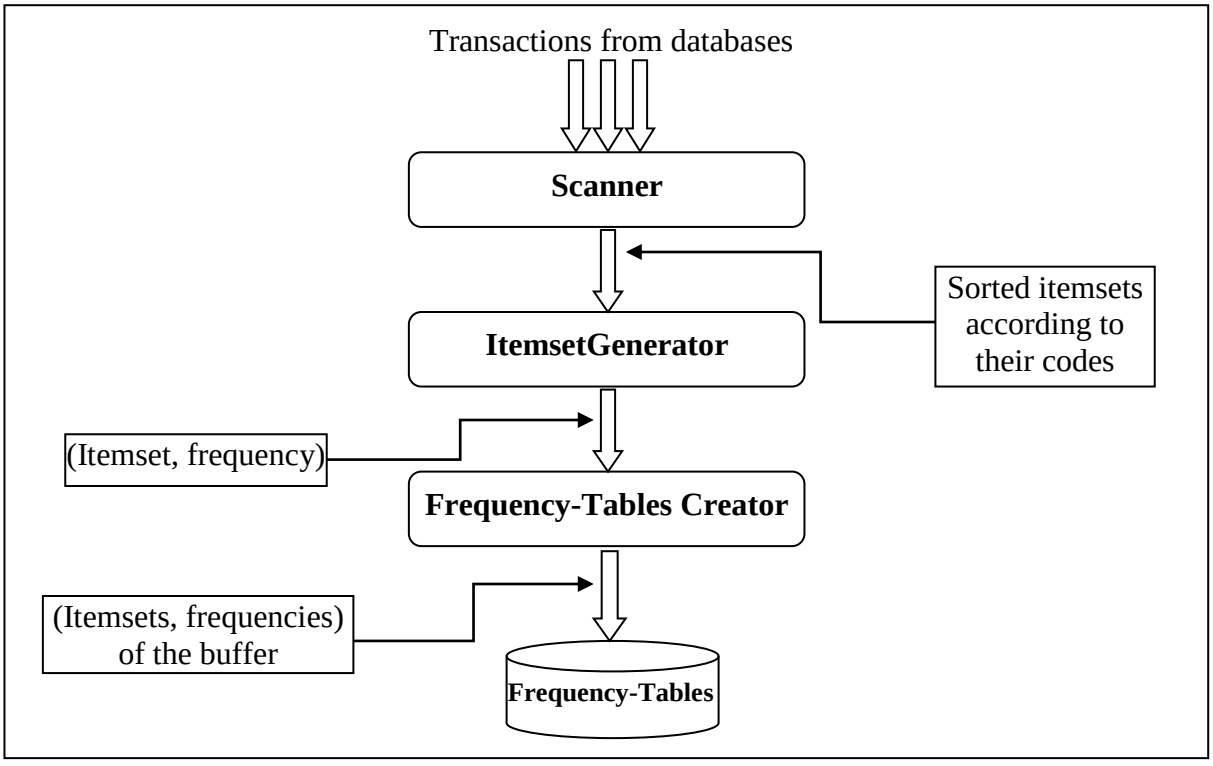


Figure 3.4 The Data-collector architecture

3.5.1 Scanner

It is the first layer of the Data-collector. It is responsible for fetching data from databases and replacing items with codes, and then sending them to ItemsetGenerator as sorted transaction according to the ReferenceTable codes. This layer has information about all databases that to be connected with and how to connect with also it keeps track for last scanned record in each server.

This layer makes an abstraction level for connecting to the DBMS servers and hides details of how to establish a connection with a specific DBMS and uses the right DBMS driver for that connection and also uses the right DBMS dependent SQL structure, this layer's featuer makes it possible to mine data for association rules from varios types of DBMS servers till they use SQL language and has transactional table. Actually it is possible to have mulitple types of DBMSs at the same time.

As mentioned before this layer keeps track of last scanned record in each server this is especially important for the next start scan to know where to begin and not rescan from scratch, which makes the proposed approach sensitive to insert operations.

Because this layer fetches data from taransactional database directly it should make some preprocesing on the feched data for preparing to be mined, hence filtering for data can be made according to some criteria just like transaction length or transaction validation.

```
foreach transactional  Database D in the list do{
        connect to D start scanning from last stop point
        foreach record R in the D do{
                if TID in R == previous R's TID then{
                        Encode current R's item value from ReferenceTable and put it
                        in its right place in the transaction list.
                }
                else {
                        copy transaction list to the circular buffer.
                        clear transaction list.
                        Encode R's item value from ReferenceTable and put it in the
                        transaction list.
                }
        }
}
```

Algorithm 3.1 The Scanner algorithm

3.5.2 ItemsetGenerator

The second layer of the Data-collector handles heavy processes of finding power sets of all itemsets which require $O(2^n)$ of the processing time [MICH01]. It gets its data (itemset) from common circular buffer between Scanner and ItemsetGenerator. The first involved pushs data to the buffer and the later pops data from it. The circular buffer is involved to manage the main memory and also to organaze data exchange between layers.

ItemsetGenarator finds power set *P* by using isomorphism with the binary numbers from 1* to 2^{n-1} where n is the number of elements in the set, the binary digit 0 implies absence of its relative element and 1 represents its existence.

Example 3.1:

Consider *P* ({x, y, z}) as shown in Table (3.1):

Table 3.1 Power set of {x, y, z}

Integer	Binary	Subset
1	001	{z}
2	010	{y}
3	011	{y, z}
4	100	{x}
5	101	{x, z}
6	111	{x, y, z}

ItemsetGenerator generates all k-itemsets (k is an integer starting from 1) for a transaction with length k, and then it stores each k length itemsets in separate tree map with their frequency. Then, it increments this frequency for every occurrence of the same itemset until tree map reaches its size limit, then flushes it in to a circular buffer, which is common between ItemsetGenerator and Frequency-base Creator, using tree map as a temporary storage area speeds up the process because first it utilizes RAM which is more faster than directly saving in the Hard disk and tree map guarantees guaranteed log(n) time cost for updating itemsets' frequencies (support count). The ItemsetGenerator is shown in algorithm 3.2

* it is actually starts from 0 for finding full power set but in the association mining the empty set is not considered.

```
temp[] = array of tree maps with length MAX_TRANSACTION_LENGTH
while there is transaction T in the circular buffer do {
        len = T^2 – 1   //number of non-empty subsets
        for i = 1 to len {
                sub = getSubset(T, i)
                k = length of sub
                if sub exists in temp[k] then
                        increment SUB's frequency by one
                else
                        put (sub, 1) in temp[k]
                if temp[k] reaches maximum limit then flush to store it
        }
}
```

Algorithm 3.2 ItemsetGenerator algorithm

```
getSubset (Transaction T, integer i){
        sub = ""
        size = size of T
        binary[size] = array represents binary number
        for i = 0 to size
                if binary[i] == 1then sub = sub + T[i]
        return sub
}
```

Algorithm 3.3 getSubset (binary isomorphism algorithm)

3.5.3 Frequency-base Creator

This is the last layer of the Data-collector. It is responsible for the main functionality of Data-collector, which is constructing Frequency-base, to be analyzed later in rule finding process. This layer is also requires a significant amount of time because of its intensive use of updating operation on Frequency-base, hence careful algorithms are needed to avoid too many UPDATE operations on the database. This is because every itemset which is exists in the table need to be updated individually, so the best solution is to replace all UPDATEs with a single SELECT-DELETE-INSERT operation consequently.

This layer gets its data from circular buffer common between this layer and ItemsetGenerator, then it updates these groups' frequency by summing with old itemsets'

frequencies from Frequency-base, then it deletes old itemsets of this group (actually TreeMap) of data in the Frequency-base before inserting new ones. These operations are rearranged to build the Frequency-base Creator algorithm shown in Algorithm 3.4

```
foreach group g of k itemsets in the circular buffer{
        oldG = SELECT * FROM frequencybase.itemset_K WHILE itemset IN g
        foreach oldItem of oldG{
                get newItem == oldItem in g
               newItem.frequency += oldItem.frequency
        }
        DELETE FROM frequencybase.itemset_K WHERE itemset IN g
        INSERT INTO frequencybase.itemset_K VALUES (g)
}
```

Algorithm 3.4 Save layer algorithm

3.6 The Reference Table

Reference Table is a table with two attributes, one for the item names and the other for item codes. It is the Frequency-base' interface to the other parts of the proposed system (the Data-collector, and the Rule-finder). It provides codes of items for Data-collector and also interprets the codes inside Frequency-base for the Rule-finder.

Reference table's functionality of providing both codes and names of items in as short as possible period of time needs to be sorted for both fields this means that there is actually two tables, first ordered by codes (provides codes for a given item name) and second ordered by names (provides all associated item names for a given code). For providing an abstraction layer it is possible to have multiple item names associated with the same code. The use reference table is:

- Reduce the required storage
- Make sorting easier
- Provides an abstraction layer, and
- Works as a consistent item codes between deferent databases

The construction of reference table can defer according to the nature of the databases intended to be mind. It can be constructed:

- Manually by typing each item or providing a list of items
- From classification or clustering mining tasks
- Do not provide any item names and construct it on the fly when Data-collector finds new item it just gives it new code.

Table sorted by item name

Name	Code
Ahmed tea	3
Al-Utor tea	3
Bread	2
Coca cola	1
Mahmud tea	3
Pepsi cola	1

Table sorted by item code

Code	Name
1	Coca Cola , Pepsi Cola
2	Bread
3	Ahmed Tea , Al-Utor Tea, Mahmud Tea

Figure 3.5 Sample of reference table's instant

3.7 The Frequency-base

The Frequency-base is a database contains of k tables where k is the maximum itemset length. Each table is dedicated to specific length of itemset and contains two attributes; first the itemsets and second the itemsets' frequency occurrence in the database(s).

Itemsets are coded from reference table and sorted according the codes, the elements (items) of each itemset is separated by comma. Sorting of itemset elements is important to make sure that there is no duplication.

Using multiple tables to store (itemset, frequency) pairs are important for achieving speed because each k^{th} table just contain k length itemset hence shorter time to access each record. A demonstration example is shown in figure 3.6.

