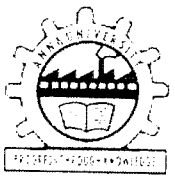


P-3130

HIDING SENSITIVE ASSOCIATION RULES WITH LIMITED SIDE EFFECTS

A PROJECT REPORT

Submitted by

C.S.PRIYADARSHINI 71206104036

D. HANU KARUNYA LAKSHMI 71206104303

In partition fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY, CHENNAI-600 025

APRIL 2010

BONAFIDE CERTIFICATE

Certified that this project report entitled "**Hiding Sensitive Association Rules With Limited Side Effects**" is the bonafide work of **C.S.Priyadarshini** and **D.Hanu Karunya Lakshmi**, who carried out the research under my supervision. Certified also, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.S.Thangasamy, Ph.D

DEAN &

HEAD OF THE DEPARTMENT

Department of

Computer Science & Engineering,

Kumaraguru College Of Technology,

Coimbatore-641006.

15/4/10

SIGNATURE

Mrs. M.Anidha, M.E.

Guide

Lecturer

Department of

Computer Science & Engineering,

Kumaraguru College Of Technology,

Coimbatore-641006.

The candidates with University Register Nos. 71206104036 and 71206104303 were examined by us in the project viva-voce examination held on 16.4.2010

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We hereby declare that the project entitled "**Hiding Sensitive Association Rules With Limited Side Effects**" is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any Institutions, for fulfillment of the requirement of the course study.

The report is submitted in partial fulfillment of the requirement for the award of the Degree of Bachelor of Computer Science and Engineering of Anna University, Chennai.

Place: Coimbatore

Date : 16.4.2010

(C.S.Priyadarshini)

(D.Hanu Karunya Lakshmi)

ACKNOWLEDGEMENT

We extend our sincere thanks to our Principal, **Dr.S.Ramachandran,** Kumaraguru College Of Technology, Coimbatore, for being a constant source of inspiration and providing us with the necessary facility to work on this project.

We would like to make a special acknowledgement and thanks to **Dr.S.Thangasamy Ph.D.,** Dean, Professor and Head of Department of Computer Science &Engineering and **Mrs.P.Devaki (Ph.D),** project coordinator for their support and encouragement throughout the project.

We express our deep gratitude and gratefulness to our Guide **Mrs.M.Anidha M.E.,** Department of Computer Science & Engineering, for her supervision, enduring patience, active involvement and guidance.

We would like to convey our honest thanks to **all Faculties** of the Department for their enthusiasm and wealth of experience from which we have greatly benefited.

We also thank our **friends and family** who helped us to complete this project fruitfully.

ABSTRACT

With rapid advance of network and data mining techniques, the protection of the confidentiality of sensitive information in a database becomes a critical issue to be resolved. Association analysis is a powerful and popular tool for discovering relationships hidden in large data sets. The relationships can be represented in a form of frequent itemsets or association rules. One rule is categorized as sensitive if its disclosure risk is above some given threshold. Privacy preserving data mining is an important issue which can be applied to various domains, such as Web commerce, crime reconnoitering, health care and customer's consumption analysis.

The main approach to hide a sensitive association rule is to reduce the support of each given sensitive association rule. This is done by modifying transactions or items in the database. However, the modifications will generate side effects, i.e. non-sensitive data falsely generated. Furthermore, it would always take huge computing time to solve the problem. In our work, we propose a novel algorithm, to hide the sensitive data and generate a minimum amount of side effects.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vi
	LIST OF ABBREVIATIONS	vii
1.	**INTRODUCTION**	1
	1.1 OVERVIEW OF DATAMINING	1
	1.2 WHAT MOTIVATED DATA MINING	2
	1.3 DATA ANALYSIS TOOLS	2
	1.3.1 Classification	2
	1.3.1.1 Learning	2
	1.3.1.2 Classification	3
	1.3.2 Clustering	3
	1.3.3 Regression	4
	1.3.4 Association Rules	4
	1.4 Frequent Itemsets	5
	1.4.1 Sensitive Frequent Itemsets	5
2.	**PROBLEM OVERVIEW**	6
	2.1 PROBLEM DEFINITION	6
	2.2 GOALS OF THE PROJECT	6
	2.3 EXISTING SYSTEM	6
	2.4 PROBLEMS IN EXISTING SYSTEM	7
	2.5 PROPOSED SYSTEM	7
3.	**LITERATURE REVIEW**	8
	3.1 APPLICATIONS OF ASSOCIATION RULES	8
	3.2 UNDESIRED EFFECTS OF ASSOCIATION RULES	8

4.	**SYSTEM CONFIGURATION**		**11**
	4.1	SOFTWARE CONFIGURATION	11
	4.2	HARDWARE CONFIGURATION	11
	4.3	FEATURES OF VISUAL BASIC .NET	12
		4.3.1 Powerful windows based applications	12
		4.3.2 Building Web-based Applications	12
		4.3.3 Simplified Deployment	13
		4.3.4 Powerful,Flexible,Simplified Data Access	13
		4.3.5 Improved Coding	13
		4.3.6 Direct Access to Platform	14
		4.3.7 Full Object-Oriented Constructs	14
		4.3.8 XML Web Services	14
		4.3.9 Mobile Applications	14
	4.4	FEATURE OF VISUAL STUDIO .NET 2010	15
		4.4.1 New Features in the Visual Studio 2010	15
		4.4.2 Call Hierarchy of Methods	15
		4.4.3 New Quick Approach	16
		4.4.4 Multi-Targeting more Accurate	16
		4.4.5 Parallel Programming and Debugging	16
		4.4.6 XSL Profiling and Debugging	16
5.	**METHODOLOGY**		**17**
	5.1	Data Loading	17
	5.2	Applying ISL	17
	5.3	Applying FHFSI	18
6.	**CONCLUSION**		**19**
7.	**FUTURE ENHANCEMENTS**		**19**
8.	**REFERENCES**		**58**

LIST OF ABBREVIATIONS

MST	Minimum Support Threshold
MCT	Minimum Confidence Threshold
PWT	Prior Weight
SFI	Sensitive Frequent Itemsets
D	Database
D'	Modified Database
ISL	Increased Support of LHS
DSR	Decreased Support of RHS

1. INTRODUCTION

1.1 Overview of Data Mining

Data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining tools perform data analysis and uncover important data patterns, contributing greatly to business strategies, knowledge bases, scientific and medical research. The tools bring out the hidden patterns from larger data sources and these patterns help in decision making, in a process.

While data mining can be used to uncover hidden patterns in data samples that have been 'mined', it is important to be aware that the use of a sample of the data may produce results that are not indicative of the domain. Data mining will not uncover patterns that are present in the domain, but not in the sample. There is a tendency for insufficiently knowledgeable 'consumers' of the results to treat the techniques as a sort of crystal ball and attribute 'magical thinking' to it. Like any other tool, it only functions in conjunction with the appropriate raw material: in this case, indicative and representative data that the user must first collect. Further, the discovery of a particular pattern in a particular set of data does not necessarily mean that pattern is representative of the whole population from which that data was drawn. Hence, an important part of the process is the verification and validation on other samples of data.

1.2 What Motivated Data Mining?

Data mining has attracted a great deal of attention in the information industry and in society as a whole in the recent years, due to the wide availability of huge amounts of data and the imminent need for turning such data into useful information and knowledge. The information gained can be used for applications ranging from market analysis, fraud detection, and customer retention, to production control and science exploration.

Data mining can be viewed as a result of natural evolution of information technology. The database system industry has witnessed an evolutionary path in the development of the following functionalities: data collection and database creation, data management (including data storage and retrieval, and database transaction processing) and advanced data analysis.

1. 3 Data Analysis Tools

Data mining commonly involves four classes of tasks:

1.3.1 Classification

Classification arranges the data into predefined groups. For example, an email program might attempt to classify an email as legitimate or spam. This helps to predict future data trends. Such analysis can help provide us with a better understanding of data at large. Data Classification is a two step process:

1.3.1.1 Learning

A classifier is built describing a set of data classes or concepts known as training set. The training data is analyzed by the classification algorithm and the learned model or classifier is represented in the form of classification algorithm.

1.3.1.2 Classification

Test data are used to estimate the accuracy of the classification rules. If the accuracy is acceptable, the rules can be applied to the classification of new data tuples.

Common algorithms include

(1) Decision tree learning classification

(2) Nearest neighbor classification

(3) Naive Bayesian classification and

(4) Neural networks.

1.3.2 Clustering

The process of grouping a set of physical or abstract objects into classes of similar objects is called clustering. A cluster is a collection of data objects that are similar to one another within the same cluster and are dissimilar to the objects in other clusters. A cluster of data objects can be treated collectively ad one group and so may be considered as a form of data compression. Although classification is an effective means fir distinguishing groups of classes of objects, it requires the often costly collection and labeling of large set of training patterns, which the classified uses to model each group. It is like classification but the groups are not predefined, so the algorithm will try to group similar items together.

In general, the major clustering methods can be classified into the following categories

(1) Partitioning methods

(2) Hierarchical methods

(3) Density-based methods

(4) Grid-based methods

(5) Model-based methods.

1.3.3 Regression

Regression analysis is a statistical methodology that is most often used for numeric prediction. It helps us understand how the typical value of the dependent variable changes when any one of the independent variables is varied, while the other independent variables are held fixed. Most commonly, regression analysis estimates the conditional expectation of the dependent variable given the independent variables — that is, the average value of the dependent variable when the independent variables are held fixed. In all cases, the estimation target is a function of the independent variables called the regression function. In regression analysis, it is also of interest to characterize the variation of the dependent variable around the regression function, which can be described by a probability distribution.

1.3.4 Association Rules

Association rule learning is a popular and well researched method for discovering interesting relations between variables in large databases. An example for association rule is,

Computer => antivirus_ software [support=2% , confidence= 60%]

This rule indicates that customers who purchase computers also tend to buy antivirus software at the same time. Rule **support** and **confidence** are two measures of rule interestingness. They respectively reflect the usefulness and certainty of discovered rules. A support of 2% means that, 2% of all the transactions under the analysis show that computer and antivirus are purchased together. A confidence of 60% means that,

60% of the customers who purchase a computer also bought the software. Typically, association rules are considered interesting if they satisfy both minimum support threshold (MST) and minimum confidence threshold (MCT). Such threshold can be set by users of domain experts.

Such information can be used as the basis for decisions about marketing activities such as, e.g., promotional pricing or product placements. In addition to the above example from market basket analysis association rules are employed today in many application areas including Web usage mining, intrusion detection and bioinformatics.

