An Efficient Data Mining Technique for Discovering Interesting Association Rules*

Show-Jane Yen and Arbee L.P. Chen

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.
Email: alpchen@cs.nthu.edu.tw

Abstract

Mining association rules is an important task. Past transaction data can be analyzed to discover customer purchasing behaviors such that the quality of business decision can be improved. The association rules describe the associations among items in the large database of customer transactions. However, the size of the database can be very large. It is very time consuming to find all the association rules from a large database, and users may be only interested in the associations among some items. Moreover, the criteria of the discovered rules for the user requirements may not be the same. Many uninteresting association rules for the user requirements can be generated when traditional mining methods are applied. Hence, a data mining language needs to be provided such that users can query only interesting knowledge to them from a large database of customer transactions. In this paper, a data mining language is presented. From the data mining language, users can specify the interested items and the criteria of the rules to be discovered. Also, an efficient data mining technique is proposed to extract the association rules according to the users' requests.

1 Introduction

Data mining has high applicability in retail industry. The effective management of business is significantly dependent on the quality of its decision making. It is therefore important to analyze past transaction data to discover customer purchasing behaviors and improve the quality of business decision. Because the amount of these transaction data is very large, an efficient algorithm needs to be devised for discovering useful information embedded in the transaction data.

An association rule describes the association among items in which when some items are purchased in a transaction, others are purchased too. The following definitions are adopted from [1]. A transaction *supports* an itemset Z, if Z is contained in the transaction. The *support for an itemset* is defined as the ratio of the total number of transactions which support this itemset to the total number of transactions in the database. To make the discussion easier, occasionally, we also let the total number of transactions which support the itemset denote the support for the itemset. The major work of mining association rules is to find all itemsets that satisfy a certain user-specified *minimum support*. Each such itemset is referred to as *large itemset*.

In order to find association rules, all large itemsets need to be generated from the database. However, the size of the database can be very large. It is very time consuming to find all association rules from the large database, and users may be only interested in the associations among certain items. Moreover, the criteria (such as minimum support) to discover rules for the users may not be the same. Many uninteresting association rules to the users can be generated when traditional methods of mining association rules are applied [1, 2, 4, 5]. Hence, a data mining language is needed such that users can query knowledge from a large database of customer transactions.

Meo, Psaila and Ceri [3] proposed a SQL-like operator for extracting association rules. The SQL-like operator is capable of expressing the problem of mining association rules. However, the expressive power of the SQL-like operator is still limitary. For example, users may want to query the associations between certain items and all the other items. The SQL-like operator cannot express this kind of query. Furthermore, the SQL-like query language is inconvenient for naive users, which is suitable to SQL programmers and experts, and the SQL-like operator performs set-oriented operations (i.e., join operations), which are very inefficient operations.

For designing a data mining language, two important issues need to be considered: the easy-to-use user interface and the efficient data mining language processing. This paper is concerned with the two issues. We present a data mining language, from which users only need to specify the criteria for discovering the rules, and the items in the antecedent and the consequent in the rules. We also propose an efficient data mining technique to process user's request. According to the user's request, the discovered large item-

*This work was partially supported by the Republic of China National Science Council under Contract No. NSC 86-2213-E-007-009.

0-8186-8147-0/97 $10.00 © 1997 IEEE

sets are called the *interesting large itemsets*, and the discovered association rules the *interesting association rules*.

For the efficient data mining technique, the interesting large itemsets are discovered firstly. After discovering all interesting large itemsets, the interesting association rules can be extracted according to the antecedent and the consequent specified in the user's request. For an interesting large itemset Z, if itemsets X and Y match the user-specified antecedent and the consequent, respectively, and $X \cup Y = Z$, then the rule $X \Longrightarrow Y$ can be generated. The *confidence* of $X \Longrightarrow Y$ in database D is the probability that when itemset X occurs in a transaction in D, itemset Y also occurs in the same transaction. That is, the ratio of the support for itemset Z to the support for itemset X. This rule is an interesting association rule if its confidence achieves the *minimum confidence* specified in the user's request. An example of such an association rule is "95% of the transactions in which coffee and sugar are purchased, milk is purchased too." The form of this rule is "coffee, sugar $\Longrightarrow$ milk 95%." The antecedent of this rule consists of coffee and sugar and the consequent consists of milk alone. The percentage 95% is the confidence of the rule.

