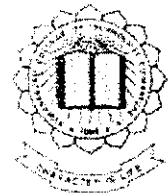


P-2589



**DATABASE ENCODING AND AN
ANTI-APRIORI ALGORITHM FOR
ASSOCIATION RULES MINING**



A PROJECT REPORT

Submitted by

M.PRADEEP VISWANATH 71205205035

K.RAJAMUTHU 71205205041

C.R.SHESH BABU 71205205055

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY



KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

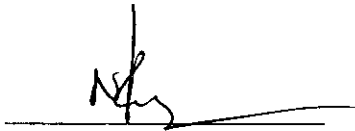
ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2009

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report “DATABASE ENCODING AND AN ANTI-APRIORI ALGORITHM FOR ASSOCIATION RULES MINING” is the bonafide work of “**M.PRADEEP VISWANATH, K.RAJAMUTHU and C.R.SHESH BABU**” who carried out the project work under my supervision.

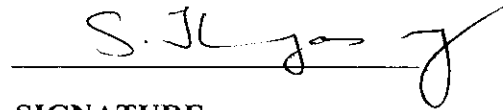


SIGNATURE

Mrs.N.Rajathi M.E.,

SUPERVISOR,

Dept of Information Technology,
Kumaraguru College of Technology,
Coimbatore – 641 006.



SIGNATURE

Dr.S.Thangasamy, B.E(Hons)., Ph.D.,

DEAN,

Dept of Computer Science and Engineering,
Kumaraguru College of Technology,
Coimbatore – 641 006.

The candidates with University Register No 71205205035, 71205205041 and 71205205055 were examined by us in the project viva-voce examination held on 28.04.2009



INTERNAL EXAMINER



EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our profound gratitude to our Chairman **Padmabhushan Arutselvar Dr. N. Mahalingam B.Sc., F.I.E.**, for giving this great opportunity to pursue this course.

We would like to thank Vice Chairman **Dr.K. Arumugam B.E., M.S., M.I.E.**, Correspondent, **Shri.M.Balasubramaniam** and Joint Correspondent **Dr.A.Selvakumar** for providing the necessary facilities to complete our project.

We are immensely grateful to our Vice Principal **Prof R.Annamalai**, for his invaluable support to the outcome of this project.

We are deeply obliged to **Dr.S.Thangasamy**, Dean, Department of Computer Science and Engineering for his valuable guidance and useful suggestions during the course of this project.

We also extend our heartfelt thanks to our project co-ordinator, **Ms. L.S Jayashree M.E., (Ph.D.)**, Associate Professor, Department of Information Technology for providing us her support which really helped us.

We are indebted to our project guide and extend our gratitude towards **Mrs.N.Rajathi M.E.**, Assistant Professor, Department of Information Technology for his helpful guidance and valuable support given to us throughout this project.

We thank the teaching and non-teaching staffs of our department for providing us the technical support during the course of this project. We also thank all of our friends who helped us to complete this project successfully.

ABSTRACT

Association rule mining seeks to discover associations among transactions encoded in a database. An association rule takes the form $A \Rightarrow B$ where A (the antecedent) and B (the consequent) are sets of predicates. Discovering association rules in these algorithms are usually done in two phases. In the first phase, the frequent itemset are generated and in the second phase, the interesting rules are extracted from these frequent itemset. The task of discovering all frequent itemset is quite challenging especially in the large database because the database may be massive, containing millions of transactions. An established algorithm called Apriori generates $(k+1)$ -candidates by joining frequent k -itemset. So all subset of every itemset must be generated for finding superior frequent itemset, although many of them may be not useful and may be not exploited for finding association rules because some of them have no interesting antecedent or consequent in the rules. This process takes a long time and it requires thousands of times of database scan. To handle these problems, we use a method for encoding database. Here, a record is denoted by only one binary number and so the size of the database is reduced sharply. If the database-encoding algorithm is used into some known modified algorithms, the efficiency will be improved remarkably. A new algorithm, anti-Apriori, which based on the proposed encoding method is introduced either. By using some properties of numbers, the itemsets of the database can be transformed into numerical fields. Different from the Apriori algorithm, the new one discovers the association rules from the largest frequent itemset at first, and then all sub itemset, which are also frequent, will be gained without any farther calculation, and all the other small frequent itemset that

must be generated in the Apriori be omitted, and the times of the database scan is also reduced.

CONTENTS

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	vii
	LIST OF TABLES	ix
	LIST OF FIGURES	x
1.	INTRODUCTION	
	1.1 INTRODUCTION TO DATA MINING	1
	1.2 ASSOCIATION RULE MINING	4
	1.3 PROBLEM DEFINITION	6
	1.4 OBJECTIVE OF THE PROJECT	7
2.	LITERATURE REVIEW	
	2.1 CURRENT STATUS OF THE PROBLEM	8
	2.2 PROPOSED SYSTEM AND ADVANTAGES	9
	2.3 HARDWARE REQUIREMENTS	10
	2.4 SOFTWARE REQUIREMENTS	10
	2.5 SOFTWARE OVERVIEW	11

3.	DETAILS OF METHODOLOGY EMPLOYED	
	3.1 DATABASE ENCODING	13
	3.2 APRIORI IMPLEMENTATION	16
	3.3 ANTI-APRIORI IMPLEMENTATION	20
	3.4 FREQUENT ITEMSET MINING	21
	3.5 ASSOCIATION RULES MINING	21
4.	PERFORMANCE EVALUATION	23
5.	CONCLUSION	24
6.	FUTURE ENHANCEMENTS	25
7.	APPENDIX	
	7.1 SOURCE CODE	26
	7.2 SCREEN SHOTS	47
8.	REFERENCES	52

LIST OF TABLES

S.NO.	TITLE	PAGE NO.
3.1.1	Vertical database layout	14
3.1.2	Prime number assignment	15
3.1.3	Final encoded table	15
3.3.1	Frequent Itemset Presentation	20
4.1	Time taken w.r.t Support values	23

LIST OF FIGURES

S.NO.	TITLE	PAGE NO.
3.2.1	Possible Subsets	17
3.2.2	Apriori Process	18
4.1	Graph - Time Taken W.R.T Support Values	23

INTRODUCTION

1. INTRODUCTION

1.1 INTRODUCTION TO DATA MINING

Data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information – information that can be used to increase revenue, cut costs or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

Data

Data are any facts, numbers, or text that can be processed by a computer. Today, organizations are accumulating vast and growing amounts of data in different formats and different databases. This includes: operational or transactional data such as cost, inventory, payroll, and accounting and Meta data - data about the data itself, such as logical database design or data dictionary definitions

Information

The patterns, associations, or relationships among all this data can provide information. For example, analysis of retail point of sale transaction data can yield information on which products are selling and when.

Knowledge

Information can be converted into knowledge about historical patterns and future trends. For example, summary information on retail supermarket

sales can be analyzed in light of promotional efforts to provide knowledge of consumer buying behavior. Thus a manufacturer or retailer could determine which items are most susceptible to promotional efforts.

Data Warehouses:

Dramatic advances in data capture, processing power, data transmission, and storage capabilities are enabling organizations to integrate their various databases into data warehouses. Data warehousing is defined as a process of centralized data management and retrieval. Data warehousing, like data mining, is a relatively new term although the concept itself has been around for years. Data warehousing represents an ideal vision of maintaining a central repository of all organizational data. Centralization of data is needed to maximize user access and analysis. Dramatic technological advances are making this vision a reality for many companies and equally dramatic advances in data analysis software are allowing users to access this data freely.

Elements

Data mining consists of five major elements:

1. Extract, transform, and load transaction data onto the data warehouse system.
2. Store and manage data in a multidimensional database system.
3. Provide data access to business analysis and information technology professional.
4. Analyze the data by application software
5. Present the data in a useful format, such as a graph or table.

Data mining benefits

Data mining is primarily used today by companies with a strong consumer focus - retail, financial, communication and marketing organization. is primarily motivated by decision support problems faced by most business organizations and is described as an important area of research [6], [7] .It enables these companies to determine relationships among "internal" factors such as price, product positioning or staff skills, and "external" factors such as economic indicators, competition, and customer demographics. And it enables them to determine the impact on sales, customer satisfaction and corporate profits. Finally it enables them to "drill down" into the summary information to view detail transactional data.