1.4 Frequent Itemsets

Frequent patterns are patterns that appear in a data set frequently. For example, a set of items, such as milk and bread that appear frequently together in a transaction data set is a frequent itemset or association rule. For example, first buying a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database a frequent pattern is formed. Finding such frequent pattern plays an essential role in mining associations, correlations, clustering and other mining tasks as well.

1.4.1 Sensitive Frequent Itemsets (SFI)

Patterns that appear frequently in a data set are frequent patterns. Not all these patterns obtained are sensitive. The pattern is categorized as sensitive if its disclosure risk is above some given threshold. With an association analyzer, if an itemset above a given minimal support, we call the itemset as a frequent itemset. Protection of confidentiality of these sensitive information has become a critical issue to be resolved. If such information goes to the unintended persons, serious damage may tend to arise. So protection of such data is the main idea.

2. PROBLEM OVERVIEW

2.1 Problem Definition

The data mining technologies have been an important technology for discovering previously unknown and potentially useful information from large data sets or databases. They can be applied to various domains, such as Web commerce, crime reconnoitering, health care, and customer's consumption analysis. Although these are useful technologies, there is also a threat to data privacy. For example, the association rule analysis is a powerful and popular tool for discovering relationships hidden in large data sets. Therefore, some private information could be easily discovered by this kind of tools. The protection of the confidentiality of sensitive information in a database becomes a critical issue to be resolved.

2.2 Goals of the project

- To successfully hide the sensitive association rule, by reducing the support of the sensitive itemsets.
- To reduce the time taken for hiding the rules.

2.3 Existing System

Many existing systems successfully hide the sensitive association rules. But those algorithms have certain limitations. Vassilios S. Verykios et al. presented algorithms to hide sensitive association rules, but they generate high side effects and require multiple database scans. Instead of hiding sensitive association rules, Shyue-Liang Wang proposed algorithms to hide sensitive items. The algorithm needs less number of database scans but the side effects generated is higher. Ali Amiri also

presented heuristic algorithms to hide sensitive items. Finally, Yi-Hung Wu et al. proposed a heuristic method that could hide sensitive association rules with limited side effects. However, it spent a lot of time on comparing and checking if the sensitive rules are hidden and if side effects are produced. Besides, it could fail to hide some sensitive rules in some cases.

2.4 Problems in the existing system

Although the existing systems successfully hide the sensitive patterns, they produce side effects. In some cases large number of database scans is required. At times, the system could fail to hide some sensitive rules, and produces large amounts of false positives. Due to large number of database scans the time taken to hide the sensitive rules becomes high. Due to the large number of side effects produced, we go in for the proposed system.

2.5 Proposed System

We propose a simple yet very effective method novel that leads fast hiding sensitive frequent itemsets (SFI). This method can hide all SFI without generating all frequent itemsets. It only generates limited side effects. It allows any minimum support thresholds, and only one database scan is required. Within this scan all the necessary information for hiding is taken. The time taken to hide the items is optimized.

3. LITERATURE REVIEW

A variety of data mining problems have been studied to help people get an insight into the huge amount of data. One of them is association rule mining. An itemset is a set of products (items) and a transaction keeps a set of items bought at the same time. The support of an itemset I (denoted as SupI) in a transaction database is the percentage of transactions that contain I in the entire database. An itemset is frequent if its support is not lower than a minimum support threshold (denoted as MST). For two itemsets X and Y where X \ Y ¼ ;, the confidence of an association rule X ! Y (denoted as ConfX!Y) is the probability that Y occurs given that X occurs, and is equal to SupX[Y divided by SupX. We say that X ! Y holds in the database if X [Y is frequent and its confidence is not lower than a minimum confidence threshold (denoted as MCT). Such a rule is called the strong association rule (strong rule for short). Association rule mining is to discover all the strong rules in the database. However, the misuse of them may bring undesired effects to people.

3.1 Applications of Association Rules

Association rules are typically used in market analysis (market basket analysis), primarily because of the utility and clarity of its results. They express how important products or services relate to each other, and immediately suggest particular actions. Association rules are used in mining categorical data - items. Besides the sole process of generating association rules, the process of application of association rules technique involves two important concerns:

1) Choice of the right set of items

The data used for association rule analysis is typically the detailed transaction data captured at the point of sale. Gathering and using this data is a critical part of applying association rule analysis, depending crucially on the items chosen for analysis. What constitutes a particular item depends on the business (problem) need. Items in stores usually have codes that form hierarchical categories (taxonomy). These categories help in generalization, and reduction of the volume of items used for a study. Dozens or hundreds of items may be reduced to a single generalized item, often corresponding to a single department or type of a product.

2) Practical limits imposed by a large number of items appearing in combinations large enough to be interesting

Number of combinations for larger itemsets rises exponentially with the number of items. Calculating the support, confidence, and improvement for a grocery store with thousands of different items, quickly rises to millions, as the number of items in the combinations grows. For example for 1000 products, total number of combinations of three products is:

$$\binom{n}{k} = \binom{1000}{3} = 166.167 * 10^{6} !!!!$$

Calculating the counts for five or more items can be completely out of hand. In that case the use of taxonomies reduces the number of items to a manageable size.

Generally, the strengths of association rule analysis are:

- It produces clear and understandable results.
- It supports undirected data mining (no target attribute).
- It works on data of variable length.
- The computation algorithm it uses is quite simple.

3.2 Undesired effects due to misuse of association rules

Consider a supermarket and two beer suppliers A and B. If the transaction database of the supermarket is released, A (or B) can mine the association rules related to his/her beers and apply the rules to the sales promotion and the goods supply. As a result, a supplier is willing to exchange a lower price of goods for the database with the supermarket. From this aspect, it is good for the supermarket to release the database. However, the conclusion can be opposite if a supplier uses the mining methods in a different way. For instance, if A finds the association rules related to B's beers, saying that most customers who buy diapers also buy B's beers, he/she can run a coupon that gives a 10 percent discount when buying A's beers together with diapers. Gradually, the amount of sales on B's beers is down and B cannot give a low price to the supermarket as before.

Finally, A monopolizes the beer market and is unwilling to give a low price to the supermarket as before. From this aspect, releasing the database is bad for the supermarket. Therefore, for the supermarket, an effective way to release the database with sensitive rules hidden is required. This is not only in the case of a supermarket, this is just a small example to make us understand the importance of the confidentiality of the data present in database. More serious problems can arise when the confidentiality of the defence industry or crime database is lost.

4. SYSTEM CONFIGURATION

4.1 Software Configuration

The software used for the development of the project:

Operating System	:	Windows Vista Home Premium
Environment	:	Visual Studio .NET 2010
Language	:	VB.NET
Back End	:	Microsoft Access 2007

4.2 Hardware Configuration

The hardware used of the development of the project is:

Processor	:	Intel Core 2 Duo 1.50Gz
RAM	:	2038 MB SD RAM
Hard Disk	:	120 GB
Monitor	:	15'' Color
Keyboard	:	Standard American Type

4.3 Features of Visual Basic .NET

Visual Basic .NET provides the easiest, most productive language and tool for rapidly building Windows and Web applications. Visual Basic .NET comes with enhanced visual designers, increased application performance, and a powerful integrated development environment (IDE). It also supports creation of applications for wireless, Internet-enabled hand-held devices. The following are the features of Visual Basic .NET with .NET Framework 1.0 and Visual Basic .NET 2003 with .NET Framework 1.1. This also answers why should we use Visual Basic .NET, what can we do with it?

4.3.1 Powerful Windows-based Applications

Visual Basic .NET comes with features such as a powerful new forms designer, an in-place menu editor, and automatic control anchoring and docking. Visual Basic .NET delivers new productivity features for building more robust applications easily and quickly. With an improved integrated development environment (IDE) and a significantly reduced startup time, Visual Basic .NET offers fast, automatic formatting of code as you type, improved IntelliSense, an enhanced object browser and XML designer, and much more.

4.3.2 Building Web-based Applications

With Visual Basic .NET we can create Web applications using the shared Web Forms Designer and the familiar "drag and drop" feature. You can double-click and write code to respond to events. Visual Basic .NET 2003 comes with an enhanced HTML Editor for working with complex Web pages. We can also use IntelliSense

technology and tag completion, or choose the WYSIWYG editor for visual authoring of interactive Web applications.

4.3.3 Simplified Deployment

With Visual Basic .NET we can build applications more rapidly and deploy and maintain them with efficiency. Visual Basic .NET 2003 and .NET Framework 1.1 makes "DLL Hell" a thing of the past. Side-by-side versioning enables multiple versions of the same component to live safely on the same machine so that applications can use a specific version of a component. XCOPY-deployment and Web auto-download of Windows-based applications combine the simplicity of Web page deployment and maintenance with the power of rich, responsive Windows-based applications.

4.3.4 Powerful, Flexible, Simplified Data Access

You can tackle any data access scenario easily with ADO.NET and ADO data access. The flexibility of ADO.NET enables data binding to any database, as well as classes, collections, and arrays, and provides true XML representation of data. Seamless access to ADO enables simple data access for connected data binding scenarios. Using ADO.NET, Visual Basic .NET can gain high-speed access to MS SQL Server, Oracle, DB2, Microsoft Access, and more.

4.3.5 Improved Coding

You can code faster and more effectively. A multitude of enhancements to the code editor, including enhanced IntelliSense, smart listing of code for greater readability and a background compiler for real-time notification of syntax errors transforms into a rapid application development (RAD) coding machine.

4.3.6 Direct Access to the Platform

Visual Basic developers can have full access to the capabilities available in .NET Framework 1.1. Developers can easily program system services including the event log, performance counters and file system. The new Windows Service project template enables to build real Microsoft Windows NT Services. Programming against Windows Services and creating new Windows Services is not available in Visual Basic .NET Standard, it requires Visual Studio 2003 Professional, or higher.

4.3.7 Full Object-Oriented Constructs

You can create reusable, enterprise-class code using full object-oriented constructs. Language features include full implementation inheritance, encapsulation, and polymorphism. Structured exception handling provides a global error handler and eliminates spaghetti code.

4.3.8 XML Web Services

XML Web services enable you to call components running on any platform using open Internet protocols. Working with XML Web services is easier where enhancements simplify the discovery and consumption of XML Web services that are located within any firewall. XML Web services can be built as easily as you would build any class in Visual Basic 6.0. The XML Web service project template builds all underlying Web service infrastructure.

