利用 SQL 改良建構 FP-tree 之技術

趙景明[1] 戴玉娟[2]
chao@cis.scu.edu.tw Jennifer_Tai@walsin.com
東吳大學資訊科學系

摘　要

一般日常營運系統的交易資料庫蘊含著豐富的未知知識，例如客戶交易明細資料的資料探勘可以瞭解消費者的購物習性，由消費者的購買歷史，可分析出購買商品之間的關聯性，並可藉此組合相關的產品，實行交叉銷售。在資料探勘的演算法中，一個很重要的問題就是如何在大型資料庫裡有效率找出我們需要的資料。本研究結合資料庫與結構化查詢語言，應用於資料探勘演算法中一般化關聯法則（Generalized Association Rules）的 FP-tree 的建構過程，利用 SQL 語法來簡化 FP-tree 之資料建構及挖掘過程的複雜度。實驗結果顯示我們所提出的方法其建構的時間比傳統 FP-tree 演算法來得快。是一個能改善系統執行效能以提升現實應用的高效率一般化關聯法則探勘方法。

關鍵字：資料探勘、一般化關聯法則、結構化查詢語言、FP-tree

---

[1] 趙景明：民國79年取得美國愛荷華大學電腦科學博士。現任東吳大學資訊科學系專任教授。專長為資料庫、資料倉儲、資料探勘、WEB技術以及物件導向技術。

[2] 戴玉娟：民國 92 年進入東吳大學資訊科學系碩士在職專班就讀。主修為資料庫、資料倉儲、資料探勘以及分散式資訊系統

An SQL-based Improvement of the FP-tree Construction Technique

Ching-Ming Cha
Yu-Chuan Tai

Department of Computer and Information Science, Soochow University

Abstract

In general, the transaction database of the daily operation system contained lots of unknown knowledge. For example, from the detailed transaction data we can get the customer buying behavior, and from the historical buying data, we can analyze the relation of products, and also base on that, we can combine with the relative products for cross selling. In the deductive method for data mining, one of the most important things is how to retrieve the data effectively. In this paper we will integrate database technology with SQL and SQL to apply to the process of FP-tree from Generalized Association Rules of data mining. By using SQL to simplify the data structure of FP-tree. The result shows that our proposal is faster than the original FP-tree. This will be a highly effective method of data mining to improve on system execution performance.

Keywords: Data Mining, Generalized Association Rules, Structured Query Language, FP-tree

壹、緒　論

一、研究動機

低成本、高利潤一直是企業所追求的目標，企業在面對激烈的競爭，無不採取電腦化和網路作業來提高效益。利用資料探勘（Data Mining）的技術，可以幫助業界探勘龐大資料庫中所隱藏的有用資訊。資料探勘的技術是現今相當熱門的研究領域，特別是在關聯法則（Association Rules）技術方面的探討。透過關聯法則可以找出資料庫中某些商品項目間彼此的關聯性，若能善加利用這些資訊，則能幫助使用者發覺隱含的知識，如消費者的購物習性等提供給決策者參考。對於如何才能有效地推導出關聯法則，在以往已經有許多的方法相繼被提出，但大部份的演算法皆是在處理單一層次間的關聯法則。隨著商品項目的多樣化，找出的關聯法則數目可能會變的較少，更多隱藏的知識便不能被探勘出來。另外，上層的決策者往往沒時間處理瑣碎的訊息，他們所需要的是大方向性的資訊來幫助做決策。所以，提供一個能探勘多層次關聯法則的演算法便能滿足各方的需求。

傳統上在實作多層次關聯法則時，傾向將資料轉換成能夠對應到階層化架構的型式，然後再對各層次使用單層次關聯法則演算法來求取關聯法則。以往單層次演算法的做法，容易產生太多的候選商品項目組，因而需要多次比對資料庫的動作。資料探勘的處理都是面對龐大的資料庫，一個很重要的問題就是如何在大型資料庫裡快速找出我們所需要的資料。

二、研究目的

過去對於一般日常營運系統而言，資料的最終目的只求正確的儲存以及快速的運算，以求達成線上交易的快速回應需求。在大量交易資料中，蘊藏著許多知識是值得去探討與了解的，像是客戶交易明細資料是零售業中龐大且重要的資料庫。這些過去強調以正確計算與儲存為基本要求的日常營運系統，目前因為資料探勘技術的快速發展，從這些資料庫中挖掘出知識已不再是一件困難的事。但是這些資料庫具有資料欄位複雜及實際資料量龐大的特性，因此，如何利用一個有效的演算法，結合資料庫管理資料庫的特性進行資料探勘，且能讓管理者藉由產生出來的結果，透過簡單的模式，就可以做好管理的工作，將是一件重要且急待完成的工作。

