在大型交易資料庫中有效提供各種型態的複合項關聯法則之研究

A Study of Supporting Association Rules for Various Composite Itemsets

李建億
國立台南師範學院資訊教育研究所
leeci@ipx.ntntc.edu.tw

曾佩熙
國立台南師範學院資訊教育研究所
pei326@ms67.hinet.net

吳孟淞
國立台南成功大學資訊工程研究所
p7890109@ccmail.ncku.edu.tw

摘要

隨著資料庫日益漸增，要如何從這些大型資料中挖掘出有用的知識，已成為一個重要的研究課題。資料探勘正是目前從資料庫中探索知識的重要技術，其中關聯式法則的探勘又是資料探勘中重要的技術之一，主要是用來發掘各個物件項目之間的關係，以便找出具有價值的規則。然而，傳統的探勘演算法，都要求每一筆記錄中必須包含所有在規則中出現的項目，如 $\{A\} \rightarrow \{B, C\}$，必須同時包含 A、B 和 C 三個項目的記錄才會計算其出現次數。如此，一些潛在的規則便無法被發掘出來，例如在疾病的診斷上，往往因為患者並未完全出現該疾病的所有症狀，而延誤了早期發現早期治療的契機。然而，目前所提出的方法中，僅探討的複合項型態，僅具有 OR 的關係。為了使查詢的結果能更廣泛，在本文中，將提出其他布林運算式 (如 AND 和 NOT 等)複合項關聯式法則探勘，除了能找到原有的關聯式法則之外，還能找出更多傳統探勘演算法所找不到的規則。

關鍵詞：複合項關聯式法則探勘

一、緒論

近年來，隨著科技不斷的創新與進步，電腦已被廣泛的使用在各行各業中，如學校單位、政府機關、公司行號以及大型企業等等。如何從大型資料庫中找出有用的資訊，已成為一個重要的研究課題。資料探勘 (Data Mining) 又稱為資料庫知識發掘 (Knowledge Discovery in Databases)，是從大量資料中挖掘出有用且不易被發現的資訊或知識的一種過程。近幾年來，資料探勘已成為一個新興且熱門的研究焦點 [1, 2, 3, 5, 8, 9, 10, 11, 14, 15, 16, 18]。其中，關聯式法則的探勘是找出資料庫中項目 (Items) 之間的關係，亦即發掘哪些事物會同時發生，最典型的例子就是分析超級市場消費者的購物行為。

關聯式法則的探勘問題敘述如下：給定一個資料庫 D，由 $\|D\|$ 筆交易 (Transactions) 所組成，其中每筆交易都有一個交易識別碼 (Transaction ID，TID) 及一組交易項目，而項目則代表消費者所購買的商品。以數學模式來定義，令 I 是 n 個項目所組成的集合，表示法為 $I = \{i_1, i_2, i_3, \ldots, i_n\}$，每筆交易 $T \subseteq I$。假設 X 表示若干個在 I 中的項目所組成的集合，也就是 I 的子集合(Subset)，我們稱 X 為項目集 (itemset)，若 $X \subseteq T$，則表示一筆交易包含 X。關聯式法則說明資料庫中項目和項目之間的關係，其表示方法為 $X \rightarrow Y$，其中 X 和 Y 都是項目集，且 $X \cap Y = \phi$。我們稱 X 為關聯式法則的前項 (Antecedent)，Y 為法則的後項 (Consequent)。

項目集 X 的支持度 (Support) 表示法為 $S(X)$，其意義是有幾筆交易包含所有在 X 中的項目，此交易數量就是 X 的支持度。給定一個最小支持度 δ (Minimum Support)，若 $S(X) \geq \delta$，則稱 X 為高頻項目集 (Frequent itemset 或 Large itemset)。若

項目集 X 有 k 個項目，則稱 X 為 k-itemset；若 X 為 large，則稱為 large k-itemset (簡記為 L_k)。如果在所有包含 X 的交易中有 c%同時也包含了 Y，則稱關聯式法則 $X \rightarrow Y$ 的信賴度 (Confidence) 為 c%，其表示法為 $S(X \cup Y) / S(X)$。換句話說，支持度是表示統計上的出現頻率，而信賴度則可用來度量規則強弱的標準 [13]。

