第一章 緒論

在目前電腦日漸普及的現代生活中，許多人工作業記錄的資料已逐漸的被電腦所取代，如商業行為中的客戶基本資料、交易資料；醫學中的病例資料、藥品資料；網路上使用者的瀏覽資料，客戶基本資料…等，諸如此類的各種資料已逐漸由人工撰寫於紙張的作業方式，轉換為電腦中的資料，再進一步的由文字資料轉換為資料庫中各個不同欄位的資料。加上排序、索引、不同表格的關聯，這樣的資料在新增與維護上，比起早期的人工作業方式既快速又正確，大大的增進了使用者對於資料使用的便利與效率。

但是隨著科技的進步，人們生活的方式越來越多元化，資料的儲存大量的增加，儲存在電腦的資料有什麼作用？到底什麼是有用的資料？什麼是使用者感興趣的資料？如何將有用的資訊從一堆看來雜亂無章的資料中擷取出來？如何利用擷取出來的資料？類似此類的問題就是資料挖掘(Data Mining)[1]的探討範圍了。

第一節 資料挖掘的定義

資料挖掘簡單來說，就是依照使用者設定的參數，在一群未經處理的資料中找到使用者感興趣的資訊，經過某些特殊的處理後，作為使用者決策判斷的參考依據。一般人認為，資料挖掘僅為將資料匯入資料挖掘的系統，以某一特定的演算法或是程式執行之後，取出使用者所需的資訊，即為資料挖掘。然而真正的資料挖掘必須分為三個主要步驟，分別為前置階段、資料挖掘階段、規則詮釋階段，而依照此三個階段又可以將資料挖掘的過程又分為以下數項：

1. 資料的取得與淨化階段

在進行資料挖掘前，使用者必須確認本身需要對何種領域(domain)的資料進行資料挖掘的動作，例如零售業在資料挖掘之前，必須確認挖掘的是客戶資料或是交易資料；醫院在對其內部資料庫進行挖掘時，也必需確認所要挖掘的資料是病人的病歷資料，或是各種疾病之間的相關性資料等。確認資料挖掘時所要挖掘的資料來源後，才能有效的將所需要的資訊挖掘出來。

在取得資料的來源後，亦必須確認資料本身的資料型態以及資料是否為有效的資料。例如日期型態的資料欄位內存放的是日期格式，年齡型態的欄位中存放的是數字，且數字的範圍是在年齡中是有效合理的範圍。此一過程即為資料的淨化階段。

一般來說，以上的兩個階段是資料挖掘過程中的前置工作，將資料取得與淨化之後，就可以交由電腦進行資料挖掘動作，但上述的兩個階段所需要花費的人力與時間卻是整個資料挖掘過程中最多的部分。

2. 資料倉儲階段

將資料取得與淨化後，必須選擇一個良好的資料庫系統，作為資料存放的位置。在此必須要考量的重點是企業成本、資料庫大小及效能、以及資料庫相關應用工具的種類與實用性。對一般大型企業來說，可以選擇大型的資料庫工具搭配中型的資料挖掘的應用程式，建構一個完整的資料挖掘的環境。而一般中小企業的使用者也可以利用各種資料庫系統作為資料倉儲的工具，自行開發資料挖掘的應用程式，以達到資料挖掘的目的。

3. **知識擷取階段**

此一階段中首先確定資料倉儲中哪些欄位的資料是必須的，先將這些主要的欄位從整體資料庫中取出，再配合利用各種資料挖掘的演算法，將使用者想要挖掘的資料從資料倉儲中取出，執行演算法相關程式後，將關聯的資訊從資料庫中擷取出來，成為有用的知識，此一階段為資料挖掘階段，也是各種演算法在改進效能時所著重的階段。

4. **規則詮釋階段**

以上的幾個階段中，由電腦程式中擷取出來的規則或項目組合，或許僅是幾個項目的關聯或是一些代碼，對於非相關領域的人看來或許只是幾組沒有意義的代碼或名詞。資料挖掘至此必須經過此一領域相關的學者加以詮釋，並確認資訊的可用性，將新奇有用且讓使用者感興趣的資訊取出，用一般大眾都能了解的名詞解釋出來。此一階段就是規則詮釋的階段。而經過一般化的資訊或是規則，就是一般所說的知識或是規則了。

此種將有用的知識或是規則，從資料庫系統中挖掘出來的整體過程，就是本篇論文所探討的資料挖掘範疇了，所以也有人說，資料挖掘的過程是也就是知識發現(Knowledge Discovery)**的過程。**

整個資料挖掘的過程示意圖如下所示：

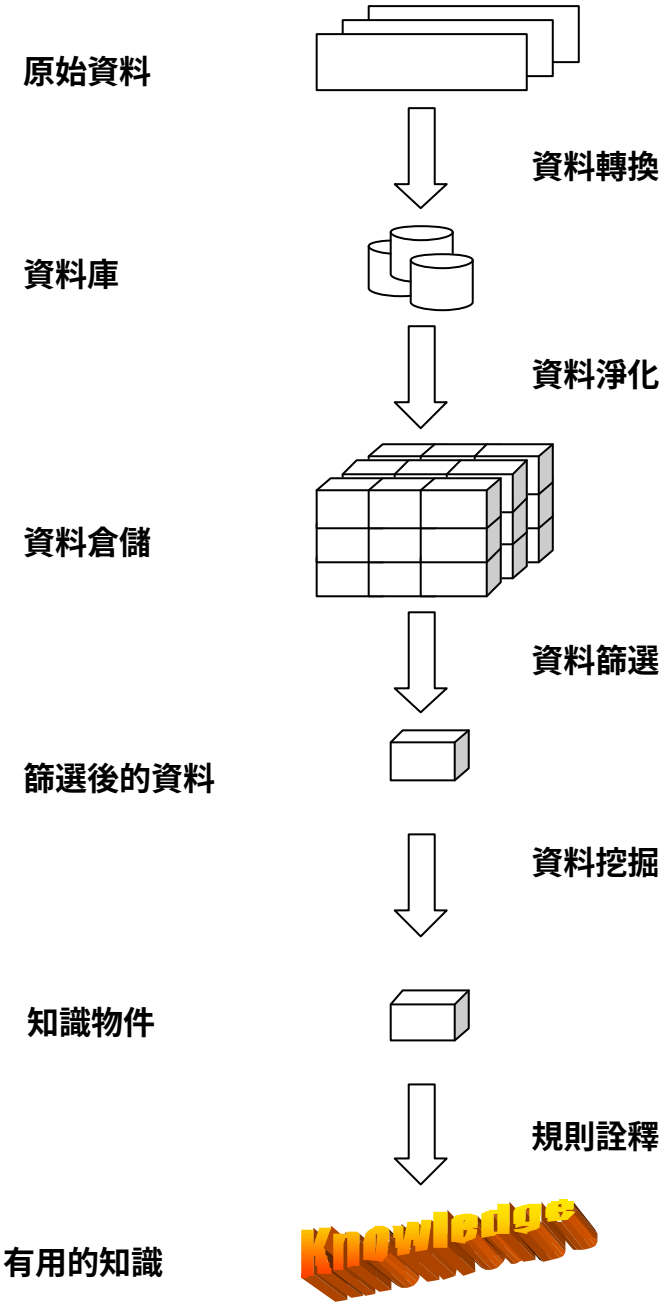

圖 1.1 資料挖掘過程示意圖

第二節 資料挖掘的方法

資料挖掘的研究，分為方法論與技術論兩種，或是說為了配合某種方法，而發展出某一項技術，以下是方法論與技術論在學術上的幾項名稱：

一、 方法論

為瞭解決不同類型的領域中資料挖掘的問題，資料挖掘區分為許多不同的方法，例如分群法[2]、分類法[3]、時間序列[4]、關聯法則[5,6]、路徑行走樣型[7]等不同的資料挖掘方式。

二、 技術論

從技術論的觀點去看資料挖掘，也就是如何將是資料從原始資料庫中擷取出來的演算方式，學術上常見的幾項有名的資料挖掘方式及演算法有統計學、作業研究、類神經、基因演算法與模糊理論等方式，此外尚有 Apriori 演算法[8]、Partition 演算法[9]、Sampling 演算法[12]、DHP 演算法[13]、DIC 演算法[14]、Column-Wise 演算法[15]等不同的演算方式。

為了配合各種不同的領域與不同類型的資料，所以必須以不同的方法配合不同的技術，才能完成資料做擷取的動作。例如當需要分析股市交易行情時，必須考慮時間點的問題，所以在方法論上需要用時間序列的方式分析資料；而在分析使用者在網路上瀏覽網頁習慣的資料挖掘時，則可以利用路徑行走樣型的方式，分析使用者按下網頁連結的習慣，進而分析使用者的喜好等。選擇了分析的方式後尚需配合相關的演算法進行挖掘的動作，而這些相關的演算法會在下一章中做詳細的說明。

第三節 資料挖掘的應用

不同的資料挖掘方式，應用在不同的領域與不同的時機。例如在探討股市的預測時，需考慮時間的變動與各股之間相互關係，此一研究適合時間序列的資料挖掘方式；另外如在探討使用者在網際網路間的瀏覽行為時，資料的變化在於使用者點選網頁的連結的先後次序，此種資料處理的方式就可以套用在路徑行走樣型的資料挖掘上。

此外，同一領域中的資料，應用不同的資料挖掘手法，亦可以得到不同的相關資訊。例如在研究旅遊業的客戶資料與其旅遊資料時，若以使用者為挖掘的對象，則可以得到使用者的偏好與其本身屬性的關係(如學歷、興趣、職業與教育背景等)。如果將旅遊景點的資料加以分析，則可以得到例如季節與人潮之間的類似關聯。

原始資料經過資料挖掘的整個過程，得到了相關的關聯之後，使用者便可以設計更能符合使用者需要的功能或是提供更佳的便利性。如利用股市預測機制提供股友投資理財的資訊，或是將網頁中的連結，依照不同的使用者與其相關喜好，給予客製化的頁面顯示。

此外尚需注意的是，挖掘出來的資料在應用層面的考量，例如在錄影帶出租店中的使用者租片資料，可以挖掘出使用者的喜好，應用的方式是推薦給其他的客戶，因為同一部影片看兩次的人不多，但是在KTV 點歌的使用者，設定“我的歌本”，其中的點歌紀錄不但可以作為他人的推薦歌曲外，亦可同時提供點歌者下次進入歡唱時的點歌參考。

資料挖掘的演算方式本身屬於一門應用科學，應用的範圍卻是可以應用在各行各業的領域中，但各領域都需要有專業的人員參與資料挖掘的工作，將挖掘出來的資料做一有意義的詮釋，才能有效的將資料挖掘的工作做好。

第四節 研究動機與目的

相對於以往電腦的使用情形，以及資料的記錄量，目前的電腦可以說已經深入各行各業之中。不論是大小型的企業、醫院、學術領域、亦或是銀行業，電信業…等，資料的儲存已經由人工登錄的方式轉成電子檔案的格式，甚或是資料庫的型式。相較於之前的記錄方式，現在以電腦為儲存媒介的優勢已廣為人們所知。

以往的企業總是認為擁有越多客戶資料越佔優勢，然而在盛行的網路傳銷泡沫化之後，此種論調逐漸被淘汰，此時使用者開始真正的去思考：什麼才是有用的資料？哪些客戶是具有潛力開發的？客戶所感興趣的是哪些服務或商品？如何利用手上的資料，找出決策或是經營的模式？於是在現在的企業間，類似客戶資源管理(Customer Resources Management)、系絡行銷(Contextal Marketing)[16]等相關議題已漸漸的被企業主們所接受。而企業利用本身的交易資料作為輸入，而得到的結果是能夠被企業使用的關鍵資訊，其中的處理過程，就是將關鍵性資訊挖掘出來的方式，對於企業主來說，如何從眾多的資料中，快速且正確的找出使用者所感興趣的資料，提供使用者做決策時有一個系統化的參考依據，而不是靠使用者主觀的感覺去做判斷，此一系統化的方式所產生的資料，將是使用者所需要的關鍵性資訊，而此一系統化的過程也就是我們所感興趣的資料挖掘了。

最近幾年以來，資料挖掘已經是學術界及企業間熱門的領域，從資料中找尋項目的相關性就成了大家的討論議題，相關研究的學者已經就此一領域提出許多演算方式，企圖解決資料挖掘時的問題，繼 Apriori 演算法以來，相繼提出了 FUP 演算法、DHP 演算法，DIC 演算法，Sampling 演算法…，而每種演算法都企圖改善 Apriori 演算法的缺點，

增進資料挖掘時的速度，以期望在資料挖掘時能得到效率的提昇。

然而面對現實生活，企業交易是隨時產生的，而資料庫中的資料也必須隨著交易的新增而動態的記錄新的資料，反應交易量的增加。但在應用舊有的演算法處理動態新增資料庫時，若需要計算新的關鍵性資訊，均需要將整體的資料庫納入考量，重新計算關鍵資訊，無法單獨處理新增資料的部分。當資料量日漸增多時，此種方式將會使系統的負擔越來越重，且無法在使用者需要的時候，即時的產生使用者所需要的關鍵資訊。因此產生了動態資料挖掘的需求。

