CHAPTER-1

INTRODUCTION

1.1 Data Mining

Data Mining refers to extracting or "mining" knowledge from large amounts of data. Data mining, the extraction of hidden predictive information from large databases, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Data mining t1ools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. The automated, prospective analyses offered by data mining move beyond the analyses of past events provided by retrospective tools typical of decision support systems. Data mining tools can answer business questions that traditionally were too time consuming to resolve. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations.

Most companies already collect and refine massive quantities of data. Data mining techniques can be implemented rapidly on existing software and hardware platforms to enhance the value of existing information resources, and can be integrated with new products and systems as they are brought on-line. When implemented on high performance client/server or parallel processing computers, data mining tools can analyze massive databases to deliver answers to several typical questions.

This paper provides an introduction to the basic technologies of data mining. Examples of profitable applications illustrate its relevance to today's business environment as well as a basic description of how data warehouse architectures can evolve to deliver the value of data mining to end users.

1.2 Data Mining Techniques:

There are several major data mining techniques have been developed and used in data mining projects recently including association, classification, clustering, prediction and sequential patterns.

1. **Association:**

Association is one of the best known data mining techniques. In association, a pattern is discovered based on a relationship of a particular item on other items in the same transaction. For example, the association technique is used in market basket analysis to identify what products that customers frequently purchase together. Based on this data businesses can have corresponding marketing campaign to sell more products to make more profit.

2. Classification:

Classification is a classic data mining technique based on machine learning. Basically classification is used to classify each item in a set of data into one of predefined set of classes or groups. Classification method makes use of mathematical techniques such as decision trees, linear programming, neural network and statistics. In classification, we make the software that can learn how to classify the data items into groups. For example, we can apply classification in application that “given all past records of employees who left the company, predict which current employees are probably to leave in the future.” In this case, we divide the employee’s records into two groups that are “leave” and “stay”. And then we can ask our data mining software to classify the employees into each group.

3. Clustering:

Clustering is a data mining technique that makes meaningful or useful cluster of objects that have similar characteristics using automatic technique. Different from classification, clustering technique also defines the classes and put objects in them, while in classification objects are assigned into predefined classes. To make the concept clearer, we can take library as an example. In a library, books have a wide range of topics available. The challenge is how to keep those books in a way that readers can take several books in a specific topic without hassle. By using clustering technique, we can keep books that have some kind of similarities in one cluster or one shelf and label it with a meaningful name. If readers want to grab books in a topic, he or she would only go to that shelf instead of looking the whole library.

4. Prediction:

Prediction as its name implies is one of the data mining techniques that discover relationship between independent variables and relationship between dependent and independent variables. For instance, prediction analysis technique can be used in sale to predict profit for the future if we consider sale is an independent variable, profit could be a dependent variable. Then based on the historical sale and profit data, we can draw a fitted regression curve that is used for profit prediction.

5. **Sequential Patterns:**

Sequential patterns analysis in one of the data mining techniques that seeks to discover similar patterns in data transaction over a business period. The uncover patterns are used for further business analysis to recognize relationships among data.

1.3 Data Mining Process:

The data mining process typically involves the following:

1. Data cleaning - handling of noisy, missing or irrelevant data;
2. Data integration - integration of multiple data sources;
3. Data selection - retrieving relevant data;
4. Data transformation - reducing the size of the data-set to be examined to a minimum;
5. Data analysis - applying intelligent pattern extraction algorithms;
6. Pattern evaluation - assign measures of "interestingness" to the patterns; and
7. Presenting the extracted knowledge to the user.

Steps 1 or 4 are different forms of data preprocessing, where the data are prepared for mining. The data mining step may interact with the user or a knowledge base. The interesting patterns are presented to the user and may be stored as new knowledge base. Note that according to this view, data mining is only one step in the entire process, albeit an essential one because it uncovers hidden patterns for evaluation. We agree that data mining is a step in the knowledge discovery process. However, in industry, in media, and in the database research milieu, the term data mining is becoming more popular than the longer term of knowledge discovery from data. Therefore, in this book, we choose to use the term data mining. We adopt a broad view of data mining functionality: data mining is the process of discovering interesting knowledge from large amounts of data stored in databases, data warehouses, or other information repositories.

1.4 Apriori Algorithm:

In computer science and data mining, Apriori is a classic algorithm for learning Association Rules. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation). Apriori uses a "bottom up" Approach, where frequent subsets are extended one time at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found. The purpose of the apriori algorithm is to find association between different sets of data. It is sometimes referred to as "Market basket analysis". Each set of data has a number of items and is called a transaction. The output of apriori is sets of roles that tell us how often items are contained in sets of data.

Apriori uses Breadth-first search and a hash tree structure to count candidate item sets efficiently. It generates candidate item sets of length k from items sets of length k-1. Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent k-length item sets. After that it scans the transaction database to determine frequent item sets among the candidates. Apriori, while historically significant, suffers from a number of in efficiencies or trade-off, which have spanned other algorithms. Candidate generation generates large number of subsets(The algorithm attempts to load up the candidate sets with as many as possible before each scan). Bottom-up subset exploration(essentially a breadth-first traversal of the subset lattice) finds any maximal subset S only after all 2^{|S|}-1 of its proper subsets.

To improve the efficiency of the level-wise generation of frequent item sets, an important property called the Apriori property, presented below, is used to reduce the search space. We will first describe this property, and then show an example illustrating its use.

Apriori property: All nonempty subsets of a frequent item set must also be frequent. The Apriori property is based on the following observation. By definition, if an item set I does not satisfy the minimum support threshold, min_sup, then I is not frequent; that is, $P(I)<\text{min sup}$. If an item A is added to the item set I, then the resulting item set(i.e., I[A) cannot occur more than I. therefore, I[A is not frequent either; that is, P(I[A]<min sup. This property belongs to a special category of properties called anti monotone in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well. "How is the Apriori property used in the algorithm?" To understand this, let us look at how $L_{k-1}$is used to find L_k for k-2. A two step process is followed, consisting of join and prune actions.

1. The join step: To find $L_{k,}$ a set of candidate K-item sets is generated by joining L_{k-1} with itself. This set of candidates is denoted C_k . Let l_1 and l_2 be item sets in $L_{k-1.}$ The notation $l_i[j]$ refers to the j_{th} item in L_i,(e.g., $l_1[k-2]$ refers to the second to the last item in l_1). By convention, Apriori assumes that items within a transaction or itemset are sorted in lexicographic order. For the (k-1)-itemset, l_i this means that the items are sorted such that $l_i[1]< l_i[2]<\ldots<li[k-1]$. The join,

L_k-1 on L_k-1, is performed where members of L_k-1 are joinable if their first(K-2) items are in common. That is l_1 and l_2 of L_k-1 are joined if $(l_1[1]= l_2[1])$^$(l_1[2]= l_2[2])$^…^$(l_1[K-2]=$ l2[K-2])^$[l_1[K-1]< l_2[K-1])$. The condition $l_1[K-1]< l_2[K-1]$ simply ensures that no duplicates are generated. The resulting itemset formed by joining l_1 and l_2 is $l_1[1]$, $l_1[2]$,…, $l_1[k-2]$, $l_1[k-1]$, $l_2[k-1]$.

2. The prune step: C_k is a superset of $L_{k,}$ that is, its members may or may not be frequent, but all of the frequent k-itemsets are included in C_k. A scan of the database to determine the count of each candidate in C_k would result in the determination of L_k (i.e., all candidates having a count no less than the minimum support count are frequent by definition, and therefore belong to L_k). C_k however, can be huge, and so this could involve heavy computation. To reduce the size of C_k, the apriori property is used as follows. Any (k-1)-itemset that is not frequent cannot be a subset of frequent k-itemset. Hence, if any (k-1)-subset of a candidate k-itemset is not in L_{k-1}, then the candidate cannot be frequent either and so can be removed from C_k. This subset testing can be done quickly by maintaining a hash tree of all frequent itemsets.

1.5 Improved Apriori Algorithm

This section will address the improved Apriori ideas. The following figure shows the steps used by the improved Apriori algorithm.

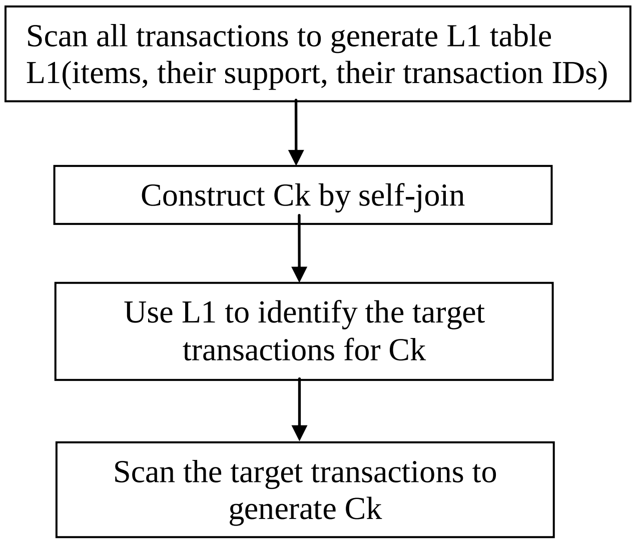

Scan all transactions to generate L1 table L1(items, their support, their transaction IDs)
Construct Ck by self-join
Use L1 to identify the target transactions for Ck
Scan the target transactions to generate Ck

Figure 1.1 Steps for C_k generation

Here instead of scanning all transactions, we use L1 to identify the target transactions for generating Ck. This is explained in the next few steps.