傳統的探勘演算法，一筆記錄必須包含出現在規則中的所有項目，才會計算其次數 [16, 17]，如法則 $\{A\} \rightarrow \{B, C\}$，表示一筆記錄中必須同時包含 A、B 和 C 三個項目，才能列入計算，因此，一些潛在的規則便無法被發掘出來。以病歷診斷為例，每筆記錄包含病人的症狀和所患的疾病，假設疾病 A 會使病人出現症狀 B，或是出現症狀 C。若出現症狀 B，或是出現症狀 C 的病人，被診斷出患有疾病 A 的人數未超過最小支持度時，則表示不會產生 $\{B\} \rightarrow \{A\}$ 和 $\{C\} \rightarrow \{A\}$ 這二條關聯規則。但若出現 B 症狀，或是出現 C 症狀而被診斷出患有 A 疾病的病患人數超過最小支持度時，則結合 B 症狀和 C 症狀可以產生一個新的項目 $B \vee C$（亦即一筆交易包含 B 或 C 中任何一項，則此筆交易就包含了新產生的項目 $B \vee C$）。因此，可以找出 $\{B \vee C\} \rightarrow \{A\}$ 這條關聯規則。

所謂複合項 (Composite Item，如項目 $B \vee C$) 是指由原本的幾個項目所組成的。而複合項關聯式法則探勘 (Mining Association Rules with Composite Items) [16]，就是允許在原來的項目集中，可以包含 $B \vee C$ 此類型的複合項，其作法是經由使用者先提出若干個項目來產生高頻複合項 (Large Composite Items)，然後再找出所有的高頻項目集。

在傳統的關聯式法則探勘中並沒有考慮到複合項，而先前所探討的複合項關聯式法則探勘僅具有 OR 關係，因此，在本文中，將提供其他布林運算式 (如 AND 和 NOT) 和特殊型態 (如 XOR 和 XNOR) 等複合項關聯式法則探勘。

二、文獻探討

在本章，將探討相關文獻的關聯式法則探勘演算法及複合項關聯式法則探勘演算法。

（1）Apriori 演算法

目前，已有許多的演算法被提出來找出關聯式法則，其中以 Apriori 演算法最為眾人所知。為了改善計算候選 k 項目集 (Candidate k-itemset，簡記為 C_k) 出現次數的效率，Apriori 演算法 [2] 產生 C_k 的方式是任取二個 large k-1 itemsets (簡記為 L_{k-1})，其前面 k-2 個項目必須相同，只有最後一個項目不同，此過程稱為合併 (Join)。接下來，則檢查這個 k-itemset 的任何一個次集合是否都存在於 L_{k-1} 的集合中，如果這個條件成立，則將該 k-itemset 列入 C_k 的集合中；反之，則將該 k-itemset 除去，此過程稱之為修剪 (Prune)。最後，再掃瞄整個資料庫以計算所有 C_k 的出現次數，當 C_k 的出現次數高過最小支持度的門檻時，則將其列入 L_k 的集合中。一直重複此程序，直到不再有新的 C_{k+1} 產生為止。

（2）複合項關聯式法則探勘演算法

令 I 為 n 個基本項 (Atomic Items) 所成的集合，表示法為 $I = \{a_1, a_2, a_3, \ldots, a_n\}$，一個複合項是由若干個基本項組成的，其表示法為 $a_1 \vee a_2 \vee \ldots \vee a_k$，其中 $a_j \in I$ for $1 \leqq j \leqq k$，且 $a_i \neq a_j$ for $i \neq j$ 時。例如，$A \vee B \vee C$ 為複合項，而 A，B 和 $C \in I$。一般而言，不論是基本項或是複合項，皆簡稱為項目。假如一筆交易中包含至少一個組成複合項中的基本項，則此筆交易會使複合項的出現次數加 1。複合項關聯式法則探勘 [16] 的作法是允許使用者輸入若干個項目，然後掃瞄資料庫多次後產生所有的高頻複合項，接著再以 Apriori 演算法為基礎，以產生所有高頻項目集。

三、適用於計算複合項的資料結構

在本節中，將提出一個適用於計算含有複合項的資料結構。

（一）複合項的 Trie 結構

在本節中，將先說明 Apriori 演算法中的 trie 結構，並根據其特性，提出一個適用於計數含有複合項候選項目集的 trie 結構，以有效地計算候選項目集之出現次數。

（1）Apriori 的 Trie 架構

為了有效地計算所有 C_k 的出現次數，Aproiri 使用了 trie 的結構以加速計算過程，trie 和樹 (Tree) 的資料結構很相似，是由一群節點 (Nodes) 所構成，每個非葉節點 (None Leaf Node) 皆有鏈結 (Link) 連接到子節點，而且每一個節點的子節點個並不一定相同，葉節點 (Leaf Node) 存放著此候選項目集。