如果有些資料在短期間內產生的資料量不能在整體的交易資料庫中，佔有足夠的交易次數，但這些資料是具有潛力的敏感性資訊，且實際的反映了市場的實際變化，以往的演算法也無法將此類的資訊即時挖掘出來，即時提供使用者市場變化的訊息，因此產生的敏感性資料挖掘的議題。

對於上述兩個資料挖掘的需求，相信是目前及未來現實生活中所必須面臨的資料挖掘方式，為此，本篇論文對於動態資料及敏感性資料的挖掘，提出了相關的研究方式，目的即為解決以上提出的兩個資料挖掘的問題。

第五節 研究方法

以 Apriori 演算法為基礎所推導的演算方式，都是在記憶體內計算資料庫中的相關欄位資料，產生候選項目組，再以使用者設定的門檻值來產生高頻項目組。當高頻項目組產生後，計算過程的相關資訊即從記憶體中移出，僅留下高頻項目的相關資訊。當資料新增時，此類的演算方式需要將原有的資料與新增的資料合併，重新掃描整體資料

庫，計算高頻項目。

本論文為改善 Apriori 演算法在動態新增資料庫挖掘時，挖掘效率不佳的情形，試圖利用以往的演算法，加上資料結構的觀念，提出一個多維陣列的儲存空間，紀錄每次資料挖掘處理過的資訊。而當資料新增時，僅對新增的資料加以處理，即時拆解單一資料項目，並與舊有資料合併計算，以此一方式解決交易資料動態新增的問題。

另外，為解決敏感性資料需即時挖掘出來的問題，本文亦提出了一個演算方式，以上述的演算方式及資料結構為基本架構，加上敏感度指標的計算方式，取出交易資料庫中近期的敏感性資訊，即可將使用者設定之敏感性資料挖掘出來，避免重要資訊無法即時產生之遺憾。

第二章 相關研究

資料挖掘的有許多不同的的方式，如第一章所介紹的數種不同的方式，而現在廣泛為大家所討論的方法之一就是關聯法則(Association Rule)的資料挖掘方式。在實際介紹本篇論文所提出的兩個演算法之前，本章中主要介紹幾種在學術領域中常見的關聯法則的推導方式，以及動態資料挖掘方式的相關研究資料。由於我們研究的方向所著重的演算方式為關聯法則的推導，所以在本章節中所介紹的也是以此一方向做一個概略性的介紹。

本章主要區分為三個部分，依次為關聯法則的演算方式、動態資料庫挖掘演算方式以及其他相關的資料挖掘法則。

第一節 關聯法則資料挖掘的演算方式

關聯法則的定義為在資料庫中的不同項目或屬性之間共同發生的關係，例如在便利商店中，“80%購買麵包的客戶，也會同時購買牛奶”，則可稱為麵包與牛奶具有某種程度的關聯。

在本章節中介紹的是目前學術上較常見到的幾種關聯法則的演算方式，以 Apriori 演算法最早提出，而後續有許多的演算法都是以 Apriori 為基礎加以改良，希望能減少資料挖掘時所花費的成本，尤其是在搜尋整體資料庫的次數，進而提昇演算法本身的效能。

2.1.1 Apriori 演算法

Apriori 演算法首先於 1994 年由 Agrawal *et al.*提出，此一演算法是目前研究資料挖掘時最具代表性的演算法之一。Apriori 演算

法演算方式如下：

```
L1 = {large 1-itemsets};
  for (k=2;Lk-1 ≠ 0;k++) do
  begin
      Ck=apriori-gen(Lk-1);          // 產生新的候選項目集
      forall transaction t∈ D do
      begin
          Ct = subset(Ck,t);         // 取出 Ck 中的各項目子集
          forall candidates c∈ Ct do
              c.count++;
      end
      Lk={c∈ Ck|c.count≧minsup}
  end
Answer = ∪kLk;
```

圖 2.1 Apriori 演算法

從其中的虛擬碼可以看出，Apriori 演算法中包含了兩個重要的步驟：

1. 反覆的產生候選項目組和搜尋整個資料庫，直到找出所有的大項目組。
2. 利用(1)所找出的大項目組，推導出所有的相關法則。

在圖(2-1)所列出的虛擬碼中，subset(C_k,t)函式用來判斷部分交易項目 t 是否包含在候選項目組 C_k 內，也就是在判別 t 是否為 C_k 的子集合。

而圖 2.1 中所示的虛擬碼中，apriori-gen()副函式的主要的最重要的動作，就是將 L_k 的項目組合，經過聯結(join)之後，產生下一階段的候選項目組 C_{k+1}，以提供主函式繼續往下搜尋資料庫的項目組之用。其副函式虛擬如下所示：

insert into C_{k+1}
select $p.item_1, p.item_2, \ldots, p.item_k, q.item_k$
from $L_{k\,p}, L_{k\,q}$
where $p.item_1 = q.item_1, \ldots, p.item_{k-1} = q.item_{k-1}, p.item_k < q.item_k$

圖 2.2 apriori-gen()**副函式**

在步驟(1)**產生大項目組的過程中，** Apriori **演算法由單一項目組**(1-itemset)**開始，逐層產生相關項目組。此過程分為兩個階段，第一個階段為產生新的項目組，若相關項目的長度為** k**，則稱為候選** k-**項目組**(candidate k-itemset)**，記為** C_k**；第二階段為搜尋資料庫中** C_k **的支持度是否大於使用者最初設定的最小支持度門檻值的限制，符合條件的項目組** C_k **便稱為大項目組(或稱為高頻項目組)，稱為大** k-**項目組**(large k-itemset)**，記為** L_k **而不符合最小支持度限制的** C_k **項目組則刪除。**

根據以上的步驟，而後再由 L_k **與** L_k **的聯集產生下一層的新候選項目組** C_{k+1} **，並再搜尋資料庫以產生** L_{k+1} **。如此反覆遞迴產生下一層級的** C_{k+1} **與** L_{k+1} **直到資料庫中所有的大項目組均被搜尋出來為止。**

根據以上所描述的步驟，推導 Apriori **演算法的範例如下：**

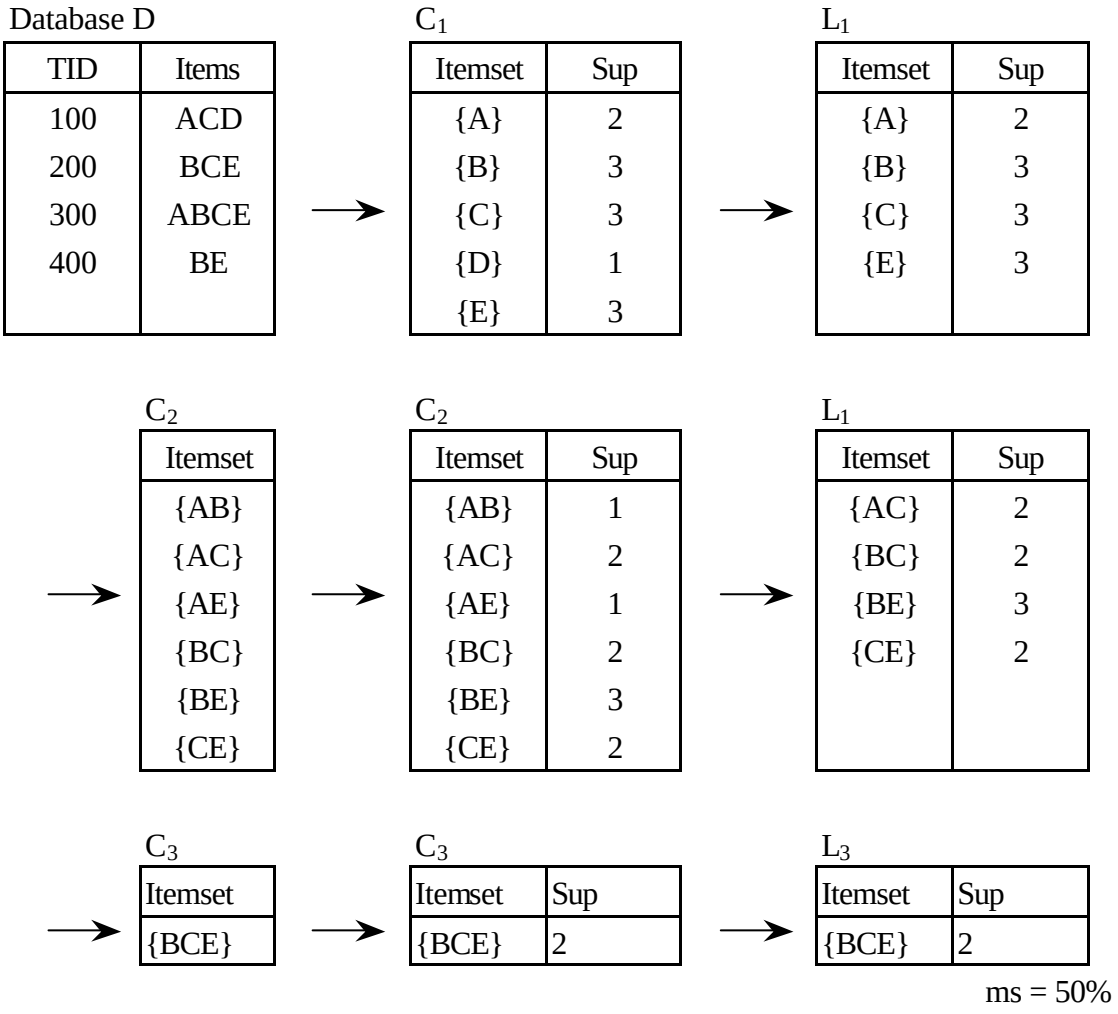

圖 2.3 Apriori **演算法之推導範例**

而由 Apriori **演算法作關聯法則的推導時，必須多次的掃描資料庫，產生效能不良的情形。隨著資料庫中的資料項目與交易次數的增多，挖掘的時間便以指數型態的成長，如此不能適應大型資料庫或是動態新增資料庫在資料挖掘的需求。在以往的研究中，**DHP **演算法利用減少** 2-itemset **的個數，減少計算量增進效能，而** DIC、Partition、Sampling **等演算法則是減少掃描次數的方式來改善效能。以下為此四種演算法的介紹。**

2.1.2 DHP 演算法

Chen *et al.*於 1997 年提出了 DHP(Direct Hashing and Pruning)演算法，此一演算法是利用 hash table 的架構，在資料挖掘的動作時，預先刪除不必要的 C_2 項目組，以增進資料挖掘的效率。DHP 的演算方式如下所示：

表 2.1 DHP 原始資料

TID	Items
100	ACD
200	BCE
300	ABCE
400	BE

1.即時分解及計算單一高頻項目組

C_1	count
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

→

L_1
{A}
{B}
{C}
{E}

Minimum support, s=50%

圖 2.4 DHP 演算法推導單一高頻項目組

2.**將交易資料表即時拆解成** 2-Itemset

100	{AC},{AD},{CD}
200	{BC},{BE},{CE}
300	{AB},{AC},{AE},{BC},{BE},{CE}
400	{BE}

圖 2.5 DHP **演算法拆解** 2-Itemset

3.**依照資料庫的大小，事先建立一個** hash function**，將交易資料表中的資料依照** hash function **計算之後得到的數值，放入** hash table **中。**

