Commerce & Management Quarterly, Vol. 7, No.1, pp. 27 ~ 57 (2006)

高效率之遞增式探勘演算法－QPD

AN EFFICIENT INCREMENTAL MINING ALGORITHM－QPD

黃仁鵬
南台科技大學資訊管理所
黃南傑
南台科技大學資訊管理所
郭煌政
嘉義大學資訊工程所

Jen-peng Huang
Department of Information Management
Southern Taiwan University of Technology

Nan-Jie Huang
Department of Information Management
Southern Taiwan University of Technology

Huang-Cheng Kuo
Department of Computer Science and Information Engineering
National Chiayi University

摘　要

近年來，客戶關係管理（CRM）是個相當熱門的議題，因為企業必須了解消費者購物行為與商品間的關聯關係，才能妥善安排商品陳列順序。如此可以提昇客戶滿意度，減少購物的搜尋時間。再者可以刺激購買商品數量，用以增加企業的利潤。所以在大型交易資料庫中，利用資料探勘技術找出有用的關聯規則，來提供企業的決策支援是非常重要的。

本研究提出新的演算法 QPD（Quick Patterns Decomposition）來找出商品間的關聯規則。QPD 演算法的優點如下：1.只需掃描資料庫一次；2.利用型樣化方式來提昇執行效

率；3.利用遮罩（mask）與布林模式（Boolean）來產生拆解項目因子型樣；4.當最小門檻值變動時，不需重新探勘；5.資料庫有異動時，可方便進行漸進式探勘。

上述得知，透過本演算法做關聯分析，其效能將優於以往 Apriori-Base 的演算法。此外，關聯規則的推導過程中，將不會重複產生多餘的候選項目組，因此更勝於拆解模式的演算法。快速得到正確、有效用的資訊，是企業在數位時代中最大的利器，由此能降低時間成本、快速反映市場需求，是提昇競爭力的最大利基。

關鍵字：資料探勘、關聯規則、Apriori 演算法、高頻項目集、遞增式探勘

ABSTRACT

Recently Customer Relationship Management is one of the hottest issues in cooperation. In order to properly arrange the positions of products, Cooperation need to understand customers' shopping behaviors and the associations between products. In this way, we can increase the customers' satisfactions and decrease the searching time during shopping. Besides, we can increase the quantity of purchase products and the profits. Thus, it is very important to use the technology of data mining to find the useful association rules and to provide the cooperation's decision supports.

In this paper we propose a new algorithm QPD（Quick Patterns Decomposition）to find the association rules from large transaction databases. The merits of QPD algorithm are: 1. In data mining process it only needs to scan whole transaction database once. 2. Using Patterns method to increase the performance of data mining process. 3. Using mask and Boolean method to decompose the itemsets to sub-itemsets. 4. When minimum support changed we do not need to process mining process again. 5. It does not need to rescan the original database for mining the association rules from the incrementally growing databases.

From above illustration we know that by using QPD algorithm to process association analysis has better performance than Apriori-based algorithms. In association rule's reasoning process, it won't produce the unnecessary candidate items. Therefore it can fast obtain the information correctly and effectively and reduce the time cost, and fastly reflect the market demand. That will greatly promote the competitive advantage.

Keywords：Data Mining、Association Rule、Apriori、Frequent itemsets、Incremental mining

壹、前言

近年來電腦等高科技產業快速成長, 進而加速資訊化的過程。使得資料庫（Database）中的資料呈現倍數性成長，對企業決策者而言，要從大量資料中找出有用的隱含性資料是件非常困難的事。然而這些隱藏在資料庫中有用的資訊對於企業管理方面或決策支援方面等都有莫大的幫助。因此資料庫內的知識探索（Knowledge discovery）議題也隨之興起，資料探勘（Data Mining）的技術，更是重要的一環。

資料探勘的應用相當廣泛，例如在零售業中，我們可以藉由分析所有顧客的交易記錄來分析出顧客購買東西的習慣，然後將顧客經常一起購買的商品擺設在一起，以增加營業額。另外顧客關係管理（CRM）也是一種相當好的應用，可藉由分析顧客行為的資料，找出顧客的喜好、厭惡等資訊，作為高層決策人員在做決策時的參考。

資料探勘是指在大量資料儲存體中，探勘出隱藏、從未發現、且有用資訊的技術，所分析出具有價值的資訊提供決策者作有效的決策。Chen, Han, and Yu（1996）指出資料探勘是一種有效率的方法與整合技術，可以從資料中找出先前不知道，卻隱含於其中的有用資訊。而 Cabena, Hadjinian, Stadler, Verhees, and Zanasi（1997）指出資料探勘為將潛在或原本不知道的資訊從大型資料庫萃取出來的過程，並將萃取出來的資訊提供主管做決定性的決策。以上為各學者對資料探勘所下的定義，而如何有效的利用資料探勘技術從大量資料中找出其中所隱含的資訊為目前各研究學者努力的目標。

在本研究第二節中, 介紹關聯規則的相關文獻。在第三節中, 提出新的演算法—QPD 演算法來改進 Apriori 演算法的缺點，輔以實例說明運作過程，並比較 QPD、Apriori 與 FP-Growth 演算法 的執行效率。最後，在第四節中對本研究做結論。

貳、文獻探討

在眾多的資料探勘技術中，針對不同的應用領域、資料庫型態，目前已有許多不同的技術相繼被提出，每一種技術都有其特性及應用，其中主要的技術種類如下：描述與辨別（Characterization and Discrimination）、關聯分析（Association Analysis）、預測與分類（Classification and Prediction）、叢集分析（Cluster Analysis）、異常分析（Outlier Analysis）、趨勢分析（Evolution Analysis）等熱門技術。

Han and Kamber（2000）指出，關聯規則（Association Rules）是目前在資料探勘中最為成熟和利用最多的一個領域。關聯規則最早由 Agrawal, Imilienski, and Swami 於 1993 年所提出，主要是被用來尋找資料庫中項目之間的關聯性，Brin, Motwani, Ullman, and Tsur（1997）指出關聯規則最初被用於分析市場購物籃資料（Market Basket Data）的研究，藉由分析顧客之購買行為，找出相關商品集間彼此的關聯性，提供給決策者做為商品擺設、進貨、儲貨的參考，並有助於提昇商品的競爭力，以增進商品銷售週轉率提昇利潤。例如：『”顧客可能在購買牛奶之後，會接著再買麵包”，所以應該將牛奶類商品的陳設台靠近麵包類的陳設台，對於這樣的資訊就稱為”關聯規則”，表達方式為：牛奶→麵包[minsup=2%，minconf = 80%]』。在關聯規則分析中有兩個重要的參數，也就是支持度（support）跟信賴度（confidence），這兩個參數是用來評估所找出的關聯規則是否能滿足使用者的期望。而常見求取關聯規則的演算法有 Apriori 演算法、DHP、AprioriTid、AprioriHybrid、Boolean、FP-Tree、ICI、AIM…等。

以下將介紹關聯規則的主要定義、Apriori 演算法、DIC 演算法與 FP-Growth 演算法。

一、關聯規則探討

最早由 Agrawal et al.於 1993 年提出，主要是在大型交易資料庫中擷取項目（Item）之間的關聯性。關聯規則主要的問題定義如下：令 I={i1,i2,…,im}，I 為所有商品項目（Items）的集合，D 為資料庫中所有交易記錄的集合，T 為每筆交易記錄項目的集合，T 為 I 的子集合（T ⊆ I），而 T 中的交易記錄內容是不考慮商品項目購買的數量，TID 為每一筆交易記錄的編號。

關聯規則以 X→Y 表示，其中 X⊂I，Y⊂I 且 X∩Y=∅。然而，一個關聯規則衡量的規則是什麼？如何才能讓一個關聯規則成立？衡量關聯規則的標準有二，一是支持度（support）、二是信賴度（confidence），以下將詳細介紹這兩種衡量標準。

1. 支持度（Support；s）：支持度的定義為在 D 中包含 X 且包含 Y 交易記錄個數和 D 中所有交易記錄個數的比例，公式如(1)所示。

$$s = \frac{\text{資料項目}X\text{與}Y\text{同時在資料庫}D\text{中出現的次數}}{\text{資料庫}D\text{的總筆數}},\ (0 < s \le 1) \qquad (1)$$

2. 信賴度（Confidence；c）：信賴度的定義為在 D 中包含 X 的交易記錄中也包含 Y 交易記錄所佔的比例，公式如(2)所示。

$$c(X \rightarrow Y) = \frac{\text{資料項目}X\text{與}Y\text{同時在資料庫}D\text{中出現的次數}}{\text{資料項目}X\text{在資料庫}D\text{中出現的次數}}, (0 < c \leq 1) \quad (2)$$

其探勘關聯規則的工作主要可分為二個階段，(一)找出資料庫所有的高頻項目集（Frequent itemsets），也就是找出所有滿足最小支持度的項目組，若一個項目組含有 k 個項目，則稱為 k-項目組（k-itemsets），若 k-項目組滿足最小支持度，則稱為 k-高頻項目集；(二)根據第一階段產生的高頻項目集產生關聯規則；例如 BCE 為 3-高頻項目組，且 B、C、E⊆I，若關聯規則 BC→E 滿足最小信賴度，表示此關聯規則成立。

二、Apriori 演算法

Apriori 演算法由 Agrawal and Srikant（1994）所提出，此一演算法是最具代表性的關聯規則演算法之一，而許多推導關聯規則技術的相關演算法，都是以 Apriori 為基礎加以改良或延伸，目前改進的方法有 AprioriTid、AprioriHybrid、Boolean、Partition、DIC、Cloum-Wise Apriori、Multiple-Leve…等。Apriori 演算法主要包含以下步驟：

(1) 利用（k-1）-高頻項目集（L_{k-1}）來產生候選項目集合（C_k）。
(2) 掃描資料庫 D，計算所有候選項目集合的支持度，將所有支持度大於等於最小支持度的候選項目集合選出來成為長度為 K 的高頻項目集（L_k）。
(3) 重複上面(1)(2)步驟，直到無法再產生新的候選項目集合為止。