例如，假設 C_3 的集合為 $\{\{A, B, C\}, \{B, C, D\}, \{B, D, E\}, \{C, D, E\}\}$，其 trie 的結構如圖 1 所示。

（2）基本問題描述

若一筆交易中包含至少一個組成複合項中的基本項，則此筆交易會使複合項的出現次數加 1。例如，一筆交易為 *ADE*，則使得複合項 $A \vee B$ 的出現次數加 1。為了簡便的計算複合項的出現次數，在本中先定義以下的符號。

定義：令複合項 $CI = X_1 \vee X_2 \vee \ldots \vee X_q$，其中 X_i $(1 \leqq i \leqq q)$ 皆為基本項，CIC_k 表示組成複合項的基本項之集合，則 $CIC_k = \{ X_1, X_2, \ldots, X_q \}$；$TN(\{CI\})$ 表示一筆交易包含在 CIC_k 中的基本項出現次數總和，若一筆交易有包含 X_i，則 *TN* 值加 1。

因此，若 $TN(\{CI\}) \geqq 1$，則表示一筆交易包含複合項，因此會使晶複合項的出現次數加 1。例如，一筆交易為 *ABD*，則 $TN(\{A \vee B \vee E\}) = 2$，$A \vee B \vee E$ 出現的次數會加 1。

對於包含一個複合項的 C_2 如 $\{A, B \vee C\}$，可將它展開成 $\{A, B\} \vee \{A, C\}$，因此，使得 C_2 出現的次數加 1 的交易中，至少包含 $\{A, B\}$ 或至少包含 $\{A, C\}$。同樣地，當 $TN(C_2) \geqq 1$ 時，此複合項的次數會加 1。例如，一筆交易為 *BDF*，則 $TN(\{B, D \vee E\}) = 1$，含有複合項的 2-itemset $\{B, D \vee E\}$ 的次數會加 1。對於任何含有複合項的 $C_k (k \geqq 3)$ 而言，同理，只要其 $TN(C_k) \geqq 1$ 時，此複合項的次數亦會加 1。

（3）適用於計數複合項的資料結構

由上一小節可知，對於含有複合項的 $C_2 = \{A, B \vee C\}$ 而言，將它展開後可得 $\{A, B\} \vee \{A, C\}$，$\{A, B\}$ 和 $\{A, C\}$ 這二個 2-itemsets，在傳統的關聯式法則探勘中會被視為二個不同的候選項目集並分別存放在二個不同的葉節點。然而，在複合項關聯式法則探勘中，由於 $\{A, B\}$ 和 $\{A, C\}$ 都會使得 $\{A, B \vee C\}$ 的次數加 1，因此，在建構 trie 的過程中，將 $\{A, B\}$ 和 $\{A, C\}$ 這二個候選項都鏈結到 $\{A, B \vee C\}$。

再則，對於項目集 $\{A, B \vee C\}$ 和 $\{A, B \vee D\}$ 而言，展開後的 $\{A, B\}$ 都會使得 $\{A, B \vee C\}$ 和 $\{A, B \vee D\}$ 的次數加 1，所以，在建構 trie 的過程中，當發現 $\{A, B\}$ 這個候選項目集已有鏈結時，則會配置一個鏈結串列 (Link List)，再鏈結到 $\{A, B \vee C\}$，以節省記憶體的使用空間。除了原有的項目集欄位和支持度欄位之外，每個候選項目集還需要一個旗標欄位 (Flag) 來記錄 *TN* 值（初始值為 0），和一個判斷欄位（*AC*，表示基本項或複合項）來記錄此候選項目集是否含有複合項。

在讀入一筆交易後，從 trie 的樹根開始走訪，依項目集中的項目走訪邊 (Edge)，若是到達中間節點 (Internal Node)，則選擇項目集中的下一個項目的邊繼續走訪，直到到達葉節點為止。若繞到葉節點，則先判斷 *AC* 欄位的值，假如其值為 1，表示選項目集沒有包含複合項，則直接把它的出現次數加一。若不等於 1，表示項目集有複合項，則把此候選項目集鏈結到的複合項的 *TN* 值累加。

當處理完一筆交易之後，則把 *TN* 值大於等於一的候選項目集的出現次數加 1，並將其 *TN* 值歸零，接著再讀入下一筆交易繼續處理。重覆上述步驟，當掃瞄完資料庫後，則計算出所有候選項目集的出現次數。其建構 trie 的演算法如圖 3 所示。由上述可知，為了計算含有複合項候選項目集的出現

