DEVELOPMENT OF HASH TABLE BASED WEB-READY DATA MINING ENGINE

SK MD OBAIDULLAH

Department of Computer Science & Engineering, Aliah University,
Saltlake, Sector-V, Kol-900091, West Bengal, India
sk.obaidullah@gmail.com
http://www.aliah.ac.in

Abstract

This paper focus on analysis and implementation of a hash table based data mining engine used to discover knowledge from transaction database for electronic commerce industry. Apriori algorithm is analyzed and implemented to discover association relationship among amounts of business transaction records. Such relationships can be derived based on individual product purchased in all transactions. They also can be applied to single category level that will provide more granular analysis. This engine can derive both single and multidimensional association rules. The resulted association relationships help in much business decision-making processes, such as catalog design, shelf layout, cross-marketing, bundled advertisement, etc.

***Keywords*:** Data mining, Hash table, Apriori, Association rules.

1. Introduction and Background

1.1. *Statement of Problem Area*

Data mining has become a booming topic of research in Computer Science. This fast growing phenomenal is driven by many reasons. First, data mining and data warehousing are extremely fertile with research problems and yet present an extreme helpful tool to manage vast amount of data. Secondly, information grows at an exponential rate and Internet technology has made it very easy to gather that information from all over the world so that companies and organizations nowadays find themselves inundated with data, and eager to extract useful information from them to benefit their business. Therefore, from computer science point of view, the two dominant problems in e-commerce become (1) How to extract information efficiently – this is the topic of data mining algorithm; (2) How to be serviced and benefited by results of data mining – this is the topic of e-commerce. This paper is aimed to present a software design and implementation to solve the first part of these problems for e-commerce businesses.

1.2. *Previous and Current Work, Methods and Procedures*

There have been an extraordinary number of papers published to present ideas and solution for data mining algorithms [1][2][3][5]. The application fields are e-commerce, genetic engineering, building architecture, etc [6][7][9]. Tools used by e-commerce business are also growing very fast. Many companies have developed various data-mining packages for user, such as IBM DB2 Intelligent Miner for Data, Nuggets True Data Mining, etc. However, these tools did not specially focus on mining association rules from huge amount of electronic transactions and provide capability to be utilized by a web based interface.

1.3. *Brief Work Description*

This paper analyzes and implements enhanced Apriori algorithm [4] in data mining for electronic commerce industry to discover association relationship among amounts of business transaction records. Both single dimensional and multiple dimensional [10] association rules are derived from this algorithm. This algorithm can also be applied based on product and category level for more granular analysis. This work also can be used to build a set of data mining interfaces for web based tools and provides formatted results to those tools.

1.4. *Work Objectives*

The objective of this work: Data Mining in E-Commerce is to provide a reliable, fast and web-ready data mining engine for electronic commerce industry. The discovered association relationships by this data mining engine will help in many business decision-making processes, such as catalog design, shelf layout, cross-marketing, bundled advertisement.

2. System Functional Specification

2.1. *Functions Performed*

Functionality supported and performed in this work is:

- Create table for transaction and store it in a file.
- Compute frequent itemsets based on specified minimum support. The computation is an iterative process: scan the records in database, count the occurrence of a certain combination of product, remove those whose count is less than minimum support, and continue the algorithm until no new frequent item can be found.
- Compute association rules based on frequent itemsets and specified minimum confidence. Since frequent itemsets can be for product level or category level. The association rules derived for it are also for product level or category level. The computation process is also an iterative process: enumerate all possible combination among all products shown in frequent itemsets, then compute the confidence, if confidence is less than minimum confidence, remove the combination, continue until no combination can be derived from frequent itemsets.

2.2. *User Interface Design*

User interface is mandatory at both ends of data mining engine:

- Provide database that data mining will discover association relationship from.
- Specify minimum support and minimum confidence to discover association rules.

2.3. *User Output Preview*

The outputs are the association rules [1][4] in the form of

$$X \Rightarrow Y \ \{\text{support, confidence}\}$$

Where X and Y contain a set of products. X => Y is interpreted as customer who bought products in X also bought products in Y. Support is the percentage of all transactions that include all products in X. Confidence is the probability that transaction which bought all products in X also contains all products in Y. For specified minimum support and minimum confidence, a list of such association rules are the outputs to the user or web based tools.

