Mining association rules using formal concept analysis.

Jong-Seok. Kim

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

Kim, Jong-Seok., "Mining association rules using formal concept analysis." (2002). Electronic Theses and Dissertations. 2739.
https://scholar.uwindsor.ca/etd/2739

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6” x 9” black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI
MINING ASSOCIATION RULES
USING FORMAL CONCEPT ANALYSIS

By
Kim, Jong-Seok

A Thesis
Submitted to the Faculty of Graduate Studies and Research
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2002
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-67631-5
Abstract

The flood of data has led to new techniques with the ability to assist humans intelligently and automatically in analyzing the overflow of data for retrieving useful knowledge. Association mining is an important problem in data mining. A lot of research contributing to association rules has been proposed in recent years. Many of them are the algorithms that effectively deal with a large itemset method. Although these algorithms increase the efficiency of association mining, they have some critical problems such as flexibility, substantial computational efforts, and redundant comparisons for generating rules.

In this thesis, we propose an alternative approach for the problem of mining association rules based on Formal Concept Analysis. Using this approach, association rules can be discovered dynamically, and the cost of generating rules can be reduced. We also show that the many-valued context of Formal Concept Analysis could be used for finding more detailed quantitative information.
To my parents and my wife, with love
Acknowledgements

I would like to thank to my principal advisor, Dr. Park, for his great advice and guidance on my thesis work. I would also like to thank to Dr. Tsin, Dr. Li and Dr. Suh, who served as a Chair, internal and external readers on my Master’s Committee, for their comments and suggestions on this thesis. Finally, I would like to express my gratitude to my wife and all my friends for their encouragement and help, which made it possible for me to complete the study.
Table of Contents

Abstract ........................................................................................................ iv
Acknowledgements ..................................................................................... vi
List of Tables ................................................................................................ x
List of Figures ............................................................................................... xi

1 Introduction ................................................................. 1
  1.1 Problem description ................................................ 1
  1.2 Thesis outline .............................................................. 3

2 An Overview of Association Rules ............................. 5
  2.1 Knowledge discovery in database and data mining ...... 5
      2.1.1 Definition of knowledge discovery in database ..... 5
      2.1.2 KDD process and data mining ........................ 6
      2.1.3 Data mining methods ........................................ 7
  2.2 General description of association rules .................. 9
  2.3 Initial algorithms ....................................................... 10
  2.4 Later improvements .................................................... 13
  2.5 Generalizations of the association rules .................. 16
  2.6 Online association rules ............................................. 19

3 The Critical Problems of Large Itemset Algorithms .......20
4 An Alternative Approach using Formal Concept Analysis. 22

4.1 An overview of Formal Concept Analysis ........................................... 22

4.1.1 Formal contexts .............................................................................. 23

4.1.2 Concept lattice .............................................................................. 24

4.1.3 Many-valued contexts ................................................................. 26

4.2 Applying Formal Concept Analysis to mining association rules
............................................................................................................... 28

4.2.1 From a relation to a concept lattice ................................................. 29

4.2.2 Computing support and confidence .............................................. 33

4.2.3 Hierarchical comparison ............................................................... 39

4.2.4 Flexibility ....................................................................................... 39

4.2.5 Applying many-valued contexts
to multi-value basket data ............................................................... 41

5 Implementation of a prototype .......................................................... 54

5.1 Data structures .................................................................................. 54

5.2 Retrieving association rules ............................................................ 57

5.3 Browsing association rules ............................................................. 58

6 Conclusions and Future Work ............................................................. 65

References ............................................................................................. 67

APPENDIX A: Source code for data structures ...................................... 71

APPENDIX B: Source code for browsing association rules ................. 97

APPENDIX C: Source code for retrieving association rules ............. 106
Vita Autoris ................................................................. 112
List of Tables

Table 1: A formal context as a cross table ............................................. 24
Table 2: Many-valued context and a scale for the grade attribute...27
Table 3: The derived context using the scale in Table 2 ............... 27
Table 4: Ordinal scales .......................................................................... 28
Table 5: Basket data ............................................................................. 29
Table 6: Context for the basket relations ........................................... 30
Table 7: The extent and intent of the concepts ............................... 31
Table 8: A many-valued context ............................................................ 42
Table 9: Applied ordinal scales ............................................................... 42
Table 10: A derived one-valued context ............................................. 43
Table 11: The extent and intent of the concepts ............................... 45
List of Figures

Figure 1: A concept lattice ................................................................. 25
Figure 2: Concept lattice for the basket context ......................... 32
Figure 3: A part of lattice with concepts and confidence ............ 35
Figure 4: Concept lattice with minimum support ....................... 40
Figure 5: A concept lattice derived from Table 10 ....................... 44
Figure 6: Comparison of one-valued context and many-valued context ................................................................. 46
Figure 7: A part of the transaction table ....................................... 54
Figure 8: Concept abstraction .......................................................... 55
Figure 9: A part of the formal context table ................................... 56
Figure 10: A part of the concept lattice table ............................... 57
Figure 11: Enumeration of association rules ................................. 58
Figure 12: Steps for browsing association rules ......................... 59
Figure 13: Architecture of the system ........................................... 60
Figure 14: The interface of the sample e-commerce web site ....... 61
Figure 15: The results of dynamic web pages using JSP ............... 62
Figure 16: A screen capture with high confidence item suggested. 63
Figure 17: Recommended items order by confidence ..................... 63
Chapter 1  Introduction

1.1 Problem description

The flood of data
The generation and collection of data have developed explosively in last decades. Those are the use of bar codes for almost all commercial items for the commercial transactions, the transaction of banking systems, and other innumerable sources. This flood of data obviously overwhels the traditional methods of data analysis such as spreadsheets and simple usual queries. Although those methods can create informative reports from data, they cannot analyze the contents of those reports to focus on the discovery of important knowledge. It is clear that gathering, and analyzing, acting upon information are most important in the new information age. Therefore, new techniques with the ability to assist humans intelligently and automatically in analyzing the overflow data for retrieving useful knowledge are required. These techniques are the subject of the field of knowledge discovery in database (KDD).

The most popular pattern discovery methods of data mining
One of the central KDD tasks is the discovery of association rules [31]. Since association mining was introduced in 1993 [3], the task of association rules mining has received a great deal of attention [4, 6, 8, 14,
17, 21, 23, 24, 27, 28, 32]. It is apparent that the association mining is one of the most popular pattern discovery methods of data mining in KDD. The initial application was the analysis of supermarket sales or basket data, which can be formally stated as follows: Let \( I = \{i_1, i_2, \ldots, i_n \} \) be the set of items. Each transaction, \( T \), in the database, \( D \), contains a set of items called an itemset (set of items). The support of an itemset is the percentage of transactions in \( D \) that contain the itemset. An association rule is a conditional implication among itemsets, \( A \Rightarrow B \). The confidence of a rule \( A \Rightarrow B \) is the percentage of transactions containing \( A \), which also contain \( B \). Thus, if a rule has 90% confidence then it means that 90% of the itemsets containing \( A \) also contain \( B \). The data mining task of association rules can be decomposed into two steps [4], the first step consists of finding all frequent itemsets, i.e., itemsets that occur in the database with a certain user-specified frequency, called minimum support\(^1\). The second step consists of forming the rules among the frequent itemsets, considering minimum confidence\(^2\). This step is relatively easy, compared to the first step.

**Problems of association rules**

In the recent years, a lot of research has been proposed to discover association rules. Most involve algorithms that successfully modify the

---

\(^1\) **Minimum support** s: the union of items in the consequent and antecedent of the rule is presented in a minimum of s% of transactions in the database.

\(^2\) **Minimum confidence** c: at least c% of transactions in the database that satisfy the rule.
Apriori proposed by Aggrawal [4]. These algorithms increased the efficiency of association mining. Using these algorithms, however, it is difficult to find association rules at any level of minimum support and minimum confidence. If minimum support and minimum confidence are set up too low, then as a result, an unacceptably large number of rules might be generated, and this makes the purpose of data mining be meaningless. On the other hand, if minimum support and minimum confidence are set up too high, the results might be too small to be useful.

An alternative approach using Formal Concept Analysis

From this point of view, a different approach is required, which can dynamically find association rules. Therefore, this thesis proposes an alternative approach using Formal Concept Analysis (FCA). The thesis presents how FCA can be manipulated to find frequent itemsets. Further, it demonstrates how the many-valued context of FCA can be used in order to uncover more detailed quantitative information.

1.2 Thesis outline

The thesis is structured as follows:

Chapter 2 discusses about knowledge discovery in database and data mining. Within the context of these two general concepts, it can be
determined where association mining belongs. Then, the thesis focuses on explanation of association rules, mentioning the most common algorithms.

Chapter 3 brings up critical problems of the large itemset methods.

Chapter 4 proposes an alternative approach using Formal Concept Analysis. For this, the theoretical foundation of Formal Concept Analysis is discussed briefly. Afterward, this chapter focuses on the application of Formal Concept Analysis for mining association rules. Further, using application of the Many-valued context to multi-valued basket data, it is also presented how quantitative information can be manipulated.

Chapter 5 presents the implementation how association rules can be discovered using a concept lattice to reduce the cost of generating association rules and how association rules can be applied to E-business.

Finally, the conclusion of the thesis is outlined in Chapter 6.
Chapter 2 An Overview of Association Rules

This Chapter provides a general description of knowledge Discovery in Databases, Data mining and Methods and the place of association rules within these concepts. This is followed by a discussion of association rules, which is the main subject of the thesis, presenting a review of the most common algorithms, emphasizing the problems addressed by each algorithm.

2.1 Knowledge discovery in database and data mining

The term knowledge Discovery in Databases can be used to denote the overall process of turning low-level data into high-level knowledge. There is still some confusion about the terms ‘knowledge Discovery in Databases (KDD)’ and ‘data mining’. Often these two terms are used interchangeably.

2.1.1 Definition of knowledge discovery in database

A simple definition of KDD is as follows: Knowledge discovery in databases is the nontrivial process of identifying valid, novel, potentially
useful, and ultimately understandable patterns in data [11]. For this
definition, each term is defined as follows [11]:

- **Validity:** The discovered pattern should be valid on new data with some
degree of certainty.
- **Novel:** The patterns are novel. Novelty can be measured with respect to
changes in data or knowledge.
- **Potentially useful:** The patterns should potentially lead to some useful
actions, as measured by some utility function.
- **Ultimately Understandable:** A goal of KDD is to make patterns
understandable to human in order to facilitate a better understanding of
the underlying data.

### 2.1.2 KDD process and data mining

Brachman et al. [7] give a practical view of the KDD process emphasizing
the interactive nature of the process. That involve:

- **Developing an understanding of the application domain, the relevant
prior knowledge, and the goals of the end-user.**
- **Creating a target data set**
- **Data cleaning and preprocessing**
- **Data reduction and projection**
- **Choosing the data mining task**
- **Choosing the data mining algorithm**
Data mining

Interpreting mined patterns, possible return to previous steps for further iteration

Consolidating discovered knowledge

Hence, data mining is just one step in the overall KDD process [11]. Data mining uses statistical analysis and modeling techniques to uncover patterns and relationships hidden in the database.

2.1.3 Data mining methods

There exist a wide variety of data mining methods. For this section, two terms need to be introduced: patterns and models. Fayyad et al. [11] explains those terms using the following example. $f(x) = 3x^2 + x$ is a pattern whereas $f(x) = \alpha x^2 + \beta x$ is considered a model. A pattern can be thought of as instantiation of a model. Choosing the proper model influences a successful knowledge discovery. Model evaluation estimates how well a particular model and its parameters meet the criteria of the KDD process. The specific factors that influence the impact and interestingness of a pattern and hence the criteria of model evaluation will vary for different databases and tasks. Deogun et al. [8] describe the popular 4 Data mining methods in light of model representation and evaluation.
Data Dependency

Data dependencies in DBMS are used for normalizing relations and indexing relations. When a relation is used as a knowledge structure, the set of attributes are partitioned into two groups. The first group is called the set of condition attributes or the feature set, depending on the application domain. The second group is called the set of decision attributes. A block in the partition induced by the decision attributes is called a concept. Concepts are based on a partial ordering of propositions.

It is sometimes useful to determine associations among values of an attribute. For example, planning department at a supermarket may like to know if the customer who purchase 'bread' and 'butter' also tends to purchase 'milk', where 'butter', 'bread', and 'milk' are usually stored within the same attribute of a sales transaction. Agrawal et al. [3] has suggested this type of query along with interval classification. For example, a user may be interested in all associations that have 'milk' in the consequent and 'bread' in the antecedent.

Classification

This kind of query involves inducing a classification function that partitions a given set of tuples into meaningful disjoint subclasses with respect to user-defined labels or the values of some decision attributes.
**Clustering**

Clustering is a task that identifies a finite set of categories or clusters to describe the data. The categories may be mutually exclusive or overlapping categories according to condition. User-defined parameters such as the maximum number of tuples within a cluster or the number of clusters can influence the result of a clustering query. Clustering queries may be helpful for the following two reasons. First, the user may not know the nature or structure of the data. Second, even if the users have some domain knowledge, labeling a large set of tuples can be surprisingly costly and time consuming.

**Characterization**

A classification query emphasizes the finding of features that distinguish different classes. On the other hand, the characterization query describes common features of a class regardless of the characteristics of other classes. A characteristic description can be used to describe what the records in a class share in common among them.

**2.2 General description of association rules**

**Support and confidence**

Let $A$ be a set of items, and $T$ a database of transactions, where each transaction has a unique identifier (tid) and contains a set of items. A set
of items is also called an itemset. The support of an itemset is the percentage of transactions in the database that contain the itself. The support of an itemset $X$, denoted $\sigma(X)$, is the number of transactions in which it occurs as a subset. An itemset is frequent if its support is more than a user-specified minimum support value. The confidence of a rule $A \Rightarrow B$, where $A$ and $B$ are itemsets, is the percentage of transactions containing $A$, which also contain $B$. Thus, if a rule has 90\% confidence then it means that 90\% of the itemsets containing $A$ also contain $B$ [32].

**Frequent itemsets and generating rules**

The mining task consists of two steps: The first of them is finding all large itemset\(^3\). This steps computationally and I/O intensive. The second step is generating high confidence. This step is relatively straightforward; rules are generated for all frequent itemsets, provided the rules have at least minimum confidence [5].

Note that this overview follows the classification of [2].

**2.3 Initial algorithms**

Agrawal et al. [3] introduced the problem of mining a large collection of basket data type transactions for association rules between sets of items in

\(^3\) Itemsets over minimum support are called large itemset [4].
a large database of customer transactions. Agrawal first introduced support and confidence concepts as a statistical significance, a measure of the rule strength respectively. And He also described Minimum transactional support and minimum confidence. These are: Minimum transactional support s – the union of items in the consequent and antecedent of the rule is present in a minimum of s% of transactions in the database. Minimum confidence c – at least c% of transactions in the database that satisfy the antecedent of the rule also satisfy the consequent of the rule. The algorithms have one item in the consequent and a union of any number of items in the antecedent. And this algorithm decomposed the problem into two sub-problems: 1) Finding all itemsets, called large itemsets, that are present in at least s% of transactions. 2) Generating from each large itemset, rules that use items from the large itemset. The disadvantage of this algorithm is that this result can be generated unnecessarily and count too many candidate sets that turn out to be small.

Agrawal et al. [4] presented some algorithms (Apriori, AprioriTid, AprioriHybird) to generate association rules. Apriori algorithm also decomposes the process into two sub-problems, finding all large itemsets, generating the rules using the large itemsets. The process of Apriori algorithm is that the first step of algorithm simply counts item occurrences to determine the large I-itemsets. A subsequent subsets k, consists of two phases. First, the large itemsets \( L_{k-l} \) found in the \((k-l)th\)
subsets are used to generate the candidate itemsets $C_k$. Using the apriori-gen function that is function of scanning database and generating candidate itemsets. Apriori makes additional advantage of using ‘support’ by pruning those candidates that have an infrequent subset before counting their supports. Apriori counts all candidates of cardinality $k$ together in one scan over the database. The critical part is looking up the candidates in each of the transactions. For data structure, Agrawal used hashtree structure that the items in each transaction are used to move down in the hashtree.

AprioriTID also uses the apriori-gen function to determine the candidate itemsets. Instead of counting on the raw database, AprioriTID represents each transaction by the current candidates it contains. Therefore, it only works when the size of candidates relatively small and fit in memory. AprioriHybrid combines both approaches.

SETM [18] is an Apriori-Like algorithm that intends to be implemented directly in SQL. SETM algorithm also generates candidates on the fly based on transactions read from the database. It thus generates and counts every candidate itemset. However, to use the standard SQL join operation for candidate generation, SETM separates candidate generation from counting.
2.4 Later improvements

After the initial algorithms proposed by Agrawal, other researchers have widely studied the problem and a number of variants have been proposed.