資料探勘是從大量的資料中，發掘出潛在有用的資訊與知識的技術，而資料探勘技術中的關聯法則主要的功能乃是分析大量的交易資料，找到商品之間的相關資訊，以便提供企業主管做行銷上的規劃或商品設計上的決策，因隨著商品項目的多樣化，找出的關聯法則數目可能會變的較少，更多隱藏的知識便不能被探勘出來，故提供一個能探勘多層次關聯法則的演算

法日趨重要。

傳統上在實作多層次關聯法則時，傾向將資料轉換成能夠對應到階層化架構的型式，然後以 Candidate Generation-and-Test 方式找出頻繁集(Frequent Itemsets)，此方式需要大量的執行時間產生候選商品項目組，而且需要多次比對資料庫的動作。資料探勘的處理都是面對龐大的資料庫，如此反覆的搜尋資料庫，將造成 I/O 上大量時間的浪費。後來就有學者提出 FP-growth 演算法，不產生候選商品就可以利用 FP-tree 來產生頻繁集，大大節省了時間與空間。

本研究之目的，有鑑於目前一般企業日常營運資料庫或資料倉儲大都為關聯式資料庫，若能以結構化 SQL 語法，利用關聯式資料庫在處理資料的特性以集合(set)為運作方式，也就是 SQL 可以一次處理多筆符合資料集合的特性，有效率地執行資料清理以及資料運算的工作，以求快速建構出 FP-tree 來產生一般化關聯法的頻繁集。

貳、 文獻探討

一、 資料探勘

Wal-Mart 的成功關鍵來自於有效掌握顧客需求活動。大型資料庫普遍存在，例如：超級市場付款櫃檯的會員資料、信用卡刷卡機持卡人資料、醫療資料或是電話通訊紀錄，隨著市場競爭日趨激烈，如何有效運用這些資料顯得格外重要。因為資料儲存技術的進步，得以保存大量的歷史資料，因此，企業紛紛利用資料倉儲（Data Warehouse）儲存歷史資料，將資料經過一連串的處理程序後，進入倉儲內提供各式資料探勘的用途。資料探勘的結果通常提供決策分析用途，如何提供決策者在進行商業決策活動時的一項重要的參考：在最適時間與地點，提供顧客最適合的商品，將成為穩固顧客忠誠度與滿意度的最佳利基。

二、 關聯法則

隨著資料儲存技術的進步，使我們更容易從資料庫裡的資料萃取出來我們感興趣的資料。從資料庫轉換成知識的過程稱為知識發掘。知識發掘後產生的結果多應用在提供給決策者做決策的參考。而資料探勘是從大量的資料中，發掘出潛在有用的資訊與知識的技術。首先由 Agrawal 等人[3]提出探勘關聯法則，此方法經常應用於商業上的購物車分析(Market Basket Analysis; MBA)，藉由銷售點系統(Point of Sale; POS)所記錄的消費者購買歷史，可分析出購買商品之間的關聯性，並可藉此組合相關的產品，實行交叉銷售，提高行銷績效[4]。

關聯規則主要是要找出資料項間的關聯性，而關聯規則的形式可表示為：X→Y，其中 X 和 Y 分別代表項目的集合，即若購買項目集合 X 時，可能會再購買項

目集合 Y。若要找出規則 X→Y，則必須先計算項目集合 X 及項目集合 X∪Y 的支持度(Support)，意即資料庫中有幾筆資料包含此項目集合。接著必需確定 X∪Y 是否為頻繁集，若是，我們即可把 X∪Y 的支持度除以 X 的支持度，所得的值代表 X→Y 的信賴度(Confidence)。若信賴度超過最小信賴度(Minimum Confidence)，則關聯規則 X→Y 成立。探討關聯規則中，最重要的演算法為 Apriori。

在大型資料庫中，如何發現交易資料的關聯規則，Apriori 演算法是很重要的方法，關聯規則的探勘過程主要分為兩個階段：首先找出滿足最小支持度的所有項目組(Itemsets)，一項目組若由 k 個項目組成，k >= 0，則稱之為 k-項目組，以 $Itemset_k$ 表示之，若滿足最小支持度，則稱之為頻繁集，以 $Frequent_k$ 表示之，因為頻繁集亦稱為 Large Itemsets，故 $Frequent_k$ 通常以 L_k 表示；然後再以最小信賴度為條件，計算所有頻繁集所能形成的關聯規則。因為當資料項增加時，所要考慮的項目組合將呈指數成長，在這兩個步驟中，找出符合最小支持度的頻繁集，需要大量的執行時間。須對資料庫做多次的掃瞄，以找出符合最小支持度值的頻繁集，因此相當耗費時間。

J. Han, J. Pei, and Y. Yin [5] 在 2000 年提出 Frequent-Pattern tree (FP tree) 的 FP-Growth 演算法，不會產生出 Candidate Item Sets 就可以產生頻繁集，反覆針對資料庫中的每種 Pattern(即資料項組合)計算出它的出現次數(通稱為 Support)，此類方法通常會利用樹狀資料結構來表達原始交易資料，並將此結構存放於記憶體中，因此反覆處理的動作是針對記憶體中的樹狀資料結構，故速度較 Candidate Generation-and-Test 來得快許多，但需要較多記憶體。其探勘關聯法則的步驟如下：

