從交易資料庫中探勘具代表性的頻繁項目集

顏秀珍 李御璽 吳政瑋
銘傳大學 銘傳大學 銘傳大學
資訊工程學系 資訊工程學系 資訊工程學系
sjyen@mail.mcu.edu.tw leeys@mail.mcu.edu.tw silvemoonfox@hotmail.com

摘要

探勘頻繁項目集(Frequent Itemset Mining)是資料探勘(Data Mining)中一門重要的研究方向。從交易資料庫(Transaction Database)中探勘頻繁項目集(Frequent Itemsets)的主要目的是找出資料庫中經常一起被購買的商品，也就是相關的商品。然而，傳統的頻繁項目集探勘往往在探勘過程中產生太多的頻繁項目集，不僅降低執行速度並佔用許多記憶體。而且過多的頻繁項目集，往往造成決策者的困擾。因此有學者提出頻繁封閉項目集(Frequent Closed Itemset)的觀念。頻繁封閉項目集是頻繁項目集的一種精簡表示法，且數量往往較頻繁項目集的數量少。在這篇論文中，我們提出一個從資料庫中有效率探勘頻繁封閉項目集的演算法，以及一個有效率從頻繁封閉項目集產生所有頻繁項目集的方法。

關鍵詞：頻繁項目集、頻繁封閉項目集、資料探勘、交易資料庫。

Abstract

Mining frequent itemsets is an important task to data mining. However, there are too many patterns generated from database, and many of them are redundant. Frequent closed itemsets is a well-known condensed representation of frequent itemsets and provides complete information for frequent itemsets. In general, the number of frequent closed itemsets is much smaller than the number of frequent itemsets. Therefore, mining only the closed itemsets reduces the processing time and memory space. Besides, frequent closed itemset has been proven to be more meaningful and representative for analysis. In this paper, we propose an efficient algorithm for mining frequent closed itemsets.

Keywords: data mining, frequent closed itemsets, frequent itemsets.

1. 前言

資料探勘 (data mining) [1, 2, 4] 是從大量資料中發掘出潛在、未知、且可能有用的資訊。*從交易資料庫*(transaction database)中探勘*頻繁項目集*(frequent itemset) [1, 2]，在資料探勘領域中是一個重要的研究議題。其相關的定義如下: 給定一個含有 N 筆交易的交易資料庫 D，每一筆交易 $t \in D$ 代表顧客一次購買商品的紀錄。令 $I = \{a_1,a_2,\ldots,a_M\}$ 為所有*項目*(Item)的集合。每一個項目可視為一種商品。每一筆*交易* $t = \{i_1,i_2,\ldots,i_m\}$ ($i_j \in I$，$1 \leqq j \leqq m$)，是由數個相異項目所組成，且每筆交易具有唯一的識別碼稱為 Tid (transaction identifier)。一個*長度為 k 的項目集* (k-itemset) $X = \{i_1,i_2,\ldots,i_k\}$ ($i_j \in I$，$1 \leqq j \leqq k$)，是包含 k 個相異項目的集合。一個項目集 X 的*支持數*(support count)為資料庫中包含 X 的交易筆數，以 SC(X)表示之；X 的支持度(support)為 X 的支持數除以總交易筆數。一個項目集的支持度若不小於使用者定義的*最小支持度門檻值 min_sup* (minimum support)，則稱此項目集為一個*頻繁項目集*。反之，則稱為*非頻繁項目集* (infrequent itemset)。在交易資料庫中，頻繁項目集可視為顧客經常一起購買的商品，也就是相關的商品。

以表一為例，並假設最小支持度為 50%時，最小支持數(minimum support count) 為 (50% * 6) = 3。所以頻繁項目集有{C}:6，{W}:5，CW:5，{A}:4，{D}:4，{T}:4，{AC}:4，{AW}:4，{CD}:4，{CT}:4，{ACW}:4，{AT}:3，{DW}:3，{TW}:3，{ACT}:3，{ATW}:3，{CDW}:3，{CTW}:3 與 {ACTW}:3。