2.4. *System Data Base and File Structure*

This work uses a file named iii.txt which stores the user defined data (the transaction table). This file is open in append mode to take the input from user. The output results are also stored in this file as well in sorted order.

2.5. *External and Internal Limitation and Restrictions*

This work assumes that database where data mining engine retrieves data composes of records in three predefined tables and their formats are also as defined above. It is also required that user has knowledge of data mining concept so that the specified minimum support and minimum confidence are well defined and within the range. For example, minimum support and confidence may be defined between 0 and 1 or between 0% and 100%.

3. System Performance Requirements

3.1. *Efficiency*

The efficiency requirement to data mining algorithm developed in this work is dependent on two factors: algorithm and database retrievals. The requirements to algorithm can be listed as following:

- Scanning records in one stage to compute a set of frequent itemsets. The scanning should be done once at one stage.
- Computing super set from a set of frequent items so that all supersets are found efficiently.
- Sorting frequent items in frequent itemsets to achieve high efficiency for above two requirements.

The database retrievals only go to database once to retrieve all information about transaction. Thus the cached data can be used subsequently to achieve high efficiency.

3.2. *Reliability*

The reliability of data mining algorithm is partially derived from that of Apriori algorithm [4]. The enhancements made to Apriori algorithm are mainly using hashing to count occurrence when scanning transactions and computing association rules based on frequent itemsets. The reliability of both is well tested and proven to work as size of the problem grows.

3.3. *Scalability*

Data mining algorithm developed in this work does not put any restrictions on the size of the problem. It leads to good scalability when number of transactions, number of products and number of categories grows. However, as size of problem increases, the speed of the algorithm will be slowed.

4. System Design

4.1. *System Data Flow Diagrams*

The data mining algorithm developed in this work constructs a system which retrieves data from user specified minimum support and confidence, process on electronic transactions stored in the database, then compute frequent itemsets and ultimately association rules. The system data flow diagram can be summarized in Figure 1:

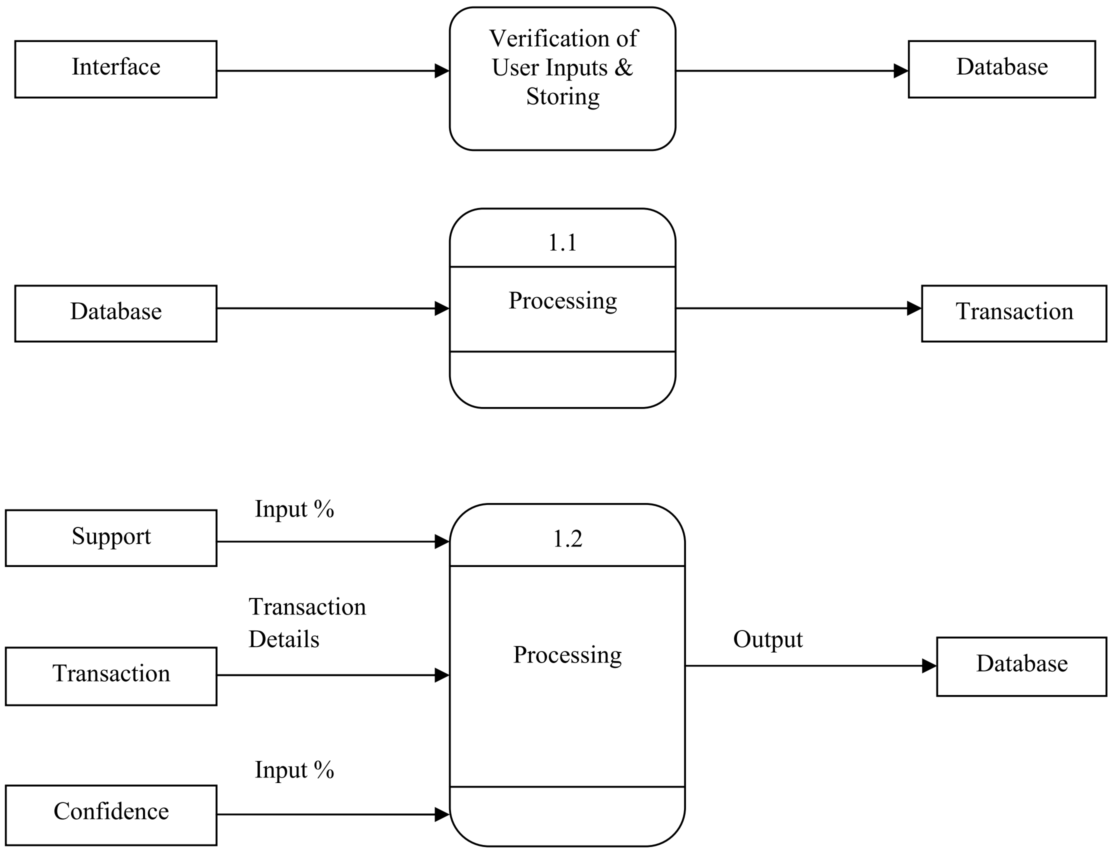

Fig. 1: System data flow diagram of the proposed data mining engine

4.2. *Flow Chart*

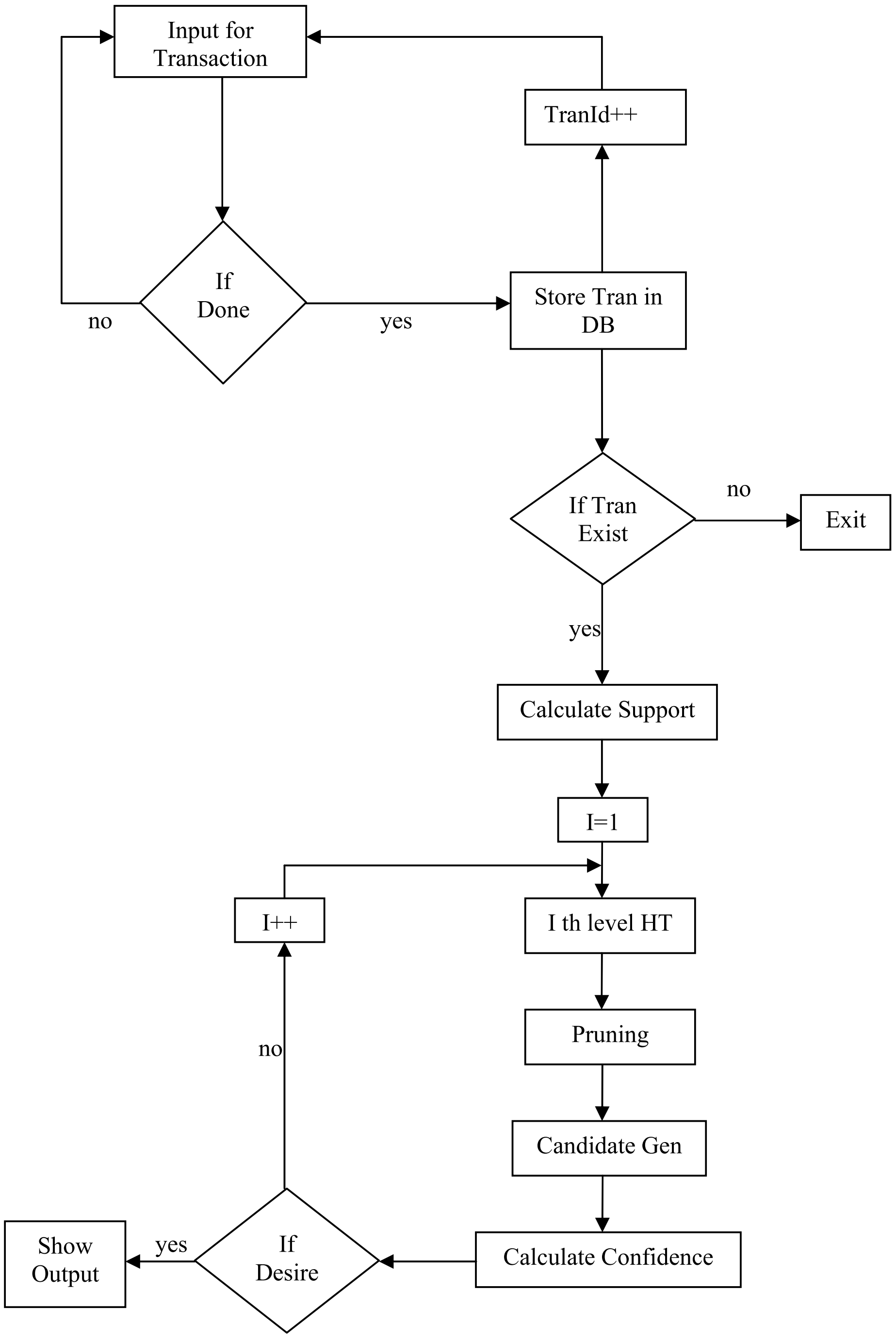

Fig. 2: Flow chart for the proposed data mining engine

4.3. *Required Support Software*

Required support software in this work is:

- Sun Microsystem's Java Development Kit (JDK) 2.
- Windows 98/2000/NT/higher

- Oracle 8i

5. System Data Structure Specification

5.1. *Other User Input Specification*

5.1.1. Identification of Input Data

Input data to data mining engine is divided into two parts: from user and from database or data warehouse. From user, minimum support, minimum confidence and interested category are provided to data mining engine. The minimum support and minimum confidence shall be a percentage value in the range of 0% to 100%. Interested category shall be either 0 for all categories or exist in database or data warehouse where data mining engine retrieves data.

Database or data warehouse is another input data to data mining engine. The database shall store electronic transaction information in three tables, tran, product and category. The commercial e-commerce database may not comply the definition as the tables are defined, but a customized database or data ware house can be built on top of e-commerce database to be input data to this engine. Thus the engine developed in this work shall only focus on such extracted data warehouse for data mining input.

5.1.2. Source of Input Data

The input data from user is originated from application program which provides graphic user interface to receive user's input and pass them to data mining engine. The source for database or data warehouse is a file.

5.1.3. Input Medium and/or Device

There is no input medium for data mining engine. For web based tools which feed user's input to data mining, the input medium is mouse and keyboard.

5.1.4. Legal Value Specification

Input data to data mining engine shall be valid as following specification:

• Minimum Support:	range from 0 to 1
• Minimum Confidence:	range from 0 to 1
• Interested Category:	0 or exist in database or data warehouse for mining

5.2. *Other User Output Specification*

5.2.1. Output Data Medium and/or Device

For standalone mode of data mining engine, the output data device is computer screen where all association rules are displayed. If the data mining engine is used by web based tools, there is no output data medium or device for data mining engine itself, but web based tools will use web browser on computer screen to display association rules to the user.

5.2.2. Output Format/Syntax

Output format for data mining engine is a list of association rules. Each association rule follows the format of X => Y {s, c} where X represents preconditions, Y is post conditions, s is the support for such association, c is the confidence. The example of one association rule could be Product A => Product B {10%, 40%}.

5.2.3. Output Interpretation

The association rule X => Y {s, c} is interpreted as following:

- Among all transactions in the database, the percentage (support) of those that purchased X is s.
- Among all those transactions that bought X, the percentage (confidence) of those that also purchased Y is c.

5.2.4. Example

The example of an association rule X => Y {s, c} is Product A => Product B {10%, 50%}. This rule means that among all transactions in the database, 10% transactions brought Product A. Among these 10% transactions, 50% also bought product B.

6. Algorithm Specification

The data mining engine uses Apriori algorithm [4] with enhancement to discover frequent itemsets and then compute association rules. In this section, at first Apriori algorithm is briefly outlined, then enhancement is presented in details, finally the algorithm to efficiently compute association rules are provided.
Apriori algorithm is to discover frequent itemsets for mining Boolean association rules.

Apriori Algorithm: Find frequent itemsets using an iterative level-wise based on candidate generation.
Input: Database, D, of transactions; minimum support threshold, min_sup.
Output: L, frequent itemsets in D
Method:

```
   I.   $L_1$ = find_frequent_1-itemsets(D);
  II.   for(k = 2, $L_{k-1} \neq Ø$; k++) {
 III.       $C_k$ = apriori_gen($L_{k-1}$, min_sup);
  IV.       for each transaction t D {          //scan D for counts
   V.           $C_t$ = subset($C_k$, t); //get the subsets of t that are candidates
  VI.           for each candidate c € $C_t$
 VII.               c.count++
VIII.       }
  IX.       $L_k$ = {c € $C_k$ | c.count $\geq$ min_sup}
   X.   }
  XI.   return L = $U_k$ $L_k$; //union of all $L_k$
```