The rest of the paper is organized as follows: Section 2 presents the data mining language. Section 3 proposes the efficient data mining algorithm for the presented data mining language. The performance analysis for the data mining algorithm is presented in Section 4. Finally, we conclude this paper and present directions for future research in Section 5.

2 Data Mining Language

In this section, we present a data mining language. Users can query association rules by specifying the related parameters in the data mining language. The data mining language is defined as follows:

Mining Association Rules
From < *Database* >
With
(**Antecedent** < *Items* > (*))
(**Consequent** < *Items* > (*))
Support s
Confidence c

< *Items* > ::= $item[, item]^k$

Where $0 \leq s, c \leq 1$ and $k \geq 0$. The parameter < *Database* > is used to specify the database name to which users query the association rules.

In the **With** clause, users can specify items in the antecedent and the consequent of the rules to be discovered after the keywords **Antecedent** and **Consequent**, respectively. If the items are specified in < *Items* > after the keyword **Antecedent** (**Consequent**), then the antecedent (consequent) of each discovered rule will contain these items. Besides, users need to specify the two criteria: minimum support and minimum confidence by the keywords **Support** and **Confidence**, respectively.

Notice that the phrase within the parentheses is optional. If the user does not specify the keyword **Antecedent** (**Consequent**), then any item can appear in the antecedent (consequent) of the discovered rules. If the user does not specify the two keywords **Antecedent** and **Consequent**, all association rules which achieve the user-specified criteria will be discovered. The notation "*" represents all items except the items specified in < *Items* >. If the notation "*" is specified after the keyword **Antecedent** (**Consequent**), then in addition to the items specified in < *Items* >, other items can also be contained in the antecedent (consequent) of each discovered rule.

3 Efficient Data Mining Algorithm

In this section, we describe how to process a user's request. We develop an efficient data mining (EDM) algorithm to generate the interesting association rules according to the user's request.

For a user's request, if both the two keywords **Antecedent** and **Consequent** are specified in the **With** clause and there is no notation "*" specified, then the antecedent and the consequent of the discovered rule will contain only the items specified in < *Items* >'s after the keywords **Antecedent** and **Consequent**, respectively. We call this type of users' requests the *Type I* request. If the user likes to extract association rules whose antecedent or consequent can contain other items except the items specified in < *Items* >, then the noataion "*" has to be specified in the **With** clause. We call this type of users' requests the *Type II* request. The request in which only one of the two keywords **Antecedent** and **Consequent** is specified also belongs to the Type II request. If both keywords **Antecedent** and **Consequent** are not specified in the **With** clause, all association rules achieve the user-specified criteria will be discovered. For this type of users' requests, we call it the *Type III* request.

There are four phases for the EDM algorithm to generate interesting association rules. The first phase is the *large item generation phase*. In this phase, EDM algorithm scans the database to record related information for each *interested item* and find large items. The interested items for the Type I request are the items specified in the **With** clause. The interested items for the Type II and Type III requests are all items in the database.

The second phase is the *association graph construction phase* which constructs an association graph to indicate the associations between every two large items generated in the first phase. The third phase is the *interesting large itemset generation phase* which generates all interesting large itemsets by traversing the constructed association graph according to the user's request. The final phase is the *interesting association rule generation phase* which generates all interesting association rules according to the discovered interesting large itemsets, the items specified after the two keywords **Antecedent** and **Consequent**, and the user-specified minimum confidence in the user's request.

3.1 Large item generation

In the first phase, algorithm EDM scans the database and builds a bit vector for each interested

item. The length of each bit vector is the number of transactions in the database. If an item appears in the ith transaction, the ith bit of the bit vector associated with this item is set to 1. Otherwise, the ith bit of the bit vector is set to 0. The bit vector associated with item x is denoted as BV_x. The number of 1's in BV_x is equal to the number of transactions which support the item x, that is, the support for the item x.

Property 1: The support for the itemset $\{i_1, i_2, ..., i_k\}$ is $BV_{i_1} \bullet BV_{i_2} \bullet ... \bullet BV_{i_k}$, where "$\bullet$" is the inner product of two vectors.

