MINING ASSOCIATION RULE ALGORITHMS IN LARGE DATABASES

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of
Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science
in Computer Engineering, Computer Engineering Program.

by
Semih UTKU

July, 2004
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We certify that we have read this thesis and **"MINING ASSOCIATION RULE ALGORITHMS IN LARGE DATABASES"** completed by **Semih UTKU** under supervision of **Prof. Dr. Alp KUT** and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

[illegible]

Prof. Dr. Alp KUT
Supervisor

[illegible]

Yrd.Doç.Dr. Zafer Dicle
(Committee Member)

[illegible]

Doç. Dr. Yalçın Çebi
(Committee Member)

Approved by the
Graduate School of Natural and Applied Sciences

[illegible]

Prof. Dr. Cahit HELVACI
Director

ACKNOWLEDGEMENTS

I would like to thank my advisor, Prof. Dr. Alp KUT, for the encouragement, support and assistance in this thesis.

I also thank my officemates Hulusi BAYSAL and Gıyasettin ÖZCAN for their guidance and motivation.

I thank to my family, they were always with me. Finally, thanks to my wife Sündüz for her patience and understandings in support of my pursue of the M.Sc. degree.

ABSTRACT

This study explains the fundamentals of association mining and analyzes implementations of the well known association rule algorithms. Study focus on algorithms Apriori, Eclat, FP-Growth, Partition and Dynamic Itemset Counting.

Mining association rules consist of two main steps. First step finds the set of all frequent itemset; where the second step generates all high confidence rules among itemsets. There are several efficient association rule algorithms. This study briefly describes and systematizes most common association rule algorithms. Common principles and differences between the algorithms have been shown. Next, study thoroughly investigates pros and cons of these algorithms.

Five well known association rule algorithms are implemented. This experimental system is developed using Java under Windows 2003 Operating System. Run time behaviours of this algorithms are analyzed and compared with using different dataset. Finally, new approach is proposed for mining association rules. Its basic design and details are explained for this new approach

Keywords: Database Management Systems, Data mining, Association rules.

ÖZET

Bu çalışma, ilişkili veri madenciliğinin temel kavramlarını açıklar ve bilinen ilişkisel kural algoritmalarının uygulamasının analizlerini içerir. Uygulamalar Apriori, Eclat, FP-Growth, Partition ve Dynamic Itemset Counting algoritmaları üzerinde gerçekleştirilmiştir.

Veri madenciliği ilişkisel kuralları iki temel aşamadan oluşmaktadır. Birinci aşamada tüm sık geçen parça kümelerini bulunmakta, ikinci aşamada ise tüm sık geçen parça kümeleri arasında kuralları oluşturulmaktadır. Yaygın olarak kullanılan ilişkisel kural algoritmaları mevcuttur. Bu çalışmada yaygın kullanılan ilişkisel kural algoritmaları sistemleştirilmekte ve tanımlanmaktadır. Temel prensipleri ve birbiriyle olan farkları gösterilmektedir. Araştırmanın son aşamasında ise algoritmaların artı ve eksileri ayrıntılı olarak incelenmektedir.

Yaygın olarak bilinen beş algoritmanın uygulaması yapılmıştır. Bu oluşturulan sistem Windows 2003 işletim sistemi ve Java platformunda geliştirilmiştir. Uygulamaları yapılan algoritmaların çalışma performansları analiz edilmiş ve farklı veri kümeleri kullanılarak karşılaştırmaları yapılmıştır. Son olarak, yeni bir ilişkisel kural yaklaşımı önerilecektir. Bu önerilen yapının temel tasarımı ve detayları açıklanacaktır.

Keywords: Veri Yönetim Sistemleri, Veri Madenciliği, İlişkisel kurallar.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE
INTRODUCTION

Data Mining is a process of inferring knowlegde from databases or data warehouses. Data mining presents new perspectives for data collection analysis. The purpose of data mining is to idetify trends and patterns in data.

Over the past few decades, some new methods have been obtained about the capabilities of data collection and data generation. Data collection tolls have provided us with a huge amount of data. Data mining processes have required an integration of techniques from multiple disciplines such as, statistics, machine learning, database technology, pattern recognition, neural networks, information retrieval and spatial data anaylsis.

Data mining methods has used in many different area such as, business management, scientific, engineering, banks, data management, goverment administration and many other applications. Data mining is an iterative process containing several steps, beginning with the understanding and definition of a problem and ending with the analysis of results and determine a strategy with using the result.

Data mining, also known as knowledge discovery in databases, is a nontrial extraction of implicit, previosly unknown and potentially useful information from data (Frawley et al. 1991 p.127). 'Data Mining' and 'Knowledge Discovery' have been used interchangeably. In spite of this usage, there is different meaning between the two. Data mining algorithm cannot operate on raw data many data mining algorithms can read and process from flat files without additional steps. Data stores in a databases or datawarehouses so, data mining process may need to extract, format and convert the raw data before invoking the data mining algorithm. The extraction information can often help marketting, decision making and business management. Data mining process is an important area in database systems.

1.1 Applications and Uses of Data Mining

Data mining has three major components. (WEB_1, 1997).

- Clustering and classification: analyze a set of data and generate a set of grouping rules which can be used to classify future data.
- Association: implies certain association relationships among a set of objects in a database.
- Sequence analysis: seek to discover patterns that occur in sequence.

Many business and organization collect data about their operations and use data mining methods. Some organizations use data mining techniques to marketing. For example to anayzes customer buying habits. To obtain of such information can help retailers to improve marketting strategies. Banks use data mining to predict credit risks. Insurance companies use data mining to anaylze claims patterns for fraud and to predict high risk situations. Other applications include stok market analysis, predicting foreign exchange rates based on current financial indicators, determining commonalities or anomalies among classes of medical patients, modelling proteins, and finding genes in DNA sequences.

Another data mining application area is within the operations and communications of the organization itself. Security offices and law enforcement agencies have been applying data mining technologies to their data sets. They have analyzed all sorts of data sets including telephone tool calls, narcotics operations, financial crime enterprises, criminal organizations, border crossing, terrorist activities and a wide range of other activities. (Herwestphal C. et al., 1998, p.48).

1.2 Objective

Association Rule mining is one of the fundamental data mining method. Association is a rule, which implies certain association relationships among set of objects such as occur together or one implies the other. Goal of association rule is finding associations among items from a set of transactions which contain a set of items. Main problem of Association Rule inductions is that there are so many possible rules.

The objective of this thesis is to explain the fundamentals of Association Mining. Next, briefly describe and systematize most common association rule algorithms. There are several efficient association rule algorithms. Additionally, new proposals offered for algorithms that improve run time for generating association rules or frequent itemset. We thoroughly investigate pros and cons of these algorithms. In this thesis, a new approach proposed for mining association rules and its basic design and details explained.

Most of association rule algorithms generate Association Rule in two steps:

1. Find all frequent item sets;

   The foundation of Association Rule algorithm is fact that any subset of a frequent itemset must also be a frequent item set. i.e., if $\{AB\}$ is a frequent item set, both $\{A\}$ and $\{B\}$ should be a frequent item set

   Iteratively find frequent item sets with cardinality from 1 to *k (k*-item set*)*
2. Use frequent item sets to generate strong rules having minimum confidence.

1.3 Organization

This thesis includes five chapters except the first introduction chapter. Each chapter gives a summary on the topic mentioned on this chapter firstly, and then makes a detailed explanation of this topic. The thesis is organized as follows:

Chapter Two explain the basic fundamental description of data mining and mining process. Chapter Three, identify the association rule concepts, Chapter Four describe the most common algorithms furthermore we show the common principles and differences between the algorithms. Chapter Five describes the design and implementation of algorithms and experimentation results are presented. Later on, algorithms are compared and explain the design and details of our new approach. Chapter Six concludes the study explained in this thesis with a summary of our result and comments.

CHAPTER TWO
MINING PROCESS

The aim of the database system is to store of raw data. Definition of the data mining is the process of inferring knowledge from the database system. Discovery process for reach the knowledge consists of an iterative sequence of the steps. This chapter describes the mining process steps and define the types of data mining pattern.

2.1 Knowledge Discovery Process

Understanding and definition the problem is the first step in data mining process. Once a problem has been defined, relevent data must be collected. The relevent data is extracted from an existing database or data warehouse. Figure 1.1 illustrate the data mining process. Data mining process consist of the following steps. (Han J. & Kamber M., 2000).

2.1.1 Data Extraction

Data extraction process extracts usefull subsets of data for mining. Goals of the extraction process are, identifying concerned information in the database and processing the database into some suitable form to analysis by the data mining algorithms.

Data for modelling can be stored in many different sources. These sources are, databases, data-warehouses, web sites, flat files,etc.. Relevant data are collected from these sources during the extraction process. While the extraction process proceed some operations can be applied on the data. If the source includes the dublicate data which requires more storage space and creates the problem of maintaining consistency, these data are eliminated from the source database.

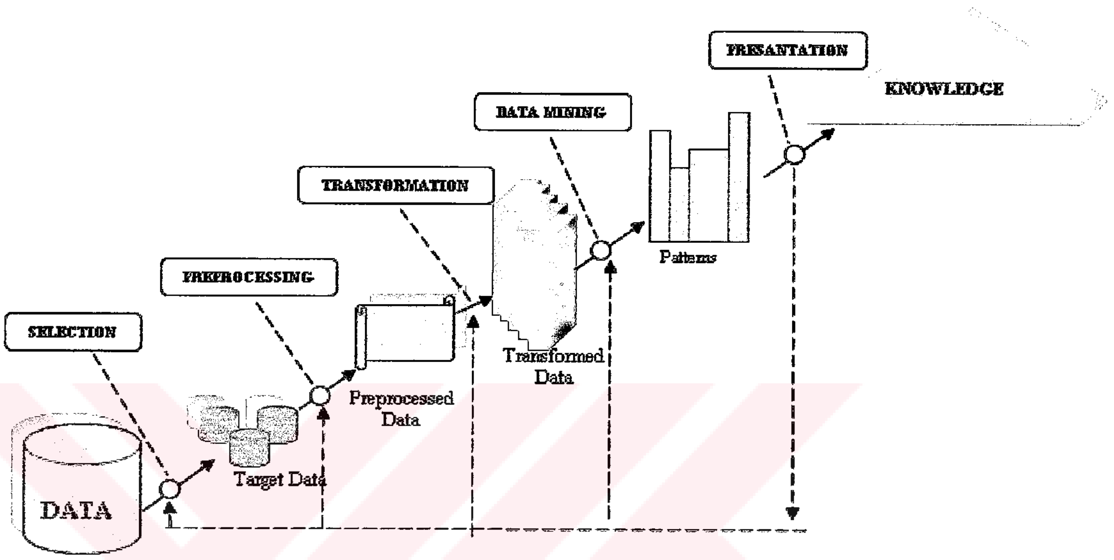


Figure 1.1 Data Mining Process

2.1.2 Data Preprocessing

Data preparation is one of the most important steps in the data mining process. Large database systems generally contain errors in the stored data. Examine the data for errors, outlines and missing values to the quality of the data. This is the most time consuming and most important step in the data preparation process. Robustness is an important property for the data mining systems. So, some techniques are used to managing to data in this process.

Data cleaning can be applied to remove noise and inconsistency in the data. There are many reasons for noisy and incomplete data. Some of methods are used to filling in the missing values. The most famous one is the regression methods for data cleaning. Data cleaning. Data integration merges data from multiple sources. These sources may include multipe databases, data cubes, or flat files. The main problem of the integration is data confliction. Data transformation operations used for normalizations and aggregation. Data are transformed or integrated into form suitable for mining. Data transformation process includes smoothing, generalization, normalization and aggregation techniques. Data reduction operation used for reduce the data size by using one of the data aggregation, dimension reduction or data comparison methods. Data reduction methods can be used to minimizing represantation of the data, while reducing the loss of information content.

2.1.3 Data Mining

All of the raw data are created and cleaned in the previous steps. Thus, data are prepared for the data mining stage. Prepared data might contain many attributes and we have to select a subset of the attibutes for using in data mining process.

A Data mining algorithm takes data as input and produces output in the form of models or patterns. In this step an intelligent methods are applied in order to extract data patterns. Visualization, classification, clustring, regression or association algorithms are used for different problem. There are many algorithmic approaches to extracting useful information from data. As these are the basis for this thesis we will defer discussion of the specifies until later sections.

2.1.4 Patern Evaluation

Data mining system generate lots of patterns, or rules. Only small fraction of the patterns which are generated from the data mining system are interested to any given problem. Patterns have to be easily understood, useful, and interesting. An interesting pattern represents knowledge. After determination of the knowledge function, some measured functions which are used to separate uninteresting patterns from knowledge, are used for the data mining process. For example, association rules use support and confidence measure functions to determine the rules.

Completeness of the data mining is an another patern evaluation problem. Generally, data mining system cannot generate all of the interesting pattern. Thus, data mining algorithms use some constraints and interestingness measures to ensure the completeness of mining. Hence, optimization problem is one of the important problem for the data mining.

2.1.5 Knowledge Presentation

Visualization and Knowledge represantation techniques are used to present the mined knowledge to the user. We have made a final pass through the data mining process, we are ready to implement some strategy based on the discovered patterns.

An important goal of data mining is to apply what has been learned. In the knowledge presantation step some possible actions formed from the successful apllication. Creation of a report or technical article, relocation of retail items for purchase or placement of selected items on sale together, funding of a new scientific study motivated by what has been learned from a knowledge discovery process are some possible example.

2.2 Types of Data Mining Pattern

This section describes the types of data mining pattern which are classification, association rule and clustring. There are many different algorithms are used in these data mining pattern.

2.2.1 Classification and Prediction

Classification is the process of finding set of functions. One important feature of a classifer that is important to data miners is that the resulting function be ultimately understandable. The predicted attribute is called the class. Result functions describe data classes or concepts. The purpose of the classification and prediction method is to predict the class of objects whose class label is unknown. Each record consists of a goal attribute and a set of predicting attributes.

Training data set used to derive this model. There are more powerful classification technique like Classification Rules (IF - THEN), Mathematical Formula, Decision Trees or Neural Networks. In this part, I describe decision tree which is the famous methods for classification and prediction.

2.2.1.1 Decision Tree Induction

A decision tree is a tree structure using for a visual representation of the classification technique. Internal nodes denotes a tests on predicting attributes, branch denotes a outcome of the test, and leaf nodes represents the value predicted for the goal attribute.

There are many other classification methods can be used to finding set of classes. These methods are Bayesian classification which is the statistical classifiers based on Bayes Theorm. Classification by Backpropagation is a neural network learning algorithm. K – nearest neighbour classification, genetic algorithm, fuzzy set and rough set are the other classification methods. A decision tree is constructed by initially selecting a subset of instances from a training set. This subset is then used by the algorithm to construct a decision tree. The steps of the decision tree algorithms summarized as follows: (Hofer J., & Brezany P., 2004, p.41)

```
create a node N;
 if samples are all of the same class, C then
      return N as a leaf node labeled with the class C;
  if attribute-list is empty then
      return N as a leaf node labeled with the most
      common class in samples;
  select test-attribute, the attribute among attribute-
list
    with the highest information gain;
  label node N with test-attribute;
for each known value $a_i$ of test-attribute grow a branch
  from node N for the condition test-attribute=$a_i$;
    let $s_i$ be the set of samples in samples for which
  test-
      attribute=$a_i$;
    if $s_i$ is empty then
      attach a leaf labeled with the most common class in
        samples;
    else attach the node returned by Generate
      decision tree;
```

2.2.1.2 A Decision Tree Example

Decision Tree algorithm searches through the space of possible decision trees from simplest to increasingly complex, guided by the information gain heuristic. Table 2.1 includes some data for construct the decision tree and figure 2.1 illustrates the example.