- In our proposed approach, we enhance the Apriori algorithm to reduce the time consuming for candidates itemset generation.
- We firstly scan all transactions to generate L1 which contains the items, their support count and Transaction ID where the items are found. And then we use L1 later as a helper to generate L2, L3 ... L_k.
- When we want to generate C2, we make a self-join L1 * L1 to construct 2-itemset C (x, y), where x and y are the items of C2. Before scanning all transaction records to count the support count of each candidate, use L1 to get the transaction IDs of the minimum support count between x and y, and thus scan for C2 only in these specific transactions.
- The same thing for C3, construct 3-itemset C (x, y, z), where x, y and z are the items of C3 and use L1 to get the transaction IDs of the minimum support count between x, y and z, then scan for C3 only in these specific transactions and repeat these steps until no new frequent itemsets are generated.

CHAPTER-2

LITERATURE SURVEY

2.1 Introduction to Apriori Algorithm

In the process of Apriori, the following definitions are needed:

Definition 1: Suppose D={T1, T2, … , Tm},(m>=1) is a set of transactions, Ti= {I1, I2, … , In},(n>=1) is the set of items, and k-itemset = {i1, i2, … , ik},(k>=1) is also the set of k items, and k-itemset ⊆I.

Definition 2: Suppose Sup.Count (itemset), is the support count of itemset or the frequency of occurrence of an itemset in transactions.

Definition 3: Suppose Ck is the candidate itemset of size k, and Lk is the frequent itemset of size k.

Definition 4: Suppose min_sup is the minimum support threshold value for pruning the itemsets.

The algorithm uses the following property to find all sets of frequent item sets.

Apriori property All nonempty subsets of frequent item set must also be frequent.

An Example

Consider the database D containing the following 9 transactions

Item1	Item2	Item3	Item4
1	2	5	
2	4		
2	3		
1	2	4	
1	3		
2	3		
1	3		
1	2	3	5
1	2	3	

Table 2.1 Data base

- Suppose minimum support count is 2.
- We have to find out the frequent item set using Apriori algorithm.

Step 1: Generating frequent 1-itemsets.

Item	Count
1	6
2	7
3	6
4	2
5	2

No. of searches
9
9
9
9
9

Table 2.2 Frequent 1-itemset

Deleting item sets whose Sup.Count<min_sup:

- The set of frequent 1-itemsets, L1 consists of the candidate 1-itemsets satisfying minimum support.
- Here all the items have their support count greater than the min_sup. Hence none of the items are deleted.
- The column no. of searches in the table is used so that we can compare the number of searches required to find the count of the itemsets with the number of searches required for the improved apriori algorithm.

Step 2: Generating frequent 2-itesmsets

Item1	Item2	Count
1	2	4
1	3	4
1	5	2
2	3	4
2	4	2
2	5	2

No. of searches
9
9
9
9
9
9

Table 2.3 Frequent 2-itemsets

Deleting item sets whose Sup.Count<min_sup:

- To discover the set of frequent 2-itemsets, L2 the algorithm uses L1 join L1 to generate a candidate set of 2-itemsets, C2.
- The set of frequent 2-itemsets, L2 is then determined, consisting of those itemsets having minimum support.

Step 3: Generating frequent 3-itemsets:

Item1	Item2	Item3	Count
1	2	3	2
1	2	5	2

No. of searches
9
9

Table 2.4 Enumerated frequent itemsets

Deleting itemsets whose Sup.Count <min_sup:

- The generation of the set of candidate 3-itemsets, C3, involves use of the Apriori Property.
- In order to find C3, we compute L2 join L2.
- C3=L2 join L2.
- Now, Join step is complete and Prune step will be used to reduce the size of C3. Prune step helps to avoid heavy computation due to large C_k.
- Based on the Apriori property that all subsets of a frequent item set must also be frequent, we can determine that four candidates cannot possibly be frequent.
- The algorithm uses Ln-1 join Ln-1 to generate a candidate set of 4-itemsets, Cn.
- Thus Cn=0, and algorithm terminates, having found all of the frequent items. This completes our Apriori Algorithm.

2.2 Introduction to Improved Apriori Algorithm:

An example

Here we consider the previous example of Apriori algorithm.

T_ID	Item1	Item2	Item3	Item4
T1	1	2	5	
T2	2	4		
T3	2	3		
T4	1	2	4	
T5	1	3		
T6	2	3		
T7	1	3		
T8	1	2	3	5
T9	1	2	3	

Table 2.5 Database

- The table stores the transaction ids along with the items.
- Assume minimum support count as 2.

Firstly, scan all transactions to get frequent 1-itemset, and then eliminate the candidates that are infrequent or their support are less than the min_sup to form L1.

Step 1: Generating frequent 1-itemsets

Item	Count	T_ID
1	6	T1,T4,T5,T7,T8,T9
2	7	T1,T2,T3,T4,T6,T8,T9
3	6	T3,T5,T6,T7,T8,T9
4	2	T2,T4
5	2	T1,T8

No. of searches
9
9
9
9
9

Table 2.6 Frequent 1-itemsets

- Here the step 1 is same for both the algorithms.
- The L1 table is further used to get the support count of the items.

Step 2: Generating frequent 2-itemsets:

Item1	Item2	Count
1	2	4
1	3	4
1	5	2
2	3	4
2	4	2
2	5	2

No. of searches	
6	(T1,T4,T5, T7, T8, T9)
6	(T1,T4,T5, T7,T8,T9)
2	(T1,T8)
6	(T3,T5,T6, T7,T8,T9)
2	(T2,T4)
2	(T1,T8)

Table 2.7 Frequent 2-itemsets

- The next step is to generate candidate 2-itemset from L1.
- To get support count for every itemset, split each itemset in 2-itemset into two elements then use L1 table to determine the transactions where you can find the itemset in, rather than searching for them in all transactions.
- For example, let’s take the first item (I1, I2), in the original Apriori we scan all 9 transactions to find the item (I1, I2); but in our improved algorithm we will split the item (I1, I2) into I1 and I2 and get the minimum support between them using L1.
- Here I1 has the smallest minimum support. After that we search for itemset (I1, I2) only in the transactions T1, T4, T5, T7, T8 and T9 to find the count of (I1, I2).
- After pruning the itemsets whose support count is less than the min_sup the frequent 2-itemset table is constructed.

Step 3: Generating frequent 3-itemsets:

Item1	Item2	Item3	Count
1	2	3	2
1	2	5	2

No. of searches
6 (T1,T4,T5,T7,T8,T9)
2 (T1,T8)

Table 2.8 Frequent 3-itemsets

- The generation of 3-itemsets involves use of Apriori Property similar to Apriori algorithm.
- Since C4=0, the algorithm terminates having found all frequent itemsets.

Comparison between Apriori algorithm and Improved Apriori algorithm:

- We can differentiate between both the algorithms by observing the no. of searches column for all the tables in both the algorithms.
- The improved Apriori uses less number of scans to find the count of all the itemsets thereby reducing the time.

2.3 Description of Front End

2.3.1 Java Framework

Introduction

Choosing appropriate tool for creating multimedia is the first step in multimedia design and production. Various tools that are used by educators, designers and programmers include Visual Basic, Java, Flash, Dreamweaver (standalone and web based applications)

Significance

Java is a powerful programming language. It can be used to create complex standalone applications or small components which can be used over the network. But its potential seems to be unexploited in the field of education. This paper discusses some significant features of Java along with a discussion of why Java not so popular among educators.

A. What is Java?

Java: A simple, object-oriented, network-savvy, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, dynamic language. Platform independence made java very useful for the Internet developers.

B. What makes Java programs portable, secure, and robust?

1. Portability

Java programs are portable across operating systems and hardware environments.

Portability is to your advantage because:

- You need only one version of your software to serve a broad market.
- The Internet, in effect, becomes one giant, dynamic library.
- You are no longer limited by your particular computer platform.