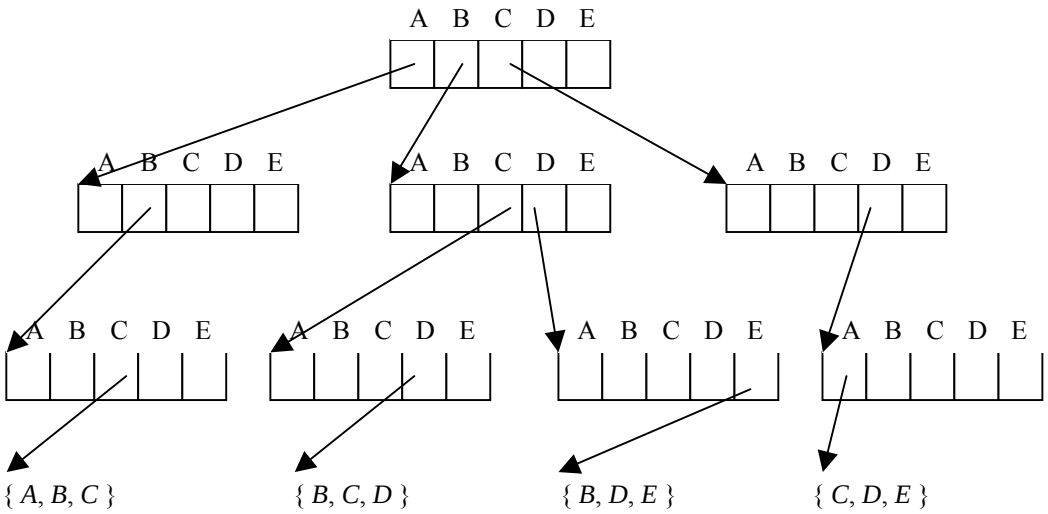

圖 1.、Apriori 的 trie 結構

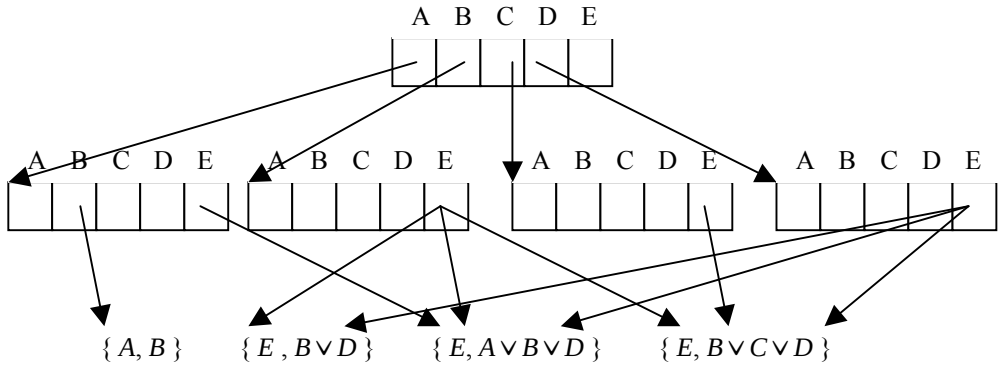

圖 2、基本複合項的 trie 架構

次數，其資料結構和 Apriori 使用的 trie 結構類似，且須在每個節點需要記錄一些額外的資訊。假設 C_2 的集合為 { {A, B} ,{E , B∨D}, {E, A∨B∨D}, {E, B∨C∨D}，其 trie 的結構如圖 2 所示。

```
演算法 : 累加候選項目集的出現次數
輸入 : 給定一筆交易 t ( with items t[0] … t[n] )，
          Trie T
說明 : 累加 candidates 的出現次數
Increment(T, t)
{
 if T is a leaf node
    if AC = = 1
       T.count++;    //Increment this node's count
    else
       T.flag++;    //Increment this node's flag
 if T is not a leaf
    for each i, 0 ≤ i ≤ n
    {   //Increment branches as necessary
       if T.branches[ t[i] ] exists
          Increment( T.branches[ t[i] ], t[i+1 … n] )
    end_for
  return;
}
 if T.falg ≧ 1
 {
    T.count++;
    T.flag = 0;
 }
```

圖 3、建 trie 的演算法

（二）其他型態的複合項

早期的複合項關聯式法則演算法，使用者輸入感興趣的項目，系統會自動產生所有的高頻複合項，接著再產生所有的高頻項目集。然而，若使用者感興趣的項目為 m 個，則可能產生的高頻複合項的個數為 $2^m - m - 1$ 個，這和使用者輸入的基本項個數呈指數關係而遽增。因此，本文提出一個新的方法，將不再產生所有的高頻複合項，而是允許使用者針對自己感興趣的項目，輸入多個不同的複合項來作查詢，然後再繼續產生所有的高頻項目集。對於使用者要求查詢的複合項，提出 *FLCIS* (Finding Large Composite ItemSets) 演算法來產生所有的高頻項目集，其方法分為下列三個階段：