(一) 候選項目合併（Join）與修剪（Pruning）規則：

(1) 由上述的步驟(1)找出有兩個 k-2 項目相同的（k-1）-高頻項目集，組合成 k-項目組。
(2) 判斷(1)步驟中的 k-項目組，其所有的（k-1）-項目組之子集合是否都出現，假如成立則保留此 k-項目組。

(二) Apriori演算法的兩個瓶頸

(1) 產生大量的候選項目集（Itemset）

在產生 2-候選項目時是由 1-頻繁項目兩兩合併產生，若 1-頻繁項目集中有 k 個項目，共會產生（k-1）+（k-2）+…+1 個 2-候選項目，即 k*（k-1）/2 個。假設 1-頻繁項目集中有 1000 個項目，則產生 45 萬個 2-候選項目。

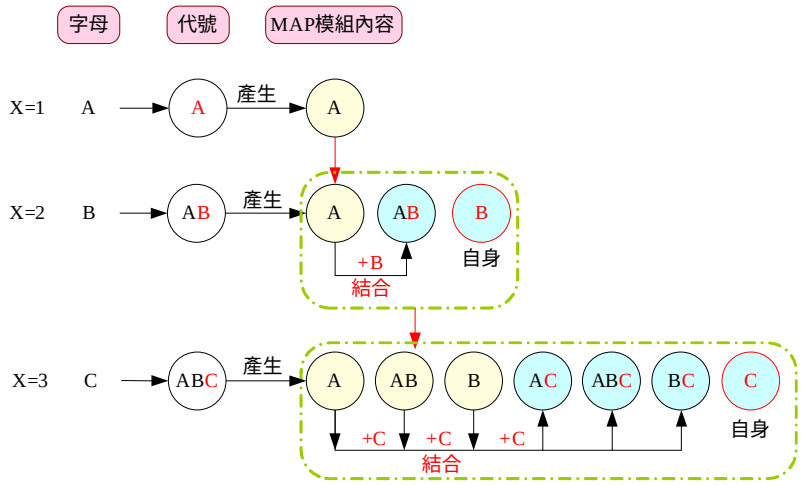

圖 1 ICI 演算法的 MAP 模組產生方法

(2) 需要多次掃瞄資料庫

由(1)結果可知，因為有大量的候選項目，而且每一個項目都必須掃瞄整個資料庫求取其支持度（support），造成整體執行效率不佳。所以本研究的目的在於改善，產生頻繁項目集時所需花費的時間。

三、ICI 快速關聯規則演算法

ICI 演算法由黃仁鵬、錢依佩與吳聲弘(2003)提出，此演算法是利用模組產生器(MAP 模組）來快速配對交易項目，模組產生器中採用一個新的配對方式。其配對內容分成三個部分：一、保留原有的項目組（如圖 1 中 x=2 的 A 項目）；二、直接加入新欲加入的項目（如圖 1 中 x=2 的 B 項目）；三、將舊有與新的項目進行結合。（如圖 1 中 x=2 的 AB 項目）如圖 1 所示。

產生 MAP 模組後，將交易項目利用替代的方式，直接將資料應對至模組中，並記錄對應所產生項目組於因素項目表（Element Table；ET）中，再檢視最小支持度後快速得到關聯規則，因此無須對資料進行一一拆解，其效率因而大幅度提昇，其優點有：一.僅掃描資料庫一次、二、利用模組產生器配對，不產生重複的候選項目組，故 ICI 演算法的執行效率比 Apriori 演算法快。

四、Frequent-Pattern tree（FP-Growth 演算法）

FP-Growth 演算法是由 Han, Pei, and Yin（2000）提出，此演算法是不產生 candidate itemsets 作法的代表。它將資料庫壓縮在 Frequent-Pattern tree 的結構中，因為不用產生 candidate itemsets，所以只需掃瞄資料庫兩次，可以避免多次的高成本的資料庫掃瞄，節省了大量 I/O 的時間，因此整體的效率相當不錯。FP-Growth 演算法的作法可分為兩個階段，主要步驟如下：

(一) 第一階段建立Frequent-Pattern tree：

(1) 第一次掃瞄資料庫，找出符合 minimum support 的 large 1-itemset。
(2) 將每一筆記錄中 large items 依其出現在資料庫中的次數，作降冪排列。
(3) 第二次掃瞄資料庫時，建立 FP- tree。

(二) 第二階段探勘Frequent-Pattern tree：

(1) 對 FP-Tree 中的每一個分支 node，建立 conditional pattern base。
(2) 再對每一個 conditional pattern base 分別建立其 conditional FP-Tree。
(3) 再對 conditional FP-Tree 進行挖掘，並逐次增加包含在 conditional FP-Tree 的 Frequent Pattern。
(4) conditional FP-Tree 中有包含一條路徑，就可列舉出所有 pattern。

FP-Growth 演算法的優點為，不用產生候選項目集，而且將資料庫壓縮在 FP-Tree 的結構中，也改進了多次掃描資料庫的次數，其缺點為，所運用到的資料結構較複雜。

參、研究方法

本研究針對 Apriori 演算法的缺點做改進，因為在每一個產生頻繁項目的階段都必須掃描資料庫 N 次，相當的費時，也造成整體執行效率不佳。所以本研究的目的在於改善找尋關聯規則時所需花費的時間。在本章節中詳細介紹整個 QPD 演算法的執行流程。一、將演算法運作過程繪製成流程圖，並依據流程圖詳細說明整個流程步驟。二、細說 QPD 之演算法。三、說明型樣內部的拆解、配對、組合的演進過程。四、依照流程步驟透過實例來說明。五、與 Apriori、FP-Growth 演算法執行效率比較。

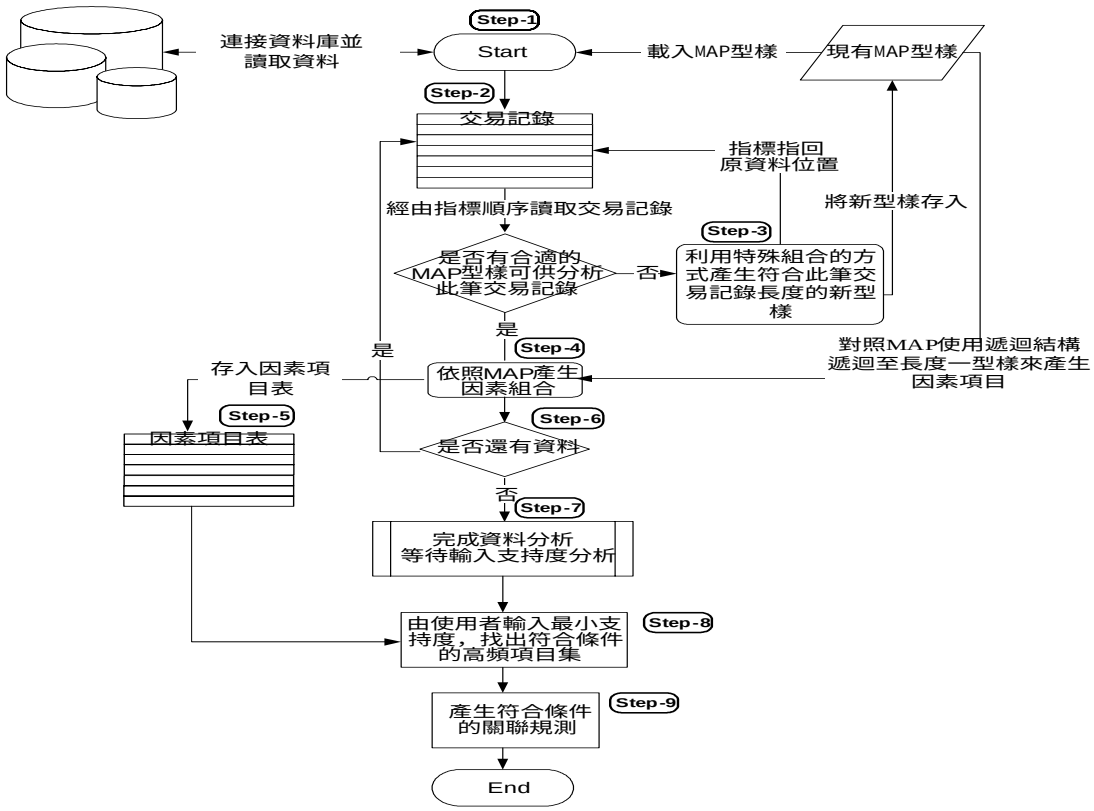

圖 2 QPD 演算法流程圖

一、演算法流程執行步驟

此節將詳細介紹 QPD 演算法的流程步驟與流程圖，如圖 2 所示。

二、QPD 演算法說明

此節將詳細說明 QPD 演算法與型樣產生器的虛擬碼。

演算法程式說明：如圖 3 所示。（行 1）建立字母對照表，內容預設 52 個英文字母從大寫到小寫（A,B,…,Z,a,…,z）以供對映用，例如：交易記錄長度為一（X=1）會擷取到 A 代碼、長度為三（X=3）會擷取到 C 代碼。（行 2 行 3）設定初值，MM 為存放型樣資訊用，初值為 null；FBT 所存放的是終值的二元樹，初值為 null。（行 4）開啟型樣檔案檔案，將檔案內容依序存放至 MM 中，以供對映用或新增型樣用。（行 5 行 23）

```
QPD_Algorithm
Input：CM，MM，Transaction DB
Output：All_Itemsets
行 1  create CharacterMapping CM;
行 2  modelMapping MM;
行 3  Final_Binary_Tree FBT;
行 4  open Model File add to MM;
行 5  forall transaction t contains DB do begin
行 6    n = count(t.Elements);
行 7    if (MM.contains(n)) then
行 8      ModelElements ME = RecursiveModelElements(MM(n,1));
行 9      itemsets = Replace(ME,t);
行 10   else
行 11     produceNewModel(MM.max,n);
行 12     ModelElements ME = RecursiveModelElements(MM(n,1));
行 13     itemsets = Replace(ME,t);
行 14   endif
行 15   forall itemsets i add to FBT do begin
行 16         if (i contains FBT)
行 17           i.count++;
行 18         else
行 19           add i to FBT;
行 20           i.count = 1;
行 21         endif
行 22   end
行 23 end
```