With data mining, a retailer could use point of sale record of customer purchases to send targeted promotions based on individual's purchase history. By mining demographic data from comment or warranty cards, the retailer could develop products and promotions to appeal to specific customer segments.

1.2 ASSOCIATION RULE MINING

In data mining, finding or learning association rules is a popular and well researched method for discovering relations between variables in large databases. One of the reasons behind maintaining any database is to enable the user to find interesting patterns and trends in the data. For example, in a supermarket, the user can figure out which items are being sold most frequently. But this is not the only type of ‘trend’ which one can possibly think of. The goal of database mining is to automat this process of finding interesting patterns and trends. Once this information is available, we can perhaps get rid of the original database. The output of the data mining process should be a “summary” of the database. The goal is difficult to achieve due to the vagueness associated with the term “interesting”. The solution is to define various types of trends and to look for only those trends in the database. One such type constitutes the association rule.

In the present context, an association rule tells us about the association between two or more items. For example: In 80% of all the cases when people buy bread they also buy milk. This tells us of the association between bread and milk. We represent it as:

Bread => Milk | 80%

This should be read as “Bread means or implies milk, 80% of the time” .

An association rule has two numbers that express the degree of uncertainty about the rule. In association analysis the antecedent and consequent are sets of items (called itemsets) that are disjoint (do not have any items in common).

The first number is called the **support** for the rule. The support is simply the number of transactions that include all items in the antecedent

and consequent parts of the rule. (The support is sometimes expressed as a percentage of the total number of records in the database.)

The other number is known as the **confidence** of the rule. Confidence is the ratio of the number of transactions that include all items in the consequent as well as the antecedent (namely, the support) to the number of transactions that include all items in the antecedent.

Bread => Milk | 80%

Here 80% is the “confidence factor” of the rule.

Association rules can be between more than two items. For example:

Bread, Milk => Jam | 80%

Bread => Milk, Jam | 80%

Given any rule, we can easily find its confidence. For example for the rule:

Bread, Milk => Jam

We count the number say n_1 records that contain bread and milk. Of these how many contain jam as well? Let this be n_2 . Then required confidence is n_2 / n_1 .

This means that the user has to guess which rule is interesting and ask for its confidence. But our goal was to “automatically” find all interesting rules. This is going to be difficult because the database is bound to be very large. We might have to go through the entire database many times to find all the interesting rules.

1.3 PROBLEM DEFINITION

Association rule mining is done in two phases –

1. Frequent itemset extraction
2. Discovery of strong rules

Extracting frequent itemsets is very difficult in large databases. Also, the existing algorithm Apriori uses a top-down approach where it generates candidate item sets of length k from item sets of length $k - 1$. So all subset of every itemset must be generated for finding superior frequent itemset, although many of them may be not useful. This process takes a long time and it also requires thousands of times of database scan. Additionally, the size of database is the main problem of this algorithm. Some modified algorithms of Apriori are proposed to solve this problem but these algorithms also have the database size problem [3].

1.4 OBJECTIVE OF THE PROJECT

To overcome the limitations and address the shortcomings of the current association rules mining algorithms, we propose a way to encode the database [2] where using some properties of numbers; the itemsets of the database can be converted into numerical fields. By this way the size of the database is greatly reduced. When a known modified algorithm is applied in the encoded database, the efficiency is seen to be greatly improved. Also a new algorithm is also proposed that discovers the association rules from the largest frequent itemset at first, and then all subitemsets, which are also frequent, will be found without any calculation and all the other small frequent itemset that must be generated in the Apriori be omitted, and the times of the database scan is also reduced. Test results show the new algorithm based on the encoding database has a lower complexity of time and space.

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 CURRENT STATUS OF THE PROBLEM

The current system of datamining takes care of the simply datamining of information based on classification goals like frequent item sets or classification constraints. Data mining especially association rule discovery tries to find interesting patterns from databases that represent the meaningful relationships between products and customers or other relationships in some other applications. Discovering association rules in these algorithms are usually done in two phases. In the first phase, the frequent itemsets are generated and in the second phase, the interesting rules are extracted from these frequent itemsets. If the support and confidence of a rule is above the minimum threshold, the rule will be interesting. The task of discovering all frequent itemsets is quite challenging especially in the large database because the database may be massive, containing millions of transactions.

A famous algorithm, called Apriori, is proposed in reference [1]. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time. It generates $(k+1)$ candidates by joining frequent k -itemset. So, all the subsets of every itemset must be generated in order to the find superior frequent itemset. Many of them may be not useful and not exploited to find association rules, because some of them have no interesting antecedents or consequents in the rules. This process takes a long time. And it also requires thousands of times of database scan. The complexity of the calculation increases exponentially. Additionally, the size of database is the main problem of this algorithm.

2.2 PROPOSED SYSTEM AND ADVANTAGES

This project uses a method to encode database. Here a record is denoted by only one binary number, so the size of the database is reduced sharply. If some known modified algorithms are used on the database encoded, the efficiency will be improved significantly. Also, a new algorithm based on the proposed encoding method is introduced too. By using some properties of numbers, the itemsets of the database can be converted into numerical fields. Different from the Apriori, the new algorithm called anti-Apriori algorithm uses a "top down" approach, by which it discovers the association rules from the largest frequent itemset at first, and then all subitemsets, which are also frequent will be gotten without any calculation, and all the other small none-frequent itemsets that must be generated in the Apriori will be omitted, and the scan times of the database are also reduced. Test results show the new algorithm based on the encoding database has a lower complexity of time and space.

2.3 HARDWARE REQUIREMENTS

Processor	:	Pentium IV
Speed	:	Above 500 MHz
RAM capacity	:	128 MB
Floppy disk drive	:	1.44 MB
Hard disk drive	:	20 GB
Key Board	:	Samsung 108 keys
Mouse	:	Logitech Optical Mouse
CD Writer	:	52x LG
Printer	:	DeskJet HP
Motherboard	:	Intel
Monitor	:	17" Samsung

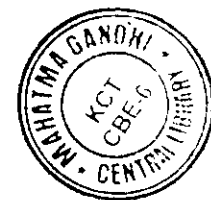
2.4 SOFTWARE REQUIREMENTS

Operating System	:	Windows XP
Front end used	:	Java
Back End	:	SQL Server 2000

2.5 SOFTWARE OVERVIEW

Java:

Java is a high-level, object-oriented programming language that is portable, platform-independent, robust and secure. It derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java solves the problem of platform-independence by using byte code. The Java compiler does not produce native executable code for a particular machine like a C compiler would. Instead it produces a special format called byte code. Java was designed from the ground up to allow for secure execution of code across a network, even when the source of that code was untrusted and possibly malicious. This required the elimination of many features of C and C++. Most notably there are no pointers in Java. Java programs cannot access arbitrary addresses in memory. All memory access is handled behind the scenes by the (presumably) trusted runtime environment. Furthermore Java has strong typing. Variables must be declared, and variables do not change types when you aren't looking. Casts are strictly limited to casts between types that make sense. Thus you can cast an int to a long or a byte to a short but not a long to a Boolean or an int to a String. Java implements a robust exception handling mechanism to deal with both expected and unexpected errors. The worst that an applet can do to a host system is to bring down the runtime environment. It cannot bring down the entire system. Java byte codes can be compiled on the fly to code that rivals C++ in speed using a "just-in-time compiler." Several companies are also working on native-machine-architecture compilers for Java. These will produce executable code that does not require a separate interpreter, and that is indistinguishable in speed from C++.



SQL:

SQL is a database computer language designed for the retrieval and management of data in relational database management systems (RDBMS), database schema creation and modification, and database object access control management. SQL is used as a querying language for querying and modifying data and managing databases. It allows the retrieval, insertion, updating, and deletion of data. It is a set-based, declarative query language, not an imperative language such as C or BASIC. However, there are extensions to Standard SQL which add procedural programming language functionality, such as control-of-flow constructs, They are - PL/SQL, T-SQL, SQL/PSM etc. In addition to the standard SQL/PSM extensions and proprietary SQL extensions, procedural and object-oriented programmability is available on many SQL platforms via DBMS integration with other languages.

