

St Xavier's College (autonomous), Kolkata



IMPLEMENTATION OF APRIORI ALGORITHM USING HASHING CONCEPT TO VIEW FREQUENT ITEMSETS IN A SET OF TRANSACTIONS

By

Soumodeep Dutta, 513

Under Guidance

Of

Prof Debabrata Datta

Submitted to the Department of Computer Science

In partial fulfilment of the

Requirements for the

Degree of B.Sc

CERTIFICATE OF AUTHENTICATED WORK

This is to certify that the project report entitled _____ submitted to
Department of Computer Science, ST. XAVIER'S COLLEGE [AUTONOMOUS],
KOLKATA, in partial fulfilment of the requirement for the award of the degree of
BACHELOR OF SCIENCE (B.SC.) is an original work carried out by Mr./ Ms.

_____ Registration no. _____
under my guidance. The matter embodied in this project is authentic and is genuine work
done by the student and has not been submitted whether to this College or to any other
Institute for the fulfilment of the requirement of any course of study.

.....

Signature of the
Student

Date:.....

.....

Signature of the
Supervisor

Date:

Registration No.

ROLES AND RESPONSIBILITIES FORM

Name of the Project...Implementation of Apriori Algorithm using Hashing Concept to View Frequent Itemsets in a Set of Transactions.....

Date:.....31/03/2017.....

Name of the Team Member	Role	Tasks and Responsibilities
1.Soumodeep Dutta	Designer and Programmer Database Manager	Coding Algorithm Design Management of Database
2.Deborupa Roy	Team Coordinator Network Manager	Algorithm Design Management of the Server
3.Atindriya De	User Interface Builder and Designer Auditor Quality Manager	Building and designing the User Interface Testing, Debugging

Name and Signature of the **Project Team members:**

1...Soumodeep Dutta..... Signature.....

2...Deborupa Roy..... Signature.....

3...Atindriya De.....Signature.....

Signature of the Supervisor:.....

Date: 31/03/2017

ABSTRACT

Data Warehousing, data mining and analysis play a very important role in decision support. Various commercial organisations and database management systems have many products in this area and are still developing new products and software at every instance of time. Apriori algorithm is a classic algorithm which works on a set of data in the database and provides us with the set of most frequent itemsets.

It is used to find the association rules and mines the most frequent itemsets in a set of transactions. The approach which is followed by the Apriori algorithm is a “bottom up” approach. Here the frequent subsets are extended one item at a time. this particular step is known as the candidate generation step and in this step the groups of candidates are tested or checked against some data.

This algorithm was mainly designed for the purpose of operating on those databases which include transactions. In this paper we propose a hash based technique to implement Apriori algorithm. Hashing helps in improving the spatial requirements as well as makes the process faster.

The main purpose behind our project is to help in decision making. The user will be selecting an item which he wishes to purchase, and his item selection is analysed to give him an option of two and three item set (inclusive of his item). He can consider choosing the two item set combination or the three item set combination or he can choose to go with his own purchase. Either ways, the algorithm helps him in making a decision.

ACKNOWLEDGEMENT

Words can never simply express the deep sense of gratitude I have for my institution pre-dominant by itself – ST. XAVIER'S COLLEGE [AUTONOMOUS], KOLKATA, kindest of all, which has provided me the opportunity to pursue my UG Level Education.

I owe my thanks to the Principal REV. **Dr. John Felix Raj, S.J.**, ST. XAVIER'S COLLEGE [AUTONOMOUS], KOLKATA for allowing me to do this project work in partial fulfilment of the Degree of Bachelor of Computer Science.

In performing our assignment, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this assignment gives us much Pleasure. We would like to show our gratitude **Prof. Shalabh Agarwal**, Head of Computer Science Department for giving us a good guideline for assignment throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in writing this assignment.

I express my gratitude to our mentor **Prof. Debabrata Datta**, Assistant Professor of Computer Science Department for his timely suggestion, able guidance and painstaking interest at every stage of this project. Without his guidance and his suggestions, it would have been impossible for us to complete this project.

I sincerely, thank all the Staff members of Computer Science Department, ST. XAVIER'S COLLEGE [AUTONOMOUS], KOLKATA. My hearty thanks to one and all those who stood by me to mould this project properly. Many people, especially our classmates and team members itself, have made valuable comment suggestions on this proposal which gave us an inspiration to improve our assignment. We thank all the people for their help directly and indirectly to complete our assignment.

TABLE OF CONTENTS

CHAPTER NO.	CHAPTER	PAGE
1.	Introduction	9-13
2.	Survey of Technologies	14-22
3.	Requirements and Analysis	23-33
4.	System Design	34-60
5.	Implementation and Testing	61-88
6.	Results and Discussion	89-114
7.	Conclusion	115-117
8.	References	118

TABLE OF FIGURES

FIGURE NO.	NAME	PAGE
2.1	MySql vs Oracle	20
3.1(a)	Pert chart	25
3.1(b)	Gantt Chart (1)	26
3.1(c)	Gantt Chart (2)	26
3.4(a)	Transaction Table	28
3.4(b)	C1 Table	28
3.4(c).	L1 Table	28
3.4(d).1	C2 Table	29
3.4(d).2	C2 Table with support count	29
3.4(e)	L2 Table	30
3.4(f)	C3 Table	30
3.4(g)	L3 Table	30
3.5(a)	Client Site Flowchart	32
3.5(b)	Server End System FlowChart	33
4.3.1(a)	Process Diagram	50
4.4(a)	Item Page	56
4.4(b)	Option Page	57
4.4(c)	Items to Buy Page	58
4.4(d)	Transaction Page	58
6.1(a)	Transaction table	90

6.1(b)	C1 Table	90
6.1(c)	L1 Table	91
6.1.1(a)	H2 table (Case 1)	91
6.1.1(b)	L2 Table (Case 1)	92
6.1.1(c)	H3 table (Case 1)	92
6.1.1(d)	L3 table (Case 1)	93
6.1.2(a)	H2 table (Case 2)	93
6.1.2(b)	L2 Table (Case 2)	94
6.1.2(c)	H3 table (Case 2)	94
6.1.2(d)	L3 table (Case 2)	95
6.1.3(a)	H2 table (Case 3)	95
6.1.3(b)	L2 Table (Case 3)	96
6.1.3(c)	H3 table (Case 3)	98
6.1.3(d)	L3 table (Case 3)	97
6.1.4(a)	H2 table (Case 4)	99
6.1.4(b)	L2 Table (Case 4)	99
6.1.4(c)	H3 table (Case 4)	99
6.1.4(d)	L3 table (Case 4)	100

CHAPTER 1: INTRODUCTION

1.1: Background

1.2: Objectives

1.3: Purpose, Scope, and Applicability

1.3.1: Purpose

1.3.2: Scope

1.3.3: Applicability

1.5: Organisation of Report

INTRODUCTION

1.1: Background

Every year the amount of data being generated increases exponentially thus making the process of extracting useful information from them all the more tedious and critical. Most of these information is stored in a data repository known as the data warehouse. The data warehouse consists of data which are gathered from various sources inclusive of external sources, summarized form of information from the various internal systems as well as from the corporate databases. Also storing the data is not enough, extracting useful information from these wide collection of data is equally important. This is where data analysis comes into place and it consists of writing simple queries, doing complex multidimensional analysis, data mining as well as presenting reports of statistical analysis.

Data mining and data analysis falls under the wider and bigger concept of business intelligence which also consists of Online Analytical Processing(OLAP), data warehousing and database management systems.

A data warehouse consists of the data model, metadata (i.e., data about data) as well as all the different rules in relation to data aggregation, distribution, replication, error handling and all the other various information required for the purpose of mapping the data warehouse. Developing useful information from the raw data in an easy and fast method is what a data warehousing strategy all about. The various data mining and data analysis tools make use of pattern recognition, cluster analysis, quantitative analysis, associations and correlation discovery for the purpose of data analysis without any kind of IT intervention.

There are two main types of pattern mining:

Sequential Pattern Mining: Study the order in which items are frequently purchased in a sequence data set, where sequence records an ordering of events.

Structured Pattern Mining: Search for frequent sub-structures in a structured data set. Structure can consist of – graphs, lattices, sequence, sets, tree, single item or combination of above structures.

Each element of an itemset may contain a subsequence, a subtree and so on and such containment relationship can be defined recursively. Therefore structural pattern mining can be considered as the most general form of frequent pattern mining.

Once the useful information is extracted, it is presented to the user in a comprehensible form using processes which are collectively called BI (Business Intelligence). Managers can choose in between the various types of analysis tools available such as reports and queries, OLAP and its many variants (MOLAP, HOLAP and ROLAP). Data mining supports these and the patterns formed are later used for further data analysis thus completing the process of BI.

Online Analytical Processing (OLAP) is one of the most popular data analysis technology. Here, data is organised into cubes or multidimensional hierarchies using OLAP servers to enable fast analysis of data. The various data mining algorithms uncover patterns or relationships by scanning the databases. Data mining and OLAP are complementary where data mining takes up a bottom up approach for data analysis and OLAP provides the top-down approach. One such data analysis algorithm which are working on is the already established algorithm- Apriori algorithm. We are enhancing the efficiency of the algorithm by implementing it with the hashing technique.

There have been several works which have been done using the Apriori algorithm.

1. ***“An enhanced Apriori Algorithm for Discovering Frequent Patterns with Optimal Number of Scans”*** by Sudhir Tirumalasetty, et al^[1]

There has been a wide and extensive application of data mining in the various areas. One important role which is played by data mining is the identification of frequent patterns. This particular paper has focused on the inadequacy of the Apriori algorithm of scanning the entire transactional database for the purpose of discovering association rules thus wasting a lot of time. This paper has proposed an improvement on the Apriori algorithm to overcome this particular problem. The amount of time spent in scanning the entire transactional database is reduced by restricting the number of transactions while doing a calculation of the frequency of item-pairs or an item. Thus this improved algorithm optimises the time complexity.

2. ***“Implementation of Association Rule Mining using Reverse Apriori Algorithmic Approach”*** by Asma Chawla, et al^[2].

This paper has made an improvement on the reverse Apriori and has compared the results with the existing classical one.

3. ***“A parallel Apriori Algorithm for Frequent Itemsets Mining”*** by Yanbin Ye, et al^[3]

This paper has revised the Bodon’s implementation of finding the frequent itemsets into parallel ones where parallel computers are used to read the input transactions. They have presented the effect of the parallel computers on this implementation.

4. ***“Enhancing the Apriori Algorithm for Frequent Set Counting”*** by Raffaele Perego, et al^[4]

This paper proposes a new and enhanced form of the Apriori algorithm by presenting solution to the Frequent Set Counting problem. They have optimised the beginning iterations of the Apriori, which consume the maximum time when datasets with frequent length patterns of short or medium are considered. They have used a method of storing candidate items sets and keeping a count of their support and have also exploited the effective pruning methods, which has brought about a reduction in the size of the data set with the progress of the execution.

1.2: Objectives

We have enhanced the already established Apriori algorithm to view Frequent Itemsets in a set of transactions through the hashing concept. The Apriori algorithm is one of the most important data mining algorithm to generate frequent itemsets for the Boolean association rules. The approach is an iterative one and is known as level wise search where k number of items are used to explore the (k+1) items.

1.3: Purpose , Scope And Applicability

1.3.1: Purpose

The project aims to achieve a method to find the frequent itemsets using a transaction table and transaction strings. When the number of transactions increases Apriori algorithm takes up large space and huge time to compute. The method we proposed, implements Apriori algorithm using hash table which reduces the space complexity vastly.

1.3.2: Scope

The main aim of the project is to find the frequent itemsets reducing the previous space requirements that the Apriori algorithm needed. The programme works on every transaction that is taking place and maps them to a hash table when they are meeting the required threshold condition. The hash table is then consulted and once items in the hash table meet the next set of threshold conditions they are displayed.

1.3.3: Applicability

The algorithm helps in decision support. This concept can be used in various e-commerce sites helping the users to make better decision of the products to buy based on the past history of all the users who bought similar items. For example, suppose in a grocery shopping e-site most customers who buy bread also buys jam or butter or eggs or milk or all of them or some of them in combination. It has been observed this is quite a frequent combination in transactions. Hence next time when a user selects bread to buy he will be suggested to buy jam or butter or eggs or milk or all of them or some of them in combination.

1.4: Achievements

Working on this project has given us an idea about hashing and how it can be implemented on the apriori algorithm for the purpose of reducing the spatial complexity as well as making the code much more optimised and faster.

It will help people in the process of making decisions as the primary goal of the algorithm is to analyse the data and give output in the form of two item set and three item set combinations which the user is most likely to purchase. Thus the user can now make decisions with more ease.

1.5: Organisation of Report

An outline is given to the reader up until now. The introduction is next followed by the main phases of Analysis, Planning, design, implementation and testing. These topics cover the topic of the project in more details. The planning and implementation phase has dealt with the various hardware and software requirements of this project. Survey of technologies will also give the reader an idea about why the following languages and software are chosen over the other languages and software present in the market and will basically give you a comparative study of the two. The source code along with the algorithm and a brief description of each of the modules are also to follow shortly. Various screenshots of the output has been taken as during the designing of the test case. The main hardships as faced during the implementation of the project as well as the limitations faced in the project are also given.

CHAPTER 2. SURVEY OF TECHNOLOGIES

SURVEY OF TECHNOLOGIES

Various softwares has been used to achieve our project. Java Eclipse Neon IDE has been employed for the main coding part. JDBC has been used to establish a connection between Java and MySql Server & Workbench. Database management and handling has been achieved using MySql Server & Workbench. Apache Tomacat 9 connects Java with the Web Server and provides an environment to run Java code and the GUI has been designed using A detailed study of all the softwares are as follows:

1. Java Eclipse Neon IDE

Eclipse is an integrated development environment (IDE) used in computer programming, and is the most widely used Java IDE. Eclipse software development kit (SDK) is free and open-source software, released under the terms of the Eclipse Public License. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including: Ada, ABAP, C, C++, COBOL, D, Fortran, Haskell, JavaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Rust, Scala, Clojure, Groovy, Scheme, and Erlang. It also supports development for Tomcat, GlassFish and many other servers and is often capable of installing the required server (for development) directly from the IDE.

2. MySql Server and Workbench

MySQL is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius' daughter, and "SQL", the abbreviation for Structured Query Language. MySQL is written in C and C++. Its SQL parser is written in Yacc. It works on many system platforms such as Microsoft Windows, Linux, macOS, Open Solaris, Oracle Solaris etc.

It performs extremely well in the average case and has also been tested to be a "fast, stable and true multi-user, multi-threaded SQL database server".

Major features as available in MySQL 5.6:

- ❖ A broad subset of ANSI SQL 99, as well as extensions
- ❖ Cross-platform support

- ❖ Stored procedures, using a procedural language that closely adheres to SQL/PSM
- ❖ Triggers
- ❖ Cursors
- ❖ Updatable views
- ❖ Online DDL when using the InnoDB Storage Engine.
- ❖ Information schema

MySQL Workbench is the official integrated environment for MySQL. It was developed by MySQL AB, and enables users to graphically administer MySQL databases and visually design database structures. MySQL Workbench replaces the previous package of software, MySQL GUI Tools. Similar to other third-party packages, but still considered the authoritative MySQL front end, MySQL Workbench lets users manage database design & modelling, SQL development (replacing MySQL Query Browser) and Database administration (replacing MySQL Administrator).

MySQL Workbench is available in two editions, the regular free and open source *Community Edition* which may be downloaded from the MySQL website, and the proprietary *Standard Edition* which extends and improves the feature set of the Community Edition.

3. Java 2 Platform Enterprise Edition

Java 2 Platform Enterprise Edition or J2EE is a platform-independent, Java-centric environment from Sun for developing, building and deploying Web-based enterprise applications online. The J2EE platform consists of a set of services, APIs, and protocols that provide the functionality for developing multi-tiered, Web-based applications. It is a container and contains a set of components and infrastructure code. With this components and infrastructure code a developer can build the server side of the enterprise web application.

Some of the key features and services of J2EE:

- At the client tier, J2EE supports pure HTML, as well as Java applets or applications. It relies on Java Server Pages and servlet code to create HTML or other formatted data for the client.
- Enterprise JavaBeans (EJBs) provide another layer where the platform's logic is stored. An EJB server provides functions such as threading, concurrency, security and memory management. These services are transparent to the author.
- Java Database Connectivity (JDBC), which is the Java equivalent to ODBC, is the standard interface for Java databases.
- The Java servlet API enhances consistency for developers without requiring a graphical user interface.

Now, an enterprise web application is a request response model. The one who makes a request is called a client. The client can be made using the technologies HTML or JavaScript. The one who gives a response is called a server. The server can be made using the technologies "J2EE" or ".NET". Client sends a request to the server. Server does a processing on the request. And then server send backs a response.

Some of the issues in this enterprise web application are:

1. Server should be able to accept the request from the client and should be able to send back the response.
2. Server should be able to do some transactions.
3. Server should be able to communicate with the database.

Oracle developers have found these issues in an enterprise web application. And they have developed a container called "J2EE". This J2EE container contains components like JSP/Servlet(for solving issue 1), EJB(for solving issue 2) and JDBC(for solving issue 3) and many more. This J2EE container also contains infrastructure code (example when client sends a request then J2EE should invoke the corresponding Servlet/JSP).

4. Apache Tomcat

Apache Tomcat, often referred to as Tomcat Server, is an open-source Java Servlet Container developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, JavaServer Pages (JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment in which Java code can run.

Tomcat is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation, released under the Apache License 2.0 license, and is open-source software.

The main components of the Tomcat Server are as follows:

❖ Catalina

Catalina is Tomcat's servlet container. Catalina implements Sun Microsystems's specifications for servlet and JavaServer Pages (JSP). In Tomcat, a Realm element represents a "database" of usernames, passwords, and roles (similar to UNIX groups) assigned to those users. Different implementations of Realm allow Catalina to be integrated into environments where such authentication information is already being created and maintained, and then use that information to implement Container Managed Security as described in the Servlet Specification

.

❖ Coyote

Coyote is a Connector component for Tomcat that supports the HTTP 1.1 protocol as a web server. This allows Catalina, nominally a Java Servlet or JSP container, to also act as a plain web server that serves local files as HTTP documents.

Coyote listens for incoming connections to the server on a specific **TCP** port and forwards the request to the Tomcat Engine to process the request and send back a response to the requesting client. Another Coyote Connector, Coyote JK, listens similarly but instead forwards its requests to another web server, such as Apache, using the **JK protocol**. This usually offers better performance.

❖ Jasper

Jasper is Tomcat's JSP Engine. Jasper parses JSP files to compile them into Java code as servlets (that can be handled by Catalina). At runtime, Jasper detects changes to JSP files and recompiles them.

As of version 5, Tomcat uses Jasper 2, which is an implementation of the Sun Microsystems's JSP 2.0 specification. From Jasper to Jasper 2, important features were added:

- JSP Tag library pooling - Each tag markup in JSP file is handled by a tag handler class. Tag handler class objects can be pooled and reused in the whole JSP servlet.
- Background JSP compilation - While recompiling modified JSP Java code, the older version is still available for server requests. The older JSP servlet is deleted once the new JSP servlet has finished being recompiled.
- Recompile JSP when included page changes - Pages can be inserted and included into a JSP at runtime. The JSP will not only be recompiled with JSP file changes but also with included page changes.
- JDT Java compiler - Jasper 2 can use the Eclipse JDT (Java Development Tools) Java compiler instead of Ant and Javac.

A number of additional components may be used with Apache Tomcat. These components may be built by users should they need them or they can be downloaded.

5. Java Database Connectivity

Java Database Connectivity (JDBC) is an application programming interface (API) for the programming language Java, which defines how a client may access a database. It is part of the Java Standard Edition platform, from Oracle Corporation. It provides methods to query and update data in a database, and is oriented towards relational databases. A JDBC-to-ODBC bridge enables connections to any ODBC-accessible data source in the Java virtual machine (JVM) host environment.

JDBC allows multiple implementations to exist and be used by the same application. The API provides a mechanism for dynamically loading the correct Java packages and registering them with the JDBC Driver Manager. The Driver Manager is used as a connection factory for creating JDBC connections.

JDBC connections support creating and executing statements. These may be update statements such as SQL's CREATE, INSERT, UPDATE and DELETE, or they may be query statements such as SELECT. Additionally, stored procedures may be invoked through a JDBC connection.