Table 2.1 Example Weather Data

Outlook	Temperature	Humidty	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainly	Mild	High	False	Yes
Rainly	Cool	Normal	True	No
Rainly	Cool	Normal	True	No
Overcast	Cool	High	True	Yes
Sunny	Cool	Normal	True	Yes

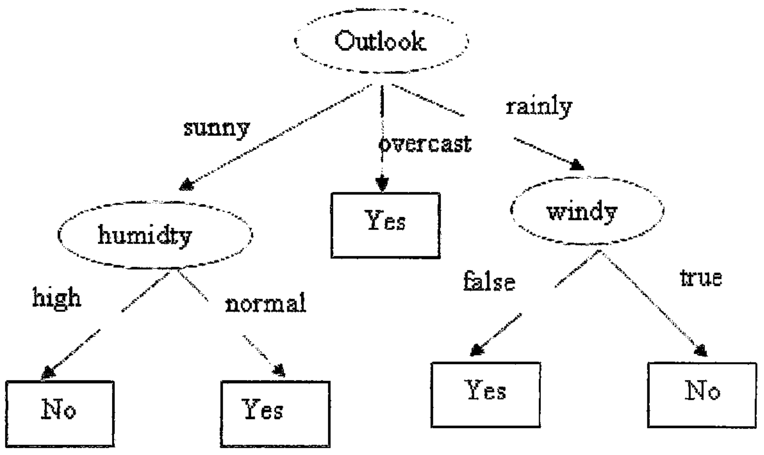


Figure 2.1 Play - Decision Tree depend on the weather

2.2.2 Association Rules

The objective of this thesis is to explain the fundamentals of Association Mining. Association rules and association algorithms are the basis for this thesis we will defer discussion of the specifies until later chapters.

Association rule mining helps in finding interesting association relationships among large set of data items. The discovery of such associations can help develop strategies to predict.

An association rule is a relationship. An Association Rule is a rule of the format LHS (left hand side) => RHS (ride hand side). X and Y where X, Y $\subset$ I (both side contains sets of items) and $X \cap Y = \varnothing$ (don't share common items).

Briefly, an Association Rule is an expression;

$X \Rightarrow Y$, where X and Y are set of items.

Each rule is assigned two factors:

Support $_{X \Rightarrow Y} = \sigma\ (X \cup Y) = {}_s(X \cup Y)$ / # of tubles (where probability denotes as σ)

Confidence $_{X \Rightarrow Y} = \sigma\ (X \mid Y) = {}_s(X \cup Y) / {}_s(X)$

The aim of the algorithm is to discover all association rules with

$\text{Sup} \geq \text{minimum\_Sup}$ and

$\text{Conf} \geq \text{minimum\_Conf}$.

2.2.3 Clustring

Another kind of data mining pattern is a clustring model, also called a segmentation model. Clustring is the process of grouping the data into class. The goal of the clustring is to take a set of objects which are records in a database or datawarehouses and to partition them into number of groups or clusters. (Berkhin P. & Software A., 2002).

Clustring is the process of grouping the data into classes. The process of deciding how to use clustring pattern is the task for domain expert and data mining analyst. They work together to understand each cluster. Cluster analysis method has been used in pattern recognition, data analysis and image processing. After clustering, we can apply classification methods to discover rules predicting membership in a given class.

Clustering techniques have been studied overall in statistics, machine learning and data mining. Clustering can be used on the principle of maximizing the intra-class similarity and minimizing the interclass similarity. Cluster analyzes is used in different area. In marketing, characterize customer groups based on purchasing pattern. In Earthquake studies, observed earthquake epicenters should be clustered.

In biology, it can be used to categorize genes with similarity, animal taxonomies. Clustering may also used in Land use, City planning and Insurance.

There are many techniques which are hierarchical methods, partitioning methods, density–based methods, grid–based methods, and model–based methods, are used in clustring analyzes. I describe the classical partioning methods which is the commonly used for clustring. K – means method is the most well known technique used in partioning methods.

The first step of the k- means algortihm, choose a value of K, next, k instances are selected randomly. Each instance is then placed in the cluster to which it is most similar. Then cluster center are updated until an iteration of the algorithm shows no change in the cluster centers.

Here is the algorithm of k-means.

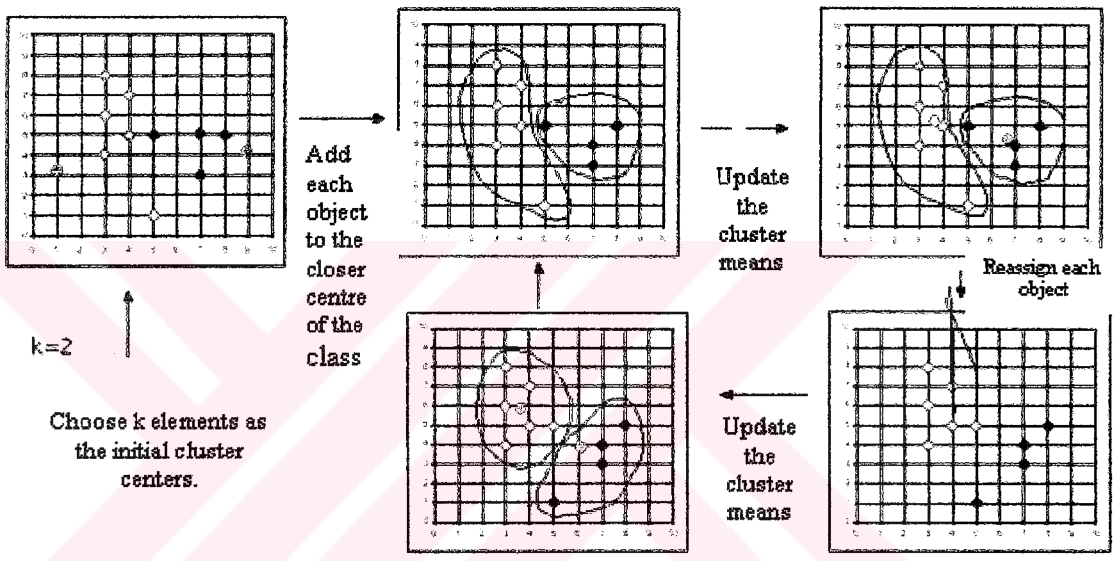
```
arbitrarily choose k objects as the initial cluster
centers;
repeat
     assign each object to the cluster to which the
object is the most similar,
     based on the mean value of the objects in the
cluster;
     update the cluster means;
until no change;
```

2.2.3.1 K -Means Example

For our example we will use the ten instances in table 2.2. For to make easy name of the attributes x and y, respectively and map the instances onto an x-y coordinate system. All Data objects are arrenged in the rectangle. Let k=2, and k- means algorithm illustrated in figure 2.2. The repeatation operation continue until no change of the objects. The running steps of the k-means algorithm is given above. (This example is inspired from the Han J. & Kamber M., 2000, p.352)

Table 2.2 k – means input values.

Instance	X	Y
1	3	4
2	4	5
3	3	6
4	5	1
5	4	7
6	3	8
7	5	5
8	7	3
9	8	5
10	7	5



k=2
Choose k elements as the initial cluster centers.
Add each object to the closer centre of the class
Update the cluster means
Reassign each object
Update the cluster means

Figure 2.2 Clustring example.

2.3 Classification vs. Association vs. Clustring

In this chapter defines the types of data mining pattern which are classification, association and clustring. These data mining pattern explained above. (Association Rule is investigated in next chapter). Here is the comparison of the pattern.

Classification:

- ✓ a single goal attribute and a set of predicting attributes
- ✓ part of the problem is to determine the relevance of the predicting attributes
- ✓ quality of a discovered rule is much more difficult to assess

- ✓ the difficulty is to design *effective* algorithms to discover high-quality knowledge

Association:

- ✓ all items can appear either in the antecedent or in the consequent of the rule
- ✓ quality of a discovered rule is depended on Sup and Conf factors.
- ✓ the system simply has to find *all* rules determined by user-defined Sup and Conf
- ✓ the difficulty is to design *efficient* algorithms

Clustering:

- ✓ no special attribute
- ✓ all attributes are usually considered equally relevant
- ✓ the difficulty is the quality of the discovered clusters
- ✓ used mainly for data exploration and data summarization

CHAPTER THREE
ASSOCIATION RULE

Types of data mining patterns are explained in the previous chapter. Concepts and fundamentals of Association Mining which is the important and popular for data mining application thoroughly investigated in this chapter.

3.1 Associaiton Rules

Association rule mining explores for interesting relationships among items in a given data set. Association analysis is used for market basket or transaction data analysis. Market basket process finds associations between the different items. An objective of association rule mining is to develop a systematic method using the given data set and finds relationships between the different items.

Association is a rule, which implies certain association relationships among set of objects such as occur together or one implies the other. (Kumar P., 2001) Goal of association rule is finding associations among items from a set of transactions which contain a set of items. Main problem of Association Rule inductions is that there are so many possible rules. A typical association rule mining algorithm has the following components:

- Structure (used for implementation of the pattern)
- Score function (simply binary function, support and confidence)
- Search method (systematic search methods)
- Data management technique (number of linear scan through the database)

Search and data management components are the most critical components. Thus, aim of the all proposals is to design *efficient* algorithms for association mining.

Efficient algorithms are needed which restrict the search space for minimize the algorithmic complexity and interesting rules must be picked from the set of generated rules. The search space for enumeration of all frequent item sets is 2^m, where m represents the number of items. The rule generation step's complexity is O ($r.2^l$), where r is the number of frequent item sets, and *l* is the longest frequent item set. (M. J. Zaki, 2000).

Mining association rules can be divided into two sub-problems:

- Find all frequent itemsets which have the support greater than the minimum predetermined *min_sup* in the transaction database.
- Generate all association rules X⇒Y, which have confidence greater than predetermined *min_conf.*

3.2 Formal Problem Description

Association Rule problem can be described as follows (R. Agrawal et al. 1993):

Let $I = \{i_1, i_2, i_3, \ldots i_n\}$ be a set of items. Each transaction, *T*, is also a set of items which non-empty subset of *I*, is called an *itemset,* and $T \subset I$. An itemset with *k* items is called a *k-itemset.*

An Association Rule is a rule of the format LHS (left hand side) => RHS (ride hand side). X and Y where X, Y $\subset$ I (both side contains sets of items) and $X \cap Y = \emptyset$ (don't share common items).

Briefly, an Association Rule is an expression;

X ⇒ Y, where X and Y are set of items.

Above rule shows the transaction of the database which contain X tend to contain Y. various metrics describe the utility of an association rule. The most common one is the percent of all transaction containing $A \cup B$, called the *support*. The support of the rule denoted as $\sup(X \Rightarrow Y)$. The other one is the percent of transactions containing B among transactions containing A, called the *confidence*. The confidence of the rule denoted as $\text{conf}(X \Rightarrow Y)$. They defined as;

Support $_{X \Rightarrow Y} = \sigma (X \cup Y) = {}_{s}(X \cup Y)$ / # of tubles (where probability denotes as σ)

Confidence $_{X \Rightarrow Y} = \sigma (X \mid Y) = {}_{s}(X \cup Y) / {}_{s}(X)$

Generally, the algorithm finds a subset of Association Rule that satisfies certain constraints. Minimum Support and Minimum Confidence are commonly used constraints for the association rules. Given a user specified minimum support threshold and minimum confidence threshold, the aim of the association rule mining is to find complete set of association rules in the database with support and confidence passing thresholds, respectively.

An itemset is *frequent* if its support is higher than or equal to a given minimum support (min_sup) value $Support_{X \Rightarrow Y} \geq min\_sup$.

A rule is *confident* if its confidence is more than or equal to a given minimum confidence (min_conf) value $Confidence_{X \Rightarrow Y} \geq min\_conf$.

Most of association rule algorithms generate Association Rule in two steps (Z. Zheng et al. 2001):

1. Find all frequent item sets;

   The foundation of Association Rule algorithm is fact that any subset of a frequent item set must also be a frequent item set. i.e., if $\{AB\}$ is a frequent item set, both $\{A\}$ and $\{B\}$ should be a frequent item set

   Iteratively find frequent item sets with cardinality from 1 to *k (k*-item set*)*

2. Use frequent item sets to generate strong rules having minimum confidence.

The first step, find all frequent item sets, is expensive in terms of computation, memory usage and I/O resources. Much of the research effort in association rule algorithms has been related to improving the efficiency of this first step. The second step, use frequent item sets to generate strong rules having minimum confidence, is relatively certain, but it can still be very expensive when solving real-world problems.

Example-3.1 (Association rule mining):

Transaction database, transaction_DB1, shown in Table 3.1. We use <2, {A, B, C}> is a transaction, in which 2 is the transaction identifier, and {A, B, C} is an itemset. Given min_support = 40% and min_confidence = 50%, association rules can be mined as two-step process.

1. Find all frequent itemsets.
   80% of all transactions contain {A}
   40% of all transactions contain {B}
   80% of all transactions contain {D}
   40% of all transactions containing {A, B}
   40% of all transactions containing {A, D}
2. Generate all association rules on each frequent itemset.

Table 3.1 the transaction database transaction_DB1

Transaction_id	Items in transaction
1	A, B
2	A, C, D
3	A, F
4	A, D, B
5	D

Min support =	40 %
Min confidence =	50 %

For rule $A \Rightarrow D$:

Support = Support ($\{A \cup D\}$) = 40%

Confidence = Support ($\{A \cup D\}$) / Support ($\{A\}$) = 50%

In this example, we generate the association rules $A \Rightarrow D$ (all transactions which contain {A} also contain {D}), $A \Rightarrow B$, $AB \Rightarrow D$, $AD \Rightarrow C$: with given min_sup & min_conf thresholds.

These rules are very simple, understandable and useful for association rule mining. However, the determination of these rules is a major challenge due to the very large data sets and the large number of potential rules.

Example-3.2 (Association rule mining):

A subset of market basket database, shown in Table 3.2. Subset database is used to construct association rules in this example. The first step is to create single item – sets. Single item – set represents the items count in the given subset database. In table 3.3, the single item– sets are illustrated. Then combine single item – sets to create two item – sets. The next step is to use the attribute value combinations from the two item – sets table to generate three item – sets.

Table 3.2 A Subset of Market Basket

Bread	*Milk*	*Cheese*	*Detergent*	*Sex*
Yes	Yes	Yes	Yes	Male
Yes	No	No	No	Male
Yes	Yes	Yes	No	Female
No	No	No	No	Female
Yes	No	No	No	Male
Yes	No	Yes	Yes	Male
No	No	No	No	Male
Yes	Yes	Yes	No	Female
Yes	No	Yes	No	Female
No	Yes	No	No	Male

Table 3.3 Single Item – Sets

Item - Set	*# of Items*
Bread = yes	7
Milk = no	6
Cheese = yes	5
Detergent = no	8
Sex = male	6
Sex = female	4

Table 3.4 Two Item – Sets

Two Item - Set	*# of Items*
bread = yes & milk= no	4
bread = yes & cheese = yes	5
bread = yes & detergent = no	5
bread = yes & sex = male	4
milk = no & cheese = no	4
milk = no & detergent = no	5
milk = no & sex = male	4
cheese = no & detergent = no	5
cheese = no & sex = male	4
detergent = no & sex = female	4
detergent = no & sex =male	4

Association Rules are generated from the two and three item – sets tables. Finally, any rule not meeting the minimum confidence value is discarded.

Possible two item – sets rules are:

Bread = yes ⇒ Cheese = yes (confidence = 5 / 7, 71%)

Milk =no ⇒ Detergent = no (confidence = 5 / 6, 83%)

Possible two item – sets rules are:

Cheese =no & Milk =no ⇒ Detergent = no (confidence = 4 / 4 , 100%)

Milk =no ⇒ *Cheese =no &* Detergent = no (confidence = 4 / 6 67%)

3.3 Discovery of Frequent and Closed Frequent Itemsets

A fundamental problem for mining association rules is to mine frequent itemsets. An temset is *frequent* if its support is higher than or equal to a given minimum support. In a transaction database, if we know the support values of all frequent itemsets, the association rules generation is simple. Nevertheless, when the given database which includes set of items contains large frequent itemsets, finding all the frequent itemsets steps might be difficult.

Typical algorithms are needed which restrict the search space for minimize the algorithmic complexity and interesting rules must be picked from the set of generated rules. The search space for enumeration of all frequent item sets is $2^m - 2$, where m represents the number of items. (M.J. Zaki., 1999). In other words, if frequent itemsets are long, it simply gets hard to mine the set of all frequent itemsets.

Typical algorithms begin with, computation of all the frequent 1-itemsets and then extends one level up in every pass (2, 3, ... , n itemsets) until all frequent itemsets are discovered. We can categorize the all itemsets in the database into three sets.

i. *frequent :* the set of discovered as frequent
ii. *infrequent :* the set of discovered as infrequent
iii. *unclassified :* the set of all the other itemsets.

During the generation of the the frequent 1- itemsets some of the itemsets are pruned whose count is lower than minimum support, as they don't need to be further. The remaining itemsets form the set of candidate. As the second step the support for these itemsets is computed, and they are classified as either frequent or infrequent.

All these generation itemsets process requires to database scan for discover the frequent itemsets. The cost of the frequent itemsets finding process comes from the reading of the database (I / O time) and the generation of the new candidates (CPU time). The number of candidates the most important parts the entire processing time. Decreasing the number of candidates not only can reduce the I / O time but also can reduce the CPU time, since fewer candidates require to be counted and generated. (Lin D. & Kedem M. 1997). Hence, reducing the number of candidates is the critical importance for the efficiency of the process.

