以 XQuery 為基礎之漸進式 XML 關聯規則探勘方法

董建弘　　曾守正
國立高雄第一科技大學 資訊管理系
u9324717@ccms.nkfust.edu.tw　　imfrank@ccms.nkfust.edu.tw

摘要

XML 已逐漸成為組織間在網際網路上交換資料的標準，未來將勢必有大量的資料會以 XML 格式儲存著。雖然在關聯式資料庫系統中已經有很成熟的資料探勘技術，但對於 XML 資料的探勘還像是處女地一般。本論文嘗試提出一個方法，不僅能探勘出 XML 文件中的關聯規則，並且可以在 XML 文件有新增刪除修改等異動時，只需針對異動的部份做計算，便可以與前次探勘的結果整合成新的項目集資料，供使用者查詢關聯規則，我們稱這個演算法為－IXARM (Incremental XML Association Rules Mining)。經由實驗證明本演算法的正確性以及在處理資料異動時效能上的表現相當優異。

關鍵詞：資料探勘、關聯規則、XML、XQuery、漸進式方法。

1. 前言

自從 1998 年 W3C (World Wide Web Consortium) 公佈 XML 1.0 版規格以來[10]，全球資訊網 (World Wide Web) 的 XML 應用趨勢就急速地增加。愈來愈多 XML 資訊充斥在網路上，且已逐漸成為組織間在網際網路上交換資料的標準。同時 W3C 也持續修訂 XML 規格，在 2004 年同時公佈 XML 1.0 (Third Edtion) 與 XML 1.1 使 XML 規範與功能更趨於完備。由此可見，資訊的傳遞與交換將逐漸根基於 XML，未來將勢必有大量的資料會以 XML 格式儲存著，因如何有效存取及管理 XML-based 的資料也愈顯重要。

隨著 XML 的發展趨勢漸次成長，可以預見未來將「XML 資料」轉變成有價值的「資訊」將是另一波的發展重點。但依據 KDnuggets 於與 2006 年對於資料探勘所使用的資料形態所做的調查，可知目前用來用來進行資料探勘的資料格式仍以文字檔、統計工具軟體以及關聯式資料庫系統等為主，如**表 1**所示，但對於 XML 資料的關聯規則探勘還像是處女地一般。因此，我們希望提出一個能直接在 XML 文件中探勘出關聯規則的方法。

基於 XML 文件分散且獨立的特性，我們所需要的探勘技術不僅要能從一份文件中找出關聯規則，而且要能同時整合不同文件的探勘結果，以期在資料異動頻繁與資料持續成長的情況下，能做到關聯規則的漸進式探勘 (Incremental Mining)，讓資料庫更動後不需要重新掃瞄所有資料，而僅需要掃瞄異動部份，以及前次探勘保留的一些相關資訊，找出更新後資料庫中的關聯規則即可完成。

表1　Data Storage for Data Mining(2006)

Data Storage for Data Mining	Radio
Text files (e.g. tab or comma delim) (75)	52.8%
Data mining system format (SAS, SPSS, arff) (57)	40.1%
Excel (28)	19.7%
Oracle (25)	17.6%
SQL Server (15)	10.6%
mySQL (12)	8.5%
Other format (10)	7.0%
Other commercial DBMS (7)	4.9%
Other free DBMS (4)	2.8%

(資料來源：http://www.kdnuggets.com/polls)

2. 文獻探討

2.1　關聯規則

關聯規則是用來描述資料項目間關聯性的一種型式，最早由 Agrawal [2] 等人於 1993 年所提出，主要是用來尋找資料庫中項目 (Item) 之間的關聯性。目前已有許多不同的演算法相繼被提出，如圖 1所示[1][4][5]。其中最廣為人知的演算法便是 Apriori 演算法，而許多演算法也大多以此為基礎來改良。