1. 第一階段：*LAI* 產生階段 (Large Atomic Item Generation Phase)。在此步驟中會先掃瞄資料庫一次以產生所有的 large atomic items，也就是找出傳統演算法中的 L_1。
2. 第二階段：輸入複合項階段 (Input Composite Items Phase)。在此步驟直接輸入使用者感興趣的複合項。
3. 第三階段：L_k 產生階段 (Large Itemsets Generation Phase)。將 *LAI* 和感興趣的複合項當作 L_1，以 *Apriori* 為基礎，再加以修正使其符合產生含包複合項的 L_k。

（1）具有 AND 關係的複合項

對複合項 $A \vee BC$ 而言，其意義是一筆交易只要包含 A，或包含 BC (B 和 C 一起出現)，則此筆交易才會使得 $A \vee BC$ 的次數加 1。

對於 2-itemset $\{A, B \vee CD\}$ 而言，可將它展開為 $\{A, B\} \vee \{A, C, D\}$，因此，只要一筆交易至少包含 $\{A, B\}$ 或 $\{A, C, D\}$，其出現次數加 1。所以，在建構 trie 的過程中，將 $\{A, B\}$ 和 $\{A, C, D\}$ 都鏈結到 $\{A, B \vee CD\}$，以方便計算其出現次數。

對於含有複合項的 $C_k (k \geq 3)$，計算其次數的方式同上節所述，只是在含有 AND 關係的項目時，要一起出現才會能把 *TN* 值加 1。例如，C_2 的集合為 $\{\{D, E\}, \{D, A \vee BC\}, \{E, A \vee BC\}\}$，其 trie 的架構如圖 4 所示。

舉例說明，找出含有 AND 關係複合項的高頻項目集。交易資料庫 D 如表 1 所示，假設最小支持度為 4。

表 1、範例資料庫

TID	Items
T1	*A D E F*
T2	*C D E F*
T3	*A B E F*
T4	*C D E F*

先掃瞄資料庫，以產生 *LAI*。假設使用者輸入的複合項為 $A \vee CD$。經過合併之後產生 C_2，並掃瞄資料庫以產生 L_2；再由 L_2 產生 C_3，並掃瞄資料庫產生 L_3。由於 L_3 無法產生 C_4，整個程序就停止，其執行過程如圖 5 所示。從結果可得知，除了可以找出 $L_2 = \{E, F\}$ 之外，還可以找到 $\{E, A \vee B \vee CD\}$ 和 $\{F, A \vee B \vee CD\}$；也就是說，傳統演算法只能找到 L_2，而複合項探勘可以找到如 $L_3 = \{E, F, A \vee B \vee CD\}$ 等，更多的高頻項目集。

（2）具有 NOT 關係的複合項

對複合項 $A \vee B\overline{C}$ 而言，其意義是一筆交易只要包含 A，或只要包含 B 但一定不包含 C，則此筆交易才會使得 $A \vee B\overline{C}$ 出現的次數加 1。其 *TN* 值的計算方式如下：交易若有包含 NOT 基本項 (如 C) 則 *TN* 值減 1，包含非 NOT 基本項 (如 A、B) 則 *TN* 值加 1。如此，若交易中若有包含 B 和 C，則 B 會 *TN* 值加 1，C 會使 *TN* 值減 1，剛好正負相減抵消，因此，此筆交易不會使複合項的次數加 1。例如，對於 2-itemset $\{E, A \vee B\overline{C}\}$ 而言，將它展開可得 $\{A, E\} \vee \{B, E\} \vee \{B, C, E\}$，因此，若一筆交易若有包含 $\{A, E\}$ 或 $\{B, E\}$，則 TN 值加 1，若一筆交易中有包含 $\{B, C, E\}$，則 TN 值減 1。對於任何含有一個複合項的 $C_k (k \geq 3)$，其計算方式同上所述。例如，C_2 的集合為 $\{\{D, E\}, \{D, A \vee B\overline{C}\}, \{E, A \vee B\overline{C}\}\}$，其 trie 的架構如圖 6 所示。