Three features make Java String programs portable:

- **The language**. The Java language is completely specified; all data-type sizes and formats are defined as part of the language. By contrast, C/C++ leaves these "details" up to the compiler implementer, and many C/C++ programs therefore are not portable.
- **The library**. The Java class library is available on any machine with a Java runtime system, because a portable program is of no use if you cannot use the same class library on every platform. Window-manager function calls in a Mac application written in C/C++, for example, do not port well to a PC.
- **The byte code**. The Java runtime system does not compile your source code directly into machine language, an inflexible and non portable representation of your program. Instead, Java programs are translated into machine-independent byte code. The byte code is easily interpreted and therefore can be executed on any platform having a Java runtime system. (The latest versions of the Netscape Navigator browser, for example, can run applets on virtually any platform).

2. Security

The Java language is secure in that it is very difficult to write incorrect code or viruses that can corrupt/steal your data, or harm hardware such as hard disks.

There are two main lines of defense:

- Interpreter level:
- No pointer arithmetic
- Garbage collection
- Array bounds checking
- No illegal data conversions
- Browser level (applies to applets only):
- No local file I/O
- Sockets back to host only
- No calls to native methods

3. Robustness

The Java language is robust. It has several features designed to avoid crashes during program execution, including:

- No pointer arithmetic Garbage collection--no bad addresses
- Array and string bounds checking
- No jumping to bad method addresses
- Interfaces and exceptions

2.3.2 Java Programming Structure

A file containing Java source code is considered a compilation unit. Such a compilation unit contains a set of classes and, optionally, a package definition to group related classes together. Classes contain data and method members that specify the state and behavior of the objects in your program. Java programs come in two flavors:

• Standalone applications that have no initial context such as a pre-existing main window

2.3.2.1 Java virtual machine:

- Compact JVM interprets byte code, does GC
- Written in C, has OS/platform layer
- Uses threads
- Has just-in-time compilation
- JVM has simple instruction set
- Byte code register based, byte order independent
- Easy to generate machine code

2.3.3 Java Class Library:

The Java Platform is not dependent on a specific operating system, applications cannot rely on any of the platform native libraries. Instead, the Java Platform provides a comprehensive set of standard class libraries, containing the functions common to modern operating systems.

JCL serves three purposes within the Java Platform:

- Like other standard code libraries, they provide the programmer a well-known set of useful facilities, such as container classes and regular expression processing.
- The library provides an abstract interface to tasks that would normally depend heavily on the hardware and operating system, such as network access and file access.

Some underlying platforms may not support all of the features a Java application expects. In these cases, the library implementation can either emulate those features or provide a consistent way to check for the presence of a specific feature.

JCL is almost entirely written in Java, except for the parts that need direct access to the hardware and operating system (such as for I/O, or bitmap graphics). The classes that give access to these functions commonly use Java Native Interface wrappers to access operating system APIs.

Almost all of JCL is stored in a single Java archive file called "rt.jar", which is provided with JRE and JDK distributions. The Java Class Library (rt.jar) is located in the default bootstrap class path, and does not have to appear in the class path declared for the application.

1. Some Advantages

Run compiled code across platforms.

- Managed memory (garbage collector).
- Huge wealth of excellent open-source libraries.
- Large developing market
- Easy migration for C++ developers.

2. Some Disadvantages

- Aging language has not kept up with language advances IMO.
- Future uncertain after Oracle acquisition (will become clearer with time).
- Low level programming difficult.

Java Documentation

- Available for browsing on the Web or download to your own computer
- Demos, tutorials, guides to topics
- Description of Java tools

2.4 Description of Back End

2.4.1 Database

Microsoft Office Access 2007 provides a powerful set of tools that you to quickly start tracking, reporting, and sharing information in a manageable environment with its new, interactive design capabilities, prebuilt library of tracking application templates and ability to work with data from many data source, including Microsoft SQL server, Office Access 2007 allows you to rapidly create attractive and functional tracking applications without requiring deep database knowledge. You can quickly create and adapt applications and reports to changing business need and with its new, enhanced deep integration with Microsoft windows Share point Services3.0, Office Access 2007 helps you share, manage, audit, and back up information.

Office Access 2007 includes a suite of prebuilt tracking application in the form of templates that you can use to get started quickly, use them right out-of-the-box or enhance and refine them to track information your way. you can employ new views and layouts, enhanced sorting and filtering, rich text, multi valued fields ,split forms, and a host of new features to create richer, better tracking applications and effectively share tracked information with other.

The new user interface in Office Access 2007 comprises a number of elements that define hoe you interact with the product. These new elements were chose to help you master Access, and to help you find the commands that you need faster. The new design also makes it easy to discover features that otherwise might have remained hidden beneath layers of toolbars and menus. And you will get up and running faster, thanks to the new Getting Started with Microsoft Office Access page, which provides you with quickly access to our getting started experience, including a suite of professionally designed templates.

The most significant new interface element is called the ribbon. This is the strip across the top of the program window that contains group of commands. The ribbon provides a single home for commands and is the primary replacement for menus and toolbars. Within the ribbon, there are tabs that combine commands in ways that make sense. In Office Access 2007, the main ribbon tabs are Home, create, External data, and Database Tools. Each tab contains groups of related commands, and these groups surface some of the additional new UI elements, such as the gallery, which is new type of control that presents choices visually.

2.5 **Tools used (rational rose)**

The application's method recommends the use of static and dynamic views of a logical model and a physical model to capture the in-process products of object-oriented analysis and design. Using the notation, the application enables you to create and the views with in an overall model representing your problem domain and software.

This overall model contains classes, use cases, objects, packages, and also contains operations, component packages, components, devices and the relationships between them. Each of these model elements possesses model properties that identify and characterize them. The notation provides graphical icons to represent each kind of model elements and relationship.

A model also contains diagrams and specification, which provide a mean of visualizing and manipulating the model's elements and their model properties. Since diagrams are used to illustrate multiple views of model, icon representing a model element can appear in none, one or several of a model's diagrams. The application therefore enable you to control which element, relationship and property icons appear on each diagram, using facilities provided by its application window. With its application window, it displays each diagram in a diagram window, and each specification in a specification window.

CHAPTER-3

DESIGN

System design serves to bridge the gap between the requirements specified for this system and the final solution for satisfying the requirements. While the requirement specification activity is entirely the problem domain, the system design is the first step in moving from the problem domain towards the solution domain. The goal of the design process is to produce a model or a representation of a system that can be used later to build that a system. The produced model is called the design of the system. The design of the system is essentially blue print, or a plan for a solution for the system.

Software design sits at the technical kernel of the software engineering and is applied regardless of the development paradigm that is used. Without a design we risk building an unstable system, on that will fail when small changes are made, one that is difficult to test, and one whose quality cannot be assessed until late in the process of software engineering. System design involves architectural and detailed design of the system. Architectural design involves identifying software components, decomposing them in to processing modules and conceptual data structures and specifying inter connection among components.

Detailed design is concerned with how to package processing modules, how to implement the processing algorithms, data structures and interconnections among modules. It involves adaptation of existing code, modification of standard algorithms, invention of new algorithms, design of data representations and packaging of software product.

3.1 Software requirements specifications

A Software Requirements Specification (SRS) is a document that clearly and precisely specifies each and every requirement for the software product as well as the external interfaces to hardware and firmware. Each requirement should be defined so that it can be verified by a method such as inspection, demonstration, analysis and testing. There a number of desirable properties that a SRS should possess. In particular, the requirements documents should be:

- Correct
- Complete

- Consistent
- Functional
- Verifiable
- Traceable
- Easily Changed

Hardware requirements specifications:

• Processor	:	Intel Pentium
• Memory	:	512MB (or above) RAM
• Hard Disk	:	1.0 GB

Software requirement specifications:

• Front End	:	Java
• Back End	:	Microsoft Access or later
• Operating System	:	Windows XP/7

3.2 Unified modeling language (UML)

What is UML?

- Is a **language.** It is not simply a notation for drawing diagrams, but a complete language for capturing knowledge (semantics) about a subject and expressing knowledge (syntax) regarding the subject for the purpose of communication.
- Applies to **modeling** and systems .Modeling involves a focus on understanding a subject (system) and capturing and being able to communicate in this knowledge.
- It is the result of **unifying** the information systems and technology industry's best engineering practices (principles, techniques, methods and tools).
- Used for both database and software modeling
- The UML is a language for visualizing, specifying, constructing, documenting.

Conceptual model of the UML

To understand the UML, you need to form a conceptual model of the language, and this requires learning three major elements.

Elements:

A. Basic building blocks
B. Rules
C. Common Mechanisms

A. Basic Building Blocks of the UML

The vocabulary of the UML encompasses three kinds of building blocks.

i. Things
ii. Relation ships
iii. Diagram

i. Things in the UML
There are four kinds of things in the UML

1. **Structural** these are the nouns and statics parts of the model. These represent elements that are conceptual or physical.

   There are seven kinds of structural things:

   1. Class
   2. Interface
   3. Collaboration
   4. Use Case
   5. Active Class
   6. Component
   7. Node

2. Behavioral These are the Dynamic parts of the model. The verbs represent behavior over time and space. There are two kinds of behavioral things: Interaction and state machine.