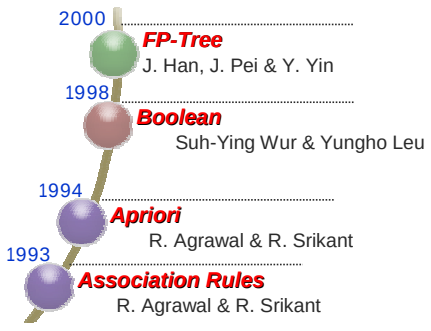

圖1　Association Rules 的發展

關聯規則主要的問題定義如下：令 $I=\{i_1,i_2,\cdots,i_m\}$，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（$T\subseteq I$），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 $X \rightarrow Y$ 表示，其中 $X \subseteq I$，$Y \subseteq I$ 且 $X \cap Y = 0$。衡量關聯規則是否成立的標準有二，一是支持度、二是信賴度。

(1) 支持度(Support；s) ：支持度的定義為「D 中包含 X 且包含 Y 交易記錄筆數和 D 中所有交易記錄筆數的比例」，公式 (2-1) 如所示，。

$$S(X \cap Y) = \frac{|X \cap Y|}{|D|} \qquad (2\text{-}1)$$

(2) 信賴度(Confidence；c) ：信賴度的定義為「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 (2-2) 所示。

$$C(Y \mid X) = \frac{S(X \cap Y)}{S(X)} \qquad (2\text{-}2)$$

2.2 XQuery

為了制定 XML 的查詢方式，W3C 於 2000 年的 1 月首先提出了 XML Query Requirements，討論對 XML 資料做查詢的需求，並於 2001 年 6 月 W3C 以 Jonathan Robie 等人共同開發的 Quilt 語言[3]為基礎定義 XQuery，此後 XQuery 便為 XML 查詢語言的標準，目前最新版本已於 2006 年 6 月 8 日出版，發展的過程概略如**圖 2**所示。

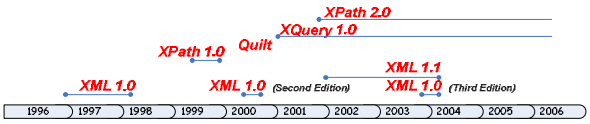

圖2 XQuery 的發展

一個 XQuery 的敘述式為一組 FLWOR (發音同 “flower”) 表示法，是由 FOR、LET、WHERE、ORDER BY 及 RETURN 子句等結構而成，並以特定的順序組合而成如**圖 3**所示。

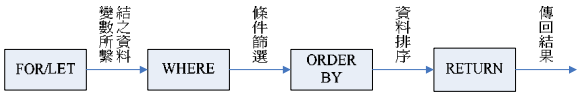

圖3 FLWOR 表示法之資料結構順序

FOR 讓變數可以遞迴取得 (iterate over) 一個路徑表示法的結果，而 LET 則是將變數直接與某一個的路徑表示法結合 (binding)，WHERE 則允許對變數做條件的限制，ORDER BY 可依據條件重新排序 FOR 語法的元素進而影響接下來 RETURN 的輸出，RETURN 則可以根據原有的、或是建構新的 XML 元素為查詢的輸出。

2.3 XML 資料的關聯規則探勘

在 XQuery 尚未出現以前，並無法直接由 XML 探勘關聯規則，當時常用的做法都必須要對 XML 進行探勘的前處理 (pre-processing) 或後處理 (post-precessing) 的過程，常用的作法如：將 XML 資料轉入關聯式資料庫，再以關聯式資料庫的方法進行關聯規則的探勘。

XQuery 於 2001 年成為 W3C 的標準後，相關廠商也陸續開發相關的應用工具，Wan 與 Dobbie 於 2002 年第一次提出以 XQuery 進行 XML 關聯規則的探勘的概念 (以下簡稱 W&D) [6][7] ，他們以一段 XQuery 的陳述式便可由 XML 文件中產生 C_1，如**圖 4**所示，再將 C_1 以 Apriori 演算法產生高頻項目集，最後再依據信賴度篩選高頻項目集以產生關聯規則。

```
let $src := document("/transactions.xml")/items
let $minsup := 0.4
let $total := count($src) * 1.00
let $C := distinct-values($src/*)
let $1 :=(for $itemset in $C
          let $items := (for $item in $src/*
                         where $itemset = $item
                         return $item)
          let $sup := (count($items) * 1.00) div $total
                      where $sup >= $minsup
                      return <largeItemset>
                                <items> {$itemset} </items>
                                <support> {$sup} </support>
                                </largeItemset>)
let $L := $1
return <largeItemsets> {apriori($1, $L,$minsup, $total, $src)} </largeItemsets>
```