(一) 掃描交易資料庫(表 1)，得到 L_1(表 2)。(假設 Support 次數>= 3)

表 1 交易資料庫

Tid	Items
100	A,C,D,F,G,I,M,P
200	A,B,C,F,L,M,O
300	B,F,H,J,O
400	B,C,K,P,N
500	A,C,E,F,L,M,P

表 2 L_1

Itemset	Support
C	4
F	4
A	3
B	3
M	3
P	3

(二) 將每一交易的產品項目依符合最小支持度的 L_1 由大至小排序如表 3 所示。

表 3 交易項目依 L_1 由大至小排序

Tid	Items	Frequent Items
100	A,C,D,F,G,I,M,P	C,F,A,M,P
200	A,B,C,F,L,M,O	C,F,A,B,M
300	B,F,H,J,O	F,B
400	B,C,K,P,N	C,B,P
500	A,C,E,F,L,M,P	C,F,A,M,P

(三) 掃描交易資料庫建構 FP-tree 如圖 1 所示，再利用 FP-tree 產生頻繁集。

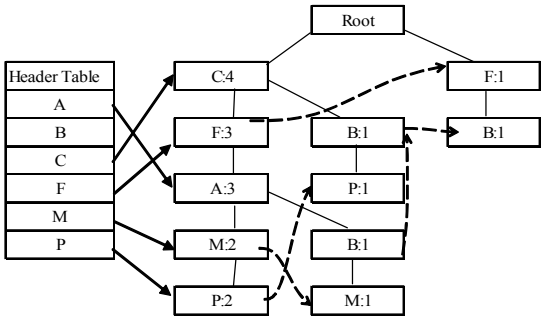

圖 1 FP-tree

三、一般化關聯法則

隨著商品項目的多樣化，找出的關聯法則數目可能會變少，更多隱藏的知識便不能被探勘出來，在資料庫組織中，本身就會存在一些抽象階層，例如產品項目的階層：筆記型電腦、桌上型電腦、電腦等，我們發現有趣的關聯通常發生在概念性的階層而不是在產品項目本身，以圖 2 分類為例，夾克是外套亦可以說是衣服，我們可能不會發現若顧客買夾克，則他同時也會買鞋子所存在關聯，但是我們可能可以發現 30%的消費者買了外套同時會買鞋子。此外還能夠察覺到 30%的消費者買了外套同時他會買登山靴；後者被表示在一個較低的觀念階層，但是通常附帶較特定及比較具體的資訊。

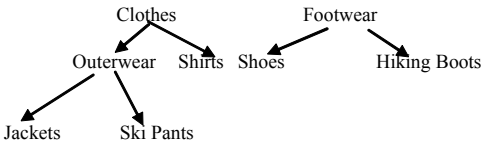

圖 2 產品項目階層例子

1995 年 Agrawal 等人[9]提出一個能夠探勘多層次關聯法則的演算法便能滿足各方的需求，稱為一般化關聯法則演算法，主要的將抽象階層的產品項目加入交易資料庫中，經由 Candidate Generation-and-Test 方式找出頻繁集，在 Candidate Generation 過程中加入過濾抽象階層的產品項目及剪除(Pruning)的作業，以加快產生頻繁集的速度。因為 Candidate Generation-and-Test 方式找出頻繁集需要大量的執行時間，故 2004 年 Pramudino 等人[6]提出以 FP-tree 的樹狀結構產生一般化關聯法則演算法，以不會產生出

Candidate Item Sets 就可以產生頻繁集，作法如同一般傳統 FP tree 只需要掃瞄交易資料庫兩次，最大的不同點是將抽象階層的產品項目加入交易資料庫中，利用樹狀資料結構來表達原始交易資料及抽象階層的產品項目，並且將此結構存放於記憶體中，反覆處理的動作是針對記憶體中的樹狀資料結構，其探勘一般化關聯法則的 FP-tree 演算法如圖 3 所示，並且以下列步驟說明：

（一） 掃描交易資料庫，將抽象階層的產品項目加入交易資料中，並且去除重複的產品項目，同時找出要大於或等於最小支持度的 L_1。

（二） 將每一交易的產品項目依 L_1 由大到小作排序。

（三） 第二次掃描交易資料庫建構 FP-tree。再由 FP-tree 中找出所有的頻繁集。

```
construct_fptree(database D, flist FList)
input : database D, F-list FList;
output : FP-tree FPtree;
{
1:while not eof(D) do
2: tranline = read_trans(D);
3: begin
4: add all ancestors of each item in tranline
5: removing any duplicates in tranline
6: end
7: o_trans = get_ordered_trans(Flist, tranline);
8: insert_fptree(FPtree, o_trans);
9:end while
}
```