3. Grouping These are organizational parts of UML models. These are boxes into which models can be decomposed. There is only one kind of grouping thing, the package.

4. Annotational These are explanatory parts of UML model. Used to describe, illuminate, remark any element of a model. There is only one kind of annotational thing, the note.

ii. **Relationships** These relationships tie things together. It is a semantic connection among elements. These relationships are basic relational building blocks of the UML.

There are four kinds of relationships:

1. **Dependency** is a semantic relationship between two things, in which a change to one thing (the independent thing) may affect the semantics of the other thing (The dependent thing).
2. **Association** is structural relationship that describe a set of links, a link being a connection among objects
3. **Generalization** is a specialization / relationship in which objects of the specialized element (The child) are more specific than the objects of the generalized element.
4. **Realization** is semantic relationship between two elements, where in one element guarantees to carry out what is expected by the other element.

iii. Diagrams: A diagram is a graphical representation of a set of elements. Represented by a connected graph, vertices are things, arcs are behaviors.

UML includes 9 diagrams:

1. **Class diagrams** describes the static structure of a system, are how it is structured rather than how it is behaves at last diagram shows the existence of classes and their relationships in the logical view of a system.
2. **Object diagrams** shows a set of objects and their relationships. Object diagrams describe the static structure of a system at a particular time. Where as a class mode describes all possible situations, an object model describes a particular situation.
3. **Use case diagrams** Use Case Diagrams describe the functionality of a system and users of the system.
4. **Sequence Diagrams** describes interactions among classes. These interactions are modeled as exchanges of messages, these diagrams focuses on classes and messages. They exchange to accomplish some desired behavior. Sequence diagrams are a type of interaction diagrams.
5. **Collaboration Diagram** Collaboration diagrams describes interactions among classes and associations. Collaboration diagrams are a type of interaction diagram.
6. **State chart diagram** (or state) diagrams describes the states and responses of a class. State chart diagram describe the behavior of a class in response to external stimuli.

7. **Activity diagram** describes the activities of a class. These diagrams are similar to state chart diagrams and use similar conventions, but activity diagrams describes the behavior of a class in response to internal processing rather than external events as in state chart diagram.
8. **Component diagram** describes the organizations and dependencies among software implementation components. These diagrams contain components, which represent distributable physical events; including source code, object code, and executable code. These are static implementation view of a system.
9. **Deployment diagram** describes the configuration of runtime processing resource elements and the mapping of software implementation component on to them. These diagrams contain components and nodes, which represents processing or computational resources including computer, printers, etc…

B. Rules of the UML:

- The UML building blocks cannot simply put together in a random fashion.
- Like any language, the UML has a number of Rules. That specify what a well formed should look like.

Well-Formed Models is a model semantically self-consistent and in harmony with all its related models.

Semantic Rules for**:**

Names- What you can call things.

Scope- Context that gives meaning to a name.

Visibility- How names can be seen and used.

Integrity- How things properly and consistently relate to one another.

Execution- What it means to run or simulate a dynamic model.

Avoid Models:

Elided- Certain elements are hidden for simplicity.

Incomplete- Certain elements may be missing.

Inconsistent- No guarantee of integrity.

C. Common Mechanisms:

The UML is made simpler by the presence of some features called common mechanisms. There are four kinds of mechanisms;

I. **Specifications** It provides a textual statement of the syntax and semantics of the building blocks.

II. **Adornments** It provides detail from an element specifications added to its basic graphical notation.

III. **Common divisions** There is a division of class and object and also the division of interface and implementation.

IV. **Extensibility Mechanisms** the UML provides a standard language for software blueprints.

The UML is opened, making it possible to extend the language in controlled ways to Express all possible nuances of all models across all time.

3.2.1. Use Case diagram

Use-Case diagrams graphically depict system behavior (Use Cases). These diagrams present a high level view of how the system is used as viewed from an outsider's (Actor's) perspective. A Use-Case diagram may depict all or some of the Use Cases of a system.

A Use-Case diagram can contain:

- Actors("Things" outside the system)
- Use Cases (System boundaries identifying what the system should do).
- Interactions or relationships between Actors and Use Cases in the system including the Associations, Dependencies, and Generalizations.

Use-Case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use Use-Case diagrams to specify the behavior of the system as implemented.

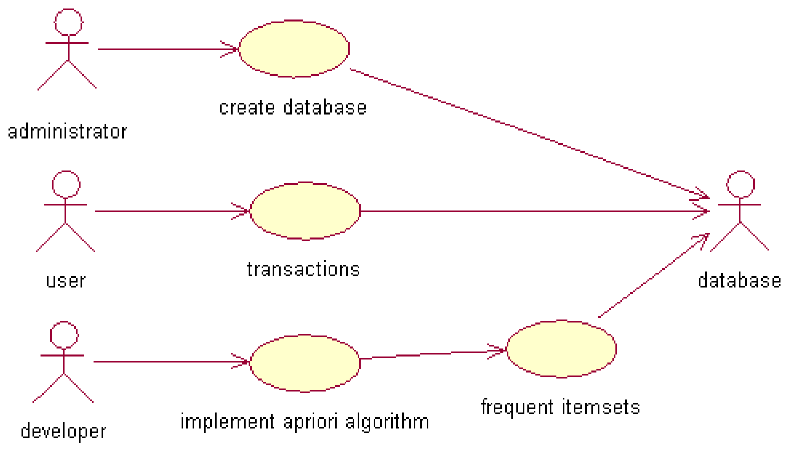


Figure 3.1 Use Case Diagram

3.2.2 Sequence Diagram

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence ¾ what happens first, what happens next. Sequence diagrams establish the roles and objects and help provide essential information to determine class responsibilities and interfaces. This type of diagrams is best used during early analysis phases in design because they are simple and easy to comprehend. Sequence diagrams are normally associated with Use Cases.

The following tools located on the sequence diagram Toolbox enable you to model sequence diagrams:

- Object
- Message Icon
- Focus of control
- Message to self
- Note
- Note Anchor

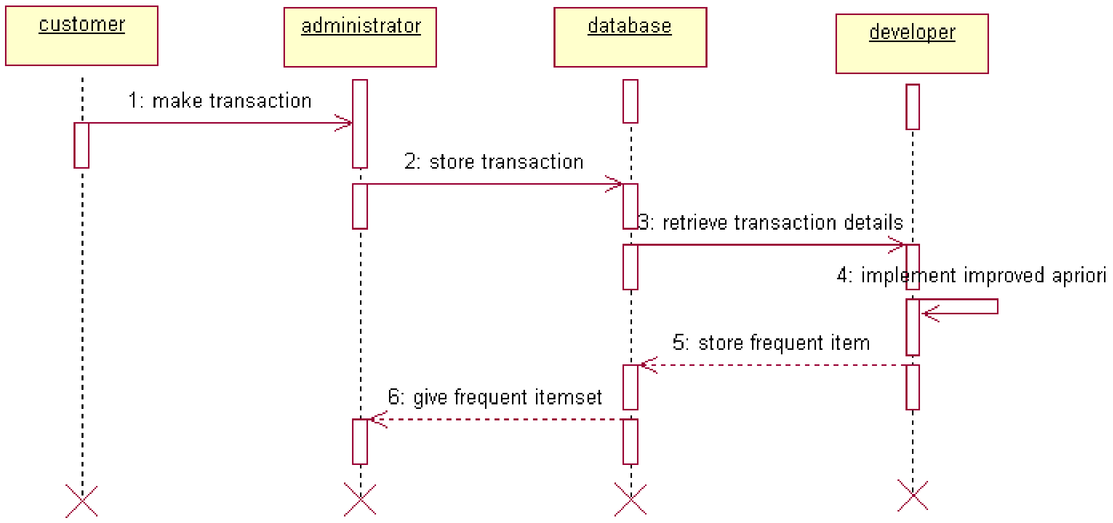


Figure 3.2 Sequence Diagram

3.2.3. Collaboration Diagram.

Collaboration diagrams and sequence diagrams are alternate representations of an interaction. A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a Transaction. A sequence diagram shows object interaction in a Time-based sequence.

Collaboration diagram show object, their links, and their message. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of the interactions or structural relationships that occur between objects and object-like entities in the current model.

An object specification enables you to display and modify the properties and relationships of an object. The information in a specification is presented textually. Some of this information can also be displayed inside the icons representing objects in collaboration diagrams.

