以 CIT 結構為基礎之漸進式關聯規則探勘方法

張簡尚偉
中興大學電子商務研究所
swc8@nchu.edu.tw

廖虹雲
朝陽科技大學資訊管理系
s9114602@mail.cyut.edu.tw

摘要

如何有效率地從大量的資料庫中探勘出隱含的關聯規則，在資料探勘研究領域中已成為一個重要的議題。隨著交易的持續進行，資料庫中的資料亦會隨著時間而變動，對於每一個環節的變動，這些先前被探勘出的關聯規則，在更新後的資料庫中可能已經不符條件，亦可能會有新的規則產生。因此，為了能夠有效並即時獲取正確而完整的關聯規則，本研究提出 ICIT(Incremental CIT)漸進式關聯規則探勘方法，只需對原始資料庫進行一次完整掃描，為每個屬性項目建立相對應的等值類別(Equivalence Classes)，並以具有繼承觀念的 CIT(Class Inheritance Tree)結構來儲存前次探勘時資料庫中的高頻項目集與相關資訊，即可進行相對應的關聯規則探勘。當資料庫發生增加或刪除交易紀錄等異動時，亦只需動態掃瞄異動的資料庫部分，隨之調整 CIT 結構，便能從調整後的 CIT 中找出更新後資料庫中的關聯規則，不需再重複掃瞄整個資料庫。

關鍵詞：漸進式探勘、資料探勘、關聯規則

一、研究背景及動機

隨著網路與資訊科技不斷的提昇與進步，大量的資料庫因應而生，而過去資料庫著重在蒐集、儲存與有效管理資料，但由於龐大資料中潛藏了無數珍貴的資訊與知識，因此近年來「資料探勘」技術蓬勃發展，在各領域的應用也相當廣泛。然而在瞬息萬變的環境中，要對資料庫與資料倉儲中數量龐大的資料進行完整的探勘，面臨了兩大挑戰：需耗費相當的時間與成本。若能適當地儲存計算結果，並進一步地隨著資料之變動，迅速修正之前所取得的規則與知識，就不需再重新挖掘整個資料庫而能維持資訊的價值和正確性，而漸進式資料探勘(Incremental data mining)的目的便是在對動態資料庫(Dynamic Database)作關聯規則挖掘時，可以讓使用者在資料不斷更動的情況之下，也能即時獲取與資料庫相符合的規則。

關聯規則探勘主要是希望能找出資料庫中項目或屬性間的共同關聯性，如在交易資料庫中找出某些產品具有共同被購買的關係。舉例來說，在超市中有“80%購買奶粉的人，也會同時購買尿布”，藉由這些資訊的取得，也許就能考慮重新安排商品的陳列方式，將顧客常買的兩項商品放在鄰近的位置，或是將奶粉和尿布一併進行促銷，以刺激買氣。所以利用關聯規則挖掘的目的，便是希望能從資料相對的發生次數之分析上著手，找出其中項目間的隱含關係，作為決策參考之依據。在關聯規則的基本定義中，每筆交易(Transaction)皆由項目(Item)所集合而成，各項目則為屬性配合相對應的值所構成，當多個項目彼此有相關性時，則結合成項目集(Itemset)。當一個集合中若含有k個項目集時，稱為k-itemsets。藉由讀取資料庫中的交易紀錄，可找出出現次數頻繁的高頻項目集(Large Itemsets)，再利用高頻項目集來產生關聯規則。其中用來判定是否為高頻項目集的門檻值稱為最小支持度(Minimum Support)，未確認滿足最小支持度的項目集稱為候選項目集(Candidate Itemsets)，而判定能否由高頻項目集形成規則的門檻值稱為最小信賴度(Minimum Confidence)，通常以百分比來呈現。使用者可以藉由調整這兩種過濾規則的門檻值，來找出最合適的規則。

但是對動態的資料庫系統而言，利用傳統的關聯規則挖掘是件相當耗費時間的工作，主要的原因有二點：

1. 必須多次掃描資料庫：在傳統資料探勘的過程中，必須多次讀取資料庫中的資料以獲得規則，這對資料量龐大的資料庫而言相當費時，而且大量的磁碟讀取動作亦會造成資料庫系統的效能不彰。
2. 在資料庫更動時必須重新作探勘：對資料庫而言，資料的異動是相當平常而且頻繁的事，但在資訊探勘完成後，若資料有所變更，則先前所得到的規則就與資料庫中的資料產生不一致。為了得到與資料庫內容相符的規則，不論資料庫更動之程度是大是小，使用者都必須再重新進行探勘，否則無法在資料庫更動時即時獲得最新之