COMPARATIVE STUDY:

1. PHP VS JSP:

- JSP developed by Sun, runs on Apache Tomcat, slower than PHP but good for big level of projects. PHP is open source, runs on Apache, faster than all and best for small and middle level of projects.
- JSP is Completely object oriented. PHP gives scripting with object oriented support.
- The main advantage of JSP is Clean Code. PHP provides functional and quick coding, we can use OOP practices at our convenience but the code gets clumsy.
- PHP is quick as compared to Java. But maintenance of PHP code tends to be difficult due to lack of constraints on the developer for structuring the code.
- PHP is much more well established with libraries, frameworks, patterns and code base for general web development, while JSP (java's) base is much more standard's oriented.
- JSP (java platform) has huge availability of developers for the platform, whereas PHP falls short on this point.
- PHP has in built support for Web Services whereas JSP needs addon library like JAX-RS, Axis.

2. PHP MyAdmin VS MYSQL Workbench:

➤ PHP MyAdmin

Pros

- Commonly installed on managed hosting environments
- Web Based which means you can access from any computer
- Local resources aren't used when connecting
- Simplicity

Cons

- No schema visualization
- If remote database working offline can be more difficult

➤ MySQL Workbench

Pros

- Visualizing DB structure (Highly recommended for beginners)
- Saved SQL statements
- Offline access to remote DB's
- Handle/Store multiple connections in one location

Cons

- Resource consumption
- More complex than the average user would need

3. MySQL VS Oracle:

Given below is a comparative study between MySQL and Oracle database server based upon various properties, features and functionality.

Features/Functionality	MySQL	Oracle
Strengths	Price/Performance Great performance when applications leverage architecture.	Aircraft carrier database capable of running large OLTP and VLDBs.
Administration	Can be trivial to get it setup and running. Large and advanced configurations can get complex.	Requires lots of in-depth knowledge and skill to manage large environments. Can get extremely complex but also very powerful.
Popularity	Extremely popular with web companies, startups, small/medium	Extremely popular in Fortune 100, medium/large enterprise business applications

	businesses, small/medium projects.	and medium/large data warehouses.
Application Domains (most popular)	Web (MySQL excels) Data Warehouse Gaming Small/medium OLTP environments	Medium/Large OLTP and enterprise applications. Oracle excels in large business applications (EBS, Siebel, PeopleSoft, JD Edwards, Retek, ...) Medium/Large data warehouse
Development Environments (most common)	1) PHP 2) Java 3) Ruby on Rails 4) .NET 5) Perl	1) Java 2) .NET 3) APEX 4) Ruby on Rails 5) PHP Note: Oracle focusing on Java for next generation business applications.
Database Server (Instance)	Database Instance stores global memory in mysqld background process. User sessions are managed through threads.	Database instance has numerous background processes dependent on configuration. System Global Area is shared memory for SMON, PMON, DBWR, LGWR, ARCH, RECO, etc. Sessions are managed through server processes.
Database Server (Physical Storage)	Made up of database schemas. Each storage engine stores information differently. Common storage engines: MYISAM – stores data in .FRM, .MYD and .MYI files. InnoDB – stores data in a common tablespace or individual tablespaces per table.	Uses tablespaces for system metadata, user data and indexes. Common tablespaces include: SYSTEM SYSAUX USER DATA USER INDEXES TEMPORARY UNDO Redo and archive log files are used for point in time recovery.

	Binary logs are used for point-in-time recovery	
Tables	Tables use storage engines. Each storage engine provides different characteristics and behavior.	A few tables with tons of features.
Replication	Free, relatively easy to setup and manage. Basic features but works great. Great horizontal scalability.	Lots of features and options. Much higher complexity with a lot of features. Allows a lot of data filtering and manipulation.
Backup/Recovery	No online backup built-in. Replication OS Snapshots InnoDB Hot Backup	Recovery Manager (RMAN) supports hot backups and runs as a separate central repository for multiple Oracle database servers.
Export/Import	Easy, very basic.	More features.
Data Dictionary (catalog)	Information_schema and mysql database schemas offer basic metadata.	Data dictionary offers lots of detailed information for tuning. Oracle starting to charge for use of new metadata structures.
Storage	Each storage engine uses different storage. Varies from individual files to tablespaces.	Tables managed in tablespaces. ASM offers striping and mirroring using cheap fast disks.
Stored Procedures	Very basic features, runs interpreted in session threads. Limited scalability.	Advanced features, runs interpreted or compiled. Lots of built in packages add significant functionality. Extremely scalable.

Fig:2.1 MySql vs Oracle

CHAPTER 3. REQUIREMENTS AND ANALYSIS

3.1: Problem Definition

3.2: Planning and Scheduling

3.3: Software and Hardware Requirements

3.3.1: Software Specification

3.3.2: Hardware Specification

3.4: Preliminary Product Description

3.5: Conceptual Models

REQUIREMENTS AND ANALYSIS

3.1: Problem Definition

In the present generation technology forms an integral part of every person's life, and in every stage of life a person is required to make decisions or choice. The Apriori algorithm is probably one of the most basic algorithm out there which helps in the process of data mining and data analysis. Paying a visit to the shopping centres you will find that it is very easy to find the products which you wish to purchase.

This is a real world example of the implementation of the Apriori algorithm. Based on the pattern of purchase found in the customers, the items which are most likely to be purchased by the customers are placed near each other. While this implementation of the Apriori algorithm is purely real life, the actual application of Apriori algorithm can often be noticed while visiting an online shopping place.

In the online stores, we often find that after choosing a particular item, we get an option of choosing either a combination of two items or three items (inclusive of your item). This combination of option is given by studying the pattern of your purchase. A choice is given to the customer to either go with any of the options or he can only choose the item he wishes to purchase and check out.

In this project we are working to enhance the already established Apriori Algorithm. We are viewing the frequent item sets from a set of generated transactions and improving the spatial complexity of the algorithm by using the hashing technique.

Initially we are taking the transaction string as an input from the user. The transaction string comprises of all the different items which the user is willing to purchase. Once the transaction string has been taken as input, we send the user a possible combination of frequent one item set, two item set and three item sets based on his present transaction. The basic idea behind this particular algorithm is to help the user in decision support.

3.2: Planning And Scheduling

1. Group discussions were held and a fundamental outline of the task was planned.
2. The entire work was divided in an equal manner amongst the members of the project.
3. The basic algorithm was designed.
4. The pseudo code based on the algorithm was generated.
5. The background coding was started and each of the modules were developed individually.
6. Java Database Connectivity(JDBC) was used to establish a connection between the MySQL server and the background code.
7. Each of the modules were individually scrutinised and unit tested.
8. Connection was then established between the server and the client using Apache Tomcat 9.

9. Each of the modules were then combined into a group and integration testing was done on them.

PERT CHART: A PERT chart is a project management tool that provides a graphical representation of a project's timeline. PERT, or Program Evaluation Review Technique, allows the tasks in a particular project to be analyzed. Although PERT chart are preferable to Gantt charts because they more clearly identify task dependencies, PERT chart are often more difficult to interpret.

A PERT chart is a visual representation of a series of events that must occur within a project's lifetime. The direction of arrows indicates the flow of the sequence of events that need to occur. Dotted activity lines represent dummy activities, which are items located on another PERT path. Numbers assigned to each task, as well as time allotments, are located inside each vector.

The PERT chart for our project is show below:

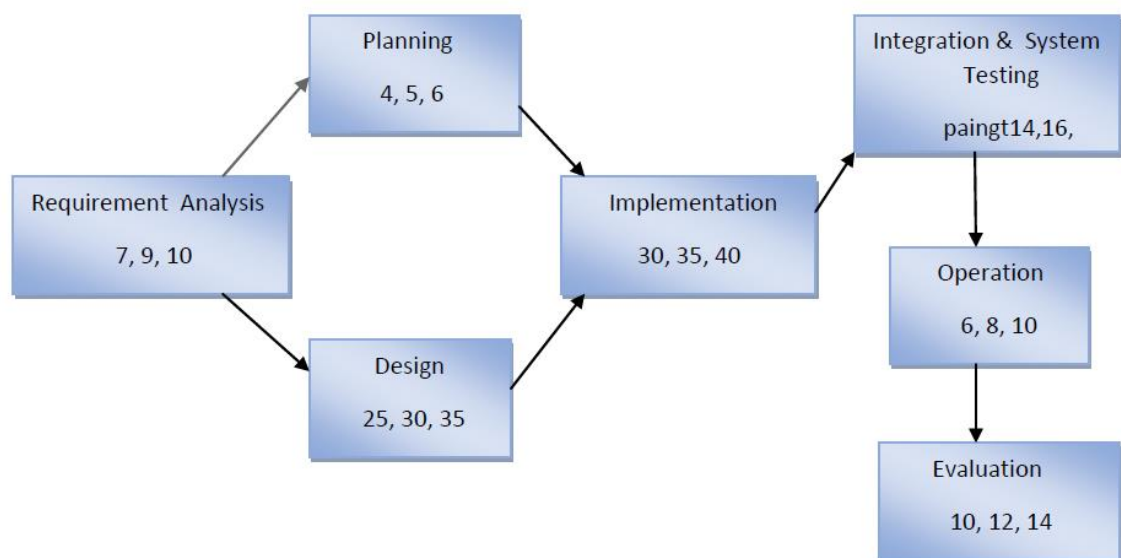


Fig: 3.1(a) Pert Chart

GANTT CHART: A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity.

NAME	BEGIN DATE	END DATE
Requirement and Analysis	12/12/16	21/12/16
Identifying Requirements	19/12/16	28/12/16
Planning	26/12/16	30/12/16
Scheduling	02/01/17	06/01/17
Design	09/01/17	17/02/17
Project Design	16/01/17	24/02/17
Implementation and Testing	23/01/17	10/03/17
Coding and Debugging	30/01/17	10/03/17
Testing	06/02/17	28/02/17
Integration and System Testing	13/02/17	07/03/17
Module Integration	20/02/17	07/03/17
System Testing	27/02/17	10/03/17
Operations	06/03/17	15/03/17
Checking Operations	13/03/17	23/03/17
Evaluation	13/3/17	28/03/17

Fig: 3.1(b) Gantt Chart

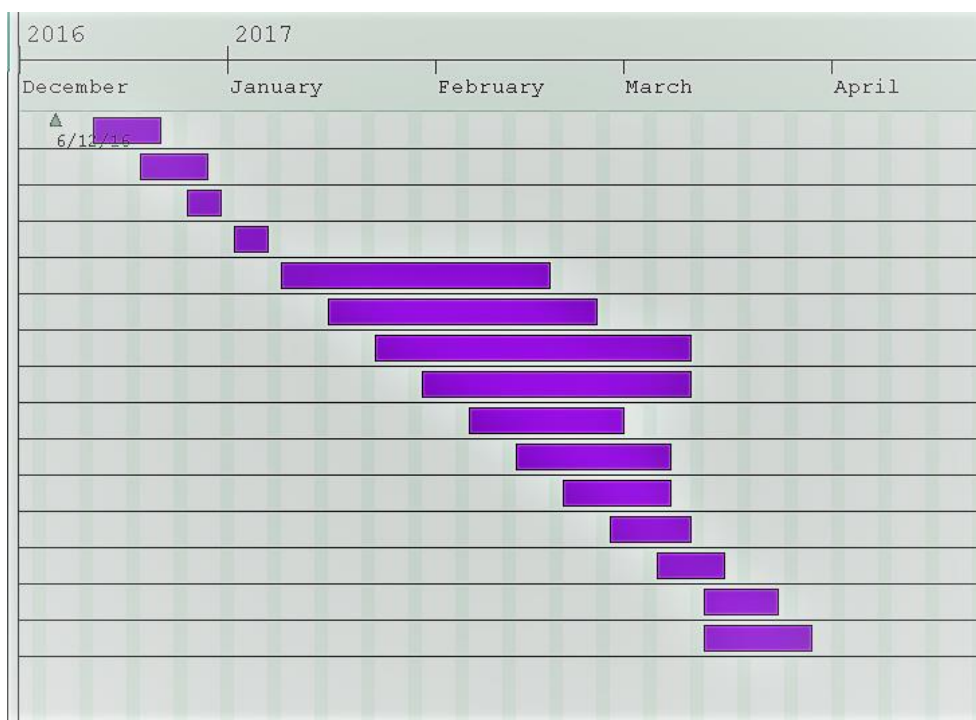


Fig: 3.1(c) Gantt Chart

3.3: Software And Hardware Requirements

3.3.1: Software Specification

- OPERATING SYSTEM: WINDOWS 10
- BACK END CODING: Eclipse Java Neon, Eclipse Java EE

MySql Server and Workbench

Apache Tomcat Server

- FRONT END CODING: HTML, Java Script, CSS

3.3.2 Hardware Specification

- SYSTEM: 2 CPU cores minimum
Intel i5 5th Gen recommended
- RAM: 2 GB minimum
8 GB or more recommended
- DISK SPACE: 800 Mb minimum required by service manager
500 Mb minimum required by monitoring agent
- Phone with net connection or WIFI support
- Disk I/O subsystem applicable to a write intensive database

3.4: Preliminary Product Description

The main aim of this project is to provide decision support to the user. We are building up on the Apriori algorithm and generating the frequent item sets in combination of one, two and three. The user is giving an input of all the items he/she wishes to purchase. This input is taken in the form of a transaction string. Based on the various items present in the transaction string, the user will be given a choice of frequent item sets (one, two and three). The value of these frequent item sets will be generated based on a particular parameter known as the support count.

For example, let there be a transaction table T1 (figure 3.4(a)). It consists of the transaction IDs and the corresponding transaction strings. The given table consists of four transactions.

From T1 we construct the one-item candidate table(C1) as shown in Figure 3.4(b). The candidate table contains all the items and their corresponding frequency. Let the minimum support count be 2. Comparing the minimum support count and the frequency of the items in C1, the frequent one item set table(L1) is generated (figure 3.4(c)) which comprises of items having a frequency greater than or equal to 2.

TRANSACTION ID	ITEM
T01	1,3,4
T02	2,3,5
T03	1,2,3,5
T04	2,5

Fig: 3.4(a) Transaction Table

ITEM	SUP_COUNT
1	2
2	3
3	3
4	1
5	3

Fig. 3.4(b) C1 Table

ITEM	SUP_COUNT
1	2
2	3
3	3
5	3

Fig. 3.4(c) L1 table

A two item-set candidate table(C2) is generated (figure 3.4(d)), which consists of item pairs (i,j) where $i < j$ and $i, j \in L1$.

A frequent two item-set table(L2) is now generated from the already constructed C2 table. L2 consists of all those item pairs (i,j) whose frequency in C2 is greater than or equal to 2 (minimum support count) as given in figure 3.4(e).

ITEM SET
{1,2}
{1,3}
{1,5}
{2,3}
{2,5}
{3,5}

Fig: 3.4(d).1 C2 without Support

ITEM SET	SUP_COUNT
{1,2}	1
{1,3}	2
{1,5}	1
{2,3}	2
{2,5}	3
{3,5}	2

Fig: 3.4(d).2 C2 with Support Count

ITEMSET	SUP_COUNT
{1,3}	2
{2,3}	2
{2,5}	3
{3,5}	2

Fig: 3.4(e) L2 Table

In a similar manner, a three item-set candidate table(C3) and a frequent three item-set table (L3) is generated as shown in figure 3.4(f) and 3.4(g) respectively.

ITEM SET
{2,3,5}

Fig: 3.4(f) C3 table

Fig: 3.4(g) L3 table

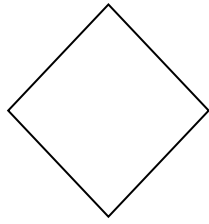
ITEM SET	SUP
{2,3,5}	2

Now when the user selects a certain set of items to buy, the various combination of the items (1, 2 and 3) are formed and compared with the already generated frequent item-set tables (L1, L2, L3). If the combination of items forms a match, then they are displayed.

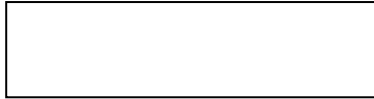
3.5: Conceptual Models

System Flowcharts are a way of displaying how data flows in a system and how decisions are made to control events. They are very similar to data flow charts but data flowcharts do not include decisions, they just show the path that the data takes, where it's held, processed and output.

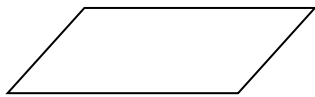
To illustrate System Flowchart, symbols are used. They are connected together to show what happens to data and where it goes. The basic ones include:



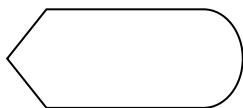
Decisions (yes or no)



Process (something that happens)



