FREQUENT SET CALCULATION FOR DATA MINING

By

S.V.Manisekaran

Reg. No. 71203405008

of

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE- 641006.

K C T LIBRARY Coimbatore-641 006

P-1535

A PROJECT REPORT

Submitted to the

FACULTY OF INFORMATION AND COMMUNICATION ENGINEERING

In partial fulfillment of the requirements

for the award of the degree

of

MASTER OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

June, 2005

BONAFIDE CERTIFICATE

Certified that this project report titled **FREQUENT SET CALCULATION FOR DATA MINING** is the bonafide work of **Mr. S.V.Manisekaran** who carried out the research under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report of dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

GUIDE **HEAD OF THE DEPARTMENT**

The candidate with University Register No. **71203405008** was examined by us in Project Viva-Voce examination held on 22/6/05

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

Association rule mining plays a vital role in the commercial and research environment. The support and confidence of various item sets of products have to be mined from a huge database consisting of a high degree of heterogeneity in data. The critical part of the association rule mining is the frequent set calculation and it occupies approximately 75%of the processing time.

Processing time holds CPU time (the time required for candidate set generation) and I/O time (the time required for accessing the database for information retrieval). In order to make the frequent set disco very efficiently both of these parts have to be considered.

Each and every technology that is invented so far for this problem has its own advantages and limitations depending upon the nature of the problem domain. The size of the maximal frequent set of the problem domain, heterogeneity present in the data and support threshold selected for the calculation are the factors greatly effecting the actual processing of the frequent item set discovery methods.

It is necessary for determining which algorithmic procedure is best – suited for the nature of the application domain. This project aims at analyzing various programming techniques for frequent set mining and association rule mining on application database and to trace the efficient approach best suited for the nature of the application, in terms of candidate set generation and database access.

கருத்துச் சுருக்கம்

சேர்க்கை விதிக் கொணர்வு வணிக மற்றும் ஆய்வுக்களத்துள் இன்றியமையாததொரு பங்களிப்பை ஆற்றி வருகிறது. பல்வேறுபட்ட பொருட்சேர்க்கைகளின் ஆதாரத்தையும், உறுதிப்பாட்டையும் பன்முகத்தன்மை கொண்டதும் பரந்ததுமான தகவல் களஞ்சியத்தினூடே தேடிக்காணல் அவசியமாகிறது. சேர்க்கை விதிக்கொணர்வின் சிக்கலான பகுதி நிகழ்ம கணக் கணக்கீடாகும். அது தோராயமாக 75 விழுக்காடு பணிநேரத்தை எடுத்தாள்கிறது.

பணி நேரமாவது இங்கு மையப்பொறி (CPU) எடுத்துக்கொள்ளும் நேரத்தையும் தகவல் களஞ்சியப் பயன்பாடு நேரத்தையும் உள்ளடக்கியதாகும். நிகழ்ம கணக் கணக்கீட்டினை நற்பயன்பாடமைய ஆக்குவதற்கு, இவ்விரு பகுதிகளும் கருத்தில் கொள்ளப்பட வேண்டியனவாகும்.

ஆய்வுக்கு இதுவரை அறிமுகபடுத்தப்பட்ட இந்த ஒவ்வொரு தொழில்நுட்பமும் தனக்கே உரித்தான சாதக பாதகங்களை, எடுத்துக்கொண்ட ஆய்வுக் களத்தின் இயல்புகளுக்கேற்பக் கொண்டுள்ளன. மீப்பெரு நிகழ்ம கணத்தின் அளவு, தகவல் தொகுப்பின் பன்முகத்தன்மை, கணக்கீட்டிற்கு எடுத்துக்கொள்ளப்படும். ஆதார எல்லை ஆகியன நிகழ்ம கணக் கணக்கீட்டுப்பணி நேரத்தை மிகையாகப் பாதிக்கும் காரணிகள் ஆகும்.

பயனீட்டு களத்திற்கு மிகப்பொருத்தமான திட்ட வரைவை உறுதி செய்வது இன்றியமையாததாகிறது. இந்த ஆய்வு நிகழ்ம கணக் கணக்கீட்டிற்கான பல்வேறு தொழில்நுட்ப முறைபாடுகளையும், எடுத்துக்கொண்ட பயனீட்டு தகவல் களஞ்சியத்தின் இயல்போடு உட்படுத்தி பகுத்தாய்கிறது. இந்த ஆய்வு பயன்பாட்டில் மிக்க அணுகுமுறையை மையப்பொறி நேரத்தையும், தகவல் களஞ்சியப் பயன்பாடு நேரத்தையும் கொண்டு தெரிந்தெடுக்கிறது.

ACKNOWLEDGEMENT

I sincerely thank our principal **Dr.K.K.Padmanabhan Ph.D.,** Kumaraguru College of Technology, Coimbatore, for the support and innumerable facilities provided by him for my project work.

I express my sincere thanks to our Head of the department **Dr.S.Thangasamy Ph.D.,** Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore, for his valuable suggestions and constant encouragement throughout my project.

I am extremely grateful to my guide **Mr.K.R.Baskaran M.S.,** Assistant Professor, Department of Information Technology, Kumaraguru College of Technology, Coimbatore, for his valuable guidance.

I express my sincere thanks to our course coordinator **Mrs.L.S.Jayashree M.E.,** Senior Lecturer, Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore, for her motivation.

I record my sincere thanks to our project coordinator **Mr.R.Dinesh M.S.,** Assistant Professor, Department of Computer Science and Engineering, Kumaraguru College Of Technology , Coimbatore, for his constant encouragement and motivation.

I feel pleasure to acknowledge all the faculty members of the Department of Computer Science and Engineering for their useful hints, novel ideas and encouragements through out this project phase.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1THE CURRENT STATUS OF THE PROBLEM TAKEN UP

Data mining is the knowledge discovery in databases, as it is also known, is the non – trial extraction of implicit unknown and potentially useful information from the data. Association rule mining plays a vital role in the commercial and research environment. The support and confidence of various itemsets of products have to be mined from a huge database consisting of a high degree of heterogeneity in data.

The critical part of the association rule mining is the frequent set calculation mining and it occupies approximately 75 % of the processing time.

Each and every technology invented so far for solving the problem has its own advantages and limitations depending upon the nature of the problem. Heterogeneity present in the data and support threshold level are the vital factors having a great impact on the application domain.

When frequent items sets to be mined from the database are in a small number Apriori bottom – up processing technology holds more efficiency compared to other mining techniques. The Pincer search two-way approach plays efficient role when the database contains a huge number of frequent itemsets for the given minimum support level.

Frequent pattern tree approach deals with the voluminous database containing high degree of heterogeneity among its item sets.

The discovery of interesting association relationships among huge volume of data is continuously collected and stored. Many industrial organizations are becoming interested in mining association rules from their databases. The discovery of interesting association relationships among huge volume of business transaction records can aid in many business decision making process such as catalog design cross marketing and loss-leader analysis.