規則。

而漸進式資料探勘可以縮短整個資料探勘時間，花最少的時間進行探勘，得到相同的探勘效果。以持續動態改變的交易資料庫為例，資料內容包括新增資料和刪除舊有資料。本研究針對交易資料庫，提出一個以類別繼承樹狀結構(Class Inheritance Tree, CIT)[3]為基礎的關聯規則漸進式探勘法 ICIT(Incremental CIT)，其目標旨在處理資料庫發生增加或刪除交易資料等異動時，只需掃瞄異動的資料庫部分，隨之調整 CIT 結構，使它能符合更新後的資料庫中之交易資料內容。因此，透過調整後的 CIT 結構便能找到更新資料庫後的高頻項目集，進而探勘出關聯規則，而不需要再重新掃瞄整個更新後的資料庫。

本研究將整個結構分為五大部分。第二節為文獻探討，文獻探討的前半部分為現有的漸進式探勘方法做個概括性的介紹與分析，後半段將介紹 Changchien 及 Lu[3]提出的類別繼承樹狀結構為基礎的關聯規則資料探勘演算法。第三節說明本研究所提出的 ICIT 漸進式關聯規則探勘方法，並在第四節以一個實例說明 ICIT 對資料庫異動的調整。最後在結論的部分，總結此方法的貢獻與未來研究上可供改善的方向與建議。

二、文獻探討

關聯規則探勘研究可分為下列兩個子問題：(1) 找出高頻項目集：符合最小支持度，及(2) 產生資料庫的關聯規則：符合最小信賴度。雖然關聯規則挖掘可以自動從資料庫中找出有用的資訊，但是對資料庫系統而言，利用傳統Apriori 演算法作關聯規則挖掘是一件相當耗費時間的工作。因為它需要重複掃描資料庫，花費過多的I/O時間於讀取資料庫，而且需要產生大量的候選項目集。而隨著資料量增長，近年已經有許多尋找高頻項目集的漸進式探勘方法被提出。其中，在1996年D. W. Cheung所發表之FUP(Fast Update)演算法[4]最早提出針對動態資料庫進行漸進式探勘的概念。它以Apriori[2]為基礎而建構，藉由儲存舊有關聯規則與相關資訊，以減少在資料庫異動後需掃瞄原始資料庫的次數，可達成快速更新關聯規則之目標，但是缺點在於只考慮到資料庫新增的情況。接著，在1997年已更進一步改良的FUP2演算法[5]中，能同時解決資料庫新增、刪除與修改之情況，但因演算法本身在資料量龐大時仍需掃瞄資料庫多次，以至於效率依舊無法提昇。DUP演算法[7]利用之前的DLG演算法建立關聯圖(association graph)產生高頻項目集，可降低候選項目集產生之數量，減少磁碟的讀取，但其建立之Bit-Map會隨著資料庫中項目的增加而快速成長，需要大量的儲存空間。論文[11]中提出的MMS_UP演算法亦類似FUP演算法，但其目的是要解決不同項目用不同的最小支持度門檻值來進行關聯規則探勘。論文[10]所提出之方法概念為：當一個項目集本身並非高頻項目集，但其子項目集合每個皆為高頻項目集時，就可能會較其他非高頻項目集容易成為高頻項目集，此時稱該項目集為一個Negative Border。因為保留原始資料庫中所有的高頻項目集和Negative Border，以及其支持度計數值等相關資訊，所以除了掃瞄異動的資料庫外，最多只需掃瞄原始資料庫一次。王慶堯[1]定義了一個“準大項目集”(Pre-large)的觀念，必須另外訂定一個低於最小支持度的門檻值，以保留高頻項目集與準大項目集，目的在於為了減少FUP演算法在處理高頻項目與非高頻項目間轉換的問題。在2001年Change Hung Lee等人提出的SWF演算法[8]與FUP2一樣皆採用類似Apriori-like的方法，其精神在於產生長度為2之候選項目集的數量會逼近長度為2的高頻項目集數量，再加上配合DHP演算法[9]中的scan reduction技術，大量減少候選項目集之數量與產生時間，也縮短掃瞄原始資料庫的時間，但在產生高頻項目集時仍需掃瞄原始資料庫。上述方法皆保留先前的探勘結果，以空間換取時間的策略來達到提升探勘效率之目的，且最多只需掃瞄資料庫兩次。

而本研究根據改良自 rough set [6]之基本概念，以具有繼承觀念 CIT 結構為基礎的關聯規則探勘方法，提出漸進式關聯規則探勘方法以提升探勘之效能，並降低儲存空間以建立關聯規則。該方法主要分五個基本步驟[3]，分別為：

(1) 產生條件等值類別(cause equivalence classes, CEC)與決定等值類別(decision equivalence classes, DEC)

假設選擇資料庫的欄位屬性A做為條件屬性(condition attribute)，選擇屬性X做為決定屬性(decision attribute)。在欄位屬性A中，當紀錄編號(TID)第2、3、5筆紀錄的值皆等於2時，就可以用CEC_{A2}={2, 3, 5}來表示，即為一個條件等值類別；而決定等值類別也是相同的概念。

