在動態資料庫中挖掘關聯法則之快速演算法
A Fast Algorithm for mining Association Rules in Dynamic Databases

蘇建源
南華大學資訊管理所
jienyien@mail2000.com.tw

張晉赫
南華大學資訊管理所
g1141014@mail2.nhu.edu.tw

邱宏彬
南華大學資訊管理所
hpchiu@mail.nhu.edu.tw

摘　　要

資料探勘(data mining)**為近年來廣泛地應用在客戶資源管理、行銷、醫學及其他許多領域中的一門學科。如何有效率地從大量的資料中搜尋出隱含的資訊與有用的規則，一直是關聯法則探勘研究領域中十分重視的課題。在關聯法則的挖掘上，最具代表性的方法是** Apriori **演算法，其牽涉到多次資料庫掃描以取得大項目組，因此，對於現實生活中的動態新增資料庫而言，**Apriori **演算法面臨了三個主要的問題，其一為該演算法相當耗時，不適合線上探勘；二為無法有效解決動態新增資料庫中漸進式探勘的問題，第三為相對於整體資料庫，較新時間點的敏感性資料無法被有效的擷取。**

本文提出一簡單的關聯法則之多層更新挖掘法(Multilayer Update Miner, MUM**)，此方法不需要重複掃描原始資料庫，因此，可支援線上探勘及漸進式探勘的需求。利用** MUM **多層同步處理與更新的特性，搭配敏感度指數的定義，**MUM **可以被用來挖掘對決策者有用的即時性敏感資訊。同時，本文藉由許多實驗及相關的分析以檢驗** MUM **的執行效能並探討其優缺點與可行性。**

關鍵字：資料探勘、關聯法則、線上探勘、漸進式探勘、敏感性分析。

Abstract

Data mining is the exploration and analysis of large quantities of data in order to discover meaningful patterns and rules. It is an important discipline, which has widely applied in fields ranging from customer relationship management to marketing, and medicine.

The discovery of association rules is an important task in data mining. The Apriori algorithm is the most popularly and widely used technique for mining association rules. However, the Apriori algorithm must scan the database many times to discover the large itemsets so that it has three main disadvantages: (1) it is time-consuming; (2) it is not suitable for mining of incrementally growing databases due to the need of rescanning the original databases; and (3) as databases grow, the sensitive information in the new transactions can be not mined effectively.

In this paper, we propose the Multilayer Update Miner (MUM) algorithm, which does not need to rescan the original database, to mine the association rules for the incrementally growing databases. Based on two our designed sensitivity indexes, the MUM can mine sensitive information from the newly inserted transaction. Many experiments and related analyses are conducted to validate our proposed approaches.

Keywords: data mining, association rules, dynamic databases, online mining, incremental mining, sensitive information

壹、緒論

如何有效率地從大量的資料中搜尋出隱含的資訊與有用的規則，一直是關聯法則探勘研究領域中十分重視的課題[1-10]。在關聯法則的挖掘上，最具代表性的方法是由 Agrawal et. al.所提出的 Apriori 演算法[7]。Apriori 演算法中主要包含了以下兩個主要的步驟：

(1) 反覆的產生候選項目組（candidate itemset）並搜尋整個資料庫，直到找出所有的大項目組(large itemset)。
(2) 利用(1)所找出的大項目組，推導出所有的關聯法則。

在步驟(1)中，候選項目組的支持度必須大於或是等於使用者所設定的最小支持度門檻值，才能成為大項目組。同樣的，在步驟(2)中所產生的關聯法則期可靠度也必須達到使用者最初設定的最小可靠度門檻值。因為 Apriori 演算法牽涉到多次資料庫掃描以及可能產生過多的候選項目集，因此是一件耗時的工作。有許多改善的方法被提出，如 DHP 演算法[5]利用減少候選 2-itemsets 的個數來減少計算量以增進效能，而 DIC [9]、Partition [8]、Sampling [4]等演算法則是減少資料庫的掃描次數的方式來改善效能。

最小支持度門檻值的設定並不是一件容易的事。藉由線上調整門檻值的功能，使用者可機動的挖掘出對應的關聯法則以

作為適當決策的依據，此即所謂的線上探勘(On-line miming) 。Apriori-like 演算法並不適合線上探勘，因為每次調整新的門檻值時，這些演算法都必須重新掃描原始資料庫 [2]。