One of them is a hash-based algorithm for efficiently finding large itemsets that was proposed by Park et al. [23]. They proposed DHP algorithm. It was observed that most of the time in the algorithm was spent in evaluating and finding large 2-itemsets. The algorithm attempts to improve this approach by providing a hash-based algorithm for quickly finding large 2-itemsets. Generally, given a large database, the initial extraction of useful information from the database is usually the most costly part. DHP presents the technique of hashing to filter out unnecessary itemsets for next candidate itemsets generation, especially effective for the generation of candidate set for large 2-itemsets, where the number of candidate 2-itemsets generated is smaller than that by previous methods [4]. In addition, the generation of smaller candidate sets enables us to effectively trim the transaction database at a much earlier stage of the iterations, thereby reducing the computational cost for later stages significantly.

Savasere et al. [24] proposed a partitioning algorithm that finds large itemsets by dividing the database into \( n \) partitions. The size of each
partition is such that the set of transactions can be maintained in main memory. Then, large itemsets are generated separately for each partition. The algorithm executes in two phases. In the first phase, the partition algorithm logically divided the database into a number of non-overlapping partitions. The partitions are considered on at a time and all large itemsets for that partition are generated. Let $LP_i$ be the set of large itemsets associated with the $i^{th}$ partition. Then, if an itemset is large, then it must be the case that it must belong to at least one of $LP_i$ for $i \in \{1, \ldots, K\}$.

Now, the support of the candidates $Y_{in}^x LP_i$ can be counted in order to find the large itemsets. This method requires just two passes over the transaction database in order to find the large itemsets. At the second phase, these large itemsets are merged to generate itemsets. And then actual supports for these itemsets are generated and the large itemsets are identified. The partition sizes are chosen such that each partition can be accommodated in the main memory so that partitions are read only once in each phase.

Many algorithms that were proposed after Apriori algorithm are variations of Apriori algorithm. As Agrawal mentioned in [4], for databases in which the itemsets may be long, these algorithms may require substantial computational effort. As an example, if a database in which the length of the longest itemset is 30, in this case, there are $2^{30}$ subsets of this single itemset, each of which would need to be scanned against the transaction
database. Thus, the success of the above algorithms counts on the length of the frequent patterns in the database.

An algorithm that has been proposed by Bayardo [7] uses "look-ahead" techniques in order to recognize early which are longer patterns. Then, the subsets of these patterns can be pruned from further consideration.

A random sampling method has been introduced by Toivonen [29]. Since the size of the transaction database is typically very large, it may often be attractive to use random sampling in order to generate the large itemsets. The use of random sampling to generate large itemsets may save considerable I/O costs. The weakness of random sampling is that it can make incorrect results because of the presence of data skew that may make analysis be distorted. For this weakness, anti-skew algorithm was introduced by Lin and Dunham [20]. The algorithm uses a sampling process that can collect knowledge about the data and decrease the number of scanning Database.

Hipp et al. [15] introduce new approach called Hybrid. The idea is to count occurrences whenever determining the support values of relatively small candidates and to rely on tid-set intersections for the remaining candidates. This implies additional cost for generating costs for generating the tid-sets when switching between two counting strategies. For this purpose, this algorithm uses a hashtree-like structure that
contains pointers to tid-sets instead of counters.

2.5 Generalizations of association rules

After association rule problem was proposed using the context of supermarket data that has motivation to find how the items bought in a consumer basket related to each other, a number of interesting extensions have been proposed.

Srikant et al. [28] introduces the problem of mining association rules in large relational tables containing both quantitative and categorical attributes. In such cases association rules are discovered in relational tables that have both categorical and quantitative attributes. Thus, it is possible to find rules that indicate how a given range of quantitative and categorical attributes may affect the values of other attributes in the data. This approach decomposed the problem of discovering quantitative association rules in following five step: 1) Determine the number of partitions for each quantitative attribute. 2) For categorical attributes, map the values of the attributes to a set of consecutive integers. For quantitative attributes that are not partitioned into intervals, the values are mapped to consecutive integers. If a quantitative attribute is partitioned into intervals, the intervals are mapped to consecutive integers. 3) Find the support for each value of both quantitative and categorical
attributes. Additionally, for quantitative attributes, adjacent values are combined as long as their support is less than the user-specified max support. Next, find all frequent itemsets. 4) Use the frequent itemsets to generate association rules. 5) Determine the interesting rules in the output. Often a large number of rules may be produced by such partitioning methods, many of which may not be interesting. Therefore, an interest measure was also defined and used to generate the association rules.

Lent et al. [19] proposed an algorithm for clustering association rules. The aim of this algorithm was to generate rules that were more natural in terms of the quantitative clusters with which individual rules were associated. For this, approach begins by taking source data in tuple form and partitioning those attributes that take values from a continuous domain. Then perform a single pass over the data using an association rule engine to derive a set of association rules. Next, it clusters all those two attribute association rules where the right hand side of the rules satisfies our segmentation criteria.

Another interesting issue is that of handling taxonomies of items. The term 'taxonomy' means a hierarchy on the items. Srikant et al. [27] proposed the problem of mining association rules at multi-level. A solution to the problem is to replace each transaction T with an "extended transaction" T', Where T' contains all the items in T as well as all the ancestors of each items in T. As an example, if the transaction contains
jackets, Outerwear and Clothes could be added to get the extended-transaction. And there may be several kinds of cereal, for each individual kind of cereal; there may be multiple brands. Rules that handle such classifications are called generalized associations. The motivation is to generate rules that are as general as possible and also as general as possible while taking such taxonomies into account.

Savasere et al. [25] also discusses how to find interesting negative association rules in the context of taxonomies of items. He gave an example for negative association rules as follows: What items a customer is not likely to buy given that he buys a certain set of items. “60% of the customers who buy potato chips do not buy bottled water”. The focus of this paper is to find rules that negatively correlate with rules that are discovered at higher levels of the taxonomy.

A different useful improvement of association rules is the concept of cyclic association rules. Ozden et al. [22] have proposed efficient algorithms for the problem of association rules that display cyclic variation over time. It may often be the case that when association rules are computed for data that have a time component, periodic seasonal variations may be observed such as yearly, monthly, daily, and hourly sales data.
2.6 Online association rules

Aggarwal et al. [1] proposed methods for online generation of association rules. It is difficult to provide responses rapidly against user queries because the size of the transaction database may be very large. This algorithm uses OLAP in order to generate association rules quickly by checking pre-store itemsets. The interesting feature of this work is that the rules that are generated are independent of both the size of the transaction data and the number of pre-stored itemsets. The redundancy in association rule generation has been also discussed. A rule is said to be redundant at a given level of support and confidence if its existence is implied by some other rule in the set.
Chapter 3 The Critical Problems of Large Itemset Algorithms

Many of previous algorithms are the algorithms that modify the Apriori, which discover association rules using large itemset methods\(^4\). But, these methods have following problems.

*The difficulty of setting up minimum support and minimum confidence*

The large itemset method has been proved as a useful tool for mining association rules in large datasets. However, it is impossible to find association rules at any level of minimum support and minimum confidence. If minimum support and minimum confidence are set up too low, then as a result, an unacceptably large number of rules might be generated, and this makes the purpose of data mining be meaningless. On the other hand, if minimum support and minimum confidence are set up too high, the results might be too small to be useful.

*Substantial computational effort*

In addition, for databases in which the items may be a lot, the large itemset method may require substantial computational effort. As this thesis mentioned before, if there are 30 items, in this case, there will be

\(^4\) Methods which follow same steps with Apriori for mining association rules
$2^{30}$ possible itemsets, each of which would need to be scanned against the transaction database.

Therefore, the number of scanning database is increase exponentially.

**Redundant comparison for generating rules**

Besides, this method has problems during generating rules. According to this method, after the algorithm get all large itemsets; it compares every itemset to every other itemset in order to find association rules. Therefore, $N \times N$ comparisons have to be happened, in which there might be pruned by minimum confidence criteria. Therefore, the comparison may be redundant and as a result, it increases the cost of generating rules.
Chapter 4 An Alternative Approach using Formal Concept Analysis

From this point of the critical matter described previous chapter, the large itemset methods are difficult to take a broad view to other scenarios for computational problems.

In this Chapter, an alternative approach to discover association rules using Formal Concept Analysis that is a theoretical foundation of this thesis is discussed.

4.1 An overview of Formal Concept Analysis

Formal Concept Analysis is a data analysis technique that was introduced by Wille as a way to identify groupings of objects that have common attributes [30]. It is basically a data analyzing tools that can be used to calculate a concept from a binary relation of objects and attributes, called a formal context. It also provides a graphical representation of the contexts, which is called a concept lattice. This section provides an introduction to Formal Concept Analysis (FCA) based on [13, 26].
4.1.1 Formal contexts

The elementary foundation of Formal Concept Analysis is the formal context. A formal context can be represented by triple \((G, M, I)\) where \(G\) is a finite set of element called objects of the context, \(M\) is a finite set of element called attributes of the context, and \(I\) is a binary relation \(G \times M\). If an object \(g\) has attribute \(m\), then \(g \in G\) is related to \(m\) by \(I\) and it can be written \((g, m) \in I\) or \(gIm\). The contexts can be represented by a cross table. A cross in row \(g\) and column \(m\) means that the object \(g\) has the attribute \(m\). Table 1 shows a cross table of formal context. An ‘\(\times\)’ indicates that an object has that an attribute.

The notion of a formal concept of the context \((G, M, I)\) is a pair \((A,B)\) with \(A \subseteq G, B \subseteq M\). For a set \(A \subseteq G\) of objects, it is defined that \(A' = \{m \in M \mid gIm \text{ for all } g \in A\}\), for a set \(B\) of attributes it also is defined that \(B' = \{g \in G \mid gIm \text{ for all } m \in B\}\). The derivation of a set of objects, denoted \(A'\), contains the set of attributes common to the objects in \(A\). Correspondingly, the derivation of a set of attributes, denoted \(B'\), is the set of objects which have all attributes in \(B\). So, \((A,B)\) is a concept of the formal context \((G,M,I)\) if and only if \(A \subseteq G, B \subseteq M, A' = B\) and \(B' = A\). \(A\) is called the extent and \(B\) is the intent of the concept \((A,B)\).
<table>
<thead>
<tr>
<th></th>
<th>INTELLIGENT</th>
<th>THUMBED</th>
<th>MARINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>×</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Human</td>
<td>×</td>
<td>×</td>
<td></td>
</tr>
<tr>
<td>Fish</td>
<td></td>
<td></td>
<td>×</td>
</tr>
</tbody>
</table>

Table 1: A formal context as a cross table

In this example, objects are the different kinds of animals (dog, human, fish), the attributes are the characteristics of animals (intelligent, marine, thumbed). The binary relation $I$ is given in Table 1. For example, fish and marine are in $I$, but fish and intelligent are not.

4.1.2 Concept lattice

Let $A \subseteq G$ and $B \subseteq M$. The mappings $\sigma(A) = \{ m \in M \mid glm \text{ for all } g \in A \}$ (the common attribute of $A$) and $\tau(B) = \{ g \in G \mid glm \text{ for all } m \in B \}$ (the common objects of $B$) form a Galois connection. In animals example, $\sigma(\{\text{dog, human}\}) = \{\text{intelligent}\}$ and $\tau(\{\text{marine}\}) = \{\text{fish}\}$. The concept is a pair $(\{\text{dog, human}\}, \{\text{intelligent}\})$.

A concept $(A_0, B_0)$ is a sub-concept of concept $(A_1, B_1)$, denoted by $(A_0, B_0) \leq (A_1, B_1)$, if $A_0 \subseteq A_1$ (or, equivalently, $B_1 \subseteq B_0$). The relation $\leq$ is
called the hierarchical order of the concepts. The set of all concepts of \((G, M, I)\) ordered in this way is denoted by \(B(G, M, I)\) and is called the concept lattice of the context \((G, M, I)\). In our example, \(({\text{human}}, \{\text{intelligent, thumbed}\})\) is a sub-concept of \(({\text{dog, human}}, \{\text{intelligent}\})\).

The basic theorem for concept lattice as follows [26]:

\[
X(A_i, B_i) = \tau\left(\bigcap_{\text{rel}} B_i \right) \bigcap \bigcap_{\text{rel}} B_i
\]

The theorem represents a set of concepts can be computed by intersecting their set of attributes, and by finding the common objects that contain the resulting intersection. In our notation, this theorem can be presented \((\tau(\sigma(A)), \sigma(A))\) and the most general of concepts is Top. It can be represented \((\tau(\sigma(\text{All attributes})), \sigma(\text{All attributes}))\). The bottom element of the concept lattice is \((\tau(\sigma(\emptyset)), \sigma(\emptyset))\), and this concept has all the attributes in the context relation.

![Figure 1: A concept lattice](image_url)
4.1.3 Many-valued contexts

In the real world, the object may have attributes with values such as color, weight, grade, etc. These attributes have various values, so they are called many-valued attributes in contrast to the one-valued attributes. A many-valued context \((G, M, W, I)\) consists of sets \(G\), \(M\), \(W\), \(I\) where \(G\) is a set of objects, \(M\) is a many-valued attributes, \(W\) is a set of attribute values, and a ternary relation \(I\) between \(G\), \(M\), and \(W\) (i.e. \(I \subseteq G \times M \times W\)) for which it holds that \((g, m, w) \in I\) and \((g, m, v) \in I\) it always imply \(w = v\). The many valued attributes can be regarded as partial maps from \(G\) in \(W\). Therefore, the set of all values taken on by objects for a given attribute \(m\), is denoted \(\text{dom}(g) = \{g \in G | (g, m, w) \in I \text{ for some } w \in W\}\). Each object has at most one attribute value for each attribute. The entry in row \(g\) and column \(m\) then represents the attribute value \(m(g)\).

For the explanation of data analysis, many-valued contexts can be represented by tables using conceptual scaling, like a one-valued context. The conceptual scaling is an interpretation process that each attribute of a many-valued context is interpreted by means of context. A conceptual scale for the attribute \(m\) of a many-valued context is a context \(S_m = (G_m, M_m, I_m)\) where \(m(G) \subseteq G_m\). \(M_m\) is a new set of attribute, and \(I_m\) is a new relation between \(G_m\) and \(M_m\). Table 2 illustrates an example of conceptual scaling. The conceptual scale interprets A, B and C to
'Excellent' and 'Good'. Table 3 shows the results of applying this scale to the many-valued context in Table 2.

In Table 3, the object students have the new attributes 'Excellent' and 'Good'. This simplest case called as plain scaling that can be achieved by putting together the individual scales without connecting them. This context, which is generated plain scale, is defined as: \((G, N, J)\) with 

\[ N = \bigcup_{m=1}^{M} \text{ and } gJ(m,n) : \Leftrightarrow m(g) = w \text{ and } w l_m n. \]

<table>
<thead>
<tr>
<th>Student A</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student B</td>
<td>B</td>
</tr>
<tr>
<td>Student C</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXCELLENT</th>
<th>GOOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>×</td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Many-valued context and a scale for the grade attribute

<table>
<thead>
<tr>
<th>EXCELLENT</th>
<th>GOOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student A</td>
<td>×</td>
</tr>
<tr>
<td>Student B</td>
<td></td>
</tr>
<tr>
<td>Student C</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: The derived context using the scale in Table 2
One-dimensional ordinal scales, the value of which are ordered and where each value implies the weaker ones. If an attribute has for instance the values \{loud, very loud, extremely loud\}, ordinal scaling suggests itself. It can be represented follow: \(O_n = (n, n, \leq)\). This one-dimensional ordinal scale is used to the application of many-valued context

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

Table 4: Ordinal scales

4.2 Applying Formal Concept Analysis to mining association rules

This section provides an alternative approach to discover association rules using FCA. An association query discovers dependencies among concepts depending on attributes. Some parts of the text are based on [10, 32].
4.2.1 From a relation to a concept lattice

Given a binary relation, a concept consists of transactions and items. Then, all frequent itemsets are uniquely determined by the set of frequent concepts.

The following transaction table represents market-basket-data in which transaction number groups each transaction. In total, 6 transactions and 9 items are shown in table 5.

<table>
<thead>
<tr>
<th>TRANSACTION</th>
<th>ITEM</th>
<th>QUANTITY</th>
<th>TRANSACTION</th>
<th>ITEM</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>11</td>
<td>1</td>
<td>T4</td>
<td>15</td>
<td>3</td>
</tr>
<tr>
<td>T1</td>
<td>12</td>
<td>2</td>
<td>T4</td>
<td>16</td>
<td>2</td>
</tr>
<tr>
<td>T1</td>
<td>14</td>
<td>5</td>
<td>T4</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>T2</td>
<td>11</td>
<td>1</td>
<td>T4</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>T2</td>
<td>14</td>
<td>1</td>
<td>T5</td>
<td>11</td>
<td>5</td>
</tr>
<tr>
<td>T2</td>
<td>13</td>
<td>3</td>
<td>T5</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>T3</td>
<td>11</td>
<td>2</td>
<td>T5</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>T3</td>
<td>12</td>
<td>2</td>
<td>T6</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>T3</td>
<td>14</td>
<td>1</td>
<td>T6</td>
<td>13</td>
<td>2</td>
</tr>
<tr>
<td>T3</td>
<td>19</td>
<td>1</td>
<td>T6</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>T4</td>
<td>12</td>
<td>3</td>
<td>T6</td>
<td>17</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 5: Basket data

A context is a triple \((G, M, I)\) where \(G\) and \(M\) are sets and \(I \subseteq G \times M\).
The elements of G are called objects, and the elements of M are called attributes. For the Transaction id 'g' and item id 'm' where \( g \in G \), and \( m \in M \), \((g, m) \in I (gIm)\) imply that the Transaction id 'g' purchase item id 'm'. Table 6 shows an example of a context \((G, M, I)\) that is derived from Table 5. For this example, \( G=\{T1, T2,..., T6\} \) is a set of six transactions, \( M=\{11, 12...19\} \) is a set of nine items ids which is purchased by \( G \) transactions.