表一. 交易資料庫

Tid	Transaction
1	A,C,T,W
2	C,D,W
3	A,C,T,W
4	A,C,D,W
5	A,C,D,T,W
6	C,D,T

格式化: 項目符號及編號

從鬆散型資料庫(sparse dataset)中探勘頻繁項目集往往可以獲得不錯的效能，理由是因為鬆散型資料庫中的頻繁項目集長度往往較短，即使將最小支持度設定的很低，頻繁項目集的數量仍然不會很多。許多文獻認為交易資料庫是屬於鬆散型資料庫。然而，*密集型資料庫*(dense dataset)往往導致探勘的效能低落，理由是因為密集型資料庫中的頻繁項目集長度往往較長，而且數量很多，即使將最小支持度的門檻值設定的很高，仍然會產生許多長度較長的頻繁項目集。在實務上許多資料庫屬於密集型資料庫[7]，例如人口統計資料(census dataset)，遊戲連線紀錄的資料庫都屬於密集型資料庫。從密集型資料庫中往往探勘出大量的頻繁項目集，不僅造成決策者的困擾，並且降低探勘的執行效能。

其中一種解決方式是探勘資料庫中的*頻繁封閉項目集*(frequent closed itemset)。一個項目集 X 的支持數若大於其所有超項目集的支持數，則稱此項目集為一個封閉項目集(closed itemset) [4]。反之，則 X 不為封閉項目集，也就是*非封閉項目集*(non-closed itemset)。若一個封閉項目集的支持度不小於最小支持度，則稱此封閉項目集為*頻繁封閉項目集*。

以表一為例，當最小支持數為 3 時，頻繁封閉項目集為{C}:6，{CW}:5，{CD}:4，{CT}:4，{ACW}:4，{CDW}:3 與{ACTW}:3。其中，項目集{D}與項目集{CD}的支持數相同，所以{D}為不為封閉項目集。非封閉項目集同時也是冗餘的，以{D}為例，{D}的支持數為 4，與{CD}的支持數相同，表示每一筆包含{D}的交易都包含{CD}。從交易資料庫的應用而言，顧客並不會單獨購買{D}而不買{C}，而會同時購買{CD}。

此外，所有的頻繁封閉項目集及其支持度，可推導出所有的頻繁項目集及其支持度[4]。所以只探勘頻繁封閉項目集並不會遺失頻繁項目集的資訊。而且，頻繁封閉項目集的數量比頻繁項目集的數量少，所以不僅減少大量冗餘的頻繁項目集，也減少探勘所需的時間與記憶體的使用量，更利於管理者作決策。

在這篇論文中，我們提出一個從資料庫中有效率探勘頻繁封閉項目集的演算法 CLMiner (Cid List based approach for Mining frequent Closed itemsets)。並且提出 DFI (Deriving Frequent Itemsets from closed itemsets) 演算法，從頻繁封閉項目集快速地產生所有的頻繁項目集。這兩個方法的效能皆優於先前的研究[4, 6]。

2. 相關定義

格式化: 項目符號及編號

在本章節中我們介紹封閉項目集的相關定義與性質。

定義 1 (封閉運算子)
假設 T 為資料庫 D 之交易的子集合，T ⊆D，Y 為 D 內所有項目的子集合，$Y \subseteq I$。一個封閉項目集可經由 f 與 g 兩個函式運算而得。函式 f 的定義如下:

$$f(T) = \{i \in I \mid \forall t \in T, i \in t\}$$

函式 f 回傳一個項目集，且此項目集包含於每一筆屬於 T 的交易中。函式 g 的定義如下:

$$g(Y) = \{t \in D \mid \forall i \in Y, i \in t\}$$

給定一個項目集 Y，函式 g 回傳一組交易的集合，此集合內的每一筆交易都包含項目集 Y。函式 C 由函式 f 與函式 g 所合成，也就是 $C = f \circ g$，我們稱 C 為一個*封閉運算子*(closure operator) [4]。

以表一的資料庫為例，若項目集{A}為函數 g 的輸入引數，由於 Tid 編號為 1、3、4 與 5 的交易都包含項目集{A}，所以 $g(\{A\}) = \{1, 3, 4, 5\}$。若以{1, 3, 4, 5}為 f 的輸入引數，則 $f(\{1, 3, 4, 5\})$ = {ACTW} ∩ {ACTW} ∩ {ACDW}∩ {ACDTW} = {ACW}，也就是包含項目集{A}的交易也都包含項目集{ACW}。