現實生活中，企業的交易是隨時進行的，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，因此產生了在動態資料庫進行漸進式探勘(incremental miming)的需求 [3]。Apriori-like 演算法也不適合漸進式探勘，因為每次新的交易資料進入資料庫時，這些演算法必須重新掃描更動後的整個資料庫而不能只考慮新增記錄集。Cheung et al.所提出的 FUP(Fast UPdate)的演算法[3]可算是漸進式探勘方法的代表範例，當資料庫更動時，藉由比較原始資料庫的高頻項目組(large itemset)與新增記錄集中的高頻項目組來減少其掃描原始資料庫的次數。但 FUP 不支援線上探勘。許志豪[2] 利用縮減項目集絡(Reduced Itemset Lattice)的觀念，在動態資料庫中做線上關聯法則探勘。該演算法利用 Apriori principle 將必要的項目集資訊儲存在絡(Lattice)的架構中，以方便關聯法則的探勘。在資料庫更新或最小支持度門檻值調整時，必須同步將絡中所儲存的資訊做必要的調整，若儲存的資訊越詳細，探勘時就越能減少必須到資料庫做確認的情況。

如何在新增的交易資料中，找出敏感度較高的資料項目，是一件十分重要的議題。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。這些隱含的資訊有可能淹沒在整體資料庫中。對於一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，所要累積的量一定需要相當時日才能達到使用者門檻值的限制，但熱門商品通常無法長久轟動、潛力客戶也必須即時鞏固與開發。以往的演算法也無法將此類的資訊即時挖掘出來，以提供市場即時變化的訊息作為適當決策的參考，因此產生了即時敏感性資料挖掘的議題。

本文提出一簡單的關聯法則之多層更新挖掘法 (Multilayer Update Miner, MUM)，此方法不需要重複掃描原始資料庫，因此，可支援線上探勘及漸進式探勘的需求。我們也探討 MUM 在敏感性資料挖掘上之應用。同時藉由許多實驗及相關的分析以檢驗 MUM 的執行效能並探討其優缺點與可行性。

貳、Multilayer Update Miner 演算法

一般而言，Apriori-like 演算法以批次(batch)的方式運作並且必須多次掃描資料庫以挖掘關聯法則，因此會造成以下限制：

1. 在門檻值變動時，必須重新掃描資料庫，不適合線上探勘。
2. 資料變動時必須重新掃描資料庫，不適合漸進式探勘。
3. 無法即時取出敏感性資料。

我們提出了 MUM 演算法，嘗試滿足上述的需求。

一、 MUM 演算法的特性

MUM 以即時處理的方式將每一筆交易記錄拆解成不同的項目組並將這些項目組資訊累積儲存在多層的表格陣列中。我們可在主記憶體或資料庫中建立 1-Itemsets、2-Itemsets、….、*n*-Itemsets 所需的主資料表格(base tables)，以紀錄每一筆新增資料所動態產生之相對應的項目

組。資料庫與表格中記錄著各項目組的出現次數的計數(count)值、支持度、可信度與所有交易記錄總數等資訊，以待下次資料挖掘時可快速的取得決定大項目組所需的資訊。

我們也利用備用資料表(temp tables)的觀念，將不符合支持度的項目組移至 temp tables，以減少主資料表格的資料量，加快資料庫搜尋速度。若新拆解出的項目組尚未出現在目前主資料表中，則可以到備用表格中搜尋，並新增或更改該項目組的資訊。下次新增資料時，在備用資料表中的項目組，其 count 的值還是會動態增加，當其支持度達到主資料表格的支持度門檻值時，即可移入主資料表格，成為大項目組。如此既可以提高擷取速度，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 演算法處理單一交易資料的過程如下：