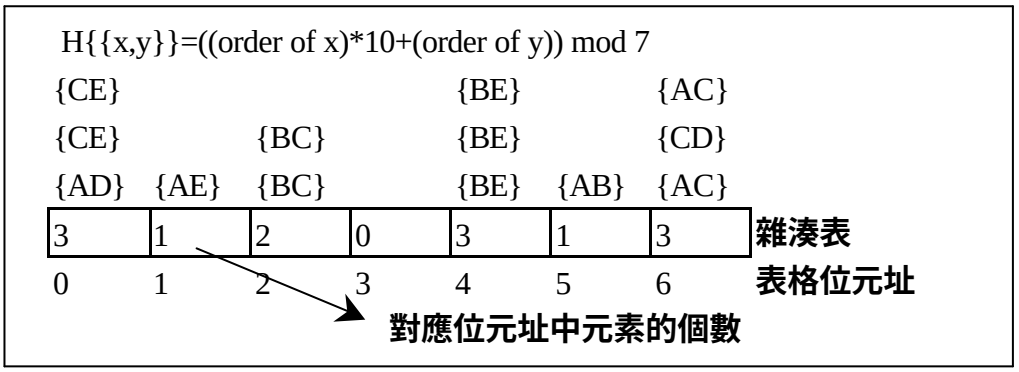

圖 2.6 DHP **演算法示意圖**

4.**由** L_1 **與** L_1 **的聯集與雜湊表中對應的數值，直接取出** C_2 **。**

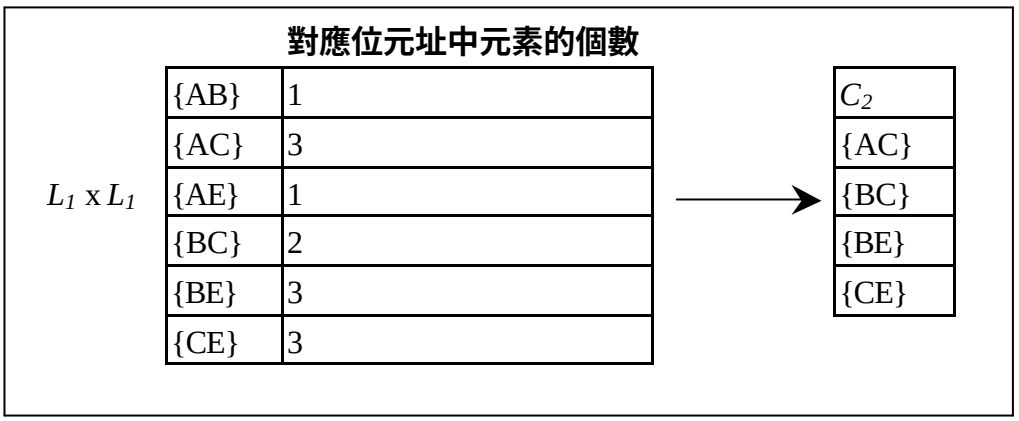

圖 2.7 DHP **演算法雜湊表對應圖**

DHP **演算法改進** Apriori **演算法的地方在於，當** L_1 **與** L_1**聯集時，**

Apriori 會產生過多的 C_2 項目組，但 DHP 演算法利用 hash table 的方式，大量的刪除不必要的項目組，在 C_2 必須掃描資料庫求得 L_2 的時候，便可以減少許多候選項目組在資料庫搜尋時所花費的 I/O 成本。雖然 DHP 演算法在一開始花費部分額外的時間在建立 hash table，但對於之後減少資料庫掃描所省下的的時間卻能提供相當大的效能改善。

以上述的例子而言，Apriori 演算法會產生{AB}，{AC}，{AE}，{BC}，{BE}，{CE}等六組候選項目組，但 DHP 卻僅產生{AC}，{BC}，{BE}，{CE}四組候選項目組，因為{AB，{AE}在 hash table 中的支持度不滿 50%的要求而被事先刪除，整體而言，此一方式的確具有節省掃描資料庫，達到效能提昇的能力。

2.1.3 DIC 演算法

Brin *et al.*在 1977 年提出了 DIC(Dynamic Itemset Counting)演算法，此一演算法以 Apriori 演算法為基礎加以改善。如前所述，DIC 演算法主要是減少 I/O 存取的次數做為資料挖掘效能的改進方向。

DIC 演算法主要的演算方式為將資料庫分割成許多區段，在搜尋資料庫時以區段為單位，而當候選項目組產生了高頻項目組時，DIC 演算法不需完整的搜尋整體的資料庫，因為在搜尋的過程中，有些候選項目組在部分區段的搜尋中以達到使用者設定的最小支援度設定，或是經由計算得知已不能成為高頻項目組，而在搜尋整體資料庫前預先濾除，以達到減少搜尋資料庫的目的。

DIC 演算法與 Apriori 演算法的差異如下所示：

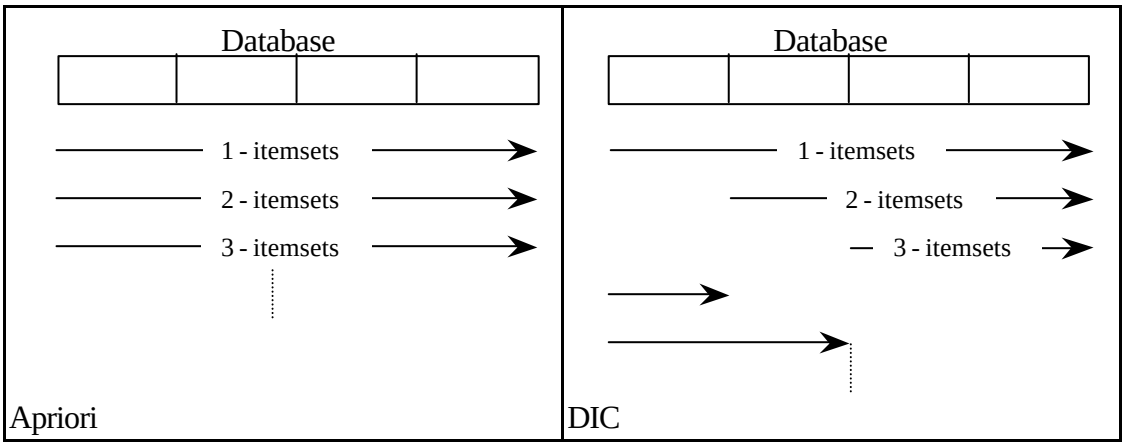

圖 2.8 DIC 演算法與 Apriori 演算法的差異圖

以上圖的 DIC 演算法為例，若將資料庫劃分為四個等份，則每個等份中所可以得到的最大支持度為 25%，也就是說，某一候選項目組為其中任一等份的每一個交易的子集合，則可以得到整個交易資料庫的 25%的支持度。而 DIC 可以用兩種方式減少資料庫搜尋的次數。

1. 若 DIC 演算法設定最小支持度為 40%，資料挖掘時某一項目組如果在最前面兩個資料區段中，已經達到 40%以上的支持度，即可將此一項目組提升為高頻項目組中，而後面兩個資料區段即可不需搜尋
2. 同上，若最小支持度為 40%，在挖掘完前面三個區段後，若支持度不滿 15%，即使在最後一個區段中可以得到 25%的支持度，也不能滿足 40%的最小支持度，所以可以在第三區段搜尋完畢時，即將此一項目組列為非高頻項目組。

以上兩種方式即為 DIC 演算法在減少資料庫搜尋時，所利用的方式，如此即可以省下許多資料庫掃描的時間，增進演算法的效能。

2.1.4 Partition 演算法

Savasere *et al*.於 1995 年依據 Apriori 演算法，提出了一種改良的演算方式，名為 Partition 演算法。正如其名稱一樣，Partition 演算法是依據系統記憶體的大小，將資料庫劃分為許多區段，使得每一區段的大小能容納於記憶體中計算，利用記憶體處理速度快的特性，加上資料分區的方式，可以增加搜尋的速度外，也可以減少演算法在資料搜尋時所花費的成本。

在 Apriori 演算法中，資料的項目與屬性的增加，會使得搜尋的時間成指數倍的成長，但 Partition 演算方式，將高頻項目組的搜尋分為兩個階段：

1. 個別計算各區段內各相關項目組的支持度，找出各區段中的所有高頻項目組。
2. 集合各區段中所有的高頻項目組，再次搜尋資料庫中真正的高頻項目組合。

在第一個階段中，Partition 演算法分別將資料區段中的資料自資料庫中取出，放入記憶體中加以處理，以 Apriori 演算方式，將各區段中的高頻項目組取出，並加以紀錄。

在第二個階段中，再根據第一階段處理過的資料，將所有區段中的高頻項目組，加以搜尋整理的資料庫，取出資料庫中真正的高頻項目。因為在第一階段中若在各區段內均為非高頻項目組，其在整體資料庫中以必為非高頻項目組。但在其中一個區段內為高頻項目組，整體資料庫不一定為真正的高頻項目組，所以在此一階段中必須加以驗

證，才能求出真正的高頻項目組。

Partition **演算法如下所示：**

```
P = partition_database(D)          // 部分資料區段
    n = Number of partitions       // 區段的總數
    //Phase I
    for i = 1 to n do
    begin
        read_in_partition(p_i ∈ P)
        L^i = gen_large_itemsets(p_i)
    end
    //Merge Phase
    for (I = 2;Lij<>0;j=1,2,…,n;I++) do
    begin
        C_i^G = ∪ j=1,2,…,n L_i^j
    end
    //Phase II
    for i = 1 to n do
    begin
        read_in_partition(p_i ∈ P)
        for all candidates c ∈ C^G gen_count(c,p_i)
    end
    L^G = {c ∈ C^G|c.count ≥ minSup}
    Answer = L^G
```

圖 2.9 Partition **演算法**

跟 Apriori **演算法比較，**Partition **演算法僅需要搜尋整體資料庫兩次，因此可以降低** I/O **的成本，但相對的在第一個階段中，各區段資料會產生較多的候選項目組，但若與之前討論的指數問題相比，**Partition **演算法的確會節省許多資料挖掘的時間。**

2.1.3 Sampling 演算法

Toivonen *et al.*在 1996 年，利用統計學中的取樣觀念，並沿用上一節中所介紹的 Partition 演算法，將資料劃分為數個區段，在記憶體中處理的技術，進而提出了 Sampling 演算法。此一演算方式即為利用取樣的技術，將資料庫中的資料隨機抽取樣本進行處理，求出樣本中的關聯法則，並以此一關聯法則特徵對整體的資料做關聯性的推導。

由於此關聯資料為取樣的方式，樣本數及資料分佈的範圍均會大幅影響資料挖掘的結果，相對的其可信度也會有所偏差。

第二節 動態資料庫挖掘相關演算方式

在第一節中所提出的各種演算方式，均將資料庫中的資料預設為靜態的資料，也就是說資料已經固定，不會隨著時間的改變而有所增加。或是在資料增加的時候，必須重新掃描整體的資料庫，才能求得關聯性的相關法則。如此一來，當資料量漸漸增多，或是經常變化的時候，以上的演算方式就不能解決動態新增資料庫的問題了。

在動態資料庫中做資料挖掘的動作，最重要的觀念，就是將已經挖掘過的資料做一處理，儲存各相關項目組的關鍵值，當資料新增的時候，僅需對新增的資料作處理，之後再與舊有的資料做一彙總的動作，得到新的關聯法則。如此的方式就可以在資料新增的時候，快速的進行資料挖掘的動作，而無須重新挖掘整體的資料庫。

以下提出幾種對動態資料庫挖掘時所提出的相關演算法：

2.2.1 FUP 演算法

Cheung *et al.*為了改善 Apriori 演算法無法對動態資料庫做一有效的挖掘，提出了一種名為 FUP(Fast UPdate)的演算法[10,11]，此一演算法以 Apriori 演算法為基礎，批次計算某一段時間所產生的交易資料，而主要的處理方式為將挖掘過的資料儲存至資料庫中，當新的交易資料進入時，FUP 演算法僅需對新增的資料做相關法則的推導，再與舊有的資料加總計算，即可得到新的關聯法則，無須重新計算整體的資料庫。

以下為 FUP 演算法的處理過程，其處理分為兩個過程，第一的過程為處理原始資料庫，求出各項目組的高頻項目組，此時處理方式與使用 Apriori 演算方式並無二致，但在處理完後，必須將各項高頻項目

組的計數值加以儲存。第二個過程是當新增資料進入 FUP 演算法時，FUP 演算就新增資料加以處理，也是利用 Apriori 演算方式一樣，將新增項目中的高頻項目組取出，此時會發生下列四種情形：

1. 舊有資料庫中為高頻項目，新增資料中亦為高頻項目。
2. 舊有資料庫中為高頻項目，新增資料中為非高頻項目。
3. 舊有資料庫中為非高頻項目，新增資料中為高頻項目。
4. 舊有資料庫中為非高頻項目，新增資料中為非高頻項目。

在第一及第四種情形中，新舊資料集均為高頻項目或是非高頻項目，其最終結果即是高頻的亦為高頻項目，非高頻的亦為非高頻項目組。而 FUP 演算法最主要的處理，就是在第二與第三種狀況，其處理的過程如下：

假設原始資料如下所示，設定最小支持度設定為 50%

表 2.2 FUP 原始交易資料

TID	Items
100	ACD
200	BCE
300	ABCE
400	ABE
500	ABE
600	ACD
700	BCDE
800	BCE

利用 Apriori 演算法可以得到如下表所計算結果

表 2.3 FUP 演算法各高頻項目組計算結果

1-item	Counts	2-items	counts	3-items	counts
A	5	BC	4	BCE	4
B	6	BE	6		
C	6	CE	4		
E	6				