圖 3　QPD 演算法

QPD 演算法主程式，資料庫內的交易記錄有幾筆，等於此區塊被執行的次數。（行 6）擷取讀入的交易記錄長度。（行 7　行 14）判斷是否有合適的 MAP 型樣，”True”執行（行 8、9）由符合該交易記錄長度（n）的型樣遞迴至長度一的型樣，以取得 MAP 型樣的資料內容，再將取得的資料內容取代為交易記錄的內容，”False”執行（行 11　行 13）先利用 produceNewModel 函式產生所需的新型樣，其中 MM.max 為目前型樣內最大的長度，n 為目前交易記錄所需的長度。在新型樣產生後，再由新型樣遞迴至長度一的型樣，以取得 MAP 型樣的資料內容，再將取得的資料內容取代為交易記錄的內容。（行 15　行 22）將對映後的項目集組合，存到 FBT 中計數，若此項目集已存在 FBT 中，則將此項目集之值加 1；若不存在，則將此項目集加入 FBT 中，並給予初值 1。重覆執行（行 5　行 23）直到沒有任何交易記錄為止。如圖 4 所示。produceNewModel（MM.max,n）此函式即為所述的型樣產生器。（行 25）讀取目前 MAP 型樣中最大長度裡資料內容長度最長的項目，除了加入新型樣用，也做為組合的依據。（行 26）取得字母對照表的字母，做為組合用。

```
QPD_produceNewModel(MM.max,n)
Input：MM.max，n
Output：New_Map_Model
行24 for (i = MM.max ; i<n ; i++)
行25     last_MaxElement = get_Length_Max_ModelElement(MM(i));
行26     item = CharacterMapping(i+1);
行27     New_MaxElement = (last_MaxElement+item);
行28     n = count(New_MaxElement);
行29     if (n>=3)
行30         for (j=1;j<n;j++)
行31             coverElements = cover_New_MaxElement(j);
行32             m = count(coverElements);
行33         end
行34         if (m==2)
行35             NewElements = New_MaxElement + item + coverElements;
行36         else
行37             forall Boolean coverElements until length = 2 do begin
行38                 BooleanElements = Boolean_coverElements(coverElements);
行39             end
行40             NewElements = New_MaxElement + item + coverElements + BooleanElements;
行41         endif
行42     else
行43          NewElements = New_MaxElement + item;
行44     endif
行45     MM.insert(NewElements,i+1);
行46 end
```

圖 4　QPD 型樣產生器演算法

（行 27）透過組合方式產生新型樣長度最長之項目（由上一個型樣長度最長之項目與字母對照表的字母組合來）。（行 28）計算新型樣最長項目之長度。（行 29　行 44）判斷新型樣最長項目之長度是否大於等於 3，”True”執行（行 30　行 41）使用特殊拆解方式來拆解其餘之子項目，”False”執行（行 43）直接將新型樣最長之項目，再加上字母本身，成為新型樣的資料內容。（行 30　行 33）使用遮罩方式來產生子項目。（行 34　行 41）判斷由遮罩所產生之子項目長度是否等於 2，”True”執行（行 35）將新型樣最長之項目，加上字母本身，再加上遮罩所產生之子項目，成為新型樣的資料內容，”False”執行（行

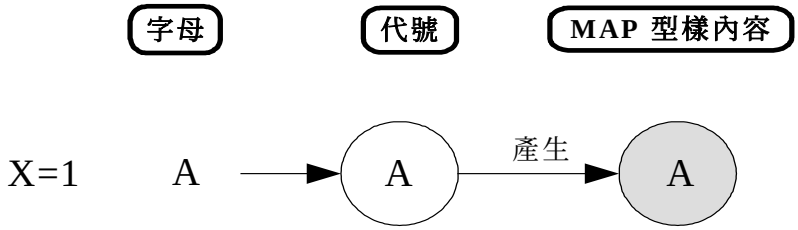

圖 5　MAP 型樣產生圖（A）

37　行 40）將遮罩之子項目彼此相互使用 Boolean 運算，直到產生所有長度二之項目。（行 40）將新型樣最長之項目，加上字母本身，再加上遮罩所產生之子項目與 Boolean 運算來之子項目，成為新型樣的資料內容。（行 45）將產生的新型樣加入 MM 中。

三、QPD 型樣產生方式實例說明

此節詳細介紹 QPD 型樣的拆解、配對、組合的演進過程，並舉一例子說明型樣與交易記錄對映方式之運作過程。

(一) 型樣組合產生方式

假設目前型樣產生器，欲產生長度為一（X=1）的型樣資料內容，此時 MAP 型樣內並沒有任何型樣，而型樣產生器會依據長度來判斷，要從字母對照表中讀出何種字元做組合。

此例是欲產生長度為一的型樣資料內容，所以從字母對照表中讀入 A 字母當做組合項目集的因子（所謂的字母對照表是指大寫字母「A」到小寫字母「z」的 52 個字母，當 Itemsets 的長度為一時，會讀取到大寫字母 A，當 Itemsets 的長度為 27 則會讀取到小寫字母 a，以此類推），來產生型樣的資料內容。讀 A 字母後，型樣產生器隨即產生長度為一的 MAP 型樣資料內容。此時，代號 A 只能產生一個項目集，其內容即為 A，如圖 5 所示。

而交易記錄長度為二（X=2），假設代號為 AB，可利用上一個 X=1 的內容來間接產生 AB 的項目組合內容，如圖 6 所示。代號 AB 比上一個 MAP 型樣的代號多增加了一個項目 B，作法只需要將項目 B 與上一個 MAP 型樣最長的資料內容結合，例如：上一型樣資料內容 A 與本身 B 結合成 AB，最後將自己本身 B 加入，而此型樣的資料內容第一個

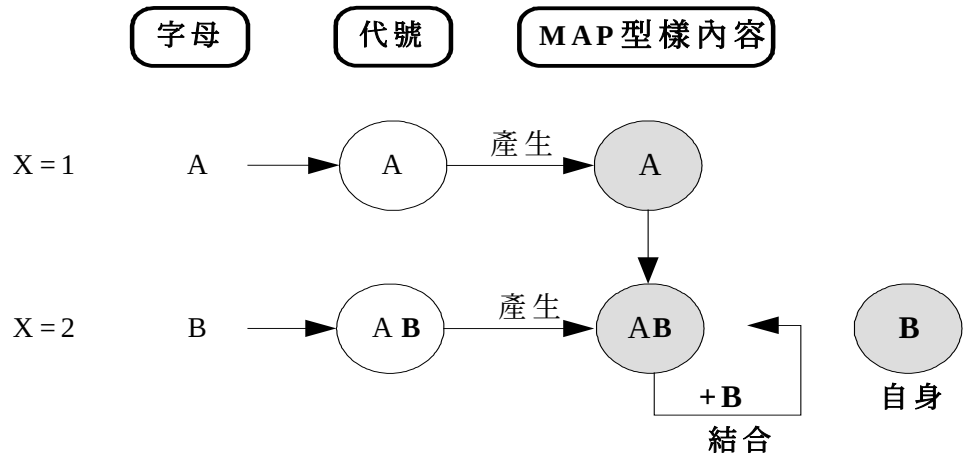

圖 6 MAP 型樣產生圖（AB）

拆解方式說明：拆解長度三（A,B,C）的子項目

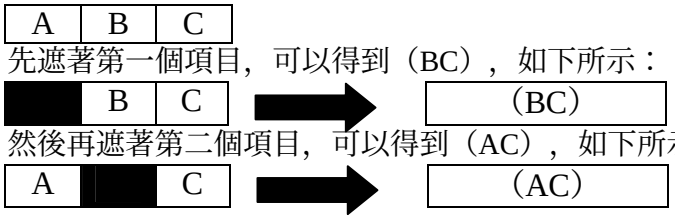

圖 7（a） 使用遮罩方式產生長度 3 子項目集

項目為長度最長之項目，如此以方便下一個型樣抓取上一型樣長度最長之項目，即完成組合這個動作。

而交易記錄長度為三（X=3），假設代號為 ABC，可利用上一個 X=2 的內容來間接產生 ABC 的項目組合內容，如圖 7（b）所示。代號 ABC 比上一個 MAP 型樣的代號多增加了一個項目 C，作法只需要將項目 C 與上一個 MAP 型樣最長的資料內容結合，例如：上一型樣資料內容 AB 與本身 C 結合成 ABC，並將自己本身 C 加入，由於此型樣的長度超過三，所以必須要使用圖 7（a）所示的拆解方式，來拆解出其他子項目集，將新產生項目 ABC 使用遮罩方式，產生 BC 與 AC，即完成組合這個動作。