From the business point of view the technology employed for collecting the required information must be efficient in terms of execution time and resource utilization. The nature of the application domain itself has a significant impact on the performance of the technology employed.

It is necessary for determining the technical procedure best suited for the given application domain in terms of efficiency factors. The maintenance of the database application and information retrieval from the dynamically updatable databases will be simplified by means of employing the appropriate frequent set mining technology.

1.2.RELEVANCE AND IMPORTANCE OF THE PROBLEM TAKEN UP

The discovery of association relationships plays a major role in the marketing strategies by gaining insight into which items are frequently purchased together by customers. For example, if customers are buying milk how likely they would also buy bread on the same transaction to the supermarket, such information can lead to increase sales by helping retailers do selective marketing and plan their inventory. For example, placing milk and bread within close proximity may further encourage the sale of these items together within single visits to the store.

Frequent set mining searches for interesting relationship among items in a given dataset. From commercial point of view, it is necessary to make this solution in an effective manner with respect to the resource utilization and processing time.

Database collection of an industrial organization has an impact on the nature of the technology with is heterorganic nature of data and memory space occupied. Database, itself in an industrial environment is dynamic and therefore to have corresponding reflection in the resulting sets, the nature of the technology employed should be flexible and efficient.

Performance evolution is justified based on the processing time, and database access time among the technologies employed on a particular application database. Hence the application domain itself deals with the nature of the technology.

It is necessary for the technology to have dynamic reflection over the database collection and efficient utilization of system resources. In business point of view, these factors play major role and for various levels of support threshold the performance can be affected.

To provide efficient service for the customers of a business sector, it is necessary to update the stack with respect to the user's point of purchasing. To contact such a voluminous and heterogeneous database, the technology employed should fulfill the requirements under expected level of service.

Through the performance evaluation factors like database access time, the best-suited technology can be adopted for the nature of the application domain.

CHAPTER 2

LITERATURE SURVEY

2.1.Data mining Concepts

Data mining is the knowledge discovery in data bases, as it is also known, is the non – trivial extraction of implicit previously unknown and potentially useful information from the data, this encompasses, finding dependency networks, analyzing changes, and detecting anomalies.

Data mining is the search for relationships and global patterns that exist in large databases but are hidden among vast amounts data such as the relationship between patient data and their medical diagnosis. This relationship represents valuable knowledge about the database, and the objects in the database are faithful mirrors of the real world registered by the database.

Data mining refers to using a variety of techniques to identify nuggets of information or decision support, prediction, forecasting and estimation. The data is often voluminous but it has low value and no direct use can be made of it. It is the hidden information in the data that is useful.

Discovering relations that connect variables in a database is the subject of data mining, the data mining system self – learns from the previous history of the investigated system, formulating and testing hypothesis about rules which systems obey. When Concise and valuable knowledge about the system of interest is discovered, it

can and should be interpreted into some decision support system, which helps the manager to make wise and informed business decision.

Data mining is the process of discovering meaningful, new correlation patterns and trends by shifting through large amount of data stored in repositories, using pattern recognition techniques as well as statistical and mathematical techniques.

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

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

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

2.1.1.The Foundations of Data Mining

Data mining techniques are the result of a long process of research and product development. This evolution began when business data was first stored on computers, continued with improvements in data access, and more recently, generated technologies that allow users to navigate through their data in real time. Data mining takes this evolutionary process beyond retrospective data access and navigation to prospective and proactive information delivery.

Data mining is ready for application in the business community because it is supported by three technologies that are now sufficiently mature:

- Massive data collection
- Powerful multiprocessor computers
- Data mining technologies

In the evolution from business data to business information, each new step has built upon the previous one. For example, dynamic data access is critical for drill-through in data navigation applications, and the ability to store large databases is critical to data mining.

The core components of data mining technology have been under development for decades, in research areas such as statistics, artificial intelligence, and machine learning. Today, the maturity of these techniques, coupled with high-performance relational database engines

and broad data integration efforts, make these technologies practical for current data warehouse environments.

2.1.2.The Scope of Data Mining

Data mining derives its name from the similarities between searching for valuable business information in a large database — for example, finding linked products in gigabytes of store scanner data — and mining a mountain for a vein of valuable ore. Both processes require either sifting through an immense amount of material, or intelligently probing it to find exactly where the value resides. Given databases of sufficient size and quality, data mining technology can generate new business opportunities by providing these capabilities:

- **Automated prediction of trends and behaviors**. Data mining automates the process of finding predictive information in large databases. Questions that traditionally required extensive hands-on analysis can now be answered directly from the data — quickly. A typical example of a predictive problem is targeted marketing. Data mining uses data on past promotional mailings to identify the targets most likely to maximize return on investment in future mailings. Other predictive problems include forecasting bankruptcy and other forms of default, and identifying segments of a population likely to respond similarly to given events.

- **Automated discovery of previously unknown patterns**. Data mining tools sweep through databases and identify previously hidden patterns in one step. An example of pattern discovery is the analysis of retail sales data to identify seemingly unrelated products that are often purchased together. Other pattern discovery problems include detecting fraudulent credit card

transactions and identifying anomalous data that could represent data entry keying errors.

Data mining techniques can yield the benefits of automation on existing software and hardware platforms, and can be implemented on new systems, as existing platforms are upgraded and new products developed. When data mining tools are implemented on high performance parallel processing systems, they can analyze massive databases in minutes. Faster processing means that users can automatically experiment with more models to understand complex data. High speed makes it practical for users to analyze huge quantities of data. Larger databases, in turn, yield improved predictions.

Databases can be larger in both depth and breadth:

- **More columns**. Analysts must often limit the number of variables they examine when doing hands-on analysis due to time constraints. Yet variables that are discarded because they seem unimportant may carry information about unknown patterns. High performance data mining allows users to explore the full depth of a database, without reselecting a subset of variables.

- **More rows**. Larger samples yield lower estimation errors and variance, and allow users to make inferences about small but important segments of a population.

The most commonly used techniques in data mining are:

- **Artificial neural networks**: Non-linear predictive models that learn through training and resemble biological neural networks in structure.

- **Decision trees**: Tree-shaped structures that represent sets of decisions. These decisions generate rules for the classification of a dataset. Specific decision tree methods include Classification and Regression Trees (CART) and Chi Square Automatic Interaction Detection (CHAID) .

- **Genetic technologies**: Optimization techniques that use processes such as genetic combination, mutation, and natural selection in a design based on the concepts of evolution.

- **Nearest neighbor method**: A technique that classifies each record in a dataset based on a combination of the classes of the k record(s) most similar to it in a historical dataset. Sometimes called the k-nearest neighbor technique.

- **Rule induction**: The extraction of useful if-then rules from data based on statistical significance.