In order to solve this problem, several solutions have been proposed that only generate a representing subset of all frequent itemsets. An interesting alternative to this problem is the mining of *frequent closed itemset*s and their corresponding association rules. Frequent closed itemset approach reduce the algorithm computation cost.

3.3.1 Closed Frequent Itemsets

A transaction database is a set of transactions, where each transaction, denoted as a tuple (tid, items), contains a set of items (i.e., X, Y ...) and is associated with a unique transaction identity tid. (Wang J. & Karypis G., 2004). Let I $= \{i_1, i_2, i_3, \ldots i_n\}$ be the complete set of distinct items appearing in transaction database. An itemset Y is a non-empty subset of I and is called an l - itemset if it contains l items. An itemset *{x_1, x_2, x_3, ... x_l}* is also denoted as x_1, ... x_l. A transaction (tid, X) is said to contain itemset Y if $Y \subset X$. The number of transactions in Transaction database containing itemset Y is called the support of itemset Y , denoted as sup(Y). Given a minimum support threshold, an itemset Y is frequent if sup(Y) ≥ min sup.

Definition : (Frequent closed itemset). An itemset Y is a ***frequent closed itemset*** if it is frequent and there exists no proper superset $Y \subset Y'$ such that sup(Y') = sup(Y). (Pasquier N. et al., 1999)

Discover all the association rules is not necessary. There exists a set of necessary association rules which can be mined from the closed frequent itemsets and these rules can be used to obtain all the association rules.

Mining frequent closed itemsets provides complete and necessary results for frequent pattern analysis. Overall studies have proposed different strategies for efficient frequent closed itemset mining. Some of them are, depth first search vs. breadth first search, vertical formats vs. horizontal formats, tree- structure vs. other data structures, top-down vs. bottom- up traversal.

Example: Transaction database, transaction_DB2, shown in Table 3.5. (This example is inspired from the J. Pei et al., 2000). Suppose min sup = 2, we can find the items count and sort the list of frequent items in support descending order. The sorted item list is called cfi_list. In this example,
cfi_ list = <f:4, c:4, a:3, b:3, m:3, p:3 >

The frequent items in each transaction are sorted according to cfi_list and shown in the third column of Table 3.5.

Itemset *(fc)* is a frequent 2-itemset with support 3, but it is not closed, because it has a superset *(fcam)* whose support is also 3. *(facm)* is a frequent closed itemset.

Table 3.5 the transaction database transaction_DB2

Transaction_id	Items in transaction	Ordered frequent item set
1	A, C, F, M, P	F, C, A, M, P
2	A, C, D, F, M, P	F, C, A, M, P
3	A, B, C, F, G, M	F, C, A, B, M
4	B, F, I	F, B
5	B, N, C, P	C, B, P
6	K, L	-

Due to generating a very large number of itemsets and association rules, it is difficult to mine complete set of itemsets.

Frequent closed itemsets provides the same analytical power as the finding all frequent itemsets. At the same time closed frequent itemsets can be used to obtain all the association rules. Thus, closed frequent itemset methods are more interesting and effective than the former.

CHAPTER FOUR
ASSOCIATION ALGORITHMS

Concepts and fundamentals of Association Mining are explained in the previous chapter. This chapter, briefly describe and systematize most common association rule algorithms. Basic design concepts of these algorithms has been examineted in details. Common principles and differences between the algorithms have been showed. Furthermore, I thoroughly investigate pros and cons of the most common association algorithms.

4.1 Introduction

There are many new and efficient algorithms have been developed in association mining. Additionally, new proposals offered for algorithms that improve run time for generating association rules or frequent itemset.

These algorithm have used different itemset generation approach. Classical algorithms have used frequent itemset generation technique. The new generation algorithms have used closed frequent itemset generation and maximal frequent itemset generation technique.

Having taken the dataset as input, Association algorithms, generates association rules, consequently final output will be a list of all whose confidence and support are above the minimum thresholds.

4.2 Organization of Itemsets

We have to count the transactions to find the frequent itemsets. Generally, tree structure is used to organize the itemsets. Aim of the itemset tree is storing itemsets efficiently and supporting the process of transaction.

Some of the association rule algorithms use different data structures which may be used for the nodes of this tree, and the other use different pruning techniques for generating the frequent-itemsets.

Figure 4.1 illustrates the complete itemset tree in our example database (see Table 3.1)

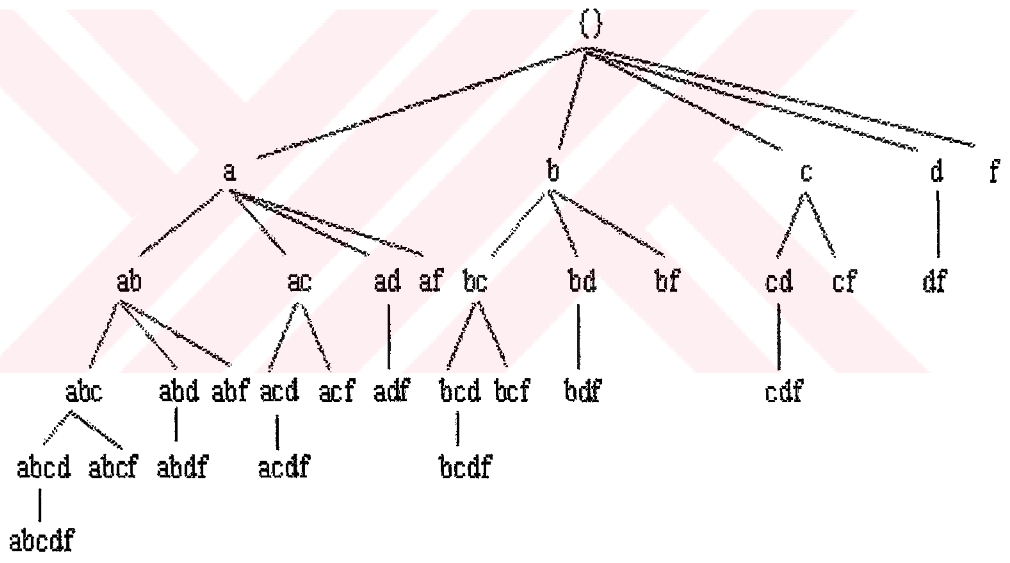


Figure 4.1 Full itemset tree, for our example database

This tree is unbalanced,because node n_{abc}, is the same as nodes n_{cba}, n_{bca}, n_{cab}, n_{bac}, n_{acb} as a result, we need one of these nodes to construct the itemset tree.

Association algorithms use different methods for generating the itemsets which are, Hash_table, transaction reduction, partitioning, sampling and dynamic item set counting. Aims of these methods are improving the performance of the algorithms and eliminating repeated database scan.

4.3 Frequent Itemsets Algorithms

The problem of mining association rules was first proposed by Agrawal [Agrawal et. al. 1993]. Agrawal has introduced one of the most popular association rule algorithms called Apriori to find frequent itemsets. Apriori and all of the other algorithms based on the principles of knowledge of frequent itemsets "all non-empty subsets of a frequent itemset must also be frequent".

There are many variations or extensions algorithms have been proposed with the aim of improving performance and efficiency. They use different data structures or methods which are hash-table, sampling, partioning, dynamic itemset counting etc.

For instance, DHP algorithm applies a hash-table technique to improve the frequent itemsets generation performance. Another algorithm partition, partitions transactions in the database and applies algorithm then verifies the frequency of the itemset in the entire database. Another alternative algorithm DIC uses dynamic itemset counting. FP-Growth, Eclat, Clique, Apriori Hybrid are the other proposed algorithms for generating association rules.

4.3.1 Systematization of the Algorithms

Association algorithms differ concerning with itemset tree traversal. Most approaches use a bottom-up search of the itemset tree to enumerate the frequent itemsets. If the size of itemset is large, a top-down approach might be preferred. Some of the others use hybrid search which combines top-down and bottom-up approaches. We evaluated some algorithms and Figure 4.2 shows the algorithms and their search methods.

Bottom-up	**Hybrid**	**Top-down**
Apriori	AprioriHybrid	FP-Growth
Apriori TID	MaxClique	Eclat
DIC	MaxEclat	
Partition		
Clique		
DHP		

Figure 4.2 Systematization of the Algorithms.

4.3.2 Apriori Algorithm

Apriori, (Agrawal et al. 1993) and (Borgelt C., & Kruse R., 2002) the Apriori algorithm is the most popular association rule algorithm. Apriori have used bottom-up search. Using multiple database scan would have been significantly slower. Appriori algorithm works as follows:

The first, Apriori algorithm counts occurences for the each item in the entire database and generates the set C_1 of Candidate 1 – itemsets. Then, itemsets count and minimum support value are compared to find the set L_1. The second,algorithm use L_1 to construct the set C_2 of Candidate 2 – itemsets. The process is finished when there are no more candidates. In each phase, all the transaction in the data set are scanned. Finally, all frequent itemsets are returned.

Apriori uses two step processes which are Join and Prune to create L_k from L_{k-1}. k-itemsets (L_k) are used to explore (k+1)- itemsets and Candidate (C_k) is generated by joining L_{k-1} with itself. Apriori algorithm scan entire dataset to generate single itemsets with a support level equal to or greater than the specified minimum support. After generating the itemsets C_1, then, itemset will be joined to generate candidate C_2 sets. If C_k greater than minimum support; C_k added in L_k else C_k pruned off.

The main idea of Apriori is outlined in the following:

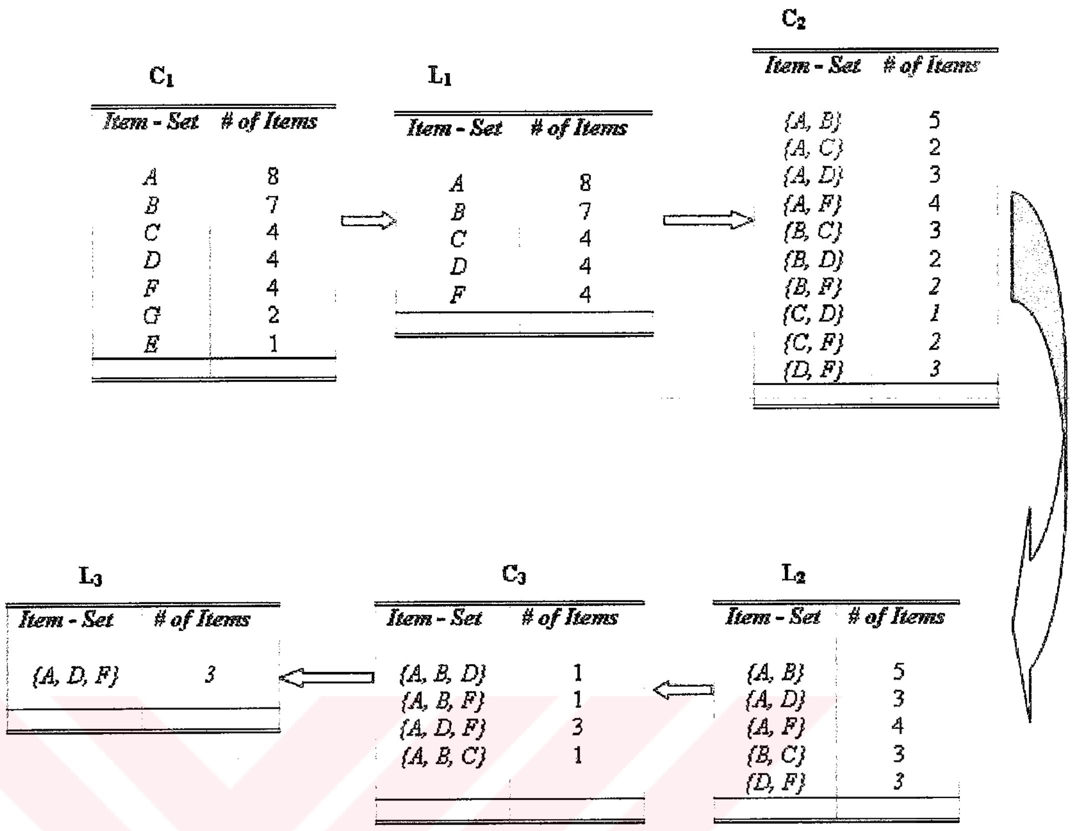
```
$L_k$ : Frequent itemset of size k
$C_k$ : Candidate itemset of size k
$L_1$ = {frequent 1- items};

For (k = 1; $L_1$ !=∅; k++) do begin
     $C_{k+1}$ = candidates generated from $L_k$;
    For each transaction t in database do
          Increment the count of all candidates in $C_{k+1}$
          that are contained in t

    $L_{k+1}$ = candidates in $C_{k+1}$ with min_support
End
Return $\cup_k$ $L_k$ ;
```

Example 4.1: Consider the following transaction_DB3.

Table 4.1 transaction_DB3

Transaction_id	Items in transaction	Ordered frequent item set
1	A, B, C, F	A, B, C, F
2	D, B, C	B, C, D
3	B, C	B, C
4	A, F, D, G	A, D, F, G
5	A, E, C	A, C, E
6	A, B, D	A, B, D
7	A, B	A, B
8	A, D, F	A, D, F
9	A, B	A, B
10	A, B, G	A, B, G

Let user defined minimum support threshold be 3 (60 %). Appriori first generates C_1 then, count the each candidate 1 – itemset and minimum support value are compared to find the set L_1. The algorithm uses the join operation $L_1 \infty L_1$ to generate C_2 candidate 2- itemsets. Next all transactions are scanned to find itemsets count in C_2. The process is finished when there are no more candidate itemsets. Figure 4.3 illustrate of this process.



C₁

Item - Set	# of Items
A	8
B	7
C	4
D	4
F	4
G	2
E	1

L₁

Item - Set	# of Items
A	8
B	7
C	4
D	4
F	4

C₂

Item - Set	# of Items
{A, B}	5
{A, C}	2
{A, D}	3
{A, F}	4
{B, C}	3
{B, D}	2
{B, F}	2
{C, D}	1
{C, F}	2
{D, F}	3

L₂

Item - Set	# of Items
{A, B}	5
{A, D}	3
{A, F}	4
{B, C}	3
{D, F}	3

C₃

Item - Set	# of Items
{A, B, D}	1
{A, B, F}	1
{A, D, F}	3
{A, B, C}	1

L₃

Item - Set	# of Items
{A, D, F}	3

Figure 4.3 Apriori Frequent itemsets generation process in our example database transaction_DB3.

Using the frequent itemset {A, D, F} which have subsets ({A}, {D}, {F}, {A, D}, {A, F}, {D, F}), generated association rules are as shown below.

$A \& D \Rightarrow F$	confidence = 3 / 3 = 100 %
$A \& F \Rightarrow D$	confidence = 3 / 4 = 75 %
$D \& F \Rightarrow A$	confidence = 3 / 3 = 100 %
$A \Rightarrow F \& D$	confidence = 3 / 8 = 37.5 %
$F \Rightarrow A \& D$	confidence = 3 / 4 = 75 %
$D \Rightarrow F \& A$	confidence = 3 / 4 = 75 %

4.3.3 Partition Algorithm

Partition is a different approach from all the others. (Savasere A. et al. 1995) This alogortihms use set intersection operation instead of counting. Supports of an itemsets evaluated with its subsets. Partition algorithm combines the Apriori with set operation (intersection).

Partition algorithm stores the complete database which includes the transaction sets, into main memory. For huge databases, this could be imposible. Neverthless, the partition algorithm partitions database into several chuncks to solve the problem that, is the main memory limitations. (Leung K. S. 2002, p.43).

The main idea of Partition is outlined in the following:

```
Output: $F(D, \sigma)$
Partition D in $D_1, \ldots, D_n$
// Find all local frequent itemsets
for $1 \le p \le n$ do
  Compute $Cp := F(Dp, [\sigma\_rel \cdot |Dp|])$
end for
// Merge all local frequent itemsets
// Compute actual support of all itemsets
for $1 \le p \le n$ do
  Generate cover of each item in Dp
  for all $I \in C_{global}$ do
    $I.support := I.support + |I[1].cover \cap \cdots \cap I[|I|].cover|$
  end for
end for
// Extract all global frequent itemsets
$F := \{I \in C_{global} \mid I.support \ge \sigma\}$
```

Partitions algorithm executes in two steps.

In the first step;

- algorithm divides transactions into several partitions. Then, Apriori algorithms applied to divided transactions in each partitions to find the itemsets. This called "*locally frequent*"

In the second step;

- the locally frequent itesets become the candidates for globally frequent itemsets. The algorithm evaluates the global support counts for locally frequent itemsets and checks if they are globally frequent.