Lemma 1: If an item i_j $(1 \leq j \leq k)$ is not a large item, then the itemset $\{i_1,...,i_j,...,i_k\}$ cannot be a large itemset.

Rationale: Because item i_j is not a large item, the number of 1's in the bit vector BV_{i_j} is less than the minimum support. Hence, $BV_{i_1} \bullet ... \bullet BV_{i_j} \bullet ... \bullet BV_{i_k}$ must be less than the minimum support. The support for the itemset $\{i_1,...,i_j,...,i_k\}$ is also less than the minimum support according to the Property 1. So, the itemset $\{i_1,...,i_j,...,i_k\}$ is not a large itemset.

For the Type I request, if there is an interested item which is not a large item, then there is no answer to the request, because any itemset which contains the interested item cannot be a large itemset according to Lemma 1. Otherwise (i.e., all the interested items are large items), the inner products are performed on the bit vectors associated with all the interested items. If the result is no less than the user-specified minimum support, then the set of the interested items is an interesting large itemset. Hence, for the Type I request, the interesting large itemset can be generated after the first phase.

TID	Itemset
1	C E A G B
2	B D A E C G
3	A B C E G
4	E C G A
5	G E A C D
6	E G A H
7	A E G
8	A G
9	D F B C
10	B D F C H
11	C B F
12	B C
13	D F B
14	F B
15	C E

Table 1: Database TDB

Consider the example transaction database TDB shown in Table 1. Each record is a <TID, Itemset> pair, where TID is the identifier of the corresponding transaction, and Itemset records the items purchased in the transaction.

For example, if a user wants to know if the rule whose antecedent contains only items A and C, and consequent contains only item E is an association rule whose support and confidence achieve 20% and 80%, respectively, then the request is described as follows:

Request 1:
Mining Association Rules
From TDB
With
Antecedent A, C
Consequent E
Support 20%
Confidence 80%

This request belongs to the Type I request and the interested items are A, C and E. Because the minimum support is 20% (i.e., 3 transactions), the interested items A, C and E are all large items, and the associated bit vectors BV_A, BV_C and BV_E are (111111110000000), (111110001111001) and (111111100000000), respectively. After performing inner products on BV_A, BV_C and BV_E, the support for the itemset {A, C, E} is 5 ($\geq$ 3). Hence, the itemset {A, C, E} is an interesting large itemset.

3.2 Association Graph Construction

After building the bit vector for each interested item, the database need not be scanned again for the algorithm EDM. For the Type II and Type III requests, EDM needs to construct an association graph for the interesting large itemset generation.

Before constructing the association graph, each large item is assigned a unique integer number. Suppose item i represents the item whose item number is i, and the bit vector associated with item i is denoted as BV_i. In the association graph construction phase, for every two large items i and j $(i < j)$, if $BV_i \bullet BV_j$ is no less than the user-specified minimum support, a directed edge from item i to item j is created. Also, itemset (i,j) is a large 2-itemset. Note that an ordered list notation is used to indicate the order of the items in an itemset for the following discussion. Since the two itemsets (i,j) and (j,i) are the same, we consider only one direction to avoid computing the supports for the same itemsets.

For example, if we like to extract the association rules whose antecedent and consequent contain item C and item E, respectively, and support and confidence achieve 20% and 80%, respectively, then the request can be written as **Request 2**.

Request 2:
Mining Association Rules
From TDB
With
Antecedent C *
Consequent E *
Support 20%
Confidence 80%

For **Request 2**, the interested items are all items in the database. After the first phase, the large items (i.e., items A, B, C, D, E, F and G) are found and the bit vectors associated with the large items are built. The item numbers for the items A, B, C, D, E, F and G are 1, 2, 3, 4, 5, 6 and 7, respectively. The association graph for **Request 2** is shown in Figure 1.