Mark 的結束條件為 N-1 次，N 為項目長度。例如要拆解的為（A,B,C）長度三的項

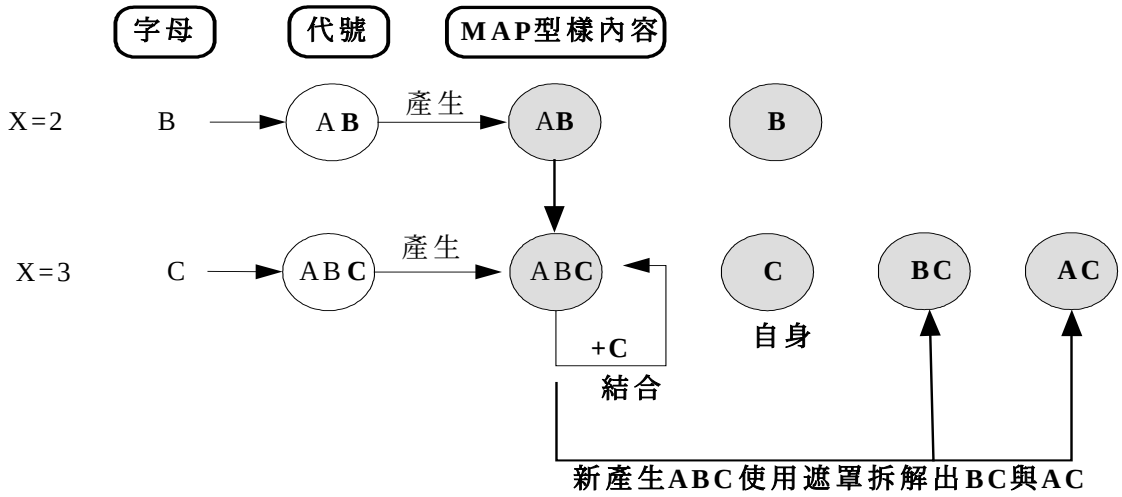

圖 7（b） MAP 型樣產生圖（ABC）

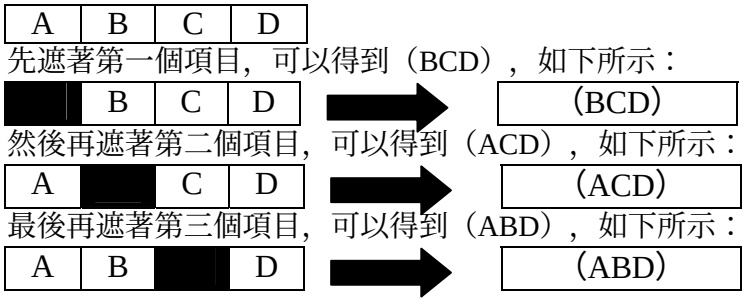

圖 8（a） 使用遮罩方式產生長度 3 子項目集

目所以 N 就為 3，遮罩的次數就是 2 次，並判斷所拆解出項目長度為二，所以就停止往下拆解。

而交易記錄長度為四（X=4），假設代號為 ABCD，可利用上一個 X=3 的內容來間接產生 ABCD 的項目組合內容，如圖 8（c）所示。代號 ABCD 比上一個 MAP 型樣的代號多增加了一個項目 D，作法只需要將項目 D 與上一個 MAP 型樣最長的資料內容結合，例如：上一型樣資料內容 ABC 與本身 D 結合成 ABCD，並將自己本身 D 加入，由於此型樣的長度超過三，所以必須要使用圖 8（a~b）所示拆解方式，來拆解出其他子項目集，將新產生項目 ABCD 使用遮罩方式與布林 AND 運算，產生 BCD、ACD、ABD、CD、BD、AD，即完成組合這個動作。

（B,C,D） AND （A,C,D），得到（C,D）				
項目	A	B	C	D
B,C,D	0	1	1	1
A,C,D	1	0	1	1
	0	0	1	1

（B,C,D） AND （A,B,D），得到（B,D）				
項目	A	B	C	D
B,C,D	0	1	1	1
A,B,D	1	1	0	1
	0	1	0	1

（A,C,D） AND （A,B,D），得到（A,D）				
項目	A	B	C	D
A,C,D	1	0	1	1
A,B,D	1	1	0	1
	1	0	0	1

圖 8（b） 使用布林 AND 運算產生其餘的子項目集

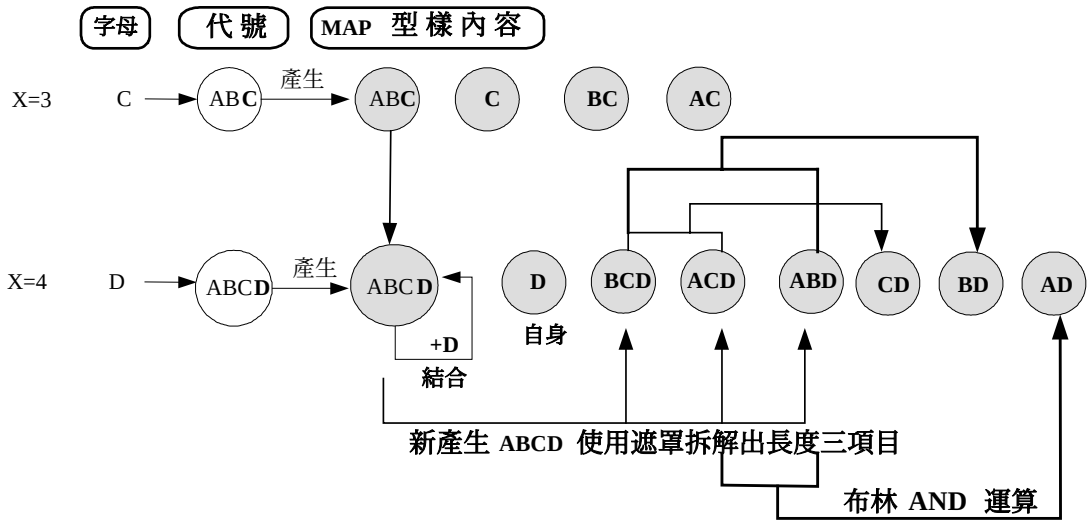

圖 8（c） MAP 型樣產生圖（ABCD）

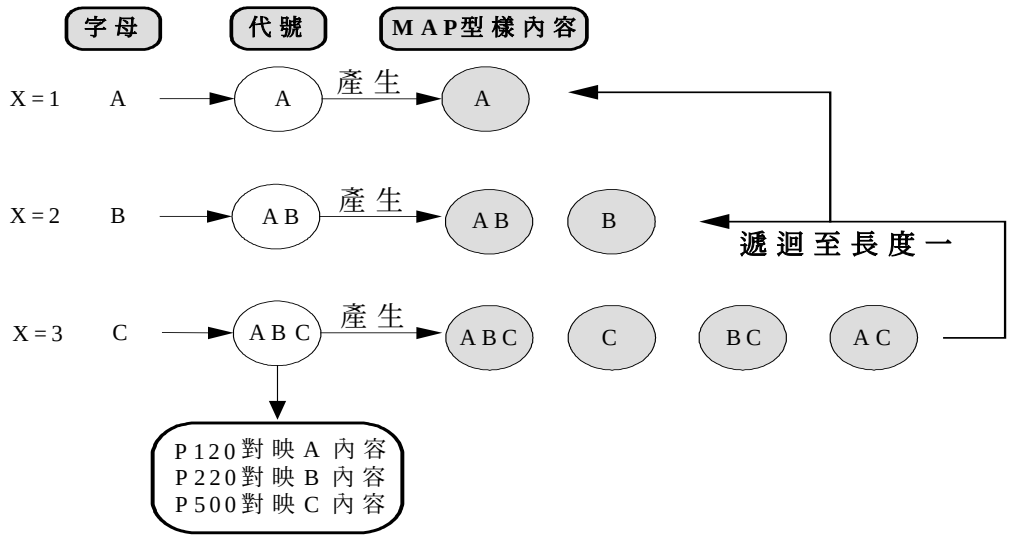

圖 9　使用遞迴結構遞迴至長度一之型樣

特殊之拆解方式說明：拆解長度四（A,B,C,D）的子項目

步驟一：使用上述遮罩方式產生 BCD、ACD、ABD（因為使用遮罩所拆解出之子項目長度大於二，所以必須執行步驟二，以求出所有長度二之項目），如圖 8（a）所示。

步驟二：利用步驟一所產生之項目相互做布林 AND 運算，直到 AND 出所有長度二項目，說明如圖示。

(二) 型樣與交易記錄對映方式實例說明

假設一交易記錄為（P120,P220,P500），數字表示為商品編號，其交易長度為三，所以對映到 X=3 的 MAP 型樣內容，即 P120⇔A；P220⇔B；P500⇔C，依據 MAP 型樣的資料內容經由取代的方式加上遞迴的觀念，遞迴至長度一的型樣，產生（P120,P220,P500）、（P500）、（P220,P500）、（P120,P500）、（P120,P220）、（P120）、（P220）如圖的完整對映圖所示，圖 9、圖 10。（“⇔”表示『對映關係』）。

四、完整實例說明

首先，步驟一，載入資料庫中的交易記錄以及 MAP 型樣，表 1 為顧客購買商品項目

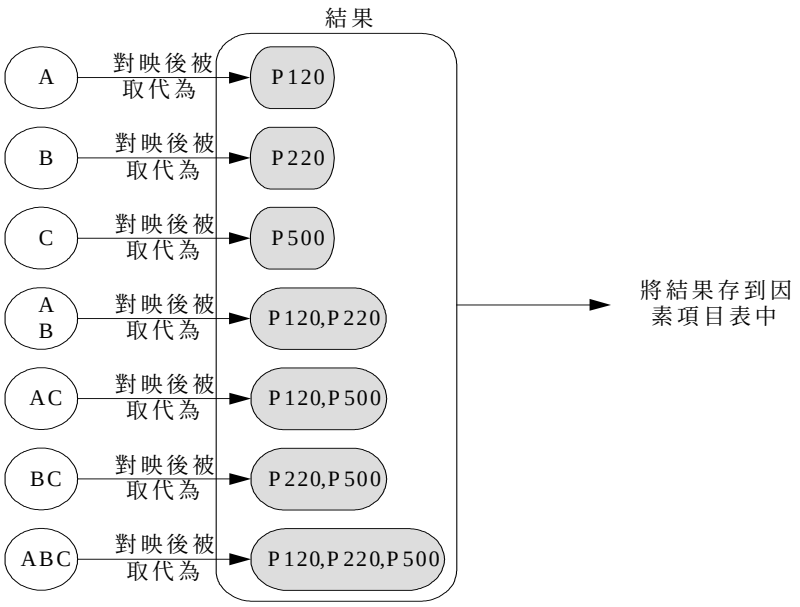

圖 10 完整對映過程圖

表 1 原始資料庫 D

TID	Itemsets
T001	P001,P003,P004
T002	P002,P003,P005
T003	P001,P002,P003,P005
T004	P002,P005

的交易資料庫內容，TID 為顧客交易的編號，Itemsets 為顧客購買商品項目的交易記錄代碼細目；而表 2 為現有 MAP 型樣的資料內容，在此假設型樣內已有 3 個長度的型樣資料內容。進入步驟二，讀入第一筆交易記錄，T001（P001,P003,P004），此筆交易記錄長度為三，隨即至 MAP 型樣中尋找是否有合適的型樣資料內容，判斷”是”即跳到步驟四，依據 MAP 型樣的資料內容使用遞迴的方式，產生（P001, P003, P004）、（P004）、（P003,P004）、（P001,P004）、（P001, P003）、（P001）、（P003）等對映後的項目集組合。步驟五，將步驟四之結果存至因素項目集中，如表 3 所示。步驟六，判斷尚有記錄，跳回步驟二讀取下一筆交易記錄。