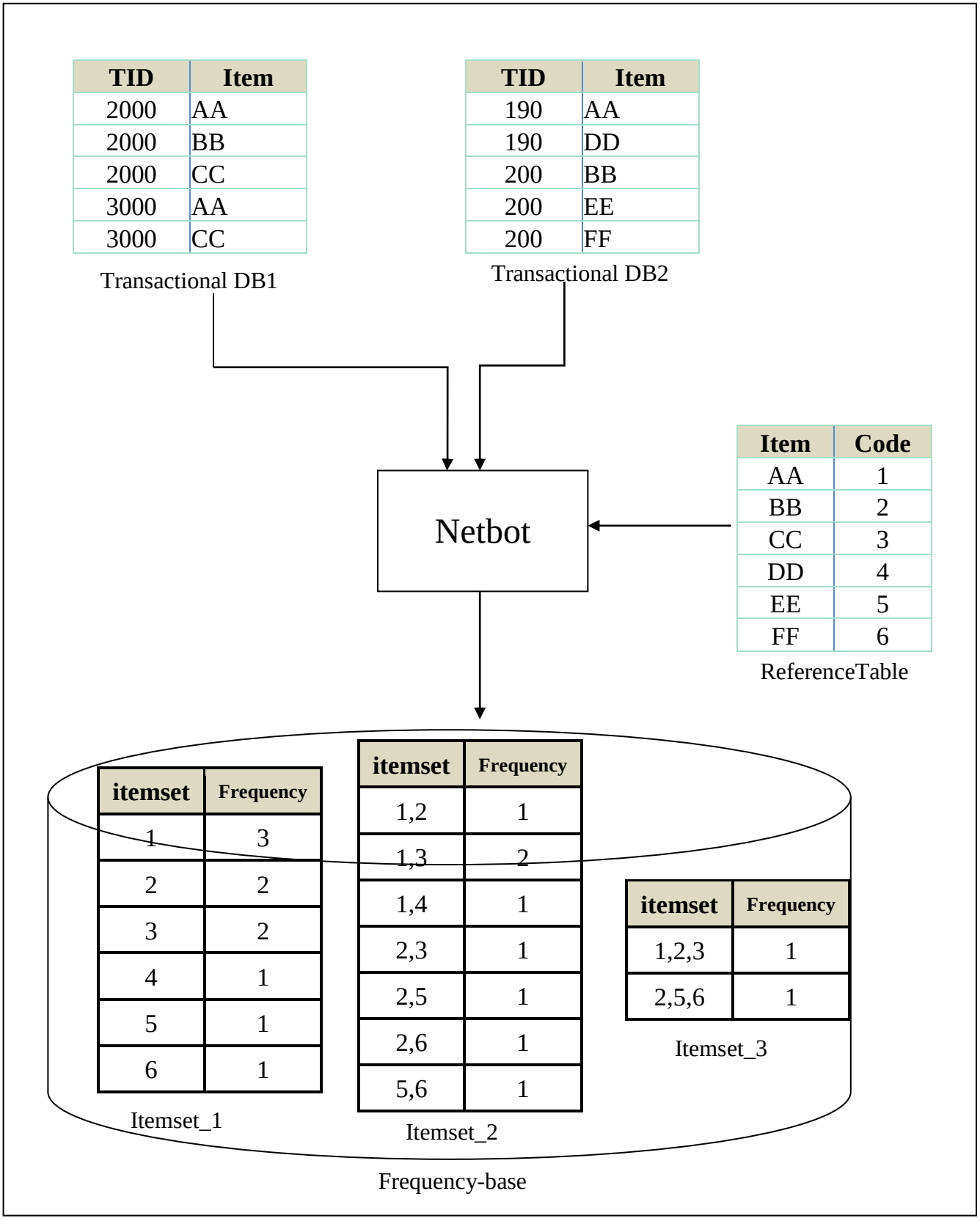


Figure 3.6 A demo Example of frequencybase construction

3.8 The Rule-finder

Rule-finder is the Third part of the proposed system. The main task of this part is to implement the queries of the users who want to find association rule. The queries may be in three forms: first querying about association rules among specified itemset to know their support count and confidence of each generated rule for this itemset, second querying for any k-itemset with the given minsup and minconf and the last one is finding strong association rules.

The proposed Rule-finder is divided into two parts: first part fetches itemset/ frequency pairs in the Frequency-base according to the minimum support the second part generates association rule for itemsets who pass minimum confidence. The second part will be time consuming because the frequencies of the items are stored in the database not in a data structure in the main memory which causes extensive I/O if there is no careful design that is suitable with database nature.

Finally before producing the output to the users, rules should be decoded to the original item name or group of items. As mentioned before ReferenceTable has capability to encode and decode items according to their name or group of items to achieve abstraction, speed and compatibility between deferent item coding for deferent databases (if differences exist).

The output could be directly sent to the screen; especially when there is a small number of rules otherwise it can be saved in a file. Figure 3.7 shows the Rule-finder's architecture with its internal dataflow. The layers are described in the following sections.

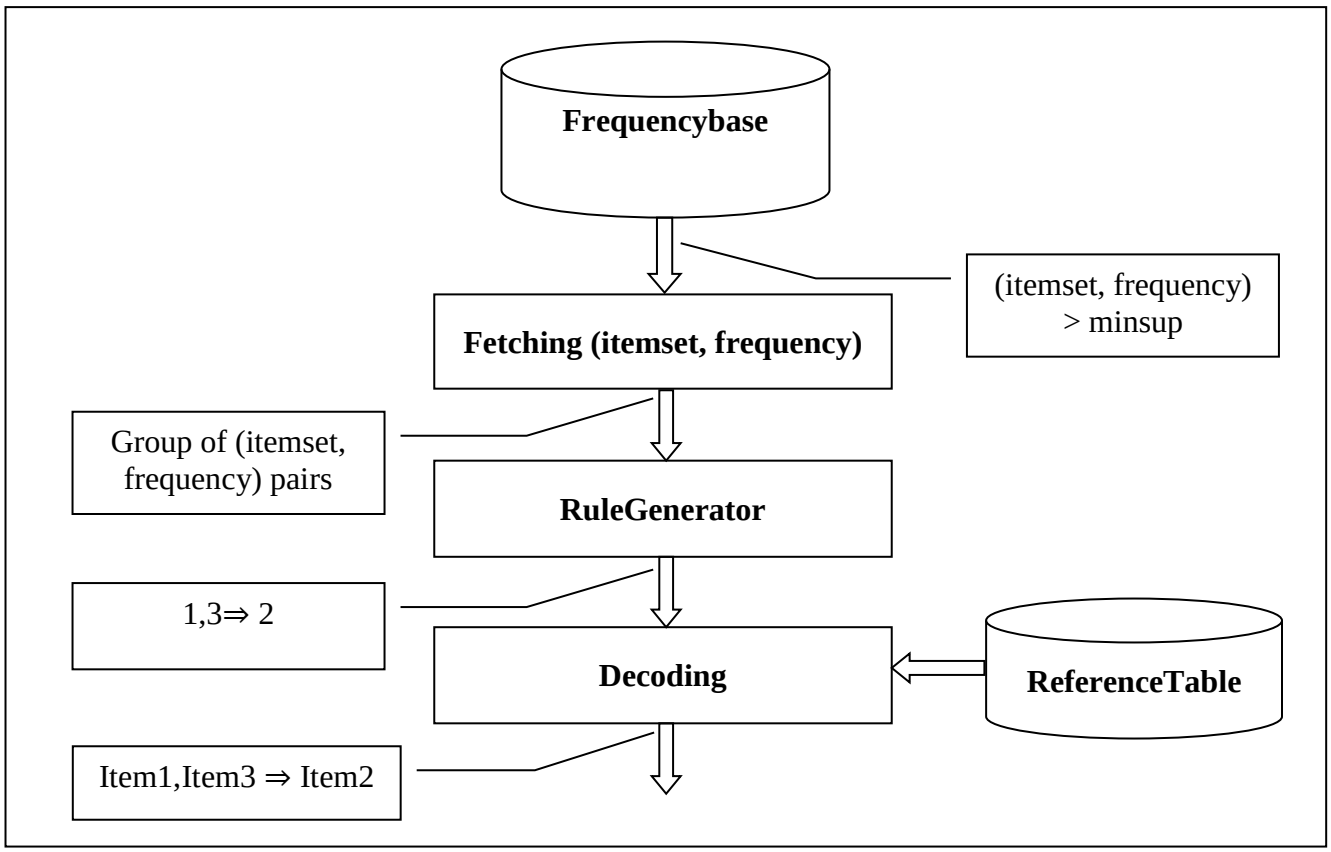


Figure 3.7 Rule-finder Architecture

3.8.1 Fetch Layer

This layer is responsible for fetching data in groups according to the user's command and organizes it in a special data structure that maps every subset to its itemset (or itemsets in the RuleGenerator). Subsets are sorted to be suitable for implementing binary search to speed up the process. Figure 3.8 shows a demo example of this data structure creations.

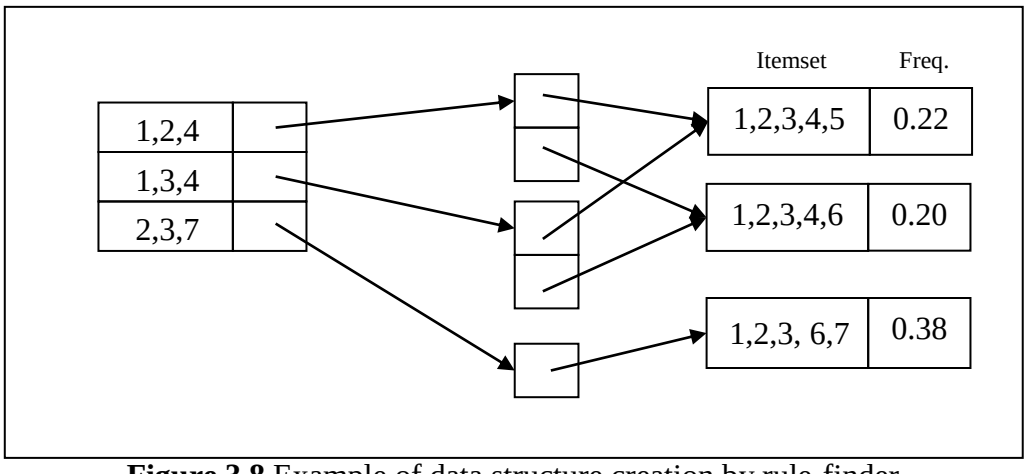


Figure 3.8 Example of data structure creation by rule-finder

This layer fetches data from Frequency-base in three different ways convenient with the manners of the user commands for association rules findings.