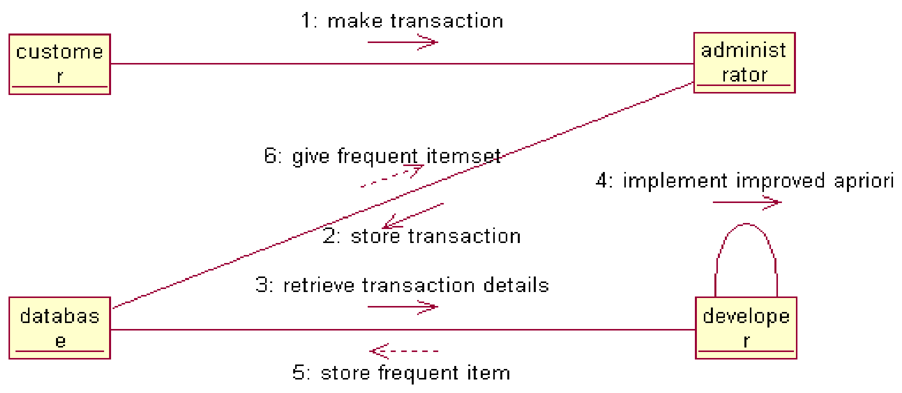


Figure 3.3 Collaboration Diagram

3.2.4 Class Diagram:

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematics of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling.[1] The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed.

A **class diagram** is a type of static structure diagram that describes the structure of a system by showing the classes, their attributes, operations and the relationships among objects.

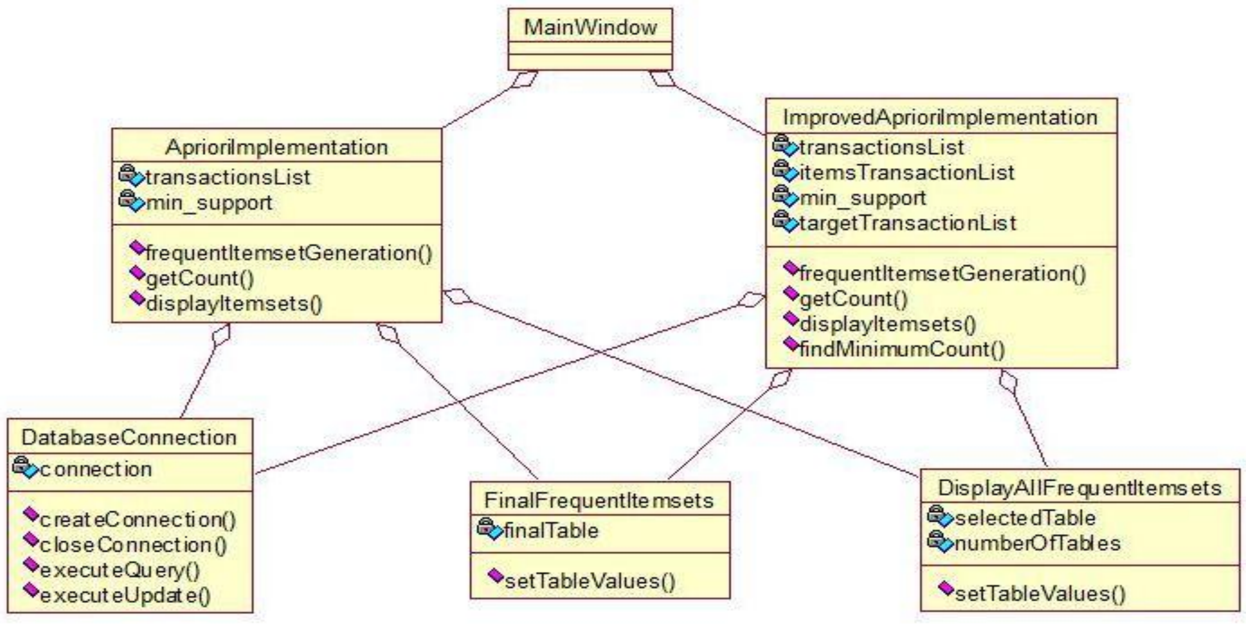


Figure 3.4 Class Diagram

3.3 Data Flow Diagrams:

A **Data Flow Diagram** (**DFD**) is a graphical representation of the "flow" of data through an information system, modelling its *process* aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel.

3.3.2 Low level DFD:

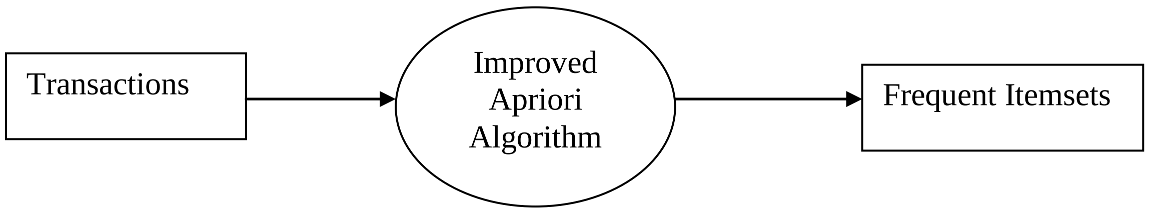


Figure 3.5 Low level DFD for Improved Apriori Algorithm

3.3.1 High Level DFD:

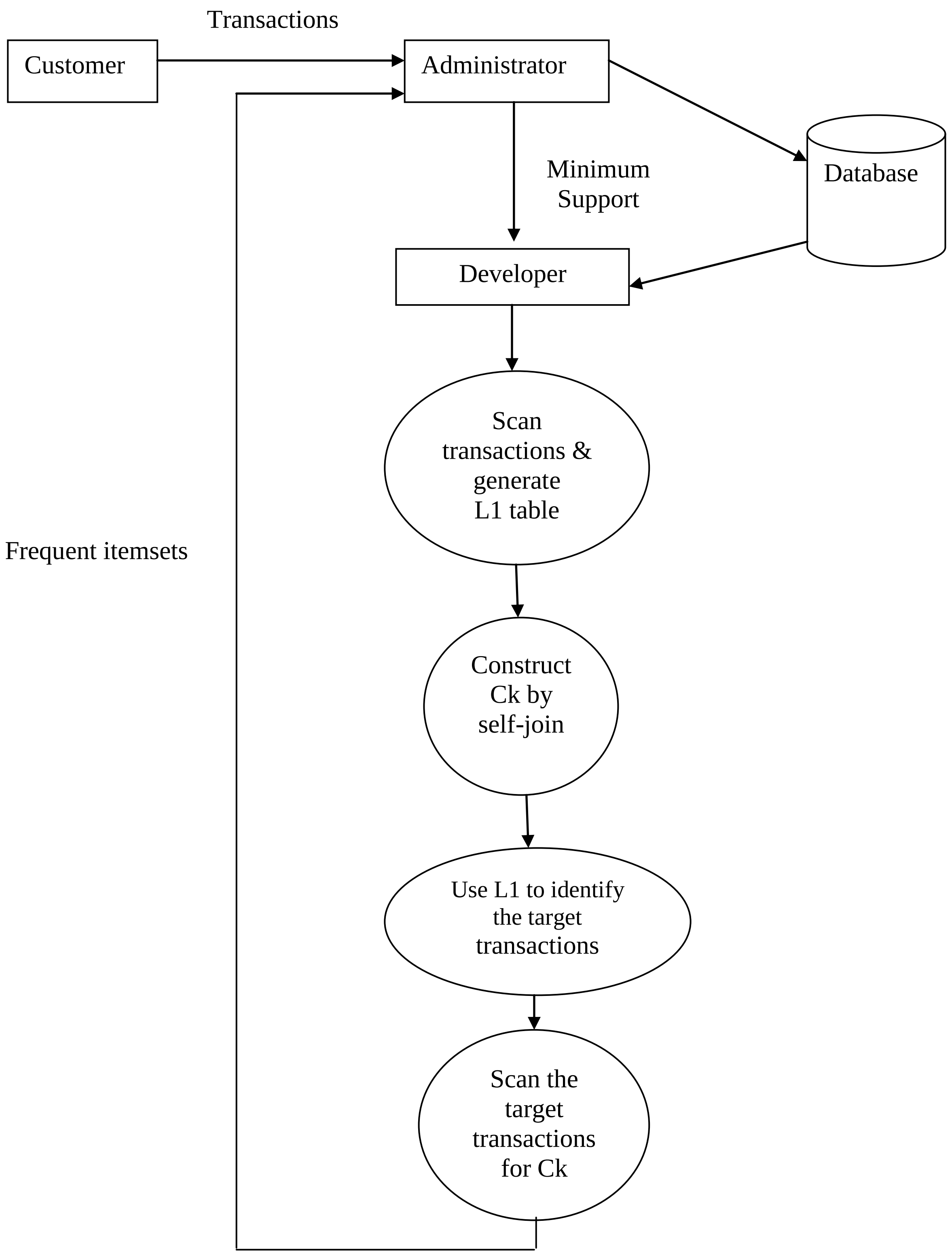

Transactions
Customer
Administrator
Database
Minimum Support
Developer
Scan transactions & generate L1 table
Frequent itemsets
Construct Ck by self-join
Use L1 to identify the target transactions
Scan the target transactions for Ck

Figure 3.6 High Level DFD

CHAPTER-4

IMPLEMENTATION

4.1 Apriori Algorithm

The Apriori algorithm is a seminal algorithm for mining frequent itemsets. It explores the level-wise mining Apriori property that all nonempty subsets of a frequent item set must also be frequent. At the k_{th} iteration (for k>=2), it forms frequent k-item set candidates based on the frequent (k-1)-itemsets and scans the database once to find the complete set of frequent k-itemsets, L_k.