舊有的資料如表 2.1.2 所示，將會記錄至資料庫中，而在新增資料進入 FUP 演算法時將合併計算。假設新增的交易資料如下所示：

表 2.4 FUP 新增交易資料資料

TID	Items
900	ABCD
1000	DEF

利用 FUP 演算法處理的過程如下圖所示：

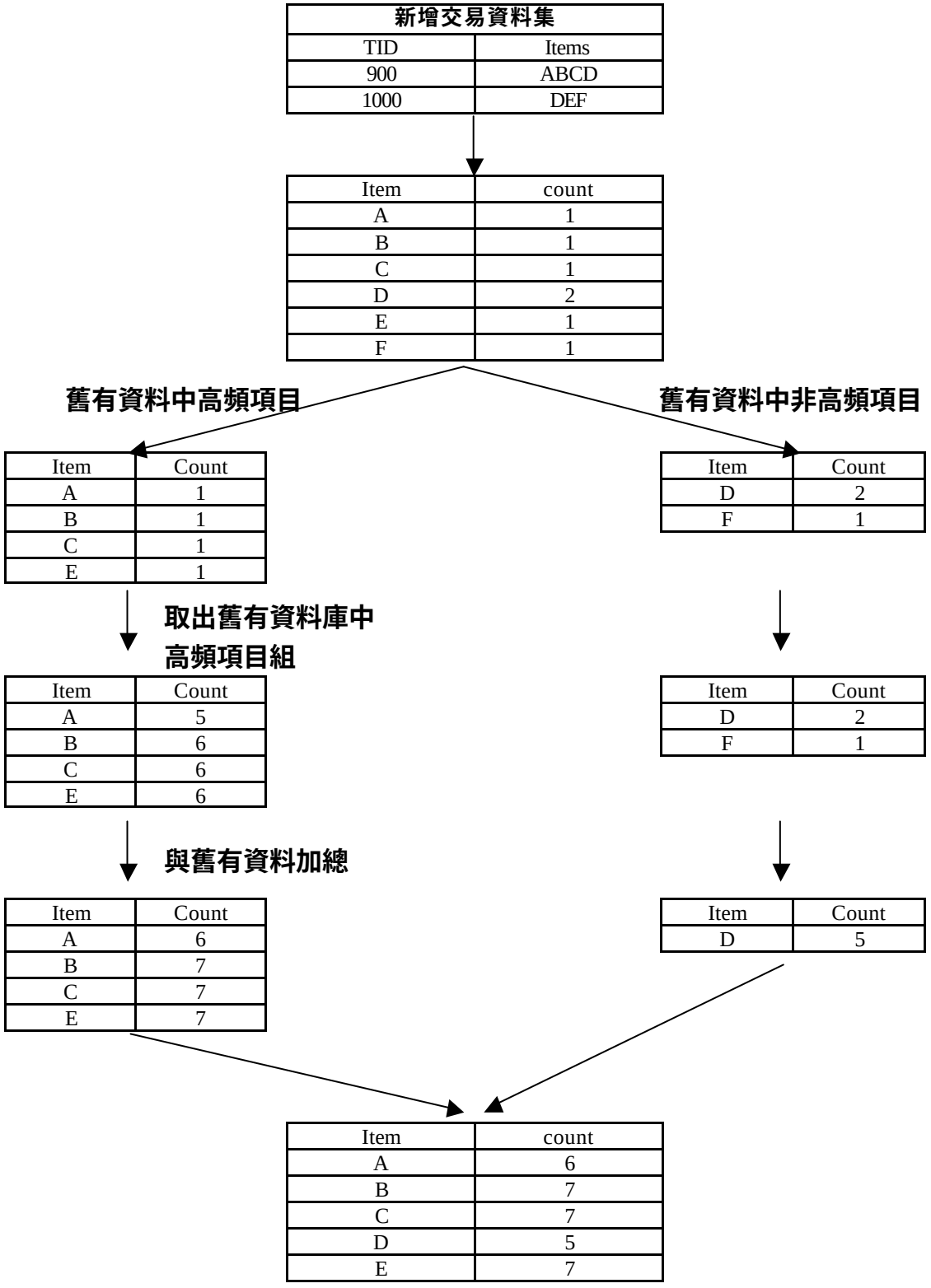

圖 2.10 FUP **演算法處理流程示意圖**

從上圖中可以得知，在新增交易進入 FUP 演算方式計算時，FUP 演算法將新增資料計算出 1-item 的 count 值後，依照原始資料是否為高頻項目，分解為兩個部分，其中一個部分為原始交易資料中高頻項目{A}、{B}、{C}、{E}，以及原始交易資料中的非高頻項目{D}、{F}，並將此兩類資料分開計算，並與原始資料加總。再依照上述的四種情形歸類，得到新的高頻項目組與非高頻項目組，再將新的高頻項目紀錄至資料庫內。

FUP 演算法的優勢在於預先紀錄高頻項目組的資訊，當新增資料進入計算時，僅需計算新增資料的部分，原始高頻項目組可以跟新增資料的 count 值加總，合併計算。如此可以避免重新搜尋整體資料庫所花費的時間及處理大量資料時所耗費的系統資源。

但 FUP 演算法在批次處理新增資料的時候，可以視為 Apriori 演算法，所以記憶體及 CPU 所消耗的系統資源 Apriori 演算法一樣，會隨著資料量的增大而變多。再者，若項目組在新增資料中為高頻項目組而原始資料中為非高頻項目組，即必須再次搜尋整體的資料庫，而從原始高頻項目與新增項目合併計算後若成為非高頻項目，亦必須移出高頻項目組。若有一項目在高頻與非高頻項目組的門檻值間來回的改變，FUP 演算法就必須反覆的計算此一項目組，並將此項目組移入移出，造成系統的負擔。

2.1.2 Pre-Large Itemsets Mining

王慶堯於八十九年提出了一種準大項目集之漸進式挖掘(Incremental Data Mining Using Pre-large Itemsets)[19]，此一演算法在於解決 FUP 演算法的缺點，提昇動態資料挖掘時之效能。

準大項目集之漸進式挖掘定義了一個“準大項目集”的觀念，將單

一的支持度門檻值轉成"高支持度門檻值"與"低支持度門檻值"，而準大項目集就是介於此二個門檻值間的項目組集合。準大項目集的運作如同一個區間，在於減少 FUP 演算法在處理高頻項目與非高頻項目間轉換的問題，且在資料庫越來越大時，此一演算法之效能也會隨之提高。

由上一節 FUP 的演算法可以得知，FUP 演算法將資料的轉換定義為四種，但在準大項目集之漸進式挖掘時定義了九種不同的案例，其型態如下所示：

1. 舊有資料庫中為高頻項目，新增資料中亦為高頻項目。
2. 舊有資料庫中為高頻項目，新增資料中為預高頻項目。
3. 舊有資料庫中為高頻項目，新增資料中為非高頻項目。
4. 舊有資料庫中為預高頻項目，新增資料中為高頻項目。
5. 舊有資料庫中為預高頻項目，新增資料中亦為預高頻項目。
6. 舊有資料庫中為預高頻項目，新增資料中為非高頻項目。
7. 舊有資料庫中為非高頻項目，新增資料中為高頻項目。
8. 舊有資料庫中為非高頻項目，新增資料中為預高頻項目。
9. 舊有資料庫中為非高頻項目，新增資料中亦為非高頻項目。

其中第 1,5,6,8 項不會影響原有的關聯結果，第 2,3 項將有可能將原有規則移出，而第 4,7 項則是有可能增加新的規則到關聯法則中。準大項目集之漸進式挖掘的原理即為保留高頻項目組與預高頻項目組，在資料新增時有一個緩衝的區間。如 FUP 演算法在高頻與非高頻間轉換的時候，必須重新掃描整體的資料庫以求得該項目組的支持度與可信度。而準大項目集之漸進式挖掘可以有效的解決此一問題，提供動態新增資料庫一個十分良好的演算方式。

2.2.3 縮減項目集洛挖掘方式

許志豪於八十九年提出了一種利用縮減項目集絡，在動態資料庫中做線上挖掘關聯式法則的方式[20]。其中最主要的方式即為縮減項目集絡(Reduced Itemset Lattice, RIL)的觀念。

此一演算法假設的重點在於：

1. 在有限的記憶體空間中，要預先計算並儲存所有的項目集是不可能的。
2. 要提供使用者'線上調整門檻值，則不管使用者設定的門檻值為何, 都必須能產生相對應的規則, 因此儲存'的資訊越詳細越好。
3. 要達到線上挖掘，則必須在有限的時間內產生所有的規則，因此在挖掘得過程中較花費時間的掃描資料庫及計算工作必須在挖掘之前完成。
4. 在資料庫更動時能同步調整所儲存的資訊，若儲存的資訊越詳細，調整時就越能減少必須到資料庫做確認的情況。

為了達到上述的四項要求，其演算法利用 Apriori principle 的概念提出了一種利用"絡"(Lattice)的方式作為資料結構儲存的架構，用以儲存其演算法產生的規則，其演算法的做法如下：

1. 先決定一個預設最小支持度門檻值來作為縮減項目集洛的標準，一方面減少所儲存的資料量，一方面也限制最高項目個數。
2. 首先掃描一次資料庫, 找出所有單一項目集(1-itemset)在資料庫出現的次數，利用預設門檻值找出所有高頻項目集(Large

Itemset)，再由這些高頻項目集相結合產生候選 2 項目集。再掃描一次資料庫，計算候選項目 2 項目集出現次數後，再找出高頻項目來結合候選項目 3 項目集。

3. 重複上述步驟，如此一再的掃描資料庫、計算規則強度即結合產生候選項目集，一直做到沒有任何候選項目集再被結合出來為止。

舉例來說，若資料庫中存在 5 個項目 A、B、C、D及E，則利用預設門檻值建立縮減項目集絡的步驟為：

1. 利用上述的方式產生高頻項目，若發現單一項目集A、B、C為高頻項目，而D、E為非高頻項目。
2. 由高頻項目集可以結合候選 2 項目為AB、AC及BC，再次掃描資料庫，找出此三個項目集出現的個數，並判定項目集AB為高頻 2 項目集。
3. 由於無法再結合出新的候選項目集，因此建構項目集的工作到此完成。

依照上述的演算過程，所產生的縮減項目集絡如下圖所示：

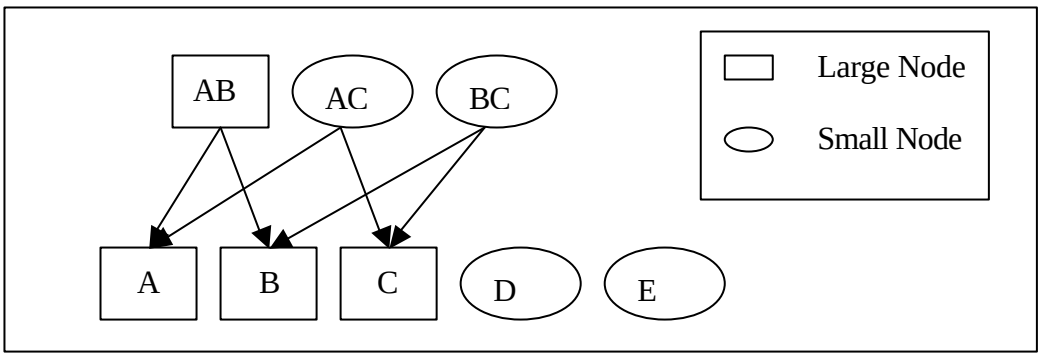

圖 2.11 縮減項目集絡範例

在紀錄相關資訊的時候，為了能夠動態產生符合條件的高頻項目，1-itemset A、B、C、D及E的計數值全部紀錄。但由於D、E為非高頻項目，在產生候選 2 項目時，僅利用A、B、C三項產生AB、AC及BC，並將此三個項目組的計數值均紀錄至資料庫中，以及找出高頻 2 項目組AB。此時，在圖中的高頻節點及非高頻節點的計數值均紀錄至資料庫中，若有新增項目進入計算時，RIL 及可利用對應的演算方式產生相關的關聯規則。

利用縮減項目集洛的方式，可以將必要的資訊預先存放在資料庫中，且能在資料新增時，利用這些預先儲存的資訊快速的推導出完整的規則，以提供動態資料庫線上挖掘關聯式法則。

第三節 其他資料挖掘相關的法則

2.3.1 廣義性之關聯法則

在本文之前所提出的演算法，都是項目與項目之間的關聯關係，屬於相當底層的關聯對應，例如“紅豆夾心的統一麵包與光泉鮮乳小包裝”的關聯，或是買了“三槍牌中號的襪子與 31 號的阿瘦皮鞋”的關聯，如此即為項目與項目的對應關係。類似如此對應關係的關聯法則，如果詳加紀錄，在資料庫中所記錄的資料相當詳細，但是對於資料挖掘的演算法而言，卻會衍生出過多項目組的對應關係，造成資料挖掘時的負擔。

所謂廣義性的關聯法則[17]，就是以較高層次項目組彼此之間的關聯，取代較低項目之間的關聯，例如將紅豆夾心與芋頭夾心，花生夾心…中，屬於統一麵包的商品歸類為“統一麵包”類，而光泉鮮乳大包裝、中包裝、小包裝等，都歸類為“光泉鮮乳”，這樣在歸類關聯法則的時候，即可將許多不同的項目，歸類為同一個項目組，而關聯法則亦可從“紅豆夾心的統一麵包與光泉鮮乳小包裝”的關聯轉成“統一麵包與光泉鮮乳”的關聯對應關係了。同理，若是商品的項目還是太多，廣義性關聯法則的對應關係亦可從“統一麵包與光泉鮮乳”的對應，轉成“麵包與鮮乳”的關聯對應關係。其項目對應關係如下所示：