<table>
<thead>
<tr>
<th>TID</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td>x</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Context for the basket relations

A concept of the context \((G, M, I)\) is a pair \((A, B)\) with \( A \subseteq G, B \subseteq M \). For a set \( A \subseteq G \) of objects, following can be defined: \( \sigma(A) = \{m \in M \mid gIm \text{ for all } g \in A\} \) (the common attribute of \( A \)) and \( \tau(B) = \{g \in G \mid gIm \text{ for all } m \in B\} \) (the common objects of \( B \)). So, \((A, B)\) is a concept of the formal context \((G, M, I)\) if and only if \( A \subseteq G, B \subseteq M, \sigma(A) = B \) and \( \tau(B) = A \). A is called the extent and B is the intent of the concept \((A, B)\).
Using computing concept methods, all concepts can be derived as follows:

<table>
<thead>
<tr>
<th>CONCEPTNO.</th>
<th>TID</th>
<th>ITEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOP</td>
<td>T1, T2, T3, T4, T5, T6</td>
<td>∅</td>
</tr>
<tr>
<td>Concept 14</td>
<td>T4, T5, T6</td>
<td>17</td>
</tr>
<tr>
<td>Concept 13</td>
<td>T5, T6</td>
<td>11, 17</td>
</tr>
<tr>
<td>Concept 12</td>
<td>T4, T6</td>
<td>16, 17</td>
</tr>
<tr>
<td>Concept 11</td>
<td>T4, T5</td>
<td>12, 17</td>
</tr>
<tr>
<td>Concept 10</td>
<td>T2, T6</td>
<td>11, 13</td>
</tr>
<tr>
<td>Concept 9</td>
<td>T1, T2, T3, T5, T6</td>
<td>11</td>
</tr>
<tr>
<td>Concept 8</td>
<td>T1, T3, T5</td>
<td>11, 12</td>
</tr>
<tr>
<td>Concept 7</td>
<td>T1, T2, T4, T5</td>
<td>12</td>
</tr>
<tr>
<td>Concept 6</td>
<td>T1, T2, T3</td>
<td>11, 14</td>
</tr>
<tr>
<td>Concept 5</td>
<td>T6</td>
<td>11, 13, 16, 17</td>
</tr>
<tr>
<td>Concept 4</td>
<td>T5</td>
<td>11, 12, 17</td>
</tr>
<tr>
<td>Concept 3</td>
<td>T4</td>
<td>12, 15, 16, 17, 18</td>
</tr>
<tr>
<td>Concept 2</td>
<td>T1, T3</td>
<td>11, 12, 14</td>
</tr>
<tr>
<td>Concept 1</td>
<td>T2</td>
<td>11, 13, 14</td>
</tr>
<tr>
<td>Concept 0</td>
<td>T3</td>
<td>11, 12, 14, 19</td>
</tr>
<tr>
<td>Bottom</td>
<td>∅</td>
<td>11, 12, 13, 14, 15, 16, 17, 18, 19</td>
</tr>
</tbody>
</table>

Table 7: The extent and intent of the concepts

A concept \((A_0, B_0)\) is a sub-concept of concept \((A_i, B_i)\), denoted by \((A_0, B_0)\)
\( \leq (A_i, B_i) \), if \( A_0 \subseteq A_1 \) (or, equivalently, \( B_1 \subseteq B_0 \)). The relation \( \leq \) is called the hierarchical order of the concepts. The set of all concepts of \((G, M, I)\) ordered in this way is denoted by \( B(G, M, I) \) and is called the concept lattice of the context \((G, M, I)\). Using Graphic Representation of Concept Lattice previously mentioned, Concept lattice could be derived like Figure 2.

![Figure 2: Concept lattice for the basket context](image)

It is clear that a correctly illustrated diagram can aid in visualizing and understanding the relationships among the objects and the attributes. All
frequent itemsets are uniquely determined by the frequent concepts. This observation can possibly aid the development of efficient algorithms since enumerating only the closed frequent itemsets is required, instead of enumerating all frequent itemsets like most current algorithms.

4.2.2 Computing support and confidence

The following rules for association mining can be introduced using Formal Concept Analysis, where:

- $C = < V, E >$: A concept lattice where $V$ and $E$ are set of vertices and edges, respectively.
- $itemset(v)$: A function yielding a subset of attributes associated with $v \in V$.
- $frequency(itemset)$: A function that returns frequency of attributes in Basket relation for a given vertex.
- $N$: An index set on the vertices of $V$.
- Each link in $C(E)$ gives us an association from $itemset(v_i)$ to $itemset(v_j)$: where $i, j \in N$.

1) support

Assume $a = itemset(v_i)$ and $\beta = itemset(v_j)$ for $i, j \in N$. Then, the
support for the association $\text{support}(\alpha\beta) = \frac{\text{frequency}(\alpha \cup \beta)}{\text{frequency}(\phi)}$

because $\text{support}$ means the number of transactions (objects) in which it occurs as an itemset (attributes).

2) confidence

i) Suppose $\alpha = \text{itemset}(v_i)$ and $\beta = \text{itemset}(v_j)$ where $i, j \in N$. Then

$$\text{confidence}(\alpha\beta) = \frac{\text{frequency}(\alpha \cup \beta)}{\text{frequency}(\alpha)}$$

because the confidence of a rule $A \Rightarrow B$ is the functions containing $A$, which also contain $B$.

ii) Suppose $\alpha = \text{itemset}(v_i)$ and $\beta = \text{itemset}(v_j)$ where $i, j \in N$.

If $\beta \subseteq \alpha$, the concept that contains $\alpha$ is an attribute labeled concept\(^5\), then $\text{confidence}(\alpha\beta) = 1$. It means that the items of $(\alpha-\beta)$ is always subsumed by $\beta$ in $C$.

Minimum support and minimum confidence can be easily examined when $\text{support}$ and $\text{confidence}$ are found using concept lattice.

---

\(^5\) An attribute labeled concept contains an attribute that is shown for the first time. It is helpful for visualizing and understanding the relation among all attributes in the lattice structure.
TOP Concept (\{T1, T2, T3, T4, T5, T6\}, \emptyset) 

\[
\begin{array}{c}
\{T1, T2, T3, T5, T6\}, \{I1\} \\
\text{P1} \\
\frac{3}{5} & \frac{2}{5} & \frac{2}{5} \\
& \text{P3} \\
\{T1, T2, T3\}, \{I1, I4\} & \{T2, T6\}, \{I1, I3\} & \{T5, T6\}, \{I1, I7\} \\
\frac{1}{3} & \frac{1}{2} & \frac{1}{2}
\end{array}
\]

\[
\begin{array}{c}
\{T2\}, \{I1, I3, I4\} & \{T6\}, \{I1, I3, I6, I7\}
\end{array}
\]

Figure 3: A part of lattice with concepts and confidence

**Examples**

As an example, in order to compute the *support* for itemset \{I1\} and \{I1, I4\}, the number of top concept transactions is 6, and the number of transactions of concept 6 is 3. Concept 6 contains I1 and I4 as attributes.
Therefore, the support is: \[
\frac{\text{frequency}\{P1, P4\}}{\text{frequency}(\emptyset)} = \frac{3}{6} = 50%.
\]

For the confidence, for itemset \{I1\} and itemset \{I1, I4\}, the number of transactions of concept 9 is 5, and the number of transactions of concept 6 is 3. Therefore, the confidence is: \[
\frac{\text{frequency}\{P1, P4\}}{\text{frequency}\{P1\}} = \frac{3}{5} = 60%.
\]

For the confidence of sub-concept to super-concept relation, for itemsets \{I1, I3\} and \{I1\}, confidence (\{I1, I3\} \cap \{I1\}) = 1 apparently. In this case, Concept 10 is a labeled concept for item ‘I3’, and \{I1\} \subseteq \{I1, I3\}. Also from Figure 3, using lattice, it is clear that the item ‘I3’ is always subsumed in the item ‘I1’; that is the item ‘I3’ always is shown with ‘I1’.

As a result, all association rules can be derived as follows:

(Explanatory) Antecedent \Rightarrow\text{Consequent} [\text{support, confidence}]

AR1: I4 \Rightarrow I1 [50.0\%, 100.0\%] From Concept 6 and 9
AR2: I3 \Rightarrow I1 [33.0\%, 100.0\%] From Concept 10 and 9
AR3: I6 \Rightarrow I7 [33.0\%, 100.0\%] From Concept 12 and 14
AR4: I9 \Rightarrow I1 [17.0\%, 100.0\%] From Concept 0 and 9
AR5: I9 \Rightarrow I2 [17.0\%, 100.0\%] From Concept 0 and 7
AR6: I9 \Rightarrow I1 \& I4 [17.0\%, 100.0\%] From Concept 0 and 6
AR7: I9 $\Rightarrow$ I1 & I2 [17.0%, 100.0%] From Concept 0 and 8
AR8: I9 $\Rightarrow$ I1 & I2 & I4 [17.0%, 100.0%] From Concept 0 and 2
AR9: I5 $\Rightarrow$ I2 [17.0%, 100.0%] From Concept 3 and 7
AR10: I5 $\Rightarrow$ I7 [17.0%, 100.0%] From Concept 3 and 14
AR11: I5 $\Rightarrow$ I2 & I7 [17.0%, 100.0%] From Concept 3 and 11
AR12: I5 $\Rightarrow$ I6 & I7 [17.0%, 100.0%] From Concept 3 and 12
AR13: I8 $\Rightarrow$ I2 [17.0%, 100.0%] From Concept 3 and 7
AR14: I8 $\Rightarrow$ I7 [17.0%, 100.0%] From Concept 3 and 14
AR15: I8 $\Rightarrow$ I2 & I7 [17.0%, 100.0%] From Concept 3 and 11
AR16: I8 $\Rightarrow$ I6 & I7 [17.0%, 100.0%] From Concept 3 and 12
AR17: I2 $\Rightarrow$ I1 [50.0%, 75.0%] From Concept 7 and 8
AR18: I7 $\Rightarrow$ I6 [33.0%, 67.0%] From Concept 14 and 12
AR19: I7 $\Rightarrow$ I1 [33.0%, 67.0%] From Concept 14 and 13
AR20: I7 $\Rightarrow$ I2 [33.0%, 67.0%] From Concept 14 and 11
AR21: I1 & I2 $\Rightarrow$ I4 [33.0%, 67.0%] From Concept 8 and 2
AR22: I1 & I4 $\Rightarrow$ I2 [33.0%, 67.0%] From Concept 6 and 2
AR23: I1 $\Rightarrow$ I4 [50.0%, 60.0%] From Concept 9 and 6
AR24: I1 $\Rightarrow$ I2 [50.0%, 60.0%] From Concept 9 and 8
AR25: I2 $\Rightarrow$ I7 [33.0%, 50.0%] From Concept 7 and 11
AR26: I2 $\Rightarrow$ I1 & I4 [33.0%, 50.0%] From Concept 7 and 2
AR27: I1 & I2 & I4 $\Rightarrow$ I9 [17.0%, 50.0%] From Concept 2 and 0
AR28: I1 & I3 $\Rightarrow$ I6 & I7 [17.0%, 50.0%] From Concept 10 and 5
AR29: I2 & I7 $\Rightarrow$ I1 [17.0%, 50.0%] From Concept 11 and 4
AR30: I6 & I7 $\Rightarrow$ I1 & I3 [17.0%, 50.0%] From Concept 12 and 5
AR31: I1 & I7 ⇒ I3 & I6 [17.0%, 50.0%] From Concept 13 and 5
AR32: I1 & I7 ⇒ I2 [17.0%, 50.0%] From Concept 13 and 4
AR33: I6 & I7 ⇒ I2 & I5 & I8 [17.0%, 50.0%] From Concept 12 and 3
AR34: I2 & I7 ⇒ I5 & I6 & I8 [17.0%, 50.0%] From Concept 11 and 3
AR35: I1 & I3 ⇒ I4 [17.0%, 50.0%] From Concept 10 and 1
AR36: I1 ⇒ I2 & I4 [33.0%, 40.0%] From Concept 9 and 2
AR37: I1 ⇒ I7 [33.0%, 40.0%] From Concept 9 and 13
AR38: I1 ⇒ I3 [33.0%, 40.0%] From Concept 9 and 10
AR39: I1 & I4 ⇒ I3 [17.0%, 33.0%] From Concept 6 and 1
AR40: I7 ⇒ I1 & I3 & I6 [17.0%, 33.0%] From Concept 14 and 5
AR41: I7 ⇒ I1 & I2 [17.0%, 33.0%] From Concept 14 and 4
AR42: I7 ⇒ I2 & I5 & I6 & I8 [17.0%, 33.0%] From Concept 14 and 3
AR43: I1 & I2 ⇒ I7 [17.0%, 33.0%] From Concept 8 and 4
AR44: I1 & I2 ⇒ I4 & I9 [17.0%, 33.0%] From Concept 8 and 0
AR45: I1 & I4 ⇒ I2 & I9 [17.0%, 33.0%] From Concept 6 and 0
AR46: I2 ⇒ I1 & I4 & I9 [17.0%, 25.0%] From Concept 10 and 0
AR47: I2 ⇒ I5 & I6 & I7 & I8 [17.0%, 25.0%] From Concept 7 and 3
AR48: I2 ⇒ I1 & I7 [17.0%, 25.0%] From Concept 7 and 4
AR49: I1 ⇒ I3 & I4 [17.0%, 20.0%] From Concept 9 and 1
AR50: I1 ⇒ I2 & I4 & I9 [17.0%, 20.0%] From Concept 9 and 0
AR51: I1 ⇒ I2 & I7 [17.0%, 20.0%] From Concept 9 and 4
AR52: I1 ⇒ I3 & I6 & I7 [17.0%, 20.0%] From Concept 9 and 3
4.2.3 Hierarchical comparison

In the large itemset methods such as Apriori algorithm, after the algorithm determines all the large itemsets, it compares every itemset to every other itemset in order to find association rules. Therefore, $N \times N$ comparisons have to occur, in which the results may or may not be pruned by minimum confidence criteria. Therefore, the comparison may be redundant and as a result, it increases the cost of generating rules.

The FCA method, however, can calculate only linked concept hierarchically in the lattice structure, instead of calculating all frequent itemsets like most current algorithms. In Figure 3, the itemset in concept 10 considers only the itemsets in concept 1, 5, and 9. This method can significantly reduce the cost of generating association rules.

4.2.4 Flexibility

FCA-based approach can dynamically find association rules since it works at any level of minimum support and minimum confidence. From this point of view, it is clear that this approach is more flexible than the large itemset method.

As an example, for item I1 and itemset \{I1, I3, I4\}, the support is $\frac{1}{6}$.
17%, and the confidence is $\frac{1}{5} = 20\%$. If minimum support and minimum confidence are set up 20%, and 30% respectively, this itemset would be excluded in the enumeration of association rules. Figure 4 shows concept lattice with minimum support of 20%.

Figure 4: Concept lattice with minimum support
4.2.5 Applying many-valued contexts to multi-valued basket data

In the real world, the customer may purchase a different number of items per each transaction. Many algorithms deal with single values, for example, if a customer buys 3 of A item, 5 of B item, 1 of C item and 2 of D item, many algorithms take into consideration only that customer buys A, B, C, and D without regarding all 11 items.

Using the Many-valued Context of Formal Concept Analysis, all these items can be considered. This is related the generalization of association rules, which is introduced in Chapter 2. The quantity of purchased items can be transformed into a one-valued context in the same way that a Many-valued context is transformed into a one-valued one. Using scaling, more detailed information can be found. Many-valued contexts can be represented by tables using conceptual scaling, like the one-valued contexts. A conceptual scale for the attribute $m$ of a many-valued context $(G, M, W, I)$ is a context $S_m = (G_m, M_m, I_m)$ where $m(G) \subseteq G_m$. $M_m$ is a new set of attribute, and $I_m$ is a new relation between $G_m$ and $M_m$.