圖4 使用 XQuery 找尋 C_1

3. IXARM 演算法

3.1 基本概念

以 XML 文件所儲存的資料不像關聯式資料庫系統會將所有的交易資料集中在一個資料庫裡，而會因為應用目的的不同而儲存在不同的文件或檔案中。因此，如果要對 XML 檔案進行資料探勘，以往的做法便得先將所有檔案彙整成一個單一文件，或將 XML 資料轉入關聯式資料庫中再進行運算，如果當文件發生異動的情況時，便必須再重覆彙整所有文件的步驟再重新運算。

基於上述的理由，本論文將提出一個適用於 XML 文件的探勘方法，可以分別探勘單獨的 XML 文件，得到個別文件的項目集資料，再將所得到的項目集資料加以整合以得到所有文件的高頻項目集，以利進一步分析出關聯規則。本方法除了可以整合數個 XML 文件的關聯規則，亦可以延伸應用到部份文件有新增、刪除與修改等異動時，僅需要針對異動部份的文件進行項目集的重新計算，便可以修正整體的高頻項目集. 此方法將更適用於 XML 文件的關聯規則探勘並提高資料探勘的整體表現。

本論文所提出的項目集漸進式探勘演算法採用保留原 XML 文件項目集資訊的概念，因此本論文方法要求在前次探勘 XML 文件後必須儲存以下資訊：

1. 前次探勘的所有項目集、支持度及交易總數。
2. 新增檔案本身的所有項目集、支持度及交易總數。
3. 檔案異動狀態資料。

保留第一項資訊的目的，是為了在 XML 文件

異動後，不需掃描原 XML 所有文件便能快速取得前次探勘結果哪些項目集，每次 XML 文件異動後，僅需針對標示為異動的文件計算，再將結果與前次探勘的結果整合，便可以求得新的項目集資訊。

異動資料可分為新增、刪除與修改。在新增時，探勘新增的文件的項目集、支持度及交易總數，讀取前次探勘的項目支持度及交易總數，再將兩個資料進行整合，並取消檔案新增狀態。刪除時，無需再進行探勘，直接讀取欲刪除文件與前次探勘的結果進行整合，再將欲刪除的文件逕行刪除即可。至於修改，只需依序執行刪除與新增兩個步驟即可，要注意的是執行刪除步驟的最後，並非刪除檔案，而是將檔案狀態改為新增狀態。

3.2 關聯規則整合方法

3.2.1 整合多個 XML 關聯規則之支持度的計算

假設所要探勘的資料共有 n 份 XML 文件，已經分別對每份 XML 文件進行探勘出關聯規則，令關聯規則 r_1 其所含項目集為 x_1 與 y_1， s_1 表示其支持度， I 表示交易資料，則依照公式(2-1) 我們可以得到以下的式子：

$$s_1 = s(x_1 \cap y_1)$$

$$s_1 = \frac{|x_1 \cap y_1|}{|D|}$$

即

$$s_1 = \frac{count(r_1)}{count(I)}$$

將項目集的依照文件分解再依照分子拆解分數式可以得到以下式子：

$$s_1 = \frac{count(r_{11})}{count(I_1)+count(I_2)+\cdots+count(I_n)} + \frac{count(r_{21})}{count(I_1)+count(I_2)+\cdots+count(I_n)} + \cdots + \frac{count(r_{n1})}{count(I_1)+count(I_2)+\cdots+count(I_n)}$$

分數上下同乘以其文件的交易總數並將分子移項：

$$s_1 = \frac{count(r_{11})}{count(I_1)} \times \frac{count(I_1)}{count(I_1)+count(I_2)+\cdots+count(I_n)} + \frac{count(r_{21})}{count(I_2)} \times \frac{count(I_2)}{count(I_1)+count(I_2)+\cdots+count(I_n)} + \cdots + \frac{count(r_{n1})}{count(I_n)} \times \frac{count(I_n)}{count(I_1)+count(I_2)+\cdots+count(I_n)}$$