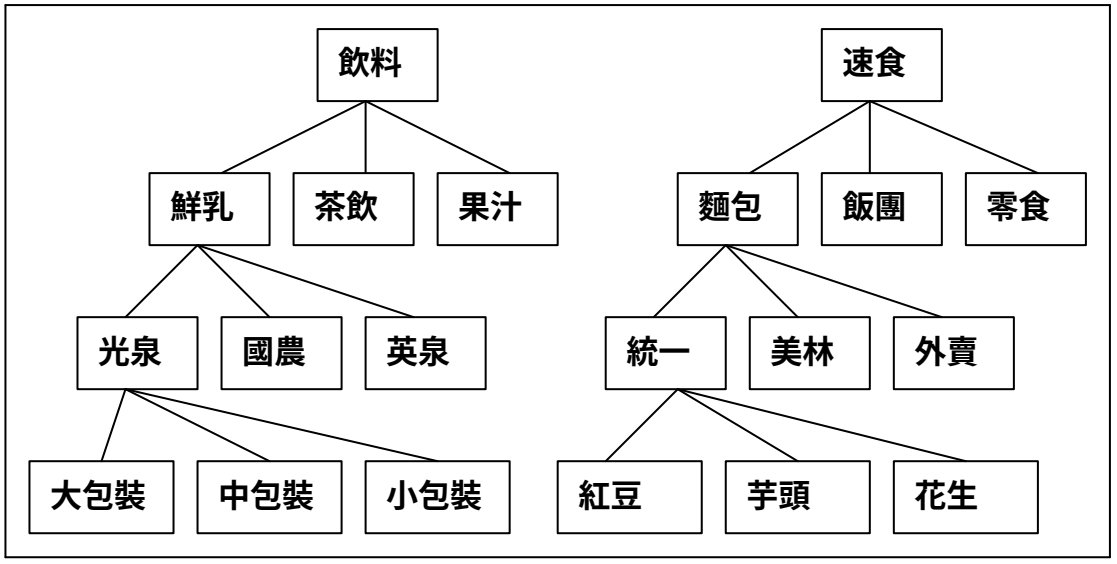

圖 2.12 廣義性關聯法則階層圖

類似以上所敘述的，即為廣義性的關聯法則，為了處理廣義性關聯法則的應用，Srikant *et al.*已於 1995 年提出了相關了演算法[19][20]來解決此一問題，以其有效的解決此類資料挖掘時所面臨的問題。而廣義性的關聯法則其最主要的目的，即為利用較高層次的項目組取代許多低層次的項目組，大幅減少項目的數量，以解決在資料挖掘時，項目組過多的問題。

2.3.2 負面性之關聯法則

之前我們所探討一般性關聯法則的推導，均屬於正面性相關法則的關聯，而只要不屬於正面性相關法則的所有規則，則全部都可以看成負面性的相關法則了。所以對於負面性的關聯法則可以說是非常的多，資料量也是相當的大。

Savasere *et al.*於 1998 年提出了相關的演算法[18]，其主要的方式即為找出廣義性的關聯法則。而在其所定義的負面性關聯法則中，其支持度亦必須滿足使用者所限制的最小門檻值，而可信度則由另一種方

式評估(RI, rule interest measure)，其值也必須滿足使用者的限制。而負面性關聯法則若是能與正面性關聯法則一起探討，兩者相輔相成，對使用者來說更能提供更多的數據資訊與潛在的規則，產生更大的效益。

2.3.3 Apriori 法則與應用

在之前所提出的 Apriori 演算法中，有一個所謂的 Apriori 法則(Apriori principle)，其最主要的意義就是限制每一個階層所產生的關聯性項目組之間的限制，其最主要的限制如下：

1. 如果某一項目組為非高頻項目組，則其包含於此一項目組的超集合亦為非高頻項目組。
2. 如果每一項目組為高頻項目組，則此一項目組的任一子集合應全部為高頻項目組。

利用 Apriori 法則可以限制項目組各個階層的對應關係，也是在 Apriori 演算法從某一階層的高頻項目組推導至下一階層的候選項目組所依循的定理。利用 Apriori principle 能確保每個階層的資料，均為實質對應的相關資料。以 Apriori 演算法的範例，其 Apriori principle 範例如下：

若下圖中 L_2 之項目組，利用 join 的方式產生 C_3，

{AC}{BC}→{ABC}，{AC}{CE}→{ACE}，{BC}{BE}→{BCE}，{BC}{CE}→{BCE}，{BE}{CE}→{BCE}，如此可以推導出{ABC}、{ACE}、{BCE}三組候選項目組，套用 Apriori principle 第二項法則，{ABC}若要成為高頻項目必須符合{AB}、{AC}、{BC}三項子集合均

為高頻項目的條件，才能進入候選項目集計算支持度與可性度, 但{AB}項目在L_2中為非高頻項目, 所以{ABC}項目組不能進入C_3, 同理, {ACE}項目也無法進入C_3。

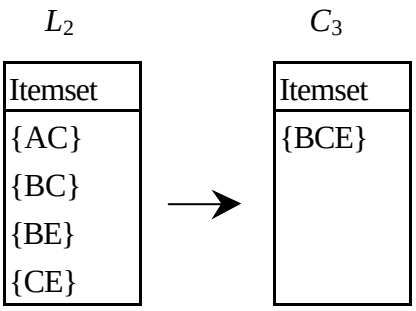

圖 2.13 Apriori Principle 推導範例

由因此證明 Apriori 演算法在產生更高階層的候選項目時，所使用的 apriori-gen()副函式是經由 Apriori principle 驗證過，確保每一階層的候選項目組是經過篩選過後的項目，無須重複驗證每一項目組的子集合均為高頻項目的限制。

2.3.4 儲存空間與即時線上挖掘

以 Apriori-like 的演算法則而言，原有的資料全部紀錄在原始交易資料庫中，當需要做挖掘的動作時，使用者給定各項初始值，再將資料庫中的資料套入演算法中，系統將計算各項目之間的支援度與可性度，最後，再將符合使用者設定的最小支援度的項目組提出，完成資料挖掘的計算過程。此一資料挖掘的動作在於需於某一特定時間，CPU 及 RAM 的使用量急遽上昇，系統在此時計算所需資訊，當資料量越多時，所需消耗的系統資源也越多，一次資料挖掘所需的時間也越長。

對於線上挖掘或是即時性資料挖掘而言，不論其套用任何的演算法，其最主要的方式即是將之前挖掘過的資料進行某種程度的處理與

紀錄，當新增的資料進入系統之後，系統僅需將新增的資料作一處理，再與原有的資料作一彙總的動作，即可求得使用者所需的線上資訊。此一系統運作的動作在資料一產生時，就即時計算處理，不需花費額外的資料挖掘時間，然而要將舊有的資料在挖掘過後，紀錄每個項目組之間的關聯性，則相對的需要花費許多的儲存空間。

以現在電腦硬體的記憶容量與 CPU 的運算速度快速增加的情形下，是要選擇線上挖掘亦或是批次挖掘所需資訊，則是使用者必須依照實際需要所考量的議題了。

第三章 動態資料挖掘演算法之研究

依據本文之前所提出的相關討論，我們不難發現利用 Apriori 演算法或是以其相關的演算方式在進行資料挖掘的時候，會造成以下問題：

1. 必須多次掃描資料庫
2. 在門檻值變動時，必須重新掃描資料庫
3. 資料變動時必須重新掃描資料庫
4. 無法即時取出高頻項目資料
5. 無法取出敏感性資料

為瞭解決以上的前四種問題，本章節中提出了一種名為 Multidimensional Update Mining(MUM)的演算方式，利用多維的陣列儲存即將挖掘的資料，並將挖掘過後的資料加以處理，待下次資料挖掘時再將之前的資料合併計算，以快速的得到需要的資訊。以下是 MUM 演算法的探討與實例。

第一節 Multidimensional Update Mining 演算法技術之探討

Apriori 演算法在最早提出的時候，是假設資料是靜態的，需要反覆掃描整體的資料庫直到找出使用者有興趣的相關法則。但是在現實生活中，交易資料經常為動態的新增，若是當每次新增資料的時候，就必須重新掃描一次原始資料庫以求取高頻項目組，則會浪費太多的時間。且隨者日漸增加的資料儲存量，資料挖掘的工作會變得負擔太重而沒有效率。

本文欲利用資料庫中的表格，產生多維陣列的資料結構方式，將 1-Itemset、2-Itemsets、….、n- Itemsets 建立一個多維陣列，以提供每一筆資料新增時，動態產生各項目組，紀錄相對應的資料項目，並將利用備用資料表(temp table)的觀念，將不符合支持度的項目組移至 temp table，以減少主資料表格(base table)的資料量，加快資料庫搜尋速度。

在資料挖掘最初階段，以多個表格來記錄大項目組的出現次數(count)值及可信度，若此一資料為目前主資料表格中沒有之資料，則動態新增此一項目，在主資料表格中搜尋不到所要尋找的項目組時，則可以轉換到備用表格搜尋，並記錄該項目組的資訊，以便下次新增資料時可以即時更改所需資料而不需要掃描整體資料庫，而在備用資料表中的項目組，相對應的資料新增的時候，其 count 的值還是會動態增加，當其支持度達到 base table 的支持度門檻值時，即可移入 base table，成為大項目組。如此既可以節省 I/O 及 CPU 運算時間，提高擷取速度，又不至於刪除日後具有潛力的項目組，對於目前動態新增的資料項目，提供一種十分良好的演算方法。

MUM 演算法其主要應用時機在於當交易資料新增至資料庫的當時，即時將資料拆解成不同的項目組，再儲存至對應的 MUM 資料表格中，以下為 MUM 演算法處理單一交易資料的過程：

step1：讀入目前資料項目。

step2：動態將資料分解成 1-Itemset、2-Itemsets、…、n-Itemsets，並掃描對應的 cube 中是否已經有資料存在。

step3：有資料：將對應資料 count 數加 1

無資料：新增項目組。

step4：計算此項目組的 support 是否達到目前的 minimum support。

a. 是，則繼續留在此 base table 中。

b. 否，則移到 temp table 中。

此一演算法雖然會記錄過多的項目組，但是在 CPU 的計算及硬體 I/O 的考量下，多維陣列的儲存方式，是可以有效提供較佳的搜尋速度，以便使用者快速尋找相關法則。而以目前硬體配備的價格日趨下降的趨勢來看，資料存放空間已經不再是一個嚴重的問題，且加上備用資料表的觀念，可以將備用資料表中，非高頻項目組加以處理或刪除，即可解決此一問題。

第二節 MUM 演算方法的推導過程

假設第一次新增的資料為原始的資料項目，其資料如下：

表 3.1 MUM 原始交易資料

Incremental database	
TID	Items
100	ACD
200	BCE
300	ABCE
400	ABE
500	ABE
600	ACD
700	BCDE
800	BCE

而在此一演算法中所提出的多維陣列，則是將每一層級的項目組(Itemsets)加以區分，分別存入不同的陣列中，而此處所提出由的每一個層級的項目組集合，都是一個資料表，如表二所示，相對於 1-Itemset 的資料項目組，就存放於 1-Itemset Table 表格中；2-Itemset 的資料項目組，就存放於 2-Itemset Table 表格中，以此類推。而由於存放的表格不

同，資料量相對的減少許多，加上備用資料表的觀念，實際儲存於主資料表，成為大項目組的資料量可以大大的減少，使得每一次新增交易時, 可以快速的將新增的項目逐一加入對應的表格中, 表 3.2 為 MUM 的資料表：

表 3.2 MUM 資料表

		3-Itemsets	Supports
	2-Itemsets	Supports	
1-Itemset	Support		

3-Itemsets Table

2-Itemsets Table

1-Itemset Table

而未通過最小支持度門檻值的項目組也必須加以紀錄，以備下次新增資料時彙總資料之用；而備用資料表的格式如同主資料表，其型式如表 3.3 所示 ：

表 3.3 MUM 備用資料表

		3-Itemsets	Supports
	2-Itemsets	Supports	
1-Itemset	Support		

3-Itemsets Table

2-Itemsets Table

1-Itemset Table

為了有效擷取出真正符合使用者所感興趣的資料，在資料挖掘時，對於每個階層的資料表格均可設定不同的支持度，彈性的調整 minute support 的數值，而資料挖掘方式如以下步驟，共分為兩個階段：

1. 將每一層的 Itemstes 定義不同的 minimum support，假定 1-Itemset 到 4-Itemsets 的 minimum support 分別為 50,20,10,5 等四個不同的門檻值。
2. 逐一將資料表格欄位中資料放入對應的資料表中，若是表格內無此項目組，則動態新增此一項目組；若已經存在時，則將其對應項目組的 Support 的值加一。並同步計算出其目前支持度的值。

以此種方式推導，直到有某一個項目的支持度經過計算不足其 minimum support 時，則將此一項目組將會移到 temp table 中。

表 3.4 是將 MUM 資料表依照其層級分解，紀錄此一交易資料表的每一個項目集合，並用行列轉換的方式，將各筆交易資料紀錄至表格中，左方的部份為資料庫中交易的資料項目組，上方為 1-itemset 的項目，而表格中的資訊則是其出現次數及支持度：