定義 2 (封閉項目集)
一個項目集 X 被稱為*封閉項目集*，必須滿足下列條件:

$$C(X) = f \circ g(X) = f(g(X)) = X$$

若項目集 X 為函式 C 的輸入引數，透過*函式 C* 而得到項目集 X，則 X 為一個封閉項目集；反之，項目集 X 不為封閉項目集。以表一的資料庫為例，並以項目集{A}為函式 C 的輸入引數，則 $C(A) = f(g(\{A\})) = \{AB\}$。由於項目集{A}透過函式 C 得到項目集為{AB}，所以項目集{A}並不是一個封閉項目集。我們稱項目集{AB}是項目集{A}的封閉項目集。對於一個項目集 X，我們以符號 $C(X)$來表示 X 的封閉項目集。依據上述的定義，我們有以下的性質。

性質 1: 假設項目集 Y 為項目集 X 的封閉項目集，也就是 $Y = C(X)$，則 SC(X) = SC(Y)。
理由: 因為項目集 Y 為項目集 X 的超項目集，

所以所包含項目集 Y 的交易必定都包含 X。又因 Y = *C*(X)，由定義 1 可知，包含 X 的交易也都包含 Y。因此，X 與 Y 的支持數相同，也就是 SC(X) = SC(Y)。

性質 2: 對於任一個項目集 X，必定存在唯一的一個封閉項目集 Y，使得 Y = *C*(X)。
理由: 假設項目集 Y 為項目集 X 的封閉項目集 *C*(X)= Y，根據性質 1，包含 X 的交易必定也都包含 Y。假設存在一個項目集 X 的超項目集 Z，Z≠Y，且 Z = *C*(X)，則包含 X 的交易也都包含 Z。因此包含 X 的交易必定都包含 Y∪Z，所以 C(X) ≠Y，這與前提假設相違背，所以 Y 必定是 X 唯一的一個封閉項目集。

3. CLMiner 演算法

在本章節中，我們描述 CLMiner 的儲存結構並詳細介紹我們所提出的演算法 CLMiner。

3.1 CLMiner 的儲存結構

CLMiner 的儲存結構包括 Tid List，Cid List 與 Sup Table。Tid List 包括 Item 欄位與 Tidset 欄位。Item 欄位紀錄項目 I，Tidset 欄位紀錄包含該項目的交易編號。我們以符號 t(I)表示項目集 I 的 Tidset，並以|t(I)|表示 t(I)中 Tid 的個數，也就是 I 的支持數。藉由交集項目集 I_i 與 I_j 的 Tidset，我們可以得到 $I_i \cup I_j$ 的 Tidset。

表二. Tid List

Item	Tidset
A	1,3,4,5
D	2,4,5,6
T	1,3,5,6
C	1,2,3,4,5,6
W	1,2,3,4,5

表二為以表一建構出來的 Tid List。其中，t({A}) = {1, 2, 3, 4}，表示項目 A 出現在 Tid 為 1，2，3 與 4 的交易中，其支持數為 4。同理，t({D}) = {2, 4, 5, 6}。我們可以得到 t({A∪D}) = {1, 2, 3, 4}∩{2, 4, 5, 6} = {2, 4}，其支持數為 2。利用 Tid List 可找出可能成為頻繁封閉項目集的頻繁項目集。

當 CLMiner 找到一個頻繁封閉項目集時，將會給予該封閉項目集一個獨特的編號，也就是 cid (closed itemset identifier)。Cid List 包括 Item 欄位與 Cidset 欄位。Item 欄位紀錄頻繁項目 I，Cidset 欄位紀錄包含 I 的頻繁封閉項目集編號，我們以符號 c(I)表示項目集 I 的 Cidset。藉由交集項目集 I_i 與 I_j 的 Cidset，我們可以得到 $I_i \cup I_j$ 的 Cidset。

表三為一個 Cid List，其中項目 A 的 Cidset 為{1, 2}，也就是 c({A}) = {1, 2}。表示項目集{A}存在兩個超封閉項目集，其 Cid 編號分別為 1 與 2。同理，c({T}) = {2}，c({A∪T}) = {1, 2}∩{2} = {2}，所以項目集{AT}的超封閉項目集為 Cid 編號 2 的封閉項目集。