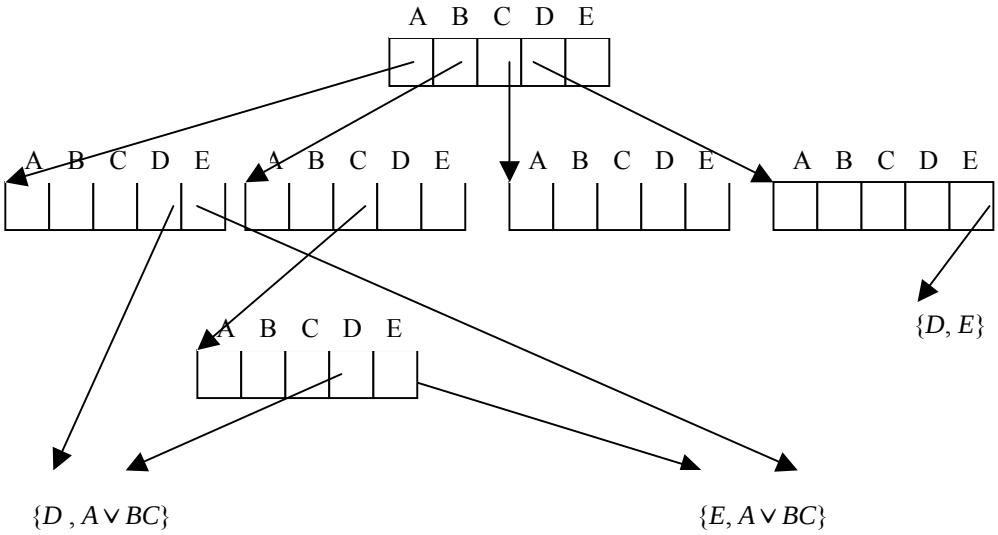

圖 4、具有 AND 關係複合項

C_1

Itemset	Support
{*A*}	2
{*B*}	1
{*C*}	2
{*D*}	3
{*E*}	4
{*F*}	4

L_1

Itemset	Support
{*E*}	4
{*F*}	4
{*A*∨*CD*}	-

C_2

Itemset	C_2(展開後)	Support
{*E*, *F*}	-	4
{*E*, *A*∨*CD*}	{*AE*, *CDE*}	4
{*F*, *A*∨*CD*}	{*AF*, *CDF*}	4

L_2

Itemset	Support
{*E*, *F*}	4
{*E*, *A*∨*CD*}	4
{*F*, *A*∨*CD*}	4

C_3

Itemset	C_3(展開後)	Support
{ *E*, *F*, *A*∨*CD* }	{*AEF*, *CDEF*}	4

L_3

Itemset	Support
{*E*, *F*, *A*∨*CD*}	4

圖 5、具有 AND 關係的複合項執行過程

舉例來說，假設交易資料庫 D 如表 2 所示，最小支持度設為 4。

表 2、範例資料庫

TID	Items
T1	*A B E F*
T2	*B C E F*
T3	*C D E F*
T4	*B E F*
T5	*A C D E F*

首先掃瞄資料庫一次，以產生 LAI，假設使用者輸入的複合項為 $A \vee B\overline{C}$ 。接著經過聯結之後產生所有的 C_2，並掃瞄資料庫以產生 L_2。再由 L_2 產生 C_3，並掃瞄資料庫產生 L_3，由於 L_3 無法產生 C_4，所以整個程序就停止，整個執行過程如圖 7 所示。

四、實驗結果與討論

（一）實驗設計

使用 IBM 公司所提供的模擬料庫產生器[1]產生幾組模擬的交易資料庫，其相關參數的意義，如表 3 所示。

表 3、資料庫產生器參數的意義

變數名稱	意義
$\lvert D \rvert$	Number of transactions
$\lvert T \rvert$	Average size of the transactions
$\lvert I \rvert$	Averagesize of the maximal potentially large itemset
$\lvert L \rvert$	Number of maximal potentially large itemsets
N	Number of items

在模擬的資料庫中，設定 $n = 1000$ 以及 $\lvert L \rvert =$ 2000，$\lvert T \rvert$ 則選用三種數值：5，10 和 20，也同時為 $\lvert I \rvert$ 選用三種數值：2，4 和 6，交易的數量均設定為 100,000 筆。將所有模擬出的 5 種資料庫之參數設定值表列於表 4。

表 4、模擬資料庫使用的參數

資料庫種類	$\lvert T \rvert$	$\lvert I \rvert$	$\lVert D \rVert$	*資料庫大小*
T5.I2.D100K	5	2	100K	3.0MB
T10.I4.D100K	10	4	100K	5.0MB
T10.I6.D100K	10	6	100K	5.0MB
T20.I4.D100K	20	4	100K	8.9MB
T20.I6.D100K	20	6	100K	8.9MB