(2) 建立類別繼承樹(CIT)

將上述所得的等值類別集，依建構節點的規則建立CIT結構，以樹狀結構來表現它們之間的關聯性，做為找尋關聯規則的依據。

(3) 找出下限近似值規則(lower approximation rules)

即一組可以表示條件等值類別完全被包

含於決定等值類別之關聯規則就稱為下限近似規則，此類的規則皆具有滿足最小支援度及100%的信賴度(confidence, CF)。

(4) 找出上限近似值規則(upper approximation rules)並計算信賴度

即一組可以表示條件等值類別完全被包含於決定等值類別的關聯規則就稱為下限近似規則，此類的規則皆滿足最小支援度及信賴度。其信賴度的計算公式如下：

$$CF = \frac{Num(X \cap I(x))}{Num(I(x))} \qquad \textbf{(1)}$$

其中，$CF \in [0,1]$，$I(x)$表示條件等值類別，X表示決定等值類別。

(5) 產生多維屬性關聯規則(multiple attributes association rules)並計算信賴度

從上述的步驟中所找到的都是一維屬性關聯規則(single attributes association rules)，必須滿足使用者定義的最小支援度才可建立。從所有的高頻項目集中可再進一步找尋可組合連結的部分，找出多維屬性關聯規則，並計算這些規則的信賴度。

而 CIT 所定義的節點結構如圖 1 所示，以下 ICIT 所採用之範例說明將以圖 2 的簡略圖示加以描述。

<table>
<tr><td colspan="2">Node[]</td><td colspan="2">Node_object[]</td></tr>
<tr><td>Parent*</td><td>Child*</td><td>Brother*</td><td>Contain[]</td></tr>
<tr><td colspan="4">Relative[]</td></tr>
</table>

圖 1 CIT之節點結構

<table>
<tr><td>N_{id} = { EC }</td></tr>
<tr><td>Att_i</td></tr>
</table>

圖 2 ICIT之節點表示

其中，Node[]記錄每個節點之 ID，即 N_{id}, id = 1,…n；Node_object[] 為所有類別中的特徵值(object)，即等值類別集 ECs。而 Parent* 做為指向該節點的父節點的指標、Child* 做為指向該節點第一個子節點的指標、Brother* 則做為指向該節點第一個兄弟節點的指標。Contain[]中記錄含有此節點之所有項目屬性值(attribute)，以 Att_i 表示，並以 Relative[]記錄此節點中所包含的部分共同項目。

在 CIT 的結構中，所有節點儲存的內容都是符合最小支持度門檻值的高頻項目集，但是當資料庫的資料有更動時，將導致某些節點不再符合門檻值，或是需要再加入新的節點，為了避免必須因資料庫異動而重新建構新的 CIT 結構，因此，我們提出針對 CIT 結構進行調整與維護的 ICIT 演算法。

三、以 CIT 結構為基礎之漸進式關聯規則探勘方法 - ICIT

本節將介紹本研究所提出的 ICIT(Incremental CIT)，以CIT結構為基礎之漸進式關聯規則探勘法，說明如何調整CIT結構以達到高頻項目集的漸進式探勘。此演算法可適用於資料庫的新增、刪除，以及同時有交易資料庫新增與刪除等異動時的漸進式探勘方法。

(一) 基本概念

本研究之方法保留原始CIT結構，根據其節點定義與演算法可知，其從根節點到葉節點間的等值類別節點乃是建構在繼承的觀念上，從資料庫交易紀錄中之產品項目所得的所有等值類別集，依照建構CIT所定義之規則建構成一棵類別繼承樹，透過樹狀結構呈現它們的繼承及關聯性，以做為尋找關聯規則之依據。然而，當原有資料庫DB中的交易資料有新增或是刪除的變動時，仍需要再次掃瞄資料庫以重新建立CIT結構，使用者也需要等待一段時間才能得到新的規則。因此，本研究針對CIT的節點結構，提出ICIT的演算法，可在資料庫發生新增、刪除與異動交易時，不需再重新掃瞄整個資料庫即可有效率地動態調整節點之順序，使它能隨時保存最新的資訊，而不用浪費過多的I/O時間來重新建構CIT。

本研究方法要求必須在前一次探勘原始資料庫後儲存以下資訊：

1. 未符合最小支持度門檻值的等值類別資訊
2. 依照原始資料庫所建構而成的CIT結構

需要保留第一項資訊之目的，主要是為了在資料庫產生異動後，不需再回去掃瞄原始資料庫便能快速計算出更新後的資料庫中，有那些是符合最小支持度的常見項目集，用以判斷是否需調整先前所建的 CIT 結構。

(二) ICIT結構調整維護方法

為隨著資料庫的變動而達到局部調整CIT的目的，CIT結構調整方法包含以下三大步驟：

Step1. 產生條件、決定等值類別與備選等值類別(Potential Equivalence Classes, PEC)