procedure apriori_gen(L_{k-1}: frequent (k-1)-itemsets; min_sup:minimum support threshold)

```
   I.   for each itemset $l_1$ to $l_{k-1}$
  II.       for each itemset $l_2$ to $l_{k-1}$
 III.           if ($l_1$[1] = $l_2$[1]) ^ ($l_1$[2] = $l_2$[2]) ^ … ^ ($l_1$[k-2] = $l_2$[k-2]) ^ ($l_1$[k-1] = $l_2$[k-1]) then {
  IV.               c = $l_1$ x $l_2$; //join step: generate candidates
   V.               if has_infrequent_subset(c, $L_{k-1}$) then
  VI.                   delete c; //prune step: remove unfruitful candidate
 VII.               else add c to $C_k$;
VIII.           }
  IX.   return $C_k$;
```

procedure has_infrequent_subset(c: candidate k-itemsets; L_{k-1}: frequent (k-1)-itemsets)

```
   I.   for each (k-1)-subset s of c
  II.   if s €$L_{k-1}$ then
 III.       return TRUE;
  IV.   return FALSE
```

There are two enhancements to Apriori algorithm in the process of discovering frequent itemsets. First at step I in the Apriori algorithm, hashing technique is used to do only one full scan of database to retrieve the occurrences of one product in the transaction database. An object is created for each product and made key to hash table. In the end the hash table looks like:

Table 1: Example of Hash Table

Key	Occurrences
Product A	2
Product B	1
Product C	0
Product D	4

Assume there are m transactions, each transaction has average n products, total products are k, then this enhancement leads to m*n times updating hash table. Every update is completed without any loop. If for every product, a full scan is done to count its occurrence, there will be k full scan of database, for each scan, possible update is m*n. The hashing technique speeds up step I by k times. Second enhancement to Apriori algorithm is at step II of procedure apriori_gen. Assume there are n frequent items in Lk-1, the number of loops in step I and II are n*n. The enhancement is achieved by sorting frequent items in frequent itemsets. Step II is cut to start from one item after l1 specified in step I. This enhancement decreases number of loops from n*n to n*n/2;

Based on frequent itemsets, the association rules can be generated as follows:

I. for each frequent itemset l, generate all nonempty subsets of l.

II. for every nonempty subset s of l, output the rule " s => (l-s) " if support_count(l)/support_count(s) = min_conf , where min_conf is minimum confidence threshold. Generating nonempty subsets for frequent item in step I in this procedure is a complicated process because number of nonempty subset is
$i\ C_i$, i = 1 to n-1
where n is number of product in frequent item, Ci is the combination of picking i products from a set of n products. The work outlines the algorithm to compute all nonempty subsets for frequent item as follows:

```
I.     Construct n 1-item subsets for frequent item F, each subset contains only one product, call S1 the set of
       all these subsets.
II.    for(k = 2; k < n; k++)
III.       for each subset $s_1$ € $S_{k-1}$
IV.            for each item $s_2$ € $S_{k-1}$
V.                 if ($s_1[1] = s_2[1]$) ^ ($s_1[2] = s_2[2]$) ^…^ ($s_1[k-2] = s_2[k-2]$) ^ ($s_1[k-1] < s_2[k-1]$)
VI.    then{
VII.               s = $s_1$ x $s_2$; //join step: generate candidates
VIII.              add s to $S_k$;identify (l-s) compute association rule
IX.          }
X.     return S = $U_k$ $S_k$ as set of all nonempty sets for F.
```

The (l-s) can also be computed in step VII when s is identified, then association rule is also calculated at the same time. As all nonempty subsets of l are computed, all association rules are also discovered.

7. System Verification

7.1. *Items/Functions to be Tested*

This work provides a simple console for user to utilize data mining engine. Following items and functions need to be verified:

- The interested category will extract out all transactions that bought products belonging to the category. There should be no product in the transaction if this product does belong to this category.
- The minimum support constrains is not violated by data mining engine. All association rules computed at the end shall have at least specified support.
- The minimum confidence constrains is not violated by data mining engine. All association rules at the end shall have at least specified confidence.
- The completeness of the data mining algorithm shall compute all association rules that satisfy minimum support and minimum confidence based on given transaction database.