3.8.1.1 Finding Association Rules for Selective Group

In this method, users provide a group of items that they have interests in to find relationship between them without providing any information about minimum support or minimum confidence. This method is possible because of the nature of Data-collector that doesn't prune itemsets according to a given threshold. The strong association rule mining is shown in algorithm 3.5

```
SelectiveGroup(String[] items){
        String ii
        Foreach I in items
                ii = ii + RefrenceTable.getCode (I)
        k = length(items)
        fre =  frequency of ii in table k in the frequencybase
        pass (ii,fre) to ruleGenerator  with minsup =0, minconf = 0
}
```

Algorithm 3.5 Finding Strong association rules algorithm

3.8.1.2 Finding Association Rules in k Itemsets

In this method users will provide the program with the length (or k) of itemsets that will be interested to find association rules, minsup and minconf. This layer fetches itemsets and their frequencies, with k elements and frequency >= minsup, in the Frequency-base then sends them in groups of limited size to the next layer for generating association rules.

The idea of this method is to enable users to select the length of the itemsets that they want to produce association rule for them with a specified minsup and minconf. Algorithm 3.6 shows the pseudo code for this method.

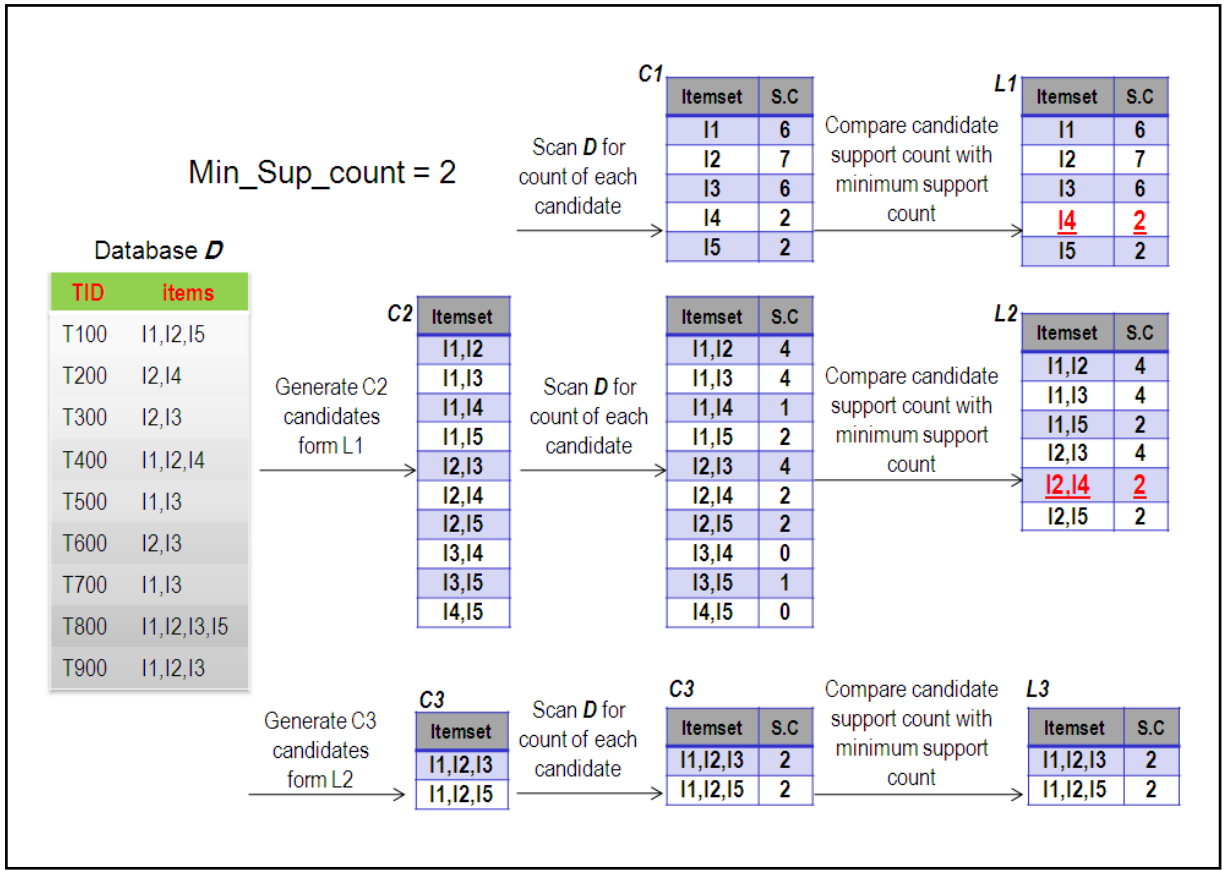


Figure 3.9 An example of Apriori strong itemset detection

3.8.1.3 Simulating Apriori Algorithm Results

In this way users just provide this layer with minsup and minconf and this method will finds the largest k table in the Frequency-base with itemsets' frequencies > minsup. This method uses previous method for passing data to the next layer after it finds suitable value for k. The pseudo code is shown in Algorithm 3.7

```
KitemsetAssociationRule(int k, float minsup, float minConf){
        QueryResult rs
        LinkedList link
        TreeMap buffer
        int elementNumer =0
        rs = executeQuery ("Select subset,frequency from itemset_" +k+"  Where frequency
                >= "+ minsup);
        while(rs.next()){
                subset = rs.getString("subset");
                freq = rs.getLong("frequency");
                link = new LinkedList();
                link.add(new ItemSet(subset, freq));
                buffer.put(subset, link);
                if(elementNumer == BUFFER_SIZE){
                        elementNumer = 0;
                        ruleGenerator (buffer,k,minConf);
                        buffer = new TreeMap();
                }
                elementNumer++;
        }
                if(elementNumer !=0)
                        ruleGenerator(buffer,k,minConf)
}
```

Algorithm 3.6 K-Itemset Association Rule Algorithm

```
simulateApriory(double minSup,double minConf){
        for(int k = 2; k <= NetNavigator.MAX_TRANSACTION_ITEMS; k++)
                if(itemsetCount(k,minSup) > 0)
                        KitemsetAssociationRule (k,minSup, minConf)
                else
                        break;

}
int itemsetCount(int k, float minSup){
        int count = executeQuery("Select count(frequency) as itemCount from itemset_" + k
                        +"  Where frequency >= "+minCount)
        return count
}
```

Algorithm 3.7 Apriori Simulation Algorithm

3.8.2 RuleGenerator Layer

The RuleGenerator generates association rules, which is the main goal of the current proposed system for itemsets coming from previous layer. This layer should use algorithms that reduce I/O between RuleGenerator and Frequency-base for this reason we choose to use a modified version of the Apriori's simple algorithm [RAKE94] for discovering rules instead of using fast algorithm because these algorithms are made to work with (itemset, frequency) in the main memory not in a database and they are tested for speed according to the number calculations and choices they made not on the number of reaching itemsets' frequency (support count), as it is obvious the last parameter is more important for the system because reaching itemsets' frequency means more I/O for Frequency-base which may wastes more time than calculation. Figure 3.8 show that there is just one time that accesses frequency for subset of the itemset to find rules confidence while in the fast algorithm as shown in algorithm 3.9 there is more than one times that accesses frequency for subsets.

The modification made on the Apriori's simple algorithm for discovering association rules is to make it possible to work with a group of itemsets simultaneously instead of individual itemset at a time. This modification will reduce I/O time by fetching groups of items and their subsets instead of using different query for each subset of itemsets. Algorithm 3.10 shows RuleGenerator's algorithm used in the proposed System.

```
forall large itemsets Ik, k ≥ 2 do
        genrules(Ik, Ik);
//The genrules generates all valid rules ă ⇒ (Ik – ă), for all ă ⊂ am
procedure genrules(Ik: large k-itemset, am: large k-itemset)
        A = {(m - 1)-itemsets am-1 | am-1 ⊂am};
        forall am-1∈A do begin
        conf = support(Ik)/support(am-1)
        if (conf ≥ minconf) thenbegin
                output the rule am-1 ⇒ (Ik - am-1), which confidence = conf and support = support(Ik)
                if(m – 1 > 1)then
                        call genrules(Ik , am-1) ;
        end
end
```

Algorithm 3.8 Apriori's Simple Algorithm for rule detection

```
forall large k-itemsets Ik, k ≥ 2 do begin
        H1 = {consequents of rules divided from Ik with one item in the consequent};
        call ap-genrules(Ik, H1);
end

procedure ap-genrules(Ik: large k-itemset, Hm: set of m-item consequents)
    if (k > m +1 ) then begin
        Hm+1 = apriori-gen(Hm);
        forall hm+1 ∈ Hm+1 do begin
            conf = support(Ik)/support(am-1)
            if (conf ≥ minconf) then
              output the rule am-1 ⇒(Ik-am-1), which confidence =conf and support= support(Ik);
            else
              delete hm+1 from Hm+1;
        end
        call ap-genrules (Ik , Hm+1) ;
  end
//apriori-gen algorithm
Insert into Ck
Select p.item1, p.item2,…, p.itemkk-1, q.item k-1
From Lk-1  p, Lk-1  q
Where p.item1 = q.item1, …, p.itemk-2 = q.item k-2, p.itemk-1 < q.item k-1
Next, in the prune step, we delete all itemsets c ∈ Ck such that some (k - 1)-subset of c is not in Lk-1

forall itemsets c ∈ Ck do
  forall (k - 1)-subsets s of c do
        if(s ∉ Lk-1) then
            delete c from Ck;
```

Algorithm 3.9 Apriori's Fast Algorithm for rule detection

```
String strItemsets;
void generateRule(TreeMap itemSet, float minConf, int k){
        if(k == 0)
                return;

        subsets = {(m - 1)-itemsets am-1 | am-1 ⊂am, am ∈ itemSet };
        LinkedList link = null;

        QueryResult rs = executeQuery("Select subset,frequency from itemset_" + currentK + "
                        Where subset IN ("+strItemsets+");");
        while(rs.next()){
            subset =rs.get ("subset");
            frequency = rs.get ("frequency");
            superSets = List of superset(s) of subset
            foreach super ∈superSets {
                conf = getSupport (super)/frequency;
                if(conf >= minConf)
                    output the rule am-1⇒(Ik-am-1), which confidence =conf and support= support(Ik);
                else{
                remove super from  superSets;
                if(count(superSets) == 0)
                        remove subset from subsets;
                }
            }
        }

        If(subset(subsets)>0)
                generateRule(subsets,minConf, k - 1);
        }
```

Algorithm 3.10 RuleGenerator Algorithm

Chapter Four

Implementation, Results and Discussion

4

4.1 Introduction

As explained in the previous chapter, the proposed technique consists of two main parts, The Data-collector and the Rule-finder. Each part runs separately during suitable times, to overcome the time consuming, database scans and itemsets frequency generations. In this chapter, the proposed algorithms, which have been developed in chapter three is coded, implemented and tested on real data.