在依據公式(2-1) 將各文件 r_1 的 s_1 帶入，可得以下式子：

$$s_1 = s_{11} \times \frac{count(I_1)}{count(I_1)+count(I_2)+\cdots+count(I_n)} + s_{21} \times \frac{count(I_2)}{count(I_1)+count(I_2)+\cdots+count(I_n)} + \cdots + s_{n1} \times \frac{count(I_n)}{count(I_1)+count(I_2)+\cdots+count(I_n)}$$

彙整上列式子，我們可以得到：

$$s_1 = \sum_{k=1}^{n} s_{k1} \times \frac{count(I_k)}{count(I)} \qquad (3\text{-}1)$$

3.2.2 漸增 XML Data 的關聯規則之支持度的計算

當新增第 $(n+1)$ 檔案後，我們令 $\sum_{k=1}^{n} s_{k1} \times \frac{count(I_k)}{count(I)}$ 為 s_{1old}，則依據公式 (3-1) 可得：

$$s_{1new} = \frac{s_{1old} \times count(I_{old}) + s_{n+1} \times count(I_{n+1})}{count(I_{old}) + count(I_{n+1})} \qquad (3\text{-}2)$$

3.2.3 刪除 XML Data 的關聯規則之支持度的計算

當刪除第 m, $1 \le m \le n$ 檔案後，我們令 $\sum_{k=1}^{n} s_{k1} \times \frac{count(I_k)}{count(I)}$ 為 s_{1old}，則依據公式 (3-1) 可得：

$$s_{1new} = \frac{s_{1old} \times count(I_{old}) - s_{n+1} \times count(I_m)}{count(I_{old}) - count(I_m)} \qquad (3\text{-}3)$$

3.3 IXARM 演算法

整合前幾節所述項目集與支持度的調整方法，本論文提出的 IXARM 演算法主要步驟如下：

Algorithm: IXARM Algorithm
Input: a XML documentset,
the last Association Rule oldAR,
minimum support minsup
Ouput: New Association Rule newAR

```
for each Deleted and Updated XML documents{
   get these documents' ItemSets;
   get the older;
   compute newAR with ItemSets, oldAR and minsup;
}
Delete all XML document which Status is Deleted;
Update all Updated XML documents set Status =
Inserted;
for each Inserted XML document{
   compute ItemSets from these documents with
minsup;
   get the oldAR;
   compute newAR with ItemSets, oldAR and minsup;
}
Update all Inserted XML documents set Status = null;
```

3.4 以 SQL Server 2005 實做 IXARM 演算法

我們使用 Microsoft® SQL Server® 2005 資料庫系統作為本研究實作的環境，該資料庫系統為關聯式架構的資料庫系統，在 2005 這個版本新增 XML 資料型態並支援 XQuery 語言對 XML 資料直接進行存取，

首先建立一個資料表 T 來儲存 XML 交易文件與相關資訊，包含 DocID、DocXML、ItemSet、AR 以及 Status 等欄位。其中 DocID 為文件編號，DocXML 儲存 XML 交易資料，ItemSet 儲存該文件的所有項目集資料，AR 儲存截這份文件為止的關聯規則，最後 Status 記載文件狀態。

接著我們設計三個主要的預存程序 (Stored Procedures)，第一個是用來產生 XML 高頻項目集的 sp_BuildItemSet、第二個是依據公式 (3-1) 與 (3-2) 以新增 XML 文件的 sp_InsertXMLDoc，最後是依據公式 (3-3) 以刪除 XML 文件的 sp_DeleteXMLDoc，透過這三個 Stored Procedures 的交互運用，便可以達到在資料表 T 新增、刪除與修改時都能得到新且正確的關聯規則。以修改 XML 文件的情況為例，其動作流程如**圖 5**所示。