DETAILS OF METHODOLOGY EMPLOYED

3. DETAILS OF METHODOLOGY

3.1 DATABASE ENCODING

The presentation of database is an important consideration in almost all algorithms. The most commonly used layout is the horizontal database layout and vertical database layout [8]. In both layouts, the sizes of the database are very large. A large database to be transformed into a smaller one with all properties of its original layout is expected. Database encoding can reduce the size of database and improve the efficiency of algorithms. Instead of maintaining a large table in the transaction database, one table is created with only two columns. The first one is the transaction identifier and another is for the entire items that occur in the transaction. All items in one transaction are converted into only one number that has all properties of these items. By this way the new database is much smaller than the previous one and can be loaded into memory easily. So the cost of memory is reduced. According to the assumption that only one number represents an itemset, when converting an itemset into a number, a measure attribute is defined, which is a numerical attribute associated with every item in each transaction in the database layout. A binary number expresses a numerical attribute, that is, those items that are occurring in one transaction are depicted with 1 and all the other items are represented with 0. The transaction measure value, denoted as $tmv(I_p, T_q)$ is a value of a measure attribute related to an item I_p in a transaction T_q . $tmv(I_p, T_q)=0$ means item I_p does not occur in the transaction T_q , while $tmv(I_p, T_q)=1$ means item I_p occurs in the transaction T_q . In table 1, for example, $tmv(I_4, T_1)$ is equals to 1. Any item I_p in the set of items is encoded as one prime number, denoted as $E(I_p)$. Prime numbers are used because any number except 1 and

themselves cannot divide them. For any item I_p in the transaction T_q , a new measure that denoted as $M(I_p, T_q)$ is equal to the product of $tmv(I_p, T_q)$ and its encoding number $E(I_p)$ is assigned. This value is gotten by equation 2. After this step, for all I_p and T_q , if $M(I_p, T_q)$ equal to 0, then convert $M(I_p, T_q)$ into 1. This operation is described in equation 3. For any transactions, the value M_{T_q} is equal to the multiplication of all $M(I_p, T_q)$. The value of M_{T_q} is represented in equation 4.

$$M(I_p, T_q) = tmv(I_p, T_q) E(I_p) \quad (2)$$

$$\text{For all } (I_p, T_q) \text{ If } M(I_p, T_q) = 0 \Rightarrow M(I_p, T_q) = 1 \quad (3)$$

$$M_{T_q} = \prod M(I_p, T_q) \quad (4)$$

for any itemset $I=(I_{p1}, I_{p2}, \dots, I_{pn})$, there is one value denoted as M_I is equal to the multiplication of all $E(I_p)$ if its I_p occur in I , as described in equation 5. The value M_I shows the number corresponds to itemset I . And then this number can be used to instead of itemset I .

$$M_I = \prod_{\forall I_p \in I} E(I_p) \quad (5)$$

With this encoding, instead of maintaining all $tmv(I_p, T_q)$ for every item and transaction, the value M_I can be stored for every transaction.

T_{ID}	I_1	I_2	I_3	I_4
1	1	0	0	1
2	0	0	0	1
3	1	1	1	0
4	0	0	1	1

Table 3.1.1: Convert Vertical Database Layout

I_p	E_{I_p}
I_1	7
I_2	5
I_3	3
I_4	2

Table 3.1.2: Prime number assignment

T_{id}	M
1	14
2	2
3	105
4	6

Table 3.1.3: Final encoded table

The database has been represented in table 3.1.1. There is one column for T_{id} and four columns for items. In table 3.1.2, the set of items in the left column, and prime numbers correspond to them in the right column. By applying equations 2,3,4, the table 3.1.3 has only two columns and for every transaction, MT_{id} has replaced some binary numbers for itemset. Under this situation if it wants to know, for example, whether the itemset $I=\{I_2, I_3\}$ is in 3th transaction, as for $MT_{id=3}=105$ can be divided by $M_I=5*3=15$, it can say that the itemset I has occurred in transaction ($T_{id}=3$). This means for verifying any itemset I_p presence in transaction T_q , dividing the value MT_q by M_{I_p} is sufficient. By using this encoding method, the efficiency of some known modified algorithms such as Apriori [4] can be improved significantly.

3.2 APRIORI IMPLEMENTATION:

In data mining, Apriori is a classic algorithm for discovering association rules. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation). Other algorithms are designed for finding association rules in data having no transactions, or having no timestamps.

As is common in association rule mining, given a set of *itemsets*, the algorithm attempts to find subsets which are common to at least a minimum number C of the itemsets. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as *candidate generation*), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.

Apriori uses breadth-first search and a hash tree structure to count candidate item sets efficiently. It generates candidate item sets of length k from item sets of length $k - 1$. Then it prunes the candidates which have an infrequent sub pattern. According to the downward closure lemma, the candidate set contains all frequent k -length item sets. After that, it scans the transaction database to determine frequent item sets among the candidates. For determining frequent items quickly, the algorithm uses a hash tree to store candidate itemsets. This hash tree has item sets at the leaves and hash tables at internal nodes. Note that this is not the same kind of hash tree used in for instance p2p systems

Frequent Itemset Generation: Apriori

Given an itemset $I = \{a, b, c, d, e\}$. If an item set is frequent, then all of its subsets must also be frequent and vice-versa.

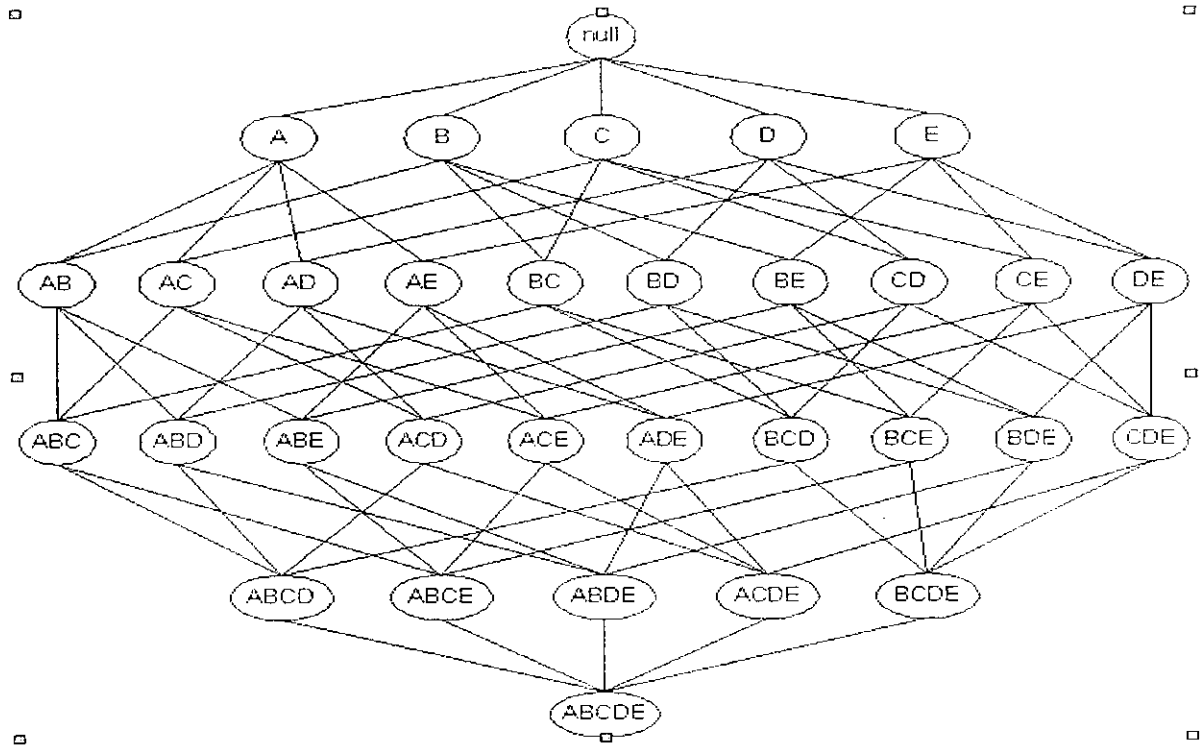


Fig 3.2.1 Possible subsets of itemset $I = \{a, b, c, d, e\}$

Process

- Determine large 1-itemsets.
- Repeat until no new large 1-itemsets are identified.
- Generate $(k+1)$ length candidate itemsets from length k large itemsets.
- Prune candidate itemsets that are not large.
- Count the support of each candidate itemset.
- Eliminate candidate itemsets that are small.