Many of these technologies have been in use for more than a decade in specialized analysis tools that work with relatively small volumes of data. These capabilities are now evolving to integrate directly with industry-standard data warehouse and OLAP platforms. The technique that is used to perform these feats in data mining is called modeling. Modeling is simply the act of building a model in one situation where the answer is known and then applied to another situation that is unknown.

This act of model building is thus something that people have been doing for a long time, certainly before the advent of computers or data mining technology. What happens on computers, however, is not much different than the way people build models. Computers are loaded up with lots of information about a variety of situations where an answer is known and then the data mining software

on the computer must run through that data and distill the characteristics of the data that should go into the model. Once the model is built it can then be used in similar situations where the answer is unknown.

This model could then be applied to the prospect data to try to tell something about the proprietary information that this telecommunications company does not currently have access to. With this model in hand new customers can be selectively targeted.

2.1.3.An Architecture for Data Mining

To best apply advanced data mining techniques, they must be fully integrated with a data warehouse as well as flexible interactive business analysis tools. Many data mining tools currently operate outside of the warehouse, requiring extra steps for extracting, importing, and analyzing the data. Furthermore, when new insights require operational implementation, integration with the warehouse simplifies the application of results from data mining. The resulting analytic data warehouse can be applied to improve business processes throughout the organization, in areas such as promotional campaign management, fraud detection, new product rollout, and so on. Figure 2.1 illustrates architecture for advanced analysis in a large data warehouse.

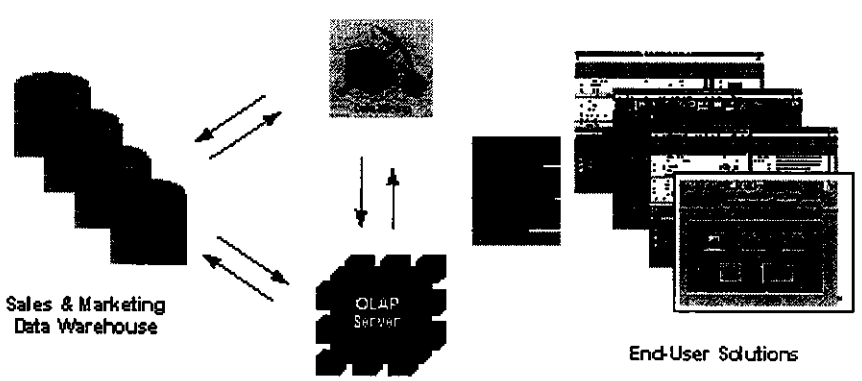

Figure 2.1. - Integrated Data Mining Architecture

The ideal starting point is a data warehouse containing a combination of internal data tracking all customer contact coupled with external market data about competitor activity. Background information on potential customers also provides an excellent basis for prospecting. This warehouse can be implemented in a variety of relational database systems: Sybase, Oracle, Redbrick, and so on, and should be optimized for flexible and fast data access.

2.2.Association Rule Mining Concepts

Among the areas of data mining, the problem of deriving associations from data has received a great deal of attention. Agarwal formulated the problem in 1993 and is often referred to as the market basket problem. In this problem, a set of items and a large collection of transactions are given, which are subsets (baskets) of these items. The task is to find relationships between the presences of various items within these baskets.

There are numerous applications of data mining, which fit into this framework. The classic example from which the problem gets its name is the supermarket. In this contest, the problem is to analyze customers' buying habits by financing associations between the different items that customers place in their shopping baskets. The discovery of such association rules can help the retailer develop marketing strategies, by gaining insight into matters which items are most frequently purchased by customers. It also helps in inventors' management sale promotion strategies etc.

It's widely accepted (except of some recent development) that the discovery of association rules is solely dependent on the discovery of frequent sets. Thus a majority of the technology are

concerned with efficiently determining the set of frequent item sets in terms of the number of items. It is not possible to count the frequencies of these sets by reading the database in just in one pass. The multiple passes for generating all the frequent itemsets is unavoidable. Thus, different technologies for the discovery of association rules exist.

In short, association rule induction is a powerful method for so-called *market basket analysis*, which aims at finding regularities in the shopping behavior of customers of supermarkets, mail-order companies and the like. With the induction of association rules one tries to find sets of products that are frequently bought together, so that from the presence of certain products in a shopping cart one can infer (with a high probability) that certain other products are present. Such information, expressed in the form of rules, can often be used to increase the number of items sold, for instance, by appropriately arranging the products in the shelves of a supermarket (they may, for example, be placed adjacent to each other in order to invite even more customers to buy them together) or by directly suggesting items to a customer, which may be of interest for him/her.

An *association rule* is a rule like "If a customer buys jam and bread, he often buys cheese, too." It expresses an association between (sets of) *items*, which may be products of a supermarket or a mail-order company, special equipment options of a car, optional services offered by telecommunication companies etc. An association rule states that if a customer is selected at random and found out that he selected certain items (bought certain products, chose certain options etc.). There will be a confidence, which is quantified by a percentage, that he also selected certain other items (bought certain other products, chose certain other options etc.).

The standard measures to assess association rules are the *support* and the *confidence* of a rule, both of which are computed from the *support* of certain item sets. The main problem of association rule induction is that there are so many possible rules. For example, for the product range of a supermarket, which may consist of several thousand different products, there are billions of possible association rules. It is obvious that inspecting each one in turn cannot process such a vast amount of rules. Therefore efficient technologies are needed that restrict the search space and check only a subset of all rules, but, if possible, without missing important rules.

2.2.1.Support of an Item Set

Let T be the set of all transactions under consideration, e.g., let T be the set of all baskets of products bought by the customers of a supermarket - on a given day. The support of an item+ set S is the percentage of those transactions in T which contain S. In the supermarket example this is the number of baskets that contain a given set S of products, for example S = { bread, jam, cheese }. If U is the set of all transactions that contain all items in S, then

support(S) = (|U| / |T|) *100%,

Where |U| and |T| are the number of elements in U and T, respectively. For example, if a customer buys the set X = { milk, bread, apples, jam, sausages, cheese, onions, potatoes } then S is obviously a subset of X, hence S is in U. If there are 318 customers and 242 of them buy such a set U or a similar one that contains S, then support(S) = 76.1%.

2.2.2.Confidence of an Association Rule

This is the measure used by the inventors of the Apriori technology, to evaluate association rules. The confidence of a rule R = "A and B -> C" is the support of the set of all items that appear in the rule divided by the support of the antecedent of the rule, i.e.

Confidence(R) = (support({A, B, C}) / support({A, B})) *100%.

For a given transaction database t, an association rule is an expression of the form x =>y, where x and y are subsets of A and x => y holds with **confidence** C, if C% of transactions in D that support x also support y. the rule x=> y has support y. The rule x => y has x also **support** s in the transactions in t if s% of transactions in T support x U Y.