表 3.4 1-ItemSet 對應表

Pass		A	B	C	D	E
1	ACD	1/100	/	1/100	1/100	/
2	BCE	1/50	1/50	2/100	1/50	1/50
3	ABCE	2/66.6	2/66.6	3/100	1/33.3	2/66.6
4	ABE	3/75	3/75	3/75		3/75
5	ABE	4/80	4/80	3/60		4/80
6	ACD	5/83.3	4/66.6	4/66.6		4/66.6
7	BCDE	5/71.4	5/71.4	5/71.4		5/71.4
8	BCE	5/62.5	6/75	6/75		6/75

minimum support = 50%

請注意掃描過程中，其中項目{D}在 pass3 的過程中, 因為不足 50% 的 minimum support，所以被移入 temp table 中，但是在往後的 pass 過程中，其 count 數值還是會累加；若是在某次交易時，在 temp table 中項目{D}的 minimum support 值大於 50%的時候，依然會移入 base table 中。以此種方式，可以迅速的新增資料，不須重新掃描原始資料庫。

表 3.5 表示在處理此交易資料時，不足支持度的資料被移入 temp table 的情形。

表 3.5 備用資料表

pass	Items	Support
1	/	/
2	/	/
3	D	33.3
4	D	25
5	D	20
6	D	33.3
7	D	42.8
8	D	37.5

在表 3.4 的 pass3 的過程中，項目組{D}被移出 base table，此時備用資料表就建立一個{D}的項目，並將其 count 的值紀錄起來，往後的資料新增時，其 support 的數值依舊會即時的做異動，若是其支持度滿足預設的 50%時，再將其項目組{D}移入 base table 中。

對於 2-Itemset、3-Itemset、….、n-Itemsets 的資料表，都是依照上述的處理流程，將各項目逐一置放到對應的表格中，以下是各表格中資料放置的情形：

表 3.6 **為** 2-Itemset **的演算過程：**

表 3.6 2-Itemset **對應表**

pass		AB	AC	AD	AE	BC	BD	BE	CD	CE	DE
1	ACD		1	1					1		
2	BCE					1		1		1	
3	ABCE	1	2		1						
4	ABE	2			2			2			
5	ABE	3			3			3			
6	ACD		2	2					2		
7	BCDE					2	1	4	3	2	1
8	BCE					3				3	

minimum support = 20%

表 3.7 **為** 3-Itemset **的演算過程：**

表 3.7 3-Itemset **對應表**

pass		ABC	ACD	ABD	ABE	BCD	BCE	CDE
1	ACD		1					
2	BCE						1	
3	ABCE	1			1		2	
4	ABE				2			
5	ABE				3			
6	ACD		2					
7	BCDE					1	3	1
8	BCE						4	

minimum support = 10%

表 3.8 **為** 4-Itemset **的演算過程：**

表 3.8 4-Itemset **對應表**

		ABCD	ABCE	ABDE	ACDE	BCDE
pass 1	ACD					
pass 2	BCE					
pass 3	ABCE		1			
pass 4	ABE					
pass 5	ABE					
pass 6	ACD					
pass 7	BCDE					1
pass 8	BCE					

minimum support = 5%

由此一方式可以快速的處理新增的交易，而不須將舊有的原始資料表重新掃描，大大的增進了資料挖掘的速度。

第三節 MUM 演算法之實作研究

MUM 程式在實作的部分，有兩個主要的步驟，如圖所示：

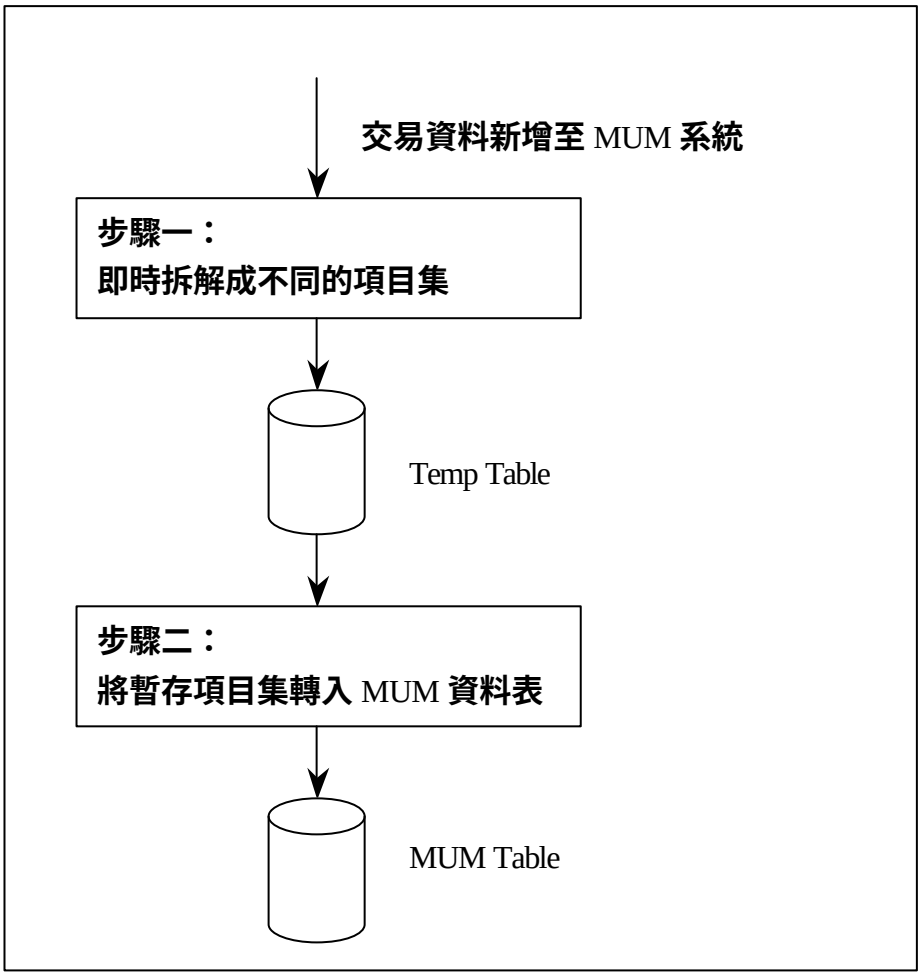

圖 3-1 MUM 演算法步驟示意圖

當一筆新增的交易資料進入處理 MUM 系統中處理時，新增資料啟動觸發程式，將交易資料即時拆解成不同的項目集合，假設交易資料為{A，B，C，D}，則拆解項目如下所示：

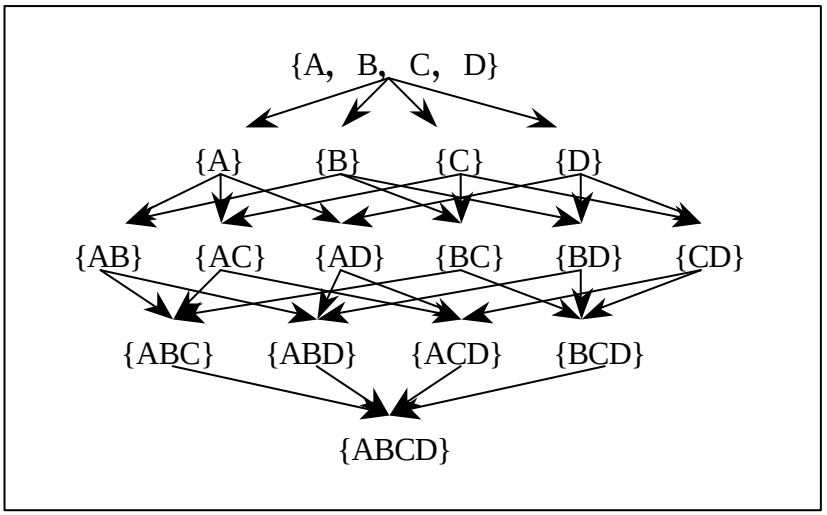

圖 3.2 **交易項目拆解樹狀圖**

設新增交易資料項目為 n **項，暫存資料表為** T**，一次交易的所有項目為** I**，每次拆解的項次為** k**，類似** Apriori **演算法由** T_k **聯集的方式，產生下一階層的項目，**MUM **產生拆解項目集的虛擬碼如下所示：**

```
T1 = split(I)
for (i=1;i<n;i++) do
begin
        insert into Ti+1
        select p.item1,p.item2,…,p.itemk,q.itemk
        from Ti p,Ti q
        where
p.item1=q.item1,…,p.itemk-1=q.itemk-1,p.itemk<q.itemk
end
```

圖 3.3MUM_gen **虛擬碼**

待上述的拆解資料進入暫存資料表後，再將暫存資料表中資料加入 MUM **資料表，其** MUM **演算法的虛擬碼程式如下：**

```
MUM_gen(I)
forall tempdata i∈ T do
begin
    if not exist(select item from MUM where item=i) do
        insert into MUM values(i,1)
    else
        update MUM set count=count+1
end
```

圖 3.4 MUM **演算法虛擬碼**

以此一方式處理單一筆資料，待下一次交易資料進入 MUM **系統時，再次觸發此一程序，進行資料挖掘的工作，如此資料庫中即可即時的累積所有的交易資料，當使用者需要設定門檻值取出高頻項目集時，僅需將** MUM **資料表中超過門檻值的項目集取出，即可完成** MUM **演算法的資料擷取動作。**

第四節 MUM 演算法與 Apriori 演算法之討論與比較

MUM **演算法與** Apriori **演算法最主要的不同點有兩以下兩個部分:**

1. **對於交易資料的處理，是直接由資料的第一筆向下逐一處理到最後一筆，將交易資料的出現次數直接加總，支持度及可信度的計算由整體資料的筆數與項目組個別計算，無須反覆的掃描資料庫以求取各階層項目組的相關資訊。**
2. **對於新增的資料而言，無須重新掃描整個資料庫，僅需要對新增的資料加以處理，再彙總之前處理過的資料，即可對所有的**

關聯項目進行計算相關資訊。

假設資料庫中的交易有 N 筆資料，而所有交易的資料中全部有 l 個項目，新增的交易有 m 筆資料，N 遠大於 m；一般以 Apriori 為主靜態資料挖掘的演算法必須在每次新增資料的時候，重新掃描包含新增資料的整體資料庫，而 MUM 演算法則僅需對新增資料做處理，再彙總原有的資料即可。

若是計算一般演算法的時間複雜度，其表示公式為：

$$\Omega\,(l^2(N+m)) \tag{1}$$

而對於 MUM 演算法的時間複雜度來說，可以表示為：

$$\theta\,(N+m) \tag{2}$$

從這兩個公式可以看出，類似 Apriori 的各演算法在資料庫中交易資料數量越多的時候，其花費的時間複雜度會隨著資料量 l 的大小做指數的增加，而由於 MUM 演算法處理資料的方式為單次資料處理，所以時間複雜度不會隨著整體資料量的變化而有所變動，且 MUM 演算法的處理時間在交易當時即時處理，所以無須花費一段額外的資料處理時間，由此可以得知 MUM 較一般演算法在處理資料挖掘時之優點。

以下為一般演算法及 MUM 演算法對於每次新增交易資料時的處理，所耗費時間之比較示意圖：

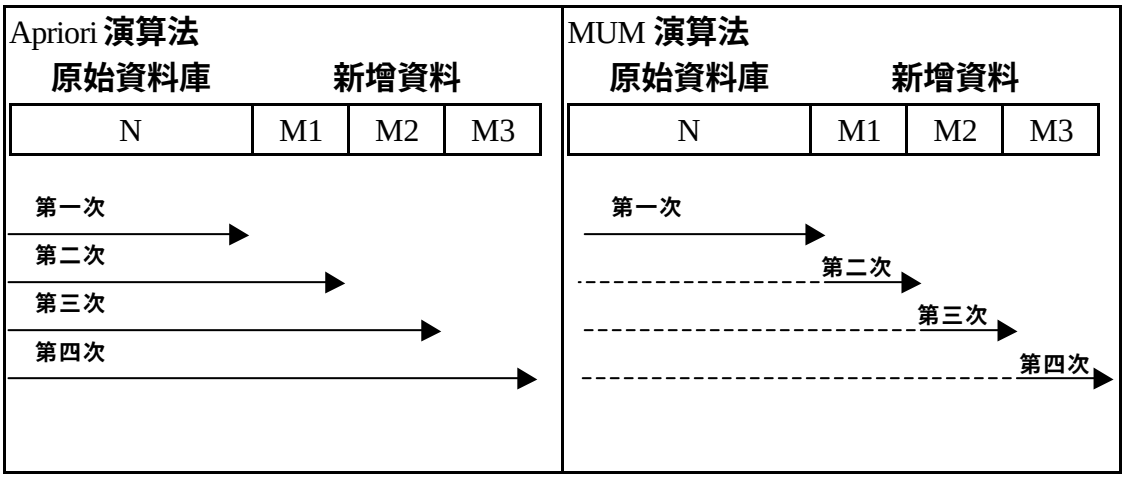

圖 3.5 MUM **演算法與** Apriori **演算法之比較**

圖 3.1 **中虛線部分即為節省掃描資料庫的時間成本，我們可明顯的看出其** MUM **演算法在新增資料時較** Apriori **演算法節省許多重複搜尋資料的時間，但是** MUM **在儲存的空間上，卻會花費比** Apriori **演算法大上數倍的存放空間，算是一種以儲存空間換取資料挖掘時間的方式。**