Apriori Algorithm Pseudo code

```
Procedure Apriori (T, min Support)
{
//Generate items, items support, their transaction ID
(1)L1= {frequent 1-items};
(2)For (k=2; Lk-1!=O; k++) {
//Generate the Ck from the Lk-1
(3)Ck= candidates generated from Lk-1 by self join of Lk-1
(4)For each transaction T in database do {
(5)Increment the count of all candidates in Ck that are counted in t
(6)Lk= candidates in Ck with min_sup }
(7)}
(8)End;}
```

The Apriori algorithm has 2 steps

i. The join step
ii. The prune step

It finds the frequent sets L in Database D.

- Find frequent set L_{k-1}.
- Join Step.
    - C_k is generated by joining $L_{k-1}$with itself
- Prune Step.
    - Any $(k - 1)$ -itemset that is not frequent cannot be a subset of a frequent k -itemset, hence should be removed where
- (C_k Candidate itemset of size k)
- (L_k frequent itemset of size k)

4.2 Improved Apriori Algorithm

The improvement of algorithm can be described as follows:

- Firstly, scan all transactions to get frequent 1-itemset L1 which contains the items and their support count and the transactions ids that contain these items, and then eliminate the candidates that are infrequent or their support are less than the min_sup.
- The next step is to generate candidate 2-itemset from L1. To get support count for every itemset, split each itemset in 2-itemset into two elements then use l1 table to determine the transactions where you can find the itemset in, rather than searching for them in all transactions.
- The same procedure is to be followed to generate n-itemset depending on L1 table until no more frequent itemsets are generated.

Improved Apriori Algorithm Pseudo code

```
Procedure Improved_ Apriori (T, min Support)

//Generate items, items support, their transaction ID
(1) L1 = find_frequent_1_itemsets (T);
(2) For (k = 2; $L_{k-1}$! =0; k++) {
//Generate the $C_k$ from the $L_{k-1}$
(3) $C_k$ = candidates generated from $L_{k-1}$ by self join of $L_{k-1}$.
//get the item $I_w$ with minimum support in $C_k$ using L1, (1<=w<=k).
(4) x = Get _item_min_sup($C_k$, $L_1$);
// get the target transaction IDs that contain item x.
(5) Tgt = get_Transaction_ID(x);
(6) For each transaction t in Tgt Do {
(7) Increment the count of all items in Ck that are found in Tgt;
(8) Lk= items in Ck _ min_support;
(9)}
(10) End; }
```

CHAPTER-5

TESTING

Software testing is one of the most important phases in the software development activity. Testing is a critical element of software quality and represents the ultimate reviews of specification, design and coding. Testing presents an interesting anomaly for the software. Testing is vital to the success of the system. Errors can be injected at any stage during development. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved.

During testing, the program to be tested is executed with set of test data and the output of the program for the test data is evaluated to determine if the programs are performing as expected. A series of testing are performed for the proposed system before the system is ready for user acceptance testing.

Here the system has met the user's requirement in the following fields:

- Data Entry
- Error Handling
- Reporting and corrections
- Data Access Protections
- System Output

5.1 A Sample Testing Cycle

Although testing varies between organizations, there is a cycle to testing:

1. **Requirements Analysis** Testing should begin in the requirements phase of the software development life cycle. During the design phase, testers work with developers in determining what aspects of a design are testable and under what parameter those test work.
2. **Test Planning** Test Strategy, Test Plans(s) and Test Bed creation.
3. **Test Development** Test Procedures, Test Scenarios, Test cases and Test Script to use in testing software.
4. **Test Execution** Testers execute the software based on the plans and tests and report any errors found to the development team.

5. **Test Reporting** Once testing is completed, testers generate metrics and make final reports on their test effort and whether or not the software tested is ready for release.
6. **Retesting the defects** Not all errors or defects reported must be fixed by a software development team. Some may be caused by errors in configuring the test software to match the development or production environment. Some defects can be handled by a Workaround in the production environment. Others might be deferred to future releases of the software, or the deficiency might be accepted by the business user. There are yet other defects that may be rejected by the development team (of course, with due reason) if they deem it inappropriate to be called a defect.

5.2 Test Cases

In software engineering, the most common definition of a test case is a set of conditions or variables under which a tester will determine if a requirement or use case upon an application is partially or fully satisfied. In order to fully test that all the requirements of an application are met, there must be at least one test case for each requirement unless a requirement has sub requirements.

In that situation, each sub requirement must have at least one test case. This is frequently done using a Traceability matrix. Some methodologies, like RUP, recommend creating at least two test cases for each requirement. One of them should perform positive testing of requirement and other should perform negative testing. Written test cases should include a description of the functionality to be tested, and the preparation required to ensure that the test can be conducted.

Under special circumstances, there could be a need to run the test, produce results, and then a team of experts would evaluate if the results can be considered as a pass. This happens often on new products performance number determination. The first test is taken as the base line for subsequent test/ product release cycles.

5.3 Types of Tests

5.3.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functionality properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected result.

5.3.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

5.3.3 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional testing is centered on the following items:

- Valid Input: identified classes of valid input must be accepted.
- Invalid Input: identified classes of invalid input must be rejected.
- Functions: identified functions must be exercised.
- Output: identified classes of application outputs must be exercised.
- Systems/Procedures: interfacing systems or procedures must be invoked.

Organizations and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing.

5.3.4 System testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

5.3.5 White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner working, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

5.3.6 Black Box Testing

Black Box Testing is testing the software without having any knowledge of inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

5.4 Test cases

5.4.1 Test cases for Apriori Algorithm:

The Apriori algorithm is implemented by perform Apriori algorithm button as mentioned in previous section. During implementation each and every module (function) is tested separately to yield better results.

There are three databases:

- Transaction1 containing 9 transactions
- Transaction2 containing 100 transactions
- Transaction3 containing 1000 transactions

The database (Transaction1) is as follows:

Item1	Item2	Item3	Item4
1	2	5	
2	4		
2	3		
1	2	4	
1	3		
2	3		
1	3		
1	2	3	5
1	2	3	

Suppose minimum support is 2. Then by implementing the Apriori algorithm (or the improved apriori algorithm) the result is as follows:

- Frequent 1-itemset we get is {1, 2, 3, 4, 5} are added to L1.
- All the items have their support count greater than or equal to 2.
- Frequent 2-itemsets we get is {{1,2},{1,3},{1,5},{2,3},{2,4},{2,5}}. The itemsets pruned are {{1,4},{3,4},{3,5},{4,5}} whose support count is less than 2.
- L3=L2*L2 should yield the following {{1,2,3},{1,2,5},{1,3,5},{2,3,4},{2,3,5},{2,4,5}} itemsets.
- After pruning L3 contains only {{1,2,3},{1,2,5}} because all the other items does not satisfy the Apriori Property.

Some other limitations provided for both the algorithms are:

- The algorithm terminates when L_n is empty.
- The algorithm cannot be executed if the minimum support count is not an integer.
- The algorithm cannot be executed if the minimum support count is greater than the number of items in the database.

CHAPTER-6

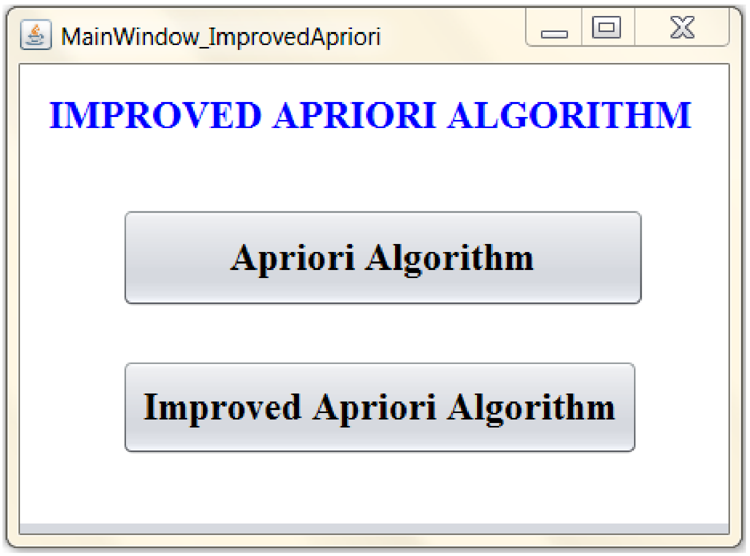
RESULTS

We show the result of our project in two steps.

- In first step we will show the results of Apriori algorithm along with the time of execution.
- In second step we will show the results of improved Apriori algorithm with its time of execution.
- Then we will compare the time of execution for both the algorithms.