For our example, the range which can be symbolized each value has to be set up. Assume the range for values is GM: 4 ~ 6 or over, M: 2 ~ 3 O: 1.
Then, $G=\{T1, T2 \ldots T6\}$ is a set of six transactions, $M=\{I1, I2 \ldots I9\}$ is a set of nine item ids, which is purchased by $G$ transactions, and $W=\{\text{Great Many, Many, And One}\}$. The example can be represented as follows:

<table>
<thead>
<tr>
<th>TID</th>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>I5</th>
<th>I6</th>
<th>I7</th>
<th>I8</th>
<th>I9</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>O</td>
<td>M</td>
<td></td>
<td>GM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>O</td>
<td></td>
<td>M</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>M</td>
<td>M</td>
<td></td>
<td>O</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>O</td>
</tr>
<tr>
<td>T4</td>
<td></td>
<td>M</td>
<td></td>
<td></td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>GM</td>
<td>GM</td>
<td></td>
<td></td>
<td></td>
<td>GM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td>O</td>
<td></td>
<td>M</td>
<td></td>
<td></td>
<td>GM</td>
<td>GM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8 A many-valued context

<table>
<thead>
<tr>
<th></th>
<th>GM</th>
<th>M</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>M</td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>O</td>
<td></td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

Table 9 Applied ordinal scales

Using ordinal scales shown in Table 9, Many-valued context can be transformed into a derived one-valued context. The result of this example is shown in Table 10, Figure 5, and Table 11.
<table>
<thead>
<tr>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15M</th>
<th>16M</th>
<th>16O</th>
<th>17M</th>
<th>17O</th>
<th>18M</th>
<th>18O</th>
<th>19M</th>
<th>19O</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10 A derived one-valued context
Figure 5: The concept lattice derived from Table 10
<table>
<thead>
<tr>
<th>CONCEPT</th>
<th>TID</th>
<th>ITEMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOP</td>
<td>T1,T2,T3,T4,T5,T6</td>
<td>∅</td>
</tr>
<tr>
<td>C16</td>
<td>T4, T5, T6</td>
<td>17M, 17O</td>
</tr>
<tr>
<td>C15</td>
<td>T5, T6</td>
<td>11O, 17GM, 17M, 17O</td>
</tr>
<tr>
<td>C14</td>
<td>T4, T6</td>
<td>16M, 16O, 17M, 17O</td>
</tr>
<tr>
<td>C13</td>
<td>T4, T5</td>
<td>12M, 12O, 17M, 17O</td>
</tr>
<tr>
<td>C12</td>
<td>T3, T5</td>
<td>11M, 11O, 12M, 12O</td>
</tr>
<tr>
<td>C11</td>
<td>T2, T6</td>
<td>13M, 13O, 11O</td>
</tr>
<tr>
<td>C10</td>
<td>T1, T2, T3, T5, T6</td>
<td>11O</td>
</tr>
<tr>
<td>C9</td>
<td>T1, T3, T5</td>
<td>11O, 12M, 12O</td>
</tr>
<tr>
<td>C8</td>
<td>T1, T3, T4, T5</td>
<td>11, 12</td>
</tr>
<tr>
<td>C7</td>
<td>T1, T3</td>
<td>11O, 12M, 12O, 14O</td>
</tr>
<tr>
<td>C6</td>
<td>T1, T2, T3</td>
<td>11O, 14O</td>
</tr>
<tr>
<td>C2</td>
<td>T3</td>
<td>11M, 11O, 12M, 12O, 14O, 19O</td>
</tr>
<tr>
<td>C1</td>
<td>T2</td>
<td>11O, 13M, 13O, 14O</td>
</tr>
<tr>
<td>C0</td>
<td>T1</td>
<td>11O, 12M, 12O, 14GM, 14M, 14O</td>
</tr>
</tbody>
</table>

Table 11: The extent and intent of the concepts
Using the many-valued context, detailed quantitative information can be obtained from a lattice completed by a derived context. Figure 6 shows that the difference between two lattices. The first one is a one-valued context; the second is from a derived context. Although both figures contain the same items information, the second one contains more detailed quantitative information. For example, in the first figure, 75% of transactions that contain I2 also contain I1 (Concept 7, Concept 8). However, in the second one, 75% of transactions that bought two or three of I2 also contain one of I1.

Figure 6: Comparison of one-valued context and many-valued context
As a result, all association rules can be derived as follow:

(Explanatory) Antecedent ⇒Consequent [support, confidence]

AR1: I4O ⇒ I1O [50.0%, 100.0%] From Concept 6 and 10
AR2: I3O ⇒ I1O [33.0%, 100.0%] From Concept 11 and 10
AR3: I3M ⇒ I1O [33.0%, 100.0%] From Concept 11 and 10
AR4: I1M ⇒ I1O & I2M & I2O [33.0%, 100.0%] From Concept 12 and 9
AR5: I6M ⇒ I7M & I7O [33.0%, 100.0%] From Concept 14 and 16
AR6: I7GM ⇒ I7M & I7O [33.0%, 100.0%] From Concept 15 and 16
AR7: I6O ⇒ I7M & I7O [33.0%, 100.0%] From Concept 14 and 16
AR8: I1M ⇒ I1O [33.0%, 100.0%] From Concept 12 and 10
AR9: I1M ⇒ I2M & I2O [33.0%, 100.0%] From Concept 12 and 8
AR10: I7GM ⇒ I1O [33.0%, 100.0%] From Concept 15 and 10
AR11: I4GM ⇒ I1O & I4O [17.0%, 100.0%] From Concept 0 and 6
AR12: I4GM ⇒ I1O & I2M & I2O & I4O [17.0%, 100.0%]
   From Concept 0 and 7
AR13: I4GM ⇒ I2M & I2O [17.0%, 100.0%] From Concept 0 and 8
AR14: I4GM ⇒ I1O & I2M & I2O [17.0%, 100.0%] From Concept 0 and 9
AR15: I4GM ⇒ I1O [17.0%, 100.0%] From Concept 0 and 10
AR16: I4M ⇒ I1O & I4O [17.0%, 100.0%] From Concept 0 and 6
AR17: I4M ⇒ I1O & I2M & I2O & I4O [17.0%, 100.0%]
   From Concept 0 and 7
AR18: I4M ⇒ I2M & I2O [17.0%, 100.0%] From Concept 0 and 8
AR19: I4M ⇒ I1O & I2M & I2O [17.0%, 100.0%] From Concept 0 and 9
AR20: I4M ⇒ I1O [17.0%, 100.0%] From Concept 0 and 10
AR21: I9O ⇒ I1O & I4O [17.0%, 100.0%] From Concept 2 and 6
AR22: I9O ⇒ I1O & I2M & I2O & I4O [17.0%, 100.0%]
   From Concept 2 and 7
AR23: I9O ⇒ I2M & I2O [17.0%, 100.0%] From Concept 2 and 8
AR24: I9O ⇒ I1O & I2M & I2O [17.0%, 100.0%] From Concept 2 and 9
AR25: I9O ⇒ I1O [17.0%, 100.0%] From Concept 2 and 10
AR26: I9O ⇒ I1M & I1O & I2M & I2O [17.0%, 100.0%]
   From Concept 2 and 12
AR27: I5O ⇒ I2M & I2O [17.0%, 100.0%] From Concept 3 and 8
AR28: I5O ⇒ I2M & I2O & I7M & I7O [17.0%, 100.0%]
   From Concept 3 and 13
AR29: I5O ⇒ I6M & I6O & I7M & I7O [17.0%, 100.0%]
   From Concept 3 and 14
AR30: I5O ⇒ I7M & I7O [17.0%, 100.0%] From Concept 3 and 16
AR31: I5M ⇒ I2M & I2O [17.0%, 100.0%] From Concept 3 and 8
AR32: I5M ⇒ I2M & I2O & I7M & I7O [17.0%, 100.0%]
   From Concept 3 and 13
AR33: I5M ⇒ I6M & I6O & I7M & I7O [17.0%, 100.0%]
   From Concept 3 and 14
AR34: I5M ⇒ I7M & I7O [17.0%, 100.0%] From Concept 3 and 16
AR35: I8O ⇒ I2M & I2O [17.0%, 100.0%] From Concept 3 and 8
AR36: I8O ⇒ I2M & I2O & I7M & I7O [17.0%, 100.0%]  
From Concept 3 and 13

AR37: I8O ⇒ I6M & I6O & I7M & I7O [17.0%, 100.0%]  
From Concept 3 and 14

AR38: I8O ⇒ I7M & I7O [17.0%, 100.0%] From Concept 3 and 16

AR39: I8M ⇒ I2M & I2O [17.0%, 100.0%] From Concept 3 and 8

AR40: I8M ⇒ I2M & I2O & I7M & I7O [17.0%, 100.0%]  
From Concept 3 and 13

AR41: I8M ⇒ I6M & I6O & I7M & I7O [17.0%, 100.0%]  
From Concept 3 and 14

AR42: I8M ⇒ I7M & I7O [17.0%, 100.0%] From Concept 3 and 16

AR43: I1GM ⇒ I2M & I2O [17.0%, 100.0%] From Concept 4 and 8

AR44: I1GM ⇒ I1O & I2M & I2O [17.0%, 100.0%] From Concept 4 and 9

AR45: I1GM ⇒ I1O [17.0%, 100.0%] From Concept 4 and 10

AR46: I1GM ⇒ I1M & I1O & I2M & I2O [17.0%, 100.0%]  
From Concept 4 and 12

AR47: I1GM ⇒ I2M & I2O & I7M & I7O [17.0%, 100.0%]  
From Concept 4 and 13

AR48: I1GM ⇒ I1O & I1GM & I7M & I7O [17.0%, 100.0%]  
From Concept 4 and 15

AR49: I1GM ⇒ I7M & I7O [17.0%, 100.0%] From Concept 4 and 16

AR50: I2GM ⇒ I2M & I2O [17.0%, 100.0%] From Concept 4 and 8

AR51: I2GM ⇒ I1O & I2M & I2O [17.0%, 100.0%] From Concept 4 and 9

AR52: I2GM ⇒ I1O [17.0%, 100.0%] From Concept 4 and 10
AR53: I2GM ⇒ I1M & I1O & I2M & I2O [17.0%, 100.0%]  
From Concept 4 and 12

AR54: I2GM ⇒ I2M & I2O & I7M & I7O [17.0%, 100.0%]  
From Concept 4 and 13

AR55: I2GM ⇒ I1O & I7GM & I7M & I7O [17.0%, 100.0%]  
From Concept 4 and 15

AR56: I2GM ⇒ I7M & I7O [17.0%, 100.0%] From Concept 4 and 16

AR57: I6GM ⇒ I1O [17.0%, 100.0%] From Concept 5 and 10

AR58: I6GM ⇒ I1O & I3M & I3O [17.0%, 100.0%]  
From Concept 5 and 11

AR59: I6GM ⇒ I6M & I6O & I7M & I7O [17.0%, 100.0%]  
From Concept 5 and 14

AR60: I6GM ⇒ I1O & I7GM & I7M & I7O [17.0%, 100.0%]  
From Concept 5 and 15

AR61: I6GM ⇒ I7M & I7O [17.0%, 100.0%] From Concept 5 and 16

AR62: I2M & I2O ⇒ I1O [50.0%, 75.0%] From Concept 8 and 9

AR63: I1O & I4O ⇒ I2M & I2O [33.0%, 67.0%] From Concept 6 and 7

AR64: I7M & I7O ⇒ I2M & I2O [33.0%, 67.0%]  
From Concept 16 and 13

AR65: I1O & I2M & I2O ⇒ I4O [33.0%, 67.0%] From Concept 9 and 7

AR66: I7M & I7O ⇒ I1O & I7GM [33.0%, 67.0%]  
From Concept 16 and 15

AR67: I7M & I7O ⇒ I6M & I6O [33.0%, 67.0%]  
From Concept 16 and 14
AR68: II0 & I2M & I2O ⇒ I1M [33.0%, 67.0%] From Concept 9 and 12
AR69: II0 ⇒ I4O [50.0%, 60.0%] From Concept 10 and 6
AR70: II0 ⇒ I2M & I2O [50.0%, 60.0%] From Concept 10 and 9
AR71: I2M & I2O ⇒ II0 & I4O [33.0%, 50.0%] From Concept 8 and 7
AR72: I2M & I2O ⇒ I7M & I7O [33.0%, 50.0%] From Concept 8 and 13
AR73: I2M & I2O ⇒ I1M & I1O [33.0%, 50.0%] From Concept 8 and 12
AR74: II0 & I2M & I2O & I4O ⇒ I4GM & I4M [17.0%, 50.0%]
    From Concept 7 and 0
AR75: II0 & I7GM & I7M & I7O ⇒ I3M & I3O & I6GM & I6M & I6O
    [17.0%, 50.0%] From Concept 15 and 5
AR76: II0 & I7GM & I7M & I7O ⇒ I1GM & I1M & I2GM & I2M & I2O
    [17.0%, 50.0%] From Concept 15 and 4
AR77: I6M & I6O & I7M & I7O ⇒ II0 & I3M & I3O & I6GM & I7GM
    [17.0%, 50.0%] From Concept 14 and 5
AR78: I6M & I6O & I7M & I7O ⇒ I2M & I2O & I5M & I5O & I8M &
    I8O [17.0%, 50.0%] From Concept 14 and 3
AR79: I2M & I2O & I7M & I7O ⇒ I1GM & I1M & I1O & I2GM & I7GM
    [17.0%, 50.0%] From Concept 13 and 4
AR80: I2M & I2O & I7M & I7O ⇒ I5M & I5O & I6M & I6O & I8M &
    I8O [17.0%, 50.0%] From Concept 13 and 3
AR81: I1M & I1O & I2M & I2O ⇒ I1GM & I2GM & I7GM & I7M & I7O
    [17.0%, 50.0%] From Concept 12 and 4
AR82: I1M & I1O & I2M & I2O ⇒ I4O & I9O [17.0%, 50.0%]
    From Concept 12 and 2
AR83: I1O & I3M & I3O \implies I6GM & I6M & I6O & I7GM & I7M & I7O
[17.0\%, 50.0\%] From Concept 11 and 5

AR84: I1O & I3M & I3O \implies I4O [17.0\%, 50.0\%] From Concept 11 and 1

AR85: I1O & I2M & I2O & I4O \implies I1M & I9O [17.0\%, 50.0\%]
From Concept 7 and 2

AR86: I1O \implies I2M & I2O & I4O [33.0\%, 40.0\%] From Concept 10 and 7

AR87: I1O \implies I3M & I3O [33.0\%, 40.0\%] From Concept 10 and 11

AR88: I1O \implies I7GM & I7M & I7O [33.0\%, 40.0\%]
From Concept 10 and 15

AR89: I1O \implies I1M & I2M & I2O [33.0\%, 40.0\%]
From Concept 10 and 12

AR90: I1O & I4O \implies I2M & I2O & I4GM & I4M [17.0\%, 33.0\%]
From Concept 6 and 0

AR91: I1O & I4O \implies I3M & I3O [17.0\%, 33.0\%] From Concept 6 and 1

AR92: I7M & I7O \implies I1O & I3M & I3O & I6GM & I6M & I6O & I7GM
[17.0\%, 33.0\%] From Concept 16 and 5

AR93: I7M & I7O \implies I1GM & I1M & I1O & I2GM & I2M & I2O & I7GM
[17.0\%, 33.0\%] From Concept 16 and 4

AR94: I7M & I7O \implies I2M & I2O & I5M & I5O & I6M & I6O & I8M & I8O [17.0\%, 33.0\%] From Concept 16 and 3

AR95: I1O & I4O \implies I1M & I2M & I2O & I9O [17.0\%, 33.0\%]
From Concept 6 and 2

AR96: I1O & I2M & I2O \implies I1M & I4O & I9O [17.0\%, 33.0\%]
From Concept 9 and 2
AR97: I1O & I2M & I2O \Rightarrow I1GM & I1M & I2GM & I7GM & I7M & I7O \\
[17.0\%, 33.0\%] From Concept 9 and 4

AR98: I1O & I2M & I2O \Rightarrow I4GM & I4M & I4O [17.0\%, 33.0\%] \\
From Concept 9 and 0

AR99: I2M & I2O \Rightarrow I1O & I4GM & I4M & I4O [17.0\%, 25.0\%] \\
From Concept 8 and 0

AR100: I2M & I2O \Rightarrow I1M & I1O & I4O & I9O [17.0\%, 25.0\%] \\
From Concept 8 and 2

AR101: I2M & I2O \Rightarrow I5M & I5O & I6M & I6O & I7M & I7O & I8M & I8O  \\
[17.0\%, 25.0\%] From Concept 8 and 3

AR102: I2M & I2O \Rightarrow I1GM & I1M & I1O & I2GM & I7GM & I7M & I7O  \\
[17.0\%, 25.0\%] From Concept 8 and 4

AR103: I1O \Rightarrow I2M & I2O & I4GM & I4M & I4O [17.0\%, 20.0\%] \\
From Concept 10 and 0

AR104: I1O \Rightarrow I3M & I3O & I4O [17.0\%, 20.0\%] From Concept 10 and 1

AR105: I1O \Rightarrow I1M & I2M & I2O & I4O & I9O [17.0\%, 20.0\%] \\
From Concept 10 and 2

AR106: I1O \Rightarrow I1GM & I1M & I2GM & I2M & I2O & I7GM & I7M & I7O  \\
[17.0\%, 20.0\%] From Concept 10 and 4

AR107: I1O \Rightarrow I3M & I3O & I6GM & I6M & I6O & I7GM & I7M & I7O  \\
[17.0\%, 20.0\%] From Concept 10 and 5
Chapter 5 Implementation of a prototype

Based on the Formal Concept Analysis discussed in the previous Chapter, a prototype has been implemented. First, Data structure is set up for generating association rules, then, this data structure is applied to the E-commerce web site using association rules. In the second section, instead of enumerating all itemsets, only the calculation of closed frequent itemsets that can be used in analysis of basket data is shown. It can reduce significantly the cost of generating useful association rules.