先依序掃瞄資料庫以產生條件與決定等值類別，並將未符合最小支持度門檻值(即等

值類別中特徵值之數量必須大於或等於資料庫所有紀錄筆數乘上使用者設定的最小支援度)的等值類別存入備選等值類別集中，直到因為資料庫產生異動而使備選等值類別集滿足最低門檻值時才可被建立成節點存入CIT結構。

Step2. 建立CIT結構並調整節點順序

將第一步驟產生符合最小支持度之條件與決定等值類別建構成CIT節點，以樹狀結構呈現資料庫之屬性值間的關聯性。以下參考CIT的定義與演算法所制訂之節點建立規則以作為建構CIT結構之依據：

■ Rule 1 插入現存節點

一個等待插入至CIT結構中的等值類別稱為等候類別(Waiting class, Wclass)；一個已經存在於CIT中的節點稱為現存節點(Existing node, Enode)。當等候類別與現存節點相符時，即可直接置入該節點而不需另外建立新節點。

■ Rule 2 建立新的等候節點

若找不到任何符合等候類別的現存節點，就需建立一個包含此等候類別的新節點，稱為等候節點(Waiting node, Wnode)。

■ Rule 3 建立新的父節點(parent node)

若有個Wnode的特徵值完全包含於Enode，則需將Wnode節點置入CIT結構，使之成為Enode之父節點。例如有一個Wnode為{2, 4, 5}，而CIT結構中現存節點$N_{3=}${2, 4, 5}，則將此Wnode往上提升為N_3之父節點。

■ Rule 4 連結等候節點做為父節點的子節點(child node)

若有個Enode的特徵值完全包含於Wnode，則將Wnode節點連結至Enode，使之成為Enode之子節點。例如有一個Wnode之值為{2, 3, 5, 6}，而CIT結構中有現存節點$N_{4=}${2, 3, 5}，則將此Wnode建立在N_4之下連結成為其子節點。

■ Rule 5 連結等候節點為現存節點的兄弟節點(brother node)

若Wnode未能完全包含於任一Enode之中，但是它們皆擁有共同的起始特徵值(object)，如此便可將Wnode連結為Enode之兄弟節點，如N_1={2, 3, 5}與N_3={2, 4, 5}都有特徵值"2"為共同的起始特徵值。

■ Rule 6 插入關係節點(Relative node, Rnode)

當Wnode擁有比其父節點還多的特徵值，就需要另外再連結到關係節點。若該關係節點已存在，則可直接在Wnode與關係節點間建立關係連結；反之則需再建立新的關係節點並連結至Wnode，以利於探勘上限近似值規則。舉例來說，目前已有現存節點N_7={2, 4}，當輸入一個新的Wnode值為{2, 4, 5, 6}時，除了將此節點列為N_7的子節點外，另外需要再連結到N_{10}={5}和N_{11}={6}的關係節點；假使在現存的CIT結構並未存在節點N_{11}={6}，則必須另外建立新節點後再設定連結。

Step3. 處理異動資料於CIT結構之調整與維護

CIT結構建構完成後，當資料庫發生變動時，就必須利用ICIT所定義的維護規則對CIT結構中節點進行調整與維護。異動資料產生時，資料庫可能同時需要進行紀錄的新增與刪除動作，透過ICIT方法來對CIT結構進行調整即可同時處理新增與刪除的異動，維護方法包含下面所述三個步驟：

(1) 產生條件等值類別、決定等值類別

先掃瞄異動資料庫(db^+、db^-)，依序產生條件等值類別與決定等值類別。此時，若備選等值類別在資料庫異動後達到滿足門檻值條件者，則取出該備選等值類別合併新的CEC成為等候類別。

(2) 將異動資料加入CIT節點或從CIT節點中扣除，並儲存備選等值類別

將屬性特徵值與等候類別相符的現存節點取出，在現有的特徵值之後加入等候類別中新增加的特徵值，接著檢查現存節點中是否含有必須刪除的特徵值，予以扣除。若更新後的現存節點變成無法滿足最小支援度門檻值，則直接移置備選等值類別集中。下面為產生異動資料時可能發生的情況與處理方式：

■ Case 1 $If\ n_i \notin N_i\ and\ \{EC'\} - \{EC\} = \varphi$

假設以n_i與N_i表示目前等候類別與所有現存節點的項目屬性Att_i；$\{EC'\}$和$\{EC\}$則分別代表等候類別與現存節點之等值類別中的特徵值。若等候類別中之項目屬性不屬於任何一個現存節點，而且等候類別中的特徵值也找不到與現存節點相符者，則視為必須產生新增之等候節點，並連結關係節點。