圖 3 一般化關聯法則的 FP-tree 演算法

四、利用 SQL 建構關聯法則

利用資料探勘的技術探勘龐大資料庫中所隱藏的有用資訊時，有效整合資料探勘及資料庫系統是個趨勢，Agrawal [8]就曾指出資料探勘流程與資料庫系統整合有多種不同的架構選擇，因為現今資料庫管理系統的功能精進，而且提供平行作業加快執行速度。故不斷有學者提出利用 SQL 語法整合資料探勘的研究[1,7,8]，早期的研究大部份是利用 Apriori 演算法找出候選商品項目，需要多次比對資料庫的動作，如此反覆的搜尋資料庫，將造成 I/O 上大量時間的浪費，Agrawal[8]就提出 K-way Joins 來提高產生候選商品項目的產生速度。近年來不產生候選商品就可以產生頻繁集的演算法陸續有學者提出，大大節省了時間與空間。Xuequn Shang 等人[10]提出利用 SQL 語法結合以不產生候選商品方式來產生頻繁集的演算法，其作法有兩個步驟：

一、 以 FP-table 來存放 FP-tree 的相關資料，FP table 有三個欄位：產品項目(Item)、產品項目出現的次數(Count)及出現的路徑(Path)。在 FP-table 的建構過程中，在決定每一個產品項目的路徑時，都必須對已存在的 FP-table 中的路徑進行比對，檢查此路徑是否已經存在，若存在則將此產品項目的路徑之出現次數累加 1，否則將新增此產品項目、路徑、出現次數為 1 的資料至 FP-table，如此作法對全部交

易資料逐筆比對再一筆一筆資料對 FP-table 做資料更新。為了改善執行效能提出利用使用暫存表格存放每一個的產品項目的路徑，再利用暫存表格的資料以 SQL 指令一次完成 FP-table 的建置，大大簡化建構 FP-table 的複雜度亦證明提升執行效能。但是原文作者並未詳細說明暫存表格如何產生及實際完成之技術。

二、 利用 FP-table 表格取代傳統存放於記憶體中的 FP-tree 來產生頻繁集。主要步驟有 3 點:

（一） 對 FP-table 中每一個產品項目建立 Conditional Pattern Base Table。

（二） 對每一個 Conditional Pattern Base Table 分別建立其 Conditional FP-table。

（三） 對 Conditional FP-table 進行挖掘，並逐次增加包含在 Conditional FP-table 的頻繁集，當 Conditional FP-table 中包含一條路徑時，就找到該產品項目的所有的頻繁集。

利用 FP-table 來產生頻繁集的過程中，因為要對每一筆的產品項目產生 Conditional Pattern Base Table 及 Conditional FP-table，無法利用一個 SQL 語法就完成，使用者端及伺服器之間需要多次的溝通及網路傳輸，故利用表格 FP-table 來產生頻繁集的執行效能不及以傳統方式存放於記憶體中的樹狀資料結構之 FP-tree 來得快。

參、 建構一般化關聯法則的 FP-tree

為達到本研究目的，我們利用簡單的結構化 SQL 語法與結合關連式資料庫在處理資料的特性，有效率地執行資料清理以及資料運算的工作，充分使用資料庫的索引架構及聚合函數直接存取資料的方法，並且參考 Xuequn Shang 等人[10]利用暫存表格簡化 FP-tree 之資料建構及挖掘過程的複雜度，同時亦證明能夠提升執行效能。接下來以下列六個步驟說明：

一、 將產品項目階層加入到相對應的每一筆交易項目中，並且將產生的結果放入一個暫存表格；利用一句 SQL 的 INSERT INTO 語法，善用 SELECT 子句就可以完成，此時資料庫管理系統可以將查詢透過自己對資料的統計，經過最佳化的處理後，以最有效的方式找出符合的資料存入至表格中。

二、 將步驟一存放在暫存表格中所重覆的產品項目去除。因為購買的產品項目可能對應到相同的產品項目階層，利用一句 SQL 的 INSERT DISTINCT 語法一次就可以將所有重覆的商品項目去除。

三、 找出步驟二的產品項目 L_1 並且存放於表格中，這個方式的優點使得

不容易訂定最小支持度門檻值, 因為 L_1 由大到小排序存放於表格中, 得以能讓原本沒有程式經驗的使用者, 透過簡單的介面很容易地經由產品項目購買出現次數多寡而訂定出比較有意義的最小支持度。利用簡單的結構化查詢語言而不需要撰寫任何程式下, 就可以重覆執行產生出符合最小支持度的 L_1, 利用 SQL 中的欄位函數 COUNT()的運算可以幫助計算每一個產品項目出現的總次數, 若同時與 ORDER BY COUNT() 就可以很容易產生 L_1 且按照 L_1 由大到小自動排序, 免除了資料排序上的麻煩。