For example, considering the set of six transactions in a database:

T1={A, C, T, G}
T2={C, D, G}
T3={A, C, T, G}
T4={A, C, D, G}
T5={A, C, D, T, G}
T6={C, D, T}
Here A={A, C, D, G, T}
T= {T1, T2 ,T3, T4, T5, T6}

Here, item D is supported by 4 out of 6 transactions in T. Thus the Support of D is 66.67%. Considering the association rule A => C, the confidence of this rule is 100%, because all the transactions that

support A also support C. A => C hold the support 66.67% because only 4 out of 6 transactions support both A and C.

2.3.3.Minimum Support

Minimum support is the threshold value selected from the user. It is used to mark the frequency in numeric manner among the database transaction.

For example, if there are 20000 transactions and the minimum support value is 20%, then for satisfying the minimum support threshold level an item set must present in atleast 20% of the transactions, i.e., in 4000 transactions.

Since minimum support is a user defined value, it highly effects the results and performance of the mining methodologies employed. It is suggested that different levels of the frequent itemset discovery to maintain different values of minimum support. That is, if a minimum support level is defined as 40% for discovering frequent 1-itemsets, then it is suggested that for mining 2-itemsets some lower value can be given as a minimum support. Clearly, the presence of 2-itemsets will be less than 1-itemsets. Hence this suggestion is made for new methodologies. In this project, equal minimum support level is maintained for all categories of itemsets in order to make the performance study under standard operating conditions.

2.2.4.Frequent set

Let 't' be the transaction database and 'm' be the user specified minimum support. An item set X is said to be a frequent in T with respect to if.

$S(X) \geq m.$

Maximal Frequent Set :

A frequent set a maximal frequent set if its is frequent set and no superset of this is a frequent set.

Considering a maximal frequent set { 1, 2, 3, 4, 5 }, then none of its supersets must be a frequent set with respect to the transaction collection. If there is a frequent set { 1, 2, 3, 4, 5, 6 }exists in this domain, then the first one will not be the maximal frequent set.

It is possible for the presence of more than one maximal frequent set in the application domain with respect to the minimum support value.

Border Set:

An item set is a border set if it is not a frequent set, but all its proper supports are frequent sets.

2.2.5.Closure Properties of Frequent Itemsets

Downward Closure Properly

Any subset of a frequent set is a frequent set.

For example, the item set {1, 2, 3, 4}, that is proved to be the frequent with respect to database T is taken in to consideration. Hence,{1, 2, 3 } which is a subset of this frequent set is also an frequent item set. It need not be examined to verify whether it satisfies the minimum support threshold.

Every subset of a frequent itemset is also frequent. Technologies make use of this property in the following way - the count of an itemset won't be considered, if all its subsets are not frequent. Hence the counts of some short itemsets in one pass of the database

are traced out first. Then longer and longer itemsets in subsequent passes are taken into consideration. When a long item set is considered, all its subsets must be frequent for making that set a frequent one. This can be done because the counts of all those subsets have been collected in previous passes.

Upward Closure property

Any superset of an infrequent set is an infrequent set.

For example, the item set {1, 2, 3}, that is proved to be an infrequent itemset with respect to database T is taken in to consideration. Hence, {1, 2, 3, 4} which is a superset of this on frequent set is also an infrequent item set. It need not be examined to verify whether it satisfies the minimum support threshold.

2.2.6.Association rule

Association rule mining finds interesting associations and/or correlation relationships among large set of data items. Association rules show attribute value conditions that occur frequently together in a given dataset. A typical and widely used example of association rule mining is Market Basket Analysis.

For example, data are collected using bar-code scanners in supermarkets. Such 'market basket' databases consist of a large number of transaction records. Each record lists all items bought by a customer on a single purchase transaction. Managers would be interested to know if certain groups of items are consistently purchased together. They could use this data for adjusting store inventory (placing items optimally with respect to each other), for cross-selling, for

promotions, for catalog design and to identify customer segments based on buying patterns.

Association rules provide information of this type in the form of "if-then" statements. These rules are computed from the data and, unlike the if-then rules of logic, association rules are probabilistic in nature.

In addition to the antecedent (the "if" part) and the consequent (the "then" part), 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.

For example, if a supermarket database has 100,000 point-of-sale transactions, out of which 2,000 include both items A and B and 800 of these include item C, the association rule "If A and B are purchased then C is purchased on the same trip" has a support of 800 transactions (alternatively 0.8% = 800/100,000) and a confidence of 40% (=800/2,000). One way to think of support is that it is the probability that a randomly selected transaction from the database will contain all

items in the antecedent and the consequent, whereas the confidence is the conditional probability that a randomly selected transaction will include all the items in the consequent given that the transaction includes all the items in the antecedent.

For a given transaction database T, an association rule is an expression of the form X -> Y where X and Y are subsets of A and X∪Y holds with confidence T if T% of transaction in D that support X also support Y.

The intuitive meaning of such a rule is that a transaction of the database which contains X tends to contain Y, with respect to the given a set of transactions T. The problem of mining association rules is to discover all rules that have support and confidence greater than or equal to the user – specified minimum support and minimum confidence, respectively.

There is a theory in data mining that involves an apocryphal discovery that the sales of apple and orange are correlated. Association (and sequencing) tools discover rules like: When people buy orange they also buy apple half of the time.

Each rule has a left-hand side (buy orange) and a right–hand side (buy apple). The left hand side is also called the antecedent and the right hand side is also called the consequent. In general both the left hand side and the right hand side can contain multiple items.

Considering a database consisting of 500,000 transactions 20,000 transactions of these contain orange 30,000 transaction contain

apple, 10,000 transactions contain both orange and apple, the following values can be obtained.

Support (or prevalence) measures how often apple and orange occur together as a percentage of the total transactions 2 % in this case (10, 000/ 500,000) confidence (or predictability) measures how much a particular item is dependent on another. Since 20,000 transactions contain orange and 10,000 also contain apple, when people buy orange they also apple half of the time.

The inverse rule has a confidence of 33.33 % (completed as 10,000/30,000) and would be stated as: When people buy apple they also buy orange half of the time.

Support does not depend upon the direction (or implication) of the rule. It is only dependent on the set of items in the rule. In the absence of knowledge about what else bought, the following assertions are made: People buy orange 4 % of the time. People buy apple 6 % of the time.

These numbers 4 % and 6 % are called the expected confidence of buying orange or apple, respectively.