step1：讀入目前欲處理的一筆交易資料。

step2：動態將交易資料分解成 1-Itemsets、2-Itemsets、…、*n*-Itemsets，並掃描對應的資料表中是否已經有該項目組存在。

step3：已存在：將對應項目組 count 數加 1。
未存在：新增項目組，count 數設為 1。

step4：計算此項目組的 support 是否達到目前的 minimum support。
a. 是，則留在主資料表中。
b. 否，則留在備用資料表中。

MUM 演算法其主要精神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，再儲存至對應的資料表格中並更新相關的資訊，以隨時反應目前交易資料庫所含之最新的項目組資訊。藉由這些資訊，不需要重複掃描整體資料庫即可快速的挖掘關聯法則，因此，MUM 演算法可滿足漸進式探勘的需求。再者，由於 MUM 隨時保持資料庫所含之最新的項目組資訊，所以，當使用者改變支持度時，可快速找出所有符合條件的大項目組，因此，MUM 亦可滿足線上探勘的需求。特別的是，MUM 允許為每個層級的項目組彈性的設定不同的支持度，如此可讓使用者更有效地擷取出真正符合所感興趣的資訊。然而，MUM 會記錄相當多的項目組，以目前儲存硬體的發展趨勢來看，資料存放空間應該不再是一個嚴重的問題，若再加上精巧的資料結構或平行處理的技巧，可望克服此一問題。

二、MUM 演算法的範例說明

假設第一次的原始交易資料如表 2.1 所示：

表 2.1　MUM 原始交易資料

Incremental database	
TID	Items
1	ACD
2	BCE
3	ABCE
4	ABE
5	ABE
6	ACD
7	BCDE
8	BCE

MUM 將每一層級的項目組(Itemsets)分別存入不同的資料表格陣列中，如表 2.2 所示。1-Itemset 的項目組，就存放於 1-Itemset 表格中；2-Itemset 的項目組，就存放於 2-Itemset 表格中，以此類推。由於存放的表格不同，每一表格儲存的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組逐一加入對應的表格中。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；備用資料表的格式如同表 2.2。

表 2.2 MUM 主資料表與備用資料表的格式

1-Itemset	Counts

1-Itemset Table

2-Itemsets	Counts

2-Itemsets Table

3-Itemsets	Counts

3-Itemsets Table

假定 1-Itemsets 到 4-Itemsets 的 minimum support 分別為 50%, 20%, 10%, 5%等四個不同的門檻值。逐一將拆解出的項目組，儲存至對應的資料表格中並更新相關的資訊。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 Count 值加一。並同步計算出其目前支持度的值。以此種方式推導，直到有某一個項目組的支持度不足其 minimum support 時，則將此一項目組移到備用資料表中。

表 2.3 列出 1-itemset 的主資料表資訊在處理過程的變化，左方的部份為資料庫的交易資料，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。請注意在 TID = 3 時，項目{D}因為不足 50%的 minimum support，所以被移入備用資料表中。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，備用資料表中項目{D}的 minimum support 值大於 50%時，依然會移入主資料表中。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 2.3 1-ItemSet 對應表

TID		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%

表 2.4 表示在處理此交易資料時，不足支持度的資料被移入備用資料表的情形。在表 2.3 中 TID = 3時，項目組{D}被移出主資料表，此時備用資料表就建立一個{D}的項目，並將其 count 值紀錄起來。隨著交易資料的處理，其 support 的數值會即時

的做異動，若是其支持度滿足預設的50%時，再將其項目組{D}移入主資料表中。

表 2.4 備用資料表的變化

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

對於 2-Itemset、3-Itemset、….、*n*-Itemsets 的資料表，都是依照上述的處理流程，來儲存對應的項目組及其相關的最新狀態。表 2.5、表 2.6 和表 2.7 分別為 2-Itemset、3-Itemset 和 4-Itemset 的演算過程。藉由此種多層即時更新的方式，MUM 可以快速的處理新增的交易，而不須重新掃描原始資料庫，大大的增進了資料挖掘的速度。

表 2.5 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		3	2					2		
7	BCDE					2	1	4	3	2	1
8	BCE					3				3	

minimum support = 20%

表 2.6 3-Itemset 對應表

TID		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 2.7 4-Itemset 對應表

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

minimum support = 5%

三、 MUM 演算法之實作

MUM 程式在實作的部分，有兩個主要的步驟，如圖所示：

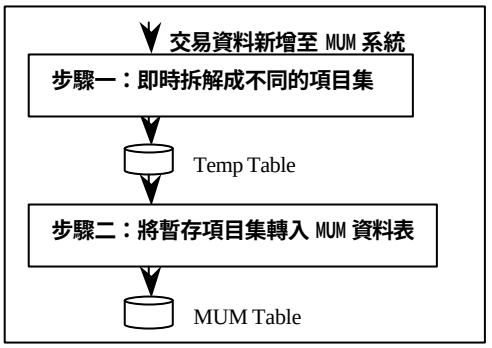

圖 2-1 MUM 演算法步驟示意圖

當一筆新增的交易資料進入系統中處理時，將啟動 MUM 的觸發程式，而將該交易資料即時拆解成不同的項目組集合。假設交易資料為{A，B，C，D}，則拆解項目組如下所示：