5.1 Data structures

The lattice structure can be stored in the relational table. All concepts are stored as a form of tuples in the relational table. For this implementation, 1000 transactions and 100 items are used, using Oracle 7 Personal edition.

The Formal concept of the context \((G, M, I)\) is a pair \((A, B)\) with \(A \subseteq G\), \(B \subseteq M\).
The mappings \( \sigma(A) = \{ m \in M \mid glm \text{ for all } g \in A \} \) (the common attribute of \( A \)) and \( \tau(B) = \{ g \in G \mid glm \text{ for all } m \in B \} \) (the common objects of \( B \)).

By transferring the information that is from Figure 7, following context table can be made.

Using this mapping, all sets of objects and attributes are obtained. Figure 8 shows the contents of Concept vector in which each concept contains ObjectSet, AttributeSet, ChildrenSet, and ParentsSet. ChildrenSet and ParentsSet can be obtained through sub-concepts and super-concepts, which mentioned in the previous Chapter.

The Formal context table is shown in Figure 9. In this table, ‘1’ represents that each customer bought a specific item. ‘0’ means there is no relation
between transactions and items. (See appendix A)

![Image](image_url)

**Figure 9: A part of the formal context table**

In Figure 10, all concepts are stored as a form of tuples in the relational table. (See appendix A) There are several columns. A column 'Object set' contains all transactions (object) and column 'Attribute set' contains all item ids (attributes). Each tuple is stored as a single string connecting each piece of data using ' _ '. Column 'Child set' and 'Parent set' contain children concept number and parent concept number in the concept lattice structure, respectively.
5.2 Retrieving association rules

The FCA-based approach can compare with only itemsets of linked concept hierarchically from the lattice table, instead of calculating all frequent itemsets like most current algorithms. This method can significantly reduce the cost of generating association rules.

FCA-based approach works at any level of minimum support and minimum confidence. From this point of view, it is clear that this approach is more flexible than the large itemset method.
Figure 11: Enumeration of association rules

Figure 11 shows the enumeration of the closed frequent itemsets. The itemset of each concept generates association rules comparing with the itemsets of sub-concepts and their parent concept. In this figure, *minimum support* and *minimum confidence* is set up as 20%, 30% respectively.

5.3 Browsing association rules

This section shows how association rules can be applied to E-business using concept lattice. As Aggarwal et al. [1] mentioned, it is difficult to
provide responses rapidly against user queries because the size of the transaction database may be very large. The solution that this thesis introduces is checking sequentially children nodes that contain pre-store itemsets. This is shown in Figure 12.

Whenever a customer purchases some items, strongly related items can be suggested. Using support formula represented in the previous Chapter, statistical significance, which distinguishes if the transaction is important enough to consider, is found. Then the next items can be displayed with confidence. For the previous example, when a customer purchases I2, then item I1 and I7 can be suggested with confidence. Figure 12 shows the steps.

![Diagram](image)

**Figure 12: Steps for browsing association rules**
For the implementation of this sequential search on the web, the following architecture is designed.

![Diagram showing the architecture of the system]

*Figure 13: Architecture of the system*

This architecture explains how to implement mining association rules to web-based sequential search. The client connects with the Web server through the HTTP protocol using a web browser. The web Server\(^6\) delivers static HTML pages to the client, namely index.html file. After the customer chooses a category, the value is translated to Java Server Page

---

\(^6\) For implementation, MS Web Server IIS is used
(JSP) file via web server. The JSP engine\(^7\) contacts the database to retrieve or update data.

![Figure 14: The interface of the sample e-commerce web site](image)

As a result, the JSP engine accumulates dynamic web pages and reports the result to the Web server, which, in turn, delivers the information to the client. Figure 14 shows a static HTML page as an interface. The next figure shows the results of a web page that is made by connecting DBMS using JSP. (See appendix B)

---

\(^7\) For this implementation, JSP engine Tomcat is used
Figure 15: The results of dynamic web pages using JSP

Suppose a customer has chosen item ‘10012’; then this value is translated to the JSP engine via a web server. Before this query, the JSP engine need to connect to database using JDBC and ODBC connections. Then, using query, equivalent rows can be found in the lattice Table that contain all concept information.

The next line shows how to use this parameter to find exact item.

“SELECT ChildSet, OBJECTSETSIZE FROM LatticeTable WHERE AttributeSet = ” + choice. (See appendix B)

Using the lattice Table, some items are suggested in declining order of confidence in Figure 16. In this example, ‘10020’ (13.3%) and some related items are recommended. After choosing one more item, in this
example, if item ‘10020’ is chosen, then a high confidence item will also be suggested.

Figure 16: A screen capture with high confidence item suggested

Figure 17: Recommended items order by confidence
Whenever one item is chosen, before all items of sub-concept are suggested, *minimum support* and *minimum confidence* can be easily examined using the "if" condition. Any item below these two criteria will be excluded among suggested items.
Chapter 6 Conclusions and Future work

In this thesis, we presented an alternative approach to discover association rules based on Formal Concept Analysis. Given a binary relation, a concept consists of transactions and items. All frequent itemsets are determined by the set of frequent concepts. Support and confidence can be obtained by considering frequency of the itemset using lattice structure, which is a structure of the set of frequent concepts. These frequent itemsets can be calculated only by comparing hierarchically linked concepts, in the concept lattice, instead of calculating all frequent itemsets in most current algorithms. This method can significantly reduce the cost of generating association rules.

FCA-based approach can dynamically find association rules since it works at any level of minimum support and minimum confidence. From this point of view, it is clear that this approach is more flexible than the large itemset method.

We also showed that the many-valued context of Formal Concept Analysis could be used for finding more detailed quantitative information.

In addition, the many-valued context could be approached to categorization of items. It could provide generalized confidence. Further
study of this matter is remained as future work.
References


[9] J. S. Deogun, V. V. Raghavan, A. Sarkar and H. Sever, *"Data mining: Research trends, challenges, and applications,"* Roughs Sets and Data


APPENDIX A: Source code for Data Structure

1. Concept.java

//Concept.java makes an object that contains 4 Vectors. It will be used as a concept object.

import java.util.Vector;

public class Concept {
    public Vector objectSet = new Vector(1, 1);
    public Vector attributeSet = new Vector(1, 1);
    public Vector parentSet = new Vector(1, 1);
    public Vector childSet = new Vector(1, 1);

    public Concept(Vector v1, Vector v2, Vector v3, Vector v4) {
        objectSet = v1;
        attributeSet = v2;
        parentSet = v3;
        childSet = v4;
    }
}

2. createContext.java

// createContext.java creates a formal context table.

import java.sql.*;

class CreateContext {
    public static void main(String args[]) {
        String url = "jdbc:odbc:kim";
        Connection con;
        String createString;
        createString = "create table CONTEXT1 " + "(TID INTEGER, ";

        for(long i=10001; i<10100; i++) // total item numbers
        {
            createString = createString + "P " + String.valueOf(i) + " INTEGER, ";
        }

        createString = createString + "P10100 INTEGER")";
        Statement stmt;

        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        } catch(ClassNotFoundException e) {
            System.err.println("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }

        try {
            con = DriverManager.getConnection(url, "myuser", "password");
            stmt = con.createStatement();
            stmt.executeUpdate(createString);
            stmt.close();
            con.close();
        } catch(SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
        }
    }
}
3. makeContext.java

// makeContext.java initializes a formal context table to '0'.

import java.sql.*;

public class MakeContext {

    public MakeContext() {

        MyRandom R = new MyRandom();

        String url = "jdbc:odbc:kim";
        java.sql.Connection con = null;

        Statement statement = null;

        String query = "SELECT * FROM TRANSACTION1";
        ResultSet results = null;
        ResultSetMetaData rsmd = null;

        int result;

        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        } catch (Exception e) {
            System.err.println("Exception while loading JDBC-ODBC bridge driver");
        }

        try {
            con = DriverManager.getConnection(url, "myuser", "password");
        } catch (SQLException e) {
            System.err.println("SQLException while connecting to the database");
        }

        try {
            statement = con.createStatement();
        } catch (SQLException e) {
            System.err.println("SQLException while creating statement");
        }

        try {
            results = statement.executeQuery(query);
        } catch (SQLException e) {
            System.err.println("SQLException while executing query");
        }

        String zero = "0, ";
        for(int i=0; i<98; i++)
            zero=zero+"0, ";
        zero= zero+ "0:"

        try {
            for(int i=0; i<R.ObjectNum; i++)// you need to change whenever you change tid manually
            
        }
int tid = i + 101;
String update = "INSERT INTO CONTEXT VALUES ("+tid+", "+zero+");
result = statement.executeUpdate(update);
}

} catch (SQLException e) {
    System.err.println("SQLException while examining ResultSet");
}

try {
    statement.close();
    con.close();
} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}

}

public static void main(String[] args) {
    MakeContext m = new MakeContext();
}
}
4. fillContext.java

// fillContext.java update a formal context table to '1' value in which each transaction bought that item
// Transferring the item from transaction table.

import java.sql.*;
import java.util.Vector;

public class FillContext {
    public static void main(String[] args) {
        Vector TidSet = new Vector(50, 1);
        Vector itemSet = new Vector(50, 1);

        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        } catch (Exception e) {
            System.err.println("Exception while loading JDBC-ODBC bridge driver");
        }

        String url = "jdbc:odbc:kim";
        java.sql.Connection con = null;

        try {
            con = DriverManager.getConnection(url, "myuser", "password");
        } catch (SQLException e) {
            System.err.println("SQLException while connecting to the database");
        }

        Statement statement = null;
        try {
            statement = con.createStatement();
        } catch (SQLException e) {
            System.err.println("SQLException while creating statement");
        }

        String query = "SELECT * FROM TRANSACTION 1";
        ResultSet results = null;
        try {
            results = statement.executeQuery(query);
        } catch (SQLException e) {
            System.err.println("SQLException while executing query");
        }

        int result = -1;
        try {
            while(results.next()) {
                int tid = results.getInt(1);
                String tms = results.getString(2);
                String tid2 = String.valueOf(tid);
                TidSet.addElement(tid);
                itemSet.addElement(tms);
            }
        } catch (SQLException e) {
            System.err.println(e.getMessage());
            System.err.println("SQLException while examining ResultSet");
        }
    }
}
try {
    statement.close();
    con.close();
} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}

try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (Exception e) {
    System.err.println("Exception while loading JDBC-ODBC bridge driver");
}

try {
    con = DriverManager.getConnection(url, "myuser", "password");
} catch (SQLException e) {
    System.err.println("SQLException while connecting to the database");
}

try {
    statement = con.createStatement();
} catch (SQLException e) {
    System.err.println("SQLException while creating statement");
}

try {
    results = statement.executeQuery(query);
} catch (SQLException e) {
    System.err.println("SQLException while executing query");
}

try {
    for(int i=0; i < itemSet.size(); i++) {
        int tid = Integer.parseInt(String.valueOf(itemSet.get(i)));
        int tms = Integer.parseInt(String.valueOf(itemSet.get(i)));
        System.out.println(tid);
        System.out.println(tms);
        String update = "UPDATE CONTEXT SET " + "+tms+" = " + tms + " WHERE TID like " + "+tid+";";
        result = statement.executeUpdate(update);
    }
} catch (SQLException e) {
    System.err.println(e.getMessage());
    System.err.println("SQLException while examining ResultSet");
}

try {
    statement.close();
    con.close();
} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}
5. rawData.java

// rawData.java stores each value in memory from a formal context table using "Concept" object.

import java.sql.*;
import java.util.Vector;

public class rawData {
    public Vector FCA = null;
    public int NoOfRow = 0;
    public int NoOfItem = 0;

    public rawData()
    {
        getRawData();
    }

    public void getRawData()
    {
        String [] columnName = null;
        Vector attribute = new Vector(1,1);

        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        } catch (Exception e) {
            System.err.println("Exception while loading JDBC-ODBC bridge driver");
        }

        String url = "jdbc:odbc:kim";
        java.sql.Connection con = null;

        try {
            con = DriverManager.getConnection(url, "myuser", "password");
        } catch (SQLException e) {
            System.err.println("SQLException while connecting to the database");
        }

        Statement statement = null;
        try {
            statement = con.createStatement();
        } catch (SQLException e) {
            System.err.println("SQLException while creating statement");
        }

        String query = "SELECT * FROM CONCEPT1";
        ResultSet results = null;
        ResultSetMetaData rsmd = null;

        try {
            results = statement.executeQuery(query);
        } catch (SQLException e) {
            System.err.println("SQLException while executing query");
        }

        int result;
        try {
            // Code continues here...
        }
    }
}
rsmd = results.getMetaData();
NoOfItem = rsmd.getColumnCount();
while(results.next()) {
    NoOfRow = results.getRow();
}
} catch (SQLException e) {
    System.err.println(e.getMessage());
    System.err.println("SQLException while examining ResultSet");
} System.out.println("NoOfItem: "+NoOfItem+" NoOfRow: "+NoOfRow);

columnName = new String[NoOfItem-1];
FCA = new Vector[10,1];

try {
    results = statement.executeQuery(query);
} catch (SQLException e) {
    System.err.println("SQLException while executing query");
}

try {
    for(int i=0; i < NoOfItem-1; i++) {
        rsmd = results.getMetaData();
        columnName[i] = rsmd.getColumnName(i+2);
    }
    int i=0;
    while(results.next()) {
        int k;
        String s = results.getString(1);
        Vector objects = new Vector(1,1);
        objects.addElement(s);
        attribute = new Vector(1,1);
        for(k=0; k < NoOfItem-1; k++) {
            if(results.getInt(k+2) == 1)
                {String n = columnName[k];
                attribute.addElement(n);
            }
        } FCA.addElement(new Concept(objects,attribute,null,null));
        i++;
    }
} catch (SQLException e) {
    System.err.println(e.getMessage());
    System.err.println("SQLException while examining ResultSet");
}

for(int i=0; i < NoOfRow; i++) {
    System.out.println("" +
    for(int x=0; x < ((Concept)FCA.elementAt(i)).objectSet.size(); x++) {
        System.out.println(String.valueOf(((Concept)FCA.elementAt(i)).objectSet.get(x)) +
                        System.out.println("" +
    for(int x=0; x < ((Concept)FCA.elementAt(i)).attributeSet.size(); x++) {
        System.out.println(String.valueOf(((Concept)FCA.elementAt(i)).attributeSet.get(x)) +
                        System.out.println("" +
}
try {
    statement.close();
    con.close();
} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}
public class makeConcept {} {
    public Vector FCAconcept = null;
    public int NoOfRow = 0;
    public int NoOfItem = 0;
    Vector grandChildren = new Vector(1, 1);
    Vector grandParents = new Vector(1, 1);
    Vector allParents = new Vector(1, 1);
    Vector label = new Vector(1, 1);

    public makeConcept() {
        FCAconcept = new Vector(10, 1);
        getAtomic();
        getSubConcept();
        getParentChild();
        getConfidence();
    }

    public void getAtomic() {
        RawData a = new RawData();
        NoOfRow = a.NoOfRow;
        int pointer = 0;

        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            catch (Exception e) {
                System.err.println("Exception while loading JDBC-ODBC bridge driver");
            }
            String url = "jdbc:odbc:kim";
            java.sql.Connection con = null;

            try {
                con = DriverManager.getConnection(url, "myuser", "password");
                catch (SQLException e) {
                    System.err.println("SQLException while connecting to the database");
                }
            }
        }

        Statement statement = null;
        try {
            statement = con.createStatement();
            catch (SQLException e) {
                System.err.println("SQLException while creating statement");
            }
        }

        // Finding atomic concept during this for loop
        for (pointer = 0; pointer < a.NoOfRow; pointer++) {
            //...
String where = "";
where = "WHERE " +
    ((String)((Concept)FCA.elementAt(pointer)).attributeSet.elementAt(0) + " like '1'";
    
for (int j = 0; j < ((Concept)FCA.elementAt(pointer)).attributeSet.size() - 1; j++)
    
where = where + " and " + ((String)((Concept)FCA.elementAt(pointer)).attributeSet.elementAt(j + 1) + " like '1'";
    
String query = "SELECT TID FROM CONTEXT1 " + where;
ResultSet results = null;
    
try {
    results = statement.executeQuery(query);
} catch (SQLException e) {
    System.err.println(e.getMessage());
    System.err.println("SQLException while executing query");
}
    
Vector v = new Vector(1, 1);
    try {
        while (results.next()) {
            String obj = results.getString(1);
            v.addElement(obj);
        }
    } catch (SQLException e) {
        System.err.println(e.getMessage());
        System.err.println("SQLException while examining ResultSet");
    }
    
Vector tmp1 = new Vector(1, 1); // for parentSet
Vector tmp2 = new Vector(1, 1); // for childSet
    
Concept c1 = new Concept (v, ((Concept)FCA.elementAt(pointer)).attributeSet, tmp1, tmp2);
    int signal = 0;
    
for (int w = 0; w < FCAConcept.size(); w++)
    
if (c1.attributeSet.equals(((Concept)FCAConcept.elementAt(w)).attributeSet))
    
break;
    else signal++;
    
if (signal == FCAConcept.size())
    
FCAConcept.addElement(c1);
    