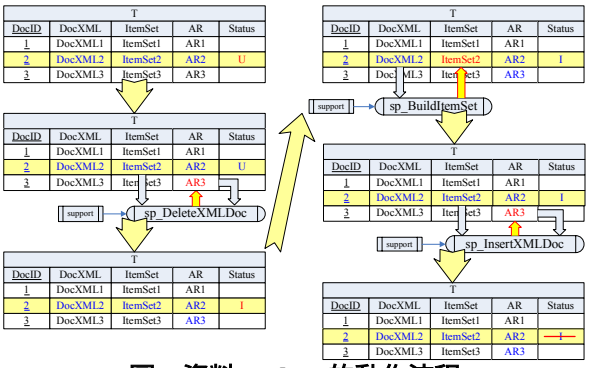

圖5 資料 update 的動作流程

在更新 DocXML 資料後，第一步先將已更新資料的 Status 欄位為設為 U，第二個步驟是呼叫 sp_DeleteXMLDoc 將所有已更新資料的 ItemSet 資料由最後一筆 AR 中移除，將新產生的 AR 存入最後一筆 Status 非標示為 D 的 AR 欄位中，接下來將所有 Status 標示為 U 的資料更正為 I。最後一個步驟是呼叫 sp_InsertXMLDoc，sp_InsertXMLDoc 會呼叫 sp_BuildItemSet 重新產生所有 Status 標示為 I 的 ItemSet 資料，再將所有 Status 標示為 I 的 ItemSet 與最後一次的 AR 整合並修正該 AR 欄位資料，最後再將所有標示為 I 的 Status 設為空值。

在 XQuery 的撰寫方面，我們參考 W&D 的方法[6]，在這篇文章中他們提出如**圖 4**所示的 XQuery 陳述式來產生 Apriori 演算法的 C_1，但他們所使用的原生型 XML 資料庫系統 X-Hive/DB 4.1[11]，該資料庫系統可以支援完整的 XQuery 語法，而我們為了能在關聯式資料庫系統中實現探勘 XML 資料的可能，而選擇的 SQL Server 2005 並不支援 FLWER 中「LET」語法，因此我們除了 XQuery 以外必須要輔以 T-SQL 以及 SQL Server 2005 針對 XQuery 所提供的方法與函數來修改 Wan 以及 Dobbie 所提出的方法，如**圖 6**所示。

```
declare @l xml
select @l=DocXml from T where DocID = @DocID
set @l = @l.query('
     <ItemSet docid="" total="{count(//transaction)}">
     {for $itemset in distinct-values(//item)
     order by $itemset
     return
          <items total="{count(//transaction)}">
          <item>{$itemset}</item>
          {for $item in data(//item)
          where $item=$itemset
          return <temp>{$item}</temp>}
               <support>
               {count(//transaction)}</support>
          </items>
     }</ItemSet>
')
set @l = @l.query('
<ItemSet total="{/ItemSet/@total}">
{for $items in //items
return
     <items itemsid="{data($items/item)}"
               support="{count($items/temp) div number($items/
               @total)}">
     <item>{data($items/item)}</item>
     </items>
}</ItemSet>
')
```

圖6 使用 SQL Server 2005 XQuery 找尋 C_1

4. 驗證與評估

4.1 效能評估

實驗評估將分成兩個部份，第一個部份是比較 IXARM 演算法和非漸進式探勘演算法，第二個部份則評估 IXARM 演算法的執行效率受不同資料集參數之影響。實驗採用的資料集 (dataset) 是以 IBM Almaden 研究中心所提供的交易資料模擬程式所產生，該程式下載位置為[8]。因為該程式所產生的資料並非 XML 格式，因此我們自行撰寫程式將其所產生的資料重新編輯為 XML 格式，模擬程式來產生實驗中所需的 XML 資料集。

4.1.1 和非漸進式探勘演算之比較

此部份之實驗，我們將比較漸進式 (IXARM) 與非漸進式(W&D) 探勘演算法，的在探勘資料效能上的差異。

【實驗 1】本實驗考慮每次只有新增一份 XML 文件的情形下，將實驗分成兩個部份分別採用兩種新增方式，【實驗 1-1】每份文件儲存 400 筆交易資料，【實驗 1-2】資料總筆數固定為 4000 筆，按比例分配新舊文件中的資料筆數，觀察新增 XML Data 交易筆數時，IXARM 與 W&D 的執行情況。