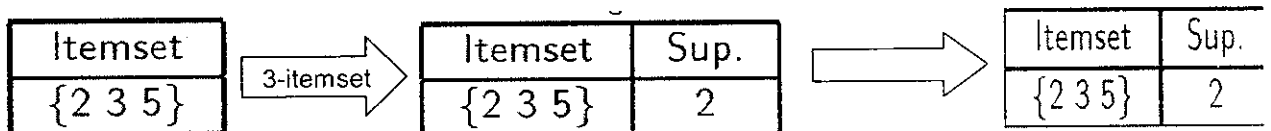
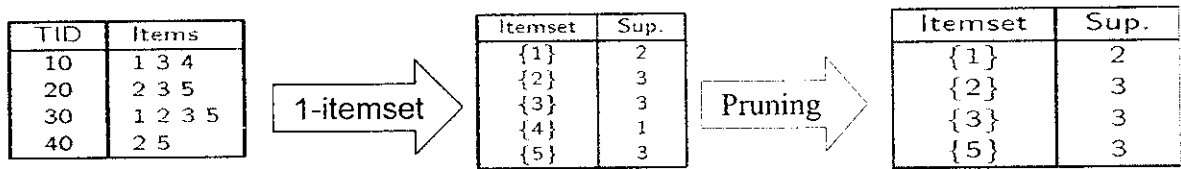


Fig 3.2.2 Apriori process

Apriori Property

- If an itemset I does not satisfy the minimum support threshold, min_sup , the I is not frequent, that is, $P(I) < min_sup$
- If an item A is added to the itemset I , then the resulting itemset (i.e., $I \cup A$) cannot occur more frequently than I . Therefore, $I \cup A$ is not frequent either, that is, $P(I \cup A) < min_sup$.

Example:

This example suggests the process of selecting or generating a list of likely ordered serial candidate item sets. The technique's goal is to construct a set of k node ordered serial item sets from $k - 1$ length item sets. For example, with $k = 4$, suppose there are two such sets of length $k - 1$...

$$A \rightarrow B \rightarrow C, \quad \text{and} \quad A \rightarrow B \rightarrow D,$$

Two candidate item sets are generated, namely

$$A \rightarrow B \rightarrow C \rightarrow D \quad \text{and} \quad A \rightarrow B \rightarrow D \rightarrow C.$$

Algorithm:

Association rule mining is to find out association rules that satisfy the predefined minimum support and confidence from a given database. The problem is usually decomposed into two sub problems. One is to find those itemsets whose occurrences exceed a predefined threshold in the database; those itemsets are called frequent or large itemsets. The second problem is to generate association rules from those large itemsets with the constraints of minimal confidence. **Apriori** algorithm:

- First pass: counts item occurrences to determine the large 1-itemsets.
- 2nd and subsequent passes:
 - for ($k=2$; L_{k-1} not empty; $k++$)
 - $C_k = \text{apriori-gen}(L_{k-1})$; // (New candidates)
 - forall transactions t in D do
 - $C_t = \text{subset}(C_k, t)$ // (Candidates contained in t)
 - forall candidates c in C_t do
 - $c.\text{count}++$
 - $L_k = \{c \text{ in } C_k \mid c.\text{count} \geq \text{minsupport} \}$

- Answer = Union_k (L_k)

In apriori, candidate itemsets to be counted are generated using only itemsets found large in the previous pass, without considering the transaction that is in the database.

3.3 ANTI-APRIORI ALGORITHM:

In Apriori algorithm, itemset mining starts from finding frequent 1-itemset. Here, finding frequent itemset is done in bottom to up manner. Different from this algorithm, a new algorithm is called as anti-Apriori is proposed, in which the discovery of frequent itemset is done in top-down style. It means that the large frequent itemset are found at first and then all of their subsets (that are certainly frequent) are extracted [5]. In this technique, it is supposed that any frequent itemset must be at least one time occurs in the transactions lonely (without any other items that are not member of that itemset). In other words, if itemset (I₁, I₂, I₃) is frequent, the itemset at least in one transaction without any other items, such as shown in table 3.3.1

	I ₁	I ₂	I ₃
T _{ID}	1	1	1	0000000000

Table 3.3.1 : Frequent Itemset Presentation

It has two stages:

1. Frequent Itemset Mining
2. Association Rules Mining

3.4 FREQUENT ITEMSET MINING

In this method for every transaction T_q , the GCD (greatest common divisors) between MT_q and M_T correspond to other transactions are computed and frequency of these greatest common divisors are stored in GCD-set. GCD-set is the candidate for frequent set. For any GCD in GCD-set, if its frequency is above the required (frequent GCD itemset). For every transaction is maintained, a set is denoted as GCD T_{id} , composed of GCDs and frequency of any GCD. For example, if GCD-set and frequency between first transaction and other transactions is equal to $GCD1 = \{(42,3), (6,8), (21,2), (15,4), (105,1)\}$ and the required threshold for support is equal to 7 and then the set (6,8) has a frequency is equal to 8, greater than 7, and then 6 is inserted into FGCD-set. The itemset mining schema of this algorithm is given in Algorithm 1. Notation $M[i]$ is used to represent the value M for i^{th} transaction and s is the required threshold for the support. The candidates are gotten from the GCD-set and their frequency compose the set CF . Any GCDs in CF having a frequency greater or equal to s are appended into FGCD-set.

3.5 ASSOCIATION RULES MINING

Discovering association rules based on all FGCD, which has been found in the previous phase. Measure M corresponds to any frequent itemset is maintained in FGCD-set. Every measure M in FGCD-set is decomposed into the multiplication of prime number and each prime number corresponds to one item. The itemset that correspond to M is identical and frequent, and all subset of it must be request. Every M in FGCD-set is decomposed into a candidate head Y and a body $X=M/Y$. This algorithm iteratively generates candidate heads C_{k+1} of $k+1$ size, starting with $k=1$. If the head and the body are interesting and valuable, the confidence C of the rule $X \Rightarrow Y$ is computed

as the quotient of the supports for the itemset. $C = \text{Support}(M) / \text{Support}(X)$
Support(X) is computed by counting the number of MT_{id} that can be
divided by X). If any rule has a C greater than or equal to the given
threshold for confidence, the rule will be appended into association rules.
The association rules mining schema of this algorithm is given in Algorithm
2. Notation $M[i]$ is used to represent the value M for i^{th} member in FGCD-
set and τ for the required threshold for the confidence.

PERFORMANCE EVALUATION

4. PERFORMANCE EVALUATION

The performance of the data mining is studied in this module. The efficiency and time taken to complete data mining operations are taken into the consideration and studied. The scalability problem with respect to growing number of transactions in the database are carefully studied and evaluated. Implementation on Food mart dataset shows that by applying this encoding method, the size of database can be reduced at least half. At the meantime, experiment shows that the speed of the algorithm by using this encoding method has been increased especially in the sparse databases. The improvement for the speed of discovering association rules is at least 25%. Although the efficiency of this algorithm is achieved at the cost of losing about 5% of the association rules, the bottleneck for the frequent patterns discovery in the large database is overcome. Under some condition, the efficiency for discovering of association rules is more important than the accuracy, and this algorithm has a lower complexity of time and space.