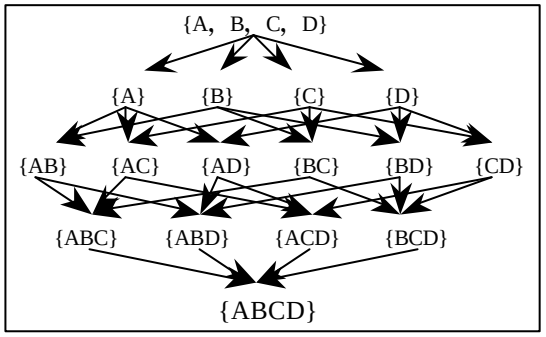

圖 2.2 交易項目拆解樹狀圖

假設新增交易資料為 *I*，其項目為 *n* 項，暫存資料表為 *T*，每次拆解的項次為 *k*，則類似 *Apriori* 演算法，MUM 藉由 T_k 的聯集產生下一階層 T_{k+1} 的項目以拆解出所有的項目集。其虛擬碼如下所示：

```
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
```

圖 2.3 MUM_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逐項加入 MUM 資料表中。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 2.4 MUM 演算法虛擬碼

？、MUM 在敏感性資料挖掘上之應用

從動態新增的交易資料中挖掘出有潛力的敏感性資訊，可提醒使用者必須注意的目標以利決策的進行。通常敏感性資料無法在短期間內達到使用者設定的最小支援度而被捨棄，因此，不易被挖掘出來。我們將利用 MUM 多層同步處理與更新的特性，搭配敏感度指數的定義，以便挖掘出對決策者有用的即時性敏感資訊。

一、 MUM 演算法與敏感度指數

我們將設定一段固定的時間 *t* 當作挖掘敏感性資訊的週期。使用 MUM 將週期內新增的交易資料建立資料表，並與本週期前建立的資料表做比較與合併，即可得到週期內資料與舊有資料的變化關係。以不同週期內資訊的變化量為基礎，我們分別定義了兩種敏感度指數，用來判斷各項目組的敏感度。

A. 一階敏感度指數

一階敏感度指數用來度量各項目組在相鄰兩週期支持度的變化關

係。令項目組 I 在週期 t 的支持度為 $sp_t(I)$，則其一階敏感度指數

$$s1_t(I) = sp_t(I) - sp_{t-1}(I) \qquad (3.1)$$

一階敏感度指數若是正值代表該項目組的重要性增加，若是負值則代表遞減。以超商為例，正值代表此相關商品的銷售量增加，反之則是減少。

若一項目組的一階敏感度指數絕對值超過使用者設定之門檻值，ms_1，則稱此一項目為敏感性資料。令 S_t 為週期 t 之敏感性資料的集合，其計算方式為：

$$\text{if } |s1_t(I)| > ms_1 \text{ then } S_t = S_t \cup \{I\} \qquad (3.2)$$

B. 二階敏感度指數

觀察每個週期的一階敏感度指數，可以找出當週期的變化量較大的敏感性資料，但對於漸進增加或是減少的資料則較難察覺。例如，某些項目組的變化量為逐次遞增，但每次遞增的量無法即時達到使用者設定之一階敏感度指數，如此還是無法成為敏感性資料，而被使用者忽略。因此，我們將連續 p 個週期的一階敏感度指數總和定義為二階敏感度指數。令項目組 I 在週期 t 的二階敏感度指數為 $s2_t(I)$，則

$$s2_t(I) = \sum_{n=0}^{p-1} S1_{t-n}(I) \qquad (3.3)$$

若一項目組的二階敏感度指數絕對值超過使用者設定之門檻值，ms_2，則此項目組亦為敏感性資料，也可以提供使用者參考。令 S_t 為週期 t 之敏感性資料的集合，其計算方式為：

$$\text{if } |s2_t(I)| > ms_2 \text{ then } S_t = S_t \cup \{I\} \qquad (3.4)$$

二、敏感度指數之範例說明

假設原始資料庫中已有 1000 筆交易資料，而在週期 t 時新增的交易資料如表 3.1 所示。以 MUM 演算法批次處理此八項交易資料，可以得到表 3.2 之 2-itemset 資料表。表 3.2 在資料庫中所對應的 t-1 週期的資料，如表 3.3 所示。