表 2　MAP 型樣

交易記錄長度	交易記錄代表內容	所有的交易項目集組合
X=1	A	A
X=2	AB	AB,B
X=3	ABC	ABC,C,BC,AC

表 3　因素項目表（讀入 T001 對映後項目集組合）

X=3		X=2		X=1	
Itemsets	Sup	Itemsets	Sup	Itemsets	Sup
P001,P003,P004	1	P001,P003	1	P001	1
		P001,P004	1	P003	1
		P003,P004	1	P004	1

表 4　因素項目表（讀入 T002 對映後項目集組合）

X=3		X=2		X=1	
Itemsets	Sup	Itemsets	Sup	Itemsets	Sup
P001,P003,P004	1	P001,P003	1	P001	1
P002,P003,P005	1	P001,P004	1	P002	1
		P003,P004	1	P003	2
		P002,P003	1	P004	1
		P002,P005	1	P005	1
		P003,P005	1		

步驟二　步驟六，讀入第二筆交易記錄，T002（P002,P003,P005），拆解方式與上述相同，因素項目表更新後如表 4。

步驟二，再讀取下一筆交易記錄 T003（P001,P002,P003,P005）。判斷長度為四，因為 MAP 型樣中沒有合適的長度，所以進入步驟三。利用型樣產生器，產生此筆交易記錄所需的新型樣。產生後將新產生的型樣加入 MAP 型樣中，如表 5 所示。

取得新型樣後進入步驟四，依據新 MAP 型樣的資料內容使用遞迴之方式，產生（P001,P002,P003,P005）、（P005）、（P002,P003,P005）、（P001,P003,P005）、（P001,P002,P005）、（P003,P005）、（P002,P005）、（P001,P005）、（P001,P002,P003）、

表 5 新 MAP 型樣

交易記錄長度	交易記錄代表內容	所有的交易項目集組合
X=1	A	A
X=2	AB	AB,B
X=3	ABC	ABC,C,BC,AC
X=4	ABCD	ABCD,D,BCD,ACD,ABD,CD,BD,AD

表 6 因素項目表（讀入 T003 對映後項目集組合）

X=4		X=3		X=2		X=1	
Itemsets	Sup	Itemsets	Sup	Itemsets	Sup	Itemsets	Sup
P001,P002, P003,P005	1	P001,P002,P003	1	P001,P002	1	P001	2
		P001,P002,P005	1	P001,P003	2	P002	2
		P001,P003,P004	1	P001,P004	1	P003	3
		P001,P003,P005	1	P001,P005	1	P004	1
		P002,P003,P005	2	P003,P004	1	P005	2
				P002,P003	2		
				P002,P005	2		
				P003,P005	2		

（P003）、（P002,P003）、（P001,P003）、（P001,P002）、（P002）、（P001） 等對映後的項目集組合。步驟五，將結果存至因素項目集中，如表 6 所示。步驟六，判斷尚有記錄，跳回第二步驟，讀取下一筆交易記錄。

步驟二 步驟五，再讀取下一筆交易記錄 T004（P002,P005），拆解方式與上述相同，因素項目表更新後如表 7。

步驟六，判斷已沒有任何交易記錄，進入步驟七等待使用者輸入最小支持度與最小信賴度。步驟七，在此假設使用者設定的最小支持度為 50%（最低交易次數為 50%*4=2）、信賴度為 80%。步驟八，由因素項目表中找出符合最小支持度的大項目集合，整理如表 8。

第三大項目集合（L3）：（P002,P003,P005）。第二大項目集合（L2）：（P001,P003）、（P002,P003）、（P002,P005）、（P003,P005）。第一大項目集合（L1）：（P001）、（P002）、（P003）、（P005）。

表 7　因素項目表（讀入 T004 對映後項目集組合）

X=4		X=3		X=2		X=1	
Itemsets	Sup	Itemsets	Sup	Itemsets	Sup	Itemsets	Sup
P001,P002, P003,P005	1	P001,P002,P003	1	P001,P002	1	P001	2
		P001,P002,P005	1	P001,P003	2	P002	3
		P001,P003,P004	1	P001,P004	1	P003	3
		P001,P003,P005	1	P001,P005	1	P004	1
		P002,P003,P005	2	P003,P004	1	P005	3
				P002,P003	2		
				P002,P005	3		
				P003,P005	2		

表 8　大項目集合

X=3		X=2		X=1	
Itemsets	Support	Itemsets	Support	Itemsets	Support
P002,P003,P005	2	P001,P003	2	P001	2
		P002,P003	2	P002	3
		P002,P005	3	P003	3
		P003,P005	2	P005	3

根據大項目集合（L3、L2、L1），產生下列關聯規則。

P002,P003→P005（100%>80%）　　P003,P005→P002（100%>80%）

P001→P003（100%>80%）　　P002→P005（100%>80%）

P005→P002（100%>80%）

五、演算法執行效率比較

為了驗證 QPD 演算法的執行效率，本研究分別設計了幾個實驗的資料庫，來檢測本研究提出的方法，並將實驗的結果與 Apriori、FP-Growth 演算法做效率比較，再將實驗的結果做相關的評估與說明。本研究利用以下的環境來進行資料探勘效率的測試：

(一) 實驗環境說明

1. 實驗平台

(1) CPU：PentiumⅣ 1.7GHz

表 9 定義參數

D	資料庫的總交易量（筆數）
L	資料庫中單筆交易記錄最大的項目數
I	產生的頻繁項目集平均最大的項目數
N	交易資料庫中項目的總數

(2) Memory：512Mbytes
(3) OS：Windows 2000 Sever
(4) Data Base：MS SQL Sever 2000
(5) Programming Language：Java JDK 1.4

2. 實驗資料庫

本研究實驗為求公正，實驗資料庫是由 IBM generator 產生。產生測試資料庫參數如表 9 所示。

本研究用來實驗資料庫有 15 個，資料庫筆數（D）介於 10K 至 2000K，包含的項目個數為 200 至 500，交易記錄長度（L）介於 10 至 16 個項目，平均最大頻繁項目集之項目數（I）介於 3 至 5（IBM generator 預設 I=4），根據此命名原則，用來測試的 15 組資料庫如表 10 所示。

(二) 數據效能評估

1. **實驗 1**：支持度對執行效率的影響。

依據資料庫 L10N500I4D100K，經過實驗，產生上圖的實驗結果，而圖 12，為了方便比較 QPD 與 FP-Growth 演算法，將它獨立出來。

由圖 11、圖 12 與表 11 可知本研究 QPD 演算法執行效率平穩，不會因為支持度的不同而影響執行效率，而 Apriori 與 FP-Gr 演算法很明顯的可以看出會因為支持度的不同而影響執行效率，由上結果可知，QPD 演算法在不同支持度下所花費的執行時間優於 Apriori 與 FP-Growth 演算法。

表 10　實驗資料庫內容

Database Name	L	N	I	D
L10N500I4D2000K	10	500	4	2000K
L10N500I4D1000K	10	500	4	1000K
L10N500I4D100K	10	500	4	100K
L10N400I4D100K	10	400	4	100K
L10N300I4D100K	10	300	4	100K
L10N200I4D100K	10	200	4	100K
L10N500I4D80K	10	500	4	80K
L16N500I4D50K	16	500	4	50K
L14N500I4D50K	14	500	4	50K
L12N500I4D50K	12	500	4	50K
L10N500I3D50K	10	500	3	50K
L10N500I4D50K	10	500	4	50K
L10N500I5D50K	10	500	5	50K
L10N500I4D20K	10	500	4	20K
L10N500I4D10K	10	500	4	10K

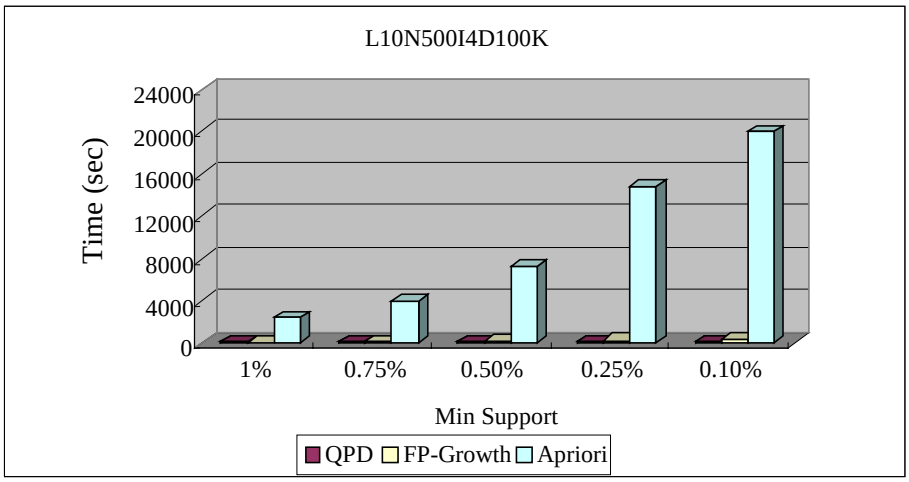

圖 11　QPD、Apriori 與 FP-Growth 演算法效率比較圖

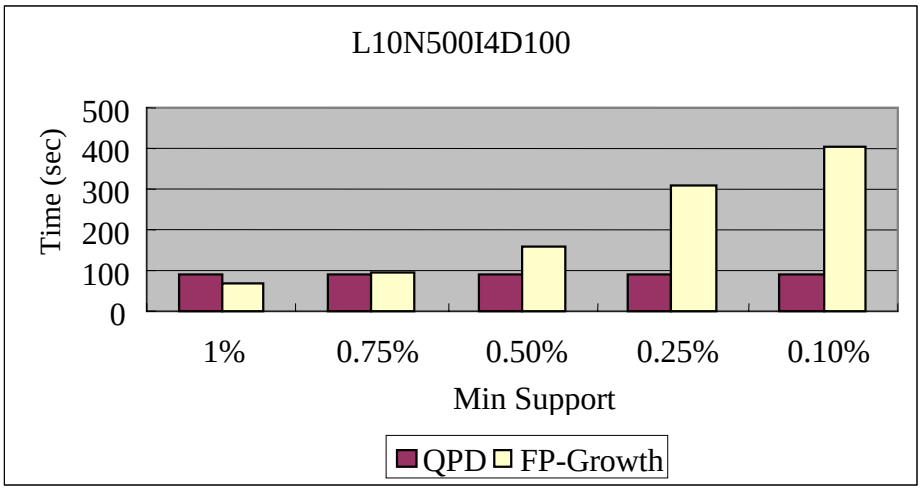

圖 12 QPD 與 FP-Growth 演算法效率比較圖

表 11 QPD、Apriori 與 FP-Growth 演算法效率比較表

Minsup	1%	0.75%	0.50%	0.25%	0.10%
QPD	94sec	94sec	94sec	94sec	94sec
FP-Tree	66.6 sec	95.7 sec	157.7 sec	308.2 sec	403.2 sec
Apriori	2463.6 sec	4000.2 sec	7214.5 sec	14867.88 sec	20043.2 sec