Support	Time	
	Apriori	Anti-Apriori
1	100	100
2	90	85
3	88	80
4	85	74
5	80	70
6	75	63
7	72	62
8	67	55
9	52	42
10	48	38
11	45	37
12	42	30
13	38	26

Table 4.1 Time Vs. Support

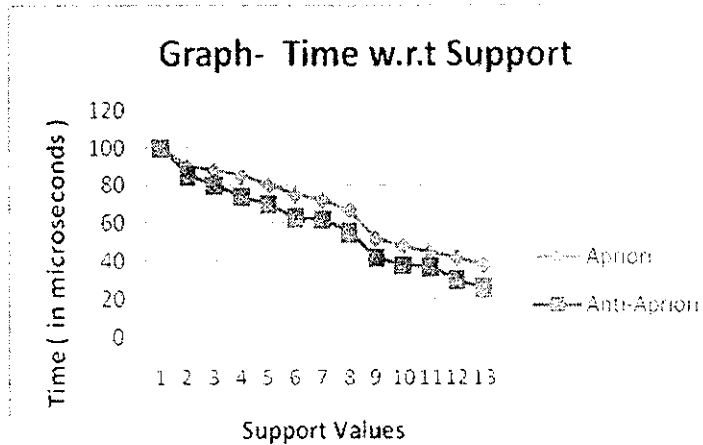


Fig 4.1 Graph – Time w.r.t Support values

CONCLUSION

5. CONCLUSION

The speed of the anti-Apriori algorithm by using this encoding method is found to be increased especially in the sparse databases. The improvement for the speed of discovering association rules is at least 25%. The comparison with this algorithm to the Apriori is shown. Although the efficiency of this algorithm is achieved at the cost of losing about 5% of the association rules, the bottleneck for the frequent patterns discovery in the large database is overcome.

FUTURE ENHANCEMENTS

6. FUTURE ENHANCEMENTS

Although the efficiency of anti-Apriori algorithm is achieved at the cost of losing about 5% of the association rules, the bottleneck for the frequent patterns discovery in the large database is overcome. Under some condition, the efficiency for discovering of association rules is more important than the accuracy, and this algorithm has a lower complexity of time and space. So, in Future there must be good blend of compromise in efficiency and accuracy of discovering of association rules mining so that required quality of service threshold can be adhered to.

APPENDICES

7. APPENDIX

7.1 Sample Code:

```
import java.io.*;
```

```
import java.util.*;
```

```
import java.awt.*;
```

```
import java.awt.event.*;
```

```
import javax.swing.*;
```

```
import java.sql.*;
```

```
import pack1.*;
```

```
class AntiApriorimdProcess extends JFrame implements  
ActionListener
```

```
{
```

```
    JLabel l1,l2,l3,l4,l5,l6,l7;
```

```
    JComboBox c1,c2,c3,c4,c5;
```

```
    JTextField t1,t2;
```

```
    JButton b,b1;
```

```
    static JTextArea ta;
```

```
    JScrollPane sp;
```

```
    String driver,query="",id,name;
```

```
    Connection cn;
```

```
    Statement st,st1,st2;
```

```
    ResultSet rs,rs1,rs2;
```

```
    Boolean rec,rec1;
```

```

center c;

int y=0,r=0,s=0,d=0;

String br="";

private final int HT=1; // state of tree node (hash table or
private final int IL=2; // itemset list)

int N=0; // total item #

int M=0; // total transaction #

Vector largeitemset=new Vector();

Vector candidate=new Vector();

Hashtable ht;

int minsup;

String fullitemset;

String configfile="config.txt";

String transafile="transa.txt";

String time="";

Iterator iter;

Iterator iter1;

TreeMap tree;//=new TreeMap();

String obj="";

ArrayList frequent,columnname;

long diff;

public AntiApriorimdProcess()

{

    super("AntiApriori Algorithm",true, true,true,true);

```

```
c=new center();
l1=new JLabel("Year");
l2=new JLabel("Date");
l3=new JLabel("Region");
l4=new JLabel("Store");
l5=new JLabel("Brand Name");
l6=new JLabel("Minsup");
l7=new JLabel("Table name");
c1=new JComboBox();
c1.addItem("Select the Year");
c1.addItem("1997");
c1.addItem("1998");
c2=new JComboBox();
c2.addItem("All Date");
c2.setSelectedIndex(0);
c3=new JComboBox();
c4=new JComboBox();
c4.addItem("All Store");
c2.setSelectedIndex(0);
c5=new JComboBox();
c1.setFont(new Font("Courier New",1,12));
c2.setFont(new Font("Courier New",1,12));
c3.setFont(new Font("Courier New",1,12));
c4.setFont(new Font("Courier New",1,12));
```

```
c5.setFont(new Font("Courier New",1,12));
t1=new JTextField(14);
t2=new JTextField(14);
t1.setFont(new Font("Courier New",1,12));
t2.setFont(new Font("Courier New",1,12));
b=new JButton("Submit");
b.setMnemonic('S');
b1=new JButton("Graph");
b1.setMnemonic('G');
sp=new JScrollPane();
ta=new JTextArea();
JPanel p=new JPanel();
p.setLayout(new GridLayout(8,2));
p.add(l1); p.add(c1);
p.add(l2); p.add(c2);
p.add(l3); p.add(c3);
p.add(l4); p.add(c4);
p.add(l5); p.add(c5);
p.add(l6); p.add(t1);
//p.add(l7); p.add(t2);
frequent=new ArrayList();
columnname=new ArrayList();
getContentPane().setLayout(null);
getContentPane().add(p);
```

```

getContentPane().add(b); getContentPane().add(b1);
getContentPane().add(sp);
p.setBounds(20,10,300,170);
b.setBounds(100,180,90,25); b1.setBounds(200,180,90,25);
connection();
regionload();
brandload();
b.addActionListener(this);
b1.addActionListener(this);
c1.addItemListener(new java.awt.event.ItemListener(){
public void itemStateChanged(java.awt.event.ItemEvent ie){
        c1itemstatechanged();        }});
c2.addItemListener(new java.awt.event.ItemListener(){
public void itemStateChanged(java.awt.event.ItemEvent ie){
        c2itemstatechanged();        }});
c3.addItemListener(new java.awt.event.ItemListener(){
public void itemStateChanged(java.awt.event.ItemEvent ie){
        c3itemstatechanged();        }});
c4.addItemListener(new java.awt.event.ItemListener(){
public void itemStateChanged(java.awt.event.ItemEvent ie){
        c4itemstatechanged();        }});
c5.addItemListener(new java.awt.event.ItemListener(){
public void itemStateChanged(java.awt.event.ItemEvent ie){
        c5itemstatechanged();        }});

```

```

        sp.setBounds(2,220,348,225);
        setSize(360,500);
    }

    public void actionPerformed(ActionEvent ae)
    {
        if(ae.getSource()==b)
        {
            try{
                submit();
                process();
                genAssoRule1 r=new genAssoRule1();
                r.genAssoRule1(frequent);
            }
            catch(Exception e){}
            sp.setViewportView(ta);
        }
        if(ae.getSource()==b1)
        {
            Enumeration name;
            name=ht.keys();
            int size=ht.size();
            String f[]=new String[size];
            String va[]=new String[size];

```



```

int val[]=new int[size];
int i=0;
while(name.hasMoreElements())
{
    f[i]=(String)name.nextElement();
    va[i]=(String)ht.get(f[i]);
    val[i]=Integer.parseInt(va[i]);
    i++;
}
(new ChartPanel(val,f,time)).display();
}
}

```

```

public void connection()
{
    driver="Jdbc:Odbc:pattern";
    try
    {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        cn=DriverManager.getConnection(driver,"","");

st=cn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSe t.CONCUR_UPDATABLE);

```

```

st1=cn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);

st2=cn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
ResultSet.CONCUR_UPDATABLE);

    }

    catch(Exception e)

    {

        System.out.println("Error "+e);

    }

}

public void regionload()

{

    try

    {

        query="select region_id,sales_city from region";

        ResultSet rs1=st.executeQuery(query);

        rec=rs1.first();

        c3.addItem("Select region");

        while(rec)

        {

            id=rs1.getString(1);

            name=rs1.getString(2);

            int l=6-id.length();

            String idname=id+c.space(l)+"* "+name;

```

```
        c3.addItem(idname);
        rec=rs1.next();
    }
}
catch(Exception ee){
    System.out.println("Error reg "+ee);
}
}
```

```
public void brandload()
{
    try
    {
        query="select distinct(brand_name) from product";
        rs=st.executeQuery(query);
        rec=rs.first();
        c5.addItem("Select the Brand");
        while(rec)
        {
            id=rs.getString(1);
            c5.addItem(id);
            rec=rs.next();
        }
    }
}
```

```

        catch(Exception ee){
            System.out.println("Error! "+ee);
        }
    }
    public void c1itemstatechanged()
    {
        try
        {
            java.text.SimpleDateFormat sf = new
            java.text.SimpleDateFormat("dd-MMM-yyyy");

            String year=c1.getSelectedItem().toString();

            if(!year.startsWith("Select"))
            {

                y=Integer.parseInt(year);

                c2.removeAllItems();

                query = "select time_id,the_date from time_by_day where the_year in ("
                +y+)";

                rs=st.executeQuery(query);

                rec=rs.first();

                if(rec) c2.addItem("All Date");

                while(rec)
                {

                    id=rs.getString(1);

                    int l=6-id.length();

                    String idname=id+c.space(l)+"* "+sf.format(rs.getDate(2));