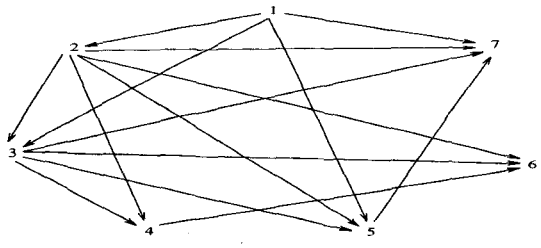

Figure 1: The association graph for **Request 2** and **Request 3**

3.3 Interesting large itemset generation

In the third phase, the algorithm EDM generates all interesting large itemsets according to the user's request. In this phase, we develop two algorithms: LGTE (Large itemset Generation by Tree Expansion) and LGDE (large itemset Generation by Direct Extension) to process the Type II and Type III requests, respectively.

3.3.1 interesting large itemset generation for Type II requests

For Type II requests, in addition to the items specified in < *Items* >'s, other items can also be contained in the antecedent and the consequent of each rule to be discovered. The Algorithm LGTE is applied to generate interesting large itemsets for Type II requests. LGTE constructs an *expansion tree* based on the association graph. Each node in the expansion tree contains an large itemset. During the expansion tree construction, LGTE expands each node which contains a large k-itemset ($k \geq 1$) to generate its child nodes which contain large $k+1$-itemsets.

First, LGTE checks if the set of the items specified in < *Items* >'s is a large itemset. If this itemset is a large itemset, then the itemset is designated the root node of the expansion tree, and the item numbers of the items in the root node is in the increasing order.

Subsequently, LGTE marks "*" between every two items, before the first item and after the last item in the root node. For example, consider **Request 2**, which is a Type II request. The itemset specified in **Request 2** is (3, 5), which is a large itemset since $BV_3 \bullet BV_5$ is 5 (≥ 3). After marking "*" in the itemset, the root node of the expansion tree becomes *3*5*.

After creating the root node of the expansion tree, the itemset in the root node will be extended to generate extended itemsets. We have the following Lemmas to check if an itemset in a node can be extended.

Lemma 2: If an itemset is not a large itemset, then any itemset which contains the items in this itemset cannot be a large itemset.

Rationale: Because the itemset is not a large itemset, the support for the itemset is less than the minimum support. Hence, the support for an itemset which contains the items in this itemset must be also less than the minimum support.

Lemma 3: For a large itemset $(i_1, i_2, ..., i_n)$, if there is no directed edge from any item i_h ($i_{k-1} < i_h$ when $k > 1$) to item i_k ($1 \leq k \leq n$), then itemset $(i_1, ..., i_{k-1}, i_h, i_k, ..., i_n)$ (or $(i_h, i_1, i_2, ..., i_n)$ when $k = 1$) cannot be a large itemset.

Rationale: Because there is no directed edge from any item i_h ($i_{k-1} < i_h$ when $k > 1$) to item i_k, the itemset (i_h, i_k) (or (i_h, i_1) when $k = 1$) is not a large 2-itemset. Hence, by Lemma 2, itemset $(i_1, ..., i_{k-1}, i_h, i_k, ..., i_n)$ (or $(i_h, i_1, i_2, ..., i_n)$ when $k = 1$) is not a large itemset.

Lemma 4: For a large itemset $(i_1, i_2, ..., i_n)$, if there is no directed edge from item i_k ($1 \leq k \leq n$) to any item i_p ($i_p < i_{k+1}$ when $k < n$), then itemset $(i_1, ..., i_k, i_p, i_{k+1}, ..., i_n)$ (or $(i_1, i_2, ..., i_n, i_p)$ when $k = n$) cannot be a large itemset.

Rationale: Because there is no directed edge from item i_k to any item i_p ($i_p < i_{k+1}$ when $k < n$), itemset (i_k, i_p) is not a large 2-itemset. Hence, itemset $(i_1, ..., i_k, i_p, i_{k+1}, ..., i_n)$ (or $(i_1, i_2, ..., i_n, i_p)$ when $k = n$) is not a large itemset according to Lemma 2.

If there is an n-itemset in a node with a "*" marked, then this n-itemset can be extended into $n+1$-itemset from the position of the mark "*" except one of the cases stated in Lemma 3 and Lemma 4 holds. Hence, for each "*" marked in a node, LGTE checks if the itemset I=$(i_1, i_2, ..., i_n)$ in the node can be extended. Suppose node N which contains itemset I is a root node, and is marked as $*i_1*i_2*...*i_n*$. In the following, we describe how to construct an expansion tree.