4.3.9 Mobile Applications

Visual Basic .NET 2003 and the .NET Framework 1.1 offer integrated support for developing mobile Web applications for more than 200 Internet-enabled mobile devices. These new features give developers a single, mobile Web interface and programming model to support a broad range of Web devices, including WML 1.1 for WAP—enabled cellular phones, compact HTML (cHTML) for i-Mode phones, and HTML for Pocket PC, handheld devices, and pagers. Please note. Pocket PC programming is not available in Visual Basic .NET Standard, it requires Visual Studio 2003 Professional, or higher.

4.4 Features of Visual Studio .NET 2010

Visual Studio 2008 may be better than sliced bread, but the development team at Microsoft has already been working on the next release. They have recently given us Visual Studio 2010 and the .NET Framework 4.0 as a Community Technology Preview (CTP), it boasts several features that would appeal to developers.

4.4.1 New Features in the Visual Studio 2010 IDE

- Call Hierarchy of methods
- A New Quick Search
- Multi-targeting more accurate
- Parallel Programming and Debugging
- XSLT Profiling and Debugging

4.4.2 Call Hierarchy of Methods

In complicated solutions, a single method may be used from several different places, and attempting to follow how a particular method is being called can be difficult. Call hierarchy attempts to address this problem by visually presenting the

flow of method calls to and from the method which is being looked at. In other words, it can look at what calls the method and what the method calls in a treeview format.

4.4.3 A New Quick Search

A nifty little feature that Microsoft has added is the Quick Search window. This isn't the same as the Search or Search and Replace window that searches for specific textual strings. It's different in the sense that it searches across symbols (methods, properties, and class names) across the solution and filters them in the result view.

4.4.4 Multi-targeting more accurate

Although VS 2008 supports targeting different frameworks from the same IDE, one problem was that the Toolbox displayed types that were available to the .NET 3.5 Framework whether or not you were working with a .NET 3.5 project. This may have caused problems when it tried to use something, only to realize that it wasn't actually available.

4.4.5 Parallel Programming and Debugging

In addition, to make things easier, VS 2010 comes with a set of visual tools that will help you debug and view simultaneously running threads. This means that the task instances can be viewed and call the stacks for each task in parallel.

4.4.6 XSLT Profiling and Debugging

Visual Studio 2010 will offer an XSLT profiler to help with writing XSLT in the context of profiling and optimization. After writing your XSLT, you can use the "Profile XSLT" option in Visual Studio to supply it with a sample XML file that gets used for the analysis.

5. METHODOLOGY

5.1 Data Loading

The data which is required for the process is present in the text database. This cannot be used as such. First it is imported to the access database and saved. Then data is viewed as a grid format in the user interface of this system. All the fields are checked for the null value. If any field is empty, a default value is replaced instead of the null value. Here the database consists of a customer identification number and the item that he has purchased. These are the two values that are present in the database.

5.2 Appling ISL

In this module we use the ISL method of hiding the sensitive items. The following are the steps carried out in this module. From the database, generate the transaction database, in which the items are present in the form of transactions. Calculate the support and confidence for the items preset using the formula: Support (X) = ||X|| / |D| , Support (XUY) = ||XUY|| / |D| ,Confidence (XUY) = ||XUY|| / | X | Where, |D| - No. of transactions in Original Database. ||X|| - No. of transactions in Database that contains the itemset X. Set the minimum support and confidence threshold levels for classifying sensitive and non-sensitive data. Select all the data items above the given threshold levels, and apply the ISL method. This method selects the item in the left hand side and replaces it on the right hand side of the rule. This automatically reduces the support count of the item on the right side and increases the support on the left hand side. So this rule will not be considered sensitive when it is being associated. The association rule is being hidden in the process.

5.3 Applying FHFSI

In this module, we implement the FHFSI to hide the association rules in a more efficient time than ISL. The following are the steps followed in the algorithm to achieve the result. Transactions database is generated along with the prior weight (PWT). The support alone is calculated for the items in the transaction database and it is displayed. The minimum support threshold is set and the itemsets above this value is considered as sensitive. After this, the all the items above the given value is sorted out in the descending order of their prior weight. Then randomly an item is chosen and replaced with a fake item. By doing so the association rule is being modified so it does not classify under sensitive association rules. These changes are updated in the database and this is given as the output. It can be clearly seen that the time taken to complete the hiding process. This is the improvement that is obtained in the proposed system

A time stamp is inserted for the hiding process in both the methods for analysing the time taken for hiding the sensitive patterns. It is clearly seen that the new process hides the sensitive items within a short time when compared to the existing system. This proves that the proposed system is effective.

6. CONCLUSION

The goals of this project are successfully accomplished. It successfully hides all the sensitive items. It protects the confidentiality of the database, for all the minimum threshold levels this algorithm works effectively. The time taken to hide the sensitive items, is within a less amount of time compared to the existing system. Thus we can say that the computational burden had been greatly reduced.

7. FUTURE ENHANCMENTS

Every application has its own merits and demerits. The project has almost covered all requirements. But our algorithm still causes some loss rule sets and the generation of the false items cannot be avoided in this system. Certain extensions on the algorithms are to be considered to solve these problems. The project calls for further investigation into possible improvement of the undesired side effects.