四、 將步驟二所產生之每一個交易的產品項目依符合最小支持度的 L_1 由大到小排序, 利用 SQL 中的 SELECT 語法, 若同時與 WHERE 條件子句結合, 則可進而找出某個屬性值的個數, 再與 ORDER BY 子句就可以將每一個交易的產品項目依 L_1 由大到小排序。

五、 建構 FP-table：

（一） 產生一個暫存表格存放步驟四所產生的每一個交易中的產品項目的出現路徑。暫存表格有兩個欄位；產品項目、路徑；以表 3 的 Tid 100 交易中依符合最小支持度的 L_1 由大到小排序的產品項目為 C、F、A、M、P 為例說明；

1. 排序第一順位的產品項目 C, 新增一筆資料到暫存表格, 產品項目欄位為 C, 路徑欄位為為 Root：並且保留產品項目 C 的路徑資料, 以便後面產品項目使用。
2. 排序第二順位的產品項目 F, 它的前面為產品項目 C, 新增一筆資料到暫存表格, 產品項目欄位為 F, 路徑欄位為 Root + C。表示產品項目 F 的路徑由 Root 經過 C。
3. 排序第三順位的產品項目 A, 它的前面為產品項目 F, 新增一筆資料到暫存表格, 產品項目欄位為 A, 路徑為 Root + C + F。表示產品項目 A 的路徑由 Root 經過 C 再到 F。
4. 排序第四順位的產品項目 M, 它的前面為產品項目 A, 故新增一筆資料到暫存表格, 產品項目欄位為 M, 路徑為 Root + C + F +A。
5. 排序第五順位的產品項目 P, 它的前面為產品項目 M, 故新增一筆資料到暫存表格, 產品項目欄位為 P, 路徑為 Root + C + F + A +M。
6. 每一個交易中排序第一順位的產品項目其路徑設定為 Root。其他順位的產品項目只要記錄前一順位的路徑, 而不需要經過複雜的比對及計算, 就可以貯存每一個產品項目的路徑, 依此類推產生出每一個

產品項目的路徑，雖然需要較大的暫時貯存空間，但卻大大節省比對及計算的時間及繁複。

（二） 利用一句簡單 SQL 的 INSERT INTO 語法，加上 SELECT 子句就可以一次完成將相同的產品項目及路徑的資料，累加其出現次數，並且將每一個產品項目的路徑及出現次數的資料新增至 FP-table 表格中。

六、 不需要掃瞄全部交易資料庫而是利用 FP-table 表格中存放每一個產品項目的路徑及出現次數之資訊，因為 FP-table 資料量遠比全部交易資料量少得多，故可以快速建構出 FP-tree，進而以 FP-tree 找出頻繁集。

在以範例來解說建構 FP-tree 六點步驟之前，我們先說明主要使用的 Table 表格有：

一、 交易資料庫：表格名稱為 Source，主要存放交易明細資料，包含：交易項目及產品項目兩個欄位(表 4)

表 4 Source 表格

欄位名稱	中文名稱
Tid	交易項目
Item	產品項目

二、 暫存表格: 表格名稱為 S_temp 存放交易項目及抽象階層項目，包含：交易項目及產品項目兩個欄位(表 5)

表 5 S_temp 表格

欄位名稱	中文名稱
Tid	交易項目
Item	產品項目

三、 暫存表格: 表格名稱為 F，存放 S_temp 的 L_1，產品項目依出現次數由大至小排序新增至 F 表格中(表 6)。

表 6 F 表格

欄位名稱	中文名稱
Item	產品項目
Cnt	出現次數

四、 暫存表格：表格名稱為 FP_temp，存放每一個的產品項目的路徑(表 7)。

表 7 FP_temp

欄位名稱	中文名稱
Item	產品項目
Path	路徑

五、 FP-table ：存放每一個的產品項目的路徑，與 FP_temp 不同之處為 FP-table 表格中同一產品項目相同路徑只會貯存一筆且累計其出現之次數(表 8)。

表 8 FP-table

欄位名稱	中文名稱
Item	產品項目
Cnt	出現次數
Path	路徑

我們以圖 2 產品項目階層及交易資料庫(表 9)為範例，存放在資料庫的資料模式如表 10 及表 11。

表 9 範例交易資料庫

Tid	Items Bought
100	Shirt
200	Jacket, Hiking Boots
300	Ski Pants, Hiking Boots
400	Shoes
500	Shoes
600	Jacket

表 10 範例交易資料庫存放在表格 Source 的 SC(Single-column) 資料模式

Tid	Items Bought
100	Shirt
200	Jacket
200	Hiking Boots
300	Ski Pants
300	Hiking Boots
400	Shoes
500	Shoes
600	Jacket

表 11 圖 2 產品項目階層存放在表格的資料模式

Item	Level 1	Level 2
Jacket	Clothes	Outerwear
Shirt	Clothes	
Ski Pants	Clothes	Outerwear
Hiking Boots	Footwear	
Shoes	Footwear	