表三. Cid List

Item	Cidset
A	1,2
D	3,4
T	2
C	1,2,4
W	1,2,3,4

表四. Sup Table

Cid	SC
1	4
2	3
3	4
4	3

當 CLMiner 找到一個頻繁封閉項目集時，Sup Table 會紀錄該封閉項目集的 Cid 編號與支持數。Sup Table 的儲存結構包括 Cid 欄位與 SC 欄位。Cid 欄位記錄封閉項目集的編號，SC 欄位紀錄該封閉項目集的支持數。表四為一個 Sup Table，其中 Cid 為 1 的封閉項目集其 SC 值為 4，代表其支持數為 4。

透過 Tid List，CLMiner 可找出可能成為頻繁封閉項目集的項目集 X，利用 Cid List 找出目前 X 之超封閉項目集的 Cid，再利用 Sup Table 查詢這些超項目集的支持數是否等於 X。依照定義，若能找到 X 的超項目集與其支持數相等，表示 X 不是一個封閉項目集。反之，則 X 是一個封閉項目集。

3.2 探勘頻繁封閉項目集

CLMiner 首先掃描原始資料庫一次，找出每個項目在資料庫中的支持數。同時將資料庫轉換成 Tid List，紀錄每個項目所出現的交易編號。保留不小於最小支持數的項目，也就是頻繁項目。並且保留 Tid List 中頻繁項目的 Tidset。

依序兩兩聯集 Tid List 中的項目 I_i 與 I_j，$I_i < I_j$，假設聯集的結果為 X。接著交集 t(I_i)與 t(I_j) 得到 t($I_i \cup I_j$)，也就是 t(X)。若|t(X)|小於最小支持數，表示 X 不是頻繁封閉項目集。反之，則 X 可能為一個頻繁封閉項目集。交集 c(I_i)與 c(I_j)得到 c($I_i \cup I_j$)，也就是 c(X)。到 Sup Table 中查詢在 c(X)中的 Cid c 所對應的支持數，c∈c(X)。如果編號為 c 的封閉項目集其支持數與 X 的支持數相同，表示 X 存在一個超項目集

與其支持數相同，依據定義 2 可得知 X 不為封閉項目集。若編號為 c 的封閉項目集之支持數都不等於 X，則依據以下的四種情況分別進行處理。情況 1: 若 $t(I_i) = t(I_j)$，則將 I_j 從 Tid List 中刪除，接著將 I_i 與其它項目的組合置換成 X 與其它項目的組合。情況 2: 若 $t(I_i) \subset t(I_j)$，則將 I_i 與其它項目的組合置換成 X 與其它項目的組合。情況 3: 若 $t(I_i) \supset t(I_j)$，則將 I_j 從 Tid List 中刪除，並將 X 與 t(X)置入新的 Tid List 中。情況 4: 若 $t(I_i)$不等於 $t(I_j)$且互相沒有包含關係，則將 X 與 t(X)置入新的 Tid List 中。若新的 Tid List 不為空，則依照相同方式處理新 Tid List 中的項目集。若新的 Tid List 為空集合，則 X 為一個封閉項目集，我們給予它一個新的 cid 編號，並且更新 Cid List。假設 X 由 k 個項目所組成，若 X 之 cid 的值為 i，則分別將 i 加入到這 k 個項目的 Cidset 中。接著更新 Sup Table，將 i 加入到 Sup Table 中的 Cid 欄位，並設定其支持數為|t(X)|。重複此執行步驟，直到 Tid List 中的每個頻繁項目都已經處理完畢。