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

表 3.1 新增的交易資料

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

表 3.2 2-itemset 資料表

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

表 3.3 **原始** 2-itemset **資料表**

假設一階敏感性指數之門檻值為 25%。**由公式**(3.1)**可以得到表** 3.4，**其中**{BC},{BD},{BE}**三個項目組的一階變化率** *s*1 **的絕對值均大於** 25%，**由公式**(3.2)**可知此三項為敏感性資料**。

2-Itemset	s1
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

表 3.4 **一階敏感度資料**

設定第二階敏感度指數門檻值為 30%。**假設** p = 3**而且表** 3.5 **為連續三個週期的一階敏感度指數資料。由表** 3.5 **可知**{AD},{CE}**項目組每次的** s1 **的絕對值都沒有達到** 25%**的水準，但是其變化值均為正值，且明顯有上升的現象。由公式**(3.3)**計算三次增加的總和，得到其二階敏感度指數分別為為** 44.7%、38.7%，**都大於** 30%**的門檻值，由公式**(3.4)**可知**{AD}、{CE}**亦為使用者感興趣的敏感資料**。

Itemset	$s1_t$	$s1_{t-1}$	$s1_{t-2}$
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

表 3.5 **二階敏感度資料**

肆、實驗結果與討論

我們實作 MUM**演算法與** Apriori**演算法，並比較其執行時間以了解** MUM **的執行效能。我們也分析** MUM **演算法的優缺點並且探討其進一步可能的改善方式**。

一、 MUM 的執行效能

本實驗將驗證 Apriori **演算法與** MUM **演算法在漸進式探勘與線上探勘時，對於資料挖掘效率的差異**。

A. 實驗環境:

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

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

C. 程式設計：

我們在 Microsoft SQL Server 7.0 的環境下實作 Apriori 演算法，程式碼以預儲程序(stored procedures)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，所以執行效率相當良好。由於 MUM 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 VB6 撰寫，其執行時會在程式與資料庫連線與截止連線時，花費太多時間，而影響前處理的效率。我們仍將依此前提來比較其執行時間

D. 實驗結果與比較：

首先是動態新增資料庫之漸進式探勘的比較。因為 Apriori 演算法採批次處理的方式運作，所以假設原始資料庫已有 10000 筆交易記錄，之後，每次新增 100 筆。每次處理的最小支持度的筆數(*ms*)設定為 100。實驗結果如表 4.1 所示。由於每次新增資料時 Apriori 演算法均需重新挖掘異動後的整個資料庫，從表 4.1 中可以明顯的看出其挖掘時間隨著資料量的增加而遞增。從表 4.1 中亦可看出 MUM 在第一次處理原始資料庫內的 10000 筆記錄時花費相當多的時間。這是因為 MUM 必須拆解新增的記錄以建立項目組資料表，但是在第一次挖掘時，原始資料庫內已有相當多記錄，而且 MUM 是以 VB 開發，當資料量大時，需要花費很多時間在 VB 程式與資料庫的連線上。但是事實上，在現實生活中交易資料是漸增的，所以在交易的同時 MUM 即可同步進行項目組資料表的更新，無須一次處理大筆資料，這也是 MUM 的特性。由於 MUM 僅需處理新增的資料，從表 4.1 中可明顯的看出其挖掘時間不會隨著資料量的增加而遞增。

	10000 筆	10100 筆	10200 筆	10300 筆
Apriori 演算法	977	1005	1036	1076
MUM 演算法	9463	132	134	132

表 4.1 漸進式探勘實驗數據比較表(單位:秒)

在資料庫的交易記錄為 10000 筆時，進行線上探勘。動態設定的 *ms* 值分別為 150, 125, 100, 和 75。實驗

結果如表 4.2 所示。由於每次給定新的 *ms* 值時 Apriori 演算法需要重新掃瞄資料庫以挖掘所有高於支持度的項目組，從表 4.2 中可以明顯的看出其挖掘時間隨著 *ms* 值的下降而快速遞增。但 MUM 隨時保有最新的項目組資訊，所以當 *ms* 值改變時，僅需查詢項目組資料表，無須重新挖掘整體資料庫，即可找出所有超過 *ms* 值的大項目組。因此，其挖掘時間不會隨著 *ms* 變化。由實驗數據可以明顯的看出 MUM 的優勢。

	ms=150	ms=125	ms=100	ms=75
Apriori 演算法	617	637	977	2159
MUM 演算法	2	2	2	2

表 4.2 線上探勘時實驗數據比較表(單位:秒)