If the position of the mark "*" is before the first item i_1 of the itemset I in node N, then LGTE checks if there are directed edges from some items to the item i_1 in the association graph. If there is no directed edge from any item i_h to the item i_1, then the itemset cannot be extended for this mark "*", because for any item i_h, itemset $(i_h, i_1, i_2, ..., i_n)$ is not a large itemset according to Lemma 3. The mark "*" is then removed from node N. Hence, node N becomes $i_1*i_2*...*i_n*$. If there is a directed edge from an item i_e to the item i_1, then the itemset I is extended into the itemset $(i_e, i_1, i_2, ..., i_n)$, because this extended itemset can be a large itemset. If the extended itemset is a large itemset, the node $*i_e i_1*i_2*...*i_n*$ is created, and this node becomes a child node of node N.

If the position of the mark "*" is between items i_k and i_{k+1} of the itemset I in node N, then LGTE checks if there are directed edges from the item i_k to the other items whose item numbers are less than i_{k+1}. If there is no directed edge from the item i_k to the other item i_p ($i_p < i_{k+1}$), then the itemset cannot be extended for this mark "*", because for any item i_p, itemset $(i_1, ..., i_k, i_p, i_{k+1}, ..., i_n)$ is not a large itemset according to Lemma 4. The mark "*" between items i_k and i_{k+1} is then removed from node N and its child nodes created so far, and node N becomes $*i_1*...*i_k i_{k+1}*...*i_n*$. However, if there is a directed edge from item i_k to an item i_q ($i_q < i_{k+1}$) and item i_q to item i_{k+1}, then the itemset I is extended into the itemset $(i_1, ..., i_k, i_q, i_{k+1}, ..., i_n)$, because this extended itemset can be a large itemset. If the extended itemset is a large itemset, then the node $*i_1*...*i_k i_q*i_{k+1}*...*i_n*$ is created, and this node also becomes a child node of node N.

If the position of the mark "*" is after the last item i_n of the itemset I in node N, then LGTE checks if there are directed edges from item i_n to the other items in the association graph. If there is no directed edge from item i_n to the other item i_p, then the itemset cannot be extended for this mark "*", because for any item i_p, itemset $(i_1, ..., i_n, i_p)$ is not a large itemset according to Lemma 4. The mark "*" is then removed from node N and its child nodes created so far, and node N becomes $*i_1*i_2*...*i_n$. However, if there is a directed edge from item i_n to an item i_t, then the itemset I is extended into the itemset $(i_1, ..., i_n, i_t)$, because this extended itemset can be a large itemset. If the extended itemset is a large itemset, the node $*i_1*i_2*...*i_ni_t*$ is created, and this node becomes a child node of node N.

For each created node, if there exist "*"'s in the node, LGTE expands all children of this node for each mark "*", and removes the mark "*" from the node after the expansion. For a node, if all extended itemsets for a mark "*" are not large itemsets, then the mark "*" is removed from the node and its child node created so far. After constructing the expansion tree, the itemset in each node is an interesting large itemset. Finally, LGTE generates all interesting large itemsets from each node of the expansion tree. The algorithm LGTE is described as follows:

For example, consider **Request 2**. First, LGTE creates the root node *3*5*. For the first mark "*" in the node, because there exist directed edges from items 1 and 2 to item 3 in the association graph shown in Figure 1, the extended itemsets are (1, 3, 5) and (2, 3, 5). Because the itemsets (1, 3, 5) and (2, 3, 5) both are large itemsets, *13*5* and *23*5* are created as child nodes of the original node *3*5*. After expanding all child nodes for this mark "*", the original node *3*5* becomes 3*5*.

For the mark "*" between item 3 and item 5 in the root node, there is only one directed edge from item 3 to item 4 whose item number is less than item 5. However, there is no directed edge from item 4 to item 5. Hence, the itemset (3, 5) in the root node cannot be extended for this mark "*". The mark "*" between item 3 and item 5 is removed from the root node and its child nodes created so far. Hence, the root node becomes 35* and the two child nodes become *135* and *235*, respectively.