```

```

        c2.addItem(idname);
        rec=rs.next();
    }
    rs.close();
}
}
catch(Exception ee)
{
    System.out.println("C1Error "+ee );
}
}
public void c2itemstatechanged()
{
    try
    {
        String str=c2.getSelectedItem().toString();
        if(!str.startsWith("All"))
        {
            String ss[]=str.split("\\*");
            str=ss[0].trim();
            d=Integer.parseInt(str);
        }
        else
            d=0;
    }
}

```

```

        catch(Exception ee)
        {
    }
}

public void c3itemstatechanged()
{
    try
    {
        String str=c3.getSelectedItem().toString();
        String ss[]=str.split("\\*");
        str=ss[0].trim();
        r=Integer.parseInt(str);
        query="select store_id,store_name from store where region_id in (" +r+" )";
        rs=st.executeQuery(query);
        rec=rs.first();
        c4.removeAllItems();
        c4.addItem("All Store");

        while(rec)
        {
            id=rs.getString(1);
            name=rs.getString(2);
            int l=6-id.length();
            String idname=id+c.space(l)+"* "+name;

```

```

        c4.addItem(idname);
        rec=rs.next();
    }
    rs.close();
}
catch(Exception ee)
{
}
}

public void c4itemstatechanged()
{
    try
    {
        String str=c4.getSelectedItem().toString();
        if(!str.startsWith("All"))
        {
            String ss[]=str.split("\\*");
            str=ss[0].trim();
            s=Integer.parseInt(str);
        }
    }catch(Exception ee){}
}

public void c5itemstatechanged()
{

```

```

        String str=c5.getSelectedItem().toString();
        br=str;
    }

public void submit()
    {
        try
        {
            if(d==0)
            {
                query="select a.product_id,b.product_name,a.time_id,a.store_id from
                sales_fact_" +y+ " a,product b where a.product_id=b.product_id and
                a.store_id="+s+ " and b.brand_name='"+br+ "'";

            }
            else
            {
                query="select a.product_id,b.product_name,a.time_id,a.store_id from
                sales_fact_" +y+ " a,product b where a.product_id=b.product_id and
                a.store_id="+s+ " and b.brand_name='"+br+ "' and a.time_id="+d+ "'";

            }
            st.executeUpdate("delete tempmart");
            rs=st.executeQuery(query);
            rec=rs.first();
            while(rec)
            {

```



```

query="Insert into tempmart(pid,pname,tid,sid) values(" +rs.getInt(1)+ ","
+rs.getString(2)+ "," +rs.getInt(3)+ "," +rs.getString(4)+ ")";

        st1.executeUpdate(query);

        rec=rs.next();

    }

query="delete from temporary";

st2.executeUpdate(query);

//System.out.println("temporary ");

query="truncate table transactdata";

st2.executeUpdate(query);

query="select distinct(pname) from tempmart";

rs=st.executeQuery(query);

rs.last();

int setitem=rs.getRow();

rs.first();

String checkfull="";

tree=new TreeMap();

TreeSet ts=new TreeSet();

int val;

int k1;

while(true){

val=Numbers();

for(k1=2; k1 < val ;k1++ )

    {

```

```

        int n = val%k1;
        if (n==0)
            {
                break;
            }
        }
        if(k1 == val)
            {
                ts.add(val);
            }
        if(ts.size()==setitem)
            break;
    } //end of while
    Iterator iter=ts.iterator();
while(rs.next() && iter.hasNext())
{
    int cross=Integer.parseInt(iter.next().toString());
    String name=rs.getString(1);
    query="insert into temporary values('"+name+"','"+cross+"')";
    st1.executeUpdate(query);
    tree.put(name,val);
} //end of while
rs=st.executeQuery("select distinct(pname) from tempmart");
rs.last();

```

```

N=rs.getRow();
rec=rs.first();
if(rec)
{
//st1.executeUpdate("drop table transact");
/*query="create table transact("
while(rec)
{
query=query+nullspace(rs.getString(1))+ " numeric(2) default 0";
rec=rs.next();
if(rec) query=query+", ";
}
query=query+")";
st.executeUpdate(query);*/

rs=st.executeQuery("select tid,sid from tempmart group by tid,sid");
rs.last();
M=rs.getRow();
System.out.println("Row "+M);
rec=rs.first();
while(rec)
{
query="select distinct(pname) from tempmart where tid=" +rs.getInt(1)+ "
and sid=" +rs.getInt(2);
rs1=st1.executeQuery(query);

```

```

rec1=rs1.first();
ArrayList al=new ArrayList();
//System.out.println("lengthfirst ");
while(rec1)
{
    al.add(nullspace(rs1.getString(1)));
    rec1=rs1.next();
}
Iterator itr=al.iterator();
//query="Insert into transact";
String field="(";
//String val=" values(";
while(itr.hasNext())
{
    field=field+itr.next().toString();
    //val=val+"1";
    if(itr.hasNext())
    {
        field=field+",";
        //val=val+",";
    }
    else
    {
        field=field+");";
    }
}

```

```

        //val=val+"");
    }
}
//query=query+field+val;
//st1.executeUpdate(query);
//System.out.println(field);
field=field.replace("(","");
field=field.replace(")","");
Set set=tree.entrySet();
int itemvalue=1;
if(field.contains(","))
{
    String fieldspl[]=field.split(",");
    for(int k=0;k<fieldspl.length;k++)
    {
        iter=set.iterator();
        String values=fieldspl[k];

        while(iter.hasNext())
        {
            Map.Entry me=(Map.Entry)iter.next();
            String keyvalue=me.getKey().toString();
            keyvalue=keyvalue.replaceAll(" ", "");

if(values.equals(keyvalue))

```

```

    {
        int temp=Integer.parseInt(me.getValue().toString());
        itemvalue=temp*itemvalue;
        //System.out.println("if "+itemvalue);
    }//end of if
} //end of while
} //end of for
query="insert into transactdata(transactvalue) values("+itemvalue+")";
    stl.executeUpdate(query);
        } //end of if
    else
    {
        iter=set.iterator();
        while(iter.hasNext())
        {
            Map.Entry me=(Map.Entry)iter.next();
            String keyvalue=me.getKey().toString();
            keyvalue=keyvalue.replaceAll(" ", "");
            if(field.equals(keyvalue))
            {
                int temp=Integer.parseInt(me.getValue().toString());

            itemvalue=itemvalue*temp;

            //System.out.println("else "+itemvalue);