■ Case 2 $If\ n_i \notin N_i\ and\ \{EC'\} = \{EC\}$

若等候類別中之項目屬性不屬於任何一

個現存節點，但與某個現存節點有完全相同的特徵值；或是發生異動後，與某個現存節點具備完全相同的特徵值，此時必須取出相符的現存節點，直接將等候類別中的項目屬性加到現存節點中，修改項目屬性為 $N_i = N_i \cup n_i$ 。

■ Case 3 $If\ n_i \in N_i\ and\ \{EC'\} - \{EC\} \neq \varphi$

假如等候類別的項目屬性與某個現存節點相同，即表示該項目屬性之等值類別已存在於原先的CIT結構中，但是其特徵值不同，也就是在等候類別中存有欲新增的特徵值，所以必須取出此現存節點，修改等值類別中的特徵值為 $\{EC\} = \{EC\} \cup \{EC'\}$ ，更新現存節點中的特徵值。在此可能會發生一個例外狀況，若更新後的現存節點與其他的現存節點擁有的等值類別完全相同者，則合併成一個現存節點。

■ Case 4

$If\ n_i \in N_i\ and\ n_i \neq N_i\ and\ \{EC'\} - \{EC\} \neq \varphi$

等候類別的項目屬性雖然屬於某個現存節點，而且在等候類別中存有欲新增的特徵值，但現存節點中含有其他的項目屬性，如 n_i, n_j ，此時必須從現存節點中移除 n_i ，產生一個等候節點存放等候類別中的項目屬性及特徵值，其特徵值為 $\{EC\} = \{EC\} \cup \{EC'\}$ ，項目屬性為 n_i 。

(3) 調整CIT結構中節點的順序

最後，再將更新後的現存節點放回 CIT 結構中。視第一個特徵值做為父節點，將現存節點連結至該節點下成為子節點，依建立節點與維護規則調整節點的順序，即可產生擁有更新後資料庫的 CIT 結構。

四、範例說明

在本節中，我們將以一個範例來說明本研究所提出的ICIT探勘方法。這裡所舉的範例同[3]，假設原始資料庫DB中有十筆紀錄，如表 1所示，共有七個欄位屬性，假設條件屬性為欄位A到欄位D，而決定屬性為欄位X、Y及Z，關聯規則探勘之最小支持度設定為20%。

Step1. 產生條件、決定等值類別與備選等值類別

由於最小支持度設定為20%，因此門檻值為資料庫總筆數10乘上20%等於2，即表示等值類別中的特徵值需達到2個以上才可成為條件等值類別置入CIT結構中，以確保CIT結構中存放的皆為高頻項目集。

表 1 原始資料庫

TID	A	B	C	D	X	Y	Z
1	1	2	4	3	2	3	3
2	3	1	2	3	2	2	3
3	4	2	3	2	1	1	1
4	2	3	1	4	2	3	2
5	4	4	3	2	3	2	1
6	4	3	2	3	1	3	1
7	3	1	4	3	3	2	3
8	2	4	1	4	3	4	2
9	2	3	1	1	4	1	2
10	1	2	2	4	2	1	3

(1) 產生條件等值類別(CEC)

CEC_{A1}= {1,10}, CEC_{A2}= {4, 8, 9},
CEC_{A3}= {2, 7}, CEC_{A4}= {3, 5, 6},
CEC_{B1}= {2, 7}, CEC_{B2}= {1, 3, 10},
CEC_{B3}= {4, 6, 9}, CEC_{B4}= {5, 8},
CEC_{C1}= {4, 8, 9}, CEC_{C2}= {2, 6, 10},
CEC_{C3}= {3, 5}, CEC_{C4}= {1, 7}, CEC_{D2}= {3, 5},
CEC_{D3}= {1, 2, 6, 7}, CEC_{D4}= {4, 8, 10}

(2) 產生決定等值類別(DEC)

DEC_{X1}= {3, 6}, DEC_{X2}= {1, 2, 4, 10},
DEC_{X3}= {5, 7, 8}, DEC_{Y1}= {3, 9, 10},
DEC_{Y2}= {2, 5, 7}, DEC_{Y3}= {1, 4, 6},
DEC_{Z1}= {3, 5, 6}, DEC_{Z3}= {4, 8, 9},
DEC_{Z4}= {1, 2, 7, 10}

(3) 產生備選等值類別

DEC_{D1}= {9}, DEC_{X4}= {9}, DEC_{Y4}= {8},
DEC_{Z2}= {3}

上述四個等值類別皆僅有一個特徵值，並未達到最小支持度門檻值2，因此被移置到PEC中，留待將來因發生資料異動而達到滿足門檻值之條件時，可再取出與條件等值類別合併成為等候類別，不需再花費I/O資源掃瞄原始資料庫，即可快速處理動態的資料庫變動。