（二）模擬實驗結果

在實驗中，將最小支持度的區間設成 2%～0.33%之間，並假設輸入的複合項個數為 1 個。對於輸入具有 AND 關係的複合項為 $A \vee B \vee CD$ 形式，輸入具有 AND、NOT 關係的複合項為 $A \vee B \vee C\overline{D}$ 形式，由圖 8 及圖 9 可以發現，隨著最小支持度的門檻降低，二者的執行時間也隨著上升，這是由於門檻調低之後，候選項目集與高頻項目集的數量也隨之增加，所以需要更多的執行時間。

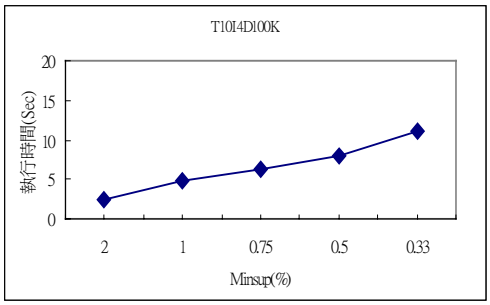

圖 8

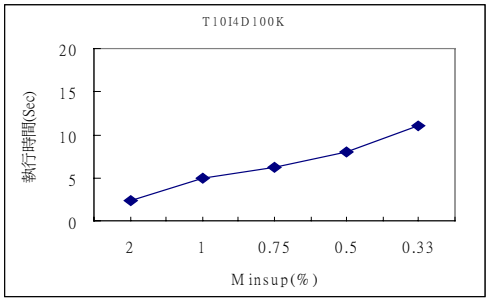

圖 9

[1]其下載網址為：
http://www.almaden.ibm.com/cs/quest/syndata.html

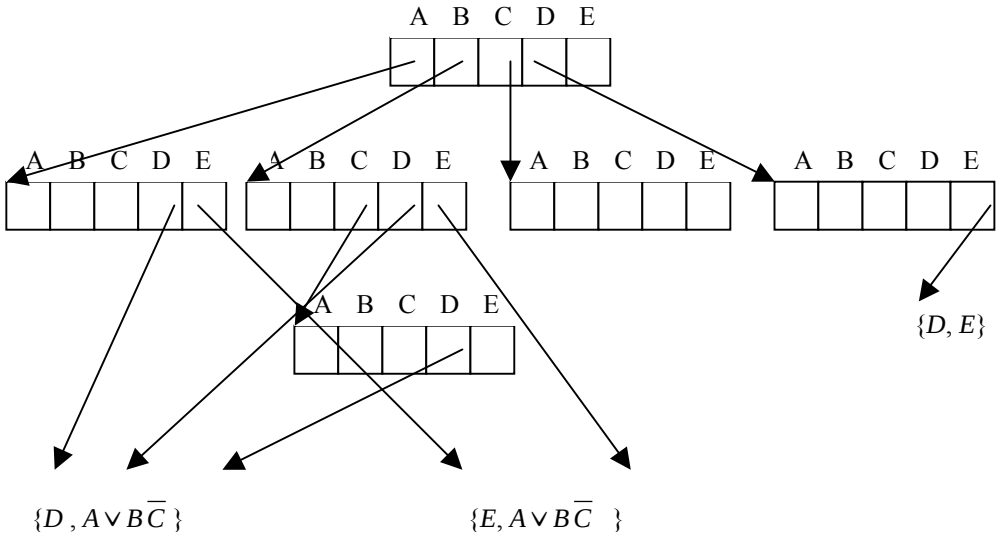

圖 6、具有 NOT 關係的複合項的架

C_1

Itemset	Support
{ A }	3
{ B }	3
{ C }	2
{ D }	2
{ E }	5
{ F }	5

L_1

Itemset	Support
{ E }	5
{ F }	5
{ $A \vee B\overline{C}$ }	-

C_2

Itemset	C_2(展開後)	Support
{E, F}	-	5
{E, $A \vee B\overline{C}$ }	{AE, BE, BCE}	4
{F, $A \vee B\overline{C}$ }	{AF, BF, BCF}	4

L_2

Itemset	Support
{E, F}	5
{E, $A \vee B\overline{C}$ }	4
{F, $A \vee B\overline{C}$ }	4