```

```

        query="insert into transactdata(transactvalue)
values("+itemvalue+")";

        st1.executeUpdate(query);
    }
}

rec=rs.next();
}

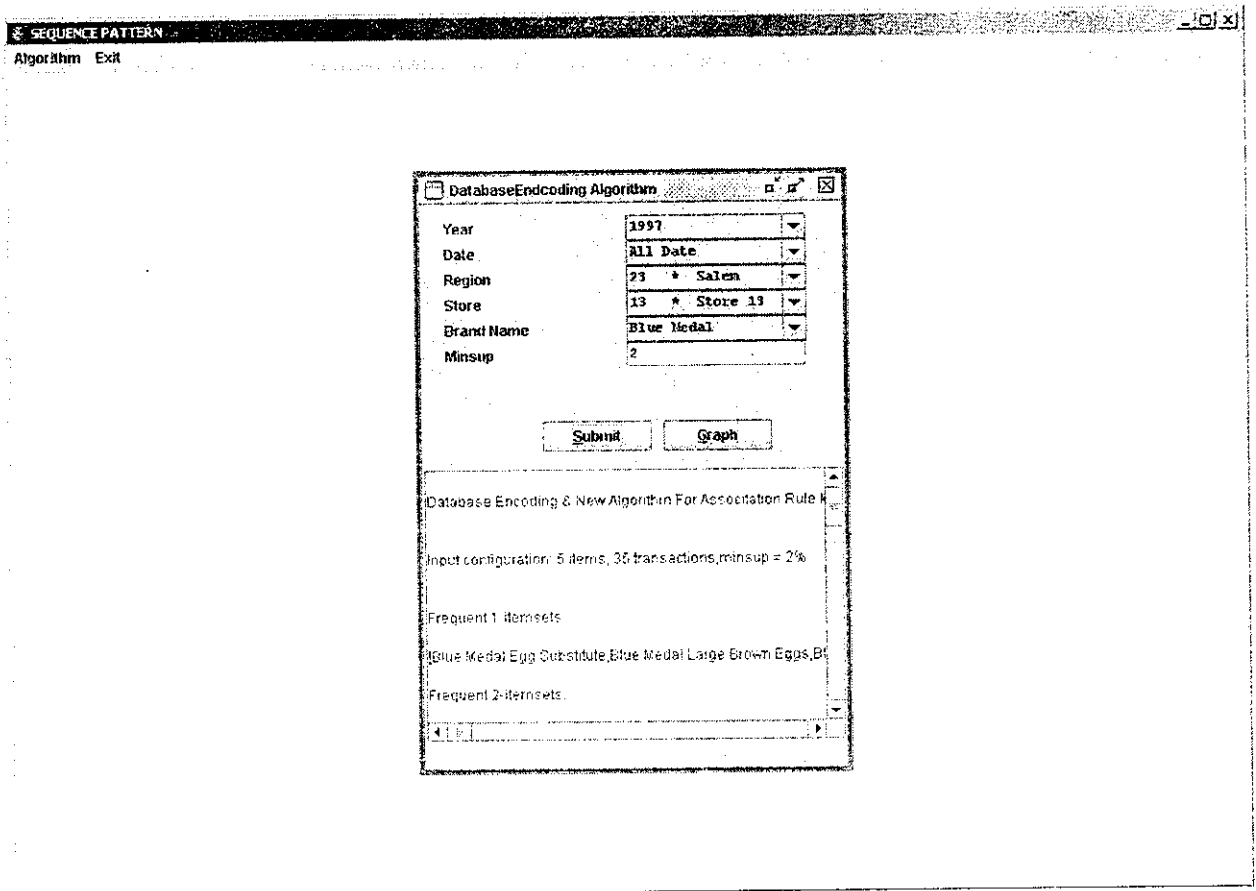
//field=field.substring(field.lastIndexOf(field.length));
//System.out.println("field "+field);
}
}

catch(Exception e)
{
    System.out.println("Errorrs "+e);
} } }

```

7.2 SAMPLE SCREENSHOTS

7.2.1 APRIORI IMPLEMENTATION



7.2.2 ANTI-APRIORI IMPLEMENTATION

The screenshot shows a window titled "SEQUENCE PATTERN" with a menu bar containing "Algorithm" and "Exit". Inside the window, there is a sub-window titled "AntiApriori Algorithm" with the following parameters:

Year	1997
Date	All Date
Region	23 * Salem
Store	13 * Store 13
Brand Name	Blue Medal
Minsup	2

Below the parameters are two buttons: "Submit" and "Graph".

The output area shows the following text:

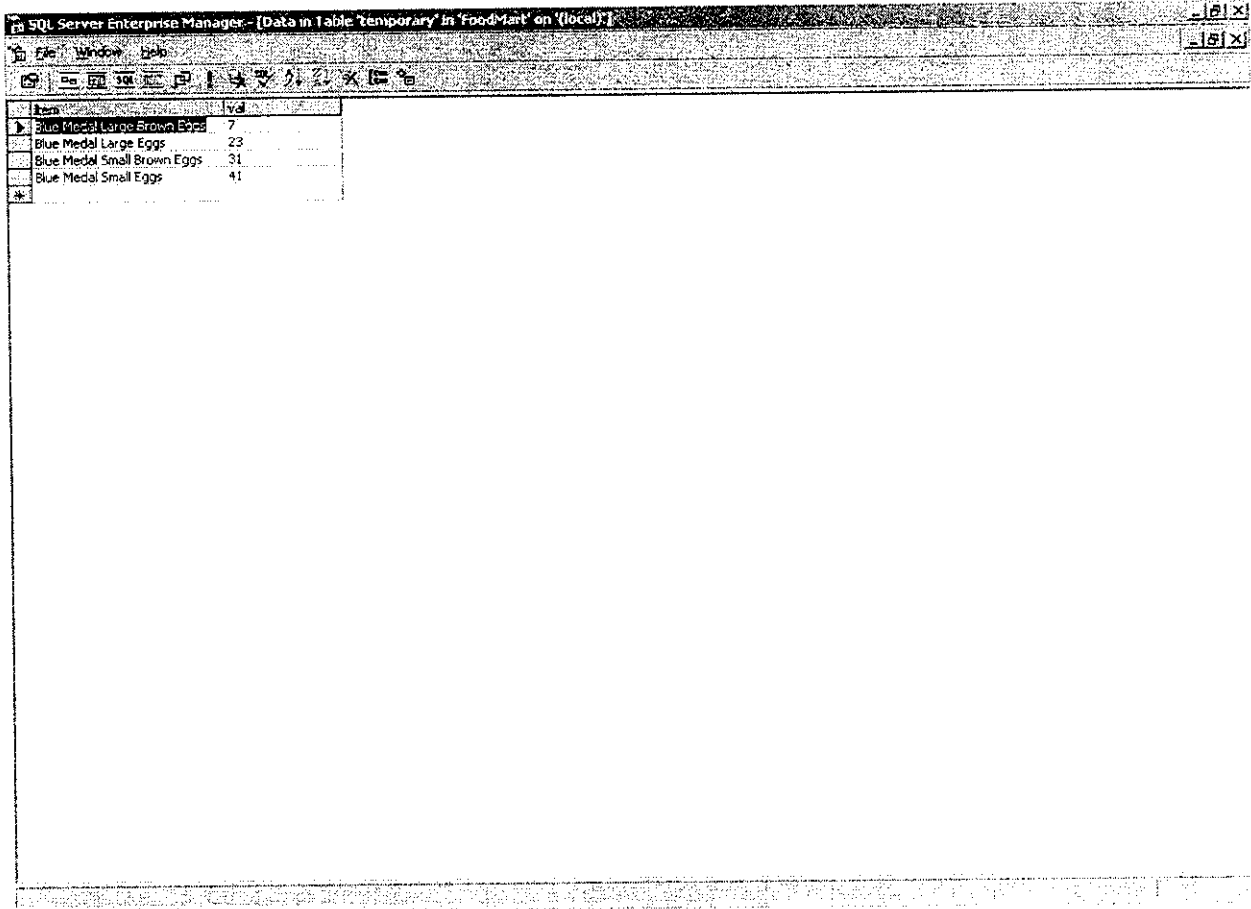
```
Execution time is: 68 milliseconds.  
End  
Strong Rules  
(Blue Medal Egg Substitute=>Blue Medal Egg Substitute E  
(Blue Medal Egg Substitute=>Blue Medal Egg Substitute E  
(Blue Medal Egg Substitute=>Blue Medal Egg Substitute E  
(Blue Medal Egg Substitute=>Blue Medal Large Brown Egg  
(Blue Medal Egg Substitute=>Blue Medal Large Brown Egg  
(Blue Medal Egg Substitute=>Blue Medal Large Brown Egg  
(Blue Medal Egg Substitute=>Blue Medal Large Eggs Blue  
(Blue Medal Egg Substitute=>Blue Medal Large Eggs Blue  
(Blue Medal Egg Substitute=>Blue Medal Small Brown Egg  
(Blue Medal Egg Substitute=>Blue Medal Small Brown Egg
```