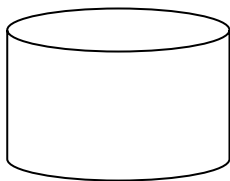
Input or Output



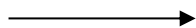
Online Display



Online Input



Data Store



Data Flow

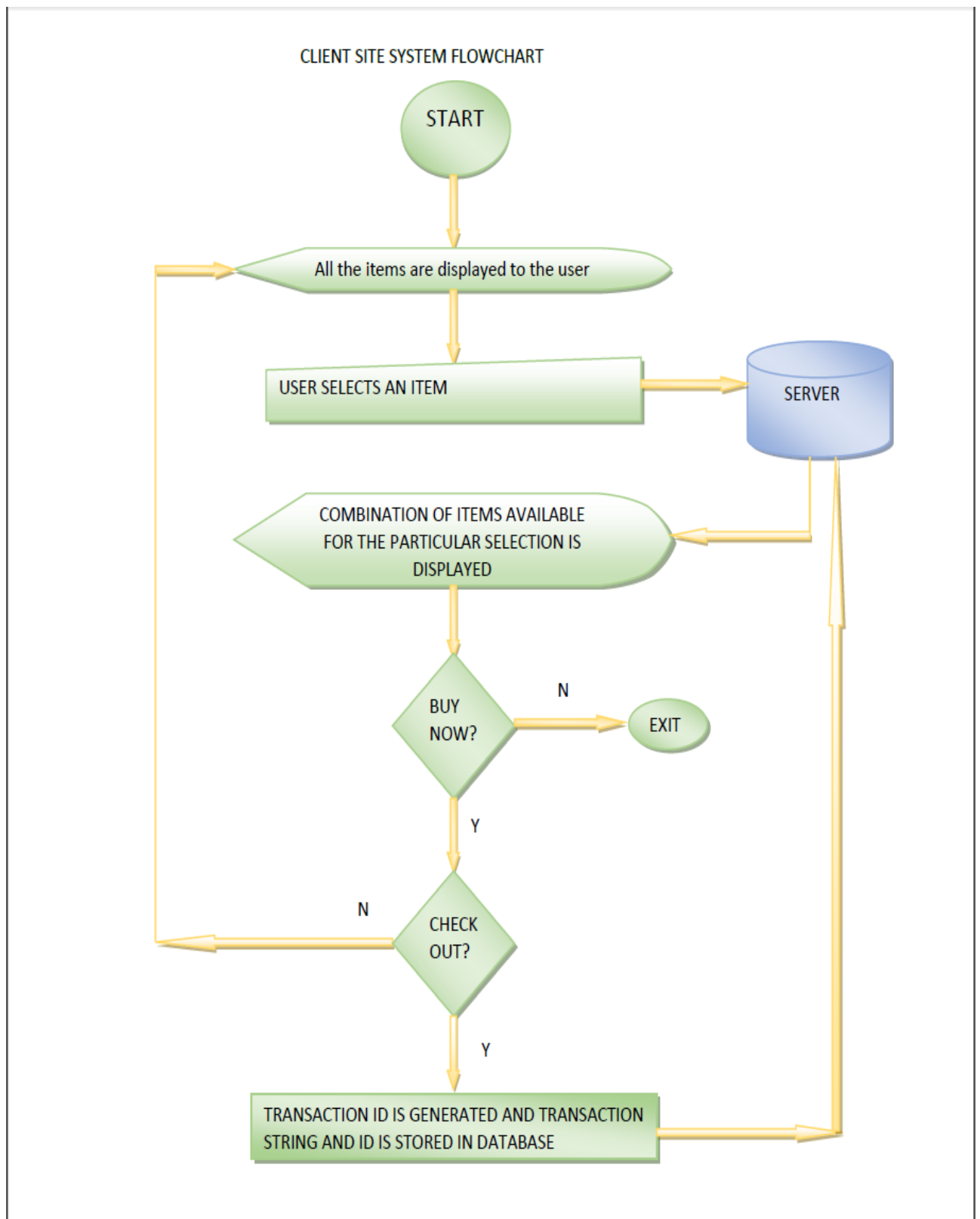


FIG: 3.5(a) CLIENT SITE FLOWCHART

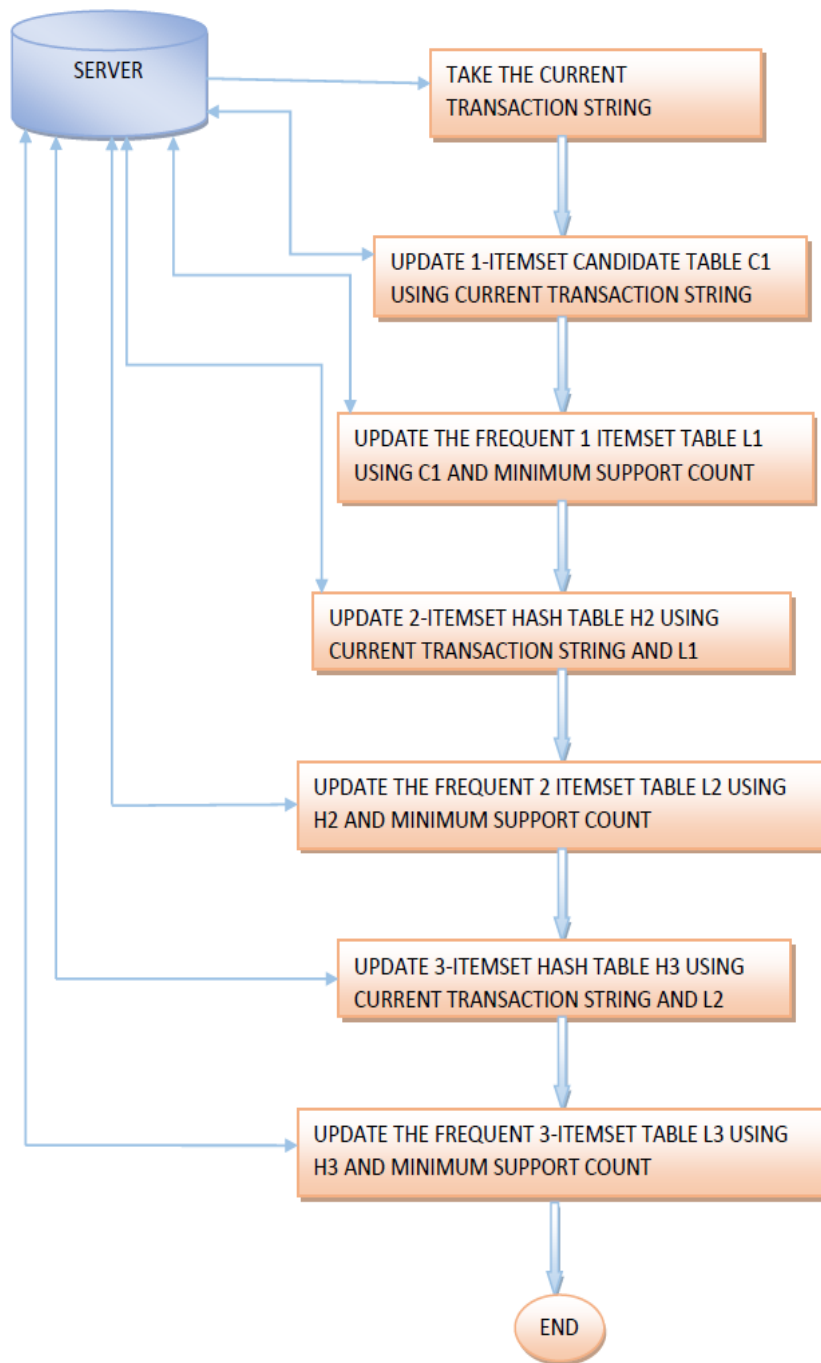


FIG:3.5(b) SERVER END SYSTEM FLOWCHART

CHAPTER 4: SYSTEM DESIGN

4.1: Pseudo Code

4.2: Basic Modules

4.3: Procedural Design

4.4: User Interface Design

4.5: Test Cases Design

SYSTEM DESIGN

4.1: Pseudo Code:

Main()

{

Do Loop{

Print the various items available

Item = user input

Print “ Action : 1. Select 2: Buy Now ”

a1= user’s action

switch(a1)

{

Case 1:

If ViewL1.isL1(item)==true

ResultSetTable.insert(item)

Print “Enter choice”

trans= trans + item selected by user

break

Case 2:

trans = trans + item

Print TransactionID + Transaction String

break

}

Print “ Check Out ? Yes/No

}While(No)

TransactionClass.transaction(trans)

```

TableC1.insert(trans)

ViewL1.insert(min_sup_count)

TableH2.insert(trans)

ViewL2.insert(min_sup_count)

TableH3.insert(trans)

ViewL3.insert(min_sup_count)

}

```

Class ResultSetTable

```

{

/* Function to create the resset table which is the resultant table containing all possible 2 and
3 itemset combination for a given item*/

    insert(int item)
    {

        Execute “Truncate resset” to reset resset table

        Check if item is present in l2 by calling ViewL2.isL2(item)

        If item is present then add the item combination to the resset table

        Else return

        Check if item is present in l3 by calling ViewL3.isL3(item)

        If item is present then add the item combination to the resset table

        Else return

        Print the resset table

    }
}

```

Class TableC1

```

{
    //function to update the I item-set candidate table after every transaction
    insert(String trans)
    {
        ar[]=trans.split(“,”);
    }
}

```

```

        Execute “Update c1 set sup_count = sup_count+1 where item = ?”

        for i = 0 to ar.length-1

            setString(1,ar[i]) /*updating the support count of various elements
                                in c1 by selecting the various elements in the array ar*/

        }

    }

```

Class ViewL1

```

{
    //Function to create l1 after every transaction
    insert(int min_sup_count)
    {
        Execute “Create or replace view l1 as select item,sup_count as count

        from c1 where sup_count>=?”

        setInt(1,min_sup_count)
    }

    // Function to check whether a particular item is present in l1
    isL1(int x)
    {
        /*creating a temporary table ‘rs’ which contains name of item which is
        required*/
        rs = Execute “Select item from l1 where item = ?”

        setInt(1,Integer.toString(x))
    }
}

```

Class TableH2

```

{

    /* Function to create a linkedlist array to temporarily store all the possible 2-itemset
    combination of a particular transaction string*/

    insert( String trans)

    {

        Creating object iset of Itemset_2 class
    }
}

```

```
ar[]=iset.combi() /* calling function combi present in Itemset_2 class
                  using an object of Itemset_2 class*/
```

Declaring and defining an linkedlist array of size 13

Creating object hash of Hashing class

```
/* storing the possible 2 itemset combination in a linkedlist array */
```

```
for i=0 to ar.length - 1
{
    s[]=ar[i].split(",")

    y=Integer.parseInt(s[0])

    x=Integer.parseInt(s[1])

    if(!ViewL1.isL1(y) || !ViewL1.isL1(x))

        continue

    hash.setY(y)

    hash.setX(x)

    index=hash.getHash()

    h[index].add(ar[i])
}

runSql(h)
}
```

```
/* Function to update the 2-itemset hash table using linkedlist array h */
```

```
runSql(LinkedList<String> h)
```

```
{

    Execute "insert into h2 values(?,?,?, ?, ?, ?)"

    flag=0

    Do loop{
        for i=0 to 12
        {
            if(h[i] is empty)
                ps.setNull(i+1,java.sql.Types.VARCHAR)
            else{
                String item=(h[i].remove()).toString()
            }
        }
    }
```

```

        ps.setString(i+1, item)

        flag=1
    }
}
}while(flag==1)
}

```

Class ViewL2

```

{

/* Function to insert all possible 2 itemset combination with frequency greater than
support count into the frequent 2 itemset table*/

```

insert(String trans)

```

{

    table_size=13

    x=-1

    Execute "TRUNCATE l2"

    for i=0 to table_size-1
    {
        String feild=(String)(Character.toString((char) ('a'+i)));

        Execute "insert into l2 SELECT distinct("+feild+") FROM h2
        where "+feild+" is not null group by "+feild+" having count(*)
        >= ?;"

        setInt(1, minSup_count)
    }
}

```

/* Function to check whether a particular 2 item-set combination is present in the l2 table */

isL2(String x)

```

{

    Execute "select * from l2 where item=?"

    ResultSet rs=null

    setString(1,x);/* setting the value of item to x, that is the string
                    to be searched*/
}

```

```
/* Function to check the whether a particular item is present in the table */
```

```
isL2(int x)
```

```
{
```

```
/* creating temporary table rs which contains all the tuples of l2*/
```

```
rs= Execute "select * from l2"
```

```
ArrayList<String> ar=new ArrayList<String>();
```

```
while(rs.next())
```

```
{
```

```
    itemset= extracting the element of the first column of the l2  
    table
```

```
    tmp[]=itemSet.split(",");
```

```
    if( Integer.parseInt(tmp[0])==x || Integer.parseInt(tmp[1])==x)  
    ar.add(itemSet);
```

```
}
```

```
return ar
```

```
}
```

```
}
```

```
class TableH3
```

```
{
```

```
/* Function to create a linkedlist array to temporarily store all the possible 3-itemset  
combination of a particular transaction string*/
```

```
insert(String trans)
```

```
{
```

```
    Creating object iset of Itemset_2 class
```

```
    ar[]=iset.combi() /* calling function combi present in Itemset_2 class  
using an object of Itemset_2 class*/
```

```
    Declaring and defining an linkedlist array of size 13
```

```
    Creating object hash of Hashing class
```

```
    for i=0 to ar.length-1
```

```
    {
```

```
        s[]=ar[i].split(",")
```



```

        z=Integer.parseInt(s[0])

        y=Integer.parseInt(s[1])

        x=Integer.parseInt(s[2])

        if(!ViewL2.isL2(z+", "+y) || !ViewL2.isL2(z+", "+x) ||
        !ViewL2.isL2(y+", "+x))

        continue

        hash.setZ(z)

        hash.setY(y)

        hash.setX(x)

        index=hash.getHash()

        h[index].add(ar[i])

    }
    return runSQL(h)
}

```

/ Function to update the 3-itemset hash table using linkedlist array h */*

runSql(LinkedList<String> h)

```

{

    x=-1;

    Execute "insert into h3 values(?,?,?, ?, ?, ?)";

    flag=0;

    do{

        flag=0

        for i=0 to Hashing.table_size-1

        {

            if(h[i].isEmpty())

                ps.setNull(i+1,java.sql.Types.VARCHAR);

```

```

        else{

            item=(h[i].remove()).toString();

            ps.setString(i+1, item);

            flag=1;
        }
    }
    if(flag==1)

    x=ps.executeUpdate();

    }while(flag==1);

    return x

}

```

Class ViewL3

```

{

/* function to create l3 table after every transaction */

    insert(int minSup_count){

        table_size=Hashing.table_size;

        x=-1;

        Execute "TRUNCATE l3"

        PreparedStatement ps;

        for i=0; to table_size-1

            String feild=(String)(Character.toString((char) ('a'+i)))

            /* inserting those elements into l3 which are present in h3 and have a
            support count greater than minimum support count*/

            Execute "insert into l3 SELECT distinct("+feild+"),count("+feild+")
            FROM h3 where "+feild+" is not null group by "+feild+"
            having count(*) >= ?;"

            setInt(1, minSup_count);

        end of loop
    }
}

```

```

        return x
    }

    /* function to check if an element is present in the 2 itemset combinations */

    isL3(int x) {
        rs = Execute "select item from l3 order by count";

        Creating arraylist ar

        while(rs.next())
        {
            itemSet=rs.getString(1);

            tmp[]=itemSet.split(",");

            if( Integer.parseInt(tmp[0])==x || Integer.parseInt(tmp[1])==x ||
                Integer.parseInt(tmp[2])==x)

                ar.add(itemSet)

        }

        return ar
    }
}

Class ItemSet_2
{
    /* Function to find the 2 itemset combination of elements present in transaction string*/

    combi() {

        item[]=sort();

        n=item.length;

        sz=n*(n-1)/2;

        ar[]=new String[sz];

        for i=0,c=0 to i<n-1

            { for j=i+1to n-1

```

```

        ar[c]=item[i]+","+item[j];

        j++;

        c++;

    }

    i++;

}

return ar

}

```

/ function to sort the transaction string and to remove duplications*/*

```

sort() {

    Creating arraylist ar

    s[]=this.st.split(",");

    for(String x:s)

        ar.add(Integer.valueOf(x));

    Collections.sort(ar);

    i=0;

    while(i<ar.size()-1){

        if(ar.get(i).equals(ar.get(i+1)))

            ar.remove(i);

        else i++;

    }

    return ar.toArray(new Integer[ar.size()]);

}

```

/ Class ItemSet_3 inherits function sort() from class ItemSet_2*/*

```

class ItemSet_3 extends ItemSet_2

{

```

```
/* function to find the various 3 itemset combinations of elements in a
transaction string*/
```

```
    combi() {
        item[]=sort();
        n=item.length;
        sz=n*(n-1)*(n-2)/6;
        ar[]=new String[sz];
        for i=0,c=0 to i<n-2
            for j=i+1 to j<n-1
                for k=j+1 to k<n{
                    ar[c]=item[i]+","+item[j]+","+item[k]
                    k++
                    c++
                }
                j++ }
                i++ }

        return ar;
    }
}

class Hashing {
    table_size=13;

    getHash() {

/* value of x,y,z has been set in the setX,setY,setZ function which receive the
values from the calling function*/
```

```

        n=(z*100)+(y*10)+x

        return n%table_size

    }

    /* function to set the value of x*/

    setX(int x) {

        this.x = x;

    }

    /* function to set the value of y*/

    setY(int y) {

        this.y = y;

    }

    /* function to set the value of z*/

    setZ(int z) {

        this.z = z;

    }

}

```

4.2: Basic Modules

In order to bring readability and simplicity to the code, it has been divided into three primary modules:

- Driver module
- Table Set
- Connection Class
- GUI Module

1. Driver Module

The driver module displays the wide range of products a user can select from. This module accepts the choice as entered by the user. It then calls the function isL1(item) which is

present in the viewL1 class under the module Table. If the item selected by the user is present in the L1 table (frequent one-item set), then the chance of two item set combination of that item with some other item in the L2 table (frequent two-item set) is there. If the item is present in L1, then the function insert(item) is called. This function is present in the ResultSetTable class. The function insert(item) inserts the item string in the “reset” table. Then the function checks whether any two item-set combination of the item string is present in the L2 table by calling the isL2(item) function. Any combination of the item string found is then added to the “reset” table. Similar procedure is followed for the three item set table by calling the isL3(item) function. The “reset” table now comprises of the item as selected by the user as well as the two item and three item option he/she might like to opt for. This table is displayed and the user now has three option he can choose from.

- i. He can either choose from the list of two or three items.
- ii. He can cancel the entire transaction.
- iii. He can buy the item he had initially opted for.

If he chooses option i, then the above mentioned process is repeated once again. Choosing option iii leads him to the transaction gateway whereas choosing the second option, he can exit from the application.

The transaction table now gets updated with the transaction string and all the functions under the table module are called.

2. Table Module

This module is responsible for modifying all those tables which are under the direct application of the apriori algorithm. This module is further subdivided into six sub modules. The different modules affects the different tables used in the program.

- *One item-set Candidate Table(c1)*

The one item set candidate table consists of all the items which are present in the database and the corresponding frequency of their occurrence in all of the transactions. The C1 module contains the Table C1 Class. This class contains the function insert(String). The function takes the transaction string as the parameter and all the items are extracted from it and stored in an array. The array is then used to update the c1 table.

- Frequent one item-set Table(l1)

This module consists of the ViewL1 Class and has two functions insert(int) and isL1(int). The isL1(int) function was previously called by the MainClass in Driver Module for the purpose of updating the “reset” table.

The function insert(int) takes as a parameter the minimum support count and uses it to update the l1 view. The l1 view is now updated with all those items present in the c1 table and having a frequency greater than the minimum support count. The corresponding frequencies of these items are also included in the l1 view.

- Hash Table(h2)

This module consists of the class TableH2 and has two functions insert(String) and runSQL(LinkedList<String>[]). The insert(String) function takes as input the transaction string and uses it to create a two item combination (i,j) of elements present in the transaction string such that $i < j$ and $i, j \in l1$. The function also creates an array of type linked list which consists of the two item elements which satisfy the aforementioned conditions. The item combinations are mapped into the array using a hash function. The function runSQL(LinkedList<String>[]) now uses the array it takes as the parameter and uses it to enter the elements into the hash table h2.

- Frequent two item set table(l2)

This particular module comprises of class ViewL2 and has three functions insert(int), isL2(String), isL2(int). The function insert(int) is used to take the minimum support count as the parameter and enters those elements into the l2 table whose number of occurrences in the hash table(h2) is greater than or equal to the minimum support count. The function isL2() is an overloaded function taking an integer and a string as parameter respectively. The isL2(int) function is being called from the driver module for updating the “reset” table. The isL2(String) function takes a two combination item set as an input and checks whether it is present in the l2 table and returns a Boolean value to the calling function.

- Hash Table(h3)

This module is similar to that of the class TableH2. The class TableH3 works with combination of three item set whereas the h2 table worked with combination of two item sets.

- Frequent three item set table(l3)

This particular module comprises of class ViewL3 and has two functions insert(int), isL3(int). The function insert(int) takes as parameter the minimum support count and enters those combinations of three elements into the l3 table whose number of occurrences in the hash table(h3) is greater than or equal to the minimum support count. The function isL3(int) is called from the Driver Module for updating the “reset” table.

- Result Set Table (reset)

The ResultSetTable class is present in this module and it has two functions, insert(int) and show(connection). The insert(int) function is called from the Driver Module. This function takes as parameter the item chosen by the user and it is used to call the isL2(int) and the isL3(int) functions. The result from these are used to update the “reset” table. If the result returned by the isL2(int) and isL3(int) functions are not null , then the function show(connection) is called from the insert(int) function itself. It displays the “reset” table to the user.

3. Connection Module

This module deals with two connections. It connects the back end code written in Java to the MySql server & Workbench so that we can access and store the data using SQL queries written in Java as and when required. it is implemented by importing the connection package in Java.

The second connection is the apache Tomcat connection which connects the bac end Java code to the web server. It also provides a platform to run JSP.

4. GUI Module

This module consists of three java classes and HTML and JSP pages. This module is used to create the user interface. The various items that are in the item list are displayed on the web server and when an user selects one item a JSP function is called which creates a dynamic table and lists all the possible 2 and 3 itemset combination that the user might consider.