3.3 CLMiner 舉例說明

我們以表一為資料庫，假設最小支持數為 3，說明 CLMiner 的執行過程。首先，掃描原始資料庫一次，找出每個項目在資料庫中的支持數，同時將資料庫轉換成 Tid List，並保留不小於最小支持數的項目與相對應 Tidset。如表二所示，頻繁項目有{A}、{D}、{T}、{W}與{C}。接著依序處理 Tid List 中的每個頻繁項目，首先先處理項目{A}。聯集項目{A}與{D}，得到項目集{AD}，並交集 t({A}) = {1, 3, 4, 5}與 t({D})={2, 4, 5, 6}得到 t({AD}) = {4, 5}。由於|t(AD)| = 2 小於最小支持數，所以{AD}與其超項目集都不為頻繁封閉項目集。接著聯集{A}與{T}，得到{AT}，因為 t(A)與 t(T)既沒有包含關係也不相等，所以產生{AT}。接著交集 c(A)與 c(T)，得到空集合，表示目前並沒有任何超封閉項目集與{AT}的支持數相同。接著聯集{A}與{C}，因為 $t(\{A\}) \subset t(\{C\})$，得知每一筆包含{A}的交易都包含{C}，所以分別將{A}與{AT}取代成{AC}與{ATC}。接著聯集{AC}與{W}，因為 $t(\{AC\}) \subset t(\{W\})$，所以分別將{AC}與{ATC}取代成{ACW}與{ATCW}。此時找到一個頻繁封閉項目集也就是{ACW}，給予 Cid 編號為 1，並同時更新 Cid List，所以 c({A}) = {1}，c({W})={1}與 c({C})={1}。由於{ACW}的支持數為 4，所以新增 1 到 Sup Table 的 Cid 欄位，並且設定其支持數為 4。以相同的方式處理{ACW}底下的分支，也就是{ATCW}，並更新 Cid List 與 Sup Table。接著以相同的方式處理{D}與其它項目的組合，得到與{D}相關與{A}無關的頻繁封閉項目集，也就是{DC}與{DCW}。更新過後的 Cid List 與 Sup Table 分別如表三及表四所示。

接著處理{T}與其它項目的組合，以得到所有與{A}、{D}無關與{T}相關的頻繁封閉項目集。聯集{T}與{C}，得到{TC}。因為 $t(T) \subset t(C)$，得知每一筆包含{T}的交易都包含{C}，所以將{T}取代成{TC}，並交集 c(T) = {2}與 c(C) = {1, 2, 4}以得到{TC}的超項目集之編號，也就是{2}。由於編號為 2 的封閉項目集之支持數為 3，並不等於|t(TC)|，表示目前沒有超項目集與{TC}的支持數相同。接著聯集{TC}與{W}，得到{TCW}。因為 t({TC})既不等於 t({W}) 也互相沒有包含關係，所以交集 t({TC})與 t({W}) 得到 t({TCW}) = {1,3,5}，|t(TCW)| = 3。然後交集 c({TC})與 c({W}) 得到 c({TCW})，也就是{2}。因為 Sup Table 中編號為 2 的封閉項目集之支持數為 3，與|t({TCW})|相同，表示{TCW}存在超項目集與其支持數相同，所以{TCW}不為頻繁封閉項目集。依照相同的方式處理{TC}底下的分支，可得到與{T}相關與{A}，{D}無關的頻繁封閉項目集。

表五.最後的 Cid List

Item	Cidset
A	1,2
D	3,4
T	2,5
C	1,2,3,4,5,6,7
W	1,2,4,6

表六.最後的 Sup Table

Cid	SC
1	4
2	3
3	4
4	3
5	4
6	5
7	6

接著處理{W}與其它項目的組合。聯集項目{W}與{C}得到{WC}。因為 $t(\{C\}) \supset t(\{W\})$，代表每一筆包含{W}的交易都包含{C}，所以{W}與其它項目的組合就是{CW}與其它項目的組合。因此可以從 Tid List 中刪除 t({W})。依照相同的方式處理項目{C}，與其底下的分支，可得到所有與{C}有關但與{A}，{D}，{C}，{T}無關的頻繁封閉項目集。最後的 Cid List 與 Sup Table 分別如表五與表六所示。