二、 MUM 演算法之進一步探討

MUM 藉由同步更新的方式將資料庫中所有的項目組資訊即時的記錄在多層資料表中，以避免重新掃描資料庫。MUM 事實上是一種利用硬碟空間換取資料挖掘時間的方法。無可諱言的，MUM 會產生過多的非高頻項目組，而佔用大量的儲存空間，對搜尋時間也會產生負擔。以下幾種方式可以改善此一問題：

1. 在高頻項目組與非高頻項目組間設定一個準高頻項目組(pre-large itemsets) [2]，作為一個有效的緩衝。
2. 善用資料庫中的索引功能或較精巧的資料結構，如 Extensible Hashing，以加快搜尋時間。
3. 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，可以有效的減少主資料表的大小並且加快搜尋速度。可利用即時的或批次的方式來移動項目組。
4. 利用平行處行的技術。將原資料庫的交易資料分割成多個小資料庫，再透過多台電腦或多處理器電腦來處理各小資料庫，處理完後再把結果整合起來，這樣便會大大縮簡 MUM 演算法的挖掘時間。
5. 也可使用分散式資料庫技術。在網路技術與應用日益增加的情況，分散式資料庫技術早就是各企業採用的方法了，而 MUM 演算法更適用於建構於此技術之上，各伺服器將各資料庫挖掘完之後能輕易整合起來，不僅增進挖掘的效率，也可增加其應用的廣度。可達到高內聚力(High Cohesion)，鬆散耦合度(Loosely Coupling)的效果。例如，對 POS 系統而言，MUM 技術更是適合。
6. 使用快取(Cache)技術。快取記憶體裡置換的條件可用很多種情況來評估，例如使用項目相關性來分析，其置換的條件如下：

$P(AB) = P(A)\,P(B)$ 兩項目獨立
$P(AB) > P(A)\,P(B)$ 兩項目相關度高
$P(AB) < P(A)\,P(B)$ 兩項目相關度低

P(AB)即為實際上 AB 所發生的機率，P(A)・P(B)即為我們對 AB 期待值，當實際上的發生機率大於預估的機率時，我們便把它視為相關度大的情況；反之，我們

便視之為相關度小的情況。當項目與項目間有相關性高的情形，我們才將之放入快取記憶體中，相關度不高時便置換出來。如此當我們執行查詢動作而在搜尋 MUM 資料表，找出相關高頻項目組的時候，只要先尋找快取記憶體，即能有效找到相關的高頻項目組。

伍、結論與未來工作

本文提出一簡單的關聯法則之多層更新挖掘法，MUM，此方法不需要重複掃描原始資料庫。實驗結果顯示 MUM 在線上探勘及漸進式探勘上具有良好的執行效能。利用 MUM 多層同步處理與更新的特性，搭配敏感度指數的定義，可以挖掘出對決策者有用的即時性敏感資訊。我們也討論動態資料挖掘時，計算時間與儲存空間的取捨問題，及其可能的改善方式。

除此之外，未來的研究將可朝著以下的方向進行：

1. 將 MUM 應用在實際的問題上以驗證其實用性。
2. 設計更好的項目組資料表的資料結構，以節省儲存空間及加快搜尋時間。
3. 定義更嚴謹的敏感度指數，以期求得敏感性資料的最佳策略，並實際驗證 MUM 在敏感性資訊挖掘的有效性。
4. MUM 與分散式處理技術結合的可行性。
5. MUM 與快取(Cache)技術結合的可行性。

參考文獻

1. 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。
2. 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。
3. 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.
4. H. Toivonen, "Sampling Large Database for association Rule", VLDB, pp. 134-145, 1996.
5. 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.
6. 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.
7. 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.
8. Savasere, E. Omiecinski, and S. Navathe, "An Efficient Algorithm for Mining Association Rule in Large Database", Proc. 21th VLDB,pp. 432-444, 1995.
9. 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.

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.