Generation of local itemsets and generation of final itemsets are important parts of the algorithm. To generate the local itemset, initially frequent 1 – itemsets are found then to generate candidate k – itemset join the two (k-1) –itemsets. For instance, candidate itemset {1, 2, 3, 4} is generated by joining the itemsets {1, 2, 3} and {1, 2, 4}. To generate the final itemsets, all local itemsets from the all partitions are merged to generate the global frequent itemsets.

Partition algorithms improve the performance with using an important principles "any globally frequent itemset must appear as a locally frequent itemset in at least one of the partitions of the database". (M.J. Zaki., 1999).

The partition algorithm merges all local frequent itemsets of every part. If an item is frequent in the entire database, it must be relatively frequent in one of the parts. First, Every part is read into main memory using the vertical database layout next, frequent itemsets are found into each partitions then, support of every itemsets are computed. Finally, global itemset frequency are determined using the intersection operation.

If the database heterogeneous, then its decrease the performance of the algorithm. Because, there can be too many local frequent itemsets. On the other hand, if the entire database fits into main memory, then divides the transactions into several partitions not necessary.

4.3.4 DIC Algorithm

DIC (Dynamic ItemSet Counting) algorithm which uses fewer database scan, presents a new approach for finding large itemsets.

DIC is an extension of Apriori algorithm. (Brin S., et al., 1997) Apriori algorithm first, counts all the 1 – itemsets and finds frequent 1 – itemsets which exceed the min_support threshold. Then, counts all 2- itemsets and determines the frequent 2 – itemsets. Thşs process continuous until there are no more candidate itemsets. Thus, Apriori fulfil as many passes over the data.

Aim of the DIC algorithm is improving the performance and eliminating repeated database scan.

DIC algorithm divides the database into partitions (intervals M) and use a dynamic counting strategy. DIC algorithm determines some stop points for itemset counting. Any appropriate points, during the database scan, stopping counting, then starts to count with another itemsets. For instance, our database contains 60000 transactions and M = 15000, algorithm starts to count all the 1- itemsets same with Apriori. In addition to this, algorithm begin to counting 2 - itemsets after the first 15000 transactions have been scaned. Then, its begin to counting 3 – itemsets after 30000 transactions. Once the algorithms reaches the end of the data file, stop counting the 1 – itemsets then go back to count the 2 and 3 – itemsets. After 15000 transactions, algorithm finishes the counting the 2- itemsets and after the 30000 transactions algorithm finish the count 3 – itemsets. Consequently, algorithm have finished counting all the itemsets in 1.5 passes over database, while in Apriori requires 3 passes. Figure 4.4 illustrate this process.

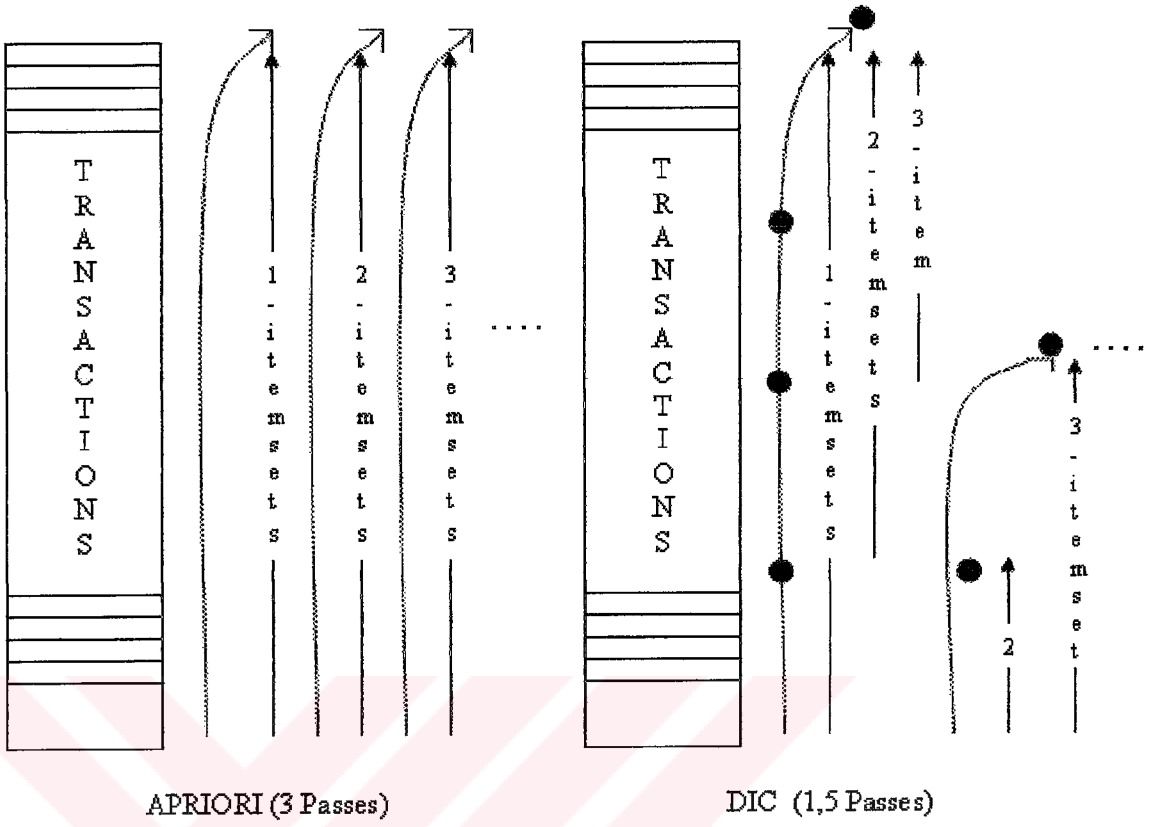


Figure 4.4 Apriori and DIC

To implement this algorithm, four symbols to indicate the different states of itemsets.

- Solid Box – indicates the itemsets that finished counting and that exceed the support threshold;
- Solid Circle – indicates the itemsets that finished counting and that do not exceed the support threshold;
- Dashed Box – indicates the itemsets that finished counting and that exceed the support threshold;
- Dashed Circle – indicates the itemsets that not finished counting and that do not exceed the support threshold.

The algorithm is described as follows:

1. First, the empty itemset is marked with a solid box and all the 1-itemsets into dashed circle.
2. After reading one interval of M transactions from database, do the following steps:
    - Check each itemset, in dashed circle. If it exceeds the support threshold, change it from dashed circle to a dashed box.
    - Check each super set of dashed circle. If all the subsets of dashed circle are in solid box or dashed box, then add it into dashed circle.
    - Check each set in dashed circle and dashed box. If it has been counted over all the transactions, change it into solid circle if it is in circle or change it into solid box if it is in box.
3. End of transactions is reached then, go back to the beginning and repeat step 2, until no itemset remains in dashed circle or dashed box.

DIC is effective in decreasing the number of database scans if data are homogeneous. If data isn't homogeneous, DIC algorithm scan database more than apriori and generates false itemsets.

4.3.5 Eclat Algorithm

Eclat is an extension of apriori algorithm and presents different design approach for association rule mining. Eclat uses the vertical database layout and uses the intersection based approach to compute the support of an itemset.

Eclat constructs candidate itemsets using the join step from apriori. Eclat differs from the apriori algorithm in prune step. (Goethals B., 2003). All items in the database are reordered in support ascending order at every recursion step of the algorithm. Eclat counts the supports of all itemsets more efficiently than Apriori.

Eclat is better than the Partition algorithm for using the memory. Because of the item sets reordering process.

The main idea of Eclat is outlined in the following:

```
Algorithm Eclat

F[I] := {}
for all i € I occurring in D do
  F[I] := F[I] ∪ {I ∪ {i}}
  // Create D^i
  D^i := {}
  for all j € I occurring in D such that j > i do
    C := cover({i}) ∩ cover({j})
    if |C| ≥ σ then
      D^i := D^i ∪ {(j,C)}
    end if
  end for
// Depth-first recursion
Compute F[I ∪ {i}]( D^i, σ)
F[I] := F[I] ∪ F[I ∪ {i}]
end for
```

Apriori and Eclat algorithms are used the prefix tree structure (described in section 4.2). The main differences between Apriori and Eclat are traversing to this prefix tree and determining the support of an item set.

Apriori traverses the prefix tree in breadth first order, it first finds the frequent 1 itemsets, then frequent 2 itemsets and so on. (Borgelt C., 2003). Supports of the item set are determined in two steps. Firstly, checking all transaction which contains the item sets or traversing the prefix tree for each item set. Then, incrementing the corresponding item set counters. Eclat traverses the prefix tree in depth first order, it extends an item set prefix until it reaches the limit between frequent and infrequent item sets and then go back to work on the next prefix. Supports of the item set are determined by constructing the list of identifiers of transactions that contain the item set. Lists of transaction identifiers of two item sets are intersecting which differ only by one item and together form the item set currently processed.

4.3.6 FP – Growth Algorithm

Given a transaction database and a minimum support threshold, the problem of finding the complete set of frequent patterns is called the ***frequent pattern mining*** problem.

Fp – tree is a compact data structure. *FP-Growth,* is an algorithm for generating frequent itemsets for association rules. This algorithm compresses a large database into a compact, frequent–pattern– tree (FP tree) structure. Fp – tree structure stores all necessary information about frequent itemsets in a database.

Definition (FP-tree): A frequent pattern tree (or FP-tree in short) is defined below, (Pei J., 2002).

1. the root labeled with "null" and set of items as the children o the root. Frequent item header table which contains items in their frequency descending order.

2. Each node contains of three fields: item-name (holds the frequent item), count (number of tranactions that share that node), and node- link (next node in the FP-tree).

3. Frequent-item header table contains two fields, item-name and head of node-link (points to the first node in the FP-tree holding the item-name).

FP- Growth method only needs two database scans when mining all frequent itemsets. In the first scan, all frequent items are found. Next scan, constructs the first FP – tree which contains all frequency information of the given dataset.

Fp – tree use compact data structure based on the following properties, (Goethals B. 2002).

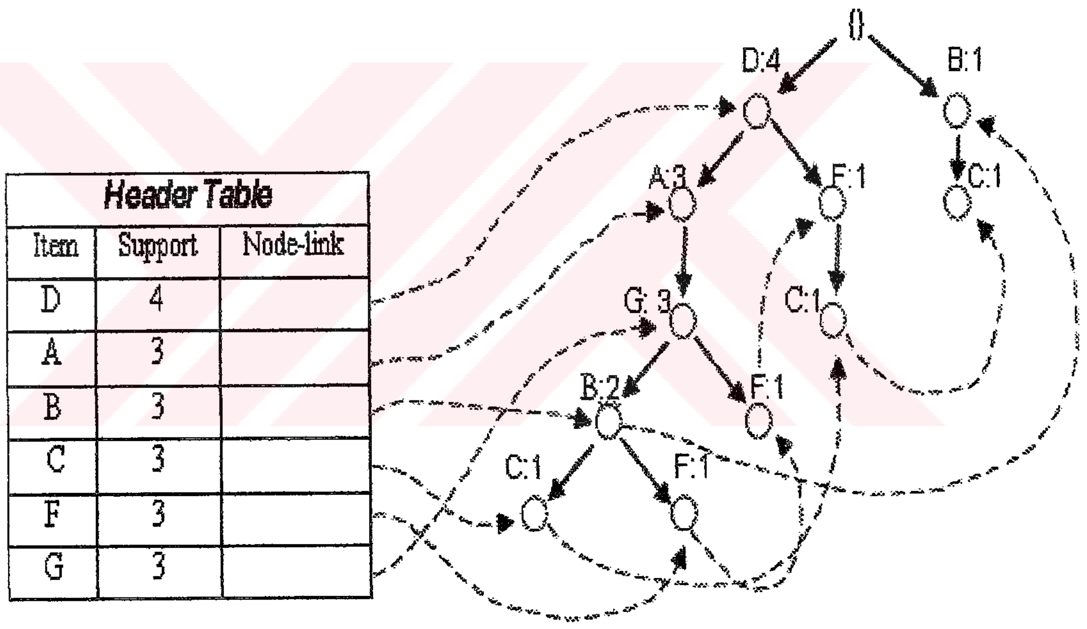
- Frequent pattern generation mining perform one scan of Database to determine the set of frequent items.
- Method needs to store each item in a compact structure, thus, more than two database scan unnessary.
- Each frequent item located in the FP – tree and each node hold items and count of the frequent item.
- Each item have to be sorted in their frequency descanding order. So, tree construction operation performs easily.

```
Algorithm

Procedure FP-growth (Tree, $\sigma$)
{
if Tree contains a single path P
then for each combination (denoted as $\beta$)
     of the nodes in the path P do
      generate pattern $\beta \cup \sigma$ with
               support =minimum support of nodes in $\beta$;
else for each $a_i$ in the header of Tree do
{
      generate pattern $\beta = a_i \cup \sigma$  with
          support = $a_i$.support;

     construct $\beta$ 's conditional pattern base and
     then $\beta$ 's conditional FP-tree $Tree_\beta$;
     if $Tree_\beta = \varnothing$ ;
          then call FP-growth ($Tree_\beta$, $\beta$)
}
}
```

In following example we use the database transaction_DB4 in table 4.2 and we construct FP-tree in figure 4.5 using the construction steps.

Table 4.2 the transaction database transaction_DB4

T_id	Items in transaction	Ordered Frequent items	
1	A, B, C, D, H, G	D, A, G, B, C	Min_support 50 %
2	A, B, D, F, G	D, A, G, B, F	
3	A, D, G, F	D, A, G, F	
4	C, D, F, K, L	D, F, C	
5	B, C, L	B, C	



Header Table

Item	Support	Node-link
D	4	
A	3	
B	3	
C	3	
F	3	
G	3	

Figure 4.5 FP tree for transaction_DB4.

Details of this example as follows:

First, frequent pattern generation mining perform one scan of Database to determine the set of frequent items and itemset count and all infrequent items are removed from the header table. Items are reordered according to count descending order. Then, create the root of a tree labeled with "null". Second, scan the all transactions and construct the FP – Tree. The first transaction form the first branch of the tree {(D: 1), (A: 1), (G: 1), (B: 1), (C: 1)}. For the second transaction, (D, A, G, B, F) ordered frequent item list shares a common prefix (D, A, G, B) with the existing path (D, A, G, B, C), the count of each node which share the same item is incremented by 1, and one new node (C:1) is created. Linked with node of (B:2). For the third transaction (D, A, G, F) ordered frequent item list shares a common prefix (D, A, G) with the existing path. The count of each node which share the same item is incremented by 1, and one new node (F:1) is created. Linked with node of (G:3). The fourth transaction (D, F, C) ordered frequent item list shares a common prefix (D) with the existing path. The count of the node (D) which share the same item is incremented by 1, and (F: 1) is created and linked as a child of (D:4) and (C: 1) is created and linked as a child of (F:1). The last transaction (B, C) form to the construction of the second branch of the tree. (B: 1) is created and linked as a child of the root and (C: 1) is created and linked as a child of (B:1).

4.4 Frequent Closed Itemsets Algorithms

Frequent itemsets algorithms which are Apriori, Eclat, FP-Growth etc. generates very large number of frequent itemsets and rules, these increase the data size, reduces the efficiency and effectiveness of association mining to find useful rules.

Frequent closed itemsets algorithms are an alternative, proposed [by Pasquier et al.]. These algorithms have used different itemset generation approach. Instead of mining the complete set of frequent itemsets and their associations, association mining only needs to find frequent closed itemsets and their corresponding rules.

Frequent closet itemsets can be divided into two step-process:

- Find all frequent closed itemsets which have the support greater than the minimum predetermined *min_sup* in the transaction database.
- Generate all association rules X⇒Y on the frequent closed itemsets which have confidence greater than predetermined *min_conf*.

Frequent closed itemsets algorithms presents a new perspective in association rule mining. Frequent closed itemsets has the same power as mining the complete set of frequent items. In this section the most popular frequent closed itemsets algortihms are described (Closet and Charm algorithms.)

4.4.1 Closet Algorithm

Closet is a well known algorithm in association mining to generate frequent closed itemsets. Closet bases on the definition of "X and Y be two itemsets, and sup(X) = sup(Y). Y is not a closed if Y⊂ X" (Pasquier N. et al., 1999).