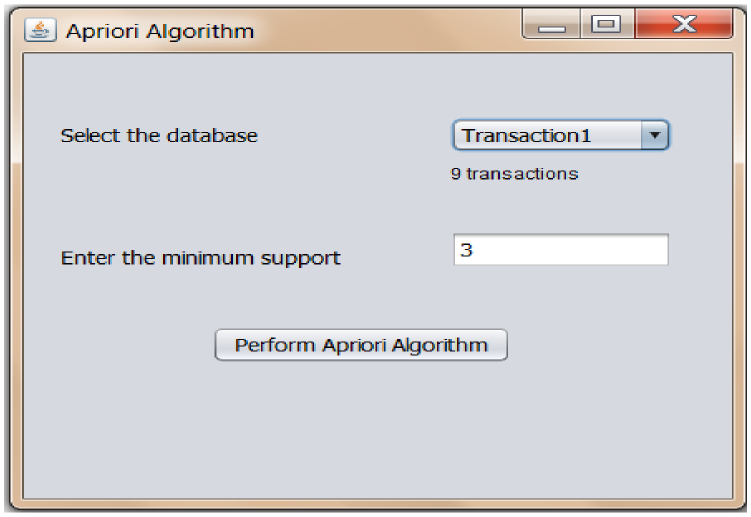
6.1 Results of Apriori Algorithm:

The main window:


MainWindow_ImprovedApriori
IMPROVED APRIORI ALGORITHM
Apriori Algorithm
Improved Apriori Algorithm

Here we should select any of the algorithms to execute it.

Showing Apriori algorithm form:



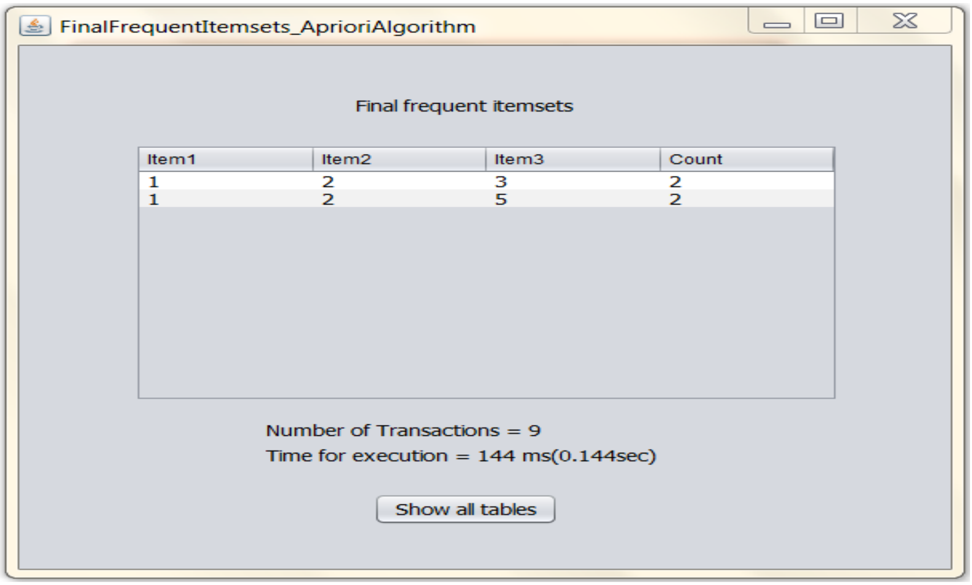
Select the database: Transcription1 → Transaction1

9 transactions

Enter the minimum support: 3

Here we should select any of the transaction databases from the Combo box. The combo box contains 3 databases namely Transaction1(9 transactions), Transaction2(100 transactions), Transaction3(1000 transactions) as mentioned before. The label below the combo box displays the number of transactions present in the selected database.

Showing output of Apriori algorithm:



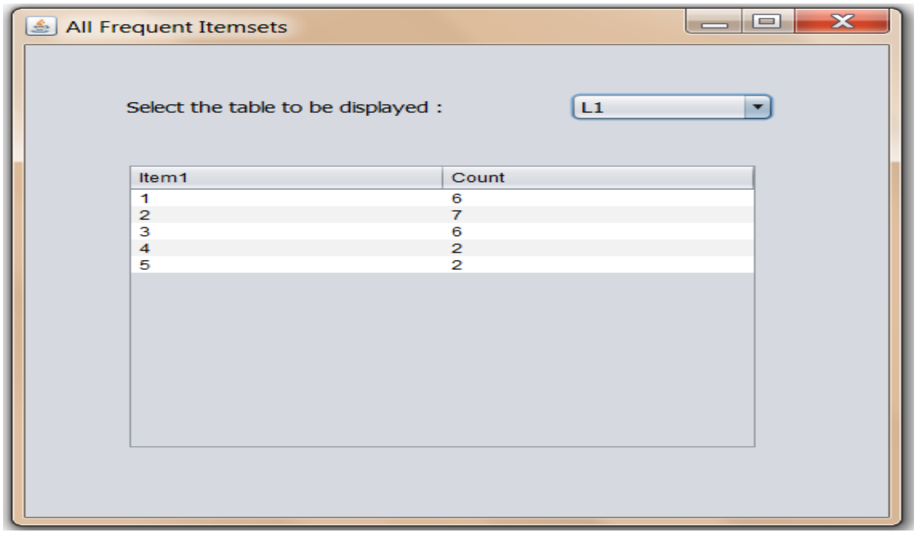
Final frequent itemsets

Item1	Item2	Item3	Count
1	2	3	2
1	2	5	2

Number of Transactions = 9
Time for execution = 144 ms(0.144sec)

This form displays the final frequent itemsets generated for the Transaction1 database with minimum support =2.

Showing the form which displays all the generated frequent itemsets:

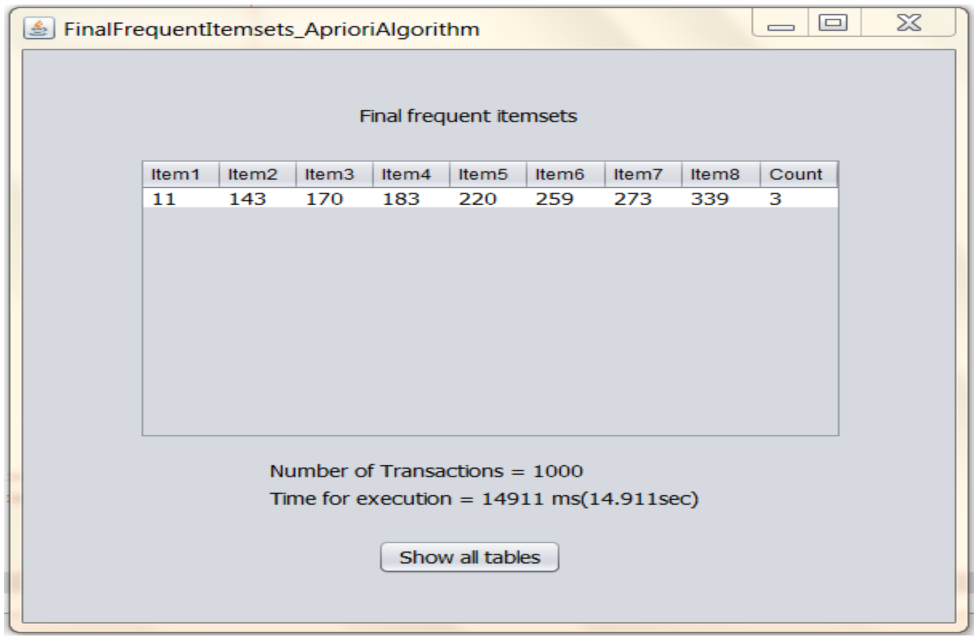


Select the table to be displayed : L1

Item1	Count
1	6
2	7
3	6
4	2
5	2

This form is used to display all the generated frequent itemsets. The combo box contains all the table names and the selected table will be displayed. The above table displays the L1 table of Transaction2 with minimum support=2.

Showing the output of Apriori for Transaction3:



Final frequent itemsets

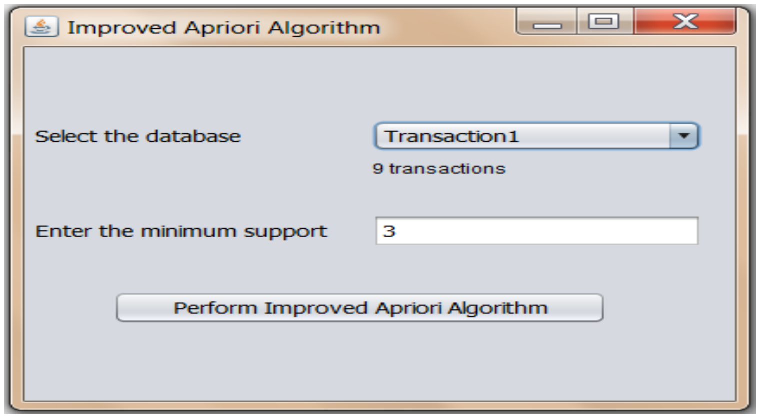
Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8	Count
11	143	170	183	220	259	273	339	3

Number of Transactions = 1000
Time for execution = 14911 ms(14.911sec)

The above form displays the output of Transaction3 database with minimum support count as 3. The itemsets are generated up to frequent 8-itemsets. The time taken for execution is approximately 14 sec.