在第五章中，本研究會針對上述之演算法提供理論上的分析及實驗數據，實際比較 MUM **演算法與** Apriori **演算法的差異性與優缺點，提供後續研究的學者參考。**

第四章 敏感性資料挖掘演算法之研究

以目前學界對於資料挖掘的技術來說，目前僅有 FUP 演算法對於新增交易的資料及有相關挖掘方式的研究，但其資料處理的方式，卻是將其挖掘的新增資料相關法則，與原有的相關法則合併計算，無法提供新增資料中敏感性資料的相關數據。

以零售業為例，新式的熱門商品剛一上架，極有可能就成為一股搶購的熱潮；或是一個新興的客戶短時間之內採購大量的商品，而應該被列入具開發潛力的客戶群之一。這些都會導致整體資料庫中相關法則的推導有所改變。但是對於一間經年累月的零售業或是 CRM 開發的資料庫來說，滑板車的銷售量或是新客戶的消費金額，想要達到一定的門檻值，所要累積的量一定需要相當時日才能達到使用者門檻值得限制，但熱門商品不能長久轟動、潛力客戶也必須即時鞏固開發，因此開發一種對於新增交易資料中，敏感度較高的資料項目，是一件十分重要的議題，為此，本篇論文嘗試著改進 FUP 演算法，將新增交易項目中敏感性資料從中挖掘出來，提醒使用者必須著重的目標及提供使用者在決策時參考的依據。

本章中提出一種名為 SUM(sensitivity update mining)的資料挖掘方式，此一演算方式為一種改進 FUP 演算法的資料挖掘方式，將 FUP 演算法每次挖掘的資料批次處理，並做一節錄的動作，並結合 MUM 演算法的優點，增加其資料挖掘速度，再將其節錄的資料與之前資料合併、比對，從資料中挖掘出即時的敏感性相關法則。而對於節錄的相關法則，加以處理，即可抽取出新增資料集中敏感度較高的資料。此種資料挖掘技術對於目前動態新增的資料庫挖掘，提供了一個有效率

的動態即時敏感性資料挖掘方式，避免敏感性資料因為無法在短期間內達到使用者設定的最小支援度而被捨棄的一種演算方式。

第一節 SUM 演算法技術之探討

結合第三章 MUM 演算法的資料挖掘方式及資料結構，SUM 演算法將延用 MUM 演算法的計算方式，且沿用其 MUM 演算法之多維陣列之表格架構，用來計算 SUM 演算法所需之來源資料。

在 SUM 演算法中，系統中可以設定一段固定的時間 t 當作 SUM 資料挖掘週期，在週期內將 MUM 演算法得到的挖掘資料表格做一彙總，並與上一個週期的資料做比較，即可以得到最新資料與舊有資料的關連。

為能精確的表示最新資料的變化量，SUM 演算法定義了兩種敏感度指數，其中一階敏感度指數(SUM level-1)為 MUM 演算法在週期 t 與 t-1 時間中每個項目組的關聯，若同一項目在兩時期的數值相差超過使用者設定之門檻值，也就是一階敏感性指標(diff-1)，則稱此一項目為敏感性資料。其二為二街敏感度指數(SUM level-2)，其計算方式為將連續三個週期(t、t-1、t-2)中的資料取出，三筆資料相加後取絕對值，若是能符合使用者設定之二階敏感度指標(diff-2)，此項目亦為敏感性資料。以下為 SUM 演算法推導一階敏感性資料的過程，假設原始資料庫中已有 1000 筆資料，而在週期 t 時新增的資料項目如表 4.1 所示

表 4.1 SUM 演算法新增資料

TID	Items
8100	ACD
8200	BCE
8300	ABCE
8400	ABE
8500	ABE
8600	ACD
8700	BCDE
8800	BCE

以 MUM 演算法批次處理此八項交易資料，且設定 1-Itemset 的 minimum support 為 50%、2-Itemset 的 minimum support 為 30%、3-Itemset minimum support 為 10%，可以得到以下的關聯法則：

表 4.2 SUM 演算法 1-itemset 資料

1-Itemset	Count/Support
A	5/0.625
B	6/0.75
C	6/0.75
D	3/0.375
E	6/0.75

其中{D}項目組因為不滿足 ms>＝50%的設定，所以移入 temp table 中，其餘的項目為高頻項目組。

以下為 2-Itemset 的項目組：

表 4.3 SUM **演算法** 2-itemset **資料**

2-Itemset	Count/Support
AB	3/0.375
AC	3/0.375
AD	2/0.25
AE	3/0.375
BC	4/0.5
BD	1/0.125
BE	6/0.75
CD	3/0.375
CE	4/0.5
DE	1/0.125

若將其相對應的項目從之前的相關法則資料庫中取出，當作 t-1 **的週期資料，如表** 4.4 **所示**

表 4.4 SUM **演算法原始** 2-itemset **資料**

2-Itemset	Count/Support
AB	430/0.43
AC	257/0.257
AD	382/0.382
AE	413/0.413
BC	135/0.135
BD	407/0.417
BE	389/0.389
CD	435/0.435
CE	512/0.512
DE	207/0.207

SUM level-1 **敏感性指標** diff **之計算方式為：**

$$diff = | \text{support of item}_t - \text{support of item}_{t-1} | \quad (4.1)$$

以公式 4.1 比較每一比資料，若是將 t 週期中批次處理的資料，與之前 t-1 週期資料庫中資料做一比較，並設定第一階敏感性指標(diff-1)為 25%，可以得到下表

表 4.5 SUM 一階敏感度資料

2-Itemset	diff
AB	-0.055
AC	0.118
AD	-0.132
AE	-0.038
BC	0.365
BD	-0.292
BE	0.361
CD	-0.06
CE	-0.012
DE	-0.082

而 SUM level-1 的符合條件即為 diff 值大於使用者所設定之 diff-1 的值，亦為 SUM level-1 之計算公式為：

$$\text{SUM level-1} = \text{diff} > \text{diff-1} \tag{4.2}$$

其中{BC},{BD},{BE}三個項目組的變化率 diff 的絕對值均大於 25%，亦即表示此次批次作業中，此三項為敏感性資料，而正值代表相關法則中的重要性增加，負值則是代表遞減。如在超商為例，正值代表此相關商品的銷售量增加，反之則是減少。

SUM level-1 主要是將此次的交易資料與上一次的資料作一比較，如此可以直接得到變化量較劇烈的項目組，但某些項目的變化量為逐次遞增，而每次遞增的量無法即時達到使用者設定之一階敏感度指

標，如此還是無法成為敏感性資料，而被使用者忽略。

SUM level-2 的計算方式即為將連續三次週期中的 diff 值綜合比較，並重新定義 diff-2 的門檻值，若是三次週期中的差異度 diff 值相加的絕對值超過 diff-2 的門檻，則項目亦為敏感性資料，也可以提供使用者參考。

SUM level-2 的 diff-2 的計算方式為：

$$\text{diff-2} = |\text{diff}_t + \text{diff}_{t-1} + \text{diff}_{t-2}| \quad (4.3)$$

如公式 4.3 所示，並設定第二階敏感度指標為 30%，若將三次連續的差異值加總，計算的結果能符合第二階敏感度指標，亦可找出使用者感興趣的敏感性資料：

表 4.6 SUM 二階敏感度資料

Itemset	Diff_t	Diff_{t-1}	Diff_{t-2}
AB	0.055	-0.203	0.134
AC	-0.118	-0.129	-0.139
AD	0.132	0.147	0.168
AE	0.038	0.093	-0.017
BC	-0.365	-0.218	-0.045
BD	0.292	0.121	0.034
BE	-0.361	0.173	0.061
CD	0.06	0.074	-0.034
CE	0.012	0.189	0.186
DE	0.082	0.154	-0.097

而 SUM level-2 的符合條件即為 diff 值相加絕對值大於使用者所設定之 diff-2 的值，亦為 SUM level-2 之計算公式為：

$$\text{SUM level-2} = \text{diff} > \text{diff-2} \quad (4.4)$$

若觀察每個批次作業的 diff 值，或許能找出當次的異動率較大的敏感性資料，但對於逐漸增加或是減少的資料則較難察覺。以上表為例，{AD},{CE}項目組每次的 diff 值都沒有達到 25%的水準，但是其差異值均為正值，且明顯有上升的現象，計算三次增加的總和分別為為 44.7%、38.7%，大於預設的 30%的敏感度指標，亦即{AD}、{CE}等項目亦為使用者感興趣的敏感資料。

第五章 實驗結果與探討

本章節中主要對於 MUM 演算法與 Apriori 演算法，提出程式執行結果的實驗數據與事後的分析與探討，提供 MUM 演算法客觀的數據比較。

第一節 實驗架構與實驗數據

本實驗共撰寫了兩個演算法的程式，做為相互表較的參考，其一為 Apriori 演算法，此一演算法程式在 Microsoft SQL Server 7.0 的環境執行，程式碼以預儲程序完成，所以執行效率相當良好。但在預儲程序中無法使用陣列的處理方式，故 MUM 演算法以 Visual Basic 6 撰寫，執行效率會因為程式與資料庫連線與截止連線時，花費太多時間，故在前處理時，耗費太多時間，但本實驗主要驗證 Apriori 演算法與 MUM 演算法在動態新增資料時，以及當支持度調整時，對於資料挖掘效率的比較，所以時間的比較上，僅以相對的數據做為比較的參考。

以下為本次實驗時的外部環境

1. 硬體架構

本次實驗的兩種演算法，均在以下的硬體環境中執行。

CPU：Authon 1G

RAM：512M

HD：IBM 40G

2. 軟體環境

Windows 2000 Professional with sp2

SQL 7.0 with sp2

Visual Basic 6.0

本研究實驗的設計方式如下

1．資料來源：

本實驗的資料來源主要為提供 Apriori 演算法與 MUM 演算法做為資料挖掘的資料來源。本實驗利用 VB 程式產生亂數寫入資料庫，做為資料挖掘的資料來源。由於實驗目的在於驗證兩種演算法的資料挖掘速度，故不涉及資料挖掘的準確性問題。

產生資料時需設定三個參數，一為產生資料的筆數，二為每筆資料最多的項目數，三為每個項目的變化數。以此三種參數來模擬現實生活中的交易資料。

2. 程式設計：

在 Apriori 的部分，利用 Microsoft SQL 7.0 中的預儲程序完成，由於利用 SQL 內部的預儲程序，將使 Apriori 演算法的程式不需要與外界程式有資料庫連線的負擔，增快資料挖掘的速度。

MUM 的程式方面，由於需要資料拆解的程序，而 SQL 中不支援陣列的使用，所以 MUM 演算法利用 VB 撰寫，在效率上會比預儲程序慢很多。

3.實驗數據

首先比較的是動態新增資料庫時的比較，設 Apriori 演算法的 ms=100。

表 5.1 動態新增資料庫實驗數據比較表

	原始資料 10000 筆	新增 100 筆 10100 筆	新增 100 筆 10200 筆	新增 100 筆 10300 筆
Apriori 演算法	16:17	16:45	17:16	17:56
MUM 演算法	2:37:43	2:12	2:14	2:12

其次是線上資料挖掘時，動態調整最小支持度時的資料挖掘，資料筆數均為 10000 筆。

表 5.2 線上挖掘時實驗數據比較表

	ms=150	ms=125	ms=100	ms=75
Apriori 演算法	10:17	10:37	16:17	35:59
MUM 演算法	2 sec	2 sec	2 sec	2 sec

第二節 實驗結果探討

MUM 演算法由於是以 VB 開發，所以在第一次挖掘的時候，需要花費很多時間在 VB 程式與資料庫的連線上，且資料量大時，連線時間將花費更多時間。

從表 5.1 中可以明顯的看出 Apriori 演算法在動態新增資料庫時，每次新增資料均需重新挖掘所有資料庫中的資料，所以資料挖掘的時間是一直增加，反觀 MUM 演算法僅需處理新增資料的部分，待新增資料處理完畢再與舊有資料彙總，即可完成資料挖掘的工作。

表 5.2 中所比較的在於當線上即時挖掘時，需要調整支持度時，MUM 演算法與 Apriori 演算法在資料挖掘所花的時間比較。此處假設資料挖掘分為兩個階段：

第一階段：資料挖掘演算法執行的時間

第二階段：將高於門檻值的項目組取出的時間

由於 Apriori 演算法在支持度變動的時候，需要重新挖掘整體的資料庫，再將符合的條件列出，也就是在支持度變動時，兩個階段的工作均需重新執行，所耗費的時間也就相對提高。但 MUM 演算法由於事先儲存所有資料，所以在改變支持度的時候，僅需在查詢與法中設定支持度，做第二階段查詢的動作，將符合條件的資料列出即可，無須重新挖掘整體資料庫。