7.2.3 TRANSACTION DATABASE

SQL Server Enterprise Manager - [Data in Table 'tempmart' in 'FoodMart' on '(local)']

pid	psname	tid	sid
77	Blue Medal Small Brown Eggs	405	13
77	Blue Medal Small Brown Eggs	519	13
77	Blue Medal Small Brown Eggs	520	13
77	Blue Medal Small Brown Eggs	602	13
77	Blue Medal Small Brown Eggs	553	13
77	Blue Medal Small Brown Eggs	654	13
77	Blue Medal Small Brown Eggs	728	13
78	Blue Medal Large Brown Eggs	373	13
78	Blue Medal Large Brown Eggs	520	13
78	Blue Medal Large Brown Eggs	574	13
78	Blue Medal Large Brown Eggs	610	13
78	Blue Medal Large Brown Eggs	709	13
78	Blue Medal Large Brown Eggs	718	13
79	Blue Medal Small Eggs	377	13
79	Blue Medal Small Eggs	419	13
79	Blue Medal Small Eggs	427	13
79	Blue Medal Small Eggs	446	13
79	Blue Medal Small Eggs	473	13
79	Blue Medal Small Eggs	464	13
79	Blue Medal Small Eggs	584	13
79	Blue Medal Small Eggs	574	13
79	Blue Medal Small Eggs	714	13
80	Blue Medal Large Eggs	400	13
80	Blue Medal Large Eggs	446	13
80	Blue Medal Large Eggs	495	13
80	Blue Medal Large Eggs	574	13
81	Blue Medal Egg Substitute	386	13
81	Blue Medal Egg Substitute	389	13
81	Blue Medal Egg Substitute	453	13
81	Blue Medal Egg Substitute	464	13
81	Blue Medal Egg Substitute	550	13
81	Blue Medal Egg Substitute	605	13
81	Blue Medal Egg Substitute	714	13
81	Blue Medal Egg Substitute	722	13
81	Blue Medal Egg Substitute	711	13
77	Blue Medal Small Brown Eggs	373	13
77	Blue Medal Small Brown Eggs	574	13
78	Blue Medal Large Brown Eggs	471	13
78	Blue Medal Large Brown Eggs	654	13
79	Blue Medal Small Eggs	438	13
79	Blue Medal Small Eggs	446	13
79	Blue Medal Small Eggs	695	13
80	Blue Medal Large Eggs	386	13
80	Blue Medal Large Eggs	491	13
81	Blue Medal Egg Substitute	438	13

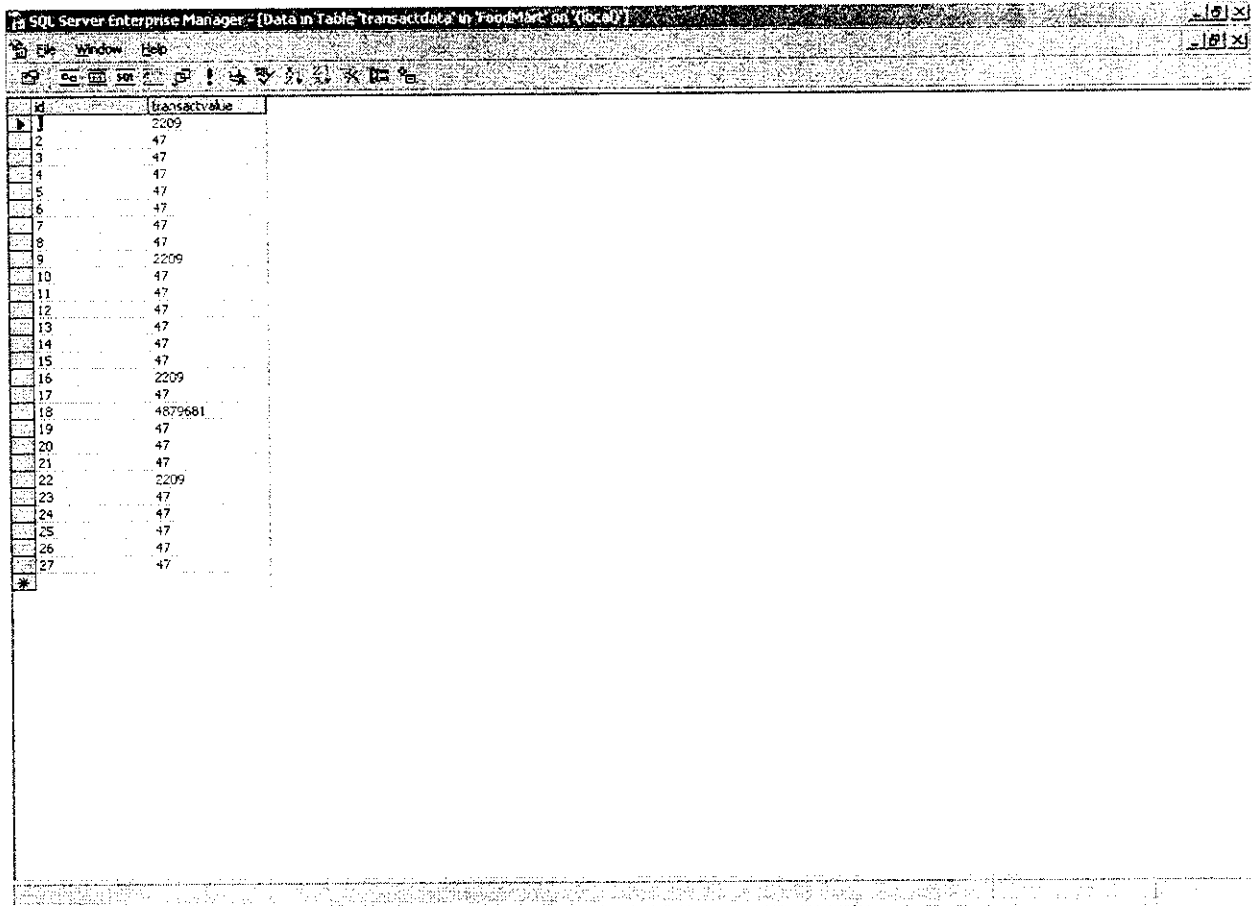
7.2.4 PRIME NUMBER ASSIGNMENT



The screenshot shows a window titled "SQL Server Enterprise Manager - [Data in Table 'temporary' in 'FoodMart' on '(local)']". The window contains a table with the following data:

Item	Qty
Blue Medal Large Brown Eggs	7
Blue Medal Large Eggs	23
Blue Medal Small Brown Eggs	31
Blue Medal Small Eggs	41

7.2.5 FINAL ENCODED TABLE



The screenshot displays a window titled "SQL Server Enterprise Manager - [Data in Table 'transactdata' in 'FoodMart' on '(local)']". The window contains a data grid with two columns: 'id' and 'transactvalue'. The data is as follows:

id	transactvalue
1	2209
2	47
3	47
4	47
5	47
6	47
7	47
8	47
9	2209
10	47
11	47
12	47
13	47
14	47
15	47
16	2209
17	47
18	4879681
19	47
20	47
21	47
22	2209
23	47
24	47
25	47
26	47
27	47

REFERENCES

8.REFERENCES

- [1] Agrawal R., Imielinski T., Swami A. (1993): Mining association rules between sets of items in large databases. | Proc. ACM SIGMOD Int. Conf. Management of Data, Washington, pp.207-216.
- [2] Tong Wang, Pilian He, "Database Encoding And An Anti-apriori Algorithm For Association Rules Mining", in Proceedings of the Fifth International Conference on Machine Learning and Cybernetics IEEE, 2006.
- [3] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. Inkeri Verkamo. Fast Discovery of Association Rules in Large Databases. In Advances in Knowledge Discovery and Data Mining, pages 307-328. AAAI Press, 1996.
- [4] Margaret H Dunham, Data Mining Introductory and Advanced Topics, Tsinghua University Press, Beijing, 2003.
- [5] C.L. Blake and C.J. Merz, UCI Repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences, <http://www.ics.uci.edu/~mlearn/MLRepository.html>, 1998.
- [6] S. Tsur. Data dedging. IEEE Data Engineering Bulletin, 13(4):58-63, December 1990.
- [7] J. T-L. Wang, G-W. Chirn, T. G. Marr, B. Shapiro, D. Shasha, and L. Zhang. Combinatorial pattern discovery for scientific data: some preliminary results. In Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, pages 115-125, Minneapolis, MN, May 24-27 1994.

[8] M.J. Zaki. Scalable algorithms for association mining . IEEE transactions of knowledge and data engineering, 12(3):372-390, May/June 2000