SAMPLE CODING

```
Imports System.Net
Imports System.IO
Imports System.Windows.Forms.DataVisualization.Charting
Public Class Home
   Dim Datas As New List(Of PatternLib.LinkType)
   Dim Orglist As New List(Of ReadData)
   Dim OldOrglist As New List(Of ReadData)
         Private Sub Button1_Click(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles Button1.Click
    RichTextBox1.LoadFile("D:\SensitivePatterns\Patterns\shoppingtest.txt",
    RichTextBoxStreamType.PlainText)
     For i As Integer = 2 To prdcount
        Dim Rd As New ReadData
        Rd.CustomerId = RichTextBox1.Lines(i).Split(",")(0).ToString()
        Rd.Product = RichTextBox1.Lines(i).Split(",")(1).ToString()
        Orglist.Add(Rd)
     Next
     Dim qry = From k In Orglist Select k
     Dim con1 As New System.Data.OleDb.OleDbConnection
      con1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=D:\SensitivePatterns\Patterns\FrequentResultSet\ResultSet.mdb;Persist
Security Info=True"
```

```
        con1.Open()
        Dim delcmd As New OleDb.OleDbCommand
        delcmd.Connection = con1
        delcmd.CommandText = "delete from originalset"
        delcmd.ExecuteNonQuery()
        con1.Close()
        Dim con As New System.Data.OleDb.OleDbConnection
                con.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=D:\SensitivePatterns\Patterns\FrequentResultSet\ResultSet.mdb;Persist
Security Info=True"
       con.Open()
       For Each tm In qry
       Dim cmd As New System.Data.OleDb.OleDbCommand
       cmd.Connection = con
                        cmd.CommandText = "insert into OriginalSet(Cartid,Item)
         values('" & tm.CustomerId & "','" & tm.Product & "')"

      cmd.ExecuteNonQuery()
       Next
       con.Close()
 MessageBox.Show("Processed And Original Set Saved  : "
 &qry.Count.ToString())
  End Sub
        DataGridView1.DataSource = qry.ToList
Dim Transset As New List(Of TransactionSet)
Dim OldTransset As New List(Of TransactionSet)
```

```
        Private Sub LoadTrans_button_Click(ByVal sender As System.Object,
ByVal e As  System.EventArgs)
    End Sub
    Dim fset As New List(Of String)
    Dim FSSET As New List(Of SUVal)
    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs)
    End Sub
     Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles Button4.Click
      Dim con1 As New System.Data.OleDb.OleDbConnection
            con1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=D:\SensitivePatterns\Patterns\FrequentResultSet\ResultSet.mdb;Persist
Security Info=True"
      con1.Open()
      Dim delcmd As New OleDb.OleDbCommand
      delcmd.Connection = con1
      delcmd.CommandText = "delete from Resultset"
      delcmd.ExecuteNonQuery()
      con1.Close()
      MessageBox.Show("Data Export Complete")
       For Each cid In SUSet
          Dim ct = cid.CID
          Dim inr = From k In Orglist Where k.CustomerId = ct Select k
          For Each t In inr
             Dim con As New System.Data.OleDb.OleDbConnection
```

```
                con.ConnectionString =
            "Provider=Microsoft.Jet.OLEDB.4.0;Data
            Source=D:\SensitivePatterns\Patterns\FrequentResultSet\ResultSet
            .mdb;Persist Security Info=True"
        con.Open()
        Dim cmd As New System.Data.OleDb.OleDbCommand
        cmd.Connection = con
                cmd.CommandText = "insert into Resultset(cartid,item)
    values('" & t.CustomerId & "','" & t.Product & "')"
        cmd.ExecuteNonQuery()
        con.Close()
      Next
    Next
    FSSET.Clear()
    Transset.Clear()
    j = 0
End Sub
Dim j = 0
Dim z = 0
     Private Sub Button5_Click(ByVal sender As System.Object, ByVal e
     As System.EventArgs) Handles Button5.Click
     TreeView1.Nodes.Clear()
          Dim Da = From k In Orglist Group k By key = k.CustomerId Into
    Group Select Cat = key, Movies = Group
    For Each d In Da
       Dim tr As New TransactionSet
       j = j + 1
```

```
        Dim i As Integer = 0
        TreeView1.Nodes.Add(d.Cat, d.Cat)
        For Each r In d.Movies
            i = i + 1
            TreeView1.Nodes(d.Cat).Nodes.Add(r.Product)
            tr.Prds.Add(r.Product)
        Next
        tr.Count = i.ToString
        tr.CustomerID = d.Cat
        tr.TId = j.ToString
        TreeView1.Nodes(d.Cat).Text = "[T" & j.ToString & "], ID" & d.Cat &
        " ,Piror Weight =" & i.ToString
        TreeView1.Nodes(d.Cat).ForeColor = Color.Blue
        Transset.Add(tr)
    Next
    TreeView1.ExpandAll()
End Sub
Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button6.Click
    Dim t1 = Now
    If ComboBox1.Text.Length = 0 Then
        MessageBox.Show("Select Treshhold Value")
    Else
        DataGridView3.DataSource = Nothing
        Dim k = From l In SUSet Order By l Descending Select l
        Dim s = k.Distinct
        Dim tset As New List(Of Double)
```

```
Dim min = Int32.MaxValue
For Each rt As NSUVal In k
  tset.Add(Double.Parse(rt.Support))
Next
Dim sset = From ss In tset Where ss > Double.Parse(ComboBox1.Text)
Select ss
For Each h In sset
  Dim hh = h
  Dim fst = From st In SUSet Where st.Support = hh Select st
  For Each hd In fst
    For i As Integer = 0 To Orglist.Count - 1
      If Orglist(i).CustomerId = hd.CID Then
        Dim con As New System.Data.OleDb.OleDbConnection
        con.ConnectionString =
        "Provider=Microsoft.Jet.OLEDB.4.0;Data
         Source=D:\SensitivePatterns\Patterns\FrequentResultSet\
         ResultSet.mdb;Persist Security Info=True"
         con.Open()
         Dim rcmd As New OleDb.OleDbCommand
          rcmd.Connection = con
           rcmd.CommandText = "select * from originalset where
        item='" & Orglist(i).Product & "'"
        Dim reader As OleDb.OleDbDataReader
        reader = rcmd.ExecuteReader
        Dim id = 0
        While reader.Read
          id = Int32.Parse(reader(0).ToString)
```

```
                End While
                reader.Close()
                Dim cmd As New System.Data.OleDb.OleDbCommand
                cmd.Connection = con
                cmd.CommandText = "update originalset set item='" &

                (Orglist(i).Product & "-temp") & "' where id=" & id & ""
                cmd.ExecuteNonQuery()
                con.Close()
                Orglist.RemoveAt(i)
                GoTo s
            End If
        Next
    s:
    Next
Next
Dim con1 As New System.Data.OleDb.OleDbConnection
con1.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=D:\SensitivePatterns\Patterns\FrequentResultSet\
ResultSet.mdb;Persist Security Info=True"
con1.Open()
Dim da As New OleDb.OleDbDataAdapter
Dim selcmd As New System.Data.OleDb.OleDbCommand
selcmd.Connection = con1
selcmd.CommandText = "select cartid,item from originalset"
selcmd.ExecuteNonQuery()
Dim ds As New DataSet
```

```
        da.SelectCommand = selcmd
        da.Fill(ds, "originalset")
        DataGridView3.DataSource = ds.Tables(0)
        MessageBox.Show("process completed")
    End If
    Dim t2 = Now
    newt1 = t2 - t1
End Sub
Dim SUSet As New List(Of NSUVal)
Dim OldSuSet As New List(Of SUVal)
Dim tsetval As New List(Of TVal)
Private Sub Button7_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button7.Click
    For Each tr In Transset
        If tr.Prds.Count >= 1 Then
            If tr.Prds.Count = 1 Then
                Dim temp As New NSUVal
                temp.TID = tr.TId
                temp.Itm = tr.Prds(0)
                temp.CID = tr.CustomerID
                temp.Support = ((Occurance(temp.Itm) / j) * 100).ToString
                SUSet.Add(temp)
            End If
            If tr.Prds.Count > 2 Then
                For i As Integer = 0 To tr.Prds.Count - 1
                    Dim temp As New NSUVal
                    temp.TID = tr.TId
```

```
                temp.Itm = tr.Prds(i)
                temp.CID = tr.CustomerID
                temp.Support = ((Occurance(temp.Itm) / j) * 100).ToString
                SUSet.Add(temp)
            Next
            For i As Integer = 0 To tr.Prds.Count - 1
                Try
                    Dim temp As New NSUVal
                    temp.TID = tr.TId
                    temp.Itm = "( " & tr.Prds(i) & "," & tr.Prds(i + 1) & " )"
                    temp.CID = tr.CustomerID
                    temp.Support = ((DOccurance(tr.Prds(i), tr.Prds(i + 1)) / j) *
                    100).ToString
                    Dim tk As New TVal    tk.AssociatedItems = " " & tr.Prds(i)
                    & " , " & tr.Prds(i + 1) &  " " tsetval.Add(tk)
                    SUSet.Add(temp)
                    i = i + 1
                Catch ex As Exception
                End Try
            Next
        End If
    End If
Next
DataGridView4.DataSource = SUSet.ToList
DataGridView9.DataSource = tsetval.ToList
End Sub
Private Function Occurance(ByVal st As String) As Integer
```

```
    Dim temp As Integer
    For Each tr In Transset
        Dim ts = From k In tr.Prds Where k.Contains(st) Select k
        If ts.Count > 0 Then
            temp = temp + 1
        End If
    Next
    Return temp
End Function
Private Function OldOccurance(ByVal st As String) As Integer
    Dim temp As Integer
    For Each tr In OldTransset
        Dim ts = From k In tr.Prds Where k.Contains(st) Select k
        If ts.Count > 0 Then
            temp = temp + 1
        End If
    Next
    Return temp
End Function
Private Function DOccurance(ByVal st As String, ByVal st1 As String) As
Integer
    Dim temp As Integer
    For Each tr In Transset
        Dim ts = From k In tr.Prds Where k.Contains(st) Select k
        Dim ts1 = From k1 In tr.Prds Where k1.Contains(st1) Select k1
        If ts.Count > 0 And ts1.Count > 0 Then
            temp = temp + 1
```

```
            End If
        Next
        Return temp
    End Function
    Private Function OldDOccurance(ByVal st As String, ByVal st1 As String)
As Integer
        Dim temp As Integer
        For Each tr In OldTransset
            Dim ts = From k In tr.Prds Where k.Contains(st) Select k
            Dim ts1 = From k1 In tr.Prds Where k1.Contains(st1) Select k1
            If ts.Count > 0 And ts1.Count > 0 Then
                temp = temp + 1
            End If
        Next
        Return temp
    End Function
    Private Function Confident(ByVal st As String, ByVal st1 As String) As
Integer
        Dim temp As Integer
        For Each tr In OldTransset
            Dim ts = From k In tr.Prds Where k.Contains(st) Select k
            Dim ts1 = From k1 In tr.Prds Where k1.Contains(st1) Select k1
            If ts.Count > 0 Or ts1.Count > 0 Then
                temp = temp + 1
            End If
        Next
        Return temp
```

```
    End Function


Public Class NSUVal
    Implements IComparable(Of NSUVal)
    Public Prds As New List(Of String)
    Dim str As String
    Dim str1 As String
    Dim str2 As String
    Dim str3 As String
    Dim str4 As String
    Public Property CID() As String
        Get
            Return str3
        End Get
        Set(ByVal value As String)
            str3 = value
        End Set
    End Property
    Public Property TID() As String
        Get
            Return str
        End Get
        Set(ByVal value As String)
            str = value
```

```
        End Set
    End Property
    Public Property Itm() As String
        Get
            Return str1
        End Get
        Set(ByVal value As String)
            str1 = value
        End Set
    End Property
    Public Property Support() As String
        Get
            Return str2
        End Get
        Set(ByVal value As String)
            str2 = value
        End Set
    End Property
    Public Function CompareTo(ByVal other As NSUVal) As Integer
Implements System.IComparable(Of NSUVal).CompareTo
        Return Me.Support.CompareTo(other.Support)
    End Function
End Class

Public Class SUVal
    Implements IComparable(Of SUVal)
    Public Prds As New List(Of String)
```

```
    Dim str As String
    Dim str1 As String
    Dim str2 As String
    Dim str3 As String
    Dim str4 As String
    Public Property CID() As String
        Get
            Return str3
        End Get
        Set(ByVal value As String)
            str3 = value
        End Set
    End Property
    Public Property TID() As String
        Get
            Return str
        End Get
        Set(ByVal value As String)
            str = value
        End Set
    End Property
    Public Property Itm() As String
        Get
            Return str1
        End Get
        Set(ByVal value As String)
            str1 = value
```

```
        End Set
    End Property

    Public Property Support() As String
        Get
            Return str2
        End Get
        Set(ByVal value As String)
            str2 = value
        End Set
    End Property
    Public Property Confidence() As String
        Get
            Return str4
        End Get
        Set(ByVal value As String)
            str4 = value
        End Set
    End Property

    Public Function CompareTo(ByVal other As SUVal) As Integer Implements
System.IComparable(Of SUVal).CompareTo
        Return Me.Support.CompareTo(other.Support)
    End Function
End Class
```

Data.vb

```
Public Class DataItem
    Implements IComparable
    Public Sub New()

    End Sub
    Public Sub New(ByVal id As Integer)
        Me.Id = id
    End Sub
    Public Sub New(ByVal id As Integer, ByVal itemName As String)
        Me.Id = id
        Me.ItemName = itemName
    End Sub
    Private m_itemName As String
    Public Property ItemName() As String
        Get
            Return m_itemName
        End Get
        Set(ByVal value As String)
            m_itemName = value
        End Set
    End Property
    Private m_id As Integer
    Public Property Id() As Integer
        Get
            Return m_id
        End Get
        Set(ByVal value As Integer)
```

```
            m_id = value
        End Set
    End Property

    Public Overloads Overrides Function Equals(ByVal obj As Object) As
Boolean
        Dim di As DataItem = DirectCast(obj, DataItem)
        Return di.Id.Equals(Me.Id)
    End Function
#Region "IComparable Members"
    Public Function CompareTo(ByVal obj As Object) As Integer Implements
System.IComparable.CompareTo
        Dim di As DataItem = DirectCast(obj, DataItem)
        Return Me.Id.CompareTo(di.Id)
    End Function
#End Region
End Class
```

Dataread.vb

```
Imports System.IO
Imports System.Text

Class CSVReader
    Public Function Read(ByVal fileName As String) As ItemSet

        Dim rowSet As New ItemSet()
        Dim colSet As ItemSet = Nothing
```

```
        Dim col As String = ""
        Dim head As String() = Nothing
        If File.Exists(fileName) Then

            ' Create a file to write to.
        Dim sr As New StreamReader(File.OpenRead(fileName),
Encoding.[Default], True)
        Dim row As String = ""
        Dim k As Integer = 0
        While Not sr.EndOfStream
          k += 1
          row = sr.ReadLine()
          Dim cols As String() = row.Split(",".ToCharArray())
          If k = 1 Then
            head = cols
          Else
            colSet = New ItemSet()
            For i As Integer = 1 To cols.Length
              col = cols(i - 1)
              If col.Equals("1") Then
                colSet.Add(New DataItem(i, head(i - 1)))
              End If
            Next
            rowSet.Add(colSet)
          End If
        End While
        sr.Close()
```

```
        End If
        Return rowSet
    End Function
End Class
```