Step2. 建立漸進式CIT結構並調整節點順序

利用上一步驟產生符合最小支持度之條件與決定等值類別建構CIT結構，如圖 3。

Step3. 處理異動資料於CIT結構之調整與維護

假設經過一段時間之後，因資料具時效性之因素而產生部分的資料異動，資料庫需要刪除前三筆紀錄TID = {1, 2, 3}，並新增五筆資料紀錄，如表 2所示。

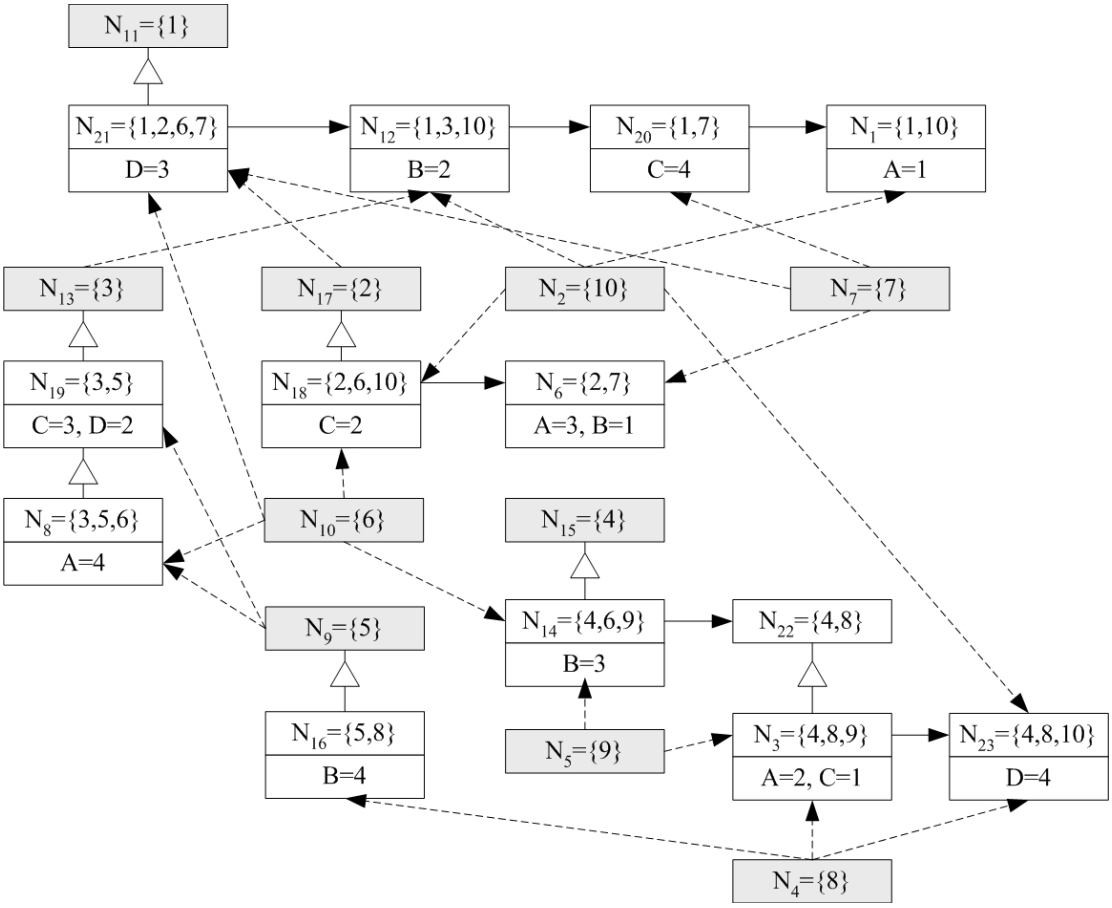


圖 3 原始資料庫之類別繼承樹

表 2 新增資料紀錄(db^+)

TID	A	B	C	D	X	Y	Z
11	3	4	4	3	2	4	1
12	2	4	1	1	3	2	4
13	1	2	4	2	4	1	2
14	2	1	3	1	2	3	3
15	4	3	2	3	1	2	4

(1) 產生異動資料之等值類別

a. 產生條件等值類別

CEC_{A1}= {13}, CEC_{A2}= {12, 14}, CEC_{A3}= {11}, CEC_{A4}= {15}, CEC_{B1}= {14}, CEC_{B2}= {13}, CEC_{B3}= {15}, CEC_{B4}= {11, 12}, CEC_{C1}= {12}, CEC_{C2}= {15}, CEC_{C3}= {14}, CEC_{C4}= {11, 13}, CEC_{D1}= {9, 12,14}, CEC_{D2}= {13}, CEC_{D3}= {11,15}