For the mark "*" after the item 5 in th root node, there is only one directed edge from item 5 to item 7 in Figure 1, and the extended itemset (3, 5, 7) is a large itemset. Hence, the node 357* is created as a child node of the root node. After the expansion, this mark "*" is removed from the root node and the root node becomes 35. Similarly, LGTE continues to expand all created child nodes. The expansion tree for **Request 2** is shown in Figure 2.

Finally, LGTE generates all interesting large itemsets from each node of the expansion tree in Figure 2. Because there are eight nodes in the expansion tree, there are eight interesting large itemsets generated.

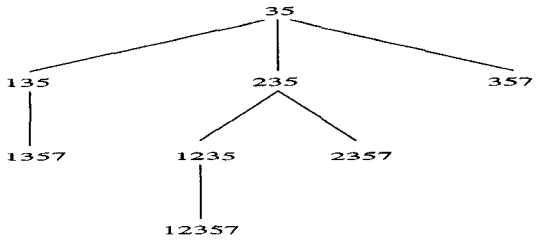

Figure 2: The expansion tree for **Request 2**

3.3.2 interesting large itemset generation for Type III requests

For Type III requests, there are no keywords **Antecedent** and **Consequent** specified in the users' requests. Hence, all association rules achieve the user-specified criteria will be discovered. For example, if we like to extract all association rules whose support and confidence achieve 20% and 80%, respectively, then the request is written as **Request 3**.

Request 3:
Mining Association Rules
From TDB
With
Support 20%
Confidence 80%

The LGDE algorithm is proposed to generate all interesting large itemsets for Type III requests. Suppose the set of large k-itemsets is L_k ($k \geq 1$). The 2-itemsets L_2 is found in the association graph construction phase. In the interesting large itemset generation phase, the LGDE algorithm generates large k-itemsets L_k ($k > 2$). For each large k-itemset in L_k ($k \geq 2$), the last item of the k-itemset is used to extend the itemset into $k+1$-itemsets. Suppose (i_1,i_2, ...,i_k) is a large k-itemset. If there is a directed edge from item i_k to item u in the association graph, then the itemset $(i_1, i_2, ..., i_k)$ is extended into $k+1$-itemset (i_1,i_2, ...,i_k,u), because this extended itemset can be a large itemset. The extended itemset (i_1,i_2, ...,i_k,u) is a large $k+1$-itemset if $BV_{i_1} \bullet BV_{i_2} \bullet ... \bullet BV_{i_k} \bullet BV_u$ is no less than the user-specified minimum support. If no large k-itemsets can be generated, the LGDE algorithm terminates.

For example, consider **Request 3**. In the association graph construction phase, 15 large 2-itemsets are generated and the association graph is shown in Figure 1. For large 2-itemset (1, 2), there are five directed edges from item 2 of the itemset (1, 2) to items 3, 4, 5, 6 and 7, respectively. Hence, the 2-itemset (1, 2) can be extended into 3-itemsets (1, 2, 3), (1, 2, 4), (1, 2, 5), (1, 2, 6) and (1, 2, 7). Because $BV_1 \bullet BV_2 \bullet BV_4$ and $BV_1 \bullet BV_2 \bullet BV_6$ are 1 and 0, which are less than the user-specified minimum support (i.e., 3), the 3-itemsets (1, 2, 4) and (1, 2, 6) are not large itemsets. The other 3-itemsets (1, 2, 3), (1, 2, 5) and (1, 2, 7) are large 3-itemsets. The LGDE algorithm terminates when no large 6-itemsets can be further generated.

3.4 Association rule generation

After discovering all interesting large itemsets, EDM generates interesting association rules according to the interesting large itemsets, the items specified after the keywords **Antecedent** and **Consequent**, and the minimum confidence specified by the keyword **Confidence** in the user's request. If the keywords **Antecedent** and **Consequent** are not specified in the user's request, i.e., it is a Type III request, then for any interesting large itemset AP, all rules that reference items in the interesting large itemset can be generated. The antecedent of each of these rules is a proper subset SAP of AP, and the consequent is $AP-SAP$. For each $SAP \Longrightarrow AP - SAP$, EDM checks if the confidence achieves the user-specified minimum confidence. If the confidence achieves the minimum confidence, then the rule $SAP \Longrightarrow AP - SAP$ is an interesting association rule.