7.2. *Description of Test Cases*

There are following test cases to test items listed in previous section

- *Test Case 1*: Provide different category, transactions prepared by DBTransactions shall consist of only those that contain products in the same category. If interested category is out of range or does not exist in database, empty transactions shall be the end of result.
- *Test Case 2*: First set minimum support to zero to retrieve frequent itemsets. The itemsets shall contain possible combination of product Id as frequent items. Set minimum support to other value, the final frequent itemsets shall not contain any frequent item whose actual support is less than minimum support. No support can be greater than one.
- *Test Case 3*: First set both minimum support and minimum confidence to zero to retrieve all possible association rules. The association rules shall have actual confidence from zero to one.
- *Test Case 4*: Given minimum support and minimum confidence, compute all association rules. To verify the completeness of data mining algorithm, retrieve all transactions from database and manually go through Apriori algorithm, calculate frequent itemsets and compute association rules. The two results shall match each other. This test case is practical only for small transaction databases.
- *Test Case 5*: Increse number of transactions, number of products and number of categories in the database, run data mining engine to test performance and speed.

7.3. *Justification of Test Cases*

Test cases shall be able to cover all possible problems in the programming logic of data mining engine. The completeness of testing is to verify in small scale that data mining engine achieve this requirement.

7.4. *Test Run Procedures and Results*

To run the application, first make sure the remote Oracle database is running. Then go to working directory where Java class files and JDBC engine are located, run java dminer 0 0 0 to test the program. First parameter is minimum support, second is minimum confidence and last one is interested category. Alter any three parameters to start all the test cases.

The test cases listed in 7.2 have all been run on command line. Transaction for test case 1 has been verified as correct. All frequent items in frequent itemsets have been checked to have at least minimum support in test case 2. All association rules have also been checked to have at least minimum confidence in test case 3. In test case 4, the completeness test has been verified correct by running data mining engine, then going through transaction database to calculate actual association rules. Test case 5 has been run to show how data mining engine performs as size of database increases.

7.5. *Discussion of Test Results*

For a small transaction database, test results listed in 7.4 are found to be sufficient to prove data mining engine extracts association rules correctly. As database gets larger, such test cases can still apply but becomes time consuming.

8. Conclusion

A data mining engine is designed and implemented to discover association rules for electronic commerce transactions. Apriori algorithm with enhancement is used to extract the associate relationship among all products. Java, JDBC, Oracle database technologies [8] are employed to implement this data mining engine. The single dimensional and multi dimensional association rules discovered by this data mining engine helps e-commerce to design product catalog, customize advertisement etc.

It use in internet to take frequently used data and put into local server. It is also used in distributed database.

References

[1] Agrawal R., Imielinski, T., AND Swami, A: Mining Association Rules between Sets of Items in Large Databases. Proc. of the 1993 ACM SIGMOD Conf. on Management of Data.

[2] Agrawal R and Srikant R: Fast algorithms for mining association rules, Proceedings of the 20th Very Large DataBases Conference (VLDB'94), Santiago de Chile, Chile (1994), pp. 487–499.

[3] Aggarwal C, Procopiuc C and Yu P: Finding localized associations in market basket data, IEEE Trans. Knowl. Data Eng. 14 (2002) (1), pp. 51–62.

[4] Han J and Kamber M: Data Mining: Concepts and Techniques, Morgan Kaufimann Publishers, 2001

[5] Hipp J, Guntzer U and Nakhaeizadeh G: Algorithms for association rules mining—a general survey and comparison, SIGKDD Explore. 2 (2000) (1), pp. 58–64.

[6] Palshikar G K, Kale M S and Apte M M: Association rules mining using heavy itemsets, IEEE Trans. Knowl. Data Eng.

[7] Park J. S, Chen M. S, and Yu P. S: An Effective Hash-based Algorithm for Mining Association Rules." Proceedings of the ACM-SIGMOD Conference on Management of Data, 1995.

[8] Reese G: Database Programming with JDBC and Java, 1997

[9] Savasere A, Omiecinski E and Navathe S: An efficient algorithm for mining association rules in large databases, Proceedings of the VLDB Conference (1995), pp. 432–444.

[10] Xu W and Wang R: A Novel Algorithm of Mining Multidimentional Association Rules, Lecture Notes in Control and Information Sciences, 2006, Volume 344/2006, 771-777