2.2.7.Methods to Discover Association Rules

The discovery of association rules is the most will study problem in data mining. One of the key features of all association rule mining technologies is that each of these methods assumes that the underlying database size is enormous and they require multiple passes over the database. For disk resident databases this required reading the database completely for each pass resulting in a large number of disks

read. In these technologies, the effort spent in performing just I /O operations may be considerable for large database. For example, a 1 GB database will require 120, 000 block reads for a single pass (for a block size of 8 KB). If the technology requires 10 passes, this results in a1, 250,000 blocks read. Assuming an average read time of 12 ms per page, the time spent in just performing the I /O is 12500 x 12 ms = 4 hours. Apart from poor response times, this problem places a huge burden on the I /O system. Usually, the data is collected by an online transaction processing system running hundreds or thousands per second. Running this technology under such workloads will adversely affect the transaction response time and may even disrupt the daily database server. Over a network such as LAN, it will create network congestion problems and lead to poor resource utilization. Thus the desirable features of any efficient technologies are to reduce the I/O operations and at the same time be efficient in computing.

The problem of missing association rules can be decomposed into two sub problems. Find all item sets whose support is greater than the user specified minimum support. Then generate the desired association rules. Much research has been focused on the first sub problem, as the database is accessed in this part to the computation, the several technologies have been proposed. To reduce the combinatorial search space, all technologies exploit the two closure properties explained above.

CHAPTER 3

LINE OF ATTACK

3.1.Collection of data

To collect an application database consisting of

i) Voluminous and heterogeneous data

ii) Exhibiting real-world nature of the application domain.

A database is chosen from a Departmental stores application domain for this project in such a way that a high degree of heterogeneity is present in the data (76 products) and vast volume of data collection (4.98 MB).

The database is taken based on the transactions on a Departmental stores. Hence it reflects the real world nature also.

Due to its heterogeneity, the complexity of the methodology can be tested. Since the database collection is a voluminous one, it is necessary for the methodology to make minimum number of scans on it for achieving best performance.

3.2.Frequent set mining Technologies

There are various technical approaches employed for frequent item set discovery. Each and every technology has its advantages and limitations. They are affected from the size of the maximal frequent set, candidate generation and number of scans required on the database transactions. The complex methodologies should work well for complex domains, theoretically. The way in which the closure properties are incorporated to reduce the scanning process also plays a major role with respect to the application domain taken. Hence it is necessary to choose the methodologies in such a way that they exhibit their unique way of approach while dealing with the problem domain.

In order to represent different characteristics of the technical approaches, the following methodologies are considered and implemented using Java on existing application database.

1. Apriori bottom-up searching methodology
2. Pincer-search methodology
3. Partitioning methodology
4. Tree construction methodology

These methodologies are tested with the database application for producing frequent sets with respect to the user defined minimum support level.

3.3.Comparative study on Frequent set mining Techniques

A Comparative study is made on the implemented algorithmic approaches in terms of candidate generation and database access parts of the algorithm.

Conclusion is made for tracing out which algorithm plays a high performance and efficiency for the nature of the application domain chosen based on the results gained on comparative study.

CHAPTER 4

DETAILS OF METHODOLOGY EMPLOYED

4.1.Apriori Methodology

Apriori technology is based on downward closure property. Apriori is an influential algorithm for mining frequent itemsets for Boolean association rules. The name of the Apriori is based on the fact that the algorithm uses prior knowledge of the frequent itemset properties. Apriori employs an iterative procedure that performs level-wise search, where k-itemsets are used to explore (k+1) itemsets. First the set of frequent 1-itemsets is found. This set is denoted L1. L1 is used to find L2, the set of frequent 2-itemsets, which is used to find L3, and so on, until no more frequent k-itemsets can be found. The finding of each L_k requires one full scan of the database.

In short, Apriori algorithm employs the following procedures.

(i) Candidate generation (from the set of frequent itemsets found in previous pass).

(ii) Pruning candidate sets in order to reduce data base access time.

(iii) Calculate support for existing candidate sets for determining frequent item sets for next pass.

These procedures are repeated until the maximum frequent sets are to be discovered.

To improve the efficiency of level-wise generation of frequent itemsets, an important property called the downward closure

property, is used to reduce the search space. In order to use this property, all non-empty sets of a frequent itemset must be also frequent. By definition, if an item set 'I' does not satisfy the minimum support threshold, then 'I' is not frequent. If an itemset A is added to the itemset I, the resulting itemset cannot occur more frequently than I. Therefore I $\cup$ A is not frequent.

This property belongs to a special category of properties called anti-monotone in the sense that if a set cannot pass a test, all of its supersets will fail the same test as well. It is called so because the property is monotonic in the context of failing a test.

The primary algorithms employed on this algorithm are Join and Prune methods.

The Join method:

To find L_k from L_{k-1}, a set of candidate k-itemsets is generated by joining L_{k-1} with itself. This set of candidates is denoted C_k. Let l1 and l2 be itemsets in L_{k-1}. The notation li[j] refers to jth item in li. By convention Apriori assumes that items within a transaction or itemset are stored in lexicographic order. The join is performed where members of L_{k-1} are joinable if their first (k-2) itemsets are in common. That is members of l1 and l2 of L_{k-1} are joined if (l1[1]=l2[1] $\cap$ l1[2]=l2[2] $\cap$ l1[3]=l2[3] $\cap$ ... l1[k-1]=l2[k-1]. The condition l1[k-1] < l2[k-1] simply ensures that no duplicates are generated. The resulting itemset formed by joining l1 and l2 is l1[1] l1[2] ... l1[k-1] l2[k-1].

The Prune method:

C_k is a superset of L_k that is, its members may or may not be frequent, but all of the frequent k-itemsets are included in C_k. A scan of the database to determine the count of each candidate in C_k would result in the determination of L_k. C_k however, can be huge and, so this could involve heavy computation. To reduce the size of C_k, the downward closure property is used as follows. Any (k-1) itemset that is not frequent cannot be a subset of frequent k-itemset. Hence if any (k-1) subset of a candidate k-itemset is not in L_{k-1}, the candidate cannot be frequent either and so can be removed from C_k.

The Apriori method can be explained with the following procedure calling Join and Prune procedures for generating L_{k-1}sets from L_k itemsets.

Apriori Method

Initialize k = 1, C_1 = all the 1 – itemsets
read the database to count the support of C_1 to determine L_1
L_1= {frequent 1–itemsets);
k = 2 // k presents the pass number //
while $L_{k-1} \neq \phi$ do
begin
C_k = gen_ candidate_ itemsets with the given L_{k-1}
prune (C_k)
For all transactions t $\in$ T do
increment the count of all candidates in (C_k) that are contained in t;
L_k =All candidates in (C_k) with minimum support
k=k+1;
end

Join method :

$C_k = \phi$

for all itemsets $l1 \in L_{k-1}$ do

for all itemsets $l2 \in L_{k-1}$ do

if $l1[1]=l2[1] \cap l1[2]=l2[2] \cap \ldots l1[k-1]=l2[k-1]$

then c = l1 [1], l1[2],...l1[k-1], l2[k-1]

$C_k = C_k \cup \{c\}$

Prune (C_k):

For all $c \in C_k$

For all (k-1) subsets d of c do

If $d \notin L_{k-1}$

Then $C_k = C_k \setminus \{c\}$

Thus Apriori methodology produces downward closure property and reduces the number of candidate sets for each pass. It employs a simple and one-way approach on the application domain.