2. **實驗 2**：交易記錄數量（D）對執行效率的影響。

依據資料庫交易記錄數量（D）為 D10K 至 D2000K，包含的項目個數（N）為 500，交易記錄長度（L）為 10 個項目，平均最大頻繁項目集之項目數（I）為 4，經過實驗，產生下圖的實驗結果。

由圖 13、圖 14 與表 12 可知隨著資料庫大小的增加，各演算法所花費的執行時間也明顯的變多，但由上可知，QPD 演算法執行速度較 Apriori 與 FP-Growth 演算法快。

3. **實驗 3**：測試總項目數量（N）的改變對執行效率的影響。

資料庫交易記錄數量（D）為 D100K，包含的項目個數（N）為 500 至 200，交易記錄長度（L）為 10 個項目，平均最大頻繁項目集之項目數（I）為 4，實驗中分別以最小

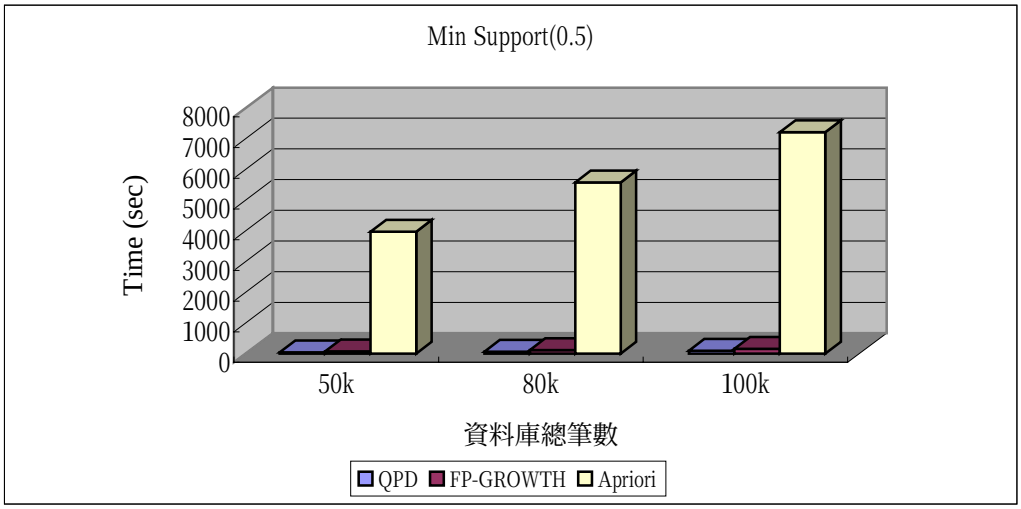

圖 13　QPD、Apriori 與 FP-Growth 演算法效率比較圖

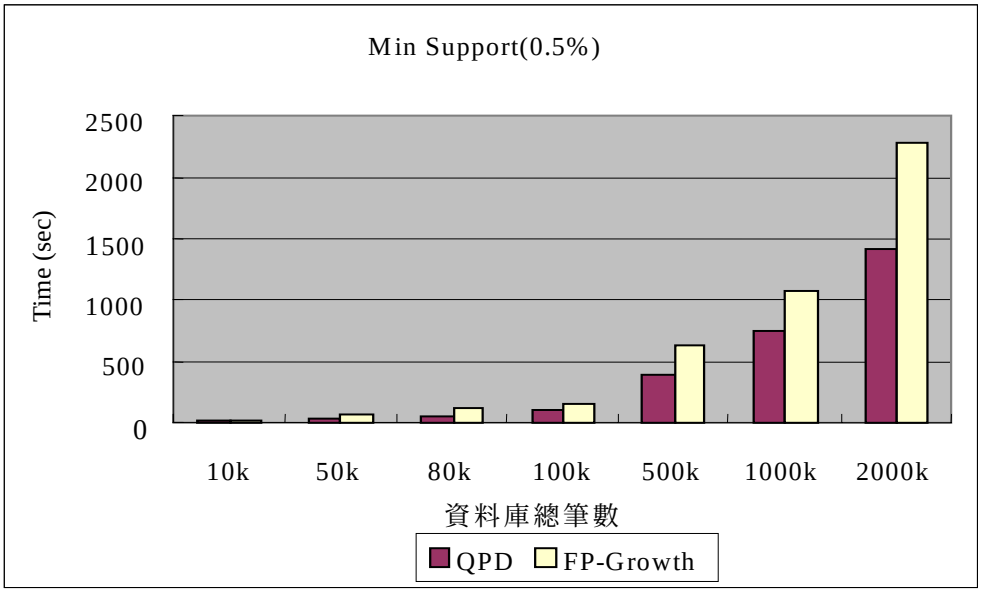

圖 14　QPD 與 FP-Growth 演算法效率比較圖

表 12　QPD、Apriori 與 FP-Growth 演算法效率比較表

資料庫名稱	D10K	D50K	D80K	D100K	D500K	D1000K	D2000K
QPD	8.7 sec	40 sec	61 sec	94 sec	381 sec	760 sec	1420 sec
FP-Tree	17 sec	72 sec	114 sec	157.7 sec	633 sec	1075 sec	2279 sec
Apriori	698 sec	3977 sec	5581 sec	7214.5 sec	略	略	略

支持度 1.5%與 0.5%進行測試，實驗結果如表 13~表 14 與圖 15~圖 18 所示。

由表 13~表 14 與圖 15~圖 18 可看出，隨著總項目數的變動，FP-Growth 演算法與 Apriori 演算法，所花費的執行時間會有明顯的改變。

4. **實驗 4**：測試交易記錄長度（L）的改變對執行效率的影響。

資料庫交易記錄數量（D）為 D50K，包含的項目個數（N）為 500，交易記錄長度（L）為 10 至 16 個項目，平均最大頻繁項目集之項目數（I）為 4，實驗中以最小支持度 0.5%進行測試，實驗結果如表 15 與圖 19 所示。

由表 15 與圖 19 可看出，隨著交易記錄長度的變動，QPD 演算法與 FP-Growth 演算法，所花費的執行時間明顯的改變，當交易記錄長度越長，演算法所花費的時間須較多，因為隨著長度增加，可拆解的子項目集個數也就大幅的增加，所以 QPD 所需對映與置換拆解的項目數也就較多，而 FP-Tree 在建 Tree 方面也會花費較多的時間。

5. **實驗 5**：測試頻繁項目集平均最大的項目數（I）的改變對執行效率的影響。

資料庫交易記錄數量（D）為 D50K，包含的項目個數（N）為 500，交易記錄長度（L）為 10 個項目，平均最大頻繁項目集之項目數（I）為 3 至 5，實驗中以最小支持度 0.5%進行測試，實驗結果如表 16 與圖 20 所示。

由表 16 與圖 20 可看出，隨著頻繁項目集項目數的變動，QPD 演算法與 FP-Growth 演算法，所花費的執行時間明顯的改變，隨著頻繁項目集項目數的越多，各演算法的執行時間明顯的增加。

6. **實驗 6**：測試新增資料對 QPD 演算法、FP-Growth 演算法 與 Apriori 演算法執行效率的影響。

分析每次新增 4K 的資料到 10K 的資料庫對各演算法所造成的影響。實驗結果如表 17 與圖 21~圖 22 所示。

QPD 演算法在探勘的過程中，對於每一筆交易記錄所產生的項目集，皆完整的保留下來，也因為這個特性，使得這兩種演算法可方便的作漸進式探勘，不管資料庫要新增或刪減，QPD 演算法皆不須重新掃瞄整個更新後的資料庫，只要對異動的資料進行探勘，便能快速的產生使用者所需要的關聯規則。FP-Growth 演算法，在第一次掃描資料庫後，

表 13　各演算法在不同總項目數下的效能比較表（Minsup 1.5%）

總項目數	N500	N400	N300	N200
FP-Tree	29.8 sec	34.9 sec	45.1 sec	53.5 sec
QPD	94.2 sec	93.5 sec	93.1 sec	93.3 sec
Apriori	784.6 sec	1058.3 sec	1290.7 sec	1474.1 sec