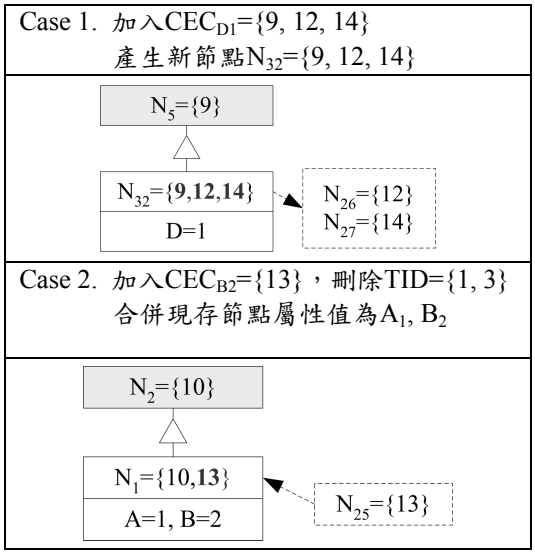
b. 產生決定等值類別

DEC_{X1}= {15}, DEC_{X2}= {11, 14}, DEC_{X3}= {12}, DEC_{X4}= {9, 13}, DEC_{Y1}= {13}, DEC_{Y2}= {12, 15}, DEC_{Y3}= {14}, DEC_{Y4}= {8, 11}, DEC_{Z1}= {11}, DEC_{Z2}= {3, 13}, DEC_{Z3}= {14}, DEC_{Z4}= {12, 15}

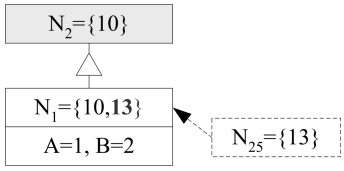
(2) 將異動資料加入CIT節點或從CIT節點中扣除，並儲存備選等值類別

透過ICIT探勘方法定義之規則，可同時處理異動資料的新增與刪除。以下列舉四個例子來說明ICIT對資料庫之異動資料的處理方式，如表 3所示。

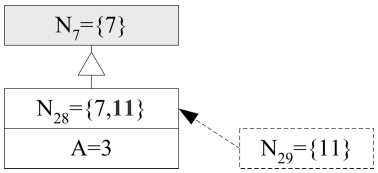
表 3 處理異動資料的四種情況



Case 3. 加入CEC_{A1}={13}，刪除TID={1}



Case 4. 加入CEC_{A3}={11}，刪除TID={2}
產生新節點N_{28}={7, 11}



最後，經過 ICIT 而調整的結構如下圖 4，呈現依更新後資料庫而建的漸進式類別繼承樹，其關係節點之連結部分以虛線表示。

五、結論與未來研究方向

本研究提出一個高頻項目集的漸進式探勘方法 ICIT，此方法不需要重複掃瞄原始資料庫，即可找出異動後的資料庫中正確且完整的高頻項目集。本研究方法採用 CIT 結構來保留原資料庫中的高頻項目集，並另外儲存備選等值類別，即小於最小支持度門檻值的等值類別，以方便在資料庫產生變動時，快速調整、維護 CIT 節點，不需再重建整個 CIT 結構，可加速關聯規則的產生。利用具有繼承觀念的 CIT 結構除了可以探勘資料庫欄位間的關聯關係，亦可進行購物籃分析的關聯探勘。

由於利用 CIT 結構與記載的資訊內容有別於其他的樹狀結構，因此所需花費的儲存空間可能相對地多於其他的樹狀結構，如 FP-tree。對於 ICIT 相較其他漸進式探勘演算法尋找關聯規則上，是否有明顯的效率提昇？或需進一步改良？或是能否善加利用所紀錄的資訊應用於解決其他問題？在未來都值得進一步加以探討。