If both keywords **Antecedent** and **Consequent** are specified in the user's request and there are no notation "*"'s specified in the user's request, i.e., it is a Type I request, then EDM checks if the rule whose antecedent and consequent contain only the items specified after the keywords **Antecedent** and **Consequent**, respectively, is an interesting association rule. If there is only one of the keywords **Antecedent** and **Consequent** specified or there is a notation "*"'s specified in the user's request, i.e., it is a Type II request, then EDM generates rules whose antecedents (consequents) need to match the items specified after the keywords **Antecedent** (**Consequent**).

For example, consider **Request 2**. For the interesting large itemset {A, C, E}, two combinations of the antecedent and consequent match the items specified in **Request 2**: AC $\Longrightarrow$ E and C $\Longrightarrow$ AE. The confidence for C $\Longrightarrow$ AE is $\frac{support for\{A,C,E\}}{support for\{C\}} = \frac{5}{10}$ which is less than the minimum confidence 80%. Hence, C $\Longrightarrow$ AE is not an interesting association rule. However, the confidence for AC $\Longrightarrow$ E is $\frac{support for\{A,C,E\}}{support for\{A,C\}} = 1$ which is greater than 80%. Hence, AC $\Longrightarrow$ E is generated, which is an interesting association rule.

4 Performance Analysis

In this section, we analyze the performance for the efficient data mining algorithm EDM. The efficient data mining technique is implemented in Sun SPARC/10 workstation.

In the large item generation phase, EDM scans the database to find large items from the interested items and build the bit vector for each generated large item. The cost for the first phase is one database scan. For the Type I request, suppose there are k items specified in the **With** clause. After the first phase, EDM performs $(k-1)$ inner products on the bit vectors.

For the Type II and Type III requests, EDM generates interesting large itemsets through the following two phases. For the graph construction phase, suppose there are l large items generated in the first phase. EDM performs $\frac{l \times (l-1)}{2}$ inner products on bit vectors to construct association graph.

For the interesting large itemset generation phase, EDM develops two algorithms LGTE and LGDE to process the Type II request and the Type III request, respectively. In the kth ($k > 2$) iteration, LGDE extends each large $k-1$-itemset into k-itemsets according to the association graph. Suppose the average out-degree of each node is q in the association graph. LGDE performs $(k-1) \times |L_{k-1}| \times q$ inner products to find all large k-itemsets, which has been demonstrated to have a better performance [5] than the other approaches.

For LGTE algorithm, suppose there are n nodes in the constructed expansion tree, and on the average, there are m extended itemsets on each node and the length of each extended itemset is k. LGTE algorithm performs $n \times m \times (k-1)$ inner products to construct expansion tree. Hence, EDM is an efficient algorithm for generating association rules according to the user's request.

5 Conclusion and Future Work

We introduce a data mining language. From the data mining language, users can specify the items in the antecedent and the consequent, and the two criteria: minimum support and minimum confidence of the association rules to be discovered.

We propose an efficient data mining algorithm (EDM) to process a user's request. The algorithm EDM needs only one database scan and some inner products to generate all interesting association rules according to the user's request, which is very efficient.

In the future, we shall extend the data mining language to allow more flexible query specifications, and develop an interactive data mining technique to discover other kinds of association rules according to the user's request, such as generalized association rules and multiple-level association rules.

References

[1] R. Agrawal and et al. Mining Association Rules Between Sets of Items in Large Databases. In *Proceedings of the ACM SIGMOD*, pages 207–216, 1993.

[2] R. Agrawal and R. Srikant. Fast Algorithm for Mining Association Rules. In *Proceedings of the International Conference on Very Large Data Bases*, pages 487–499, 1994.

[3] R. Meo, G. Psaila, and S. Ceri. A New SQL-like Operator for Mining Association Rules. In *Proceedings of the International Conference on Very Large Data Bases*, pages 122–133, 1996.

[4] J.S. Park, M.S. Chen, and P.S. Yu. An Effective Hash-Based Algorithm for Mining Association Rules. *Proceedings of ACM SIGMOD*, 24(2):175–186, 1995.

[5] S.J. Yen and A.L.P. Chen. An Efficient Approach to Discovery Knowledge from Large Databases. In *Proceedings of the International Conference on Parallel and Distributed Information Systems*, pages 8–18, 1996.