接下來我們以範例來解說建構 FP-tree 六個步驟，配合 Pramudino 等人[6]所提出產生一般化關聯法則 FP-tree 的演算法如圖 3 之說明。

一、 將產品項目階層加入到相對應的每一筆交易項目中。

```
INSERT INTO S_temp
SELECT Source.tid, S_prod.level_1
FROM Source, S_prod
WHERE Source.Item = S_prod.Item
```

利用上述的 SQL 語法替代圖 3 一般化關聯法則的 FP-tree 演算法敘述行號 4，一次就直接將商品項目階層加入到相對應的每一筆交易項目中，表 9 的 Tid 100 交易中購買襯衫，襯衫的概念性的階層為衣服，故將衣服加入到交易 Tid 100 中。

二、 將表格 S_temp 中重覆的商品項目去除。

```
SELECT DISTINCT *
FROM S_temp ORDER BY Tid
```

利用上述的 SQL 語法替代圖 3 一般化關聯法則的 FP-tree 演算法敘述行號 5，可以一次將重覆的商品項目去除。

三、 讀取 S_temp 交易資料庫找出 L_1 存放在 F Table 中。

```
INSERT INTO F SELECT Item, Count(*)
FROM S_temp
GROUP BY Item
HAVING Count(*) >= Minsup
ORDER BY Count(*) DESC
```

利用上述 SQL 語法可以將在資料庫 S_temp 中大於或等於最小支持度的商品項目由大到小新增到 F table 中，結果如表 12 所示。

表 12 L_1

Items	Items Bought
Clothes	4
Footwear	4
Outerwear	3
Shoes	2
Hiking Boots	2
Jacket	2

四、 將每一交易項目依符合最小支持度的 L_1 由大至小排序。

```
SELECT S_temp.Tid, S_temp.Item
FROM S_temp, F
WHERE S_temp.Item = F.Item
ORDER BY S_temp.Tid, F.Cnt DESC
```

利用上述的 SQL 語法替代圖 3 一般化關聯法則的 FP-tree 演算法敘述行號 7，一次就產生交易資料庫中大於或等於最小支持度的產品項目，而且按照出現次數由大到小排序，小於最小支持度的產品項目就刪除，結果如表 13 所示。

表 13 (Ordered) 頻繁集

Tid	Items Bought
100	Clothes
200	Clothes
200	Footwear
200	Outerwear
200	Hiking Boots
200	Jacket
300	Clothes
300	Footwear
300	Outerwear
300	Hiking Boots
400	Footwear
400	Shoes
500	Footwear
500	Shoes
600	Clothes
600	Outerwear
600	Jacket

五、 建構 FP-table：首先使用暫存表格 FP_temp 存放每一個的產品項目的路徑(表 14)，再利用下列 SQL 指令一次新增資料至 FP-table 中，使用欄位函數 COUNT() 及 GROUP BY 支援計算出相同 item 及 path 的出現的次數如表 15 所示。

```
INSERT INTO FP-table
SELECT Item,Count(*) ,path from
FP_temp
GROUP BY item, path
```

表 14 表格 FP_temp 存放每一個產品項目的路徑

Item	path
Clothes	Root
Clothes	Root
Footwear	Root+Clothes
Outerwear	Root+Clothes+Footwear
Hiking Boots	Root+Clothes+Footwear+Outerwear
Jacket	Root+Clothes+Footwear+Outerwear+Hiking Boots
Clothes	Root
Footwear	Root+Clothes
Outerwear	Root+Clothes+Footwear
Hiking Boots	Root+Clothes+Footwear+Hiking Boots
Footwear	Root
Shoes	Root+Footwear
Footwear	Root
Shoes	Root+Footwear
Clothes	Root
Outerwear	Root+Clothes
Jacket	Root+Clothes+Outerwear

六、 FP-table 存放每一個產品項目的路徑及出現之次數，故可以很輕易由 FP-table 中轉換成 FP-tree，而可以不用掃瞄整個交易資料庫。

表 15 表格 FP-table 存放每一個的產品項目的路徑及出現次數

Item	Count	path
Clothes	4	Root
Footwear	2	Root + Clothes
Outerwear	2	Root + Clothes + Footwear
Hiking Boot	2	Root + Clothes + Footwear + Outerwear
Jacket	1	Root + Clothes + Footwear + Outerwear + Hiking Boots
Footwear	2	Root
Shoes	2	Root + Footwear
Outerwear	1	Root + Clothes
Jacket	1	Root + Clothes + Outerwear

肆、 實驗與分析

本研究進行一系列的實驗以評估前述利用結構化 SQL 語法建構 FP-tree 的效率。實驗的環境為 windows XP professional，硬體為 Intel Pentium 4、1.2 GHz CPU 及使用 SQL server 2000 個人版資料庫管理系統，程式以 Delphi 語言撰寫。使用者端透過開放資料連結 (Open Database Connectivity, ODBC) 與資料庫連結。