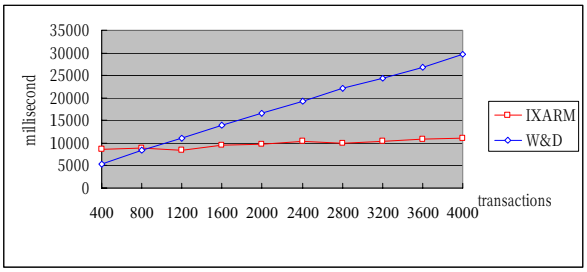

圖7 實驗 1-1 結果

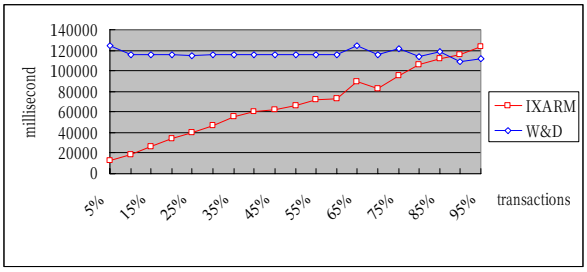

圖8 實驗 1-2 結果

由【實驗 1】的結果顯示觀漸進式探勘演算法(IXARM)，在執行時只需計算新增的 XML 文件與讀取前次的探勘結果，其執行效率的表現近似 $O(1)$。

【實驗 2】本實驗交易資料以文件為單位，每個文件中存有 400 筆交易資料，分成三個部份，觀察異動 XML 文件(新增、刪除或同時有新增刪除)數持續增加時，IXARM 與 W&D 的執行情況。

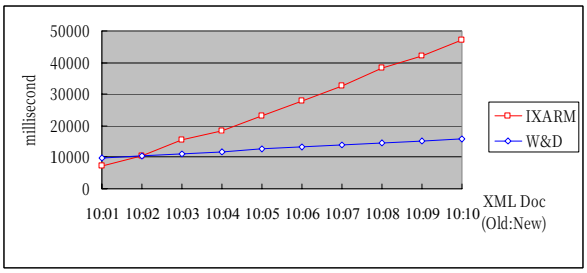

圖9 實驗 2-1 結果

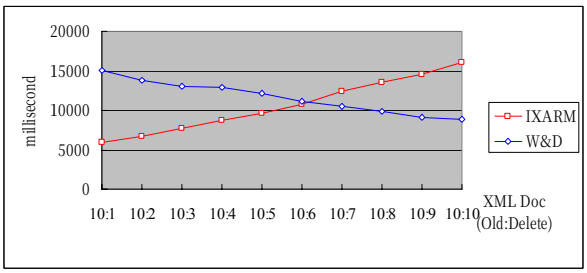

圖10 實驗 2-2 結果

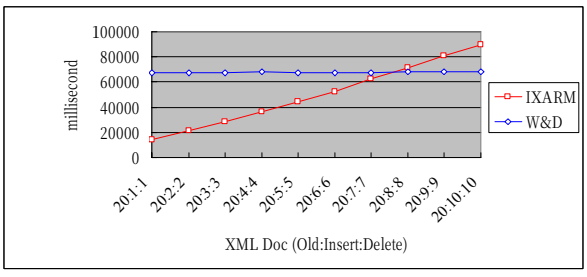

圖11 實驗 2-3 結果

由於 IXARM 演算法在處理新增文件時，需要分別計算新增文件的高頻項目集，由【實驗 2-1】顯示執行效率會隨新增文件量而變差。但在處理刪除刪除文件狀況時，並不需要針對該文件重新計算，只需讀取該文件儲存於資料庫內的前次探勘資料，所以由【實驗 2-2】可見其在處理刪除文件時執行效率較優。在同時有新增刪除資料時，由【實驗 2-3】更可得證其仍舊有較佳的執行效率。