4.2.Partitioning Methodology

The Partition technology is based on the observations that the frequent sets are normally very few in number compared to the set of all item sets. As a result the set of the transactions are partitioned to smaller segments such that each segment can be accommodated in the main memory, then the set of frequent sets of each of these partitions can be computed. It is assumed that these sets (set of local frequent sets) contain a reasonably small number of itemsets. Hence, the whole database (the global one) is read once, to count the support of the set of all local frequent sets.

The partition technology uses two scans of the database to discover all frequent sets. On one scan, it generates a set of all potentially frequent itemsets by scanning the database once. This set is a superset of all frequent item sets, i.e. it may contain false positives but no false negatives are reported. During the second scan, counters for each of these itemsets are set up their actual support is measured in one scan of the database.

The technology executes in two phases. In the first phase the partition technology logically divided the database into a number of non-overlapping partitions. The partitions are considered one at a time and all frequent itemsets for the partition are generated. Thus if there are n partitions, then in phase I, the local frequent itemsets of same lengths from all n partitions are combined to generate the global candidate itemsets. In Phase II, the actual support for these item sets are generated and the frequent item sets are identified. The technology reads the entire database once during phase I and during Phase II. The partition sizes are chosen such that each partition can be accommodated in the main memory so that the partitions are read only once in each phase.

A partition P of the database refers to any subset of the transaction contained in the database. Any two partitions are non-overlapping. Local support for an itemset is the fraction of the transaction containing that particular itemset in a partition. A local frequent itemset is an itemset whose local support in a partition is atleast the user defined minimum support. A local frequent itemset may or not be frequent in the context of the entire database.

The partition technology is based on the premise the size of the global candidate set is considerably smaller than the set of all

possible itemset. If the data characteristics are uniform across partitions, then large numbers of itemsets generated for individual partitions may be common till it reaches the largest frequent. The number of database passes is equal to the largest size of the frequent itemset. When any one of the frequent itemsets becomes longer the technology has to go through many iterations and as a result the performance decreases.

4.3.Pincer-Search Bi-directional Methodology

A bi-directional search takes advantages of both the bottom up as well as the top down process. The pincer search technology is based on this principle. It attempts to find the frequent itemsets in a bottom up manner but at the same time it maintains a list of maximal frequent itemsets. While making a database pass, it also counts the support of these candidate maximal frequent itemsets to see if any one of these is actually frequent. In that event it can conclude that all subsets of these frequent sets are going to be frequent. Hence, they are not verified for the support count in the next pass. Clearly the pincer search has an advantage over a priori technology when the largest frequent itemset is long.

In this technology, in each pass in addition to counting the supports of the candidate in the bottom up direction, it also counts the supports of the itemsets using a top down approach. These are called the Maximal Frequent candidate set (MFCS). This process helps in pruning the candidate sets very early on in the technology.

Considering pass k during which itemsets of size k are to be classified, if some itemset that is an element of the MFCS, say X of cardinality greater than k is found to be frequent in this pass, then all its subsets must be frequent. Therefore all of its subsets of cardinality k can

be pruned from the set of candidate sets considered in the bottom up direction in the pass. They and their supersets will never be candidates throughout the rest of the execution, potentially improving the performance.

Similarly, when a new infrequent itemset is found in the bottom up direction the technology will use it to update the MFCS. The subsets of the MFCS should not contain this is frequent itemsets.

Pincer- Search Method

K= 1:C: = {{i} $\in$I}:S = ϕ

L= ϕ

MF CS:= {{1,2,...,n}}; MFS:= ϕ;

do until C_k = ϕ and S= ϕ

read database and count supports for C_k and MFCS;

MFS ;= MFS $\cup${frequent itemsets im MFCS};

S_k={infrequent itmsets in C_k};

call MFCS-gen algorithm if S #O;

call MFS- pruning procedure;

generate candidates C_{k+1} from C_k;(similar to a priori's generate & prune)

if any frequent in C_k is C is removed in MFS – pruning prune}

call the recovery procedure to recover candidates to C_{k+1}

k;=k+1;

return MFS

MFCS-gen

for all itemsets s$\in S_k$

ifs is a subset of m

```
                MFCS;=MFCS\{m};
                for all items e ∈ itemset.S
                if m\{e} is not a subset of any itemset in MFCS
                        MFCS:= MFCS {m\{e}};

return MFCS

        Recovery
        for all itemsets l ∈ Ck
                for all itemsets m ∈ MFS
                if the first k-1 items in /are also in m
                /* suppose m.item =Litem */
                for i from j+ 1 to |m|
                Ck+1 = Ck+1 O{{/ Item1,/.Item2,...,m.item3}}
        MFS- Prune
                for all itemsets c in Ck_1
                if c is a subset of any itemset is the current MFS
                Delete c from C;

                MFCS- Prune
                for all itemsets c in Ck-1
                if c is not a subset of any itemset in the current MFCS
        delete c from Ck+1;
```

The MFCS initially contains single elements, the itemset of cardinality n containing all the elements of the database. If some m 1-itemsets are infrequent after the first pass (after reading the database once), the top down search goes down m levels in one pass. Unlike the pure top-down or bottom-up search can go down many levels in one pass.

4.4.Tree Construction Methodology

The above technologies suffer from the following two shortcomings

1. It is costly to handle large numbers of candidate sets. For instance, if there are 10 frequent 1 – itemsets, then approximately 36 candidate 2 item sets are generated and if there is a frequent set of size 100 then roughly 10-candidate sets are generated in this process.
2. It is tedious to repeatedly scan the database and to check a large set of candidates by pattern matching.

A new technology has recently been proposed which avoids the generation of large numbers of candidate sets called the FP Tree Growth technology. The main idea of the technology is to maintain a Frequent Pattern Tree (FP – Tree) of the database. It is an extended prefix tree structure, storing crucial and quantitative information about frequent sets. The tree nodes are frequent items and are arranged in such a way that more frequently occurring ones. The method starts from frequent 1 – itemsets as an initial suffix pattern and examines only its conditional pattern base (a subset of the database), which consists of the set of frequent items co-occurring with the suffix pattern. The technology constructs the condition FP tree and performs mining on this tree.

The technology involves two phases. In phase- I, it constructs the FP tree with respect to a given the construction of this tree requires two passes over the whole database. In phase II, the technology does not use the transaction database any more, but

it uses the FP tree. Interestingly, the FP tree contains all the information about frequent itemsets with respect to the given L set.