實驗用的測試資料使用模擬真實環境的模擬交易資料，每一個產品項目以一個數字代表，利用以 Agrawal [2]所使用的方式產生，此方法為日後多數的探勘關聯法則研究所採用。在模擬資料中我們定義了以下的參數：

D：交易資料的總筆數。

T：每筆交易的平均長度。

N：交易資料庫中的產品項目的個數。

執行時間會隨著下列原因而增加：

一、 交易長度增加。

二、 交易資料量增加。

三、 Minimum support 門檻值大小。

本研究針對這三點加以測試比較以 SQL 語法建構 FP-tree 及傳統方式建構 FP-tree 這兩種方式之執行效率。傳統方式如圖 3 演算法之說明，利用樹狀資料結構來表達原始交易資料，並將此結構存放於記憶體中，在記憶體中將產品項目階層加入到相對應的每一筆交易項目中，並且將重覆的商品項目去除，再將每一交易項目依符合最小支持度的 L_1 由大至小排序。在決定每一個產品項目的出現路徑時，都必須對所有的產品項目作運算，以計算出此產品項目的路徑及出現次數，經過兩次掃瞄交易資料庫建立出 FP-tree。

我們假設交易資料庫中的產品項目的個數 N 為 1000 個，每一筆交易的產品項目是利用程式隨機產生 1 到 1000 之間(含)亂數之整數表示，實驗所使用的資料設定如表 16 所示，依表 16 之數據設定進行實驗，在相同資料量下執行效能分析運算。利用程式記錄建構 FP-tree 的運算時間（Execution Times）。時間的記錄方法為將每個資料量開始讀取到完成 FP-tree 建置的執行時間。繪製成圖 4 圖 6 之效能比較。

表 16 模擬測試實驗設定

實驗名稱	測試數據			
不同平均交易長度	5	9	13	20
不同交易量	250K	350K	450K	550K
不同最小支持度(出現次數)	50	200	500	800

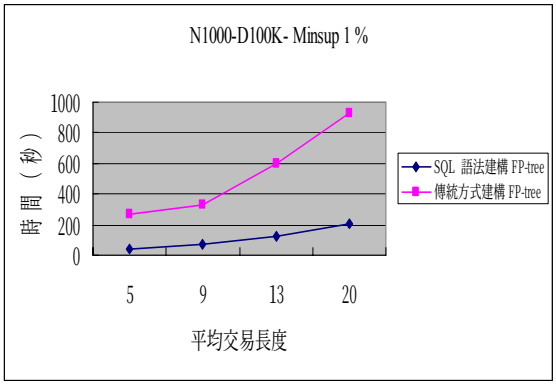

圖 4 不同平均交易長度之比較

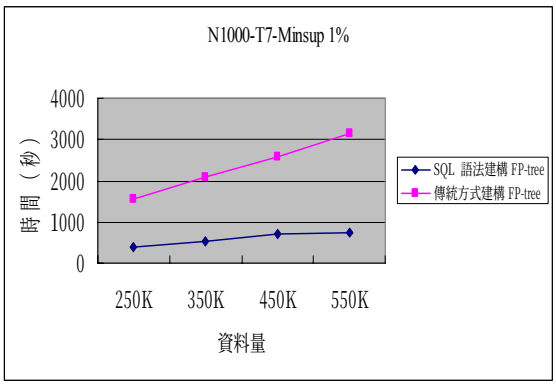

圖 5 不同交易資料量之比較

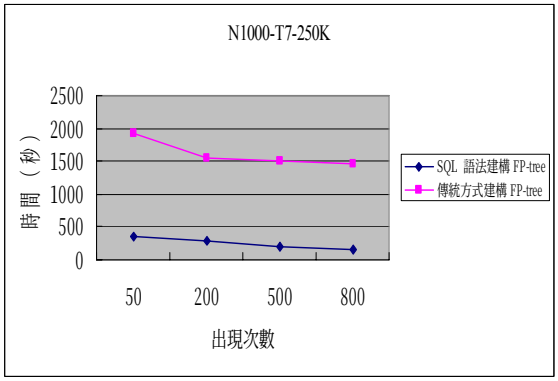

圖 6 不同最小支持度之比較

由表 16 之不同平均交易長度實驗中，資料庫的參數為 N=1000，D= 100K，Minimum Support = 1%。隨著每一筆交易的產品項目的個數增加，兩種演算的總花費時間都漸漸成長，由實驗中得知當平均交易長度大於 10 時，傳統方式建構 FP-tree 之執行時間逐漸拉長，而利用 SQL 語法建構 FP-tree 方法部分，運算時間較短。