3.4 CLMiner 虛擬碼

圖一與圖二分別為 CLMiner 的虛擬碼與副程式。在 CLMiner 的虛擬碼中(虛擬碼一)，我們以符號 TL 與 CL 分別代表 Tid List 與 Cid List，以 ST 代表 Sup Table，並以 MSC 代表最小支持數。CL.c(I_k) 表示 Cid List 中項目 I_k 的 cidset。ST[c]代表 Sup Table 中 Cid 編號為 c 的支持數。符號 C 代表一個頻繁封閉項目集，C.Cid 代表 C 的 Cid。i 代表一個新的 cid 編號。C_flag 為一個旗標，若值為 False，表示 X 不是封閉項目集。

```
01 Procedure CLMiner( TL, CL, ST, MSC)
02   for each Ii ∈ TL do
03     TLnew ← NULL
04     C_flag ← True
05     for each Ij ∈ TL with Ij< Ii  do
06       X ← Ii∪Ij
07       t(X) ← t(Ii)∩t(Ij)
08       if (|t(X)| ≥ MSC) then
09         c(X) ← c(Ii)∩c(Ij)
10         for each Cid c ∈ c(X) do
11           if ( ST[c] == |t(X)| ) then
12             C_flag ← False    //X is not closed
13
14         If (C_flag == True) then
15           Property (TL, TLnew)
16           If (TLnew ≠ NULL) then
17                CLMiner(TLnew, CL, ST, MSC)
18
19           delete TLnew
20           C ← X
21           C.Cid ← i
22           ST[i] ← |t(X)|
23           for each Ik ∈ CL do CL.c( Ik )∪i
24         end if
25       end if
26     end for
27   end for
28 end Procedure CLMiner
```

虛擬碼一. CLMiner 的虛擬碼

Property 副程式為 CLMiner 中的副程式(虛擬碼二)。CLMiner 會依照 Property 進行相對應的處理，找出頻繁封閉項目集 (虛擬碼二)。

```
01 Procedure Property (TL, TLnew)
02         if ( t(Ii) = t(Ij) ) then
03                         Remove t(Ij) from TL
04                         Replace all Ii with X
05     else if ( t(Ii)⊂t(Ij) ) then
06                         Replace all Ii with X
07     else if ( t(Ii)⊃t(Ij) ) then
08                         Remove t(Ij) from TL
09                         Add X and t(X) to TLnew
10     else if ( t(Ii)≠t(Ij) ) then
11                         Add X and t(X) to TLnew
12 end Procedure Property
```

虛擬碼二. 副程式 Property 的虛擬碼

4. DFI 演算法

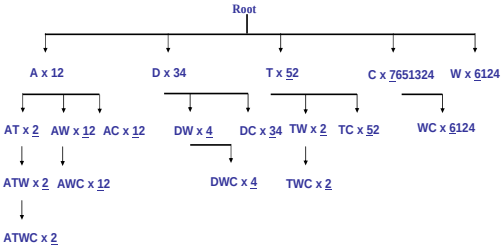
雖然之前有許多研究探討如何從資料庫中探勘頻繁封閉項目集，但是這些研究[4, 5, 6]較少提及如何從頻繁封閉項目集有效率地產生所有的頻繁項目集。所以在這篇論文中，我們又提出了一個 DFI 演算法，利用 CLMiner 的儲存結構產生所有的頻繁項目集。

DFI 演算法利用 Cid List 與 Sup Table 產生所有的頻繁項目集與其支持數。首先兩兩聯集 Cid List 中的項目 I_i 與 I_j，$I_i < I_j$，得到聯集的結果 X。然後交集 c(I_i)與 c(I_j)得到 c($I_i \cup I_j$)，也就是 c(X)。利用 Sup Table 查詢 c(X)中之支持數最高的 Cid，也就是 X 的頻繁封閉項目集。依據性質 1 與性質 2 得知，項目集 X 的支持數等於它的頻繁封閉項目集之支持數。接著將 c(X)置入新的 Cid List 中，依照相同的方式找出其它的頻繁項目集，直到新的 Cid List 為空集合為止。這個步驟將持續進行，直到 Cid List 中的每個項目都已經處理完畢。最後我們可以得到所有的頻繁項目集與其支持數。