4.3: Procedural Design

4.3.1: PROCESS DIAGRAM:

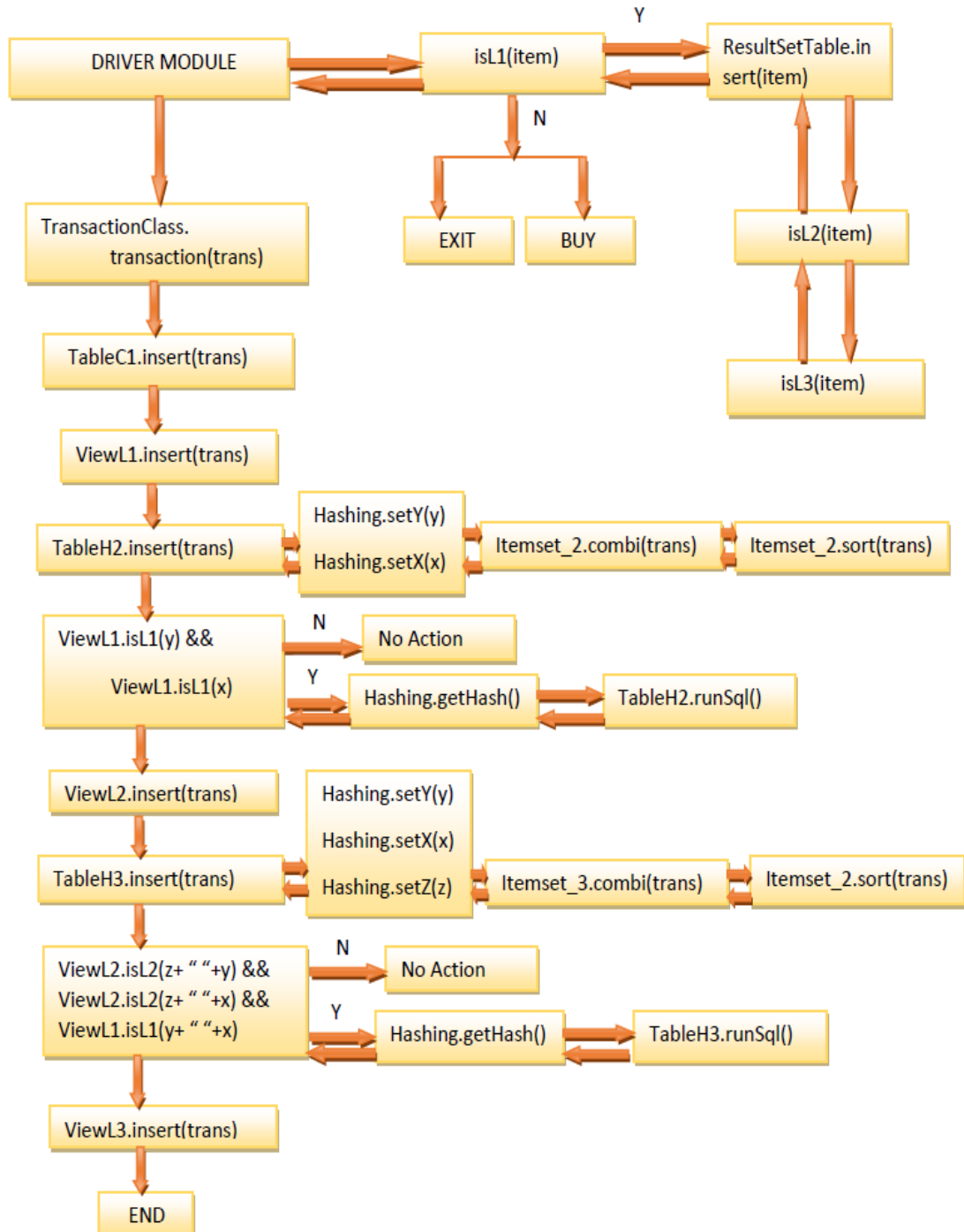


FIG: 4.3.1(a) Process Diagram

4.3.2: **DATA STRUCTURE:**

- The two itemset hash table(h2) is used to store a combination of two items (y,x). Each of these items are picked up from the l1 table and y is always less than x. The hash function used for the purpose of storing the items is $[(y*10)+x] \bmod 13$.
- The three item set hash table(h3) is used for the purpose of storing a combination of three items (z,y,x). Each of these items need to be a part of the l1 table. Also all possible two combination of items x,y,z needs to be present in the l2 table and $z < y < x$ for all the items. The hash function used for the purpose of storing the items in the hash table is $[(z*100)+(y*10)+x] \bmod 13$.
- A linked list array(h) is used for the purpose of storing the two itemset combinations and the three itemset combinations in it which will later be entered into the hash table h2 and h3 respectively.
- Array a1 is used to store items from the current transaction string.
- List ar1<> contains all the items from the current transaction string and is used to remove duplication. It contains the two item and the three item combination of the items.
- Array item[] contains the sorted list of items without any duplicate elements present.
- Each of the two items and three items present in array ar[] are stored in the String array s[]. The two items and three items separated by comma are extracted and put into the different positions in the array.

4.3.3: **ALGORITHM DESIGN:**

Step 1: Start

Step 2: trans <- “ ”

Step 4: Display item options to the user

Step 5: if an item is selected:

Step 5(a): Check if item is present in l1

Step 5(b):if true, insert item into the “reset” table

Step 5(c): Check if the item is present in the l2

Step 5(d): if true insert the two item-set combination of the item with some other item not the “reset” table else break

Step 5(e): Check if the item is present in l3

Step 5(f): if true, insert the 3-item set combination of the item with two other items into the “reset” table else break

Step 5(g): Show “reset” table

Step 6: If user selects an option, goto step 5(a) else if user selects to cancel the transaction, goto step 15

Step 7: If the user opts to continue shopping, then goto step 3.

Step 8: Generate transaction ID and update the transaction table with the item string

Step 9: Updating 1-itemset candidate table c1

Step 9(a): Extract all the items from the current transaction string and insert into an array a1

Step 9(b): $i \leftarrow 1$

Step 9(c): Loop 1 to a1.length

Step 9(d): increase frequency column of that item in c1 which matches with a1[i]

Step 9(e): $i \leftarrow i+1$

Step 9(f): loop end

Step 10: Generate l1 from c1

Step 10(a): Insert all those tuples in c1 into l1 whose frequency in $c1 \geq$ minimum support count

Step 11: Updating two item-set hash table (h2)

Step 11(a): Remove duplication from transaction string

Step 11(a).1: Create a list ar1 containing all the items in transaction string

Step 11(a).2: $i \leftarrow 0$

Step 11(a).3: loop $i < ar1.length-1$

Step 11(a).4: if $ar1(i) == ar1(i+1)$ then remove $ar(i)$

Step 11(a).5: else i++

Step 11(a).6: end loop

Step 11(b): Sort the list in ascending order

Step 11(c): Find the various two item-set combination

Step 11(c).1: Array item 1 contains the sorted list

Step 11(c).2: $n \leftarrow \text{item.length}$

Step 11(c).3: $st \leftarrow [n*(n-1)]/2$

Step 11(c).4: $i \leftarrow 0, c \leftarrow 0$

Step 11(c).5: loop $i < n-1$

Step 11(c).5.1: $j \leftarrow i+1$

Step 11(c).5.2: loop $j < n$

Step 11(c).5.3: $ar[c] \leftarrow \text{item1}[i] + \text{,”} + \text{item}[j]$

Step 11(c).5.4: $j \leftarrow j+1, c \leftarrow c+1$

Step 11(c).5.5: end inner loop

Step 11(c).6 : $i \leftarrow i+1$

Step 11(c).7: end outer loop

Step 11(d): Creating a linked list array(h) to temporarily store data

Step 11(d).1: $i \leftarrow 0$

Step 11(d).2: loop $i < ar1.length$

Step 11(d).3: $String\ s[] = ar1[i].split(\text{,”})$

Step 11(d).4: $y \leftarrow s[0], x \leftarrow s[1]$

Step 11(d).5: if x or y is not present in l1, continue Step 11(c).

Step 11(d).6: Calculating hash value using function
 $[(y*10)+x] \bmod 7$

Step 11(d).7: $h[index] \leftarrow ar1[i]$

Step 11(d).8: $i \leftarrow i+1$

Step 11(d).9: end loop

Step 11(e): Insert into two item set hash table (h2) using values from the linked list array h

Step 12: Generate l2 from h2

Step 12(a): All the items from h2 having number of occurrences \geq minimum support count are inserted into the l2 table along with their corresponding frequencies

Step 13: Updating 3-item set hash table (h3)

Step 13(a): Remove all the duplication from the transaction string

Step 13(b): Follow steps 11(a).1 to 11(b)

Step 13(c): Find various 3 item set combination

Step 13(c).1: Array item2[] contains a sorted list

Step 13(c).2: $n \leftarrow \text{items2.length}$

Step 13(c).3: $\text{sz} \leftarrow \lfloor n*(n-1)*(n-2)/6 \rfloor$

Step 13(c).4: Creating array ar[] of size sz

Step 13(c).5: $i \leftarrow 0, c \leftarrow 0$

Step 13(c).6: loop $i < n-2$

Step 13(c).6.1: $j \leftarrow i+1$

Step 13(c).6.2: loop $j < n-1$

Step 13(c).6.2.1: $k \leftarrow j+1$

Step 13(c).6.2.2: loop $k < n$

Step 13(c).6.2.3: $\text{ar}[c] \leftarrow \text{item2}[i] + "," + \text{item2}[j] + "," + \text{item2}[k]$

Step 13(c).6.2.4: $k \leftarrow k+1$

Step 13(c).6.2.5: $c \leftarrow c+1$

Step 13(c).6.2.6: end loop k

Step 13(c).6.3: $j \leftarrow j+1$

Step 13(c).6.4: end loop j

Step 13(c).7: $i \leftarrow i+1$

Step 13(c).8: end loop i

Step 13(d): Creating linked list array(h) to temporarily store data

Step 13(d).1: $i \leftarrow 0$

Step 13(d).2: loop $i < \text{ar.length}$

Step 13(d).3: String $s[] = \text{ar}[i].\text{split}(",")$

Step 13(d).4: $z \leftarrow s[0], y \leftarrow s[1], x \leftarrow s[2]$

Step 13(d).5: if the combination (z,x) or (z,y) or (y,x) is not present in l2 continue

Step 13(d).6: calculating hash value using function $[(z*100)+(y*10)+x] \bmod 7$

Step 13(d).7: $h[\text{index}] \leftarrow \text{ar}[i]$

Step 13(d).8: $i \leftarrow i+1$

Step 13(d).9: end loop i

Step 13(e): Insert into 3 item set hash table(h3) using values from the linked list array

Step 14: Generate l3 from h3

Step 14(a): All the items from h3 having number of occurrences \geq minimum support count are inserted into the l3 table along with their corresponding frequencies

Step 15: End

4.4: User Interface Design

This module shows how the items are displayed on the web server and how the transaction is taking place.

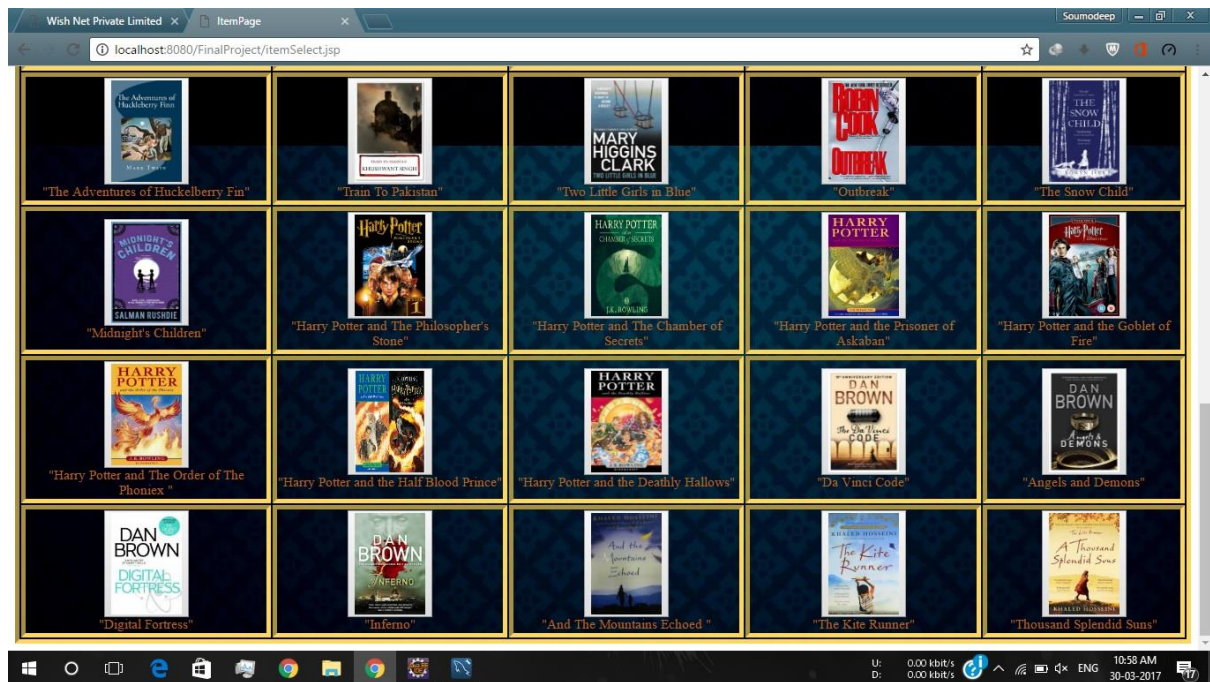


Fig: 4.4(a)

User selects Harry Potter and the
Philosopher's Stone



ResultSet Table is consulted and the
following web page is generated.



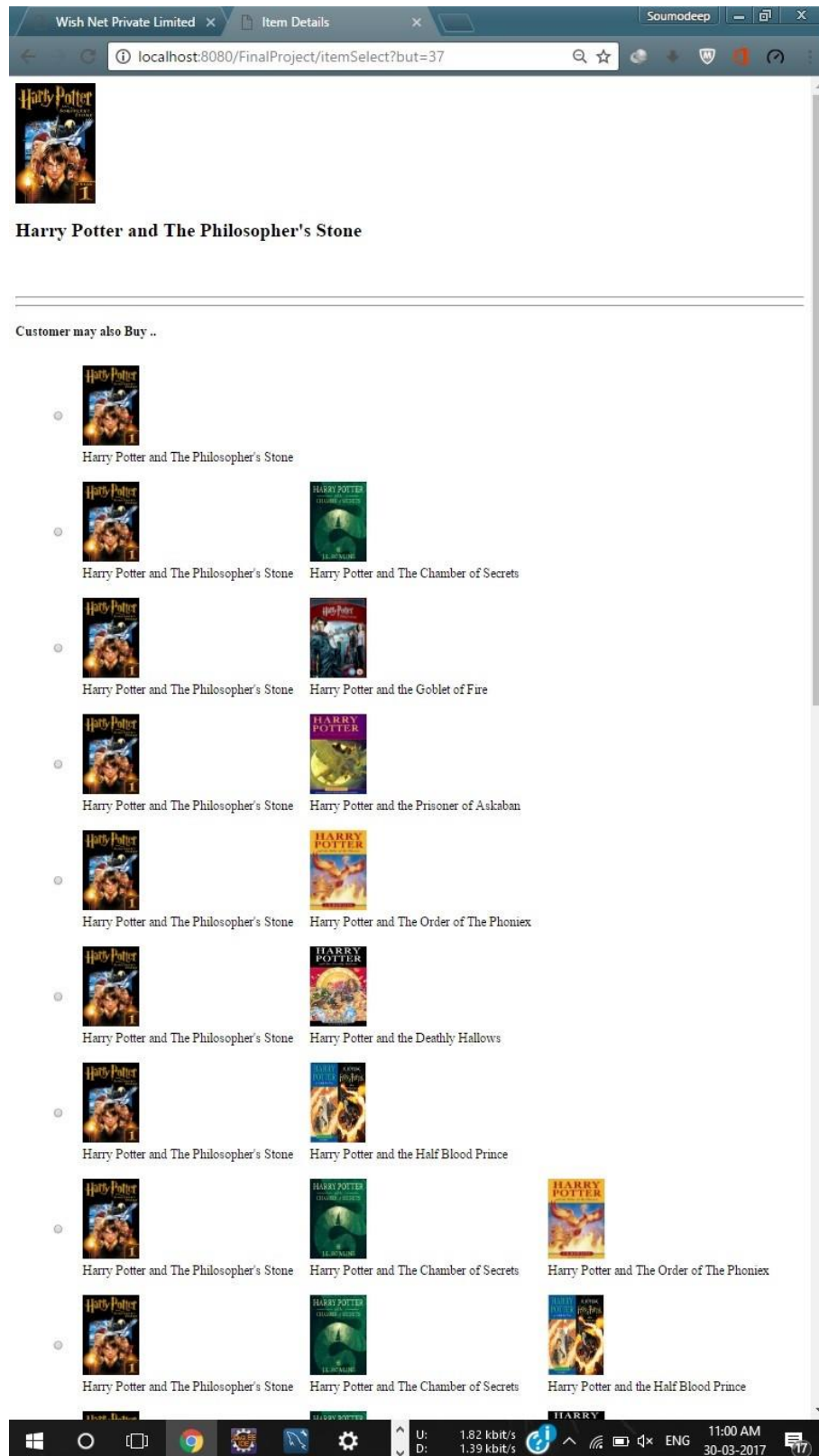


Fig 4.4(b)

User selects Harry Potter and the Philosopher's Stone, Harry Potter and the Deathly Hallows and Harry Potter and The Order of the Phoenix option

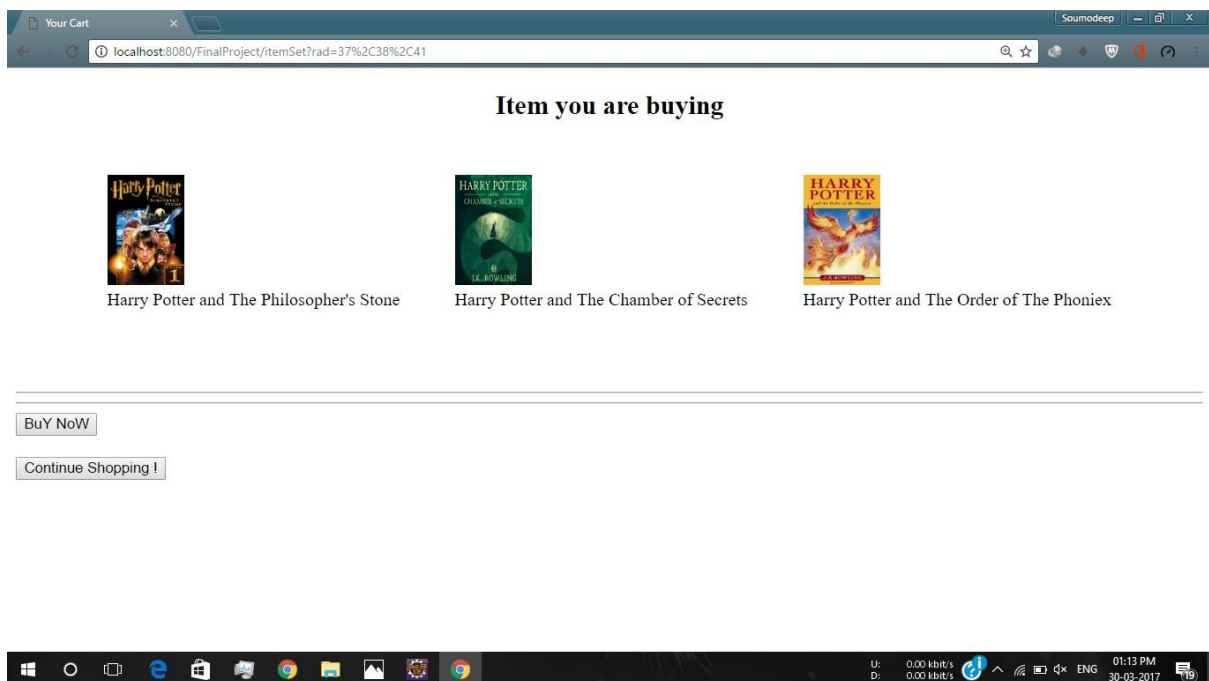


Fig: 4.4(c)

User Selects Buy Now Option. The Apriori Algorithm runs in the background and the transaction is being processed.



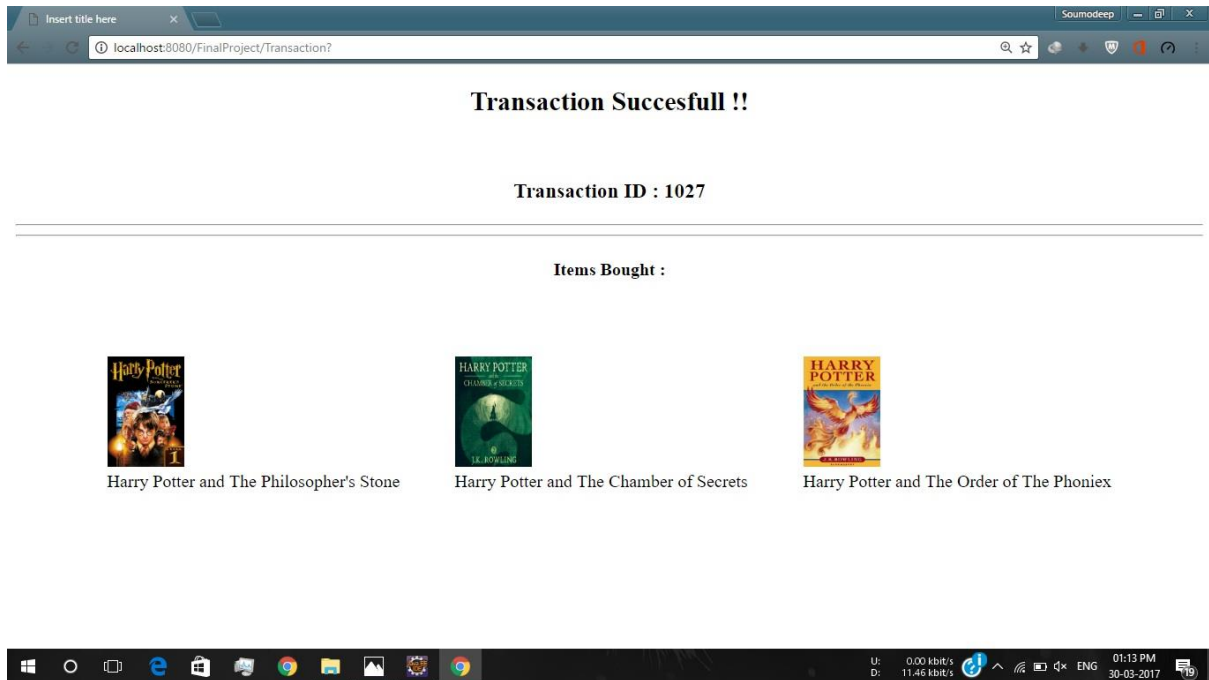


Fig: 4.4(d)



Transaction is completed and Transaction ID is generated

4.5: Test Case Design

System Integrity of a software can be defined as the state of a system where it is performing its intended function without being degraded or impaired by changes or disruptions in it's internal or external environment.