C_3

Itemset	C_3(展開後)	Support
{E, F, $A \vee B\overline{C}$ }	{AEF, BEF, BCEF}	4

L_3

Itemset	Support
{E, $A \vee B\overline{C}$ }	4

圖 7、具有 NOT 關係的複合項執行過程

六、結論及未來研究方向

複合項關聯式法則的探勘，其作法是允許使用者輸入若干個項目，產生所有的高頻複合項，然後再產生所有的高頻項目集。然而，由於產生高頻複

合項的個數和使用者輸入的項目個數呈指數關係，在探勘過程會耗費更多的時間。因此，在本文中，允許使用者輸入感興趣的複合項，作為查詢項，來找出高頻項目集，並且以 *Apriori* 演算為基礎，提出適用於計數各種型態的複合項的資料結構。

未來研究的方向：

1. 考慮權重（Weight）的問題，對於不同的項目而言，可能有不一樣的重要性，而應該給予不同的權重。

2. 考慮個數關係的複合項，如 $2A \vee 3B$，表示二個 A 或三個 B。

3. 動態資料庫，當資料庫有變動時，不論是項目增加或減少，就必須要重新作探勘。

七、參考文獻

[1] R. Agrawal, T. Imiclinski, and A. Swami. Mining Association Rules between Sets of Items in Large Databases. *Proc. of ACM SIGMOD*, pp. 207-216, 1993.

[2] R. Agrawal, and R. Srikant. Fast Algorithm for Mining Association Rules. *Proc. of International Conference on Very Large Database*, pp.487-499, 1994.

[3] R. Agrawal and R. Srikant. Mining Sequential Patterns. *Proc. Of 11th International Conference on Data Engineering*, 1995.

[4] Michael J. A. Berry, and Gordon S. Linoff. Data Mining Techniques for Marketing, Sales and Customer Support.

[5] S. Brin, R. Motwani, J. Ullman, and S. Tsur. Dynamic Itemsets Counting and Implication Rules for Market Basket Data. *SIGMOD-97*, pp. 255-264, 1997.

[6] M.-S. Chen, J. Han, and P. S. Yu. Data Mining: An Overview from Database Perspective. *IEEE Transactions on Knowledge and Engineering*, Vol. 8, no.6, pp. 866-883, 1996.

[7] J. Han, and M. Kamber. Data Mining : Concepts and Techniques.

[8] M. Houstma, and A. Swami. Set-Oriented Mining for Association Rules in Relational Database. *IEEE 11th International Conference on Data Engineering*, pp. 25-33, 1995.

[9] H. Mannila, H. Toivonen, and A. Inkeri Verkamo. Efficient Algorithms for Discovering Association Rules. *In AAAI'94 Workshop on Knowledge Discovery in Database*(*KDD-94*), pp. 181-192, 1994.

[10] J. S. Park, M.-S. Chen, and P. S. Yu. An Effective Hash-Based Algorithm for Mining Association Rules. *Proc ACM SIGMOD*, pp.175-186, 1995.

[11] J. S. Park, M.-S. Chen, and P. S. Yu. Using a Hash-Base Method with Transaction Trimming for Mining Association Rules. *IEEE Transactions on Knowledge and Data Engineering*, pp. 813-825, 1997.

[12] A. Savasere, E. Omiecinski, and S. Navathe. An Efficient Algorithm for Mining Association Rules in Large Database. *Proc. of the 21st VLDB Conference*, pp. 432-444, 1995.

[13] S. Y. Sung, K. Wang, B. I. Chua. Data mining in a large database environment. Systems, *Man and Cybernetics, 1996. IEEE International Conference* Vol 2, pp. 988 –993, 1996.

[14] J. Tang. Using Incremental Pruning to Increase the Efficiency of Dynamic Itemsets Counting for Mining Association Rules. *CIKM-98*, pp. 273-280, 1998.

[15] S.-Y. Wur, and Y. Leu. An Effective Boolean Algorithm for Mining Association Rules in Large Databases. *Proceedings, 6th International Conference on Database Systems for Advanced Applications*, *pp.*179-186, 1999.

[16] X. Ye, and J. A. Keane. Mining Association Rules with Composite Items. *Systems, Man, and Cybernetics*. *Computational Cybernetics and Simulation. IEEE International Conference* Vol. 2, pp. 1367 – 1372, 1997.

[17] X. Ye, and J. A. Keane. Mining Association Rules in Temporal Databases. *Systems, Man, and Cybernetics*. *IEEE International Conference* Vol. 3, pp. 2803 – 2808, 1998.

[18] S.-J. Yen, and Arbee L. P. Chen. An Efficient Approach to Discovering Knowledge from Large Database. *Int'l Conference on Parallel and Distributed Information System*, pp. 8-18, 1996.