} // for pointer
    try {
        statement.close();
        con.close();
    } catch (SQLException e) {
        System.err.println("SQLException while closing database");
    }
} // getAtomic
// Finding sub_concept
public void getSubConcept()
{
    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    } catch (Exception e) {
        System.err.println("Exception while loading JDBC-ODBC bridge driver");
    }

    String url = "jdbc:odbc:tim";
    java.sql.Connection con = null;

    try {
        con = DriverManager.getConnection(url, "myuser", "password");
    } catch (SQLException e) {
        System.err.println("SQLException while connecting to the database");
    }

    Statement statement = null;
    try {
        statement = con.createStatement();
    } catch (SQLException e) {
        System.err.println("SQLException while creating statement");
    }

    for(int i=0;i<FCAsize;i++)
    {
        for(int z=1;z<((FCAsize)-i);z++)
        {
            String where = "WHERE P10001 < 2";
            String s = "";
            Vector att = new Vector(1, 1);
            for(int j=0;j<((Concept)FCAelementA1[i]).attributeSet.size();j++)
            {
                for(int k=0;k<((Concept)FCAelementA1[i+z]).attributeSet.size();k++)
                {
                    if (((Concept)FCAelementA1[i]).attributeSet.elementA1[i].equals(((Concept)FCAelementA1[i+z]).attributeSet.elementA1[k]))
                    {
                        where = where + " and " + ((String)((Concept)FCAelementA1[i]).attributeSet.elementA1[i]).equals("like 1") + ((String)((Concept)FCAelementA1[i+z]).attributeSet.elementA1[k]);
                        att.addElement(s);
                    }
                }
            }
        }
        if(!where.equals("WHERE P10001 < 2"))
        {
            String query = "SELECT TID FROM CONTEXT1 " + where;
            ResultSet results = null;
            try {
                results = statement.executeQuery(query);
            } catch (SQLException e) {
                System.err.println(e.getMessage());
                System.err.println("SQLException while executing query");
            }
        }
    }
}
Vector v = new Vector(1, 1);
Vector p = new Vector(1, 1);
Vector ch = new Vector(1, 1);
try {
    while(results.next()) {
        String obj = results.getString(1);
        v.addElement(obj);
    }
} catch (SQLException e) {
    System.err.println(e.getMessage());
    System.err.println("SQLException while examining ResultSet");
}
Concept c = new Concept(v, att, p, ch);

int flag = 0;
for(int w = 0; w < FCAConcept.size(); w++)
    // compare whether there is same attributeSet
    if (c.getAttributeSet.equals(((Concept)FCAConcept.elementAt(w)).getAttributeSet()))
        // if it is include relation, it will be break.
        break;
    else flag++;
    } // for

if (flag == FCAConcept.size()) // c.getAttributeSet doesn't same with any set.
{
    FCAConcept.addElement(c);
    System.out.println("Now, Concept No." + (FCAConcept.size() - 1) + " is added.");
    reCheck((Concept)FCAConcept.elementAt(i));
    } // if
} // for i
} // for z
} // for

try {
    statement.close();
    con.close();
} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}

} // getSubConcept

// Finding parentSet & childSet during this for loop

public void getParentChild()
{
    System.out.println("Now, I'm finding parentSet & childSet");
    for(int i = 0; i < FCAConcept.size(); i++)
    {
        for(int z = 1; z <= (FCAConcept.size() - i); z++)
        {
            int flag = 0;
            for(int j = 0; j < ((Concept)FCAConcept.elementAt(i)).getAttributeSet.size(); j++)
            {
                for(int k = 0; k < ((Concept)FCAConcept.elementAt(i + z)).getAttributeSet.size(); k++)
                {

```
if (((Concept)FCAconcept.elementAt(i).attributeSet.elementAt(j).equals((Concept)FCAconcept.elementAt(i + z).attributeSet.elementAt(k)))
    
    flag++;
}

};//for k

}//for j

if (flag == ((Concept)FCAconcept.elementAt(i).attributeSet.size()))
{
    if (((Concept)FCAconcept.elementAt(i).attributeSet.size()) <
        ((Concept)FCAconcept.elementAt(i + z).attributeSet.size()))
    {
        ((Concept)FCAconcept.elementAt(i + z).parentSet.addElement(String.valueOf(i));
        ((Concept)FCAconcept.elementAt(i).childSet.addElement(String.valueOf(i + z));
    }
}

if (flag == ((Concept)FCAconcept.elementAt(i + z).attributeSet.size()))
{
    if (((Concept)FCAconcept.elementAt(i).attributeSet.size()) >
        ((Concept)FCAconcept.elementAt(i - z).attributeSet.size()))
    {
        ((Concept)FCAconcept.elementAt(i).parentSet.addElement(String.valueOf(i + z));
        ((Concept)FCAconcept.elementAt(i + z).childSet.addElement(String.valueOf(i));
    }
}

}//for i

// Decide parent and grand parent during this for loop

System.out.println("Now, I'm distinguishing parent and grand parent");

int flag = 0;
for(int j=0;j< FCAconcept.size(); j++)
{
    for(int i=0;i< ((Concept)FCAconcept.elementAt(i).parentSet.size()); i++)
    {
        if(flag == 0){ j = 0; }
        flag = 0;
        for(int z = 1; z < ((Concept)FCAconcept.elementAt(i).parentSet.size()); z++)
        {
            int flag = 0;
            for(int k=0;k< ((Concept)FCAconcept.elementAt(integer.parseInt(String)(((Concept)FCAconcept.elementAt(i).parentSet.elementAt(i)))).attributeSet.size(); k++)
            {
                for(int x=0;x< ((Concept)FCAconcept.elementAt(integer.parseInt(String)(((Concept)FCAconcept.elementAt(i).parentSet.elementAt(i))).attributeSet.elementAt(j).equals(((Concept)FCAconcept.elementAt(integer.parseInt(String)(((Concept)FCAconcept.elementAt(i).parentSet.elementAt(j + z)).attributeSet.(elementAt(x))
                
                flag++;;
            }
        };//for x
    };//for k

if (flag == ((Concept)FCAconcept.elementAt(integer.parseInt(String)(((Concept)
FCAconcept.elementAt[i].parentSet.elementAt[i].size()]
        {  
            if(flag != 0) { j = 0; }  
            if(flag = 0)
            for(int z=1 ; z < ((Concept)FCAconcept.elementAt[i]).parentSet.size() - j; z++)
            {  
                int flag = 0;
                for(int k=0; k < ((Concept)FCAconcept.elementAt[i].parentSet.elementAt[j]).size(); k++)
                {  
                    int flag = 0;
                    for(int x=0; x < ((Concept)FCAconcept.elementAt[i].parentSet.elementAt[j].size(); x++)
                    {  
                        if (((Concept)FCAconcept.elementAt[i].parentSet.elementAt[j].size(); x++)
                        {  
                            flag++;
                        }  
                    }  
                }  
            }  
            if(flag == ((Concept)FCAconcept.elementAt[i].parentSet.elementAt[j].size();)
            {  
                (Concept)FCAconcept.elementAt[i].parentSet.removeElementAt(j + z);
                flag++;
            }  
        }  
    }  
  
  // Decide child and grand child during this for loop
  
  System.out.println("Now, I'm distinguishing child and grand child");
  int bflag = 0;
  for(int i=0;i < FCAconcept.size(); i++)
  {  
      for(int j=0; j < ((Concept)FCAconcept.elementAt[i].childSet.size(); j++)
      {  
          if(bflag = 0) { j = 0; }  
          if(bflag = 0)
          for(int z=1; z < ((Concept)FCAconcept.elementAt[i].childSet.size() - j; z++)
          {  
              // Some code here...  
          }  
      }  
  }  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  
}  

int flag = 0;
for(int k=0;k< [(Concept)FCAconcept.elementAt([Integer.parseInt(String)][(Concept)FCAconcept.elementAt([i]).childSet.elementAt([j])])] )attributeSet.size(); k++);//for4
{
for(int x=0;x< [(Concept)FCAconcept.elementAt([Integer.parseInt(String)][(Concept)FCAconcept.elementAt([i]).childSet.elementAt([j + z])])] )attributeSet.size(); x++);//for5
{
    if ([(Concept)FCAconcept.elementAt([Integer.parseInt(String)][(Concept)FCAconcept.elementAt([i]).childSet.elementAt([j])])] )attributeSet.elementAt([k]).
equals([(Concept)FCAconcept.elementAt([Integer.parseInt(String)][(Concept)FCAconcept.elementAt([i]).childSet.elementAt([j + z])])] )attributeSet.elementAt([x]) ] )
    {
        flag++;
    }
};//for x
};//for k

if (flag == [(Concept)FCAconcept.elementAt([Integer.parseInt(String)][(Concept)FCAconcept.elementAt([i]).childSet.elementAt([j + z])])] )attributeSet.size())
{
    [(Concept)FCAconcept.elementAt([i]).childSet.removeElementAt([j])] ;
    bflag++;
}
};//for z
});//for i

for(int i=0;i< FCAconcept.size(); i++)
{
for(int j=0;j< [(Concept)FCAconcept.elementAt([i]).childSet.size(); i++ )]
{
    if (bflag != 0{[ j = 0; } bflag = 0;
    int z = 1 z < [(Concept)FCAconcept.elementAt([i]).childSet.size(); ] ] ; z++ )
    {
        int flag = 0;
        for(int k=0;k< [(Concept)FCAconcept.elementAt([Integer.parseInt(String)][(Concept)FCAconcept.elementAt([i]).childSet.elementAt([j])])] )attributeSet.size(); k++);//for4
        {
            for(int x=0;x< [(Concept)FCAconcept.elementAt([Integer.parseInt(String)][(Concept)FCAconcept.elementAt([i]).childSet.elementAt([j + z])])] )attributeSet.size(); x++);//for5
                if ([(Concept)FCAconcept.elementAt([Integer.parseInt(String)][(Concept)FCAconcept.elementAt([i]).childSet.elementAt([j])])] )attributeSet.elementAt([k]).
equals([(Concept)FCAconcept.elementAt([Integer.parseInt(String)][(Concept)FCAconcept.elementAt([i]).childSet.elementAt([j + z])])] )attributeSet.elementAt([x]) ] )
                    {
                        flag++;
                    }
            };//for x
        };//for k
    }
};//for z

}
System.out.println("Now, I'm sorting childSet");

sorting(FCAconcept);

// if there isn't any attribute connect with "top" "bottom"
for(int i=0; i < FCAconcept.size(); i++)
{
    if(((Concept)FCAconcept.elementAt(i)).parentSet.size() == 0)
    {
        ((Concept)FCAconcept.elementAt(i)).parentSet.addElement("10000");
    }
    if(((Concept)FCAconcept.elementAt(i)).childSet.size() == 0)
    {
        ((Concept)FCAconcept.elementAt(i)).childSet.addElement("10001");
    }
}

public void getConfidence()
{
    findLabeledConcept();

    String createString2 = "create table EXAMPLECONFIDENCE " +
        "(" +
        "CONCEPTNO INTEGER, " +
        "OBJECTSETNO1 INTEGER, " +
        "ATTRIBUTESET1 VARCHAR(100), " +
        "ANTecedent VARCHAR(100), " +
        "COMParedConcept INTEGER, " +
        "OBJECTSETNO2 INTEGER, " +
        "ATTRIBUTESET2 VARCHAR(100), " +
        "consequent VARCHAR(100), " +
        "Support FLOAT(5), " +
        "Confidence FLOAT(5)" +
        ");";

    Statement stmt = null;

    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    } catch(ClassNotFoundException e) {
        System.err.println("ClassNotFoundException");
        System.err.println(e.getMessage());
    }

    try {
        con = DriverManager.getConnection(url, "myuser", "password");
        stmt = con.createStatement();
        stmt.executeUpdate(createString2);
        stmt.close();
        con.close();
    } catch(SQLException ex) {
        System.err.println("SQLException: " + ex.getMessage());
    }

    System.out.println("Now, Confidence table is setting");

    // insert into table values
try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    } catch (java.lang.ClassNotFoundException e) {
        System.err.println("Exception while loading JDBC-ODBC bridge driver");
        System.err.println(e.getMessage());
    }

    String url = "jdbc:odbc:kim";
    java.sql.Connection con = null;
    try {
        con = DriverManager.getConnection(url, "myuser", "password");
    } catch (SQLException e) {
        System.err.println("SQLException while connecting to the database");
    }

    try {
        stmt = con.createStatement();
    } catch (SQLException e) {
        System.err.println("SQLException while creating statement");
    }

    Vector v0 = new Vector(1, 1);
    Vector v1 = new Vector(1, 1);
    Vector v2 = new Vector(1, 1);
    Vector v3 = new Vector(1, 1);
    Vector v4 = new Vector(1, 1);
    Vector v5 = new Vector(1, 1);
    Vector v6 = new Vector(1, 1);
    Vector v7 = new Vector(1, 1);
    Vector v8 = new Vector(1, 1);
    Vector v9 = new Vector(1, 1);

    try {
        for(int i=0; i < FCAconcept.size(); i++)
        {
            allParents.removeAllElements();

            for(int lb =0; lb< label.size(); lb++)
            {
                if (i == ((LabelConcept)label.elementAt(lb)).conceptNo)
                {
                    findAllParents((Concept)((Concept)FCAconcept.elementAt(i)));
                    String attribute1 = "";
                    String antecedent = "";
                    float support = 0;

                    //make first concept's attributeSet to one String
                    for(int y=0; y< ((Concept)FCAconcept.elementAt(i)).attributeSet.size()-1; y++)
                    {
                        attribute1 = attribute1 + (String)(((Concept)FCAconcept.elementAt(i)).
                        attributeSet.elementAt(y)) + " ";
                    }
                    attribute1 = attribute1 + (String)(((Concept)FCAconcept.elementAt(i)).attributeSet.
                    elementAt(((Concept)FCAconcept.elementAt(i)).attributeSet.size()-1));

                    System.out.println("attribute1:");
                    System.out.println(attribute1);
                    System.out.println("antecedent:");
                    System.out.println(antecedent);
                    System.out.println("support:");
                    System.out.println(support);
                }
            }
        }
    }

    try {
        } catch (SQLException e) {
            System.err.println("SQLException while creating statement");
        }
    }

    try {
        con = DriverManager.getConnection(url, "myuser", "password");
    } catch (SQLException e) {
        System.err.println("SQLException while connecting to the database");
    }

    try {
        stmt = con.createStatement();
    } catch (SQLException e) {
        System.err.println("SQLException while creating statement");
    }

    Vector v0 = new Vector(1, 1);
    Vector v1 = new Vector(1, 1);
    Vector v2 = new Vector(1, 1);
    Vector v3 = new Vector(1, 1);
    Vector v4 = new Vector(1, 1);
    Vector v5 = new Vector(1, 1);
    Vector v6 = new Vector(1, 1);
    Vector v7 = new Vector(1, 1);
    Vector v8 = new Vector(1, 1);
    Vector v9 = new Vector(1, 1);

    try {
        for(int i=0; i < FCAconcept.size(); i++)
        {
            allParents.removeAllElements();

            for(int lb =0; lb< label.size(); lb++)
            {
                if (i == ((LabelConcept)label.elementAt(lb)).conceptNo)
                {
                    findAllParents((Concept)((Concept)FCAconcept.elementAt(i)));
                    String attribute1 = "";
                    String antecedent = "";
                    float support = 0;

                    //make first concept's attributeSet to one String
                    for(int y=0; y< ((Concept)FCAconcept.elementAt(i)).attributeSet.size()-1; y++)
                    {
                        attribute1 = attribute1 + (String)(((Concept)FCAconcept.elementAt(i)).
                        attributeSet.elementAt(y)) + " ";
                    }
                    attribute1 = attribute1 + (String)(((Concept)FCAconcept.elementAt(i)).attributeSet.
                    elementAt(((Concept)FCAconcept.elementAt(i)).attributeSet.size()-1));

                    System.out.println("attribute1:");
                    System.out.println(attribute1);
                    System.out.println("antecedent:");
                    System.out.println(antecedent);
                    System.out.println("support:");
                    System.out.println(support);
                }
            }
        }
    }

    try {
        } catch (SQLException e) {
            System.err.println("SQLException while creating statement");
        }
    }

    Vector v0 = new Vector(1, 1);
    Vector v1 = new Vector(1, 1);
    Vector v2 = new Vector(1, 1);
    Vector v3 = new Vector(1, 1);
    Vector v4 = new Vector(1, 1);
    Vector v5 = new Vector(1, 1);
    Vector v6 = new Vector(1, 1);
    Vector v7 = new Vector(1, 1);
    Vector v8 = new Vector(1, 1);
    Vector v9 = new Vector(1, 1);

    try {
        for(int i=0; i < FCAconcept.size(); i++)
        {
            allParents.removeAllElements();

            for(int lb =0; lb< label.size(); lb++)
            {
                if (i == ((LabelConcept)label.elementAt(lb)).conceptNo)
                {
                    findAllParents((Concept)((Concept)FCAconcept.elementAt(i)));
                    String attribute1 = "";
                    String antecedent = "";
                    float support = 0;

                    //make first concept's attributeSet to one String
                    for(int y=0; y< ((Concept)FCAconcept.elementAt(i)).attributeSet.size()-1; y++)
                    {
                        attribute1 = attribute1 + (String)(((Concept)FCAconcept.elementAt(i)).
                        attributeSet.elementAt(y)) + " ";
                    }
                    attribute1 = attribute1 + (String)(((Concept)FCAconcept.elementAt(i)).attributeSet.
                    elementAt(((Concept)FCAconcept.elementAt(i)).attributeSet.size()-1));

                    System.out.println("attribute1:");
                    System.out.println(attribute1);
                    System.out.println("antecedent:");
                    System.out.println(antecedent);
                    System.out.println("support:");
                    System.out.println(support);
                }
            }
        }
    }
for(int y=0;y<((LabelConcept)label.elementAt(lb)).attName.size()-1;y++)
{
    antecedent = antecedent + (String)(((LabelConcept)label.elementAt(lb)).
attName.elementAt(y)) + "=";
}

antecedent = antecedent + (String)(((LabelConcept)label.elementAt(lb)).attName.
elementAt(((LabelConcept)label.elementAt(lb)).attName.size()-1));

for(int q=0;q<allParents.size();q++)
{
    String attribute2 = "=";
    String consequent = "=";

    int t = Integer.parseInt(String.valueOf(allParents.elementAt(q)));
    for(int z=0;z<((Concept)FCAconcept.elementAt(t)).attributeSet.size()-1;z++)
    {
        attribute2 = attribute2 + (String)(((Concept)FCAconcept.elementAt(t)).
attributeSet.elementAt(z)) + "=";
    }

    attribute2 = attribute2 + (String)(((Concept)FCAconcept.elementAt(t)).attributeSet.
elementAt(((Concept)FCAconcept.elementAt(t)).attributeSet.size()-1));

    consequent = attribute2;
    v0.addElement(String.valueOf(i));
    v1.addElement(String.valueOf(((Concept)FCAconcept.elementAt(i)).objectSet.size()));
    v2.addElement(attribute1);
    v3.addElement(antecedent);
    v4.addElement(String.valueOf(allParents.elementAt(q)));
    v5.addElement(String.valueOf(((Concept)FCAconcept.elementAt(t)).objectSet.
size()));
    v6.addElement(attribute2);
    v7.addElement(consequent);

    support = (float)(((Concept)FCAconcept.elementAt(i)).objectSet.size()) / (float)NoOfRow;

    v8.addElement(String.valueOf(support));
    v9.addElement(String.valueOf((float)1));
}
}

//first find all children and grand children
grandChildren.removeAllElements();
findAllGrandChildren(((Concept)FCAconcept.elementAt(i)));

if (!((grandChildren.isEmpty())))
{
    String attribute1 = "=";
    String antecedent = "=";
    float support;
    float childConfidence;

    //make first concept's attributeSet to one String
    for(int y=0;y<((Concept)FCAconcept.elementAt(i)).attributeSet.size()-1;y++)
    {
        attribute1 = attribute1 + (String)(((Concept)FCAconcept.elementAt(i)).attributeSet.
elementAt(y)) + "=";
    }

    attribute1 = attribute1 + (String)(((Concept)FCAconcept.elementAt(i)).attributeSet.
elementAt((Concept)FCAconcept.elementAt(i).attributeSet.size() - 1));

antecedent = attribute1;

// make complex concepts' attributeSet to one String
for(int q=0; q < grandChildren.size(); q++)
{
    String attribute2 = "";
    String consequent = "";

    int tmpInt = Integer.parseInt(String.valueOf((grandChildren.elementAt(q))));
    for(int z=0; z < ((Concept)FCAconcept.elementAt(tmpInt)).attributeSet.size()-1; z++)
    {
        attribute2 = attribute2 + (String)(((Concept)FCAconcept.elementAt(tmpInt)).
        attributeSet.elementAt(z)) + " ";
    }
    attribute2 = attribute2 + (String)(((Concept)FCAconcept.elementAt(tmpInt)).attributeSet.
    elementAt(((Concept)FCAconcept.elementAt(tmpInt)).attributeSet.size()-1));

    consequent = subtract(((Concept)FCAconcept.elementAt(tmpInt)).attributeSet,
    (Concept)FCAconcept.elementAt(i).attributeSet);

    v0.addElement(String.valueOf(i));
    v1.addElement(String.valueOf(((Concept)FCAconcept.elementAt(i)).objectSet.size()));
    v2.addElement(attribute1);
    v3.addElement(antecedent);
    v4.addElement(String.valueOf((grandChildren.elementAt(q))));
    v5.addElement(String.valueOf(((Concept)FCAconcept.elementAt(tmpInt)).objectSet.size()));
    v6.addElement(attribute2);
    v7.addElement(consequent);

    childConfidence = float(((Concept)FCAconcept.elementAt(tmpInt)).objectSet.size()) / 
    float(((Concept)FCAconcept.elementAt(i)).objectSet.size());

    support = float(((Concept)FCAconcept.elementAt(tmpInt)).objectSet.size()) / 
    float(NOOFRow);

    v8.addElement(String.valueOf(support));
    v9.addElement(String.valueOf(childConfidence));

} // for q
} // for i

int result=0;

for(int w=0; w < v0.size(); w++)
{
    result=stmt.executeUpdate("INSERT INTO EXAMPLECONFIDENCE VALUES( " +
    v0.elementAt(w)+ ", " +v1.elementAt(w)+ ", " +v2.elementAt(w)+ ", " +v3.elementAt(w)+
    ", " +v4.elementAt(w)+ ", " +v5.elementAt(w)+ ", " +v6.elementAt(w)+ ", " +v7.elementAt(w)+
    ", " +v8.elementAt(w)+ ", " +v9.elementAt(w)+ ");
}
stmt.close();
con.close();

} catch(SQLException ex) {

90
System.err.println("SQLException: " + ex.getMessage());
}

private void reCheck(int index, Concept NEW) {
    int SC = index;
    try {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        catch (Exception e) {
            System.err.println("Exception while loading JDBC-ODBC bridge driver");
        }

        String url = "jdbc:odbc:kim";
        java.sql.Connection con = null;
        try {
            con = DriverManager.getConnection(url, "myuser", "password");
        } catch (SQLException e) {
            System.err.println("SQLException while connecting to the database");
        }

        Statement statement = null;
        try {
            statement = con.createStatement();
        } catch (SQLException e) {
            System.err.println("SQLException while creating statement");
        }

        for(int a=0; a < SC; a++) {
            String where = "WHERE P10001 < 2 ";
            String s = "";
            Vector att = new Vector(1, 1);

            for(int j=0;j < ((Concept)FCAConcept.elementAt(a)).attributeSet.size(); j++) {
                for(int k=0;k < ((Concept)NEW).attributeSet.size(); k++) {
                    if (((Concept)FCAConcept.elementAt(a)).attributeSet.elementAt(k)).equals(((Concept)NEW).attributeSet.elementAt(j)) {
                        where = where + " and " + ((String)((Concept)FCAConcept.elementAt(a)).attributeSet.elementAt(j)) + " like \\
                        *";
                        s = (String)(((Concept)NEW).attributeSet.elementAt(k));
                        att.addElement(s);
                        }
                    }
            }
        }

        if (!where.equals("WHERE P10001 < 2 ")) {
            String query = "SELECT TID FROM CONTEXT1 " + where;
            ResultSet results = null;
            try {
                results = statement.executeQuery(query);
            } catch (SQLException e) {
                System.err.println(e.getMessage());
            }
        }
    }
}
System.err.println("SQLException while executing query");
}

Vector v = new Vector(1, 1);
Vector p = new Vector(1, 1);
Vector ch = new Vector(1, 1);
try {
    while(results.next()) {
        String obj = results.getString(1);
        v.addElement(obj);
    }
} catch (SQLException e) {
    System.err.println(e.getMessage());
    System.err.println("SQLException while examining ResultSet");
}
Concept c = new Concept(v.att.p.ch);
int flag = 0;

for(int w=0; w<FCAConcept.size(); w++) //compare whether there is same attributeSet
{
    if (c.attributeSet.equals(Arrays.asList(FCAConcept.elementAt(w).attributeSet)))
        //if it is include relation, it will be break.
        break;
    else flag++;
}

if (flag == FCAConcept.size()) //c.attributeSet doesn't same with any set.
{
    FCAConcept.addElement(c);
    System.out.println("Now, Concept No." + (FCAConcept.size() - 1) + " is added.");
    reCheck(a, (Concept) FCAConcept.lastElement());
}
} // if
} //for 'a'

try {
    statement.close();
    con.close();
} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}

} //recheck function

private void sorting(Vector v){

    FCAConcept = v;

    for(int i = 0; i < FCAConcept.size() ; i++)
    {
        Vector tmp1 = null;
        int [] tmp2 = null;
        Hashtable H = new Hashtable();
        int tmp1size = 0;

        tmp1 = new Vector(1, 1);
        for(int j = 0; j < ((Concept) FCAConcept.elementAt(i)).childSet.size(); j++)
        {
```java
tmp1.addElement(String.valueOf(((Concept)FAconcept.elementAt(i)).childSet.elementAt(j))).objectSet.size()) + String.valueOf(i + 1000));