To check the system integrity of our software we have designed 4 test cases by changing 2 fields:

- i. Hash Function
- ii. Hash Table Size

1. Various hash functions can be used for hashing and no hash function is ideal that is collision free. Our aim of using a particular hash function is to get minimum collisions as possible as well as keeping the hash table compact to save space.

In this paper we have tried 2 hash functions –

- a. Concatenation function: it concatenates 2 or 3 itemsets, as available, and then mod operation is performed using the hash table size.
 - b. General Hash Function: A function has been defined by us and used. The function is $((z*100)+(y*10)+x) \bmod \text{table size}$ for 3 itemsets combination and $((y*10)+x) \bmod \text{table size}$ for 2 itemsets combinations. Here z, y and x are index value of items and $z < y < x$.
2. The greater the hash table size the less is the number of collisions happening. But by increasing the hash table size we will get many unused spaces which will lead to unnecessary storage space wastage. Hence a trade-off between number of collisions and space wastage has to be made while choosing the hash table size. Furthermore, the hash table size is mostly chosen a prime number this leads to comparatively less number of collisions.

In our project we have used tested the outcome using 2 hash table sizes – 13 and 17.

Thus the 4 test cases arising by combining the above two factors are:

- i. Using concatenation hash function and keeping hash table size 13
- ii. Using the general hash function and keeping hash table size 13
- iii. Using concatenation hash function and keeping the hash table size 17
- iv. Using the general hash function and keeping the table size 17

These four cases have been tested and their result has been discussed in Test Report.

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1: Implementation Approaches

5.2: Coding Details and Code Efficiency

5.2.1: Code Details

5.2.2: Coding Efficiency

5.3: Modifications and Improvements

IMPLEMENTATION AND TESTING

5.1: Implementation Approaches

Implementation of this project required the following steps:

- Requirement Analysis
- Identifying the requirements
- Planning
- Scheduling
- Design
- Project Designing
- Implementation and Unit Testing
- Coding and Debugging
- Testing
- Integration and System Testing
- Module Integration
- System Testing
- Operation
- Checking Operation
- Evaluation

A test plan is a document that contains a complete set of test cases for a system, along with other information about the testing process. The test plan should be returned long before the testing starts. Test plan identifies:

1. A task set to be applied as testing commences,
2. The work products to be produced as each testing task is executed.
3. The manner, in which the results of testing are evaluated, recorded and reuse when regression testing is conducted.

In some cases the test plan is indicated with the project plan. In others the test plan is a separate document. The test report is a record of the testing performed. The testing report enables the acquirer to assess the testing and its results. The test report is a record of the testing performed. The testing report enables the acquirer to assess the testing and its results.

In this project, we have first analysed all the various requirements it has. Once the requirements are analysed, we made a chart of those data structures we will require in the program as well as the software and hardware requirements the project has. The third step of the project is the planning and the scheduling phase where the project was divided into basic modules and different time limit was given to each of the modules for the completion. The pert chart and the gantt chart also forms a part of this phase. Once a certain amount of time was dedicated for the completion of every module of the project, it is now time to start

designing the outline of the code. This involves forming the skeleton of the project and planning the basic concepts which will be implemented in the project. Under the implementation and testing phase, each of the modules were separately tested and the errors and bugs found were corrected. The code was written and simultaneously code was debugged. All the other tests were performed on the project and the bugs encountered along the process were debugged or removed. After the various modules were separately tested, it is now time to integrate the modules into a system. Since the previous tests were carried out on the modules separately, an integrated test is now performed on the system as a whole and simultaneous bugs will be removed. Project evaluation is a systematic and objective assessment of an ongoing or completed project. The aim is to determine the relevance and level of achievement of **project** objectives, development effectiveness, efficiency, impact and sustainability. This is the last phase of the project implementation process.

5.2: Coding Details and Coding Efficiency

5.2.1 Coding Details:

1.Package Name: Connection_MySQL

Class name: ConnectionClass

```
import java.sql.Connection;

import java.sql.DriverManager;

public class ConnectionClass
{
    public static Connection dbcon()
    {
        String DB_URL="jdbc:mysql://localhost/Schema?autoReconnect=true&useSSL=false";

        String user="user"; //MySQL user name

        String password="user"; //MySQL password

        Connection con=null;

        Try
        {
            Class.forName("com.mysql.jdbc.Driver");
            con=DriverManager.getConnection(DB_URL,user,password);
        }
    }
}
```

```

        catch(ClassNotFoundException ex)
        {
            System.out.println("Error: unable to load driver class!");
            System.exit(1);
        }
        catch(Exception e)
        {
            System.err.println("Error in DBconnection\n"+e.getMessage());
        }
    return con;
    }

}

```

2. Package Name: driverModule

Class Name: MainClass

```

import itemSetTable.ResultSetTable;

import tableLen1.TableC1;

import tableLen1.ViewL1;

import tableLen2.TableH2;

import tableLen2.ViewL2;

import tableLen3.TableH3;

import tableLen3.ViewL3;

import transactionTable.TransactionClass;

import java.io.*;

import java.sql.*;

public class MainClass

{

    public static void main(String[] args) throws SQLException, NumberFormatException,
    IOException

```



```

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

String trans = "";int a1;
do
{
    System.out.println(" Items : \n 1.Lux \n 2.Margo \n 3.Fiama DeWills \n 4.Neveda \n
    5.Jhonsons Baby\n 6. Mysore \n 7. detol \n 8. Patanjali\n 9. DOve\n10. Pears");
    int item=Integer.parseInt(br.readLine()); //accepting user's choice of item
    System.out.println(" The item you choose : ");
    System.out.println(" Your action : \n 1.Select \n 2.Buy Now ");
    a1=Integer.parseInt(br.readLine());
    switch(a1)
    {
        case 1: if(ViewL1.isL1(item)==true) /*checking if item is present in
                                                frequent 1 itemset table*/

            {

                ResultSetTable res=new ResultSetTable();

                res.insert(item);

                System.out.println("Enter choice : "); /* Taking users choice
                                                            recommended options*/

                String s=br.readLine();
                trans+=s+", ";
                System.out.println("Items to buy : "+trans);
                break;
            }
        case 2: trans+=item+", ";
                System.out.println("Item to buy : "+trans);
                break;

    }

    System.out.println("Continue to check out..(y/n)");

}while((char)br.readLine().charAt(0)=='n');

trans=trans.substring(0,trans.lastIndexOf(','));

TransactionClass obT=new TransactionClass();

// Adding the transaction string to the transaction table

```

```

int x=obT.transaction(trans);

TableC1 ob1=new TableC1();

// Updating the 1 itemset candidate table

x=ob1.insert(trans);

// Updating the frequent 1 itemset table

ViewL1 ob2=new ViewL1();

    int y=ViewL1.avg(); /*finding out the average of the count and sending it as
                        the dynamic minmum support count*/

    x=ob2.insert(y);

    System.out.println(x+" "+y);

//Adding all two itemset combination in the transaction string to the 2 itemset hash table

    TableH2 ob3=new TableH2();

    x=ob3.insert(trans);

    System.out.println(x);

// Updating the frequent 2 itemset table

    ViewL2 obl2=new ViewL2();

    y=ViewL2.avg();

    x=obl2.insert(y);

    System.out.println(x+" "+y);

/* Adding all three itemset combination in the transaction string to the 3 itemset hash
table*/

    TableH3 ob4=new TableH3();

    x=ob4.insert(trans);

    System.out.println(x);

//Updating frequent 3 itemsets table

    ViewL3 obl3=new ViewL3();

    y=ViewL3.avg();

    x=obl3.insert(y);

```

```

        System.out.println(x+" "+y);

        System.out.println(" Thank you....visit again");

    }

}

```

3.Package name: hash

Class name: Hashing

```

public class Hashing
{
    private int x,y,z;
    public static final int table_size=13;
    public Hashing()
    {
        x=y=z=0;
    }

    public int getHash()
    {
        int n= (z*100)+(y*10)+(x); // hash function
        return n%table_size;
    }

    public void setX(int x)
    {
        this.x = x;
    }

    public void setY(int y)
    {
        this.y = y;
    }

    public void setZ(int z)
    {
        this.z = z;
    }
}

```

4. Package Name: itemSet

Class name 1: ItemSet2

```

import java.util.ArrayList;
import java.util.Collections;

public class ItemSet_2
{
    String st;

    public ItemSet_2(String st)
    {
        this.st = st;
    }

    //finding the all possible two item combination of elements present in transaction string

    public String[] combi()
    {
        Integer item[]=sort();
        int n=item.length;
        int sz=n*(n-1)/2;
        String ar[]=new String[sz];
        for(int i=0,c=0;i<n-1;i++)
            for(int j=i+1;j<n;j++,c++)
            {
                ar[c]=item[i]+","+item[j];
            }
        return ar;
    }

    // Sorting the array containing items in ascending order
    Integer [] sort()
    {
        ArrayList<Integer> ar=new ArrayList<Integer>();
        String s[]=this.st.split(",");
        for(String x:s)
            ar.add(Integer.valueOf(x));
        Collections.sort(ar);
        int i=0;
        while(i<ar.size()-1)
        {
            if(ar.get(i).equals(ar.get(i+1)))

```

```

        ar.remove(i);
        else i++;
    }
    return ar.toArray(new Integer[ar.size()]);
}

}

```

Class name 2: ItemSet3

```

public class ItemSet_3 extends ItemSet_2
{
    //inherits from class ItemSet_2
    public ItemSet_3(String st) {
        super(st);
    }

    public String[] combi()
    {
        Integer item[]=sort();
        int n=item.length;
        int sz=n*(n-1)*(n-2)/6;
        String ar[]=new String[sz];    /*forming all three item
                                        combination*/

        for(int i=0,c=0;i<n-2;i++)
            for(int j=i+1;j<n-1;j++)
                for(int k=j+1;k<n; k++, c++)
                {
                    ar[c]=item[i]+","+item[j]+","+item[k];
                }

        return ar;
    }
}

```

5.Package Name: itemSetTable

Class Name: ResultSetTable

```

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

```

```

import java.sql.SQLException;

import java.util.ArrayList;

import tableLen2.ViewL2;

import tableLen3.ViewL3;


public class ResultSetTable
{
    public void insert(int item)
    {
        try
        {
            Connection con=connection_MySQL.ConnectionClass.dbcon();
            String sql="TRUNCATE resset";
            con.prepareStatement(sql).execute();
            sql="insert into resset values(?)";
            PreparedStatement ps=con.prepareStatement(sql);
            ps.setString(1,Integer.toString(item));
            ps.execute();
            ArrayList<String> ar=ViewL2.isL2(item);
            if(ar==null)
                return;
            for (String res : ar)
            {
                ps.setString(1,res);
                ps.execute();
            }

            ar=ViewL3.isL3(item);
            if(ar==null)
                return;
            for (String res : ar)
            {
                ps.setString(1,res);
                ps.execute();
            }
            show(con);
            con.close();
        }
    }
}

```

```

    }

    catch (SQLException e)
    {
        e.printStackTrace();
    }

}

public void show(Connection con) throws SQLException
{
    String sql="SELECT * FROM resset order by length(itemSet)";
    ResultSet rs = con.prepareStatement(sql).executeQuery();
    while(rs.next())
    {
        System.out.println(rs.getString(1));
    }
}
}

```

6.Package Name: tableLen1

Class Name 1: TableC1

```

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class TableC1
{
    // updating the 1 itemset candidate table

    public int insert(String trans)
    {
        Connection con= connection_MySQL.ConnectionClass.dbcon(); /*creating
                                                                    Connection*/

        int x=-1;
        String ar[]=trans.split(","); //splitting the transaction string
        try

```

```

    {
        PreparedStatement ps=con.prepareStatement("Update c1 set
                                                    sup_count=sup_count+1 where
                                                    item=?");

        for(int i=0;i<ar.length;i++)
        {
            ps.setInt(1,Integer.parseInt(ar[i]));
            x=ps.executeUpdate();
        }
        con.close();
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
    return x;
}
}

```

Class Name 2: ViewL1

```

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

public class ViewL1
{

    public int insert(int minSup_count)
    {

        Connection con=connection_MySQL.ConnectionClass.dbcon();

        int x=-1;

        String sql="Create or Replace View L1 as select item,sup_count as count from c1 where
                    sup_count>=?";           //sql query to create view l1

        Try
    }
}

```



```

    {
        PreparedStatement state=con.prepareStatement(sql);
        state.setInt(1,minSup_count );
        x=state.executeUpdate();
        con.close();
    }

    catch (SQLException e)

    {
        e.printStackTrace();
    }

    return x;
}

/* function to find an item in the view l1 */
public static boolean isL1(int x) throws SQLException
{
    Connection con=connection_MySQL.ConnectionClass.dbcon();
    String sql="select * from l1 where item=?";
    ResultSet rs=null;
    PreparedStatement state=con.prepareStatement(sql);
    state.setString(1,Integer.toString(x));
    rs=state.executeQuery();
    boolean val=rs.next();
    con.close();
    return val;
}

}

```

7.Package Name: tableLen2

Class Name 1: TableH2

```

package tableLen2;

import java.sql.Connection;

import java.sql.PreparedStatement;

```

```

import java.sql.SQLException;

import java.util.LinkedList;

import hash.Hashing;

import itemSet.ItemSet_2;

import tableLen1.ViewL1;

public class TableH2

{

    // function to insert into the 2 itemset hash table

    public int insert(String trans) throws SQLException

    {

        ItemSet_2 iset=new ItemSet_2(trans);
        String ar[]=iset.combi();    // set of all combination
        LinkedList h[];
        h=new LinkedList[Hashing.table_size];
        for(int i=0;i<h.length;i++)
        {
            h[i]=new LinkedList<String>();
        }
        Hashing hash=new Hashing();
        for(int i=0;i<ar.length;i++)
        {
            String s[]=ar[i].split(",");    /*splitting the 2 itemset combination and storing it in
                                             consecutive array memory location*/
            int y=Integer.parseInt(s[0]);
            int x=Integer.parseInt(s[1]);
            /* checking if the items are present in the l1 table*/
            If (!ViewL1.isL1(y) || !ViewL1.isL1(x))
                continue;

            hash.setY(y);
            hash.setX(x);
            int index=hash.getHash();    //finding out the hash index
            // adding the 2 itemset combination to the linked list array to store temporarily
            h[index].add(ar[i]);
        }
        return runSQL(h);

    }

}

```

```
/* inserting the 2 itemset combination into the H2 using the linked array generated in the  
insert function */
```

```
public int runSQL(LinkedList<String>[] h) throws SQLException
```

```
{  
  
    Connection con=connection_MySQL.ConnectionClass.dbcon();  
    int x=-1;  
    int size=Hashing.table_size;  
    String sql="insert into h2 values(?,?,?, ?, ?, ?)";  
    int flag=0;  
    do  
    {  
        flag=0;  
        PreparedStatement ps=con.prepareStatement(sql);  
        for(int i=0;i<size;i++)  
        {  
            if(h[i].isEmpty())  
                ps.setNull(i+1,java.sql.Types.VARCHAR);  
            else  
            {  
                String item=(h[i].remove()).toString();  
                ps.setString(i+1, item);  
                flag=1;  
            }  
        }  
        if(flag==1)  
            x=ps.executeUpdate();  
    }  
    while(flag==1);  
    con.close();  
    return x;  
}
```

Class Name 2: ViewL2

```
import java.sql.Connection;
```

```
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
```

```

import java.sql.SQLException;

import java.util.ArrayList;

import hash.Hashing;

public class ViewL2
{
    public int insert(int minSup_count) throws SQLException
    {
        final int table_size=Hashing.table_size;

        Connection con=connection_MySQL.ConnectionClass.dbcon();

        int x=-1;

        String sql="TRUNCATE l2" ;  /*deleting all the entries in the l2 table */

        con.prepareStatement(sql).execute();

        PreparedStatement ps=null;

        /* loop to create new l2 after every transaction*/

        for(int i=0;i<table_size;i++)
        {
            String feild=(String)(Character.toString((char) ('a'+i)));
            sql="insert into l2 SELECT distinct("+feild+"),count("+feild+") FROM h2 where
                "+feild+" is not null group by "+feild+"
                having count(*) >= ?;";

            ps=con.prepareStatement(sql);
            ps.setInt(1, minSup_count);
            x=ps.executeUpdate();

        }

        con.close();

        return x;
    }

    /* function to find whether a particular combination exists in the l2 table */

    public static boolean isL2(String x) throws SQLException

```

```

{
    Connection con=connection_MySQL.ConnectionClass.dbcon();

    String sql="select item from l2 where item=?";

    ResultSet rs=null;

    PreparedStatement state=con.prepareStatement(sql);

    state.setString(1,x);

    rs=state.executeQuery();

    boolean val=rs.next();

    con.close();

    return val;
}

/* function to find whether a particular item is present in the 2 itemset combination and return
that combination */

public static ArrayList<String> isL2(int x) throws SQLException
{
    Connection con=connection_MySQL.ConnectionClass.dbcon();

    String sql="select item from l2 order by count";

    PreparedStatement smt=con.prepareStatement(sql);

    smt.setString(1, Integer.toString(x+1));

    ResultSet rs=smt.executeQuery();

    ArrayList<String> ar=new ArrayList<String>();

    int c=0;

    while(rs.next() && c!=visibleNo)
    {
        String itemSet=rs.getString(1);
        String tmp[]=itemSet.split(",");
        if( Integer.parseInt(tmp[0])==x || Integer.parseInt(tmp[1])==x)
        {
            ar.add(itemSet);c++;
        }
    }
}

```

```

    }

    }

    return ar;

    }

}

/*function to find the average of the item count*/

public static int avg() {

    Connection con=connection_MySQL.ConnectionClass.dbcon();

    String sql="select avg(count) from l2";

    ResultSet rs=null;

    PreparedStatement state;

    try {

        state = con.prepareStatement(sql);

        rs=state.executeQuery();

        rs.next();

        System.out.println("AVG L2 : "+rs.getFloat("avg(count)"));

        return Math.round(rs.getFloat("avg(count)"));

    } catch (SQLException e) {

        return 0;

    }

}

```

8.Package Name: tableLen3

Class Name 1: TableH3

```

package tableLen3;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.SQLException;

```

```

import java.util.LinkedList;

import hash.Hashing;

import itemSet.ItemSet_2;

import itemSet.ItemSet_3;

import tableLen2.ViewL2;

public class TableH3

{

    public int insert(String trans) throws SQLException

    {

        ItemSet_2 iset=new ItemSet_3(trans);
        String ar[]=iset.combi();    // set of all combination
        LinkedList<String> h[];
        h=new LinkedList[Hashing.table_size];
        for(int i=0;i<h.length;i++)
        {
            h[i]=new LinkedList<String>();
        }
        Hashing hash=new Hashing();
        for(int i=0;i<ar.length;i++)
        {
            String s[]=ar[i].split(",");
            int z=Integer.parseInt(s[0]);
            int y=Integer.parseInt(s[1]);
            int x=Integer.parseInt(s[2]);
            // Condition to check if all the possible two combination of x,y,z is present in l2
            if(!ViewL2.isL2(z+", "+y) || !ViewL2.isL2(z+", "+x) || !ViewL2.isL2(y+", "+x))
                continue;
            hash.setZ(z);
            hash.setY(y);
            hash.setX(x);
            int index=hash.getHash();
            h[index].add(ar[i]);    /*add the elements in the linked
                                   list array h */

        }
        return runSQL(h);

    }

}