6.2Results of Improved Apriori Algorithm:

Showing the main form of improved Apriori algorithm:

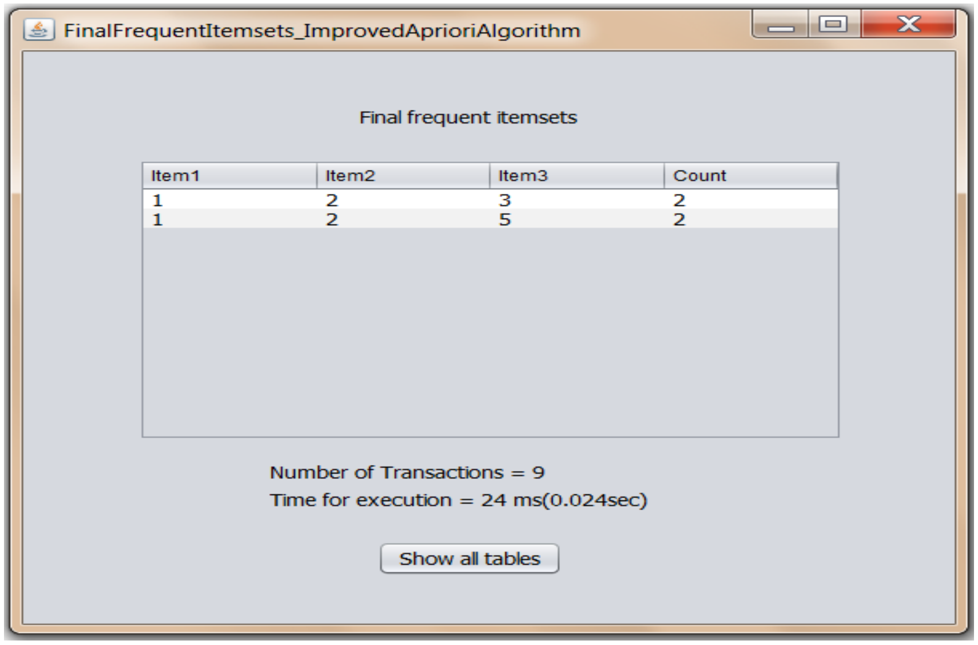

Improved Apriori Algorithm
Perform Improved Apriori Algorithm

Select the database: Transaction1

9 transactions

Enter the minimum support: 3

Showing output of improved Apriori algorithm:



Final frequent itemsets

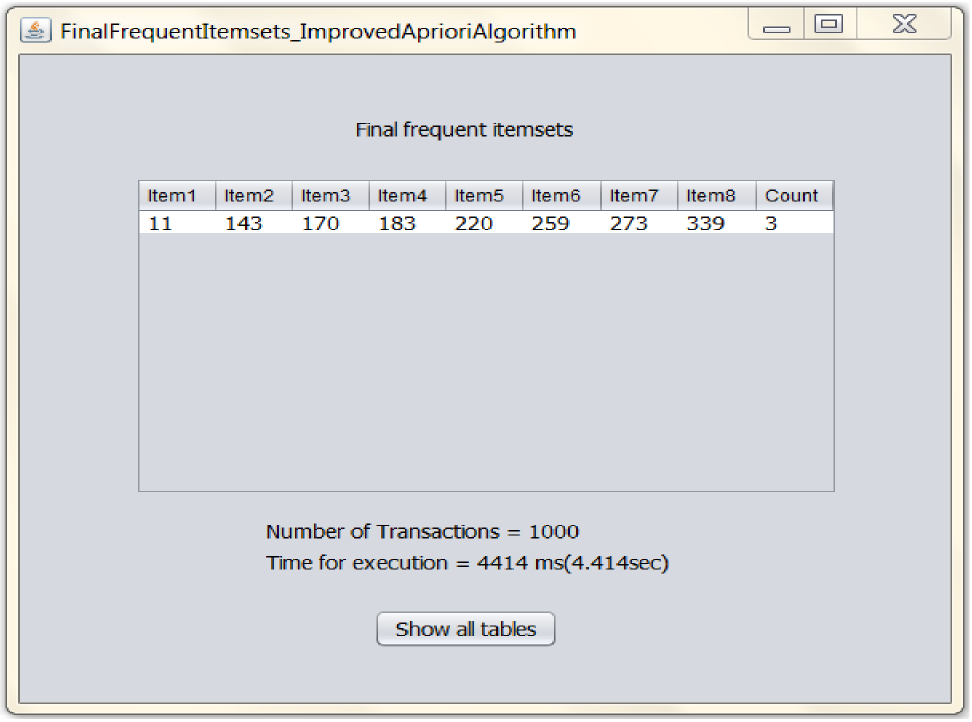
Item1	Item2	Item3	Count
1	2	3	2
1	2	5	2

Number of Transactions = 9

Time for execution = 24 ms(0.024sec)

The above form displays the output of Transaction1 database with minimum support count=2 generated by the improved Apriori algorithm. The time taken for execution is less compared to Apriori algorithm.

Showing output of Improved Apriori algorithm for Transaction3:



Final frequent itemsets

Item1	Item2	Item3	Item4	Item5	Item6	Item7	Item8	Count
11	143	170	183	220	259	273	339	3

Number of Transactions = 1000
Time for execution = 4414 ms(4.414sec)

The time for execution of Transaction3 containing 1000 transactions and minimum support =3 is approximately 4 sec. Whereas the time taken for execution by Apriori algorithm is approximately 14 sec as shown before.

6.3 Comparison of Apriori and Improved Apriori Algorithms:

Comparing the time of execution of both the algorithms for all the 3 databases with minimum support count =3, results are shown below:

Number of Transactions	**Apriori Algorithm**	**Improved Apriori Algorithm**
10	55 ms	20 ms
100	250 ms	100 ms
1000	14.4 sec	4.2 sec

The above table shows that the improved Apriori reduce the time consuming by 63.63% from the original Apriori in Transaction1, by 60% in Transaction2, and by 70.80% in Transaction3. The average of reducing time rate in the improved Apriori is **64.81**

7. CONCLUSION AND FUTURE SCOPE OF WORK

Conclusion

Apriori algorithm suffers from some weakness in spite of being clear and simple. The main limitation is costly wasting of time to hold a vast number of candidate sets with much frequent itemsets, low minimum support or large itemsets .

In this paper, an improved Apriori is proposed through reducing the time consumed in transactions scanning for candidate itemsets by reducing the number of transactions to be scanned. Whenever the k of k-itemset increases, the gap between our improved Apriori and the original Apriori increases from view of time consumed, and whenever the value of minimum support increases, the gap between our improved Apriori and the original Apriori decreases from view of time consumed. The time consumed to generate candidate support count in our improved Apriori is less than the time consumed in the original Apriori; our improved Apriori reduces the time consuming by 64.81.

Future Scope of work

There are several areas in this field calling out for additional research. A few examples are:

- We did not consider the quantities of items bought in a transaction, which are useful for some applications. Finding such rules needs further work.
- Other approaches for finding frequent itemsets.

8. BIBLIOGRAPHY

[1] "Data Mining - concepts and techniques" by Jiawei Han and Micheline Kamber.

[2] "Java The Complete Reference" by Herbert Schildt.

[3] N. Bandi, A. Metwally, D. Agrawal, and A. E. Abbadi. Fast data stream algorithms using associative memories. In ACM SIGMOD, 2007.

[4] "Introduction to data mining and its applications" S. Sumathi, S. N. Sivanandam.

[5] "Introduction to Data Mining with Case Studies", G.K. Gupta.

[6] L. Bhuvanagiri, S. Ganguly, D. Kesh, and C. Saha. Simpler algorithm for estimating frequency moments of data streams. In ACM-SIAM Symposium on Discrete Algorithms, 2006.

[7] S. Rao, R. Gupta, "Implementing Improved Algorithm Over APRIORI Data Mining Association Rule Algorithm", *International Journal of Computer Science And Technology*, pp. 489-493, Mar. 2012

[8] H. H. O. Nasereddin, "Stream data mining," *International Journal of Web Applications*, vol. 1, no. 4,pp. 183–190, 2009.

[9] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI magazine*, vol. 17, no. 3, p. 37, 1996.

[10] R.Srikant, "Fast algorithms for mining association rules and sequential patterns," UNIVERSITY OF WISCONSIN, 1996.