Closet algorithm has the same power as frequent association mining algorithms. Closet reduces redundant rules to be generated. The goals of the closet algorithms are improve efficiency and also effectiveness of mining process (Pei J. et al., 2000).

```
Algorithm CLOSET

Input: Transaction database TDB and support threshold;
Output: The complete set of frequent closed itemsets;

Method:

Initialize.
     Let FCI be the set of frequent closed itemset. Initialize FCI =
     ∅
Find frequent items.
     Scan  transaction  database  TDB,  compute  frequent  item  list
     f_list.
Mine frequent closed itemsets recursively.
     Call CLOSET(∅, TDB, f_list, FCI).



Subroutine CLOSET(X, DB, f_list, FCI)
Parameters:
      X: is the frequent itemset.
      DB: X-conditional database, which is a subset of transactions
      in
          TDB containing X.
      f_list: frequent item list of DB
      FCI: The set of frequent closed itemsets already found
```

```
CLOSET(X, DB, f_list, FCI)
  {
    Extract a set (Y) of items appearing in every transaction of
  DB, insert X∪Y to FCI, if it is not a subset of some itemset in
  FCI with the same support;

  Build FP-tree for DB, items in Y are excluded.

  Directly extract frequent closed itemsets from FP-tree.

    ∀i∈ rest of f_list, form conditional database DB|i and compute
  local frequent item list f_list i

    ∀i∈ rest of f_list, call CLOSET(iX, DB|I, f_listi, FCI), if iX
  is not a subset of any frequent closed itemset in FCI with the
  same support.

  }
```

In following example we use the database transaction_DB5 in table 4.3 and we construct Mining Frequent Closed Itemsets in figure 4.6 using the construction steps.

Table 4.3 transaction_DB5

Transaction_id	Items in transaction	Ordered frequent item set
1	A, C, D, E, F	C, E, F, A, D
2	A, B, E	E, A
3	C, E, F	C, E, F
4	A, C, D, F	C, F, A, D
5	C, E, F	C, E, F

Steps of Algorithm CLOSET;

Find Frequent Items: Scan Transaction_DB5 to find the set of frequent items and construct a frequent item list, called f_list. Minimum support =2. In our example transaction database has f_list = < C : 4, E : 4, F : 4, A : 3, D : 2 >, where the items are sorted in support descending order.

Divide search space: Divide Transaction_DB5 into non-overlap subsets (conditional database) based on the F_list. Ones conditional database containing item D, ones containing item A but no D, the ones containing item F but not A and D and so on.

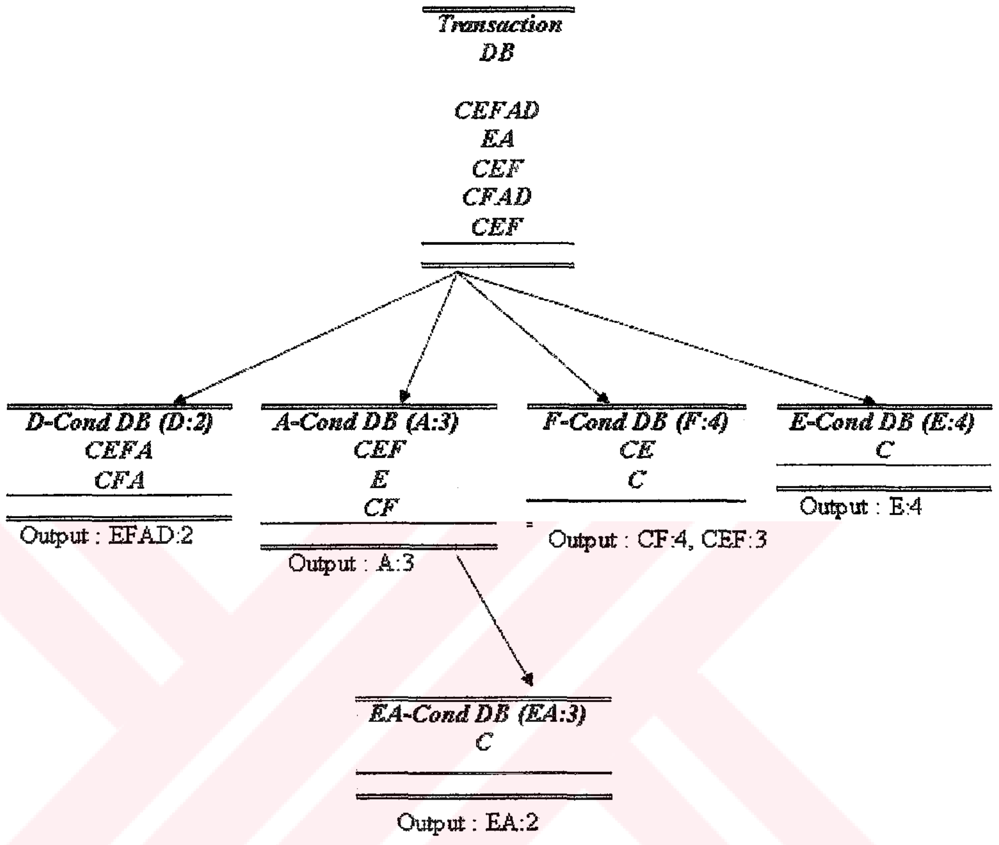


Figure 4.6 Mining Frequent Closed Itemsets using transaction_DB5.

Find subsets of frequent closed itemsets (F.C.I) from condition DBs. Details of this example as follows:

Algorithm searches closed itemsets for each conditional DBs. Closet algorithm found the conditional databases in figure 4.6 in our example database. The process of finding frequent closed itemsets

First, {D}-conditional database is used which has contains all the transactions having {D}; {CEFA, CFA} to find the closed frequent items. The support of {D} is 2. Every transaction containing {D} also contains {C}, {F} and {A}. Consequently, {CFAD: 2} is a frequent closed itemset. {E} appears only once so, it's infrequent.

Second, A-conditional database is used which has contains all the transactions having {A} but no {D}; {CEF, E, CF} to find the closed frequent items. The support of {A} is 3. There is no any item appearing in all transactions so, {A} is frequent closed itemset. Project the conditional database {A} to find remaining frequent closed itemsets. Algorithm finds items (FA, CA, and EA). {CA} is not frequent closed itemset due to the {CA} is a subset of {CFAD} and sup (CA) = sup (CFAD). {FA} is not frequent closed itemset due to the {FA} is a subset {CFAD} and sup (FA) = sup (CFAD). Consequently, {EA: 2} is a frequent closed itemset.

Third, A-conditional database is used which has contains all the transactions having {A} but no {D}; {CEF, E, CF} to find the closed frequent items. The support of {A} is 3.

Next, F-conditional database is used which has contains all the transactions having {F} but no {D} nor {A}; {CE, C} to find the closed frequent items. In this process, frequent closed itemsets {CF} and {CEF} are found.

Finally, E-conditional database is used which has contains all the transactions having {E} but no {F}, {D} nor {A}; {E} to find the closed frequent items. In this process, frequent closed itemset{E} is found.

In summary, the set of frequent closed itemsets found,
{ACDF: 2, A: 3, AE: 2, CF: 4, CEF: 3, E: 4}.

4.4.2 Charm Algorithm

Charm is an another well known algorithm in association mining to generate frequent closed itemsets. Charm algorithm has the same power as frequent association mining algorithms. Charm uses a novel search method that skips many levels to quickly find the closed frequent itemsets. Charm uses two-pronged pruning strategy.

"It prunes candidates based not only on subset infrequency as do all association mining methods, i.e., no extensions of an infrequent are tested. it also prunes candidates based on non-closure property, i.e., any non-closed itemset is pruned." (Zaki M. J., Hsiao C.J., 2002).

Charm uses no internal data structures, it uses basic set operations which are union of two itemsets and intersection of two transactions lists. Nevertheless, It makes too few database scans to find longest closed frequent set.

All previous association mining algorithms uses itemset space. But, charm uses both the transactionset space and itemset space to find frequent closed itemsets. This property allows charm to use a novel search methods that skips many levels to quickly find itemsets.

Charm algorithm uses the following itemset and transaction sets properties. (M. J. Zaki, 2000)

Property 1.

If $t(X_i) = t(X_j)$, then $t(X_1 \cup X_2) = t(X_1) \cap t(X_2) = t(X_1) = t(X_2)$. So, every occurrence of X_2 replaced with X_1.

Property 2.

If $t(X_1) \subset t(X_2)$, then $t(X_1 \cup X_2) = t(X_1) \cap t(X_2) = t(X_1) \neq t(X_2)$. So, every occurrence of X_1 replaced with $X_1 \cup X_2$. If X_1 occurs in any transaction, then X_2 always occurs too.

Property 3.

If $t(X_1) \supset t(X_2)$, then $t(X_1 \cup X_2) = t(X_1) \cap t(X_2) = t(X_1) \neq t(X_2)$. So, every occurrence of X_2 replaced with $X_1 \cup X_2$. If X_2 occurs in any transaction, then X_1 always occurs too.

Property 4.

If $t(X_1) \neq t(X_2)$, then $t(X_1 \cup X_2) = t(X_1) \cap t(X_2) \neq t(X_1) \neq t(X_2)$. So, nothing can be eliminated.

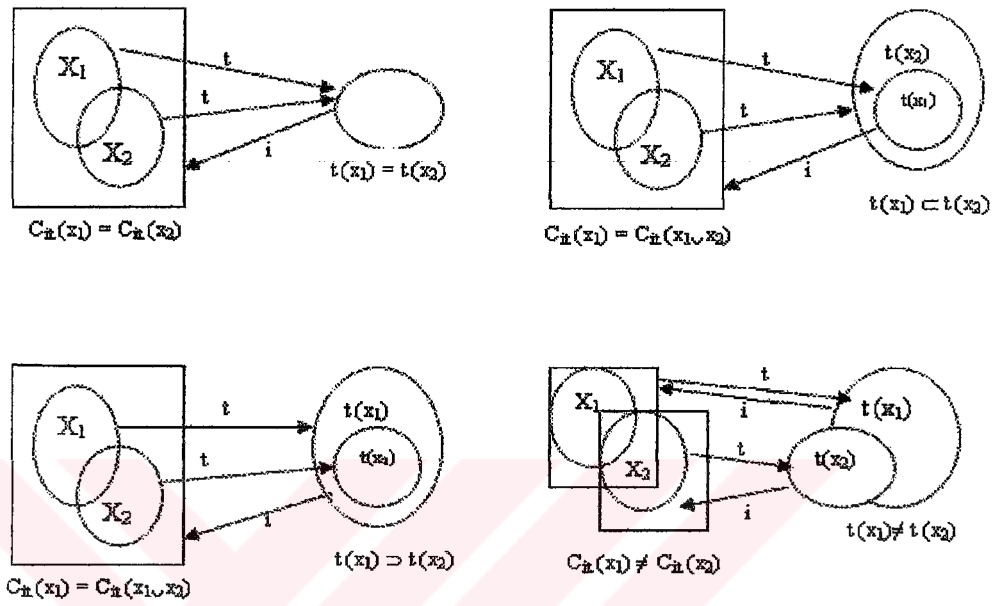


Figure 4.7 Properties of Itemset & Transaction Sets

Figure 4.7 illustrates the four basic properties. The main idea of Charm is outlined in the following:

```
CHARM ($\partial \subseteq I \times T$, minsup):
Nodes = {$I_j \times t(I_j)$ : $I_j \in I \wedge |t(I_j)| \geq minsup$}
CHARM-EXTEND (Nodes, C)


CHARM-EXTEND (Nodes, C)
for each $X_j \times t(X_i)$ in Nodes
    NewN = $\emptyset$ and X = $X_i$

    for each $X_j \times t(X_j)$ in Nodes, with f(j)>f(i)
        X = X $\cup$ $X_j$ and Y = $t(X_i) \cap t(X_j)$
        CHARM-PROPERTY(Nodes, NewN)
    if NewN = $\emptyset$ then CHARM-EXTEND (NewN)
    C = C $\cup$ X //if X is not subsumed
```

```
CHARM-PROPERTY (Nodes, NewN)

if ( |Y| ≥ minsup) then
   if  t(X_i) = t(X_j) then //Property 1
      Remove X_j from Nodes
      Replace all X_i with X
   else if t(X_i) ⊂ t(X_j) then //Property 2
      Replace all X_i with X
   else if t(X_i) ⊃ t(X_j) then //Property 3
      Remove X_j from Nodes
      Add X x Y to NewN
   else if t(X_i) ≠ t(X_j) then //Property
      Add X x Y to NewN
```

Charm uses a vertical data format which has been used for association mining. Vertical data format also used in Partition and Eclat algorithms. Charm scans database once to find the frequent pairs of items. In this scan, charm algorithm constructs the index array that stores the transaction set for each item. If index array not available additional database scan performed.

4.5 Parallel & Distributed Algorithms

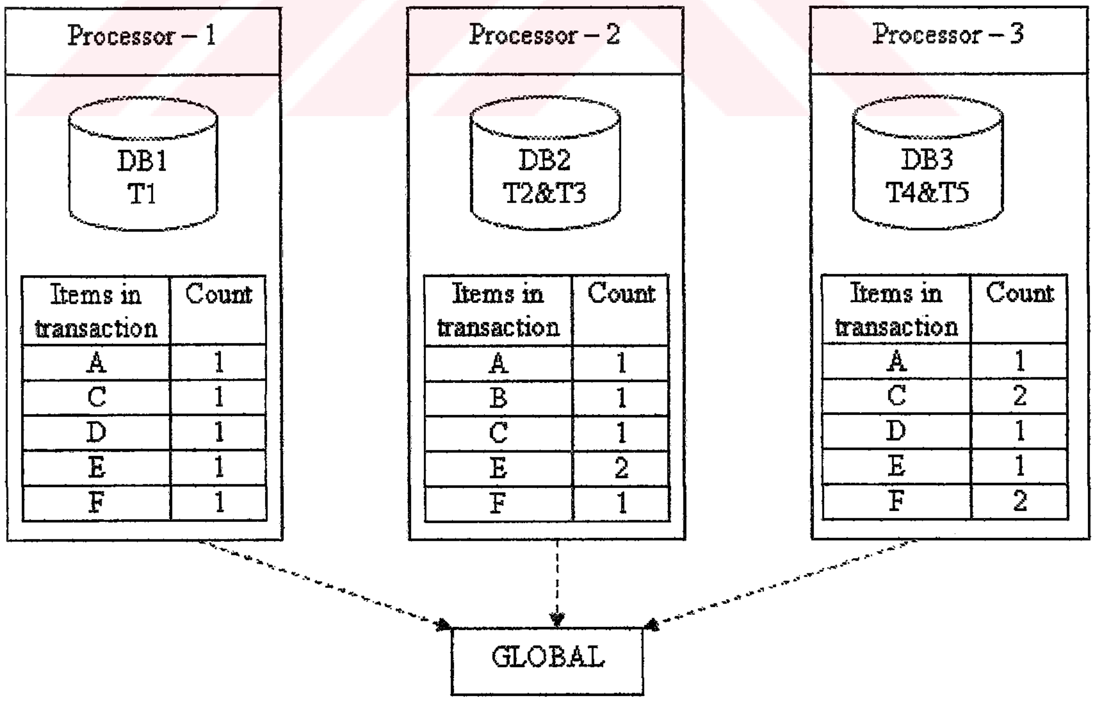
The most popular parallel and distributed algorithms are based on the Apriori algorithm. Communication cost and synchronization are the main problem for parallel algorithms. (Zaki. M.J., 1999)

Distributed and shared memory are two popular approach. Properties of the distributed and shared memory can be summarized as follows. Communication optimization is the goal for distributed memory (DMM). Minimize the synchronization is the goal for shared memory (SMP). Data decomposition is important for distributed memory but not for shared memory. Parallel I / O is free for DMM's but shared encounter with some problems.

The main objectives for distributed memory are finding good data decomosition among nodes and minimizing communication. The main objective for shared memory is obtain good data locality.

Data mining parallelism aproach used two main paradigm which are task parallelism and data parallelism.

In data parallelism, dataset is partitioned among N processor. Each processor works on its local partition of the dataset and they responsiple for computing local support counts of the candidates. After local computation completed, all the processor counting global support count of the candidates. The data parallelism is illustrated in Figure 4.8 using the data in Table 4.3. The five transactions are partioned. Three processor used for these transactions and local support counts are computed on each processor. Transaction 1 computed on processor 1. Transaction 2 and 3 computed on processor 2. transaction 4 and 5 computed on processor 3.