```

```

public int runSQL(LinkedList<String>[] h) throws SQLException
{
    Connection con=connection_MySQL.ConnectionClass.dbcon();

    int x=-1;

    //SQL statement to enter the three item combination into the hash table h3

    String sql="insert into h3 values(?,?,?, ?, ?, ?)";

    int flag=0;

    do
    {
        flag=0;
        PreparedStatement ps=con.prepareStatement(sql);
        for(int i=0;i<Hashing.table_size;i++)
        {
            if(h[i].isEmpty())
                ps.setNull(i+1,java.sql.Types.VARCHAR);
            else
            {
                String item=(h[i].remove()).toString();
                ps.setString(i+1, item);
                flag=1;
            }
        }
        if(flag==1)
            x=ps.executeUpdate();
    }
    while(flag==1);
    con.close();
    return x;
}
}

```

Class name 2: ViewL3

```

public class ViewL3
{

```


// function to check whether the item is present in the l3 table and display its combinations

```
public static ArrayList<String> isL3(int x) throws SQLException
{
    Connection con=connection_MySQL.ConnectionClass.dbcon();
    String sql="select item from l3 order by count";
    PreparedStatement smt=con.prepareStatement(sql);
    smt.setString(1, Integer.toString(x+1));
    ResultSet rs=smt.executeQuery();
    ArrayList<String> ar=new ArrayList<String>();
    int c=0;
    while(rs.next() && c!=visibleNo)
    {
        String itemSet=rs.getString(1);
        String tmp[]=itemSet.split(",");
        if( Integer.parseInt(tmp[0])==x || Integer.parseInt(tmp[1])==x ||
           Integer.parseInt(tmp[2])==x)
        {
            ar.add(itemSet);c++;}
        }
    return ar;
}
}
```

9.Package Name: transactionTable

Class Name 1: TransactionTable

```
package transactionTable;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

public class TransactionClass
{
    /* function to insert the transaction string into the transaction table */

    public int transaction(String I)
    {
        Connection con=connection_MySQL.ConnectionClass.dbcon();

        int x=-1;
```

```

try
{
    PreparedStatement smt=con.prepareStatement("Insert into transaction (Item_List) values
                                                (?");

    smt.setString(1,I);

    smt.executeUpdate();

    ResultSet rs=con.prepareStatement("select max(ID) from transaction").executeQuery();

    rs.next();

    x=rs.getInt(1);

    con.close();
}
catch(Exception e)
{
    System.out.println(e.getMessage());
}

if(x== -1)

    System.exit(1);

return x;
}
}

```

10. Package Name: servletClasses

Class Name: ItemSelect

```

import java.io.IOException;

import java.util.ArrayList;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

```

```

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import itemSetTable.ResultSetTable;


@WebServlet("/itemSelect")

public class ItemSelect extends HttpServlet {

    private static final long serialVersionUID = 1L;

    public ItemSelect()

    {

        super();

    }


    protected void service(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException

    {

        int x=Integer.parseInt(request.getParameter("but"));

        ResultSetTable rs=new ResultSetTable();

        ArrayList<String> ar=null;

        ar=rs.insert(x);

        System.out.println("ItemSelect.java --> ButPressed : "+ar.get(0));

        request.getSession().setAttribute("arrayL", ar);

        request.getRequestDispatcher("itemSet.jsp").forward(request, response);

    }

}

```

Class Name2-ItemSet

```
import java.io.IOException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@WebServlet("/itemSet")

public class ItemSet extends HttpServlet {

    private static final long serialVersionUID = 1L;


    public ItemSet() {

        super();

    }

    protected void service(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

        String x=request.getParameter("rad");


        request.getSession().setAttribute("transac",x+", "+request.getSession().getAttribute("tr
ansac"));

        System.out.println("IN ItemSelct.java --> \nX : "+x+"\nTRANAC :
"+request.getSession().getAttribute("transac"));

        request.getRequestDispatcher("cart.jsp").forward(request, response);

    }

}
```

Class Name 3: Transaction

```
import java.io.IOException;

import java.sql.SQLException;


import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import driverModule.MainClass;

import tableLen1.TableC1;

import tableLen1.ViewL1;

import tableLen2.TableH2;

import tableLen2.ViewL2;

import tableLen3.TableH3;

import tableLen3.ViewL3;

import transactionTable.TransactionClass;

@WebServlet("/Transaction")

public class Transaction extends HttpServlet {

    private static final long serialVersionUID = 1L;

    public Transaction() {

        super();

    }

    protected void service(HttpServletRequest request, HttpServletResponse response) throws

ServletException, IOException {

        String trans=null;

        try {
```

```

trans=(String)request.getSession().getAttribute("transac");
trans=trans.substring(0,trans.length()-1);
TransactionClass obT=new TransactionClass();
int x1=obT.transaction(trans);
TableC1 ob1=new TableC1();
int x=ob1.insert(trans);
System.out.println(x);
ViewL1 ob2=new ViewL1();
int y=ViewL1.avg();
x=ob2.insert(y);
System.out.println(x+" "+y);
TableH2 ob3=new TableH2();
x=ob3.insert(trans);
System.out.println(x);
ViewL2 obl2=new ViewL2();
y=ViewL2.avg();
x=obl2.insert(y);
System.out.println(x+" "+y);
TableH3 ob4=new TableH3();
x=ob4.insert(trans);
System.out.println(x);
ViewL3 obl3=new ViewL3();
y=ViewL3.avg();
x=obl3.insert(y);
System.out.println(x+" "+y);
request.getSession().setAttribute("transacID", x1);
} catch (NumberFormatException e) {

```

```

        e.printStackTrace();

    }

    request.getRequestDispatcher("completion.jsp").forward(request, response);

}

}

```

5.2.2 Coding Efficiency:

Code efficiency is a broad term used to depict the reliability, speed and programming methodology used in developing codes for an application. Code efficiency is directly linked with algorithmic efficiency and the speed of runtime execution for software. It is the key element in ensuring high performance. The goal of code efficiency is to reduce resource consumption and completion time as much as possible with minimum risk to the business or operating environment. The software product quality can be accessed and evaluated with the help of the efficiency of the code used.

Code efficiency plays a significant role in applications in a high-execution-speed environment where performance and scalability are paramount.

One of the recommended best practices in coding is to ensure good code efficiency. Well-developed programming codes should be able to handle complex algorithms.

In order to make the code efficient, following steps are recommended:

- code which goes to redundant processing or code which are unnecessary are removed.
- optimal memory is made used of and minimal amount of storage is used.
- The code should make sue of minimum number of loops.
- Project is divided into modules following the principle of divide and conquer to enable reusability of code. Functions are reused by calling them whenever required.
- Errors are handled and removed at all the various layers of the software including the interface and the logic flow.
- Programming code is written in a manner such that data integrity and consisteeency is ensured.
- The programming code should be written in a manner to keep in compliance with the design logic and flow.
- Data access and data management practises are optimised.
- To use the best keywords, data types and variables, and other available programming concepts to implement the related algorithm
- Since here, the Apriori algorithm has been implemented using the hashing technique, there has been a considerable loss in the spatial complexity of the original algorithm. Thus the space taken up by the program is reduced. Also the code has become much more optimised and faster.

5.3: Modification and Improvements

In the preliminary stages of the project, problem was faced while establishing a connection between the back end code and MySql. JDBC is used for the connection.

The second problem we faced while making the project was in determining an optimised hash function with minimum number of collisions. Also a lot of space was wasted while creating the two item set and three item set hash table. There were way too many null positions in the table since only one item could be entered in one row and column in the table. The rest of the positions were wasted. This was modified or improved by making use of the linked list array.

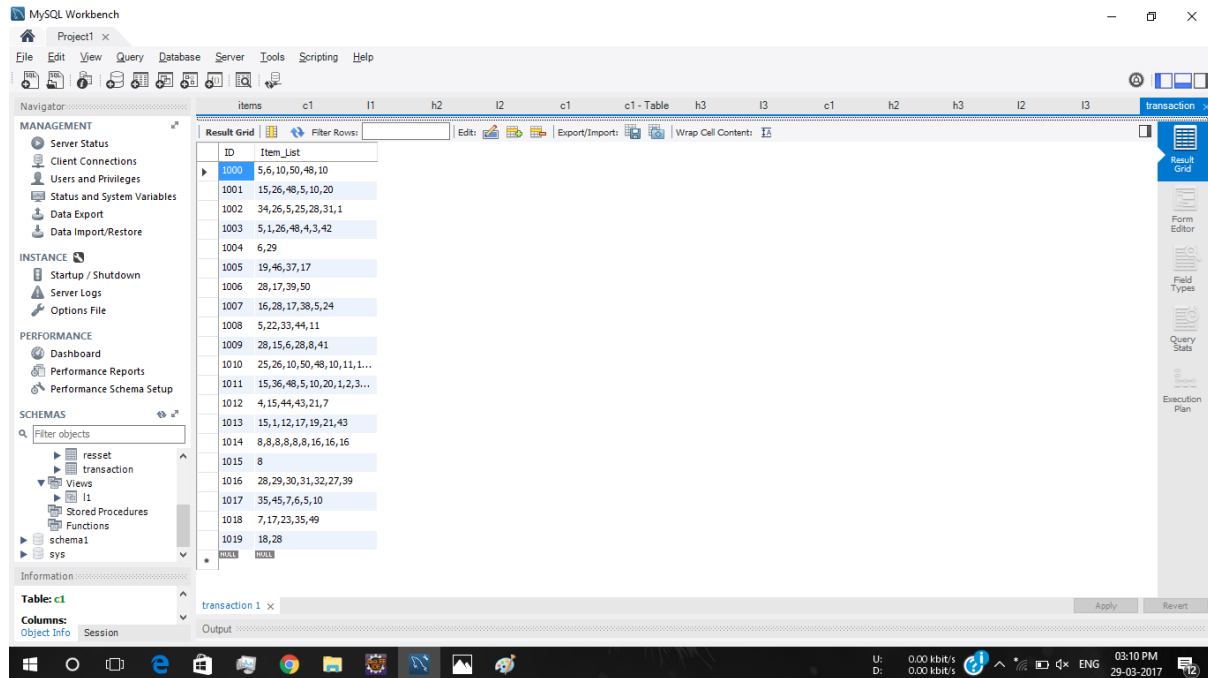
CHAPTER 6: RESULTS AND DISCUSSION

6.1: Test Reports

6.2: User Documentation

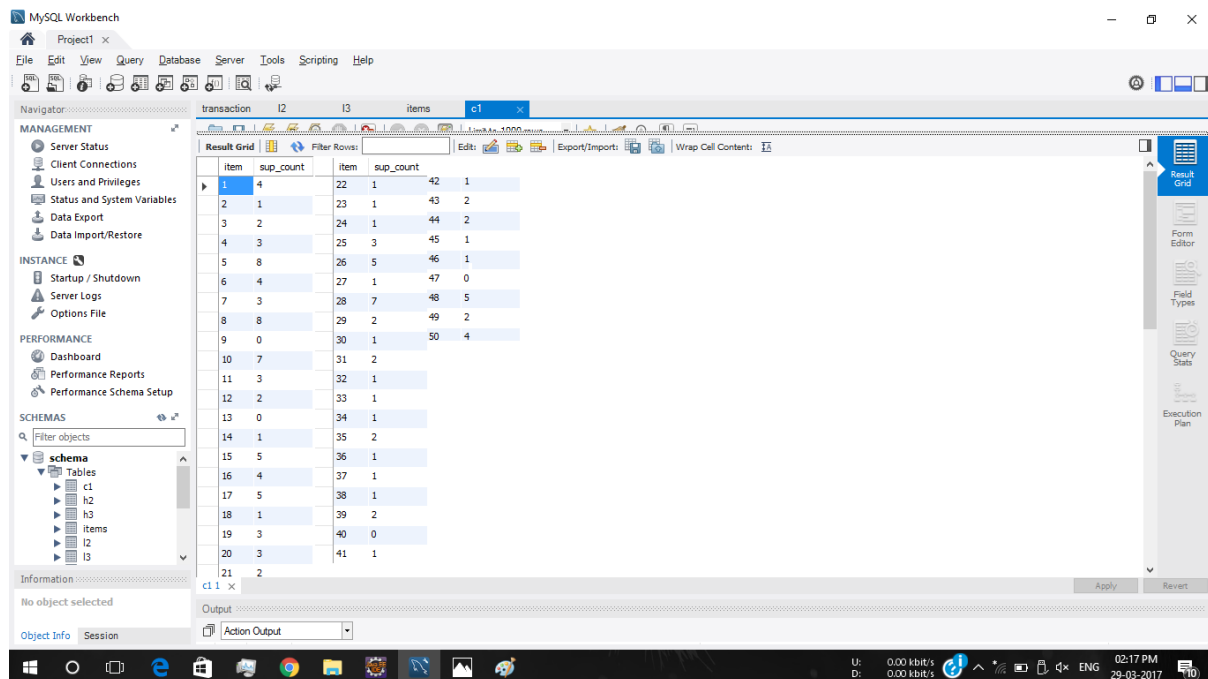
6.1 : Test Reports

For a given set of 20 transaction (Fig: 6.1(a)) the corresponding 1 itemset candidate table generated is as shown in figure Fig: and the frequent 1 itemset table L1 is as shown in Fig:



ID	Item_List
1000	5,6,10,50,48,10
1001	15,26,48,5,10,20
1002	34,26,5,25,28,31,1
1003	5,1,26,48,4,3,42
1004	6,29
1005	19,46,37,17
1006	28,17,39,50
1007	16,28,17,38,5,24
1008	5,22,33,44,11
1009	28,15,6,28,8,41
1010	25,26,10,50,48,10,11,1...
1011	15,36,48,5,10,20,1,2,3...
1012	4,15,44,43,21,7
1013	15,1,12,17,19,21,43
1014	8,8,8,8,8,8,16,16,16
1015	8
1016	28,29,30,31,32,27,39
1017	35,45,7,6,5,10
1018	7,17,23,35,49
1019	18,28
1020	18,28

Fig: 6.1(a) Transaction Table



item	sup_count	item	sup_count
1	4	22	1
2	1	23	1
3	2	24	1
4	3	25	3
5	8	26	5
6	4	27	1
7	3	28	7
8	8	29	2
9	0	30	1
10	7	31	2
11	3	32	1
12	2	33	1
13	0	34	1
14	1	35	2
15	5	36	1
16	4	37	1
17	5	38	1
18	1	39	2
19	3	40	0
20	3	41	1
21	2		

Fig: 6.1(b) 1 itemset Candidate Table

item	count
1	4
5	8
6	4
8	8
10	7
15	5
16	4
17	5
26	5
28	7
48	5
50	4

Fig: 6.1(c) Frequent 1 itemset table I1

Case 1:

Hash function: Concatenating z, y and x and then finding mod 13 in case of 3 itemsets

Concatenating y and x and then finding mod 13 in case of 2 itemsets

Table Size: 13

For the given condition the h2, I2,h3 and I3 tables generated are as follows. We are showing some part of the I2,I3 table in the figure.

a	b	c	d	e	f	g	h	i	j	k	l	m
NULL	NULL	NULL	NULL	NULL	5,26	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	5,48	NULL	NULL	5,26	NULL	NULL	26,48	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	5,28	NULL	5,17	17,28	NULL	NULL	NULL	NULL
NULL	NULL	NULL	6,28	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
48,50	NULL	NULL	NULL	NULL	NULL	10,48	26,48	10,50	26,50	10,26	NULL	NULL
5,20	10,15	1,5	1,20	5,11	1,48	1,10	1,11	1,25	1,26	10,11	1,15	10,26
25,48	15,48	5,48	5,10	11,48	5,25	5,26	11,25	5,15	26,48	11,15	10,25	15,20
NULL	NULL	11,20	15,25	15,26	10,20	20,48	10,48	20,25	20,26	NULL	NULL	NULL
NULL	NULL	NULL	25,26	NULL	NULL	11,26	NULL	NULL	NULL	4,15	NULL	NULL
1,17	NULL	1,19	17,19	NULL	NULL	15,17	NULL	1,15	NULL	15,19	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	8,16	NULL	NULL	NULL	6,10	NULL	NULL
NULL	NULL	5,10	5,6	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Fig: 6.1.1(a) 2 itemset Hash table

MySQL Workbench

Project1 x

File Edit View Query Database Server Tools Scripting Help

Navigator transaction I2 I3 items c1 I1 h2 I2

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

schema

- Tables
- c1
- h2
- h3
- items
- I2
- I3

Information: No object selected

Object Info Session

Result Grid

Filter Rows:

Limit to 1000 rows

Export/Import: Wrap Cell Contents

item	count	item	count
1,10	1	11,26	1
1,11	1	11,48	1
1,15	2	15,17	1
1,17	1	15,19	1
1,19	1	15,20	1
1,20	1	15,25	3
1,25	1	15,26	1
1,26	1	15,48	1
1,48	1	17,19	1
1,5	1	17,28	1
10,11	1	20,25	1
10,15	1	20,26	1
10,20	1	20,48	1
10,25	1	25,26	1
10,26	2	25,48	1
10,48	2	26,48	3
10,50	1	26,50	1
11,15	1	4,15	1
11,20	1	48,50	1
11,25	1	5,10	2

Output: Action Output

Apply Revert

U: 0.00 kbit/s D: 0.00 kbit/s

02:22 PM 29-03-2017

Fig: 6.1.1(b) Frequent 2 itemset table

MySQL Workbench

Project1 x

File Edit View Query Database Server Tools Scripting Help

Navigator transaction I2 I3 items c1 I1 h2 I2 c1 c1-Table h3

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

schema

- Tables
- c1
- h2
- h3
- items
- I2
- I3

Information: Table: h3

Object Info Session

Result Grid

Filter Rows:

Limit to 1000 rows

Export: Wrap Cell Contents

a	b	c	d	e	f	g	h	i	j	k	l	m
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	5,26,48	NULL	NULL
NULL	5,17,28	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
10,26,48	26,48,50	10,26,50	NULL	NULL	10,48,50	NULL	NULL	NULL	NULL	NULL	NULL	NULL
1,10,11	1,5,48	1,5,10	1,5,11	1,5,25	1,5,26	5,11,48	1,5,15	1,15,26	1,10,20	1,11,25	1,10,48	1,5,20
1,11,15	1,10,25	1,10,26	1,25,48	1,10,15	1,11,20	5,15,25	1,11,48	5,10,20	5,11,25	1,20,48	1,11,26	1,26,48
1,20,25	1,20,26	1,15,20	5,10,15	1,15,48	10,15,48	5,25,26	1,15,25	10,11,48	5,20,48	5,10,48	5,26,48	5,10,11
5,10,25	5,10,26	5,25,48	5,15,48	5,11,20	11,20,26	10,11,20	1,25,26	10,15,25	10,15,26	5,11,26	10,11,25	5,11,15
5,20,26	5,15,20	10,20,26	10,15,20	10,25,48	NULL	11,15,20	5,15,26	10,25,26	15,25,26	10,20,48	5,20,25	5,20,25
10,26,48	10,11,15	15,26,48	11,26,48	11,20,25	NULL	15,25,48	11,25,48	11,15,48	NULL	NULL	11,15,25	10,11,26
15,20,48	10,20,25	NULL	15,20,25	15,20,26	NULL	25,26,48	20,25,48	NULL	NULL	11,25,26	11,15,26	11,15,26
NULL	11,20,48	NULL	20,26,48	NULL	NULL	NULL	NULL	NULL	NULL	20,25,26	20,25,26	20,25,26
NULL	1,15,19	NULL	NULL	NULL	NULL	1,17,19	NULL	NULL	15,17,19	NULL	NULL	1,15,17
NULL	NULL	NULL	NULL	NULL	NULL	5,6,10	NULL	NULL	NULL	NULL	1,11,26	NULL