Java has been used to implement the proposed system as a programming language and MySql as a Relational Database Management System "RDBMS". The advantages of the java programming language and MySql RDBMS over their peers are:

- Java has geed properties for network programming which is suitable for the proposed system to connect multiple database clients to the Proposed System server.
- Usually java is slower than other programming languages like C# for example in Graphical User Interface GUI but it is faster in operation calculations (this is really matter in data mining).
- Java is an OOP, which allows splitting the algorithm into many classes during coding.
- MySql is fast, powerful and simple RDBMS, it works with variety of database engines for example innoDB, MEMORY, MyISAM and ISAM which provide flexibility and availability.
- MySql is widely for web development. The number of Enterprises that use MySql to handle their data are increasing day by day.
- A combination of Java/MySql is convenient, because both are open source and compatible.

4.2 The Databases

The proposed System requires two types of databases to run. The Frequency-base database which contains k tables of k-itemset generated by Data-collector and the client database(s) which contains transaction information.

MySql uses standard Structured Query Language SQL (without special instruction) for handling Queries.

4.2.1 Implementing Frequency-base

Frequency-base is a database stores frequency of every itemset and their subsets (another smaller itemset). It stores different length itemsets in different tables which leads to k tables for k lengths itemsets. The Frequency-base created automatically by Data-collector. Each table has the schema: (Subles(Varchar)**,** Frequency(Numeric)). The SQL code for creating the frequency-base is given in Program 4.1.

The Frequency-base is created by the Data-collector and used by the Rule-finder. This is required for introducing a matching block between the Data-collector and the Rule-finder. This is handled by introducing a Reference Table.

```
--Create schema of the Frequency-base
CREAT DATABASE IF NOT EXISTS frequencybase;
USE frequencybase;

--Definition of the Frequency-base tables ‘itemset_k’
--where k is the length of itemset stored in this table (it should replaced with numbers).
DROP TABLE IF EXISTS ‘itemset_k’;
CREATE TABLE ‘itemset_k’ (‘subset’ varchar (80) NOT NULL,‘frequency’ int(10) unsigned
        NOT NULL,
PRIMARY KEY (‘subset’)
)ENGINE =InnoDB DEFAULT CHARSET=latin1;
```

Program 4.1 SQL statements for creating Frequency-base and its tables

4.2.2 Implementing Client Databases

The client databases should contain a table that stores transactions. These databases can be hosted in a single host, separate host or on the same host with the Frequency-base. The schema of transactional table is (ID(Numeric),Item(Varchar)).The SQL codes for creating the client databases is shown in Program 4.2.

```
--Create schema of the client database
CREAT DATABASE IF NOT EXISTS supermarket;
USE supermarket;

--Create transactional table in the database.
DROP TABLE IF EXISTS ‘transactions’;
CREATE TABLE ‘transactions’ (
‘TID’ int(10) unsigned NOT NULL,
 ‘Items’ varchar (50) NOT NULL,
PRIMARY KEY (‘TID’, ‘Items’)
)ENGINE =InnoDB DEFAULT CHARSET=latin1;
```

Program 4.2 SQL structure for creating client database and transactional table

4.3 The Proposed Technique Implementation

The implementation strategy for the proposed system is to split each part of the system (The Data-collector and the Rule-finder) into a number of layers. Because java is a full Object Oriented Programming OOP language, each layer may be implemented as an object of class.

In addition to these layers, a number of utility classes are required to ensure the matching and compatibility between layers.

4.3.1 Utility Classes

To Run the System correctly and to make the data communicate between its parts compatible there are needs for some special objects for saving client database information, coding and decoding items, data communicating between layers in a thread safe environment. These classes are:

I. DBServers Class

This class is designed to save information about client databases. The information are: Object host URL, database name, user name, password and last scanned stop for Data-collector. This information is used by the Data-collector to connect with the clients and scanning their transactional databases. The methods (or actions) of this class are inserting new hosts and databases, updating existing hosts and databases, keeping track of last stop of the Data-collector for each client transactional database, saving these information to hard disk, clearing all information and resetting scan stop points for full implementation.

II. CircularBuffer Class

This class is designed to communicate data between layers in a thread safe environment. The protection for thread reaching is kept in an element level not buffer level which provide more efficiency for communication and minimizes waiting time of threads for reaching.

III. ReferenceTable Class

This class provides coding and decoding for information stored inside the Frequency-base. It is actually connects Data-collector, Frequency-base and Rule-finder together, see figure 4.1.

Also, numeric codes is used to keep Frequency-base as small as possible to make sorting for itemsets faster and also to keep Frequency-base consistent if there is multiple clients with deferent coding of items.

The ReferenceTable class is able to register groups of items for the same code number; this is for creating abstraction levels for itemsets. This Class utilizes TreeMap in saving information which guaranties log(n) time to access the elements for more information.

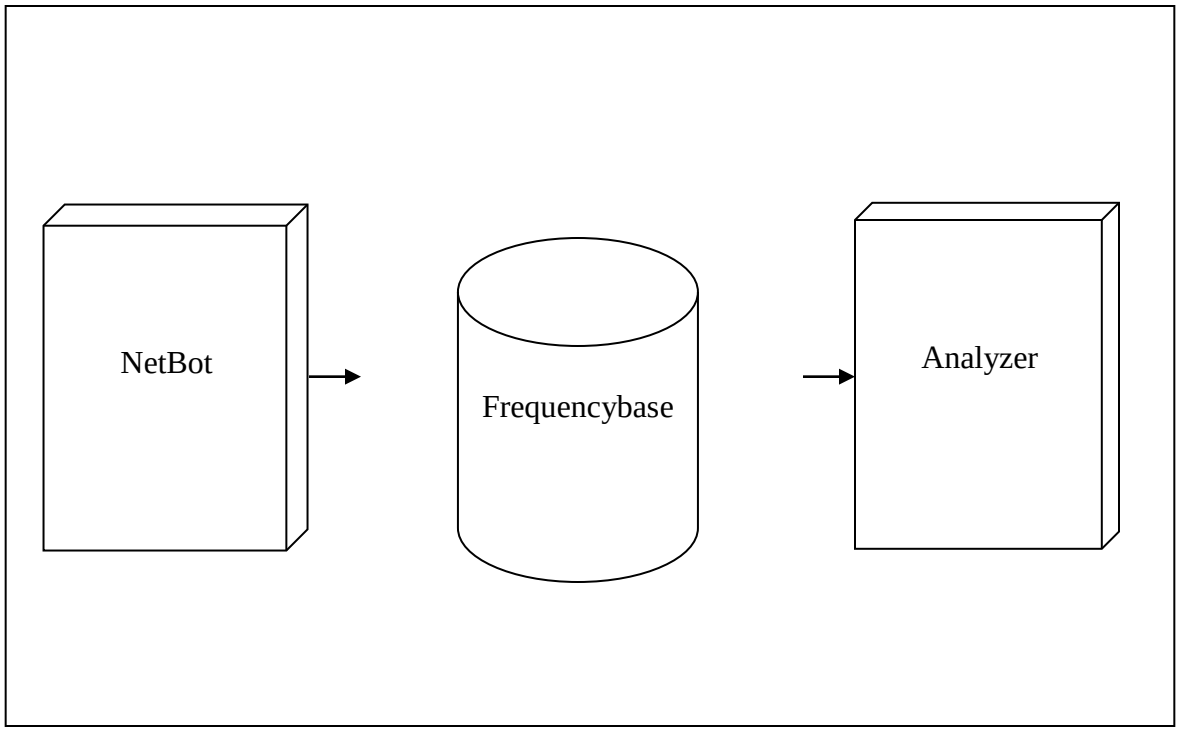


Figure 4.1 Location of ReferenceTable in the proposed system

IV. ItemSet Class

The function of this class is to store an itemset and its frequency; it implements a Comparable to be stored in the TreeMap.

4.3.2 The Data-collector Implementation

The function of the Data-collector, as it is developed in chapter three is to fetch transactional data from client databases, and then generate all subsets of fetched itemsets, then counting frequency of every subset and finally saving them in the Frequency-base Data-collector consists of three layers each layer can be represented as a class in the java environment.

I. Scanner Class

The function of this class is to scan the client databases, fetch transactional data in a block level, create the itemsets, and finally encodes the fetched data according to the ReferenceTable. The scanning process starts at the last stopped for fetching data using the following SQL routine:

SELECT tid, items
FROM tableName
Limit LAST_STOP, FETCH_SIZE**;**

Where: tableName is the name of current databases table, LAST_STOP is the latest stop point for last scan of this database by Scanner and FETCH_SIZE is the block size.

Scanner pushes the itemsets in the circular buffer then proceeds to create next itemset.

II. ItemsetGenerator Class

This class uses the temporary itemsets in the circular buffer; create non-empty subsets, then save the subsets with their frequency of appearance in buffers according to the length of each subset.

The time requirements for executing this class's routines is rising exponentially when the length of the itemset increases, see figure 4.2. When each buffer reaches its maximum

size, it pops its data to another circular buffer between the ItemsetGenerator and the Data-collector class see figure 4.3.