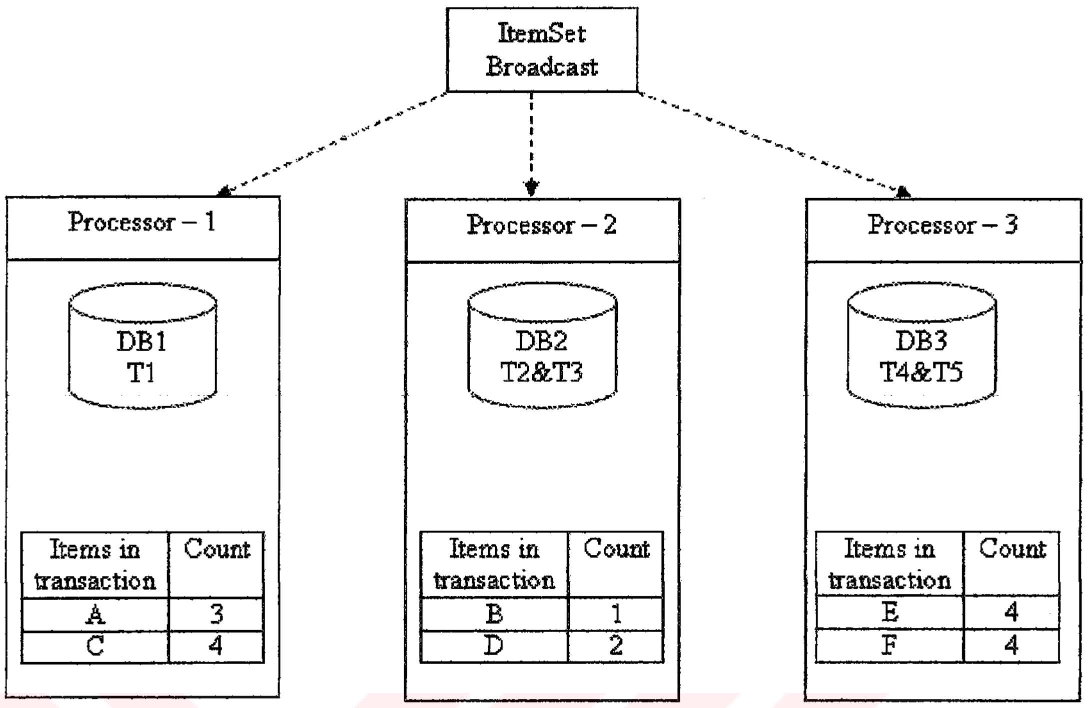


Items in transaction	Count
A	1
C	1
D	1
E	1
F	1

Items in transaction	Count
A	1
B	1
C	1
E	2
F	1

Items in transaction	Count
A	1
C	2
D	1
E	1
F	2

Figure 4.8 Data Parallelism



Items in transaction	Count
A	3
C	4

Items in transaction	Count
B	1
D	2

Items in transaction	Count
E	4
F	4

Figure 4.9 Task Parallelism

In task parallelism, N processor fulfil different computations independently. Each processor works on its local itemsets which are partition of the dataset. Candidate sets are partitioned. Each processor takes a candidate set and computes its support counts. Each processor is responsible for computing global support counts of only its candidates. Database partitioning and itemset sharing are two main process for task parallelism. The task parallelism is illustrated in Figure 4.9 using the data in Table 4.3. The five transactions are partioned. Three processor used for these transactions and global support counts are computed on each processor using the relevant candidate set.

CHAPTER FIVE

IMPLEMENTATION and EXPERIMENTATION RESULTS

This chapter describes the design and implementation of Association Rule Mining algorithms. We implemented the most common association algorithms to mine frequent itemsets which are, Apriori, Eclat, FP- Growth, Partition and DIC (Dynamin Itemset Counting), in Java platform. The experiments were performed on a Pentium PC. Detailed explanation of the system and experimentation results are presented in the following sections. Later on, algorithms are compared and explain the design and details of our new approach.

5.1 Introduction

Most of association rule algorithms generate Association Rule in two steps.

- ✓ Find all frequent item sets;
- ✓ Use frequent item sets to generate strong rules having minimum confidence.

Much of the research effort in association rule algorithms has been related to improving the efficiency of this first step. Therefore, we tested running time of the algorithms for the first step. All of the algorithms generates frequently set and they flushed to a flat file. For the second step, our algorithm works and generates the association rules from the flat file which contains the frequently itemsets.

5.2 Implemantation Details

We implemented the most common association algorithms to mine frequent itemsets which are, Apriori, Eclat, FP- Growth, Partition and DIC (Dynamin Itemset Counting), in Java. Figure 5.1 and figure 5.2 shows our program interface. In figure 5.1, there is three inputs which are dataset, support factor and confidence factor. Support factor is used for to find frequent itemsets and confidence factor is used for to generate association rules. Figure 5.3 shows our program outputs for Apriori algorithm. For instance;

`4428 ==> 170 180 184 4434 4940______%92.62` means,

All transactions which contain item {4428} also contain itemsets {170, 180, 184, 4434, 4940} with confidence %92.62.

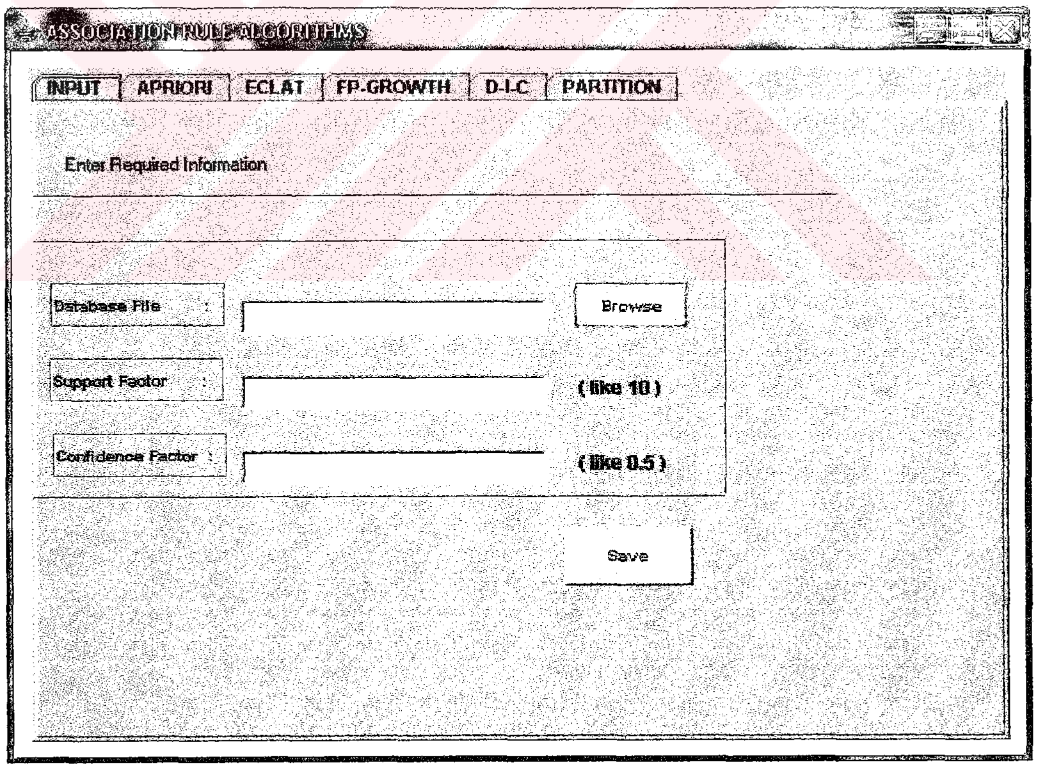
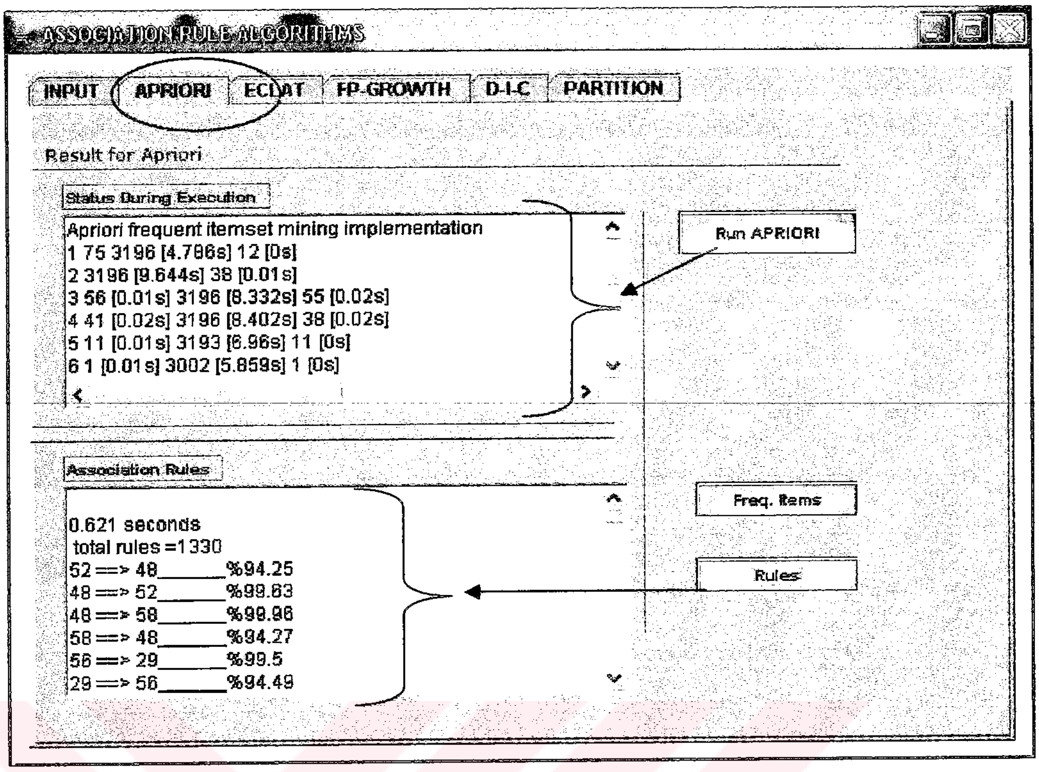


Figure 5.1 User Interface -1



```
Apriori frequent itemset mining implementation
1 75 3196 [4.786s] 12 [0s]
2 3196 [9.644s] 38 [0.01s]
3 56 [0.01s] 3196 [8.332s] 55 [0.02s]
4 41 [0.02s] 3196 [8.402s] 38 [0.02s]
5 11 [0.01s] 3193 [6.96s] 11 [0s]
6 1 [0.01s] 3002 [5.859s] 1 [0s]
```

```
0.621 seconds
 total rules =1330
52 ==> 48______%94.25
48 ==> 52______%99.63
48 ==> 58______%99.96
58 ==> 48______%94.27
56 ==> 29______%99.5
29 ==> 56______%94.49
```

Figure 5.2 User Interface -2

```
4428 ==> 170 180 184 4434 4940______%92.62
180 4940 ==> 170 184 4428 4434______%95.19
180 4434 4940 ==> 170 184 4428______%97.58
180 4434 ==> 170 184 4428 4940______%94.81
180 184 4940 ==> 170 4428 4434______%95.19
180 184 4434 4940 ==> 170 4428______%97.58
180 184 4434 ==> 170 4428 4940______%94.81
180 184 4428 4940 ==> 170 4434______%96.88
180 184 4428 4434 4940 ==> 170______%99.29
180 184 4428 4434 ==> 170 4940______%96.48
180 184 4428 ==> 170 4434 4940______%94.15
180 184 ==> 170 4428 4434 4940______%92.5
180 4428 4940 ==> 170 184 4434______%96.88
180 4428 4434 4940 ==> 170 184______%99.29
180 4428 4434 ==> 170 184 4940______%96.48
180 4428 ==> 170 184 4434 4940______%94.15
180 ==> 170 184 4428 4434 4940______%92.5
4940 7062 ==> 180 184 4428 4434______%94.38
4940 ==> 180 184 4428 4434 7062______%93.57
4434 4940 7062 ==> 180 184 4428______%96.7
4434 4940 ==> 180 184 4428 7062______%95.89
4434 7062 ==> 180 184 4428 4940______%93.9
4434 ==> 180 184 4428 4940 7062      %93.12
```

Figure 5.3 Output from the Apriori

5.3 Data Sets

We choose real and synthetic datasets for testing the performance of the algorithms. All datasets are taken from the source Web_2. Table 5.1 shows the characteristics of the real and synthetic datasets used in our experiments. It shows the number of items, average tranaction length, number of transactions and size for the each datasets.

For all experiments, we used four datasets with different characteristics. Thus, the advantages and disadvantages of the algorithms can be observed. We used three real datasets which are Chess, Mushroom and Pumsb. We used one synthetic dataset which is T10I4D100K.

	Chess	**Mushroom**	**Pumsb**	**T10I4D100K**
# Items	76	120	7117	1000
Avg.Record Length	37	23	74	20
# Records	3196	8124	49046	100000
DB Size	335 kb.	558 kb.	16289 kb.	3928 kb.

Table 5.1 DataSet Characteristics

Chess : Chess dataset is derived from the chess game. It contains real entries. There are 3196 transactions in it, while each transaction has average 37 items. Chess is very dense dataset. It produces many long frequent itemsets even for with high values of support.

Mushroom : Mushroom dataset contains characteristics of various species of mushrooms. It contains real entries. There are 8124 transactions in it, while each transaction has average 23 items. It produces many long frequent itemsets even for with high values of support.

Pumsb : Pumbs file which is Public Use Microdata Samples. Its contains actual census entries which contains a five percent sample of the state the file repsesents.

There are 49046 transactions in it, while each transaction has 74 items. It is a dense dataset with many long frequent itemsets.

T10I4D100K : T10I4D100K is a synthetic datasets, which have been used as benchmarks for testing association mining algorithms. Dataset "T10I4D100K" contains an average 20 transaction size, an average size of the maximal potentially frequent itemsets of 4. There are 100000 transactions in it. It produces many long frequent itemsets even for with high values of support.

5.4 Generating Association Rules

Use frequent item sets to generate strong rules having minimum confidence is the second step for association mining. In our application, we implemented a new method for generating association rules.

This method used the flat files which contains frequent itemsets generated from the algorithms. It read the frequent itemsets and constructed a hash tree for each line. This means that, each itemsets located in the tree with using the our algorithm and all of the rules computed for this part. tree. In hash tree, there are nodes for holding frequent itemset and a vektor pointing the next node.

Generating rule Algorithm illustrated in figure 5.4. This algorithm works as follows;

```
read Itemset from the file,
construct hash tree,
generate rules with using the recursion function,
read next itemset.
```

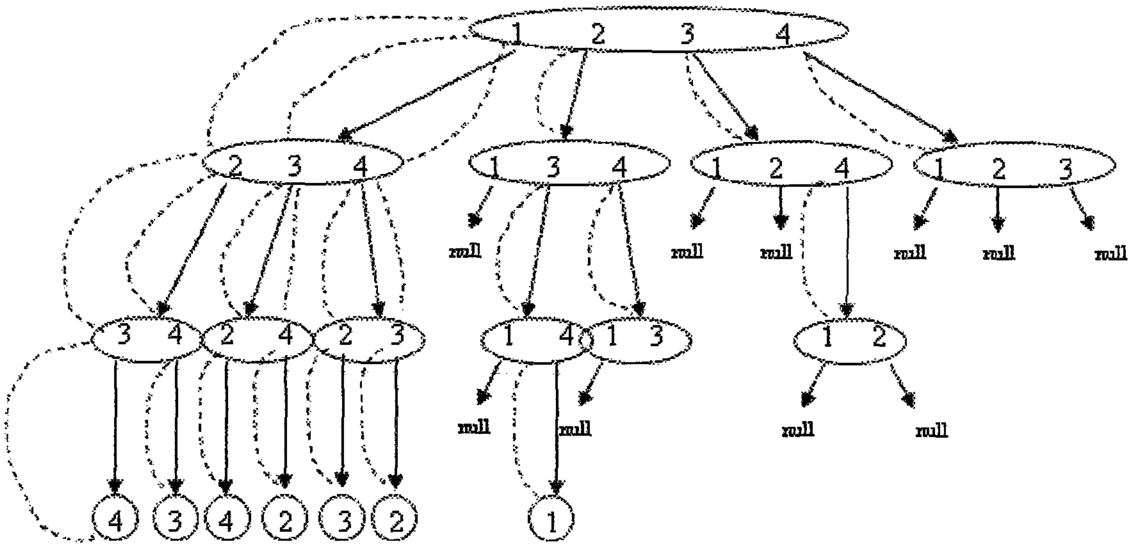


Figure 5.4 Hash Tree for Rules Generation

5.5 Experiments

In this section, we report our performance study of the five algorithms for mining frequent itemsets: Apriori, Eclat, Fp-Tree, DIC and Partition.