Output: Read Only

U: 0.00 kbit/s D: 0.00 kbit/s

02:42 PM 29-03-2017

Fig: 6.1.1(c) 3 itemset Hash Table

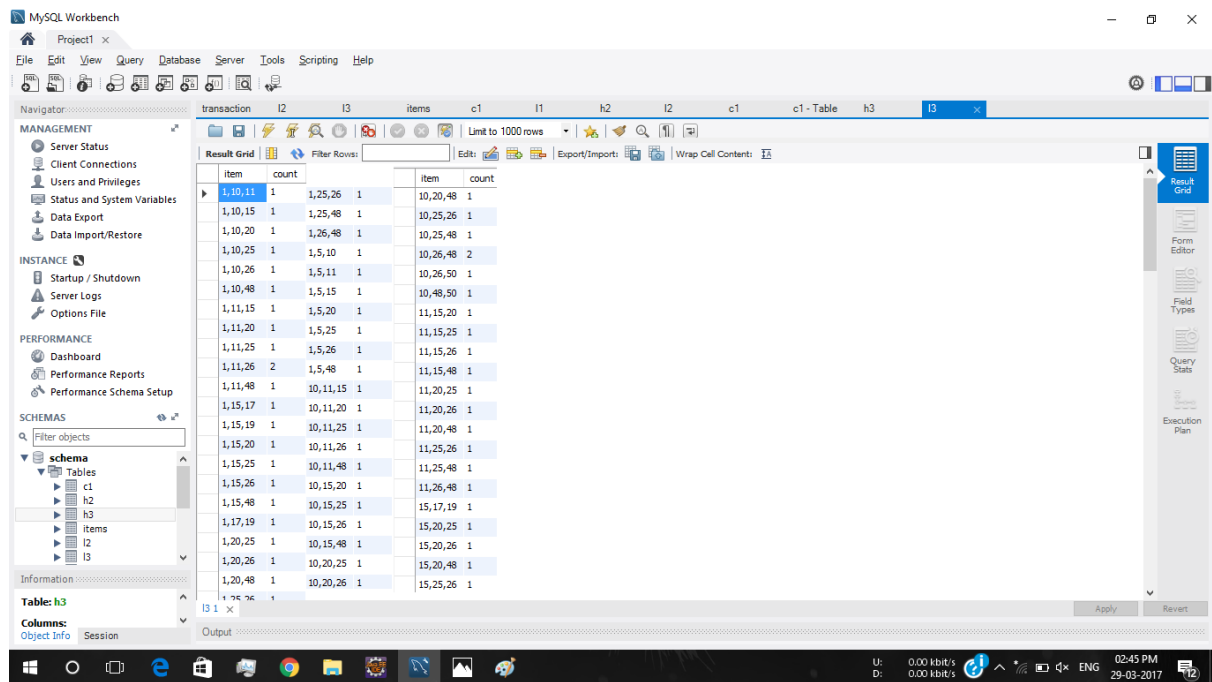


Fig:6.1.1(d) Frequent 3 itemset table 13

Case 2:

Hash function: $((z*100)+(y*10)+ x) \bmod 13$ in case of 3 itemsets

$((y*10)+x) \bmod 13$ in case of 2 itemsets

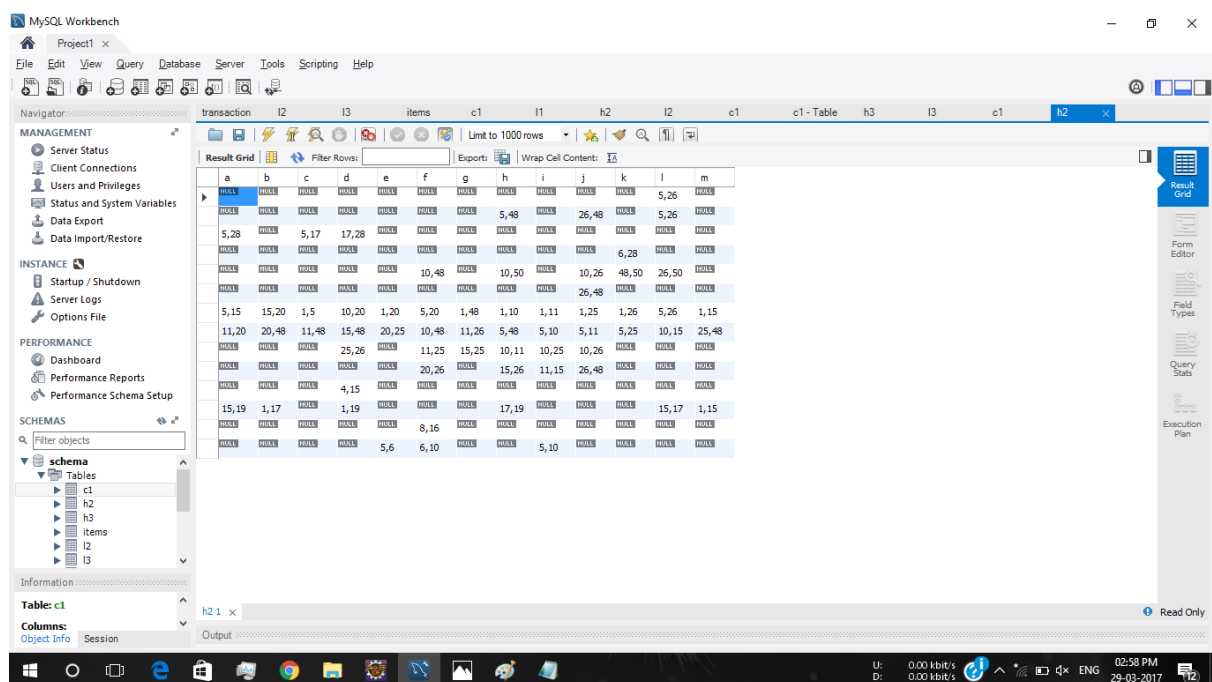
Table Size: 13

Fig: 6.1.2(a) 2 itemset Hash Table H2

MySQL Workbench

Project1 x

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

schema

- Tables
- c1
- h2
- h3
- items
- i2
- i3

Information

Table: c1

Columns: Object Info Session

Result Grid

Filter Rows

Limit to 1000 rows

Export/Import: Wrap Cell Contents

Item	count	Item	count
1,10	1	11,48	1
1,11	1	15,17	1
1,15	2	15,19	1
1,17	1	15,20	1
1,19	1	15,25	1
1,20	1	15,26	1
1,25	1	15,48	1
1,26	1	17,19	1
1,48	1	17,28	1
1,5	1	20,25	1
10,11	1	20,26	1
10,15	1	20,48	1
10,20	1	25,26	1
10,25	1	25,48	1
10,26	2	26,48	3
10,48	2	26,50	1
10,50	1	4,15	1
11,15	1	48,50	1
11,20	1	5,10	2
11,25	1	5,11	1
11,26	1	5,15	1

Output

U: 0.00 kbit/s
D: 0.00 kbit/s

ENG 03:01 PM 29-03-2017

Fig: 6.1.2(b) Frequent 2 itemset Table I2

MySQL Workbench

Project1 x

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

schema

- Tables
- c1
- h2
- h3
- items
- i2
- i3

Information

Table: c1

Columns: Object Info Session

Result Grid

Filter Rows

Limit to 1000 rows

Export/Import: Wrap Cell Contents

a	b	c	d	e	f	g	h	i	j	k	l	m
NULL	NULL	5,26,48	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	5,17,28	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	10,26,48	10,48,50	10,26,50	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	26,48,50	NULL	NULL	NULL	NULL
1,20,25	1,5,20	1,11,26	1,5,48	1,5,10	1,5,11	1,5,25	1,25,48	1,5,15	1,15,20	1,11,48	1,10,20	1,10,20
5,10,11	1,10,48	1,15,25	1,10,11	1,10,25	1,10,26	5,11,20	1,10,15	5,11,48	1,11,20	1,20,48	5,10,48	1,15,48
5,15,26	1,11,25	5,10,26	1,15,26	1,11,15	1,26,48	10,15,26	5,15,20	10,26,48	5,10,20	5,20,25	5,11,25	1,25,26
10,15,20	1,20,26	5,26,48	10,20,25	5,10,15	5,25,48	15,20,48	5,20,48	15,25,26	5,15,48	15,20,26	5,20,26	5,11,26
10,20,48	5,10,25	10,15,48	NULL	10,11,25	10,11,26	NULL	10,11,15	NULL	5,25,26	20,25,48	10,25,48	5,15,25
11,20,26	5,11,15	10,25,26	10,20,26	10,15,25	11,25,48	NULL	11,25,48	NULL	11,15,20	NULL	11,15,48	10,11,20
25,26,48	10,11,48	11,15,26	NULL	11,26,48	NULL	NULL	20,26,48	NULL	11,20,48	NULL	11,25,26	11,20,25
NULL	11,15,25	NULL	15,25,48	NULL	NULL	NULL	NULL	15,20,25	NULL	NULL	NULL	NULL
NULL	15,26,48	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	20,25,26	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
NULL	NULL	1,17,19	NULL	NULL	NULL	NULL	1,15,17	NULL	1,15,19	NULL	15,17,19	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	5,6,10	NULL	NULL	NULL

Output

U: 0.00 kbit/s
D: 0.00 kbit/s

ENG 02:59 PM 29-03-2017

Fig: 6.1.2(c) 3 itemset Hash Table h3

MySQL Workbench

Project1

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

schema

- Tables
 - c1
 - h2
 - h3
 - items
 - I2
 - I3

Information

Table: c1

Columns:

Object Info Session

Output

Result Grid

Limit to 1000 rows

Filter Rows:

Export/Import

Wrap Cell Contents

Apply Revert

U: 0.00 kbit/s
D: 0.00 kbit/s

03:06 PM
29-03-2017

Fig: 6.1.2(d) Frequent 3 itemset Table I3

Case 3:

Hash function: Concatenating z, y and x and then finding mod 17 in case of 3 itemsets

Concatenating y and x and then finding mod 17 in case of 2 itemsets

Table Size: 17

MySQL Workbench

Project1

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

Tables

- c1
- h2
- h3
- items
- I2
- I3
- reset

Information

No object selected

Object Info Session

Output

Result Grid

Limit to 1000 rows

Export

Wrap Cell Contents

Read Only

U: 0.00 kbit/s
D: 1.02 kbit/s

11:17 PM
29-03-2017

Fig: 6.1.3(a) 2 itemset Hash Table H2

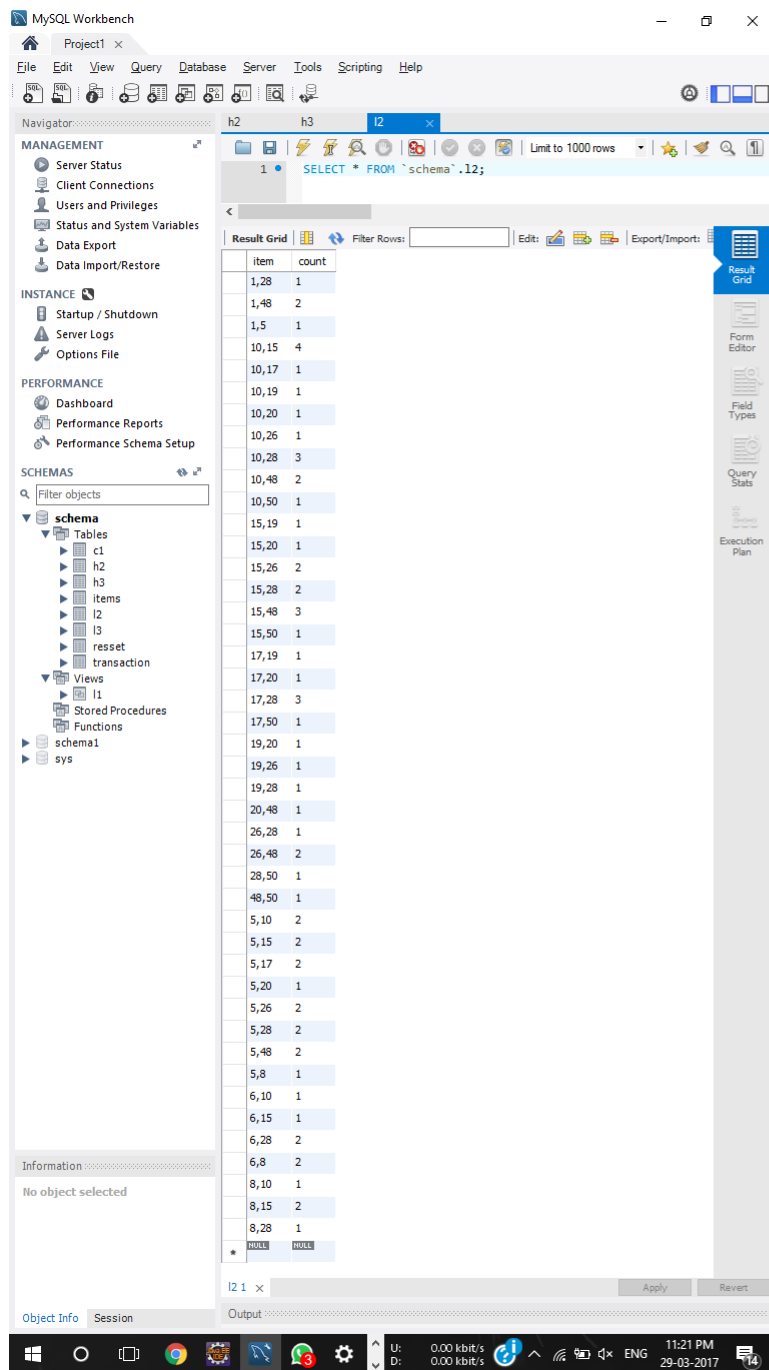


Fig: 6.1.3(b) Frequent 2 itemset Table I2

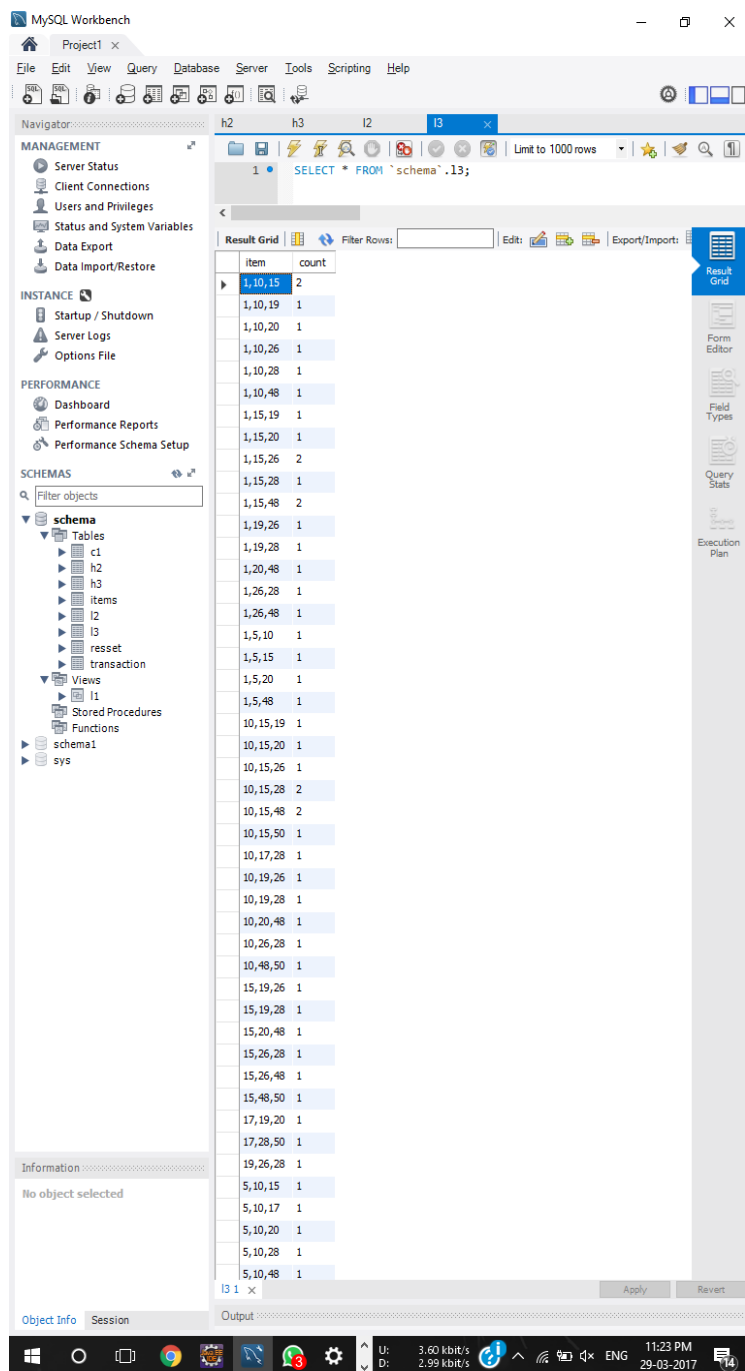


Fig: 6.1.3(d) Frequent 3 itemset Table I3

Fig: 6.1.3(c) 3 itemset Hash Table H3

Case 4:

Hash function: $((z*100)+(y*10)+ x) \bmod 17$ in case of 3 itemsets

$((y*10)+ x) \bmod 17$ in case of 2 itemsets

Table Size: 13

Fig: 6.1.4(a) 2 itemset Hash Table H2

MySQL Workbench

Project1

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

schema

- Tables
- c1
- h2
- h3
- items
- I2
- I3

Information

View: I1

Columns: Object Info Session

Output

Result Grid

Filter Rows:

Limit to 1000 rows

Export/Import:

Wrap Cell Contents

Apply Revert

U: 0.00 kbit/s D: 0.00 kbit/s

ENG 03:48 PM 29-03-2017

Fig: 6.1.4(b) Frequent 2 itemset Table I2

MySQL Workbench

Project1

File Edit View Query Database Server Tools Scripting Help

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

schema

- Tables
- c1
- h2
- h3
- items
- I2
- I3

Information

View: I1

Columns: Object Info Session

Output

Result Grid

Filter Rows:

Export:

Wrap Cell Contents

Read Only

U: 0.00 kbit/s D: 0.00 kbit/s

ENG 03:46 PM 29-03-2017

Fig: 6.1.4(c) 3 itemset Hash Table H3

MySQL Workbench

Project1 x

File Edit View Query Database Server Tools Scripting Help

Navigator: I2 I3 transaction c1 I1 h2 - Table h3 - Table h2 h3 I1 - View I1 c1 I2 I3 x

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS

Filter objects

schema

- Tables
 - c1
 - h2
 - h3
 - items
 - I2
 - I3

Information:

View: I1

Columns: Object Info Session

Output:

Result Grid

Filter Rows:

Limit to 1000 rows

Export/Import:

Wrap Cell Contents

item	count	item	count	item	count
1,10,15	2	10,15,19	1	19,26,28	1
1,10,19	1	10,15,20	1	5,10,15	1
1,10,20	1	10,15,26	1	5,10,17	1
1,10,26	1	10,15,28	2	5,10,20	1
1,10,28	1	10,15,48	2	5,10,28	1
1,10,48	1	10,15,50	1	5,10,48	1
1,15,19	1	10,17,28	1	5,15,20	1
1,15,20	1	10,19,26	1	5,15,48	1
1,15,26	2	10,19,28	1	5,17,28	2
1,15,28	1	10,20,48	1	5,20,48	1
1,15,48	2	10,26,28	1	5,26,48	1
1,19,26	1	10,48,50	1	5,8,15	1
1,19,28	1	15,19,26	1	6,10,15	1
1,20,48	1	15,19,28	1	6,10,28	1
1,26,28	1	15,20,48	1	6,15,28	1
1,26,48	1	15,26,28	1	6,8,10	1
1,5,10	1	15,26,48	1	6,8,15	1
1,5,15	1	15,48,50	1	6,8,28	1
1,5,20	1	17,19,20	1	8,10,15	1
1,5,48	1	17,28,50	1	8,10,28	1

U: 0.00 kbit/s
D: 0.00 kbit/s

03:49 PM
29-03-2017

Fig: 6.1.4(d) Frequent 3 itemset Table I3

Discussion:

1. On comparing the H2 table of size 17 and 13 for the general hash function we find that both of them are suffering quite less collisions but the H2 of size 13 is more compact and contains less NULL than that of size 17 H2. Thus for this set of transaction H2 of size 13 is preferred. But since the number of transactions will increase in the long a hash table with bigger size will be preferred to reduce the collisions. In such a case the H2 of size 13 can be rehashed in H2 of size 17.

The same holds for hash table H3 of size 13 and 17.

2. On comparing the Hash table H2 of size 13 for the two different hash functions we see that not much difference between the two tables is found. The number of elements remain quite the same only the hashed position is different.

The same holds for the H3 table also.

The same case is also true for Hash table of size 17.