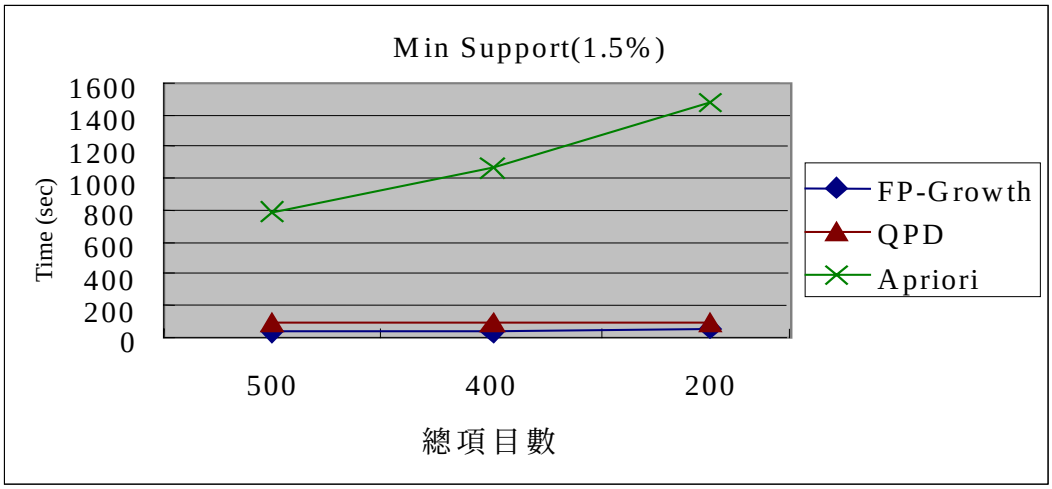

圖 15　三種演算法在不同總項目數下的效能比圖（Minsup 1.5%）

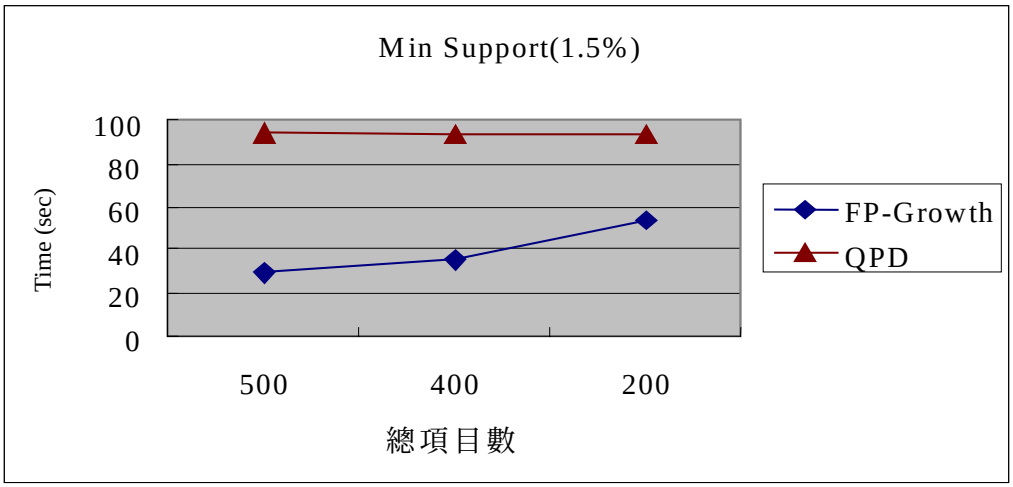

圖 16　QPD 與 FP-Growth 演算法在不同總項目數下的效能比圖（Minsup 1.5%）

表 14 各演算法在不同總項目數下的效能比較表（Minsup 0.5%）

總項目數	N500	N400	N300	N200
QPD	94 sec	93.1 sec	92.6 sec	93.5 sec
FP-Tree	157.72 sec	138.74 sec	135.80 sec	103.53 sec
Apriori	7214.45 sec	6071.92 sec	5353.33 sec	3443.05 sec

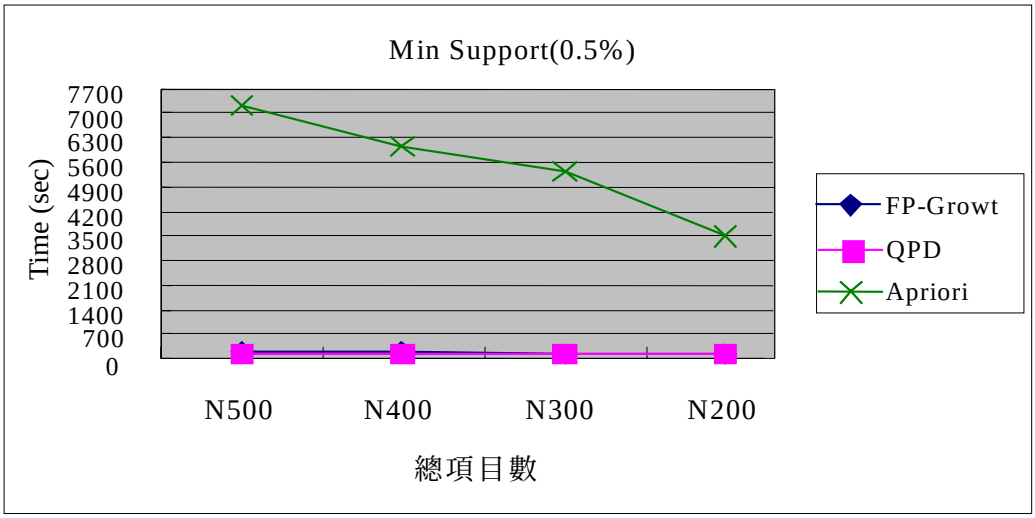

圖 17 三種演算法在不同總項目數下的效能比圖（Minsup 0.5%）

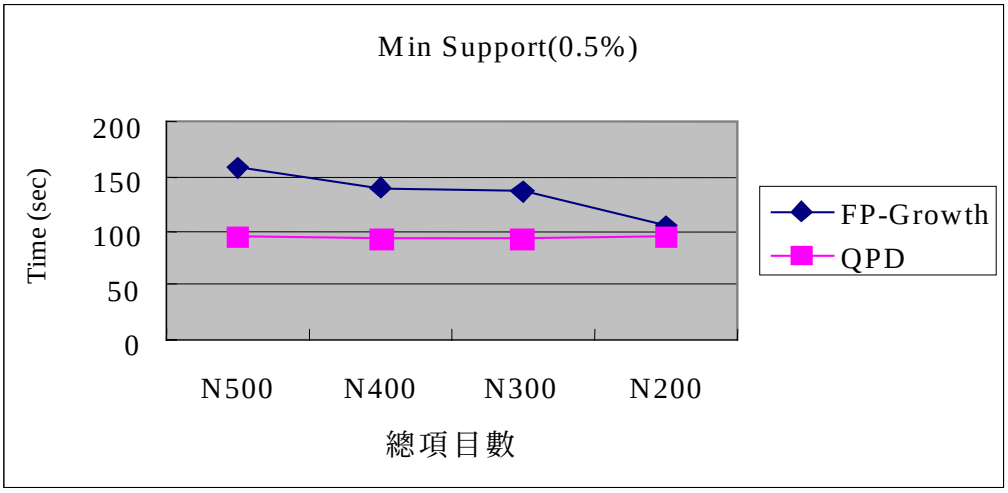

圖 18 QPD 與 FP-Growth 演算法在不同總項目數下的效能比圖（Minsup 0.5%）

表 15　各演算法在不同長度下的效能比較表（Minsup 0.5%）

交易記錄長度	L10	L12	L14	L16
QPD	40.6 sec	45.3 sec	65.2 sec	148 sec
FP-Tree	72 sec	81.2 sec	95.6 sec	152 sec

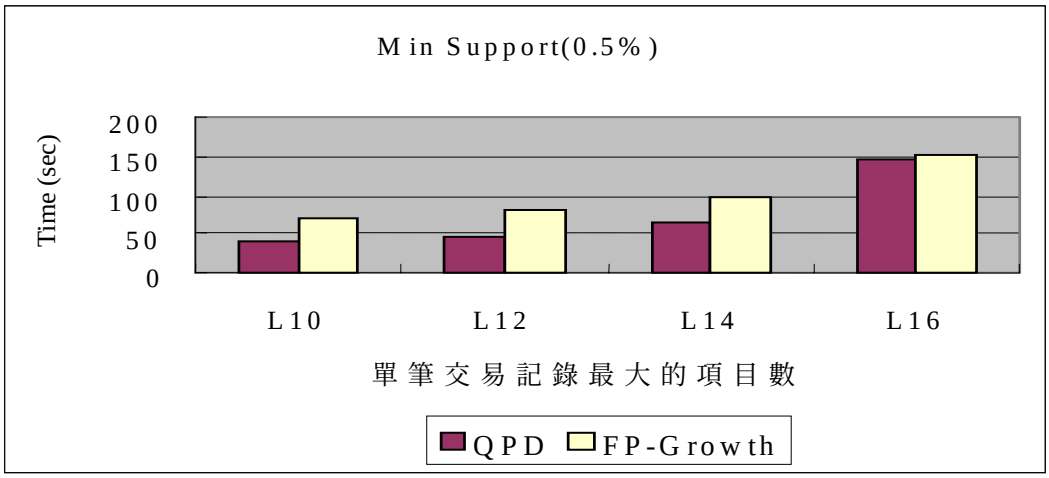

圖 19　QPD 與 FP-Growth 演算法在不同長度下的效能比圖（Minsup 0.5%）

表16　各演算法在不同頻繁項目集項目數下的效能比較表（Minsup 0.5%）

頻繁項目集平均最大項目數	I3	I4	I5
QPD	13.4 sec	40.6 sec	93.3 sec
FP-Tree	32.7 sec	74.9 sec	126.1 sec

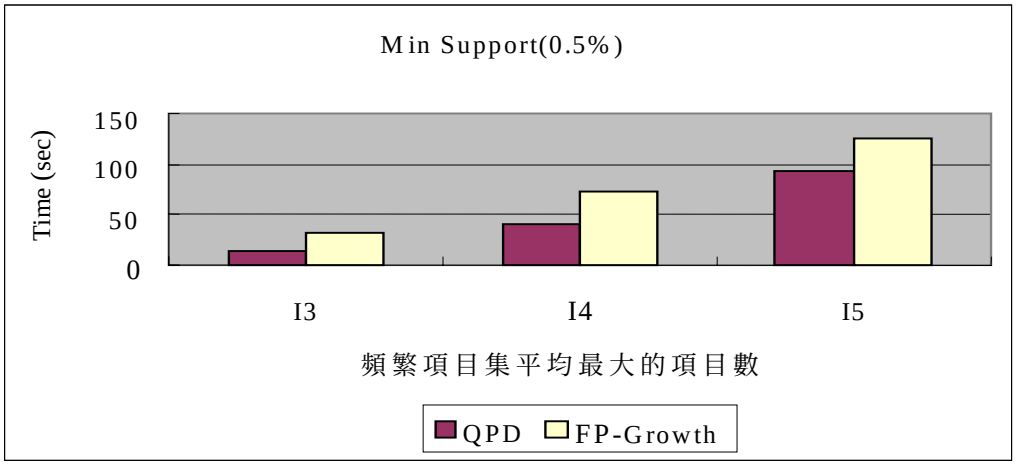

圖 20　QPD 與 FP-Growth 演算法在不同頻繁項目集項目數的效能比圖（Minsup 0.5%）

表17 各演算法在新增資料後的效能比較表（Minsup 1%）

新增後資料庫大小		原 10K	14K	18K	22K	26K	30K
I/O 時間	QPD_read	—	2.8 sec	3.1 sec	3.4 sec	3.5 sec	3.7 sec
	QPD_write	0.92 sec	1.39 sec	1.484 sec	1.49 sec	1.37 sec	1.34 sec
QPD 映對時間		8.7 sec	2.77 sec	3.14 sec	3.11 sec	3.36 sec	3.53 sec
QPD 總執行時間		9.62 sec	6.96 sec	7.724 sec	8 sec	8.23 sec	8.57 sec
FP-Tree 執行時間		6.49 sec	7.56 sec	10.61 sec	12.67 sec	13.55 sec	15.13 sec
Apriori 執行時間		218.20 sec	324.67 sec	414.88 sec	487.17 sec	589.13 sec	709.16 sec