All the experiments are performed on a 1400Mhz. Pentium PC with 256 Mb. RAM, running on Microsoft Windows 2003 Server. We tested the five algorithms on various datasets which are described in section 5.3.

The performance meausure was the execution time of the algorithms and their memory requirements on the datasets with the different minimum support values.

The x – axis of the result figures shows the support factor and y – axis shows the process time in seconds.

The detailed values of these results are listed in Tables A.1 through A.4 in Appendix A.

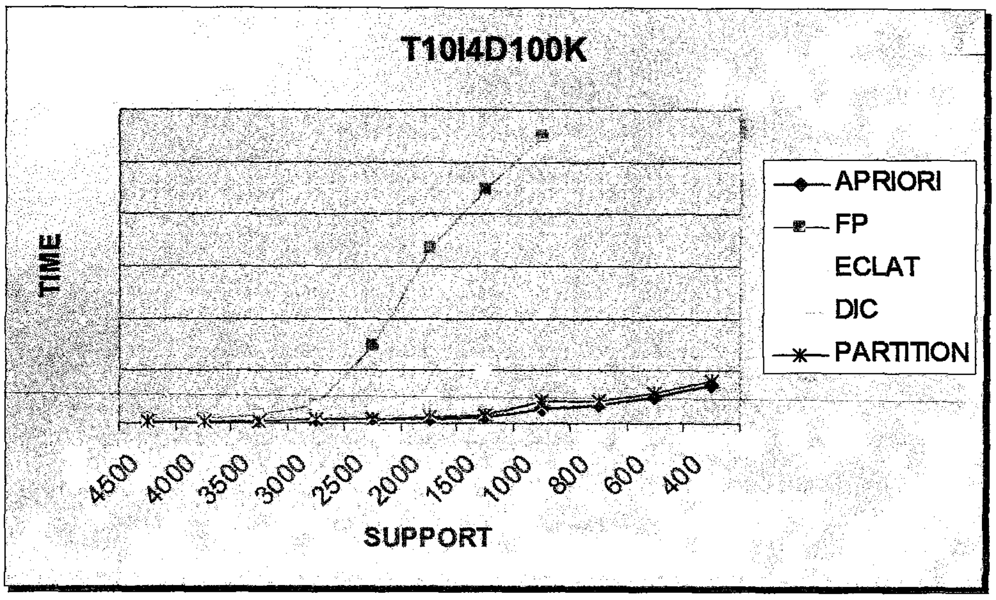


Figure 5.5 Running Time of the Algorithms for generating frequent itemsets for "T10I4D100K"

Figure 5.5 shows the results we obtained from experimentation for the dataset "T10I4D100K".

Apriori emerged 2001 frequent items set and 4194 association rules with minimum support 400.

Graphic says that, Apriori gave very good performance over the others. When the minimum support is large, all algorithms finish their process at the same time. When the minimum support has small value, FP-Tree and Eclat algorithms gave bad performance.

DIC, Partition and Apriori algorithms have the same performance. Process time of this algorithms are same for this kind of data.

- ✓ Apriori > DIC > Partition > Eclat > Fp – Tree

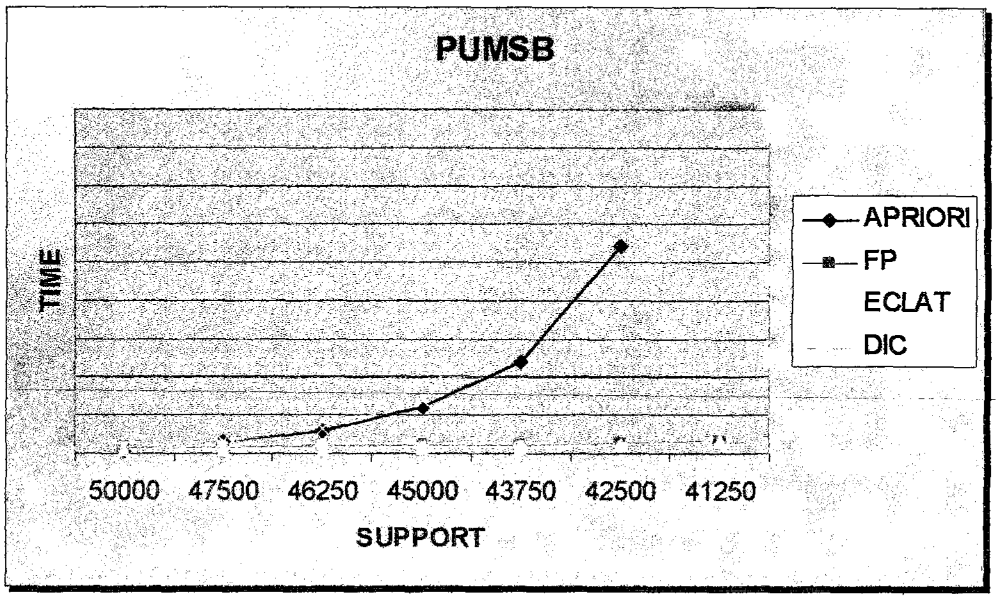


Figure 5.6 Running Time of the Algorithms for generating frequent itemsets for "Pumsb"

Figure 5.6 shows the results we obtained from experimentation for the dataset "Pumsb".

Eclat emerged 10531 frequent items set and 544998 association rules with minimum support value 42500.

Graphic says that, FP – Tree and Eclat gave very good performance over the others. Eclat is a little good over FP – Tree. When the minimum support is large, all algorithms finish their process with similar time. When the minimum support has small value, FP-Tree and Eclat algorithms gave very good performance. Because the density of the data. DIC gave the worst performance for both high and low minimum support.

✓ Eclat > FP – Tree > Apriori > DIC

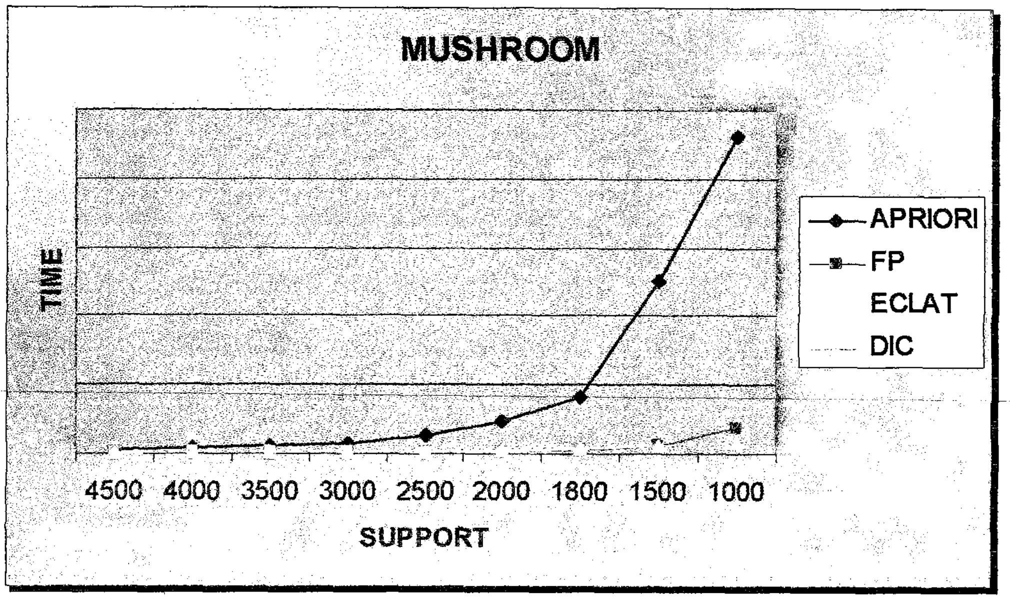


Figure 5.7 Running Time of the Algorithms for generating frequent itemsets for "Mushroom"

Figure 5.7 shows the results we obtained from experimentation for the dataset "Mushroom".

At this level 123277 frequent items sets are produced and above 10 million rules are generated from the frequent items sets.

Graphic says that, FP – Tree and Eclat gave very good performance over the others. Eclat is a little good over FP – Tree. When the minimum support has small value, FP-Tree and Eclat algorithms gave very good performance. An interesting point that FP has almost the same performance with Eclat until 1000 min support. After this point Fp-tree process time is begining to grown up. DIC is slower than the others.

✓ Eclat > FP – Tree > Apriori > DIC

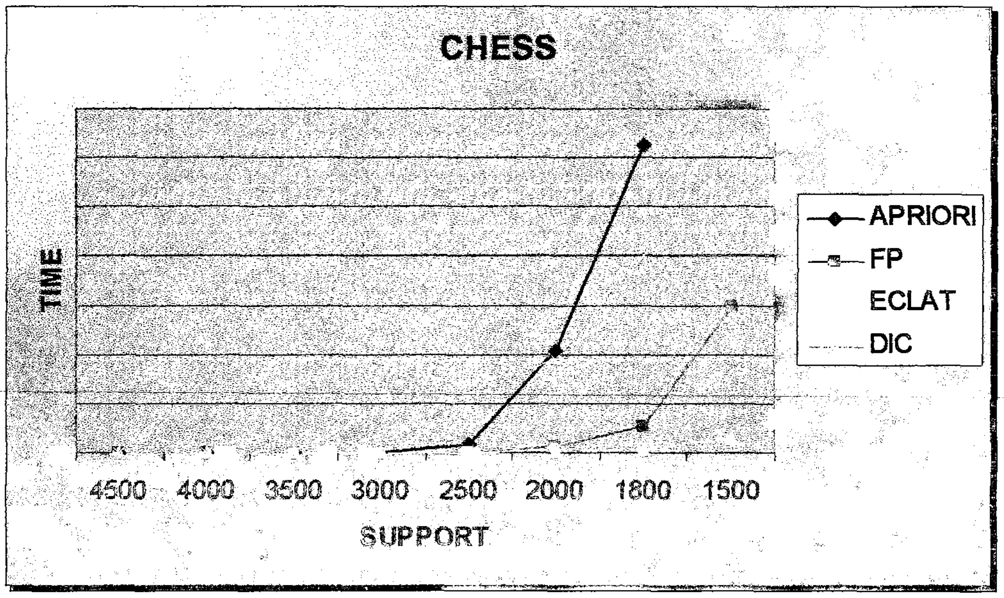


Figure 5.8 Running Time of the Algorithms for generating frequent itemsets for "Chess"

Figure 5.8 shows the results we obtained from experimentation for the dataset "Chess".

At this level 2076329 frequent items sets are produced and above 10 million rules are generated from the frequent items sets.

Graphic says that, Eclat gave very good performance over the others. FP – Tree gave the same results until support value 2000. After this point Fp-tree process time is begining to grown up rapidly. When the minimum support is large, FP-Tree and Eclat algorithms gave very good performance. DIC algorithm gave the worst performance. The reason could be that the density of the data and frequent itemsets are very large.

✓ Eclat > FP – Tree > Apriori > DIC

Frequent itemsets algorithms generate very large number of itemsets and association rules. Figures 5.5 through 5.8 show performance curves for the five algorithms that generate frequent itemsets. For all of the real – datasets, the results show that Eclat works better than the others. FP – Tree and Eclat obtains same characteristic results for the real – datasets. DIC (Dynamic Itemset Counting) is much slower than every other algorithm for the real – dataset. On the only dataset on which is T10I4D100K, Apriori is faster than the others. Partition and DIC performs same results for the synthetic – dataset.

The results show that Apriori cannot be run very effective than the Eclat and FP – Tree. The performance differences of Eclat and FP-growth are negligible. In small size of datasets both run too fast. Apriori on the other hand runs too slow because each transaction contains density data. Another remarkable result is that Apriori performs better than the others for synthetic – dataset.

In the result of experiment for data T10I4D100K, Eclat needs 10000 KB memory while FP needs 36000 KB. And both of them take over 1000 (seconds) to accomplish the process. Apriori and DIC takes 132 s and 1750 KB for the same purpose.

In the result of experiment for data Mushroom, Apriori and DIC pass under 1000 minumum support, it began to suffering. They needs at least 11000 KB memory and more than 1000 seconds to accomplish the test. Eclat is the best algorithm for the this data. At minumum support 1000, FP wasted 4000 KB and 82 seconds for the process. Eclat wasted 2000 KB memory reqirement and 41 s process time. At this level 123277 frequent items sets are produced and above 10 million rules are generated from the frequent items sets.

In the result of experiment for data Chess, Apriori and DIC are not an effective algorithms for chess. Apriori and DIC could not go under 1800 minimum support. Their memory requirement reached to 50000 KB and it took 5421 seconds to complete. Eclat and FP gave acceptable outcomes with this data. Fp took 455 s and 4000 KB while Eclat took 65 s and 1500 KB in this test. At this level 2076329 frequent items sets are produced and above 10 million rules are generated from the frequent items sets.

In the result of experiment for data Pumsb, Apriroi and DIC are inefficient. Because they took 5410 seconds with 42500 support factor. Eclat reached 20000 KB memory at 45000 support factor. But FP got over this process with 1500 KB. Total 10531 frequent items set and 544998 association rules with minimum support value 42500.

5.6 A New Algorithm for Association Rule

In this section, we describe our new algorithm to mine association rules. Our algorithm is a combination of Apriori, which is the most popular mining algorithm, and set operations. Principles of set operations which are intersection and union are used. These principles are related to lattice tree. In lattice tree, there are nodes holding frequent itemsets and transactions containing related itemsets.

In order to construct (n+1)-itemsets, frequently n-itemsets are used. Hence, intersection operation is employed between the transaction sets.

For example;

Frequent 1-itemset {A} is in the transactions which have transaction_id 1, 3 and 5.

Frequent 1-itemset {B} is in the transactions which have transaction_id 2, 3 and 5.
$\Rightarrow \{A\} \cup \{B\} = t\,(135) \cap t\,(235)$
$\Rightarrow \{AB\} = t\,(35)$

If the result is greater than minimum support, it will be joined to lattice tree. If the result is lower than minimum support, it will be pruned off.

The main idea of Our Algorithm is outlined as follows;

```
$L_k$ : Frequent itemset of size k
$C_k$ : Candidate itemset of size k
$L_1$ = frequent 1- items;
Generate frequent 2- itemsets using lattice Tree
k=2;
For each frequent k- itemset nodes in lattice
$C_{k+1}$ = new node; // Generated from $L_k$
    For each $X_i$ in Nodes
     For each $X_j$ in Nodes //where j>i
       $C_{k+1}$ =$T(X_i) \cap T(X_j)$
If $C_{k+1}$ > min_support then {// Add $C_{k+1}$ in lattice
         (Node.item       = $X_i \cup X_j$)
         (Node.transaction =$T(X_i) \cap T(X_j)$) }
k++;
End
```

This algorithms work as follows;

- ✓ Scan database and find frequent 1-itemsets, at the same time obtain transaction sets which includes the itemsets.
- ✓ Construct Lattice tree, in order to generate all frequent 2-itemsets.
- ✓ During the second step, prune off the nodes whose node count is lower than minimum support.
- ✓ Find frequent itemsets by using lattice tree. Consequently, for each frequent 3, 4 ..., n–itemset, scan the database to approve the consistence of the itemsets.
- ✓ Finally, itemsets are used to generate strong rules having minimum confidence in the lattice tree.