Tree Construction

A frequent pattern tree is a tree structure consisting of an item prefix tree and a frequent item header table

Item premix tree consists of a root node labeled null.
Each non-root node consists of three fields
Item name
Support count and
Node link

Frequent item header table consists of two fields
Items name
Head of node link which points to the first node in the FP Tree carrying the item name.

FP tree is dependent on the support threshold. For different values of support threshold, the trees are different. There are some other typical features of the FP Tree. It depends on the ordering of the items. The ordering that is followed in the original paper is the decreasing order of the support counts. However, different ordering may offer different advantages thus the header table is arranged in this order of the frequent items.

One scan of the database T is made and L1, the set of frequent 1-item sets is computed. From this stage onwards the technology ignores the entire infrequent item in the decreasing order of frequency and views any transactions as a list of frequent items in the

decreasing order of frequency. The first element of the list corresponding to any given transaction is the most frequent item among the items supported by t. For a list t head_t is used to denote its first element and body_t to the remaining part of the list (the partition of the list t after removal of head_t).

The scan of the first transaction leads to the construction of the first branch of the tree. The branch is not ordered in the same way as the transaction appears in the database. The items are ordered according to the decreasing order of frequency of the frequent items.

FP-Tree Construction Algorithm

Create a root node of the FP-Tree and label it as NULL
Do for every transaction t
If T is not empty
Insert(t,root)
Link the new nodes to other nodes with similar labels originating from header
End do
Return FP-Tree

Insert(t, any_node)
Do while t is not empty
If any_node has a child node with label head_t
Then increment the link count between any_node and head_t by 1
Else create a new child of any_node with label head_t having link count 1
Call insert (body_t,head_t)
End do

After constructing the frequent pattern tree, the prefix sub-paths for each individual frequent item are traced out from the tree. This sub-tree is termed as conditional pattern base. Once the conditional base is derived from the FP-Tree, one can compute all the frequent patterns associated with it in the conditional pattern. As a result, the procedure is repeated for all items and all the frequent itemsets are being mined from the tree constructed. In such a way, only one scan of the database is required in this approach. This technique highly reduces the database access time and CPU time taken for the process as a whole.

CHAPTER 5

RESULTS OBTAINED

Sample Database Details

Application domain :Departmental Stores

Number of items :76

Number of transactions :5577

Size of the database :4.98 MB

Minimum Support :20

(Items are numbered between 1 and 76)

Resources Employed

Hard disk capacity : 10 GB

Random Access Memory : 512 MB

OUTPUT For Apriori Technology:

Frequent Itemsets for Database Application Under Support Thresold

{4} {8} {5} {67} {37} {32}{17}{51} {35} {34} {62} {21} {36} {75} {20} {71} {64} {7} {6} {59} {16} {57} {54} {65} {22}{28} {73} {75,73}{71,59} {71,22} {71,28} {64,28}{7,73} {6,59} {6,22} {6,28}{59,22}{59,28} {16,22}{16,28} {16,73}{57,54}{57,65}{57,22} {57,28}{57,73} {54,65} {54,22} {54,28} {22,28} {54,73}{65,22} {65,28} {65,73} {22,73} {28,73} {67,73}{21,22} {36,28} {75,65}{71,6}{71,59,22} {71,59,28} {71,6,59} {75,65,73}{57,54,65}{57,54,73} {57,65,73}{54,65,22}{54,65,73} {54,22,73}{65,22,73} {65,28,73} {54,28,73}{65,22,28} {54,65,28}{22,28,73} {54,65,28,73} {57,54,65,73}{54,65,22,73} {65,22,28,73}

OUTPUT For Partition Technology:

Frequent Itemsets for Database Application Under Support Thresold

{4} {37} {32}{17}{51} {35} {34} {62} {8} {5} {67} {67,73} {21} {21,22}
{36} {36,28} {75} {75,65} {75,65,73} {75,73} {20} {71} {71,6}
{71,6,59} {71,59}{71,59,22} {71,59,28} {71,22} {71,28} {64} {64,28}
{7} {7,73} {6} {6,59} {6,22} {6,28} {59} {59,22}{59,28} {16} {16,22}
{16,28} {16,73} {57} {57,54} {57,54,65} {57,54,65,73}
{57,54,73}{57,65}{57,65,73} {57,22} {57,28} {57,73} {54} {54,65}
{54,65,22} {54,65,22,73}
{54,65,28} {54,65,28,73} {54,65,73} {54,22} {54,22,73} {54,28} {54,28,73}
{54,73} {65} {65,22} {65,22,28} {65,22,28,73}{65,22,73} {65,28}
{65,28,73}
{65,73} {22}{22,28} {22,28,73} {22,73} {28} {28,73}{73}

OUTPUT For Pincer-Search Technology:

Frequent Itemsets for Database Application Under Support Thresold

{21}{65,28,73}{57} {57,54} {7} {7,73} {6} {6,59}{59,28} {16} {16,22}{71,6,59} {71,59}{71,59,22} {71,59,28} {71,22} {71,28} {64} {64,28} {6,22} {6,28} {59} {59,22} {4} {37} {32}{17}{51} {35} {34} {62} {8} {5} {67} {67,73} {21,22} {36} {36,28} {75} {75,65} {75,65,73} {75,73} {20} {71} {71,6} {16,28} {16,73} {57,54,65} {57,54,65,73} {57,54,73{54,65,22} {54,65,22,73}{54,65,28} {54,65,28,73} {54,65,73} {54,22} {54,22,73} {54,28} {54,28,73} {54,73} {65} {65,22} {65,22,28} {65,22,28,73}{65,22,73} {65,28} {65,73} {22}{22,28} {22,28,73} {22,73} {28} {28,73}{73} }{57,65}{57,65,73} {57,22} {57,28} {57,73} {54} {54,65}

OUTPUT For Tree Construction Technology:

Frequent Itemsets for Database Application Under Support Thresold

{57,54,73}{57,65}{71,6,59} {71,59}{71,59,22} {71,59,28} {71,22} {71,28} {64} {64,28} {7} {7,73} {6} {6,59} {54,65,28,73} {54,65,73} {54,22} {54,22,73} {6,22} {6,28} {59} {65,73} {22}{22,28} {22,28,73} {22,73} {28} {28,73}{73} {16,28} {16,73} {57} {57,54} {57,54,65} {34} {62} {8} {5} {57,54,65,73} {57,65,73} {57,22} {57,28} {57,73} {54} {54,65} {54,65,22} {54,65,22,73}
{54,65,28} {54,28} {54,28,73} {59,22}{59,28} {16} {16,22}{4} {37} {32}{17}{51} {35} {67} {67,73} {21} {21,22} {36} {36,28} {75} {75,65} {75,65,73} {75,73} {20} {71} {71,6} {54,73} {65} {65,22} {65,22,28} {65,22,28,73}{65,22,73} {65,28} {65,28,73}