由表 16 之不同交易資料量實驗中，資料庫的參數為 N=1000，T= 7，Minimum Support = 1%。隨著交易資料的總筆數增加，兩種演算的總花費時間都漸漸成長，由實驗中得知當交易資料的總筆數逐漸增加時，傳統方式建構 FP-tree 之執行時間大幅拉長，而利用 SQL 語法建構 FP-tree 方法部分，則運算時間明顯較短。

由表 16 之不同 Minimum Support 實驗中，資料庫的參數為 N=1000，T= 7，D= 250K。Minimum Support 愈小，則所產生的產品項目的節點就愈多，故兩種演算的

總花費時間都隨著 Minimum Support 數值的增加而執行的時間逐漸減少。

對於利用 SQL 語法建構 FP-tree 以及傳統方式建構 FP-tree 之執行效能，我們有以下的分析：

一、 傳統方式建構 FP-tree 需要掃瞄全部交易資料庫兩次，每次都要逐一對資料做比對、排序、運算的動作，而利用 SQL 語法建構 FP-tree 僅掃瞄全部交易資料庫一次，將每一個的產品項目的路徑存放到暫存表格，不需要對資料做複雜的運算動作。因為減少一次對全部交易資料庫進行掃描，故隨著資料個數的漸增後，利用 SQL 語法建構 FP-tree 的運算時間明顯較短。

二、 資料量、交易長度的增加及 Minimum Support 門檻值的大小都會影響 FP-tree 的節點數目之增加，由實驗得知隨著 FP-tree 的節點數目之增加，SQL 語法建構 FP-tree 時間會增加，但是花費時間增加不明顯，因為利用 SQL 語法，一次就處理多筆符合資料集合的效益開始展現，分析結果來自於運用資料庫的方法加速了運算能力。

三、 隨著 FP-tree 的節點數目的漸增後，兩種演算的總花費皆漸漸成長，在利用 SQL 語法建構 FP-tree 方法部分，運算時間較短，由於應用暫存表格使得資料庫管理系統可以將查詢透過自己對資料的統計，經過最佳化的處理後，以最有效的方式找出符合的資料存入至表格中，加速了 FP-tree 的建構過程。

伍、 結論

FP-tree 的建構過程中，必須對所有的產品項目作比對，將產品項目階層加入到相對應的每一筆交易項目中，並且將重覆的商品項目去除，再將每一交易項目依符合最小支持度的 L_1 由大至小排序。在決定每一個產品項目的出現路徑時，都必須對所有的產品項目作運算，以計算出此產品項目的路徑及出現次數，其所花費的時間為 O(n)，所以當資料量愈大時，執行時間就明顯增加。

以結構化 SQL 語法為基礎，可以一次處理多筆符合資料集合的特性建構 FP-tree，充分運用資料庫的特性與方法，能有效地解決處理大量資料時所遭遇的困難，如排序問題與資料必須存放於主記憶體的問題，而且善用暫存表格使得資料庫管理系統可以將查詢透過自己對資料的統計，經過最佳化的處理後，以最有效的方式找出符合的資料存入至表格中，有效地提昇 FP-tree 的建構效能。

參考文獻

[1] 張淑珍,「利用一次性的 SQL 改良決策樹建立信用卡審核之信用評等」，私立東吳大學資訊科學系，碩士論文，2005。

[2] Agrawal, R., R. Srikant, “Fast Algorithms for Mining Association Rules ,” Proceedings of the 20th VLAB Conference Santiage, Chile, 1994, pp. 487-499.

[3] Agrawal, R., T. Imielinski, and A. Swami, “Mining Association Rule between Sets of Items in Large Databases,” Proceedings of the ACM SIGMOD International Conference Management of Data, Washington D.C., 1993, pp. 207-216.

[4] Berry, L. L., “Relationship Perspectives,” Journal of the Academy of Marketing Science, Vol. 23, No. 4, 1995, pp.236-245.

[5] Han, J., J. pei, and Y. Yin, “Mining frequent patterns without candidate generation”. In Proc. of the ACM SIGMOD Conference on Management of data, 2000.

[6] Pramudiono, I., M. Kitsuregawa, “FP-tax, Tree Structure Based Generalized Association Rule Mining” Communications of the ACM, 2004, pp. 60-63.

[7] Pramudiono, I., T. Shintani, T. Tamura and M. Kitsuregawa. “ Parallel SQL based associaton rule mining on large scale PC cluster: performance comparision with directly coded C implementation, ” In Proc. Of Third Pacific-Asia Conf. on Knowledge Discovery and Data Mining, 1999.

[8] Sarawagi, S., S. Thomas, R. Agrawal, “Integrating Association Rule Mining with Relational Database Systems: Alternatives and Implications,”

[9] Srikant, R., R. Agrawal, “Mining Generalized Association Rules,” In Proc. Of VLDB Conference, 1995 .

[10] Xuequn, S., K. U. Sattler, and I. Geist, “SQL Based Frequent Pattern Mining without Candidate Generation”, ACM, 2004.