Itemset.vb

```
Public Class ItemSet
    Inherits ArrayList
    Private m_icount As Integer = 0
    Public Property ICount() As Integer
        Get
            Return m_icount
        End Get
        Set(ByVal value As Integer)
            m_icount = value
        End Set
    End Property
    Public Overloads Overrides Function Clone() As Object
        Dim al As ArrayList = DirectCast(MyBase.Clone(), ArrayList)
        Dim [set] As New ItemSet()
        For i As Integer = 0 To al.Count - 1
            [set].Add(al(i))
        Next
        [set].ICount = Me.m_icount
        Return [set]
    End Function
    Public Overloads Overrides Function Equals(ByVal obj As Object) As
Boolean
```

```
        Dim al As ArrayList = DirectCast(obj, ArrayList)
        If al.Count <> Me.Count Then
            Return False
        End If
        For i As Integer = 0 To al.Count - 1
            If Not al(i).Equals(Me(i)) Then
                Return False
            End If
        Next
        Return True
    End Function
End Class
```

ReadData.vb

```
Public Class ReadData
    Dim ID As String
    Dim Movie As String
     Public Property CustomerId() As String
        Get
            Return ID
        End Get
        Set(ByVal value As String)
            ID = value
        End Set
    End Property
    Public Property Product() As String
        Get
```

```
        Return Movie
      End Get
      Set(ByVal value As String)
        Movie = value
      End Set
    End Property
  End Class
```

STrans.vb

```
Public Class STrans
  Implements IComparable(Of STrans)
  Dim id As String
  Dim Ct As String
  Dim cid As String
  Dim sper As String
  Public Property TId() As String
    Get
      Return id
    End Get
    Set(ByVal value As String)
      id = value
    End Set
  End Property
  Public Property Count() As String
    Get
      Return Ct
    End Get
```

```
        Set(ByVal value As String)
            Ct = value
        End Set
    End Property
    Public Property CustomerID() As String
        Get
            Return cid
        End Get
        Set(ByVal value As String)
            cid = value
        End Set
    End Property
    Public Property SPercentage() As String
        Get
            Return sper
        End Get
        Set(ByVal value As String)
            sper = value
        End Set
    End Property
    Public Function CompareTo(ByVal other As STrans) As Integer Implements
System.IComparable(Of STrans).CompareTo
        Return Me.SPercentage.CompareTo(other.SPercentage)
    End Function
End Class
```

TimeData.vb

```
Public Class TimeData
    Dim id As String
    Dim Ct As String
    Dim cid As String
    Public Prds As New List(Of String)
    Public Property Items As String
        Get
            Return id
        End Get
        Set(ByVal value As String)
            id = value
        End Set
    End Property
    Public Property OldTime As String
        Get
            Return Ct
        End Get
        Set(ByVal value As String)
            Ct = value
        End Set
    End Property
    Public Property NewTime As String
        Get
            Return cid
        End Get
        Set(ByVal value As String)
```

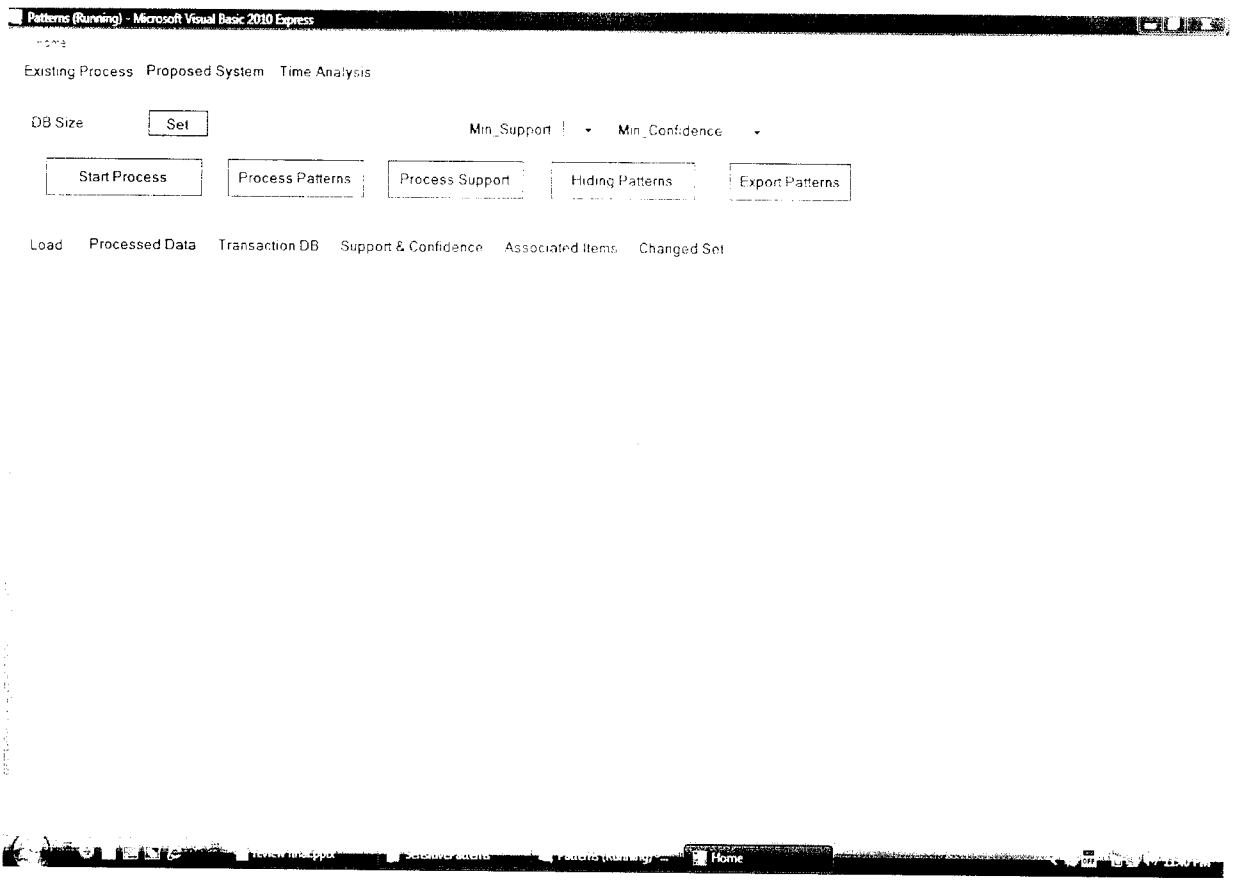
```
            cid = value
        End Set
    End Property
End Class
```

TransactionSet.vb

```
Public Class TransactionSet
    Implements IComparable(Of TransactionSet)
    Dim id As String
    Dim Ct As String
    Dim cid As String
    Public Prds As New List(Of String)
    Public Property TId() As String
        Get
            Return id
        End Get
        Set(ByVal value As String)
            id = value
        End Set
    End Property
    Public Property Count() As String
        Get
            Return Ct
        End Get
        Set(ByVal value As String)
            Ct = value
```

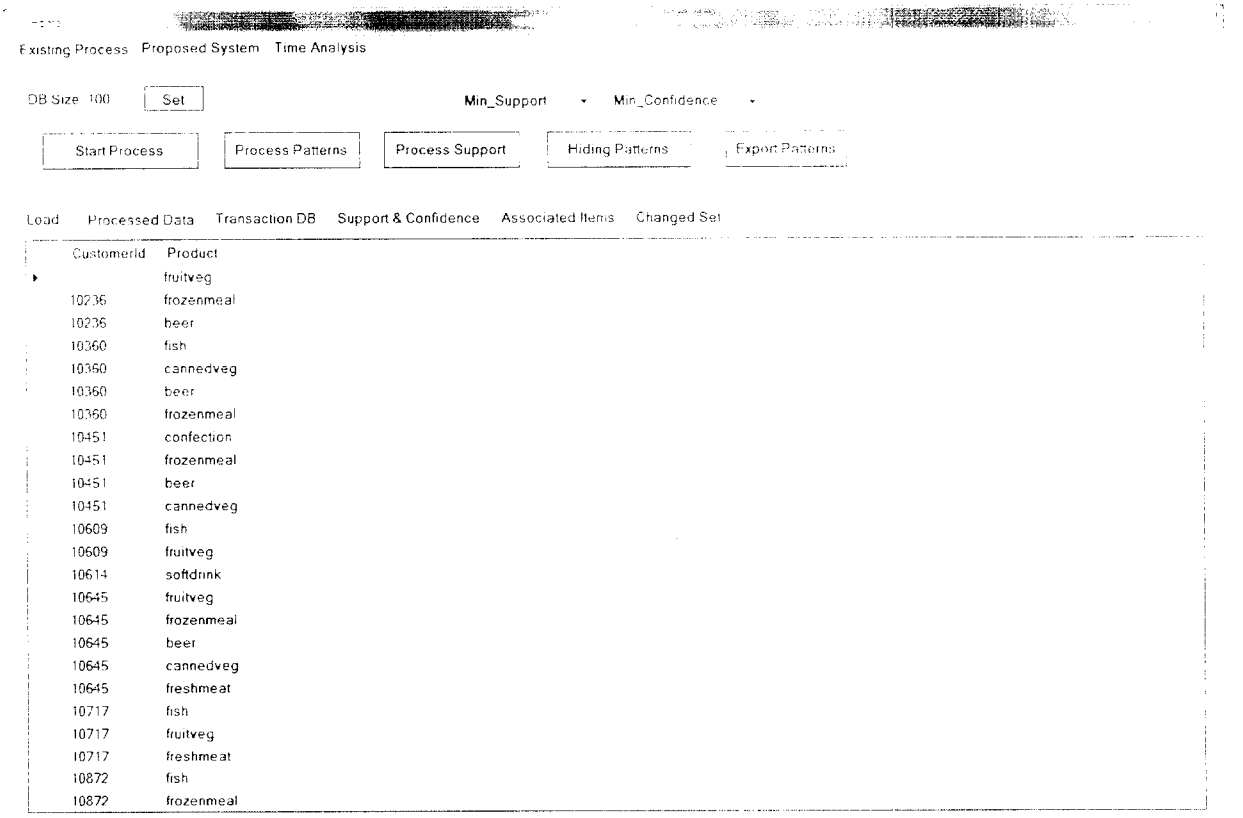
```
        End Set
    End Property
    Public Property CustomerID() As String
        Get
            Return cid
        End Get
        Set(ByVal value As String)
            cid = value
        End Set
    End Property
    Public Function CompareTo(ByVal other As TransactionSet) As Integer
Implements System.IComparable(Of TransactionSet).CompareTo
        Return Me.Count.CompareTo(other.Count)
    End Function
End Class
```