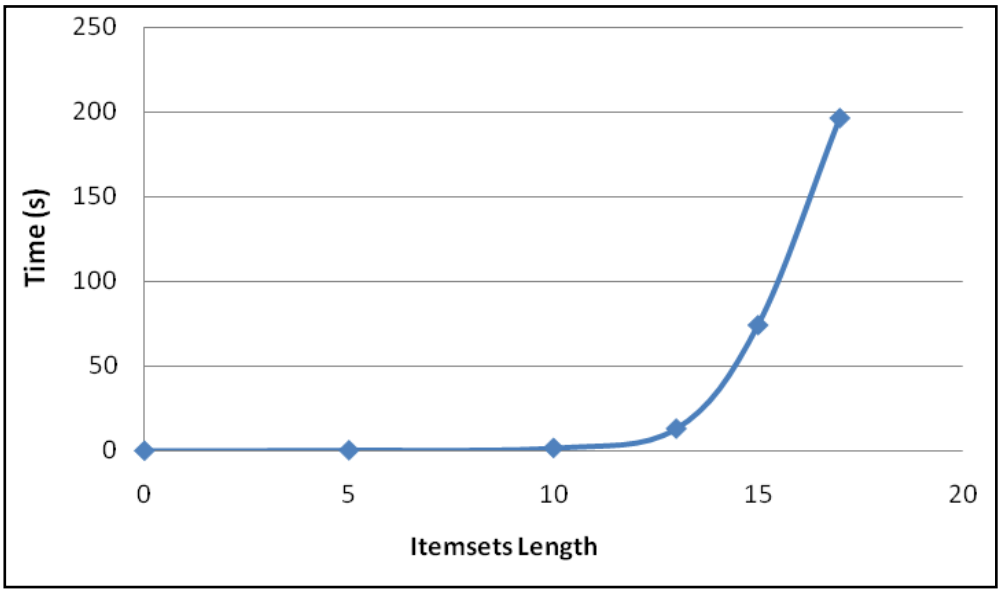


Figure 4.2 Time-Itemset relationship length
for generating all subsets of transactions for 512 records

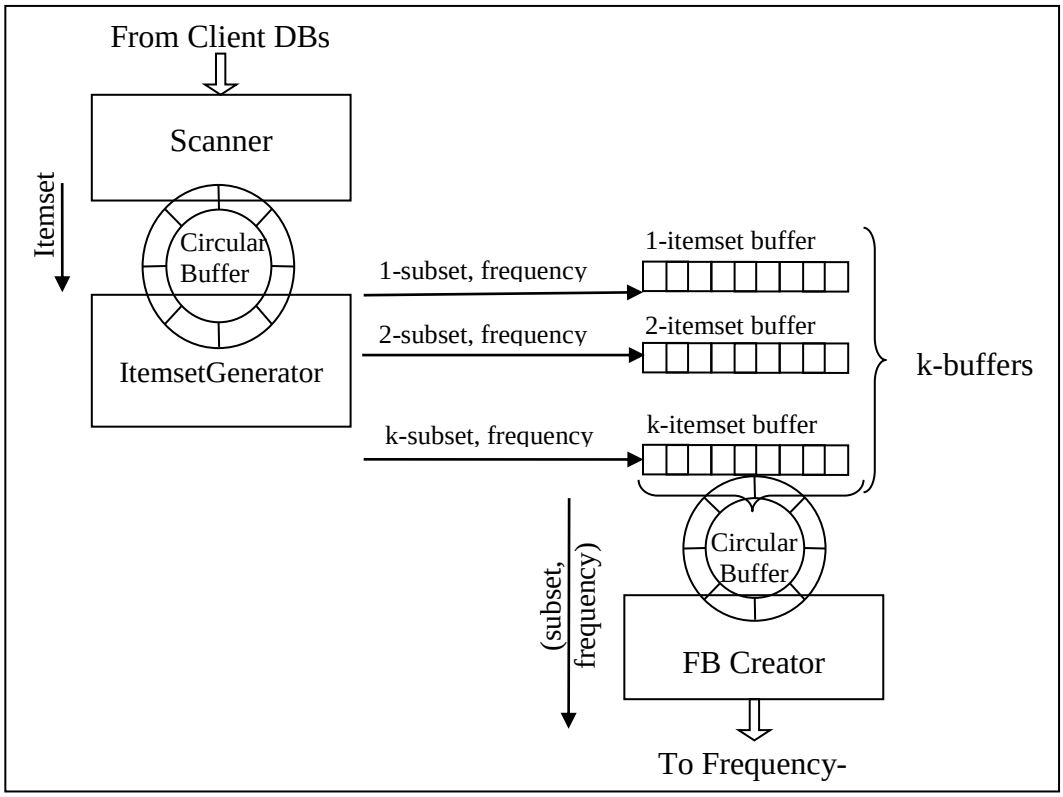


Figure 4.3 data flow between Data-collector layers

III. FBCreator Class

This class is the main class of the Data-collector. It triggers the Data-collector to start, that means, it controls other two lower level layer classes (the Scanner and the ItemsetGenerator) to support this layer in its task to create its data, and then save them in Frequency-base. Data is received to this layer in block level of (k-itemset, frequency) from the circular buffer between the ItemsetGenerator and the Data-collector.

4.3.3 The Rule-finder

Rule-finder is the last part of the proposed System. It is more interactive with the users by interpreting their commands to output reports of association rules. As stated in chapter three, the Rule-finder is divided to two layers: the Rule-finder and the RuleGenerator. Each layer can be implemented as a class in the Java environment.

I. Rule-finder Class

This class takes the order from users. It uses the Frequency-base to produce blocks of up to 256 units and send it to the RuleGenerator. It captures the user's selection of the association rule. The proposed Technique supports three types of rule generation:

1. **Selective Itemsets:** This type of rule generation takes a group of itemsets and outputs support and confidence of each rule can be generated by this group.
2. **Selected length of itemsets:** In this type, the users gives the Rule-finder minsup and minconf for specific k length of itemsets, which means the Rule-finder will search k^{th} table in the Frequency-base for generating rules.
3. **Apriori simulation:** In this type, it is only required from the user is to provide minsup and minconf to generate rules similar to the Apriori algorithm.

II. RuleGenerator Class

This layer uses the output of the Rule-finder class to generate rules. To speed up the process of finding rules and minimizing queries as low as possible, twisted Apriori's Simple rule generator is used. This class's instance outputs its data (rules) to the text file because it's more convenient for reviewing, reporting and printing.

4.4 The Proposed System Verification

The next step to the implementation of the proposed system is its verification. There are many verification approaches in Software Engineering point of view. In this thesis, a comparison method has been used to see the accuracy of the proposed and implemented System. The comparison method is working if there are some benchmarks or published solved problems. We have searched the internet and the existence resources, but we have failed to find a standard benchmark. Hence we are going to use an example in the book of Jiawi and Micheline "Data Mining: Concepts and Techniques" to check the proposed system. This example is used as a benchmark to make the verification for the two parts of the system individually. The selected example is example 6.1[JIAW01]. It applies a market basket analysis for an ALLElectronic supermarket.

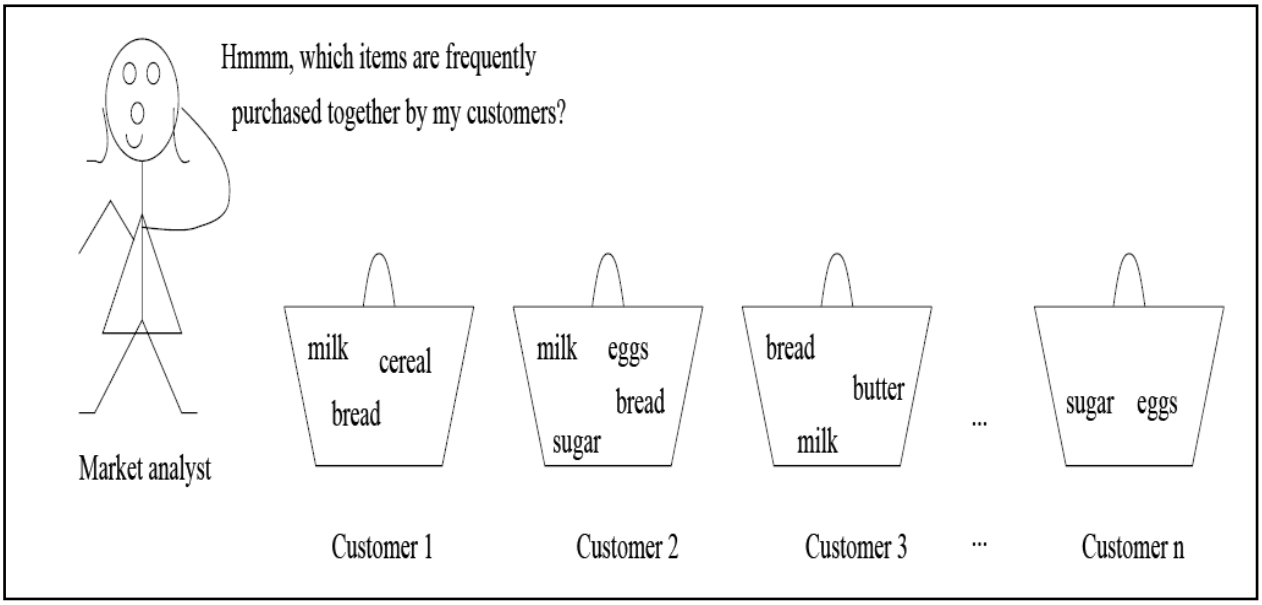
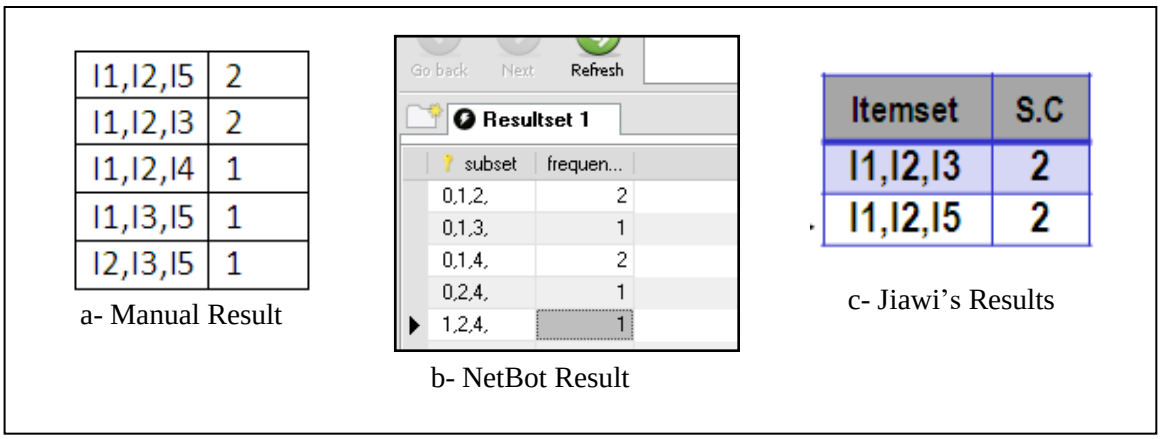


Figure 4.4 Market Basket analyses

To verify the Data-collector, all frequencies of the 3-itemsets have been determined manually, results are listed in figure 4.5.a, while figure 4.5.b shows results of the itemset_3 table of the Frequency-base, and figure 4.5.c shows the result of Jiawi. The results are 100% match. Just Note that the itemset's codes has been changed in itemset_3 table with their real names using ReferenceTable for human interpretation.