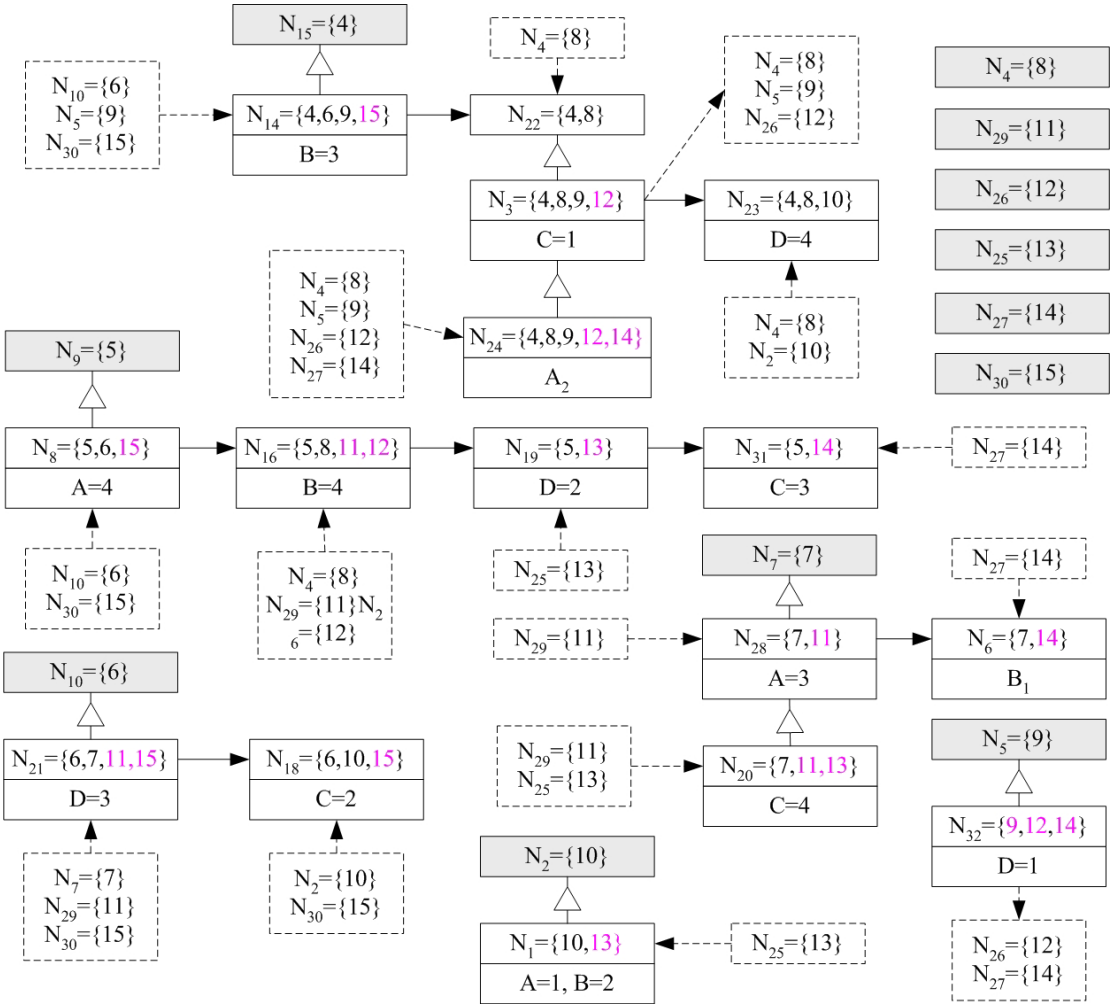


圖 4 更新後資料庫之類別繼承樹

六、參考文獻

[1] 王慶堯, 洪宗貝, "利用準大項目集之漸進式資料挖掘," *義守大學資訊工程所碩士論文*, 2000.

[2] R. Agrawal and R. Srikant, "Fast Algorithm for Mining Assoication Rule in Large Databases," in *Proc. of the 20th International Conference on Very Large DataBases*, pp. 487-499, Santiago, Chile, September 1994.

[3] S. W. Changchien and T. C. Lu, "A New Efficient Association Rules Mining Method Using Class Inheritance Tree (CIT), " in *Proc. of the 12th International Conference on Information Management*, 2001.

[4] D.W. Cheung, J. Han, V.T. Ng, and C.Y. Wong, "Maintenance of Discovered Association Rules in Large Databases: An Incremental Update Technique," in *Proc. of the 12th International Conference on Data Engineering*, pp. 106-114, February 1996.

[5] D.W. Cheung, S.D. Lee, and Benjamin Kao, "A General Incremental Technique for Maintaining Discovered Association Rules," in *Proc. of the 5th International Conference on Database Systems for Advanced Applications*, pp. 185-194, April 1-4 1997.

[6] M. Delgado, D. Sanchez, M. J. Martin-Bautista and M.A. Vila, "Mining association rules with improved semantics in medical databases," *Artificial Intelligence in Medicine*, vol.21, pp. 241-245, 2001.

[7] G. Lee, K.L. Lee, and A.L.P. Chen, "Efficient Graph-Based Algorithms for Discovering and Maintaining Association Rules in Large Databases," *Knowledge and Information Systems*, Springer-Verlag, Vol. 3, pp. 338-355, 2001.

[8] H. Lee, C.-R. Lin, and M.-S. Chen, "Sliding-Win Algorithm for Incremental Mining," in *Proc. of the ACM 10th International Conference on Information and Knowledge Management*, November 5-10, 2001.

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

[10] S. Thomas, S. Bodagala, K. Alsabti, and S. Ranka, "An Efficient Algorithm for the Incremental Updation of Association Rules in Large Databases," in *Proc. of 3rd International conference on Knowledge Discovery and Data Mining*, New Port Beach, California, pp. 263-266, August 1997.

[11] M.C. Tseng, W.Y. Lin, and B.C. Chien, "Maintenance of Generalized Association Rules with Multiple Minimum Supports," *Joint 9th IFSA World Congress and 20th NAFIPS International Conference*, vol.3, pp. 1294-1299, 2001.