6.2 : User Documentation

➤ **User-defined functions**

A function is a block of code that performs a particular task. There are times when we need to write a particular block of code for more than once in our program. This may lead to bugs and irritation for the programmer. Java language provides an approach in which you need to declare and define a group of statements once and that can be called and used whenever required. This saves both time and space. We have used many user-defined functions in this project. The functions used in the project are described below:

- 1) **Package name:** *connection_MySql*

Class name: *ConnectionClass*

Function Name:

- *Connection dbcon()*

This particular function is used for the purpose of establishing a connection between the MySql and the Java backend code. The connection is done through the JDBC.

- 2) **Package name:** *driverModule*

Class Name: *MainClass*

FunctionName:

- *main()*

This is the main function in the program. This is the function from which execution of the program takes place. All the various user-defined functions used in the program are called from this particular function. It takes as input the item as selected by the user. Based on this selection of the item and also based on the option of buy now or select which will be chosen by the user, appropriate actions will take place. This function is designed using switch case.

3) **Package name:** *hash*

Class name: *Hashing*

Function Name:

- *Hashing()*

This is a non-parameterised constructor as it has the same name as that of the class name. It is used to initialise the values of the global integer variables x, y and z to 0.

- *getHash()*

This function is used to calculate the value of the position in which the two item set combination and the three item-set combination will be placed in the hash table h2 and h3 respectively. The hash function is used for the purpose of this calculation. It returns the value of the position in which the items are to be stored.

- *setX(int)*

This works as a reference to the current Object whose Method or constructor is being invoked. The *this* keyword can be used to refer to any member of the current object from within an instance Method or a constructor. Here it refers to the variable x and sets it with the value of x from the method called.

- *setY(int)*

This works as a reference to the current Object whose Method or constructor is being invoked. The *this* keyword can be used to refer to any member of the

current object from within an instance Method or a constructor. Here it refers to the variable y and sets it with the value of y from the method called.

- *setZ(int)*

This works as a reference to the current Object whose Method or constructor is being invoked. The **this** keyword can be used to refer to any member of the current object from within an instance Method or a constructor. Here it refers to the variable z and sets it with the value of z from the method called.

4. **Package name:** *itemSet*

Class Name: *ItemSet_2*

Function Name:

- *ItemSet_2(String)*

This is a parameterised constructor and is used to set the value of the global variable. It makes use of the this keyword and sets the value of the variable st with that of the value from the variable present in the method from where it is called.

- *String[] combi()*

This function returns a string array. It is used to calculate all the possible combination of two item sets. The sort function is called from this particular function and is used to sort the item set array in ascending order. It is used to calculate all the possible combination of two item sets.

- *Integer[] sort()*

It returns the integer array. It is called from the combi() function and is used for the purpose of sorting the item set array into ascending order so that further operations can take place on the item set array.

Class Name: *ItemSet_3*

Function name:

- *Itemset_3(String)*

This is a parameterised constructor which takes as input the string st and is used to initialise this st with the value from the function/ method which calls this function. The this keyword is used for this purpose.

- *combi()*

This function has a return type of String array. It is used for the purpose of finding all those possible three item set combination from the transaction string. The sort function is also called from this particular function. As the very name suggests, the sort function sorts the item into increase order.

5. **Package Name:** *itemSetTable*

Class Name: *ResultSetTable*

Function Name:

- *insert(int)*

It has a void return type and takes as input the item which was entered by the user. The already existing “reset” table is truncated or dropped so that the new “reset” table can be formed. This function is basically used for the purpose of entering the one item and its two item set and three item set combination into the reset table.

- *show(Conection)*

It takes as input a variable of type Connection. It is used for the purpose of displaying the “reset” table with the appropriate user given item along with its two item set and three item set combination.

6. **Package Name:** *tableLen1*

Class name: *TableC1*

Function Name:

- *insert(String)*

This function takes as input the transaction string and uses it to form the array ar. The different items as present in the transaction string separated by comma are entered

into the array *ar* on the basis of this comma. The array *ar* is now used for the purpose of updating the already existing candidate table. The frequency column of the candidate table which consists of the frequencies of all the items present is basically updated with respect to the items as present in the transaction string.

Class: *ViewL1*

Function:

- *insert(int)*

This function takes as input the minimum support count. It works with the already existing frequent one item set table *l1*. It is used for the purpose of adding those items to the *l1* table which has a frequency greater than the minimum support count. this function has an integer return type.

- *isL1(int)*

It takes as input the item which was entered by the user. It is used for the purpose of checking whether this particular item is present in the *l1* table and accordingly lays the basis for the further operation to take place. It returns a Boolean value to the main class from where it was called. It returns true if the item is present in the *l1* table, else it returns false.

7. **Package name:** *tableLen2*

Class name: *TableH2*

Function:

- *insert(string)*

This function takes as input the transaction string. Two of the functions called from this particular function: the *isL1* and the *combi()*. The *combi* function will return a two item set combination of all the items which are present in the transaction table. The *isL1* function is now used to check whether each of the items present in the two item combinations are present in the *l1* table. If the result is affirmative, then that particular item is entered into the linked list array *h* by calling the appropriate functions.

- *runSQL(linked list)*

Takes as input the linked list array h and is used for the purpose of inserting all the items present in the linked list to the corresponding position in the hash table h2.

Class name: *ViewL2*

Function name:

- *insert(int)*

This function takes as input the minimum support count. it is used for the purpose of entering all those two item set combination into the l2 table if their number of occurrences in the h2 table is greater than or equal to their frequency in the hash table h2.

- *isL2(String)*

It takes as input a two item set string x. it is used to compare and find whether this two item set combination is present in the l2 table.

- *isL2(int)*

It takes as input the item which was entered by the user. It has a return type of array list. It is used to return all those two item set combinations in which one of the items should be equal to the item entered by the user(say x).

8. **Package name:** *tableLen3*

Class name: *TableH3*

Function name:

- *insert(String)*

This function takes as input the transaction string. Two of the functions called from this function are the combi function and the isL2(String) function. The combi function returns all three possible item set combination. The isL2(String) on the other hand returns to this function all those two item set combinations which are present in the frequent two item set table l2 based on all the three item set combination. A linked list array is used to store the appropriate three item set combinations into it which is done by calling all the other required functions.

- *runSQL(LinkedList)*

Takes as input the linked list array h and is used to enter all the three item set combinations present in h in their corresponding positions in the three item set hash table.

Class name: *ViewL3*

Function name:

- *insert(int)*

This particular function takes the minimum support count as the parameter. Here the already existing l3 table is truncated at the beginning so that the new frequent three item set table(l3) can be created. It is used to insert all those three item set combination into the l3 table for which the number of occurrences in the h3 table is greater than or equal to the minimum support count value. The return type of this function is an integer signifying as to whether the frequent three item set table(l3) has been updated.

- *isL3(int)*

This function takes as parameter the item which has been selected by the user. It has a return type of array list. It is used to return all those three item set combinations in which one of the items should be equal to the item entered by the user(say x).

9. **Package Name:** *transaction Table*

Class Name: *TransactionClass*

Function Name:

- *transaction(String)*

This function is used to update the transaction table with the last transaction that takes place. It has a return type integer which signifies whether the transaction has been successfully updated. The transaction ID is also generated in this function.

➤ In-Built Functions

Built in functions are those which are already predefined in the different API's of the Java Development Kit.

- ***Println***

```
public void println(String x)
```

Prints a String and then terminate the line. This method behaves as though it invokes print(String) and then println().

Parameters:

x - The String to be printed.

- ***Exit***

```
public static void exit(int status)
```

Terminates the currently running Java Virtual Machine. The argument serves as a status code; by convention, a nonzero status code indicates abnormal termination.

This method calls the exit method in class Runtime. This method never returns normally.

The call System.exit(n) is effectively equivalent to the call:

```
Runtime.getRuntime().exit(n)
```

- ***getConnection***

```
public static Connection getConnection(String url,  
                                     String user,  
                                     String password)  
    throws SQLException
```

Attempts to establish a connection to the given database URL. The DriverManager attempts to select an appropriate driver from the set of registered JDBC drivers.

Note: If the user or password property are also specified as part of the url, it is implementation-defined as to which value will take precedence. For maximum portability, an application should only specify a property once.

Parameters:

url - a database url of the form jdbc:subprotocol:subname

user - the database user on whose behalf the connection is being made

password - the user's password

Returns:

a connection to the URL

Throws:

SQLException - if a database access error occurs or the url is null

SQLTimeoutException - when the driver has determined that the timeout value specified by the setLoginTimeout method has been exceeded and has at least tried to cancel the current database connection attempt

- ***charAt***

public char charAt(int index)

Returns the char value at the specified index. An index ranges from 0 to length() - 1. The first char value of the sequence is at index 0, the next at index 1, and so on, as for array indexing.

If the char value specified by the index is a surrogate, the surrogate value is returned.

Specified by:

charAt in interface CharSequence

Parameters:

index - the index of the char value.

Returns:

the char value at the specified index of this string. The first char value is at index 0.

Throws:

IndexOutOfBoundsException - if the index argument is negative or not less than the length of this string.

- ***substring***

public String substring(int beginIndex,
int endIndex)

Returns a string that is a substring of this string. The substring begins at the specified beginIndex and extends to the character at index endIndex - 1. Thus the length of the substring is endIndex-beginIndex.

Examples:

"hamburger".substring(4, 8) returns "urge"

"smiles".substring(1, 5) returns "mile"

Parameters:

beginIndex - the beginning index, inclusive.

endIndex - the ending index, exclusive.

Returns:

the specified substring.

Throws:

IndexOutOfBoundsException - if the beginIndex is negative, or endIndex is larger than the length of this String object, or beginIndex is larger than endIndex.

- ***split***

public String[] split(String regex)

Splits this string around matches of the given regular expression.

This method works as if by invoking the two-argument split method with the given expression and a limit argument of zero. Trailing empty strings are therefore not included in the resulting array.

The string "boo:and:foo", for example, yields the following results with these expressions:

Regex	Result
-------	--------

```
: { "boo", "and", "foo" }  
O { "b", "", ":and:f" }
```

Parameters:

regex - the delimiting regular expression

Returns:

The array of strings computed by splitting this string around matches of the given regular expression

Throws:

PatternSyntaxException - if the regular expression's syntax is invalid

- ***add***

public boolean add(E e)

Appends the specified element to the end of this list.

Specified by:

add in interface Collection<E>

Specified by:

add in interface List<E>

Overrides:

add in class AbstractList<E>

Parameters:

e - element to be appended to this list

Returns:

true (as specified by Collection.add(E))

- ***equals***

public boolean equals(Object obj)

Compares this object to the specified object. The result is true if and only if the argument is not null and is an Integer object that contains the same int value as this object.

Overrides:

equals in class Object

Parameters:

obj - the object to compare with.

Returns:

true if the objects are the same; false otherwise.

- ***remove***

public E remove(int index)

Removes the element at the specified position in this list. Shifts any subsequent elements to the left (subtracts one from their indices).

Specified by:

remove in interface List<E>

Overrides:

remove in class AbstractList<E>

Parameters:

index - the index of the element to be removed

Returns:

the element that was removed from the list

Throws:

IndexOutOfBoundsException - if the index is out of range (index < 0 || index >= size())

- ***toArray***

public <T> T[] toArray(T[] a)

Returns an array containing all of the elements in this list in proper sequence (from first to last element); the runtime type of the returned array is that of the specified array. If the list fits in the specified array, it is returned therein. Otherwise, a new array is allocated with the runtime type of the specified array and the size of this list.

If the list fits in the specified array with room to spare (i.e., the array has more elements than the list), the element in the array immediately following the end of the collection is set to null. (This is useful in determining the length of the list *only* if the caller knows that the list does not contain any null elements.)

Specified by:

toArray in interface Collection<E>

Specified by:

toArray in interface List<E>

Overrides:

toArray in class AbstractCollection<E>

Type Parameters:

T - the runtime type of the array to contain the collection

Parameters:

a - the array into which the elements of the list are to be stored, if it is big enough; otherwise, a new array of the same runtime type is allocated for this purpose.

Returns:

an array containing the elements of the list

Throws:

ArrayStoreException - if the runtime type of the specified array is not a supertype of the runtime type of every element in this list

NullPointerException - if the specified array is null

- ***getString***

String getString(int columnIndex)

throws SQLException

Retrieves the value of the designated column in the current row of this ResultSet object as a String in the Java programming language.

Parameters:

columnIndex - the first column is 1, the second is 2, ...

Returns:

the column value; if the value is SQL NULL, the value returned is null

Throws:

SQLException - if the columnIndex is not valid; if a database access error occurs or this method is called on a closed result set

- ***next***

boolean next()

throws SQLException

Moves the cursor forward one row from its current position. A ResultSet cursor is initially positioned before the first row; the first call to the method next makes the first row the current row; the second call makes the second row the current row, and so on.

When a call to the next method returns false, the cursor is positioned after the last row.

Any invocation of a ResultSet method which requires a current row will result in a SQLException being thrown. If the result set type is TYPE_FORWARD_ONLY, it is vendor specified whether their JDBC driver implementation will return false or throw an SQLException on a subsequent call to next.

If an input stream is open for the current row, a call to the method next will implicitly close it. A ResultSet object's warning chain is cleared when a new row is read.

Returns:

true if the new current row is valid; false if there are no more rows

Throws:

SQLException - if a database access error occurs or this method is called on a closed result set

- ***setNull***


```
void setNull(int parameterIndex,  
            int sqlType)
```

throws SQLException

Sets the designated parameter to SQL NULL.

Note: You must specify the parameter's SQL type.

Parameters:

parameterIndex - the first parameter is 1, the second is 2, ...

sqlType - the SQL type code defined in java.sql.Types

Throws:

SQLException - if parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

SQLFeatureNotSupportedException - if sqlType is a ARRAY, BLOB, CLOB, DATALINK, JAVA_OBJECT, NCHAR, NCLOB, NVARCHAR, LONGNVARCHAR, REF, ROWID, SQLXML or STRUCT data type and the JDBC driver does not support this data type

- ***executeQuery***

```
ResultSet executeQuery()
```

throws SQLException

Executes the SQL query in this PreparedStatement object and returns the ResultSet object generated by the query.

Returns:

a ResultSet object that contains the data produced by the query; never null

Throws:

SQLException - if a database access error occurs; this method is called on a closed PreparedStatement or the SQL statement does not return a ResultSet object

SQLTimeoutException - when the driver has determined that the timeout value that was specified by the setQueryTimeout method has been exceeded and has at least attempted to cancel the currently running Statement

- ***setInt***

```
void setInt(int parameterIndex, int x)
```

throws SQLException

Sets the designated parameter to the given Java int value. The driver converts this to an SQL INTEGER value when it sends it to the database.

Parameters:

parameterIndex - the first parameter is 1, the second is 2, ...

x - the parameter value

Throws:

SQLException - if parameterIndex does not correspond to a parameter marker in the SQL statement; if a database access error occurs or this method is called on a closed PreparedStatement

- ***executeUpdate***

int executeUpdate()

throws SQLException

Executes the SQL statement in this PreparedStatement object, which must be an SQL Data Manipulation Language (DML) statement, such as INSERT, UPDATE or DELETE; or an SQL statement that returns nothing, such as a DDL statement.

Returns:

either (1) the row count for SQL Data Manipulation Language (DML) statements or (2) 0 for SQL statements that return nothing.

Throws:

SQLException - if a database access error occurs; this method is called on a closed PreparedStatement or the SQL statement returns a ResultSet object

SQLTimeoutException - when the driver has determined that the timeout value that was specified by the setQueryTimeout method has been exceeded and has at least attempted to cancel the currently running Statement

CHAPTER 7: CONCLUSIONS

7.1: Conclusion

7.2: Limitations of the System

7.3: Future Scope of the Project

CONCLUSION

The Apriori Algorithm is an already established algorithm which makes it easier to the user to take decisions. Decisions are something which forms a part of every stages of life. There is no escaping it. This algorithm can only work forward towards making your life a little more simpler. If you are at a shopping mall, you will often find that the items which are placed next to each other are highly relevant to each other and most of the time the items you are most likely to buy are placed near to each other. This is basically a real life example of the Apriori algorithm. The data based on the customer's transactions are analysed and then the items most likely to be purchased are placed together. The Apriori algorithm is a more technologically advanced version of this given example. It is the most fundamental part of the algorithms which are implemented in the online shopping websites. If you are visiting an online store, you will often find that on choosing an item, you are given recommendations and options to choose certain relevant combination of objects. This is precisely the working of the apriori algorithm.

In our project, we have taken the classical Apriori algorithm and have improved upon it. We have implemented this algorithm using the hashing technique. The project definitely has its shortcomings and drawbacks, but it is still quite an improvement over the previous algorithm. Implementing the algorithm through hashing process has brought about a considerable reduction in the spatial complexity of the algorithm and has made it more optimised and turned it faster.

While designing this algorithm, a lot of time was given to identifying and analysing the requirements. Designing the algorithm in such a way that taking decisions becomes all the more easier for the user. A review of the analysis and design phase was done. Care was taken strictly to follow the software engineering concepts and principles so as to maintain good quality in the developed system as per the user requirements.

The entire back end of the project was written using the object oriented language Java. The Java Database Connectivity(JDBC) was used for the purpose of making a connection between back end code and MySQL. The SQL queries were successfully executed. We have followed the three tier architecture between the user end and the server end. The user is sending a request to the server. The server on receiving the request is working on it, extracting the necessary information from the database server and is giving a response back to the client. There are certain limitations of the system which can definitely be worked on for further improvements.

7.2 Limitations of the System:

The project has certain limitations as well:

We are not working with transaction table but with the transaction string in the project. As a result, we are helping the users make a decision based on his current transaction string. Thus we are following the process of data analysis as opposed to the process of data mining.

The hash function we are using is not the most optimised one. Several other hash functions can be used which are much more optimised and will result in lesser collision.

Since we are making use of a structured database format for the purpose of storing the items, a lot of null spaces is generated in the two item set hash table (h2) and the three item set hash table (h3). The amount of null spaces increases with an increase in the transaction.

The value of the support count is generated dynamically. As a result, a situation might arise in which the item is present in the frequent three item set table (I3) even if it is not present in the frequent two item set table (I2).

Once an item has been added into the transaction string and has been put into the Apriori algorithm, there is no backtracking. If a user wishes to change an item he has already added to the transaction string, he cannot backtrack and replace that item with some other item or even delete that item.

7.3 Future Scope of Project:

Every application has its own merits and demerits. The project has covered almost all the requirements. Further requirements and improvements can easily be done since the coding is mainly structured or modular in nature. Changing the existing modules or adding new modules can append improvements.

Also the mentioned limitations of the project can be taken care of. This will help in enhancing the algorithm further. Since the main goal of the project is to offer decision support to the user, it can very easily be applied to the online shopping sites.

A better hash function can be used as it would reduce the number of null positions in the hash tables and improve the spatial complexity of the algorithm further.

REFERENCES

1. Sudhir Tirumalasetty, Aruna Jadda, Sreenivasa and Reddy Edara, “An enhanced Apriori Algorithm for Discovering Frequent Patterns with Optimal Number of Scans”, International Journal of Computer Science Issues 2015
2. Ashma Chawla and Kanwalvir Singh Dhindsa, “Implementation of Association Rule Mining using Reverse Apriori Algorithmic Approach”, International Journal of Computer Applications (0975 – 8887) Volume 93 – No.8, May 2014
3. Yanbin Ye and Chia-Chu Chiang, “A parallel Apriori Algorithm for Frequent Itemsets Mining”, Software Engineering Research, Management and Applications, 2006, Fourth International Conference.
4. Raffaele Perego, Salvatore Orlando and P. Palmerini, “Enhancing the Apriori Algorithm for Frequent Set Counting by Raffaele Perego”, Volume 2114 of the book series Lecture Notes in Computer Science (LNCS)
5. K.Vanitha and R.Santhi , “USING HASH BASED APRIORI ALGORITHM TO REDUCE THE CANDIDATE 2- ITEMSETS FOR MINING ASSOCIATION RULE”, Volume 2, No. 5, April 2011 Journal of Global Research in Computer Science
6. Nick Roussopoulos, “Materialized Views and Data Warehouses” , Department of Computer Science and Institute of Advanced Computer Studies, University of Maryland
7. Shikha Bhardwaj, Preeti Chhikara, Satender Vinayak, Nishant Pai, Kuldeep Meena, “Improved Apriori Algorithm for Association Rules”, International Journal of Technical Research and Applications e-ISSN: 2320-8163, www.ijtra.com Volume 3, Issue 3 (May-June 2015), PP. 238-240
8. Surajt Chaudhuri, Umeshwar Dayal in ACM Sigmond Record, “An overview of Data Warehousing and OLAP technology” ,Issue March 1997