SNAP SHOTS



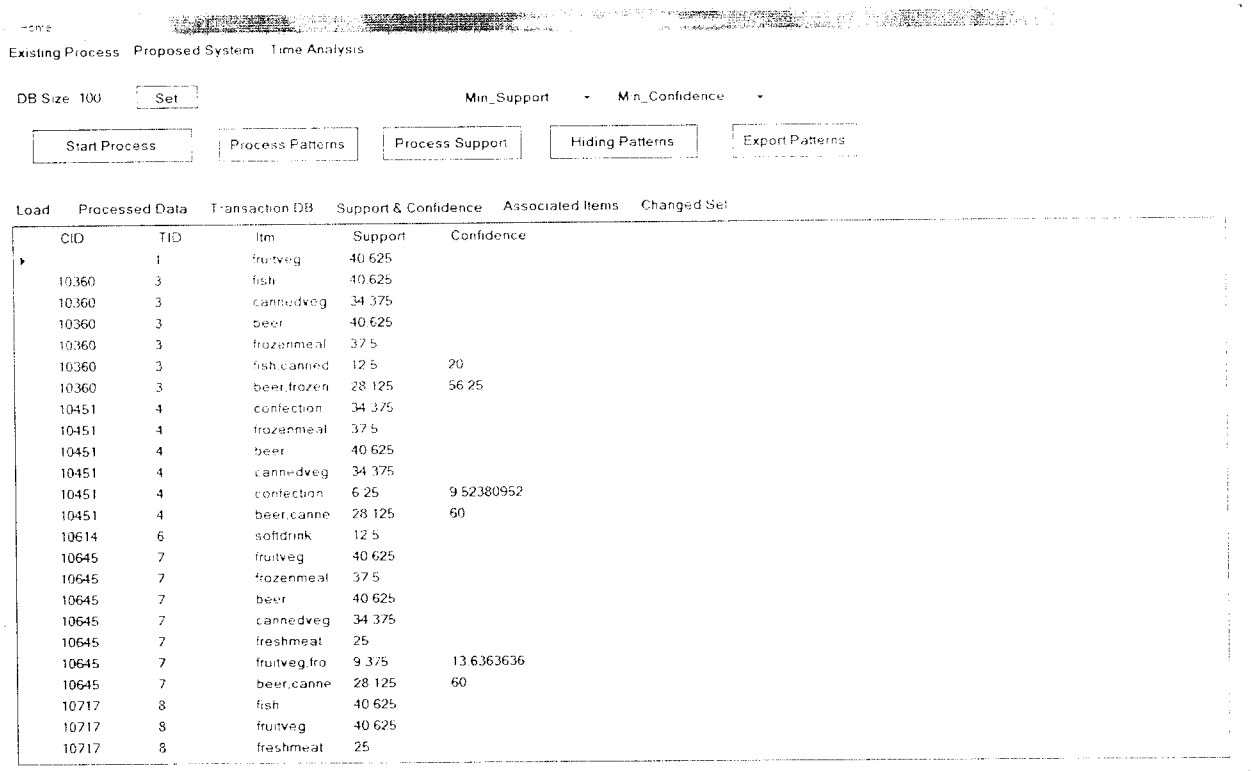
E.1.1 Start Screen

E.1.2 Data Loading



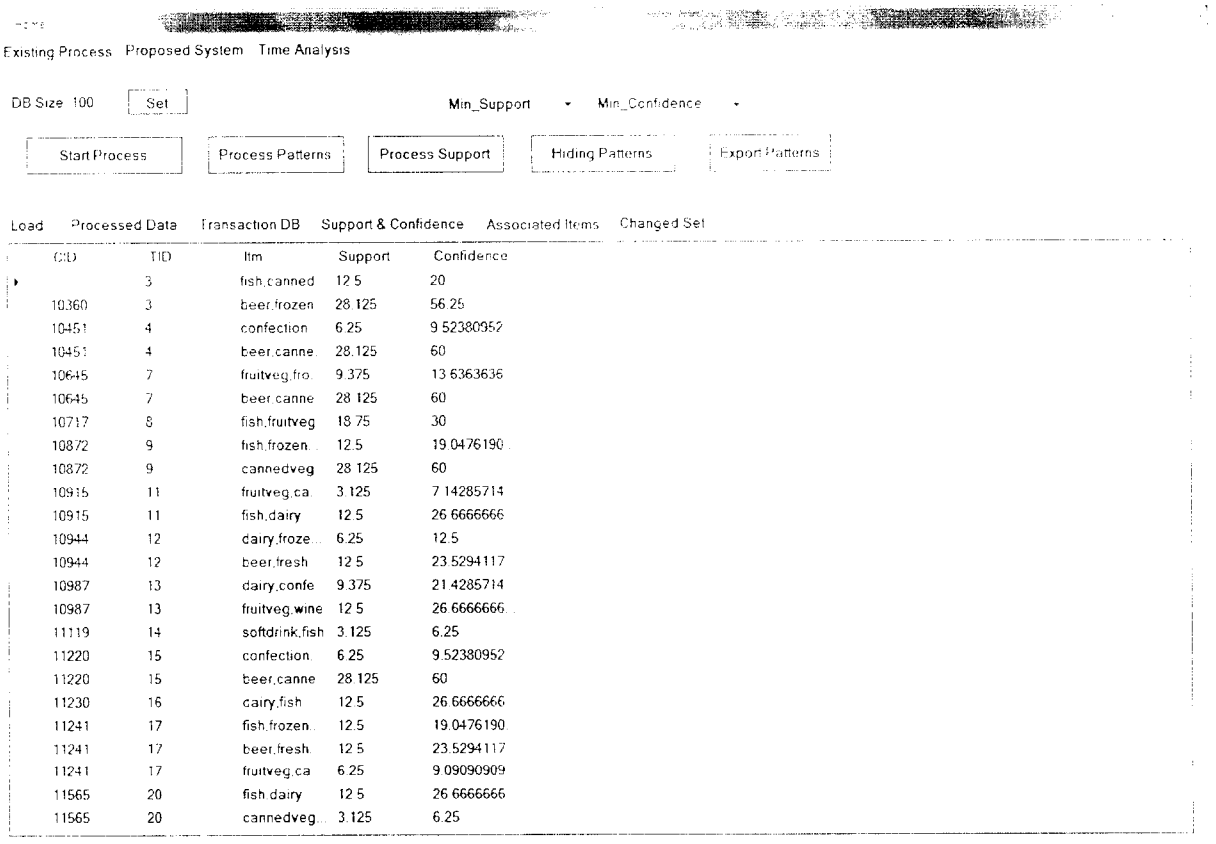
E.1.3 Grid View

E.1.4 Transaction

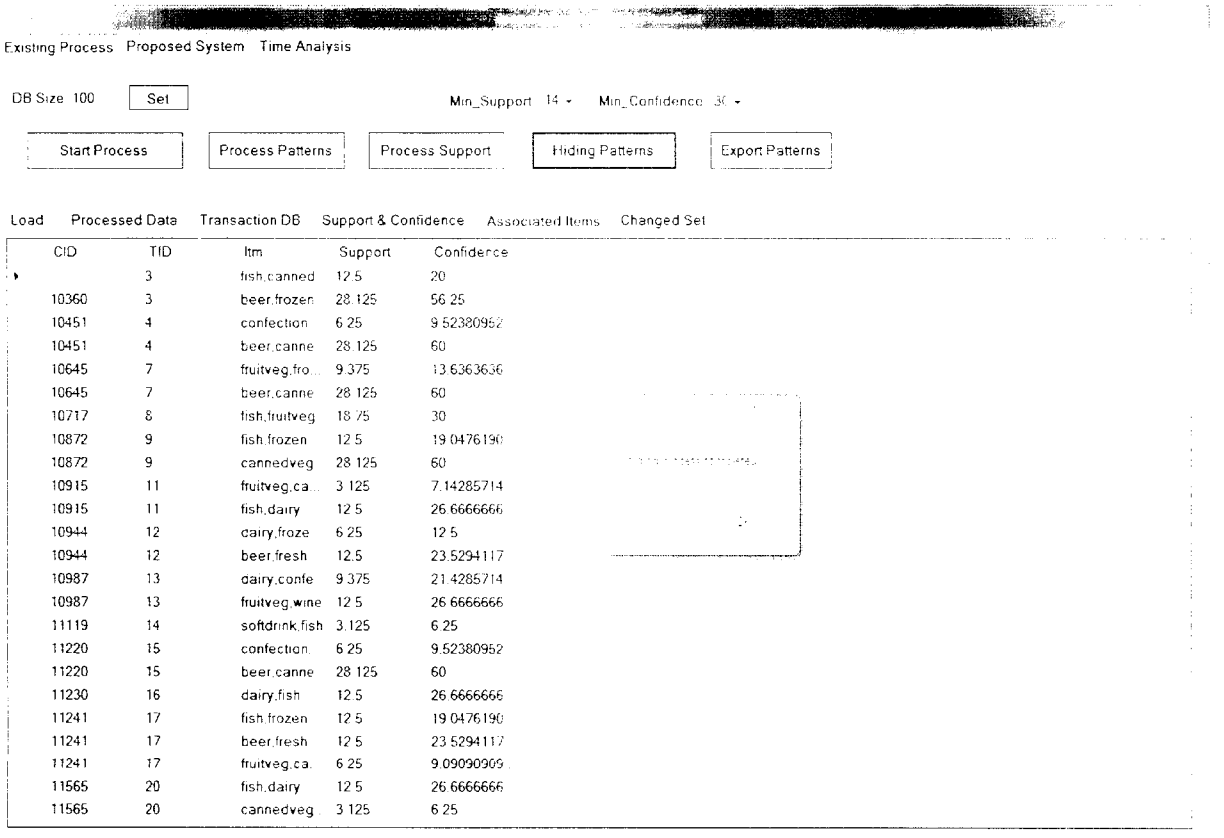


CID	TID	Itm	Support	Confidence
	1	fruitveg	40.625	
10360	3	fish	40.625	
10360	3	cannedveg	34.375	
10360	3	beer	40.625	
10360	3	frozenmeal	37.5	
10360	3	fish,canned	12.5	20
10360	3	beer,frozen	28.125	56.25
10451	4	confection	34.375	
10451	4	frozenmeal	37.5	
10451	4	beer	40.625	
10451	4	cannedveg	34.375	
10451	4	confection	6.25	9.52380952
10451	4	beer,canne	28.125	60
10614	6	softdrink	12.5	
10645	7	fruitveg	40.625	
10645	7	frozenmeal	37.5	
10645	7	beer	40.625	
10645	7	cannedveg	34.375	
10645	7	freshmeat	25	
10645	7	fruitveg,fro	9.375	13.6363636
10645	7	beer,canne	28.125	60
10717	8	fish	40.625	
10717	8	fruitveg	40.625	
10717	8	freshmeat	25	