The second verification is going to the Rule-finder. The Apriory simulation method which has been used in the Rule-finder design is applied to the output data of the Data-collector. The results are shown in figure 4.6. It shows that the published results are 100% match with the Rule-finder results.

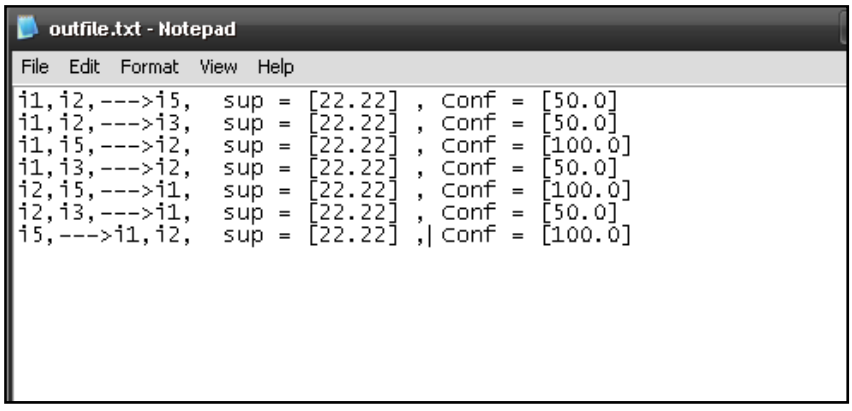


a- Manual Result

b- NetBot Result

c- Jiawi's Results

Figure 4.5 Data-collector verification



a- Rule-finder's result

Figure 4.6 Rule-finder verification

$i_1, i_2 \Rightarrow i_5$ [sup = 22.22% ; Conf = 50%]
$i_1, i_2 \Rightarrow i_3$ [sup = 22.22 % ; Conf = 50%]
$i_1, i_5 \Rightarrow i_2$ [sup = 22.22 % ; Conf = 100%]
$i_1, i_3 \Rightarrow i_2$ [sup = 22.22 % ; Conf = 50%]
$i_2, i_5 \Rightarrow i_1$ [sup = 22.22 % ; Conf = 100%]
$i_2, i_3 \Rightarrow i_1$ [sup = 22.22% ; Conf = 50%]
$i_5 \Rightarrow i_1, i_2$ [sup = 22.22% ; Conf = 100%]

b- Manual results

Figure 4.6 Contended

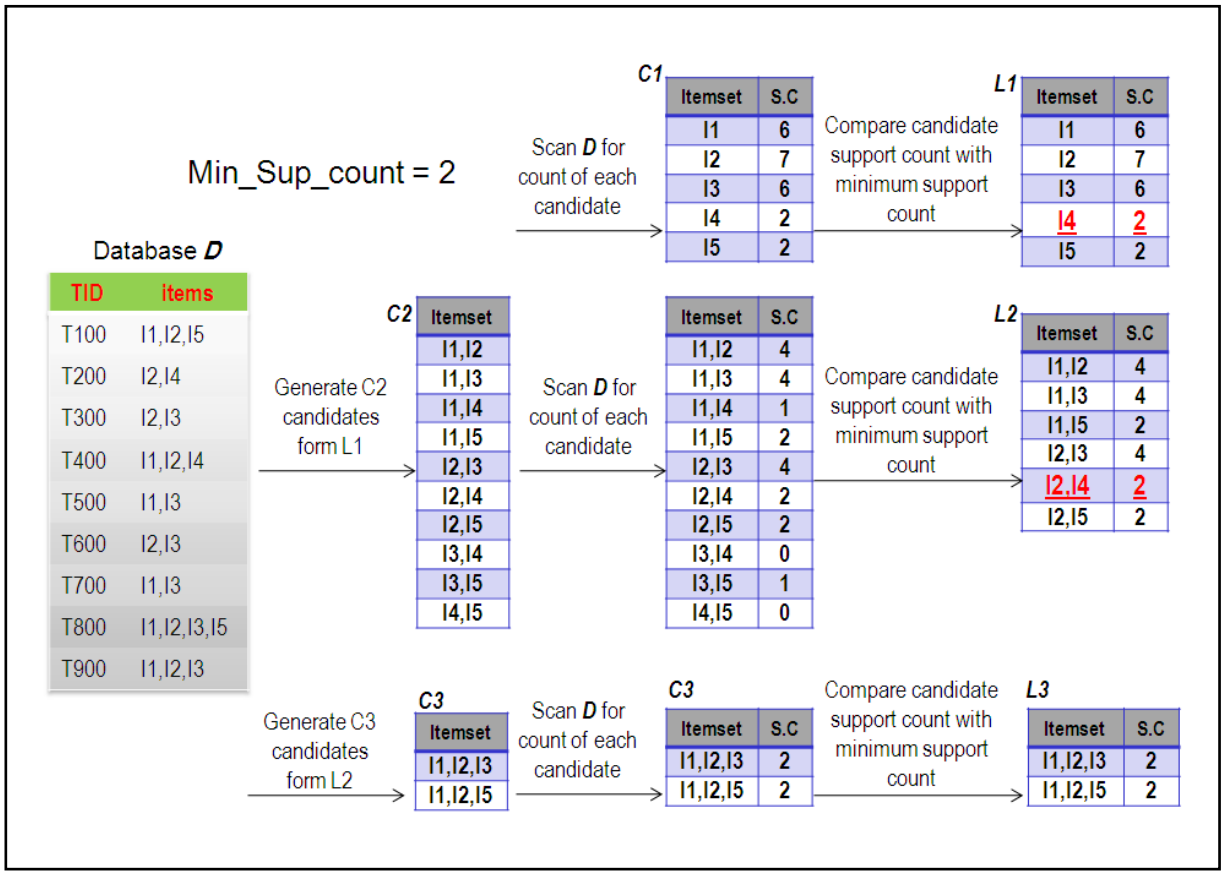


Figure 4.7 An example of Apriori strong itemset detection

4.5 Application of the Proposed System

After the completion of all implementation requirements and system verification, the proposed System has been tested on real data. In this section, the testing procedure together with the results is described.

4.5.1 Test Setting

The proposed system is ideally designed to work on a network of client databases and a proposed system server. The Data-collector collects transactional data on the network and builds Frequency-base in scheduled time. Then the Rule-finder produces association rules as reports according to the user's order. The proposed system can also work on a single computer with client databases on the same computer. In this case, the amount of time required to transfer data over the network is reduced but more disk space is required, because the client database and Frequency-base are placed on the same storage medium.

The following preparations are set for the testing:

1. The Infrastructure

    I. The server computer: HP computer with Intel core due CPU at 2.16GHz speed and 1 GB main memory.

    II. The client computers: two locally collected computers with Intel Pentium 4 at 3 GHz speed and 512 MB main memory are used.

    III. The underlying network: 100 mb/s network adapters with FTP CAT 6 are used to connect computers with a switch by using star topology.

    IV. The software: all computers' operating systems are Windows XP SP2, The database servers are MySql Essential (Version 5.0.27), and Java JDK (version 1.5) and JDBC mysql-connector-java (Version 5.0.4) are used.

2. The Data: A fraction of Kok supermarket in Erbil of one weak is considered. The data it consists of nearly 15000 records with 600 items. After data cleaning and using an abstraction level it became nearly 11000 records with 200 items.

4.5.2 Data Cleaning and Abstraction

The real data available was not clean. Latin characters were mixed with Arabic characters and numbers. Different item names were used for one item. All the data have been cleaned manually to rise the reality and reliability of the test.

A data with 15000 records and 1000 different items are used in the test. An abstract algorithm is used according to the ReferenceTable to abstract the data and reduce the items into nearly 200 groups of items. These 200 items have been considered as input to the system.

4.5.3 Testing the Data-collector

The Data-collector is used to find all itemsets according to user's criteria on the length of the iremsets, in other word, the maximum filter.For the abstracted Kok's data, a portion of the output for different itemset lengths are shown in tables 4.2. This process is time consuming. Its time-itemset length for the test data is shown in figure 4.8. The figure shows that the time is linearly rising with increasing number of records and the itemset length.