4.1 DFI 舉例說明

以表五的 Cid List 與表六的 Sup Table 為例。首先聯集{A}與{D}得到{AD}，並交集 c(A) = {1,2} 與 c(D) = {3, 4}，得到 c({AD}) = {Ø}，表示{AD}並非頻繁項目集。接著聯集{A}與 {T}得到{AT}，並交集 c(A) 與 c({T})，得到 c({AT}) = {2}，查詢 Sup Table 中 Cid 為 2 的支持數，也就是 3，表示{AT}的頻繁封閉項目集之 cid 編號為 2，其支持數為 3 ，依據性質 1 與性質 2，我們可得知{AT}的支持數為 3。接著聯集{A}與{C}得到{AC}，並交集 c({A})與 c({C})，得到 c({AC}) = {1, 2}，查詢 Sup Table 中 Cid 為 1 與 2 的支持數，分別為 4 與 3。取支持數高者為{AC}的支持數，也就是 4。接著聯集{A}與{W}得到{AW}，並交集 c({A})與 c({W})，得到 c({AW}) = {1, 2}，同理，{AW}的支持數為 4。依照相同方式處理{A}的分支，也就是{AT}、{AW}與{AC}，可得到所有包含{A}的頻繁項目集與支持數。依照相同方式依序處理頻繁項目{D}、{T}、{C}與 {W}，可得到表一中所有的頻繁項目集。圖一列出了每個

頻繁項目集與其相對應的 Cidset，項目集旁的數字代表其 Cidset。



圖一: 頻繁項目集與其 Cidset

4.2 DFI 虛擬碼

虛擬碼三為 DFI 演算法的虛擬碼。其中 CL 與 ST 分別代表 Cid List 與 Sup Table。MAX 代表編號為 c 的頻繁封閉項目集中，最高的支持數，c∈c(X)。符號 SC(X)，表示 X 的支持數。ST[c]表示 Sup Table 中 Cid 為 c 的 SC 值，也就是支持數。

```
01 Procedure DFI (CL, ST)
02   MAX ← 0
03   for each I_i ∈ TL do
04     for each I_j ∈ TL with I_j < I_i  do
05       X ← I_i ∪ I_j
06       c(X) ← c(I_i) ∩ c(I_j)
07       If ( c(X) ≠ NULL ) then
08         for each Cid c∈ c(X) do
09           If( ST[c] > Max ) then Max ← ST[c]
10           SC(X) ← Max //X is a frequent itemset
11           Add X and c(X) to CL_new
12       end if
13     end for
14     DFI(CL_new, ST) ; delete CL_new
15   end for
16 end Procedure DFI
```

虛擬碼三. DFI 的虛擬碼

5. 結論

在這篇論文中，我們提出一個從資料庫中有效率探勘頻繁封閉項目集的演算法 CLMiner。並且提出一個從頻繁封閉項目集有效率產生所有頻繁項目集的演算法 DFI。DFI 演算法採取深度優先的方式，利用 CLMiner 的儲存結構，Cid List 與 Sup Table 產生出所有的頻繁項目集與其支持數。這兩個演算法的執行效能不僅優於先前的方法，透過 CLMiner 與 DFI 演算法探勘頻繁項目集的執行速度比直接探勘頻繁項目集的演算法[4, 6]還要快。

參考文獻

[1] R. Agrawal and R. Srikant, Fast Algorithm for Mining Association Rules. In *Proc. of Int. Conf. on Very Large Data Bases*, pages 487-499, Santiago, Chile, September 1994.

[2] J. Han, J. Pei, and Y. Yin, Mining Frequent Patterns without Candidate Generation. In *Proc. of 2000 ACM-SIGMOD Int. Conf. on Management of Data*, pages 1-12, May 2000,.

[3] Liping Ji, Kian-Lee Tan, Tung, A.K.H., Compressed Hierarchical Mining of Frequent Closed Patterns from Dense Data Sets. In *IEEE Transactions on Knowledge and Data Engineering*, pages 1175-1187, September 2007.

[4] N. Pasquier, Y. Bastide, R. Taouil and L. Lakhal, Discovering Frequent Closed Itemsets for Association Rules. In *Proc. of 7th Int. Conf. on Database Theory*, pages 398-416, January 1999.

[5] J. Wang, J. Han, and J. Pei, CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets. In *Proc. of 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 236-245, 2003.

[6] M.J. Zaki and C.J. Hsiao, Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure, *Proc. of the IEEE Transactions on Knowledge and Data Engineering (TKDE)*, Volume 17, Issue 4, April, 2005, pages 462 – 478.

[7] UCI Machine Learning Repository, http://archive.ics.uci.edu/ml