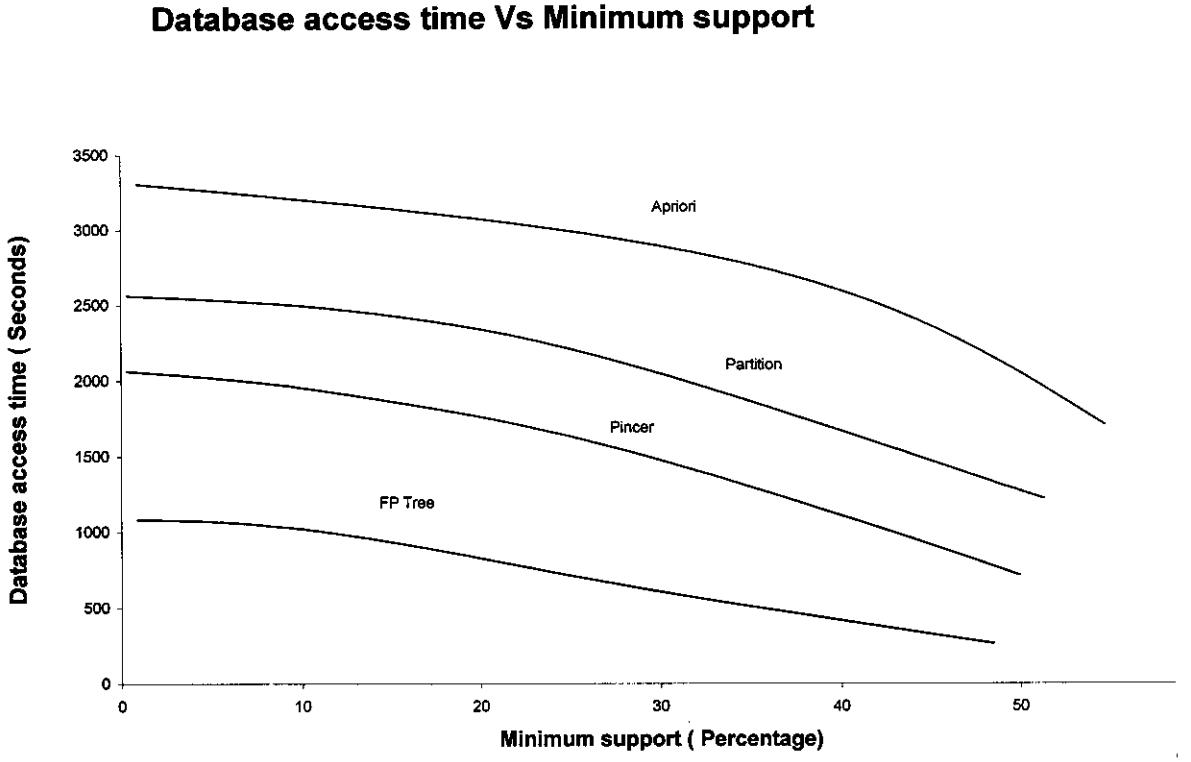

Fig 5.1. Performance evaluation based on database access time

CHAPTER 6

CONCLUSIONS AND FUTURE OUTLOOK

The comparative study is made on the implemented frequent set mining techniques by raising the minimum support level up to 50% and the time required for making database access and time taken for CPU time are measured separately. The obtained results are produced in graphical manner and from the results, it has been concluded that tree construction technology plays the most efficient role with respect to the nature of the application domain taken.

Since Tree construction technology contacts database only once, it is the most efficient technology for the database application taken which is having high degree of heterogeneity and voluminous in nature. Since database access time occupies approximately 75% of the total execution time from the results obtained, it gives the best performance with the technology that contacts database in minimum attempts. Even though the Tree construction method is built with high complexity in its nature, it provides better performance compared to all other simple methods considered for this problem in this project. Here the complexity of the technology is justified with the degree of heterogeneity and vast volume of the database application taken.

The two primary factors CPU time and database access time depend upon the number of pass required for scanning the database. Since Apriori employs a simple approach, it takes more passes and as a result it gives the lowest performance in all the approaches taken here. Partition methodology provides a way of divide-and-conquer method and hence it is able to provide better performance

in terms of both CPU time and database access time than simple Apriori approach.

Because of its two-way approach, Pincer methodology easily reduces the number of scans made on the database. Hence it plays more efficient compared to the Apriori and Partition methodologies.

This project projects discovering best-suited technical procedure for the nature of the application domain. It can be enhanced by means of making a research on set of algorithmic procedures that concentrates on the quantity of the items participating in an item set also.

This project considers two primary factors (database access time and CPU time) for measuring performance of a technology according to the nature of the database. This research can be enhanced by means of considering factors like cache misses and memory usage occurring during the process of candidate generation, which plays a significant role with respect to the system atmosphere chosen.

REFERENCES

1. Ada Wai-Chee Fu and Yin-Ling Cheung (Sep 2004) "Mining Frequent Itemsets without support Threshold : With and without item constructions" IEEE Transactions On Knowledge And Data Engineering, vol.16

2. Agarwal R., Mannila H., Srikant R., Toivonen H. and Verkamo A.I. (1995) "Fast discovery of association rules" AAAI/MIT Press

3. Arun K Pujari,(2004) "Data Mining Techniques", Universities Press, 6th Edition

4. Dao-I Lin and Zvi M.Kedem (May / June 2002) "Pincer-Search: An efficient Algorithm for discovering the Maximum Frequent Set", IEEE Transaction On Knowledge and Data Engineering

5. Dolf Zantinge and Pietri Adrians (2003) "Data Mining", Pearson Education.

6. Wee-knenge and Yew-Kwong woon (July 2004)"A support ordered trie for fast frequent item set discovery", IEEE Transactions On Knowledge and Data Engineering

(ISO 9001 : 2000 Certified Institution) (NBA Accredited Institution)

Dr. M.G.R. Nagar, HOSUR - 635 109, Tamil Nadu.

2nd NATIONAL CONFERENCE FOR RESEARCH SCHOLARS ON APPLICATIONS OF EMERGING TECHNOLOGIES

28 MARCH 2005

ORGANIZED BY ISTE CHAPTER

This is to certify that Mr. / Ms. ....S.V.....MANISEKARAN.....

of.....KUMARAGURU.....COLLEGE....OF.....ENGG.....COIMBATORE..... *has participated and presented a paper titled "*....FREQUENT.....SET.....CALCULATION.....FOR.....DATA.....MINING.....

....."*,*

in the 2nd National Conference for Research Scholars on "APPLICATIONS OF EMERGING TECHNOLOGIES" *on 28 March 2005 at* **ADHIYAMAAN COLLEGE OF ENGINEERING.**

Dr. G. Ranganath
Vice-Principal &
Organizing Secretary - National Conference

Dr. G.K. Dwaraka Nath
Principal &
Chairman - National Conference