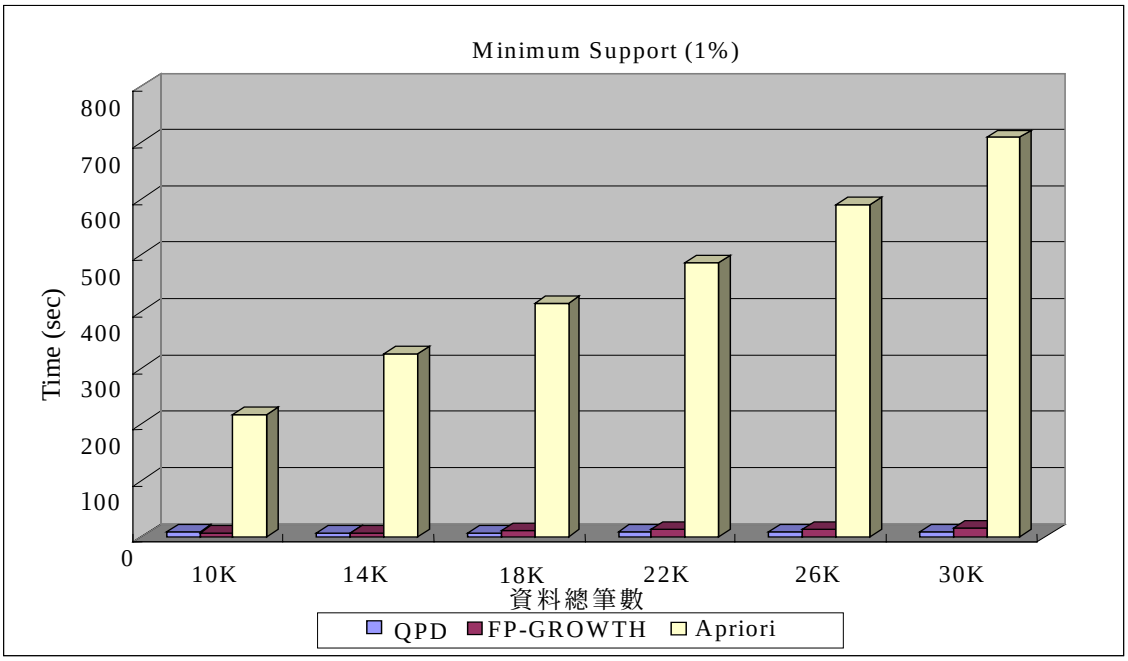

圖 21 QPD、FP-Tree 與 Apriori 演算法效率比較圖

會依據所找出的第一頻繁項目集，將資料庫的資料做適當的修剪與排序，所以有些項目的資訊並沒有完整的保留下來，這時如果資料庫有異動，則 FP-Growth 演算法必須對更新後的資料庫再作一次完整的探勘，如此才能得到正確的資訊。而 Apriori 演算法，在探勘的過程中會去除不滿足最小支持度的項目，所以項目集的資訊也沒有完整的保留下來，這時如果資料庫有異動，則 Apriori 演算法一樣必須對更新後的資料庫再作一次完整的探勘。

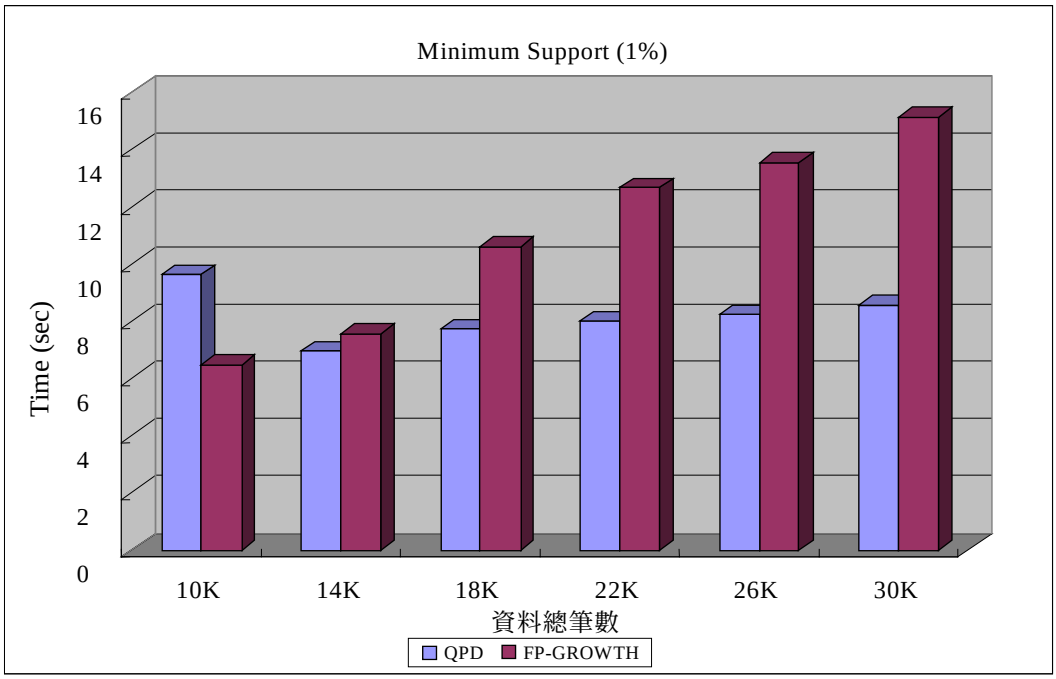

圖 22　QPD 與 FP-Growth 演算法效率比較圖

由圖 21、圖 22 與表 17 可看出實驗的結果，QPD 演算法只要對新增的資料進行探勘即可，所以每次新增 4K 的資料，QPD 演算法只需花費約 7.8 秒的時間，即可完成資料庫異動的探勘，然後再將其結果寫入硬碟中儲存，等到探勘新增的資料時再由硬碟中載入至記憶體。但可發現 I/O 之 read 時間明顯比 write 時間多，這是因為將文字檔資料載入至記憶體後，需將載入的資料還原至因素項目表，所以花費的時間明顯的比 write 來的多；而 FP-Growth 演算法與 Apriori 演算法則必須重新對更新後的整個資料庫作探勘，所以 FP-Growth 演算法與 Apriori 演算法花費的時間明顯的比 QPD 演算法來的多，也證明了 QPD 演算法在漸進式探勘上有良好的效能。綜合以上六組的實驗結果，可知 QPD 演算法執行效率相當優異，不論在支持度變動、資料庫大小、總項目數、交易記錄長度、頻繁項目集項目數、型樣產生器效能或是漸進式探勘的各個實驗中，皆優於其他的演算法。

肆、結論

一般用來作資料探勘的資料庫都非常龐大, 當進行資料探勘時需要攏長的時間掃描資料庫, 尤其是利用以 Apriori-Base 的演算法, 來做關聯分析時其效率是經常被詬病。由於 Apriori 最大的缺點就於執行效率不佳, 也就是產生每個候選項目時, 會重複掃瞄資料庫來計算出候選項目的支持度, 使得執行時間相當長。因此, 針對此缺點, 本研究提出 QPD 演算法改善, 1.只需掃描資料庫一次, 並利用型樣方式來提昇執行效率；2.利用型樣化方式來提昇執行效率；3.利用遮罩與布林模式來產生拆解項目因子型樣；4.當最小門檻值變動時, 不需重新探勘；5.資料庫有異動時, 可方便進行漸進式探勘。

透過本演算法做關聯分析, 其效能將優於以往 Apriori-Base 的演算法。此外, 關聯規則的推導過程中, 將不會重複產生多餘的候選項目組, 因此更勝於拆解模式的演算法, 並快速得到正確、有效用的資訊。由此能降低時間成本、快速反映市場需求, 是企業提昇競爭力的最大利基。

參考文獻

一、中文部份

1. 黃仁鵬、錢依佩與吳聲弘(2003, 6 月), 高效率之關聯規則探勘演算法－ICI（An Efficient Algorithm for Mining Association Rules-ICI), 第十四屆國際資訊管理學術研討會（Proceedings of the 14th international conference on information management), 中華民國資訊管理學會與中正大學資訊管理學系主辦。

二、英文部份

1. Agrawal, R., Imilienski, T., & Swami, A. (1993, May). Mining Association Rules between Sets of Items in Large Databases, In Proc. of the ACM SIGMOD Int'l Conf. on Management of Data, 207-216.

2. Agrawal, R., & Srikant, R. (1994, Sep.). Fast Algorithm for Mining Association Rules in Large Databases, In Proc. 1994 Int'l Conf. VLDB, 487-499, Santiago, Chile.

3. Brin, S., Motwani, R., Ullman, J. D., & Tsur, S. (1997). Dynamic Itemset Counting and Implication Rules for Market Basket Data, ACM SIGMOD Conference on Management of Data, 255-264.

4. Cabena, P., Hadjinian, P., Stadler, R., Verhees, J., & Zanasi, A. (1997). Discovering Data Mining From Concept to Implementation, Prentice-Hall Inc.

5. Chen, M. S., Han, J., & Yu, P. S. (1996). Data Mining: An Overview from a Database Perspective, IEEE Transactions on Knowledge and Data Engineering, 8(6).

6. Han, J., & Kamber, M. (2000). Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers.

7. Han, J., Pei, J., & Yin, Y. (2000). Mining Frequent Patterns without Candidate Generation, In Proc. of the ACM SIGMOD Int. Conf. on Management of Data, 1-12, Dallas, Texas, USA.

2004 年 10 月 29 日收稿

2005 年 01 月 25 日初審

2005 年 05 月 06 日複審

2005 年 06 月 07 日接受