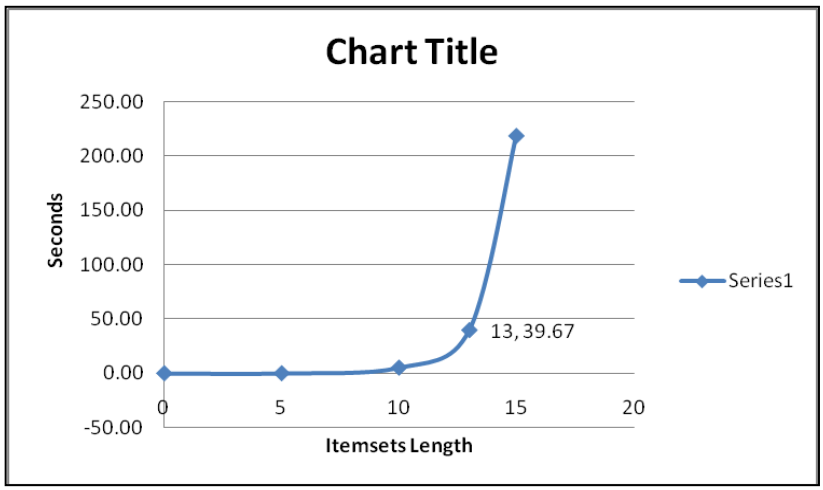


Figure 4.8 Time requirement for generating frequencies of subsets

subset	frequency
0,1,3,7,8,9,10,11,12,13,14,15,16,17,18,	1
0,8,13,18,21,23,36,40,46,65,78,82,124,143,144,	1
1,2,3,5,7,8,16,18,19,34,64,67,97,102,107,	1
1,6,8,21,29,42,79,82,89,90,93,95,96,97,98,	1
2,13,14,21,25,36,47,57,60,72,76,77,78,132,172,	1
2,3,5,7,8,9,13,18,21,31,37,62,82,108,155,	1
2,3,6,7,8,18,21,25,29,48,57,60,75,94,109,	1
2,3,7,14,21,32,33,37,46,54,65,72,77,79,168,	1
2,3,7,8,18,19,29,44,54,65,67,69,72,77,123,	1
2,6,8,18,19,29,43,44,55,66,67,82,89,97,188,	1
2,7,10,14,15,18,25,29,40,42,101,106,144,145,170,	1
2,7,13,18,21,31,34,57,61,72,80,93,95,138,152,	1
3,4,7,12,14,17,21,44,67,70,82,89,101,119,144,	1
3,5,13,14,18,19,29,46,51,55,60,67,72,73,82,	1
3,5,7,8,14,18,27,56,63,72,82,107,122,143,175,	1
3,7,11,13,16,18,21,26,33,37,74,78,88,90,95,	1
3,7,8,17,18,25,27,29,30,31,44,57,112,128,170,	1
4,7,8,13,18,24,25,65,80,83,96,97,108,109,187,	1
5,7,13,20,21,35,40,44,57,65,66,67,68,69,70,	1

a- itemset_15

subset	frequency
21,25,158,	2
7,21,103,	3
7,21,26,	14
7,21,27,	4
7,21,29,	22
7,21,31,	18
7,21,32,	1
7,21,33,	6
7,21,34,	12
7,21,35,	3
7,21,36,	3
7,21,37,	16
7,21,38,	2
7,21,39,	2
7,21,40,	16
7,21,41,	2
7,21,42,	3
7,21,43,	2
7,21,44,	7
7,21,45,	4
7,21,46,	14
7,21,47,	6
7,21,48,	7
7,21,49,	1
7,21,52,	2
7,21,53,	4
7,21,54,	5
7,21,55,	3
7,21,56,	11

b- itemset_3

Figure 4.9 Samples of Frequency-base tables

4.5.4 Testing the Rule-finder

The Rule-finder has been applied to the output of the Data-collector. Three different type tests are carried out to generate special rules accordingly. Rules are specified by the user.

I. Testing Rule Generation for Selective Itemsets

This type of rule generation is more directed and more efficient than the other two types. Four tests are carried out for this rule generation type. The time specifications and the results are given for each test.

Test 1: items selected {biscuit, honey, jelly}

Time required: 31 ms

biscuit, honey ⟹jelly sup = [0.07] , Conf = [5.55]

biscuit, jelly ⟹honey sup = [0.07] , Conf = [5.26]

jelly, honey ⟹biscuit sup = [0.07] , Conf = [100.0]

honey ⟹biscuit, jelly sup = [0.07] , Conf = [2.17]

biscuit ⟹jelly, honey sup = [0.07] , Conf = [0.15]

jelly ⟹biscuit, honey sup = [0.07] , Conf = [3.7]

Test 2: items selected { syrup, soap }

Time required: 17 ms

syrup ⟹ soap sup = [0.61] , Conf = [1.7]

soap⟹ syrup sup = [0.61] , Conf = [21.05]

Test 3: items selected {tea, biscuit}

Time required: 16 ms

tea ⟹biscuit sup = [2.94] , Conf = [43.67]

biscuit ⟹tea sup = [2.94] , Conf = [6.01]

Test 4: items selected {tea, shampoo, toy}

Time required 0 ms

No itemsets found

II Testing Rule Generation for Selected Length of Itemsets

The second criteria has been used by the user includes the minimum confidence, the support and the length of itemsets. Results for some applied criteria are shown in table 4.1. The table shows the efficiency of rule generation and number of rule generated by the system. It is shown that smaller minsup requires more execution time but it still too small relative to the Data-collector's time requirement. Figure 4.10 shows the time-minimum support for k-length itemset for minconf constant at 50%.

Table 4.1 time requirement and number of discovered rules for K length itemset

Itemset Length	Minimum support	Time (seconds)	No. of Rules
2	0.001	0.250	1305
2	0.005	0.156	508
2	0.010	0.094	283
2	0.050	0.031	35
3	0.001	1.219	17090
3	0.005	0.250	1700
3	0.010	0.156	569
3	0.050	0.031	6
4	0.001	2.953	48286
4	0.005	0.250	1190
4	0.010	0.110	184
4	0.050	0.047	0

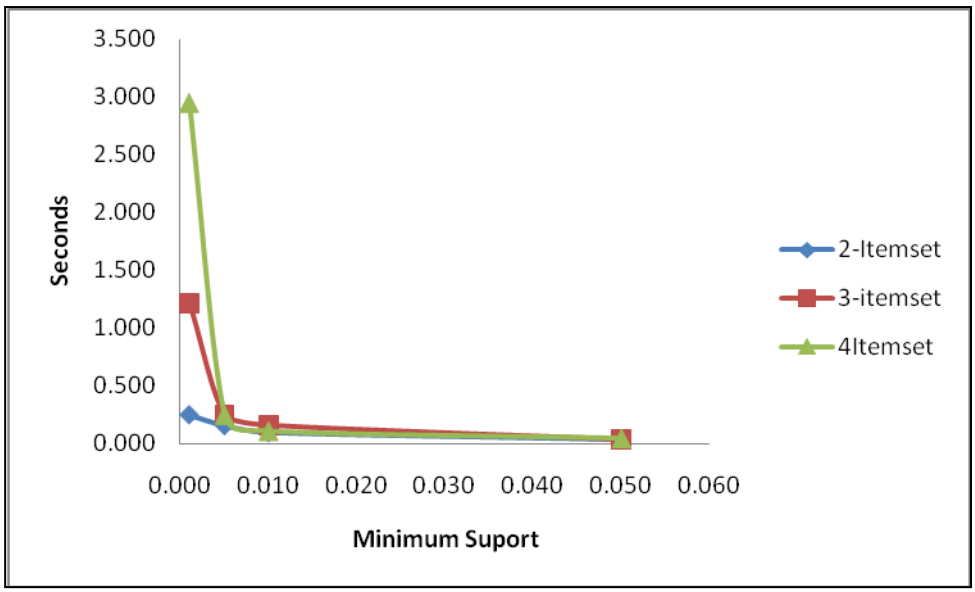


Figure 4.10 Time requirement for generating for K length itemset

III Testing Rule Generation for Apriori Simulating

In this test, the minConf constant is kept at 50% and change minSup to find the required association rules. The time requirement is larger than previous way due to searching in the tables for finding itemsets with provided minSup. In this case, the length of itemsets is determined by minSup which called frequent itemset [RAKE94].

Table 4.2 time requirement, number of discovered rules and Itemset Length for Apriory simulated results

Minimum Support	Max Length	Time (Seconds)	No. of Rules
0.001	11	37.33	216622
0.005	5	10.00	575
0.01	4	5.32	151
0.02	3	0.12	33
0.03	3	0.1	17
0.04	3	0.108	7
0.05	2	0.033	4

Chapter Five

Conclusions and Suggestions for Future Work

5

5.1 Conclusions

A proposed system is designed, implemented, coded, and tested on real data in this thesis. During the developing and testing phases, we have concluded that:

1. By applying a strategy of splitting the Databot into two separate parts and running each part in distinct time, the time needed for the rule generating phase has been reduced significantly.
    - The rule generation for selective itemsets there are an interesting results that the user do not provide minsup and minconf, instead the query is done for finding the support and confidence of a group of itemsete.
    - The rule generation for selected length of itemsets: is quite efficient for finding rules in k length of items the rule generated will be more targeted.
    - The Apriori simulated method is just generates rules as Apriori but very fast and efficient because the input for RuleGenerator is stored frequencies of itemsets
2. Using Frequency-base database with the Data-collector and the RuleGenerator makes running the Rule-finder very fast. Also, this technique led to flexibility in rule selection and generation, because there is no need for re-running the system from the scratch.
3. Results show that the only limitation of the proposed technique is its restriction in itemset length. But in most applications, the itemset length does not exceed 7 to 13, a length which the proposed system is quite efficient.

5.2 Suggestions for Further Work

More enhancements can be done to the proposed system in its time-memory domain. We will mark the following three points for further work:

1. Coding enhancement for the Frequency-base layer to optimize its size, by using ASCII codes instead of numeric codes and the hierarchical code can be implemented to find more abstracted rules.
2. More memory management by forecasting of the data to be held in the main memory and data to be flushed to the database according to pre-specified criteria.
3. In the association rule mining, all algorithms are looking for the strong rules but the week rule mining is also very useful in the industry. This is an open area for the researchers.

References

R

[ABRA97] Abraham S., Henry F. Korth, Sudarshan S., "Database System Concept", 3rd McGraw-Hill, 1997

[BONN01] Bonnie O'neil, Michael S., John D., Kiero H., "Oracle Data Warehousing", SAMS Publishing, 2001

[BUND01] Bundink D., Calimlim M., and Gehrke J., "A maximal frequent itemset algorithm for transactional database", In Int. Conf on Data Engineering, 2001

[CHAR98] Charu C. Aggarwal and Philip S. Yu., "Mining large itemsets for association rules", Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, March 1998, pp. 23-31.