String tmpsort = String.valueOf(((Concept)FAconcept.elementAt(i)).childSet.elementAt(j).objectSet.size()) + String.valueOf(i + 1000);

H.put(tmpsort, String.valueOf(((Concept)FAconcept.elementAt(i)).childSet.elementAt(j)));
tmp1.size = tmp1.size();
}

int tmp2 = new int[tmp1.size];
for(int k = 0; k < tmp1.size; k++){
    tmp2[k] = Integer.parseInt(String.valueOf(tmp1.elementAt(k)));
    System.out.println("tmp2[" + tmp2[k] + "]");
}
Sort(tmp2);
Vector tmp3 = new Vector(1, 1);
for(int x = 0; x < tmp2.length; x++){
    String s = (String)H.get(String.valueOf(tmp2[x]));
    tmp3.addElement(s);
    System.out.println("tmp3[" + tmp3.elementAt(x) + "]");
}

((Concept)FAconcept.elementAt(i)).childSet = tmp3;

} //for i

private int[] Sort(int a[]){
    int i = a.length;
    while(--i >= 0){
        for(int j = 0; j < i; j++){
            if(a[j] < a[j+1]){  
                int temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
            }
        }
    } //for
    return a;
}

public void findAllGrandChildren(Concept c){
    for(int i=0; i < ((Concept)c).childSet.size(); i++){
        if (!(((String)((Concept)c).childSet.elementAt(i)).equals(String.valueOf(10001)))){
            grandChildren.addElement(((Concept)c).childSet.elementAt(i));
            int tmpindex = Integer.parseInt(String.valueOf(((Concept)c).childSet.elementAt(i)));
            findAllGrandChildren(((Concept)FAconcept.elementAt(tmpindex));
        }
    }
}
```
public void findAllParents(Concept c) {
    for(int i=0; i < ((Concept)c).parentSet.size(); i++)
    {
        if (!(((String)(((Concept)c).parentSet.elementAt(i))).equals(String.valueOf(10000))))
        {
            allParents.addElement(((Concept)c).parentSet.elementAt(i));
            int tmpindex = Integer.parseInt(String.valueOf(((Concept)c).parentSet.elementAt(i)));
            findAllParents((Concept)FCAconcept.elementAt(tmpindex));
        }
    }
}

public void findLabeledConcept()
{
    for(int i=0; i < NoOfItems - 1; i++)
    {
        String temp = Integer.toString(i + 1);
        int conceptNo = 0;
        int max = 0;

        for(int j=0; j < FCAconcept.size(); j++)
        {
            for(int m=0; m < ((Concept)FCAconcept.elementAt(j)).attributeSet.size(); m++)
            {
                if (temp.equals(((String)(((Concept)FCAconcept.elementAt(j)).attributeSet.elementAt(m)))))
                {
                    if (max < ((Concept)FCAconcept.elementAt(j)).objectSet.size())
                    {
                        max = ((Concept)FCAconcept.elementAt(j)).objectSet.size();
                        conceptNo = i;
                    }
                }
            }
        }
        LabelConcept LC = new LabelConcept(conceptNo, temp);
        label.addElement(LC);
    }
}
7 makeLattice.java

// makeLattice.java creates lattice table and stores the information from the Vector that contain all
// information such as ObjectSet, AttributeSet, ChildSet, and ParentSet.

import java.sql.*;

public class makeLattice {
    public static void main(String args[]) {
        makeConcept MC = new makeConcept();
        RawData a = new RawData();

        String url = "jdbc:odbc:kim";
        Connection con = null;
        String createString = null;

        createString = "create table LATTICE1 " +
                        "+(ConceptNO INTEGER," +
                        "ObjectSet VARCHAR(100)," +
                        "AttributeSet VARCHAR(100)," +
                        "ObjectSetSize INTEGER," +
                        "ChildSet VARCHAR(100)," +
                        "ParentSet VARCHAR(100)," +
                        "Support DECIMAL(2,2))";

        Statement stmt = null;

        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        } catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }

        try {
            con = DriverManager.getConnection(url, "myuser", "password");
            stmt = con.createStatement();
            stmt.executeUpdate(createString);
            stmt.close();
            con.close();
        } catch(SQLException ex) {
            System.err.println("SQLException: " + ex.getMessage());
        }

        // insert into table values

        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        } catch(java.lang.ClassNotFoundException e) {
            System.err.print("Exception while loading JDBC-ODBC bridge driver");
            System.err.println(e.getMessage());
        }

        try {
            con = DriverManager.getConnection(url, "myuser", "password");
        } catch (SQLException e) {
            System.err.println("SQLException while connecting to the database");
        }
    }
}
try {
    stmt = con.createStatement();
} catch (SQLException e) {
    System.err.println("SQLException while creating statement");
}
int result=0;
try {
    for(int i=0; i < MC.FCAconcept.size(); i++)
    {
        int concept = i; int size =0; float support;
        String O = ""; String A = ""; String C = ""; String P = "";
        for(int j=0; j < ((Concept) MC.FCAconcept.elementAt(i)).objectSet.size(); j++)
        {
            O = O + (String)((Concept) MC.FCAconcept.elementAt(i)).objectSet.elementAt(j) + " ";
        }
        for(int k=0; k < ((Concept) MC.FCAconcept.elementAt(i)).attributeSet.size(); k++)
        {
            A = A + (String)((Concept) MC.FCAconcept.elementAt(i)).attributeSet.elementAt(k) + " ";
        }
        size = ((Concept) MC.FCAconcept.elementAt(i)).objectSet.size();
        for(int x=0; x < ((Concept) MC.FCAconcept.elementAt(i)).childSet.size(); x++)
        {
            C = C + (String)((Concept) MC.FCAconcept.elementAt(i)).childSet.elementAt(x) + " ";
        }
        for(int z=0; z < ((Concept) MC.FCAconcept.elementAt(i)).parentSet.size(); z++)
        {
            P = P + (String)((Concept) MC.FCAconcept.elementAt(i)).parentSet.elementAt(z) + " ";
        }
        support = ((Concept) MC.FCAconcept.elementAt(i)).objectSet.size() / (float)a.NoOfRow;
        result=stmt.executeUpdate("INSERT INTO LATTICE1 VALUES(" +
            concept + "," +O+ "," +A+ "," +size+ "," +C+ "," +P+ "," +support+");
        } //for
        stmt.close();
        con.close();
    } catch(SQLException ex) {
        System.err.println("SQLException: " + ex.getMessage());
    }
}
APPENDIX B: Source code for Browsing Association Rules

1. display.jsp

// display.jsp displays all items that is belonged a category that is chosen by user...

<%@ page import="java.sql.*,java.util.*" %>

<html>
<head>
<title>Items</title>
</head>

<body bgcolor="#ffffff">
<table border="0" width="100%" bgcolor="#000000">
<tr>
<td width="100%"><font color="#FFFFFF"> left</font></td>
</tr>
</table>

<

String parameter = request.getParameter("ch");

String url = "jdbc:odbc:kim";
java.sql.Connection con = null;
Statement statement = null;
String choice = parameter;

String firstQuery = "SELECT * FROM ItemTable WHERE Category = " + choice+ " ";
ResultSet results = null;

out.println("You chose Category " +choice+ ");

try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (Exception e) {
    System.err.println("Exception while loading JDBC-ODBC bridge driver");
}

try {
    con = DriverManager.getConnection(url, "myuser", "password");
    statement = con.createStatement();
} catch (SQLException e) {
    System.err.println("SQLException while connecting to the database");
}

try {
    results = statement.executeQuery(firstQuery);
} catch (SQLException e) {
    System.err.println("SQLException while executeQuery1");
}
System.err.println(e.getMessage());
}

try {
    while (results.next()) {
        String name1 = results.getString(2);
        String name2 = "http://localhost:8080/examples/jsp/Query.jsp?pre="&item="+name1; String descript = results.getString(4); descript.trim();
        String thumbPic = "http://localhost/* results.getString(5); String largePic = "http://localhost/* results.getString(6);<p><a href="<%= name2 %>><img border="0" src="<%= thumbPic %>>" /></a></p>
    }
} catch (SQLException e) {
    System.err.println("SQLException");
    System.err.println(e.getMessage());
}

try {
    statement.close();
    con.close();
} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}
2. Query.jsp

// Query.jsp shows all results in the sequential search that contain the chosen item information and
// suggested items order by confidence.