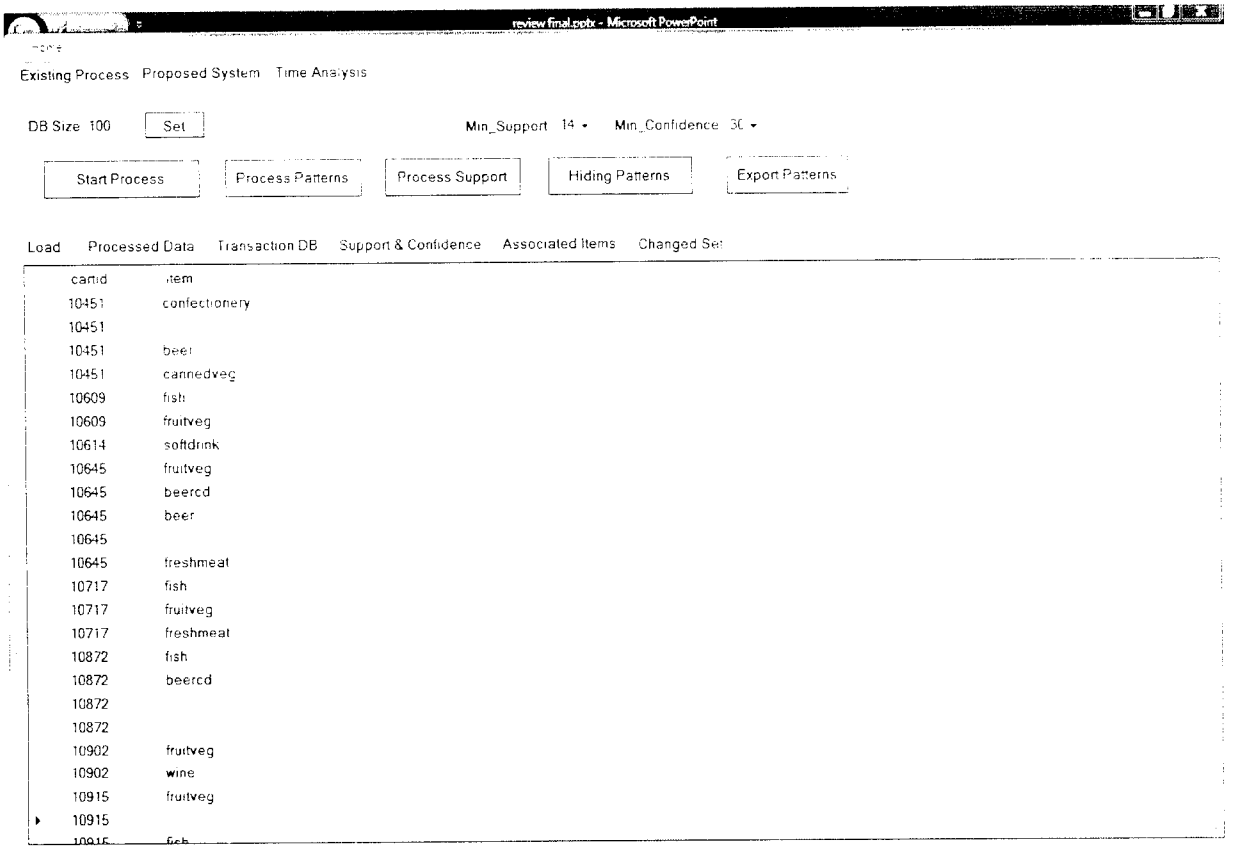
E.1.5 Calculation of Support and Confidence