由以上兩組實驗數據可以明顯的看出 MUM 演算法的優勢，而在本文中所提出的 MUM 演算法，在面對現實生活中的資料挖掘議題，由於交易資料是隨時間逐漸遞增的，所以事實上 MUM 演算法無須同一時間集中處理大筆資料，資料挖掘第一階段的工作事實上在交易的同時即已完成，這也是 MUM 演算法最主要的優勢之一。

第三節 MUM 演算法之後續探討

MUM 演算法事實上是一種利用硬碟空間換取資料挖掘時所花費的時間，對於挖掘過之後的資料均記錄在資料庫中，可以產生以下的效益：

1. 即時產生高頻項目組
2. 線上即時動態資料挖掘
3. 資料挖掘時點分散，無須額外花費時間做資料挖掘的動作
4. 可以設定每一階層的支持度，而不會導致 Apriori principle 的錯誤。

而 MUM 演算法與 FUP 演算法均為動態資料挖掘的演算方式，兩者設計的最主要目的均為解決動態資料新增的問題，而兩者相同的地

方在於處理新增交易資料的方式，均為保留已挖掘過的資訊，待新增資料時將新增資料的部分處理完畢，與舊有資料彙總後，產生高頻項目。兩者不同之處在於 FUP 演算法為批次處理一整批的資料，再與原始資料彙總，而對於單一次批次處理而言，演算方式亦為 Apriori 演算法，且在新增資料項目沒有保留在舊有資料中時，亦必須掃描整體的原始資料庫，以求取高頻項目組。而 MUM 演算法是將原始交易資料分別存放在各個不同的陣列中，新增資料無論如何變換，MUM 演算法均無須重新掃描原始資料庫；若交易資料沒有涵蓋交易者身分資料等欄位時，MUM 演算法亦可將交易資料處理至 MUM 資料表後，將原始交易資料刪除，以節省硬體存放空間，減少硬碟的空間使用量。

無可諱言的，MUM 演算法在項目組的產生時，無論其支援度與可性度為何，均記錄至關聯項目資料表中，此一方式會產生過多的非高頻項目組，佔用大量資料庫空間，在資料量及搜尋時間上都會產生負面的效益，而以下幾種方式可以解決此一問題：

1. 採用 Pre-large Itemsets 的方式，在高頻項目組與非高頻項目組間設定一個預高頻項目組，作為一個有效的緩衝，解決每一個項目組均需搜尋整體資料庫，產生效率不佳的主因。(詳如第二章 2.2.2 中，Per-large Itemset 的介紹)
2. 搜尋效率不佳的情形，可善用資料庫中的索引，且 MUM 資料表格的資料型態需設定為 char 型態，固定其資料欄位大小，較能快速搜尋出所需之資料項。
3. 使用”備用資料表”的觀念，將不符合使用者設定之最小支持度的項目組移入備用資料表中，以有效的在主資料表中搜尋及紀

錄相對應的項目關聯，但會產生資料移入移出的時間點的議題：

a. 資料新增時即時掃描整個資料表格，除交易項目組外，將所有的項目組依照設定的門檻值做主資料表與備用資料表的移入與移出作業。此一方式會使掃描資料表的次數增多，但相對的自每次新增資料時， 所需處理的資料項目較少。
b. 設定一個特定的時間點，啟動掃描整體資料表格，將項目組從主資料表格與備用資料表中依照設定的條件移入與移出。此一方式需考慮資料庫負載的時段，訂定良好的排程。
c. 資料新增時，僅對相對應產生的項目組做處理，依照設定的門檻值做主資料表與備用資料表間的移入與移出。此一做法在單次交易時計算量最少，但在掃描資料庫時卻會掃描較大的資料表格。

第六章 討論與結論

第一節 結論

相對於以往電腦的使用情形，以及資料的紀錄量，目前的電腦可以說已經深入各行各業之中。不論是大小型的企業、醫院、學術領域、亦或是銀行業，電信業…等，資料的儲存已經由人工登錄的方式轉成電子檔案的格式，甚或是資料庫的型式。相較於之前的紀錄方式，現在以電腦為儲存媒介的優勢以廣為人們所知。

本文中我們探討了兩個新的動態資料挖掘的解決方式，MUM 演算法與 SUM 演算法，此兩種演算法最主要的目的在於希望解決一般資料挖掘時所遇到的困難。對於 MUM 演算法，是利用即時拆解交易資料，直接與舊有資訊合併計算計數值，並記錄到多維陣列的資料庫，以此方式解決動態新增資料庫的問題。而 SUM 演算法提出了敏感性資料挖掘的方式，企圖在交易資料庫中挖掘敏感性資料，以避免近期的交易資料無法在短時間內達到整個交易資料庫最小支援度的限制，而被使用者忽略的問題。本文中所提出之兩種演算法可以解決以下的問題：

1. MUM 演算法可以解決 Apriori 演算法反覆搜尋資料庫，在計算支持度與可信度時效率不佳的問題。
2. MUM 演算法的演算過程所產生的關聯法則，在使用者動態變更支持度門檻值時，可以避免一般演算法在 Apriori 法則所遇到的限制。
3. MUM 演算法可以有效解決在資料動態新增時，必須重新掃描整體資料庫的缺點，大幅提昇舊有演算法的效能。

4. MUM 演算法能提供即時線上資料挖掘的資訊，當使用者需要資料挖掘的結果時，只要將關聯項目的計數值與交易總次數直接計算，即可得到支持度，無須等待漫長的資料挖掘時間。
5. SUM 演算提供了一個挖掘敏感性資訊的演算方式，此一演算方式可以挖掘出近期的敏感性資料，避免舊有的演算法在資料量項累積到足夠的次數，才能成為高頻項目為使用者發覺的盲點。

比較以往的演算方式在動態資料挖掘的領域，均需考量計算時間與儲存空間的取捨問題。如果將越完整的資訊儲存在資料庫中，新增的交易資料進入系統計算時，所需花費的計算時間就越少。相對的如果為了減少紀錄的空間，遺失部分紀錄資訊，而新增的交易資料在處理時，如果項目組相關資訊沒有在紀錄的資料中，就只有重新掃描原始資料庫以求出完整的規則了。

關於文中的理論架構，在第五章中亦有實驗數據做為比較，由實驗的數據中可以驗證本研究中所提出之 MUM 演算法確實比類似 Apriori 的各種演算法在挖掘的效率上有明顯的提昇，且在資料動態新增及線上挖掘時，即時改變支持度的情形下，均有非常優異的運算效能表現。實際的執行程式後，可以發現 Apriori 演算法在新增資料量越來越多時，挖掘的時間呈現指數倍的成長，而 MUM 演算法由於只是處理單一筆資料，所以在同時處理大量資料時，時間是呈線性的成長，相信這是動態新增資料時需要考慮的重要指標。

第二節 未來研究之方向

本文所提出的 MUM 演算法雖然在理論上具有十分良好的演算方

式以解決 Apriori 演算法的缺點，但在關聯法則產生的項目組處理上卻有著資料量過多的致命缺點，而這也是我們需要改進的思考空間，因此，如何在計算時間與儲存空間上得到一個良好的均衡，將是未來研究的一個重要的方向，除此之外，未來的研究將可朝著以下的方向進行：

1. MUM 演算法在主要資料表移入被用資料表，或是備用資料表移入主要資料表的最佳策略。
2. MUM 演算法因為需要達到即時挖掘功能，必須紀錄許多額外的項目組，浪費許多記憶空間。再者，在搜尋的時候，也會因為項目組過多而需費時許久。如何增進挖掘時的效能。
3. 以最小完美雜湊樹的演算方式導入 MUM 演算法中，將可以直接對項目組定址，節省資料搜尋與更新的時間。
4. 以統計學或是其他相關學科的理論導入 SUM 演算法則中，以期求得敏感性資料的最佳策略。

以往的企業經營策略總是認為"擁有越多客戶資料在行銷上越佔優勢"，然而在盛行的網路行銷泡沫化之後，此種論調逐漸被淘汰。此時使用者開始真正的去思考：什麼才是有用的資料？哪些人是具有潛力開發的客戶？客戶所感興趣的是哪些服務或商品？如何利用手上的資料，找出決策或是經營的模式？於是今入在企業間所廣泛討論的，類似客戶資源管理、系絡行銷等相關議題已漸漸的被企業主所接受，也願意投入大量的經費與人力從事此方面的研究與發展。企業利用本身的交易資料作為輸入，而得到的結果是能夠被企業使用的關鍵資訊，其中的處理過程，就是將關鍵性資訊挖掘出來的方式，而如何從眾多

的資料中，快速且正確的找出使用者所感興趣的資料，提供使用者做決策時有一個系統化的參考依據，而不是靠使用者主觀的感覺去做判斷，此一系統化的方式所產生的資料，將是使用者所需要的關鍵性資訊，也是我們研究的方向。

隨著現實企業中交易資料日漸增多的情形來看，以往將資料庫假設為靜態資料的資料挖掘演算法已不能滿足現實的需要，所以動態資料庫挖掘的方式以成為現行的趨勢。而要達到動態資料挖掘，最直覺的想法就是保留挖掘過後的資訊，將其紀錄在資料庫中，待動態新增的資料進入時，直接挖掘新增的部分，並與舊有資料合併計算，方能達到動態資料挖掘的需求。想要做到線上即時挖掘的功能，就必須犧牲硬碟空間；若想節省硬碟空間，就必須浪費許多資料挖掘的時間，而資料在記憶體中運算的速度是硬碟讀取速度的一千倍，所以在記憶體中運算完後，再行存放到硬碟中似乎是一個不錯的方式，然而如何預先得知挖掘的資料大小，切割有效的記憶體空間，這些都是未來思考的方向。

參考文獻

1. M. S. Chen, J. Han, and P.S. Yu, "Data Mining: An Overview from a database Perspective", IEEE Transaction on Knowledge and Data Engineering, Vol. 8,No. 6,December 1996.
2. L. Kaufman and P. J. Rousseeuw, "Finding Groups in Data: an Introduction to Cluster Analysis", John Wiley & Sons, 1990.
3. S. M. Weiss and C. A. Kulikowski, "Computer System that Learn: Classification and Prediction Method from Statistics, Neural Nets, Machine Learning, and Expert System", Morgan Kaufman,1991.
4. R. Agrawal and R. Srikant, "Mining Sequential Patterns", IEEE ICDE, pp. 3-14, 1995.
5. R. Agrawal and R. Srikant, "Fast algorithm for mining association rules", the International Conference on Very Large Database, pp. 487-499, 1994.
6. R. Agrawal, R. Srikant, and Q. Vu, "Mining association rules with item constraints", in 3th International Conference on Knowledge Discovery in Database and Data Mining, Newport Beach, California, 1997.
7. M. S. Chen, J. S. Park, and P. S. Yu, "Efficient Data Mining for Path Traversal Patterns", IEEE Transactions on Knoeledge and Data Engineering, Vol. 10, No. 2, 1998.
8. R. Agrawal and R. Srikant, "Fast Algorithm for Mining Assoication Rule in Large Databases", In Proc. 1994 Int'l Conf. VLDB, pp. 487-499, Santiago, Chile, Sep. 1994.
9. Savasere, E. Omiecinski, and S. Navathe, "An Efficient Algorithm for Mining Association Rule in Large Database", Proc. 21th VLDB,pp. 432-444, 1995.
10. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama, "Mining optimized association rules for numeric attributes", the ACM SIGACT–SIGMOD -SIGART Symposium on Principles of Database Systems, pp. 182-191,1996.
11. D. W. Cheung, S. D. Lee, and B. Kao, "A general incremental technique for maintaining discovered association rules", in Proceedings of Database Systems for Advanced Applications, DASFAA'97, Melbourne, Australia, pp. 185-194, 1997.
12. H. Toivonen, "Sampling Large Database for association Rule", VLDB, pp. 134-145, 1996.
13. 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.
14. S. Brin, R. Motwani, J. D. Ullman, and S. Tsur, "Dynamic Itemset Counting and Implication Rules for Marketing Basket Data", 1997 ACM SIGMOD Conference

on Management of Data, pp. 255-264, 1997.

15. B. Dunkel and N. Soparkar, "Data Organization and Access for Efficient Data Mining", ICDE 1999, Sydney, Australia.
16. D.Kenny and J. F. Marshall, "Contextal Marketing", Harvard Business Review, Nov-Dec, 2000.
17. R. Srikant and R. Agrawal, "Mining Generalized Association Rules", Proc. Of the 21st Int'l Conference on VLDB, pp. 407-419, Zurich, Switzerland, Sep 1995.
18. A. Savasere, E. Omiecinski and S. Navathe, "Mining for Strong Negative Associations in a Large Database of Customer Transactions", In Proc. of the IEEE 14th int. Conference. On Data Engineering, Orlando, FL, Feb 1998(forthcoming).
19. **王慶堯，民國 89 年，利用準大項目集之漸進式挖掘，義守大學資訊工程研究所碩士論文。**
20. **許智豪，民國 89 年，在動態資料庫中作線上挖掘關聯式法則，國立中興大學資訊科學研究所碩士論文。**