```html
<html>
<head>
    <title>Items</title>
</head>

<body bgcolor="#ffffff">
<table border="0" width="100%" bgcolor="#000000">
<tr>
    <td width="100%"><br>
        <font face="Century Gothic" size="7">
            <font color="#FF0000">one</font>
        </font>
        <font color="#FFFFFF">left</font>
    </td>
</tr>
</table>
<br>
<font face="geneva, arial, helvetica" size="4">

DecimalFormatDemo demo = new DecimalFormatDemo();
String parameter1 = request.getParameter("item");
String parameter2 = request.getParameter("pre");
String choice = "";

Vector previous = new Vector(1,1);

if (parameter2.equals("")){
    choice = parameter1;
    previous.addElement(parameter1);
}
else{
    StringTokenizer token = new StringTokenizer(parameter2, (String)" ");
    while (token.hasMoreTokens()) {
        String ss = token.nextToken();
        previous.addElement(ss);
    }
    previous.addElement(parameter1);
    Object [] obj= new Object[previous.size()];

    for(int i=0; i< previous.size(); i++){
        obj[i] = previous.elementAt(i);
    }

    Arrays.sort(obj);
    choice = ((String)obj[0]);

    if(obj.length > 1){
        for(int i=1; i< obj.length; i++){
            choice = choice + ", " + ((String)obj[i]);
        }
    }
```

99
String url = "jdbc:odbc:kim";
java.sql.Connection con = null;
Statement statement = null;

try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (Exception e) {
    System.err.println("Exception while loading JDBC ODBC bridge driver");
}

try {
    con = DriverManager.getConnection(url, "myuser", "password");
    statement = con.createStatement();
} catch (SQLException e) {
    System.err.println("SQLException while connecting to the database");
}

try {
    ResultSet results = statement.executeQuery(Query);
} catch (SQLException e) {
    System.err.println(e.getMessage());
    System.err.println("SQLException while executing query");
}

try {
    while (results.next()) {
        String name1 = results.getString(2);
        String desc1 = results.getString(4);
        String largePic = "http://localhost/" + results.getString(6);
    }
} catch (SQLException e) {
    System.err.println("SQLException");
    System.err.println(e.getMessage());
}

try {
    statement.close();
    con.close();
}
} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}

// check there is same AttributeSet.
try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (Exception e) {
    System.err.println("Exception while loading JDBC-ODBC bridge driver");
}

try {
    con = DriverManager.getConnection(url, "myuser", "password");
    statement = con.createStatement();
} catch (SQLException e) {
    System.err.println("SQLException while connecting to the database");
}

String Query2 = "SELECT ATTRIBUTESET FROM Lattice";

try {
    results = statement.executeQuery(Query2);
} catch (SQLException e) {
    System.err.println(e.getMessage());
    System.err.println("SQLException while executing query");
}

int flag = 0;
try {
    while (results.next()) {
        if (e[(choice).equals(results.getString(1))]) {
            flag++;
        }
    }  //while
} catch (SQLException e) {
    System.err.println("SQLException");
    System.err.println(e.getMessage());
}

if (flag == 0) { // if parameter doesn't exist, because of more than two parameters
    <br>
    <font face="geneva, arial, helvetica" size="4" color="#800000">
    Customers who bought "<font face="geneva, arial, helvetica" size="4" color="#800000"></font>"
    <font face="geneva, arial, helvetica" size="4" color="#800000"></font>"
    <font face="geneva, arial, helvetica" size="4" color="#800000"></font>"
    <font face="geneva, arial, helvetica" size="4" color="#800000"></font>"
    <font face="geneva, arial, helvetica" size="4" color="#800000"></font>"
    also bought:
    </font>
    <br>
    <font face="geneva, arial, helvetica" size="4" color="#800000">
    No same attribute !
    </font>
    </p>
}
try {
    statement.close();
    con.close();
} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}

if (!
else{

    <p>
    <br>
    <font face="geneva, arald, helvetica" size="4" color="#800000">Customers who bought</font>
    <font color="#100000" face="geneva, arald, helvetica" size="4">%</font>
    <font face="geneva, arald, helvetica" size="4" color="#800000">also bought</font>
    </font><br>
    </p>
    <p>
    //</p>

    // if there is same AttributeSet,find childSet and OBJECTSETSIZE

    double support = 0.01;
    String firstQuery = "SELECT ChildSet, OBJECTSETSIZE FROM Lattice
    WHERE AttributeSet = " + choice+""
    try {
        results = statement.executeQuery(firstQuery);
    } catch (SQLException e) {
        System.err.println("SQLException while executeQuery!");
        System.err.println(e.getMessage());
    }

    String child = "";
    int size = 0;
    Vector str = new Vector(1,1);

    try {
        while (results.next())
            {
            child = results.getString(1);
            size = results.getInt(2);
            }
    } catch (SQLException e) {
        System.err.println("SQLException");
        System.err.println(e.getMessage());
    }

    StringTokenizer st = new StringTokenizer(child, (String)" ");
    while (st.hasMoreTokens()) {
        String ss = st.nextToken();
        str.addElement(ss); //store childSet information
    }

    String str1 = (String)str.elementAt(0);
    if(str1.equals("10001") { :
try{
    statement.close();
    con.close();
} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}
}

else{
    Vector vv1 = new Vector(1,1); // it will store attributeSet of child
    Vector vv2 = new Vector(1,1); // it will store objectsize

    for(int i=0; i<str.size(); i++)
    {
        String secondQuery = "SELECT ATTRIBUTES,OBJECTSET.SIZE FROM LATTICE
        WHERE CONCEPTNO = " + Integer.parseInt((String)str.elementAt(i));
        String tmp1 = ""; int tmp2 = 0;

        try{
            results = statement.executeQuery(secondQuery);

            while (results.next())
            {
                tmp1 = results.getString(1);
                tmp2 = results.getInt(2);
                StringTokenizer stknz = new StringTokenizer(tmp1, [String]"\t");

                while (stknz.hasMoreTokens())
                {
                    String ss = stknz.nextToken();
                    vv1.addElement(ss);
                    vv2.addElement(String.valueOf(tmp2));
                }
            }
        } catch (SQLException e) {
            System.err.println("SQLException");
            System.err.println(e.getMessage());
        }
    } //for i
}
try{
    statement.close();
    con.close();
} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}
try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
}

103
} catch (Exception e) {
    System.err.println("Exception while loading JDBC-ODBC bridge driver");
}

try {
    con = DriverManager.getConnection("url", "myuser", "password");
    statement = con.createStatement();
} catch (SQLException e) {
    System.err.println("SQLException while connecting to the database");
}

for(int j=0; j<vv1.size(); j++) {
    String sss = (String)vv1.elementAt(j);
    int zflag = 0;

    for(int z=0; z<previous.size(); z++) {
        if(sss.equals((String)previous.elementAt(z))) {zflag++;}
    }

    if (zflag == 0) {
        String recommendName = sss;
        double tmp3 = (Double.parseDouble((String.valueOf(vv2.elementAt(z)))) / (double)size) / (double) 100;
        String S2 = demo.customFormat("###.##", tmp3);
        String confidence = S2 + "%";
        String thirdquery = "SELECT * FROM itemTable WHERE name = " +
                            "recommendName" +";"
        try {
            results = statement.executeQuery(thirdquery);
        } catch (SQLException e) {
            System.err.println(e.getMessage());
            System.err.println("SQLException while executing query");
        }

        try {
            while (results.next()) {
                String name1 = results.getString(2);
                String name2 = "http://localhost:8080/examples/jsp/Query.jsp?pre="+
                                "choice"+"&item="+name1;
                String descrit = results.getString(4);
                String thumbPic = "http://localhost/" + results.getString(5);
                String largePic = "http://localhost/" + results.getString(6);

                %> <a href="name2 "%> <img border="0" src="thumbPic ">
                </a> </p>
                <a href="name2 "%> <%= name1 ">%></a></br>
                <font color="#f40000" face="geneva, arial, helvetica" size="4"><%= confidence"></font>
                &nbsp;<%= descrit ">%>
                <%
            }
        } catch (SQLException e) {
            System.err.println("SQLException");
        }
    }
}
System.err.println(e.getMessage());

} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}

} // else

</body>
</html>
APPENDIX C: Source code for Retrieving Association Rules

1. sortConfidence.java

// sortConfidence.java sorts all rows in the confidence table.
import java.sql.*;
import java.util.*;

public class sortConfidence {
    public static void main(String args[]) {

        Vector v0 = new Vector(1,1);
        Vector v1 = new Vector(1,1);
        Vector v2 = new Vector(1,1);
        Vector v3 = new Vector(1,1);
        Vector v4 = new Vector(1,1);
        Vector v5 = new Vector(1,1);
        Vector v6 = new Vector(1,1);
        Vector v7 = new Vector(1,1);
        Vector v8 = new Vector(1,1);
        Vector v9 = new Vector(1,1);

        String url = "jdbc:odbc:kim";
        Connection con = null;
        String createString = null;
        createString = "create table SORTEDEXAMPLECONFIDENCE " +
                        "(ConceptNO INTEGER, " +
                        "ObjectSetNO1 INTEGER," +
                        "AttributeSet1 VARCHAR(100)," +
                        "Antecedent VARCHAR(100)," +
                        "ComparedConcept INTEGER," +
                        "ObjectSetNO2 INTEGER," +
                        "AttributeSet2 VARCHAR(100)," +
                        "consequent VARCHAR(100)," +
                        "Support FLOAT(5)," +
                        "Confidence FLOAT(5))";

        Statement stmt = null;

        // insert into table values

        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        } catch(java.lang.ClassNotFoundException e) {
            System.err.print("ClassNotFoundException: ");
            System.err.println(e.getMessage());
        }

        try {
            con = DriverManager.getConnection(url, "myuser", "password");
            stmt = con.createStatement();
        }
stmt.executeUpdate(createString);
stmt.close();
con.close();
)
} catch (SQLException ex) {
    System.err.println("SQLException: " + ex.getMessage());
}

try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
} catch (Exception e) {
    System.err.println("Exception while loading JDBC-ODBC bridge driver");
}

try {
    con = DriverManager.getConnection(url, "myuser", "password");
} catch (SQLException e) {
    System.err.println("SQLException while connecting to the database");
}

Statement statement = null;
try {
    statement = con.createStatement();
} catch (SQLException e) {
    System.err.println("SQLException while creating statement");
}

String query = "SELECT ConceptNO, ObjectSetNO1, AttributeSet1, ComparedConcept, ObjectSetNO2, AttributeSet2, MAX(CONFIDENCE) AS CON " +
                "FROM MANYCONFIDENCE " +
                "GROUP BY ConceptNO, ObjectSetNO1, AttributeSet1, ComparedConcept, ObjectSetNO2, AttributeSet2";

ResultSet results = null;
try {
    results = statement.executeQuery(query);
} catch (SQLException e) {
    System.err.println(e.getMessage());
    System.err.println("SQLException while executing query");
}

try {
    while(results.next()) {
        int tmp1 = results.getInt(1);
        int tmp2 = results.getInt(2);
        String tmp3 = results.getString(3);
        String tmp4 = results.getString(4);
        int tmp5 = results.getInt(5);
        int tmp6 = results.getInt(6);
        String tmp7 = results.getString(7);
        String tmp8 = results.getString(8);
        float tmp9 = results.getFloat(9);
        float tmp10 = results.getFloat(10);
        v0.addElement(String.valueOf(tmp1));
        v1.addElement(String.valueOf(tmp2));
        v2.addElement(tmp3);
    }

v3.addElement(tmp4);
v4.addElement(String.valueOf(tmp5));
v5.addElement(String.valueOf(tmp6));
v6.addElement(tmp7);
v7.addElement(tmp8);
v8.addElement(String.valueOf(tmp9));
v9.addElement(String.valueOf(tmp10));
}
statement.close();
con.close();
}
catch (SQLException e) {
    System.err.println("SQLException while closing database");
    System.err.println("SQLException: " + e.getMessage());
}
}
try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
}
catch (Exception e) {
    System.err.println("Exception while loading JDBC-ODBC bridge driver");
}
try {
    con = DriverManager.getConnection(url, "myuser", "password");
}
catch (SQLException e) {
    System.err.println("SQLException while connecting to the database");
}
}
try {
    statement = con.createStatement();
}
catch (SQLException e) {
    System.err.println("SQLException while creating statement");
}
int result=0;
try {
    for(int w=0; w< v0.size(); w++)
    {
        result=statement.executeUpdate("INSERT INTO SORTEDEXAMPLECONFIDENCE VALUES(" +
            v9.elementAtA[w] + ");
    }
}
catch (SQLException e) {
    System.err.println("SQLException while inserting value");
    System.err.println("SQLException: " + e.getMessage());
}
}
try {
    statement.close();
    con.close();
}
catch (SQLException e) {
    System.err.println("SQLException while closing database2");
    System.err.println("SQLException: " + e.getMessage());
}
2. Association.java

// Association.java displays the frame that contains the closed frequent itemsets.

import javax.swing.*;
import java.sql.*;
import java.awt.*;
import java.net.*;
import java.io.*;
import java.awt.event.*;
import java.util.*;

public class Association implements ActionListener
{
  JTextField TSUP, TCON;
  JLabel LSUP, LCON;
  JTextArea mainText = null;
  JButton sendButton = null;
  float support, confidence;

  public void actionPerformed(ActionEvent event)
  {
    support = Float.parseFloat(TSUP.getText());
    confidence = Float.parseFloat(TCON.getText());

    try {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    } catch (Exception e) {
      System.err.println("Exception while loading JDBC-ODBC bridge driver");
    }

    String url = "jdbc:odbc:kim";
    java.sql.Connection con = null;

    try {
      con = DriverManager.getConnection(url, "myuser", "password");
    } catch (SQLException e) {
      System.err.println("SQLException while connecting to the database");
    }

    Statement statement = null;
    try {
      statement = con.createStatement();
    } catch (SQLException e) {
      System.err.println("SQLException while creating statement");
    }

    float temp = support/100*9;

    String query = "SELECT * FROM SORTEDEXAMPLECONFIDENCE WHERE ObjectSetNO1 > " +
                    temp + " AND ObjectSetNO2 > " + temp +
                    " AND Confidence > " + confidence/(float)100 +
                    " ORDER BY Confidence DESC, Support DESC";

    ResultSet results = null;
try {
    results = statement.executeQuery(query);
} catch (SQLException e) {
    System.err.println(e.getMessage());
    System.err.println("SQLException while executing query");
}

try {
    while(results.next()) {
        String tmp1 = results.getString(4);
        String tmp2 = results.getString(8);
        float tmp3 = results.getFloat(9);
        tmp3 = tmp3 * 100;
        float tmp4 = results.getFloat(10);
        tmp4 = tmp4 * 100;
        mainText.setText(mainText.getText()+line+"\n");
    }
} catch (SQLException e) {
    System.err.println(e.getMessage());
    System.err.println("SQLException while examining ResultSet");
}

try {
    statement.close();
    con.close();
} catch (SQLException e) {
    System.err.println("SQLException while closing database");
}

private JPanel createComponents() {
    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout());

    JPanel topPanel = new JPanel();
    TSUP = new JTextField(2);
    topPanel.add(TSUP,BorderLayout.CENTER);
    LSUP = new JLabel("% minimum support");
    topPanel.add(LSUP,BorderLayout.CENTER);

    TCON = new JTextField(2);
    topPanel.add(TCON,BorderLayout.CENTER);
    LCON = new JLabel("% minimum confidence");
    topPanel.add(LCON,BorderLayout.CENTER);

    // add the send button
    sendButton = new JButton("Discover all Association rules");
    topPanel.add(sendButton,BorderLayout.CENTER);
    sendButton.addActionListener(this);
    panel.add(topPanel, BorderLayout.NORTH);

    // add the main chat text area
    mainText = new JTextArea(30,100);
    mainText.setFont(new Font("Serif", Font.BOLD, 15));
    mainText.setLineWrap(true);
mainText.setWrapStyleWord(true);
mainText.setEditable(false);
JScrollPane scrollPane = new JScrollPane(mainText);
panel.add(scrollPane, BorderLayout.CENTER);

return panel;

public static void main(String[] args) {
    try {
        UIManager.setLookAndFeel(
            UIManager.getSystemLookAndFeelClassName());
    } catch (Exception e) {
    }

    JFrame f = new JFrame("Mining Association Rules");
    Association A = new Association();
    Component contents = A.createComponent();
f.getContentPane().add(contents, BorderLayout.CENTER);

    // allow the window to be closed
    f.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent event) {
            System.exit(0);
        }
    });

    f.setSize(1300.800);
f.pack();
f.setVisible(true);
}
**Vita Autoris**

**NAME:** Kim, Jong-Seok  
**PLACE OF BIRTH:** Seoul, South Korea  
**YEAR OF BIRTH:** 1968  
**EDUCATION:**  
Hankuk University of Foreign Studies, Seoul, Korea  
1988-1994 B.A.  
University Of Windsor, Windsor, Ontario, Canada  
1999-2001  
A candidate for the Master’s degree in Computer Science