Table 5.2 transaction_DB6 (Min_support 50 %)

T_id	Items in Transaction	Frequent items
1	A, B, C, D,G	D, A, B, C
2	A, B, D, F, C	D, A, B, C
3	A, C, D	D, A, C
4	C, D, F, K, L	D, C
5	B, C, L	B, C

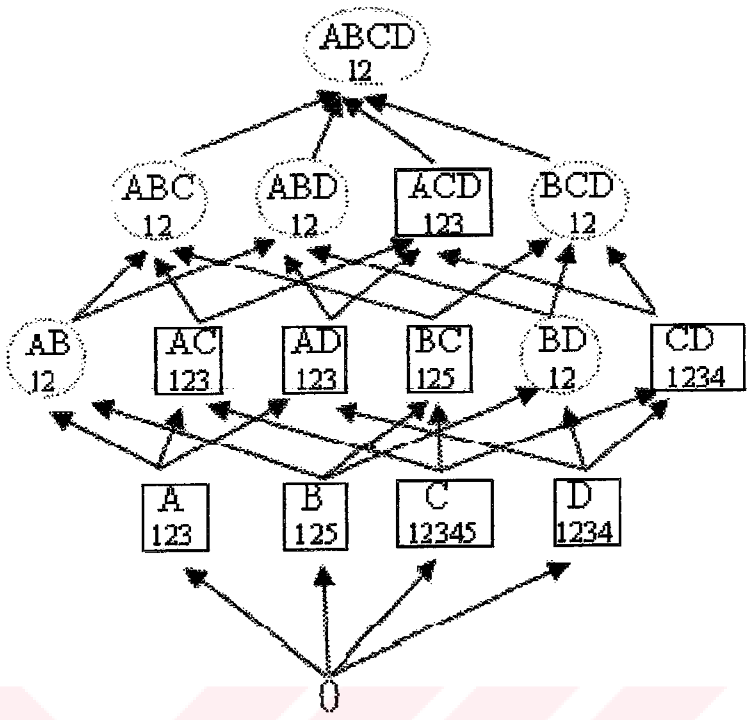


Figure 5.9 Full Lattice Tree for transaction_DB6

Figure 5.9 shows how this algorithm works in sample database in table 5.2 (transaction_DB3). If the intersection result of the frequent nodes exceeds the support thresholds, mark the node with a square. Otherwise, mark node with a dotted circle. In this example, firstly, find frequent 1-itemsets. Next, use intersection operation to find frequently 2-itemset.

$$\left.\begin{array}{l}\{A\}\text{ in t }(123)\\ \\ \{C\}\text{ in t }(12345)\end{array}\right\} \Rightarrow \quad \{A\} \cup \{C\} = t(A) \cap t(C) \Rightarrow \{AC\} = 123 \cap 12345 = 123$$

Intersection principles are used to find frequent itemsets, then, for each frequent 3, 4 …, n –itemset, we scan the database to approve the consistence of the itemsets.

5.7 Summary of Algorithms

Table 5.3 presents the summary of algorithms comparisons. We compared the algorithms in terms of data structures, search methods and database scan evaluation.

Table 5.3 Summary of Algorithms Comparisons

Algorithm	Data Structure	Search Method	Number of Database Scan
Apriori	*Hash*	*Bottom – Up*	*N*
AprioriTID	*Hash*	*Bottom – Up*	*N*
DIC	*Prefix Tree*	*Bottom – Up*	$\leq N$
Partition	*Hash + Partition*	*Bottom – Up*	*2*
Clique	*None*	*Bottom – Up*	≥ 3
DHP	*Hash*	*Bottom – Up*	*N*
AprioriHybrid	*Hash*	*Hybrid*	*N*
MaxClique	*None*	*Hybrid*	≥ 3
MaxEclat	*None*	*Hybrid*	≥ 3
Eclat	*None*	*Top – Down*	≥ 3
FP-Growth	*FP – Tree*	*Top – Down*	*2*
NEW	*Hash + Lattice tree*	*Bottom - Up*	$< N$

CHAPTER SIX
CONCLUSION

Association rule mining explores for interesting relationships among items in a given data set. An objective of association rule mining is to develop a systematic method using the given data set and finds relationships between the different items.Goal of association rule is finding associations among items from a set of transactions which contain a set of items.

There are many new and efficient algorithms have been developed in association mining. Additionally, new proposals offered for algorithms that improve run time for generating association rules or frequent itemset. The efficiency of a mining algorithm is a very important point for data mining.

Our work, the work described in this thesis, has focused on explaining the fundamentals of association mining and analyzes implementations of the well known association rule algorithms. Study focus on algorithms Apriori, Eclat, FP-Growth, Partition and Dynamic Itemset Counting. Most common association rule algorithms described and systematized in this study. Common principles and differences between the algorithms have been shown. Pros and cons of these algorithms are thoroughly investigated. At the same time, we described a new approach for association rule mining. This approach is based on lattice tree and principles of set operations.

Association Rule algorithms have used different itemset generation approach. Classical algorithms have used frequent itemset generation technique. The new generation algorithms have used closed frequent itemset generation and maximal frequent itemset generation technique. To better understand the performance characteristics of association rule algorithms, we benchmarked five well-known association rule algorithms (Apriori, Eclat, FP-growth, Partition and DIC) using the four datasets with different characteristics. Thus, the advantages and disadvantages of the algorithms can be observed, the performance meausure was the execution time of the algorithms and their memory requirements on the datasets with the different minimum support values.

The results show that Apriori cannot be run very effective than the Eclat and FP - Tree. The performance differences of Eclat and FP-growth are negligible. In small size of datasets both run too fast. Apriori on the other hand runs too slow because each transaction contains density data. Partition and DIC performs same results. DIC (Dynamic Itemset Counting) is much slower than every other algorithm for the real - dataset.

REFERENCES

Agrawal R. & Srikant R., (1994), Fast algorithms for mining association rules. VLDB, 487-499.

Agrawal R., Imielinski T., & Swami A. (1993) Mining association rules between sets of items in large databases. SIGMOD, 207-216.

Berkhin P. & Software A., (2002), Survey Of Clustering Data Mining Techniques.

Borgelt C., & Kruse R., (2002), Induction of Association Rules:Apriori Implementation. 15th Conference on Computational Statistics.

Borgelt C., (2003), Efficient Implementations of Apriori and Eclat, Workshop of Frequent Item Set Mining Implementations (FIMI).

Brin S., Motwani R., & Ullman J. D., Tsur S.,(1997), Dynamic itemset counting and implication rules for market basket analysis. SIGMOD, 255-264

Dong J., Perrizo W., Ding Q., Zhou J. (2000), The Application of Association Rule Mining to Remotely Sensed Data. SAC-1,p. 340-345

Frawley W.J., Piatetsky-Shapiro G., & Matheus C.J. (1991). Knowlegde Discovery in Databases : An Overview, in Knowledge Discovery in Databases, MIT Press.

Goethals B. (2002) Efficient Frequent Pattern Mining, Department of Computer Science University of Helsinki.

Goethals B. (2003), Frequent Pattern Mining (survey), Department of Computer Science University of Helsinki

Han J., Kamber M. (2000), Data Mining: Concepts and Techniques. Morgan Kaufmann Pub. Morgan-Kaufmann Publishers.

Han J., Pei J., Yin Y., (2000), Mining Frequent Patterns without Candidate Generation, ACM SIGMOD Int. Conf. on Management of Data.

Hand D., Mannila H., Smyth P. (2001). Principles of Data Mining. Cambridge, MA: MIT Press.

Herwestphal C. & Blaxton T. (1998) Data Mining Solutions Methods and Tools for SolvingReal-World Problems, Wiley Computer Publishing

Hipp J., Güntzer U., Nakhaeizadeh G., (2000), Algorithms for Association Rule Mining - A General Survey and Comparison. SIGKDD Explorations 2 (1), 58-64.

Hofer J., & Brezany P., (2004), Distributed Decision Tree Induction within the Grid Data Mining Framework GridMiner-Core, Institute for Software Science University of Vienna.

Kumar P., (2001), DataMining of Association Rules, Computer Science and Engineering Indian Institute of Technology

Leung K. S. (2002), Efficient and Effective Exploratory Mining of Constrained Frequent Sets.

Lin D. & Kedem M. (1997) Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Set, New York University.

Lin W, Ruiz C, Alvarez S., (2000), A New Adaptive-Support Algorithm for Association Rule Mining Report WPI-CS-TR-00-13, Department of Computer Science, Worcester Polytechnic Institute.

Pasquier N., Bastide Y., Taouil R, & Lakhal L. (1999). Discovering frequent closed itemsets for association rules. In Proc. ICDT '99, pages 398–416.

Pei J., Han J., & Mao R., (2000), CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets.

Pei J., (2002), Pattern-Growth Methods For Frequent Pattern Mining, Simon Fraser University.

Rud, P. O. (2001). Data Mining Cookbook. NY: John Wiley & Sons, Inc.

Savasere A., Omiecinski E., & Navathe S. (1995), An efficient algorithm for mining association rules in large databases. VLDB,p. 432-444.

Shoemaker C., Ruiz C., (2003), Association rule mining algorithms for set-valued data. Lecture Notes in Computer Science Vol. 2690. Springer-Verlag., p. 669-676.

Wang J. & Karypis G.,(2004), BAMBOO: Accelerating Closed Itemset Mining by Deeply Pushing the Length-Decreasing Support Constraint. SDM 79-90.

WEB_1 (1997) Karuna Pande Joshi's web site. http://userpages.umbc.edu/~kjoshi1/data-mine/proj_rpt.htm, 10/6/2004.

WEB_2 (2003) http://fimi.cs.helsinki.fi/data/ 18/2/2004

Zaki M.J., Parthasarathy S., Ogihara M., Li W., (1997), New algorithms for fast discovery of association rules. Technical Report 651, Computer Science Department, University of Rochester.

Zaki M. J., (1999), Parallel and Distributed Association Mining: A Survey. Sixth ACM SIGKDD international conference on Knowledge discovery and data mining 309–425.

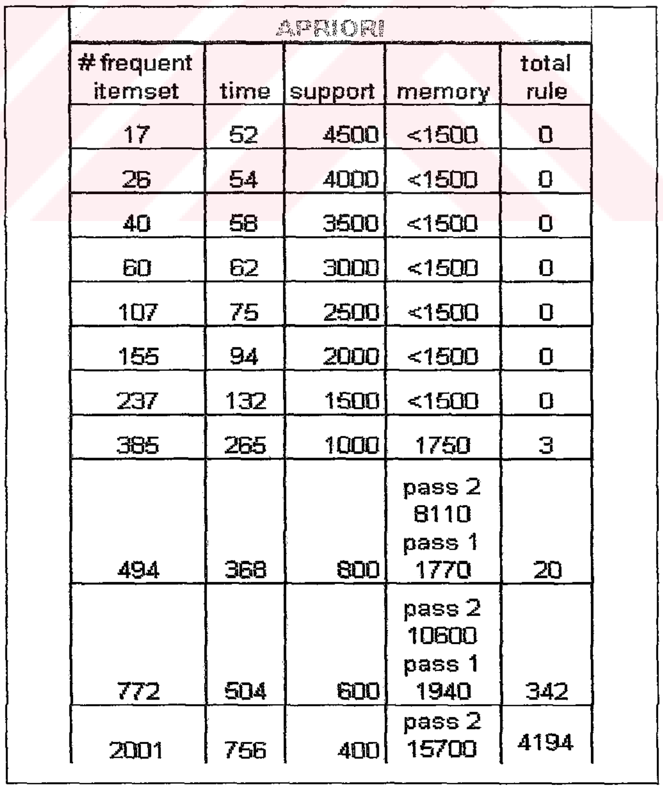
Zaki M. J., (2000), Generating Non-Redundant Association Rules.

Zaki M. J., Hsiao C.J., (2002) CHARM: An Efficient Algorithm for Closed Association Rule Mining. The 2nd SIAM International Conference on Data Mining

Zheng Z., Kohavi R., Mason L., (2001), Real World Performance of Association Rule Algorithms. Seventh ACM SIGKDD international conference on Knowledge discovery and data mining, p. 401 – 406.

APPENDIX A RUNNING TIME FOR CREATING ASSOCIATION RULES

All tables contain the time in seconds spent creating frequent itemsets. All tables also contain number of finding frequent itemsets, support values, memory usage (some of them are not observed) and total number of rules which confidence % 75. Partition algorithms tested only with dataset T10I4D100K.



APRIORI				
# frequent itemset	time	support	memory	total rule
17	52	4500	<1500	0
26	54	4000	<1500	0
40	58	3500	<1500	0
60	62	3000	<1500	0
107	75	2500	<1500	0
155	94	2000	<1500	0
237	132	1500	<1500	0
385	265	1000	1750	3
494	368	800	pass 2 8110 pass 1 1770	20
772	504	600	pass 2 10600 pass 1 1940	342
2001	756	400	pass 2 15700	4194

FP - Tree

# frequent itemset	time	support	memory	total rule
17	60	4500	1250	0
26	79	4000	2100	0
40	141	3500	4900	0
60	333	3000	13500	0
107	1490	2500	36000	0
155	3362	2000		0
237		1500		0
385	2378>	1000	-	3
-	-	800	-	20

ECLAT

itemset	time	support	memory	rule
17	43	4500		0
26	56	4000	3500	0
40	85	3500	6000	0
60	137	3000	6300	0
107	323	2500	6500	0
155	566	2000		0
237	1083	1500		0
385	2147	1000	8500	3
-		800		20

DIC

# frequent itemset	time	support	memory	total rule
17	45	4500	<1500	0
26	52	4000	<1500	0
40	67	3500	<1500	0
60	78	3000	<1500	0
107	103	2500	<1500	0
155	147	2000	<1500	0
237	188	1500	<1500	0
385	305	1000	1750	3
494	452	800	5500	20
772	857	600	7000	342
2001	1500	400	12500	4194

PARTITION

# frequent itemset	time	support	memory	total rule
17	42	4500		0
26	44	4000		0
40	54	3500		0
60	60	3000		0
107	82	2500		0
155	134	2000		0
237	149	1500		0
385	412	1000		3

Table A.1 Results of Experiments for DataSet “T10I4D100K“

APRIORI

# frequent itemset	time	support	total rule
0	77	50000	0
43	311	47500	148
282	599	46250	2618
1163	1216	45000	21394
3684	2373	43750	119012
10531	5410	42500	544998

FP - Tree

# frequent itemset	time	support	total rule
0	139	50000	0
43	176	47500	148
282	188	46250	2618
1163	210	45000	21394
3684	211	43750	119012
10531	243	42500	544998
29384	306	41250	

ECLAT

# frequent itemset	time	support	total rule
0	98	50000	0
43	115	47500	148
282	131	46250	2618
1163	162	45000	21394
3684	176	43750	119012
10531	224	42500	544998
29384	319	41250	

DIC

# frequent itemset	time	support	total rule
0	79	50000	0
43	322	47500	148
282	645	46250	2618
1163	1375	45000	21394
3684	4780	43750	119012
10531	8152	42500	544998

Table A.2 Results of Experiments for DataSet “Pumsb“

APRIORI				
# frequent itemset	time	support	memory	total rule
97	27	4500	<1000	417
167	37	4000	1000	749
369	55	3500	1000	2113
931	76	3000	1000	7290
2365	126	2500	1000	37839
6623	243	2000	2000	180848
14189	410	1800	2000	809993
56693	1256	1500	6000	14517026
123277	2308	1000	11000	

FP - Tree				
# frequent itemset	time	support	memory	total rule
97	9	4500	1000	417
167	12	4000	1000	749
369	12	3500	<1000	2113
931	14	3000	1000	7290
2365	16	2500	1000	37839
6623	21	2000	1000	180848
14189	31	1800	2000	809993
56693	63	1500	2500	14517026
123277	191	1000	4000	

ECLAT				
# frequent itemset	time	support	memory	total rule
97	7	4500	<1000	417
167	10	4000	2000	749
369	14	3500	1000	2113
931	17	3000	2000	7290
2365	21	2500	2000	37839
6623	33	2000	1000	180848
14189	42	1800	1000	809993
56693	54	1500	1500	14517026
123277	96	1000	2000	

DIC				
# frequent itemset	time	support	memory	total rule
97	76	4500	1000	417 / %75
167	119	4000	1000	749
369	210	3500	1200	2113
931	445	3000	1200	7290
2365	1080	2500	1700	37839
6623		2000		180848
14189	<	1800	3100	809993

Table A.3 Results of Experiments for DataSet "Mushroom"

APRIORI				
# frequent itemset	time	support	memory	total rule
0	2	4500	<1000	0
0	2	4000	<1000	0
0	2	3500	<1000	0
155	21	3000	<2000	1330
11493	342	2500	<2500	931674
166580	4144	2000	15000 pass 6-->12	10895753
464670	12468	1800	46000	>10 million

FP - Tree				
# frequent itemset	time	support	memory	total rule
0	5	4500	1000	0
0	5	4000	1000	0
0	5	3500	1000	0
155	7	3000	1000	1330
11493	14	2500	2000	931674
166580	257	2000	3000	10895753
464670	1046	1800	4000	>10 millior
-	5000 >	1500	5500	>10 millior

ECLAT				
# frequent itemset	time	support	memory	total rule
0	2	4500	1500	0
0	2	4000	1100	0
0	2	3500	1250	0
0	4	3000	1300	1330
11493	12	2500	1400	931674
166580	62	2000	1500	10895753
464670	149	1800	1500	>10 millior
2076329	568	1500	1500	>10 millior

DIC				
# frequent itemset	time	support	memory	total rule
0	3	4500	1000	0
0	3	4000	1000	0
0	3	3500	1000	0
155	43	3000		1330
11493	2386	2500	8000	931674
166580		2000		10895753
464670	4500>	1800		

Table A.4 Results of Experiments for DataSet “Chess“