[DANI05] Daniel T. Larose, "Discovering Knowledge in Data: an Introduction to Data Mining", John Wiley & Sons, 2005.

[DAVI01] David H., Heikki M., and Padhraic S., "Principles of Data Mining", MIT Press, 2001.

[FLOC95] Flochart IW. and Radcliffe NJ., "GA-Miner: Parallel data mining with hierarchical genetic algorithms", Report EPCC-AIKMS-GA-Miner Report 1.0, University of Edinburgh, UK, 1995.

[GIOR94] Giordand A., Sailta L., Zini F., "Learning disjunctive concepts by means of genetic algorithms", Proc 10th Int. Conference on Machine Learning (ML-94), Morgan Kaufmann 1994, pp.96-104.

[GOUD01] K. Goudai and M. Z. Jevead, "Efficiently mining maximal frequent itemsets", In 1st IEEE Int. Conf. on data mining, 2001.

[HAND97] Hand DJ., "Construction and assessment of classification rules", John Wiley and Sams, 1997.

[HIPP01] Hipp J., Ulrich G., Gholamreza N., "Algorithms for Association Rule Mining - A General Survey and Comparison", ACM SIGKDD, 2001

[HUSS02] Hussein K. Al-Khafajy, "Design and Implementation of Embedded Association Rules Miner". PhD Thesis, university of Technology, Bagdad, Iraq. October 2002.

[JIAW01] Jiawi H. and Michelin K. ,"Data Mining: Concepts and Techniques", Academic Press, 2001

[KLEM94] Klemeltinen M., Mannila H., Ronkainen P., Toivonen H., and Verkamo AI., "Finding interesting rules from sets of discovered association rules", Prog. Of 3rd Int. Conf. on information and knowledge management, Gaithersburg, Maryland, Nov/Dec 1994.

[MICH00] Michael J. A. Berry and Gordon S. Linoff, "Mastering Data mining the art and science of customer relationship", Wiley, 2000.

[MICH01] Michael M. Walter S., "Data Structures and Other Objects Using C++", Longman, 2001

[MICH02] Michael G., Le Gr., "A survey of data mining and KDD Tools", Journal of UCS, pp 84-107 ,2002.

[MICH04] Michael J., Berry A. and Gordon S. Linoff, "Data Mining Techniques For Marketing, Sales, and Customer Relationship Management", Wiley, 2004

[MICH94] Michael D., Spiegel D.J. halter and Taylor C.C., "Machine Learning, Neural and statistical classification", New York, Ellis Harwood, 1994.

[OGIH00] M. Ogihara and M. Z. Jevead, "Scalable algorithm for rule mining", IEEE Transactions on Knowledge and data Engineering, pp 372-390, 2000

[OGIH98] Ogihara M. and Jevead M.Z., "Foundation of association rules" ACM SIGNOD Workshop on Research Issues in data mining and KDD , 1998.

[PANG06] Pang -Ning Tan; Michael Steinbach and Vipin Kumar, "Introduction to Data Mining", Addison-Wesley, 2006

[PAOL03] Paolo G., "Applied Data Mining", John Wiley, 2003

[PARK98] Park Y. and Song M., "A genetic algorithm for clustering problems" Genetic Programming 1998, Proc. 3rd Annual conf., Morgan Kaufmann, 1998, PP. 568-575.

[PETE00] Peter C., Julian C., Randy K., Thomas K., Thomas R., Colin S., and Rudiger W., "CRISP–DM Step-by-Step Data Mining Guide", 2000.

[PETE98] Peter C., Pablo H., Rolf S., JaapV. , and Alessandro Z., "Discovering Data Mining: From Concept to Implementation", Prentice Hall, 1998.

[RACH01] Racel K., "Data mining Digging user info for geld", ZEND, February 7, 2001

[RAKE93] Rakesh A., Tomasz I., and Arun S.. “Mining association rules between sets of items in large databases”. Int. Proc. of the ACM SIGMOD Conference on Management of Data, Washington, D.C., May 1993, pp. 207-216.

[RAKE94] Rakesh A. and Ramakrishna S., “Fast Algorithms for Mining Association Rules”. In Proc. of the 20th Int'l Conference on Very Large Databases, Santiago Chile, September 1994, pp. 962-969.

[RAKE96a] Rakesh A., and Shafer J., “Parallel mining of association rules”. IEEE Transactions on Knowledge and Data Engineering 8(6), pages 962-969, San Jose, California 1996.

[RAKE96b] Rakesh A., Heikki M., Ramakrishna S., Hannu T., and A. Inkeri V., “Fast Discovery of association rules”, Advances in knowledge discovery and data mining, AAAI/MIT press, 1996, pp. 307 – 328.

[RAME94] Ramzi A. and Shamkant B. Navathe, “Fundamentals of Database Systems”, 2nd, the Benjamin/Cummings Pub, 1994.

[ROBE99] Roberto J. and Bayardo, "Efficiently mining long patterns from databases", In Proceedings of ACM SIGMOD Int. Conf. on Management of data (SIGMOD99), pp. 85-93, Seattle, Washington, USA, May 1997

[SEGE97] Sergey B., Rajeev M., Jeffrey D. Ullman, and Sergey T., "Dynamic itemset counting and implication rules for market basket data". In proceedings of ACM SIGMOD Int. Conf. on Management of Data (SIGMOD97), pp. 255-264, Tucson, Aruzina USA, May 1997

[TECH01] The Technology Review Ten, MIT Technology Review , January/ February may 2001

[WEIS98] Weiss SH. and Indurkhaya, “Predictive data mining: a practical guide”, Morgan Kaufmann, N. Y. 1998.

دیزاین و جێبەجێکردنی تەکنیکی پێشنیارکراو بۆ Data Mining

لێکۆڵینەوەیەکە پێشکەشە بە ئەنجوومەنی کۆلێژی ئەندازیاری زانکۆی سەلاحەددین کراوە وەکو بەشێك لە پێداویستی نامەی ماستەر لە ئەندازیاری تەکنەلۆژیای زانیاری و گەیاندن.

لەلایەن

پۆلا عبدالحمید فتاح

بەکالوریوس لە ئەندازیاری پڕۆگرامسازی

بە سەرپەرشتی

د. ئیبراهیم ئسماعیل حەمەڕەش

یاریدەدەری پڕۆفیسۆر لە ئەندازیاری کۆنترۆڵ

هەولێر-2008

تصميم و تنفيذ تقنية مقترحة لتعدين قواعد الارتباط

دراسة مقدمة الى مجلس كلية الهندسة في جامعة صلاح الدين- أربيل
و هي جزء من متتطلبات لنيل درجة الماجستير في هندسة
تكنولوجيا المعلومات و الاتصالات

من قبل

پۆلا عبدالحميد فتاح

بكالوريوس في هندسة البرامجيات

باشراف

د. ابراهيم اسماعيل حەمەڕەش

مساعد بروفيسور في هندسة السيطرة

اربيل- 2008

پوختە

لەم سەردەمەی ئێستاماندا مرۆڤ خۆی لە جیهانێکی پەرەسەندووو جەنجاڵی پڕ لە داتاو کەم لە زانیاری دۆزیوەتەوە. گەورەترین ئەرك دەکەوێتە سەر شانی پڕۆگرامسازەکان بۆ بەرەوپێش بردنی تەکنیکی نوێ بۆ دۆزینەوەی زانیاری لەناو ئەم بڕە زۆرەی داتا.

لەم لێکۆڵینەوەیەدا پێشنیاری ڕێگەی نوێ کراوە بۆ بەدواگەڕانی داتا (Data Mining)و دواتر ئەم پێشنیارە دارێژراوەو جێبەجێ کراوە لەسەر داتای راستەقینە تاقیکراوەتەوە. ئەم ڕێگە نوێیە پێشنیاری دابەشکردنی کاری پڕۆگرامەکە دەکات بۆ دوو بەشی جیا بۆ کەمکردنەوەی کاتی تێچوو بۆ جێبەجێ کردنی پڕۆگرامەکە.

بەشی یەکەم بەرپرسە لە دۆزینەوەی هەموو بەشە کۆمەڵەکانی هەر ترانزەکشنێك (Transaction) لەدواییدا هەڵگرتنی ژمارەی دووبارەبوونەوەکانی لەناو داتابەیزدا. ئەم ڕێگەیە بانگکردنی هەر ترانزەکشنێك کەم دەکاتەوە بۆ تەنها یەك جار ئەمەش لە تێچوو/ دەرچوو (I/O) کەمدەکاتەوە.

بەشی دووەم لە پڕۆگرامەکە دەرەنجامی بەشی یەکەم بەکاردەهێنێت کە بەشە کۆمەڵەکانو دووبارەبوونەوەکانی لەخۆدەگرێت. ئەم کردارە کاتی چاوەڕوانکردنی بەکارهێنەر کەمدەکاتەوەو فلێکسیبیلیتی پڕۆگرامەکە زیاد دەکات.

الخلاصة

في عصرنا هذا وجد الانسان نفسه يعيش في عالم متوسع من البيانات حيث يوجد الكثير منه مع القليل من المعرفة. يكمل التحدي الاكبر لمطوري البرامج في تطوير تقنيات حديثة لايجاد المعرفة من هذا الكم الهائل من البيانات.

في هذه الدراسة قد تم اقتراح و تصميم و تنفيذ طريقة جديدة لتعدين البيانات لايجاد قواعد الارتباط حيث جرب الطريقة على بيانات واقعية حقيقية، الطريقة الجديدة تقترح تقسيم عمل و تنفيذ البرنامج الى قسمين مختلفين من اجل تقليص زمن عمل البرنامج.

القسم الاول مسؤول عن ايجاد كل المجموعات الجزئية لجميع التعاملات من ثم خزن الترددداتهم في قاعدة البيانات. بهذه الطريقة يقلص استدعاء كل تعامل مرة واحدة فقط مما يقلل اخراج/ادخال (I/O).

القسم الثاني من البرنامج يستخدم ناتج جزء الاول من البيانات التي تتظمن المجموعات الجزئية و تردداتها. هذه العملية يقلص الوقت الانتظار لدى المستخدم و يعطي البرنامج مرونة اكبر.