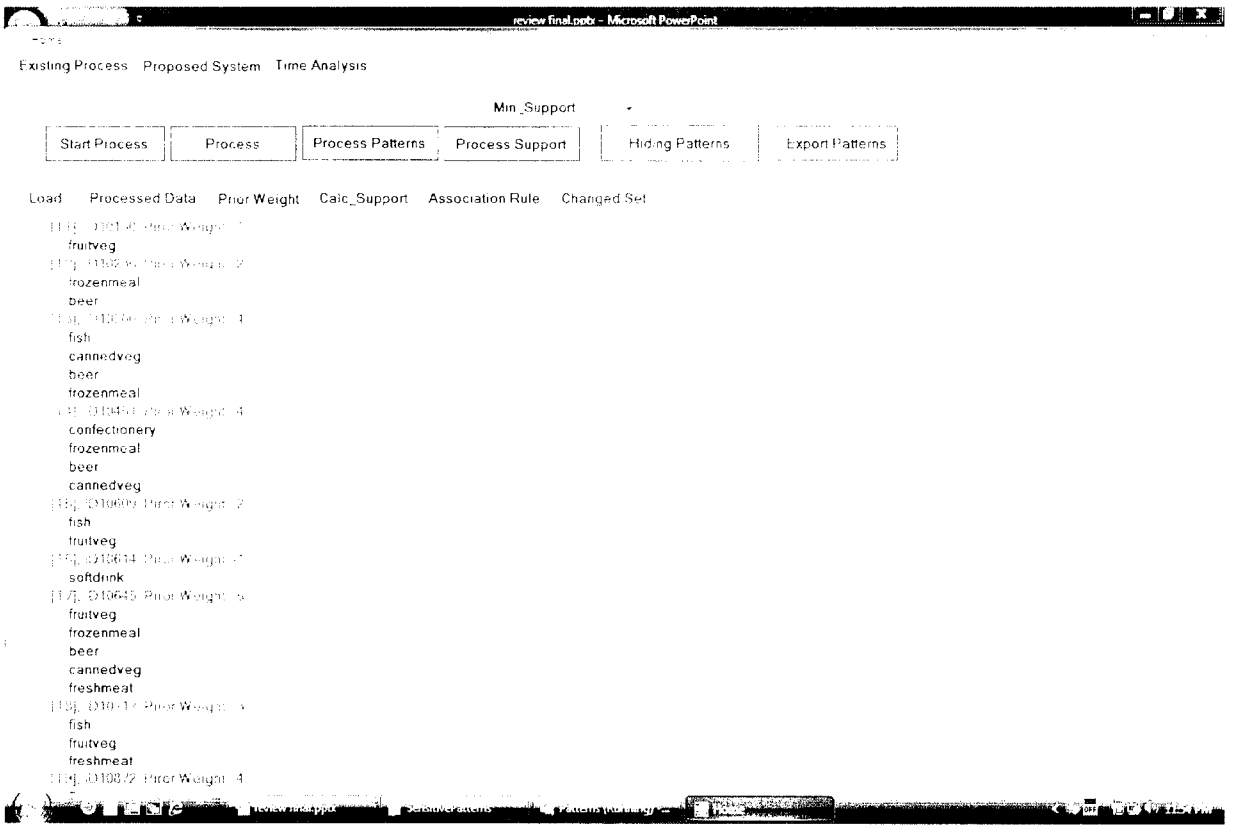
E.1.6 Associated Items



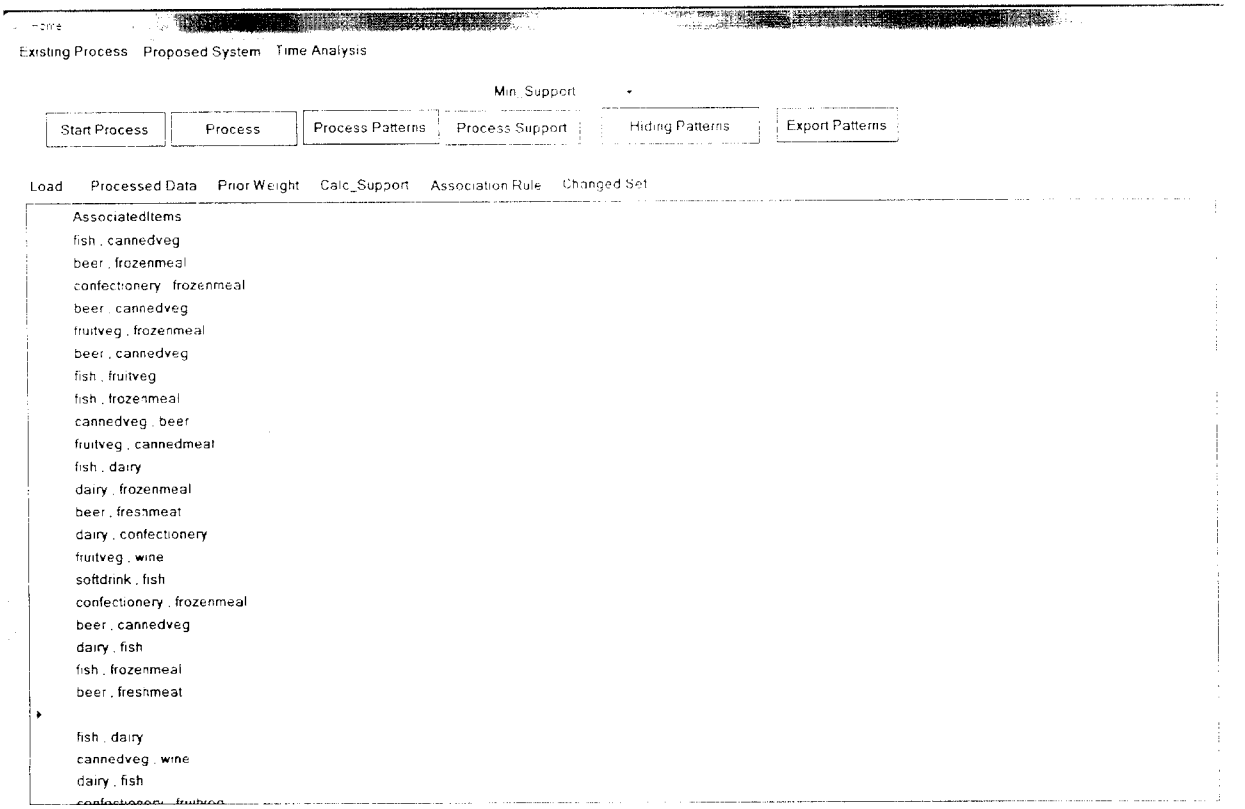
E.1.7 Hiding Process



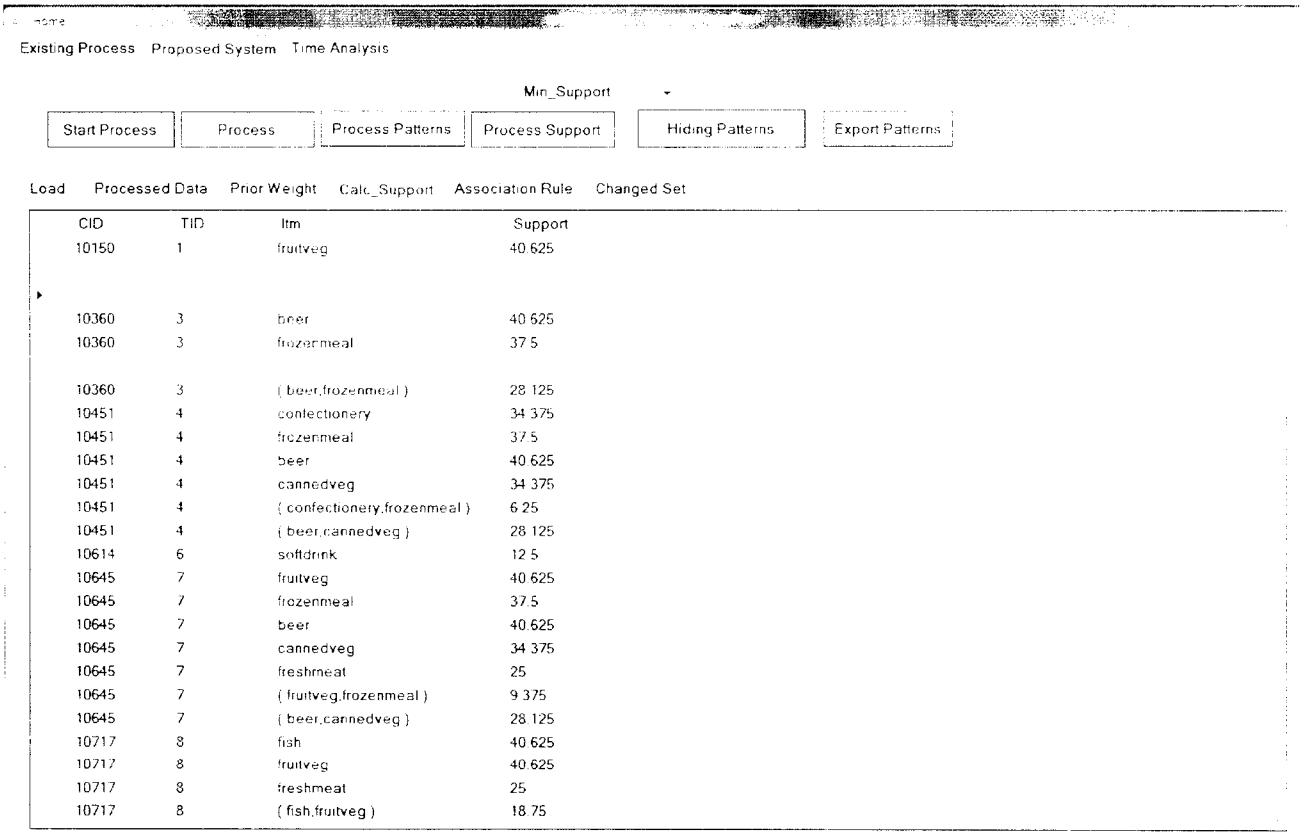
E.1.8 Modified Database



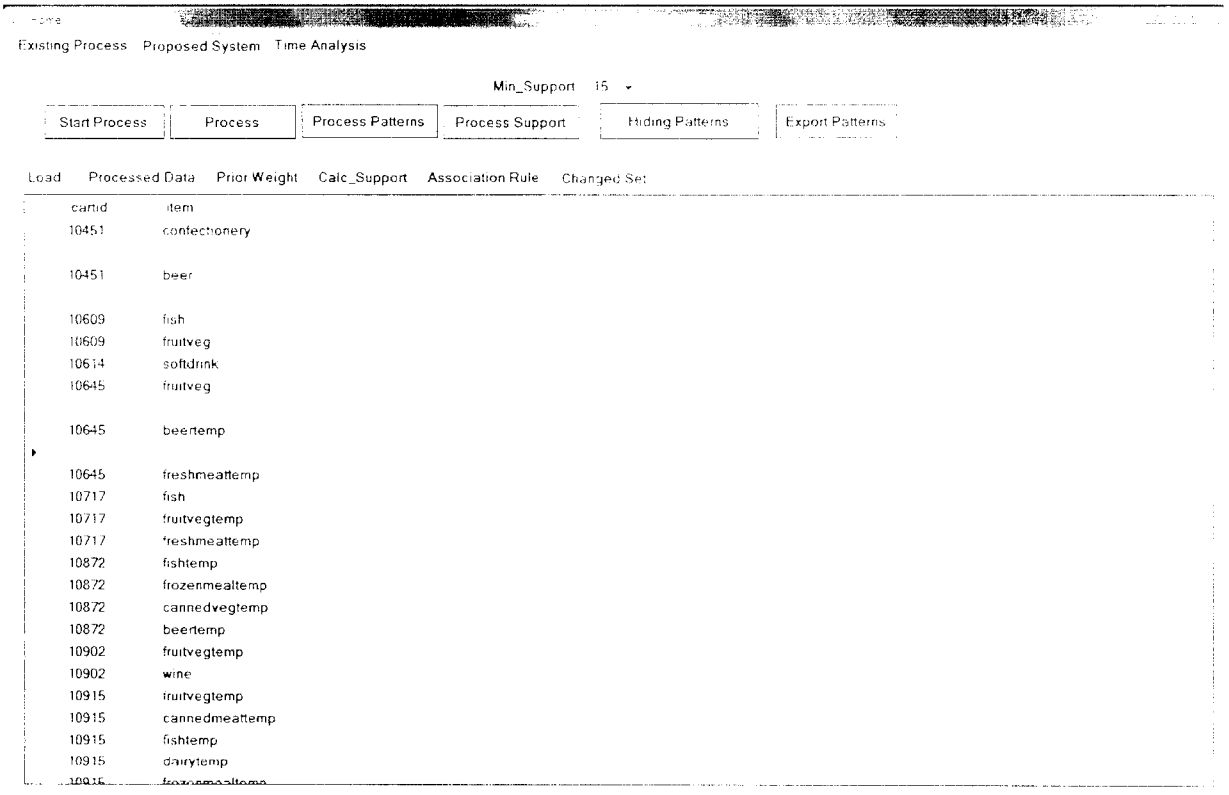
P.1.1 Transactions and prior weight



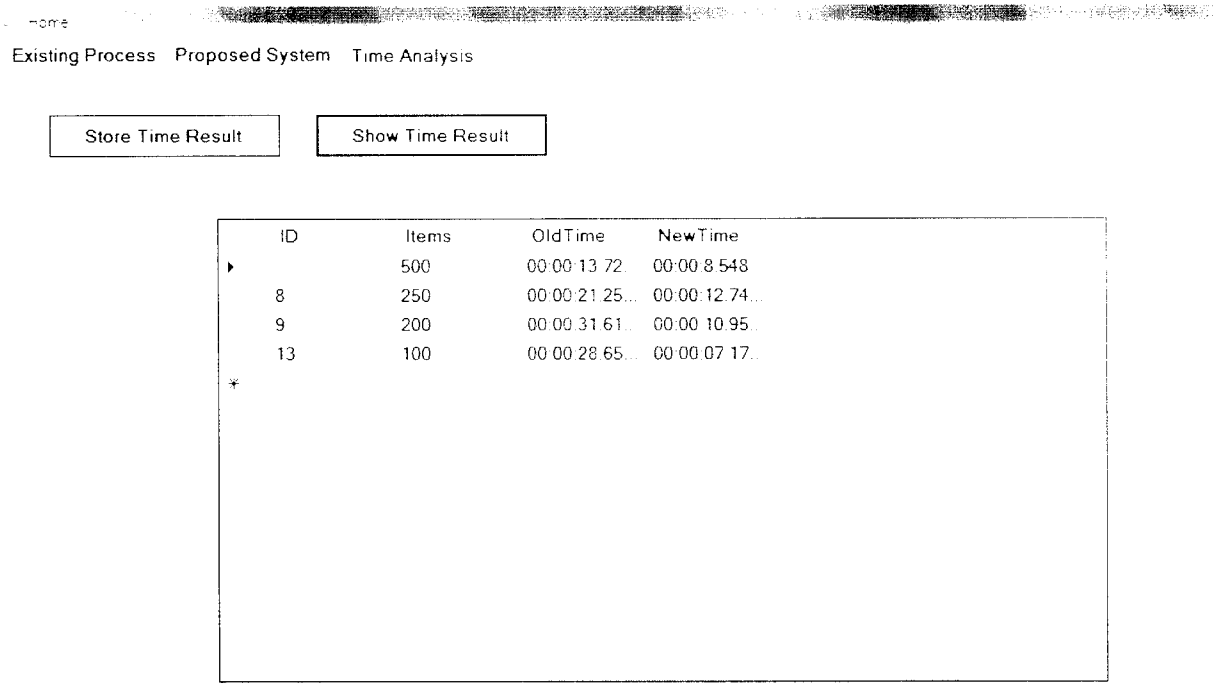
P.1.2 Associated Items Sets



P.1.3 Support calculation



P.1.4 Modified Database



P.1.5 Time Analysis Result

8. REFERENCES

1. Jiawei Han and Micheline Kamber 'Data Mining concepts and Techniques'.

2. David I.Schenider 'An Introduction to Programming using Visual Basic .NET'.

3. Shyue-Liang Wang, "Hiding sensitive predictive association rules", *Systems, Man and Cybernetics, 2005 IEEE International Conference on Information Reuse and Integration,* vol. 1, pp. 164-169, 2005.

4. Ali Amiri, "Dare to share: Protecting sensitive knowledge with data sanitization", *Decision Support Systems archive* vol. 43, issue 1, pp. 181-191, 2007.

5. Yi-Hung Wu, Chia-Ming Chiang, and Arbee L.P. Chen,"Hiding Sensitive Association Rules with Limited Side Effects", *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, issue 1, pp. 29 - 42, 2007.

6. www.google.co.in