4.1.2 實驗參數對 IXARM 演算法之效率評估

此部份的實驗皆考慮 XML 文件數量與交易筆數相同的情形下，觀察產生資料集時不同參數值的設定，對 IXARM 演算法執行時間的影響。

【實驗 3】本實驗將「項目種類數」以及「支持度」定為控制變數。實驗結果如**圖 12**以及**圖 13**所示。

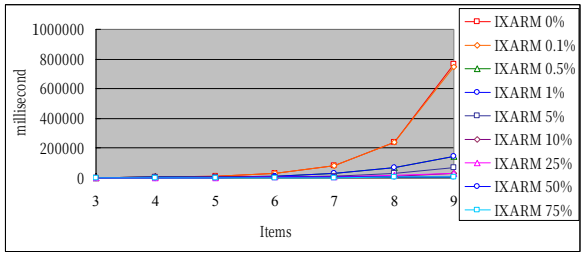

圖12 實驗 3 結果之一

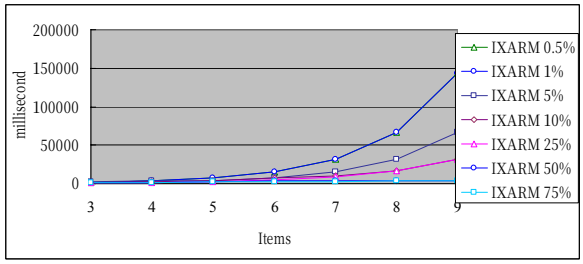

圖13 實驗 3 結果之二

由於 IXARM 在產生項目集時，採用的窮舉法雖然簡單，但其時間複雜度為 $O(2^n - 1)$，當 N 值越大時 IXARM 演算法執行效率便越差。此情況是因為當 N 值愈大時，表示 XML Data 中的項目種類數愈多，即使交易的總數與每筆交易中的平均項目數不高，項目集的產生便已花費相當多的時間。

【實驗 4】分別以 W&D 與 IXARM 演算法重新探勘【實驗 3】的資料，觀察兩種演算法所求得的項目集數量，實驗結果如**表 2**所示。

表2 實驗 4

Support	Items / Algorithm	3	4	5	6	7	8	9
0.000		7	15	31	63	127	255	511
0.001	W&D	7	15	31	63	127	255	511
	IXARM	7	15	31	63	127	255	511
0.005	W&D	7	15	31	63	127	255	511
	IXARM	7	11	23	40	65	98	127
0.010	W&D	7	15	31	63	127	255	511
	IXARM	7	11	23	40	65	98	127
0.050	W&D	7	15	15	31	63	127	255
	IXARM	7	7	14	21	35	53	61
0.100	W&D	7	7	15	31	31	63	127

0.250	IXARM	5	7	9	16	25	32	34
	W&D	3	3	7	15	31	63	127
0.500	IXARM	3	3	7	12	10	13	14
	W&D	1	3	7	7	7	7	3
0.750	IXARM	1	3	5	5	3	1	3
	W&D	1	3	1	1	1	1	0
	IXARM	1	1	1	1	1	1	0

比較【實驗 4】兩個演算法的結果，發現 IXARM 會發生項目集遺失的情況，在進一步去比較 W&D 的結果，發現這些遺失的項目集，其支持度都十分接近設定的支持度。

4.2 實驗結果總結分析

在一般作業環境中，異動資料會遠小於歷史資料，經過以上的實驗我們發現 IXARM 會有較佳的執行效率。但在資料更新的比例太大的極端情況下，W&D 的效率反而優於 IXARM，因此我們可以修正 IXARM 的流程，當在進行資料探勘時，先分析資料異動的狀況，再進行評估採用何種演算法。參考【實驗 2】的結果，假若新增資料超過所有 XML 文件數的 20%或刪除文件數量佔所有文件總數的 60%以上時，則採取 W&D 的方法進行資料探勘，反之，則採用 IXARM 演算法。

由【實驗 3】的結果可以得知，支持度的設定對於提升演算法的效能很有幫助，但由【實驗 4】的結果發現支持度的設定會造成部份關聯規則的遺失，原因在於產生各別文件的高頻項目集時，很可能會過濾掉一些將會成為關聯規則的項目集，可能造成的影響中，較輕微的是這些項目集在 AR'的支持度會略小於 AR，較嚴重的情況是這些項目集彙總後的支持度尚無法滿足預設值，導致無法呈現在 AR'中。

依據中央極限定理 (Central Limit Theorem) 以及統計學的經驗法則，當常態分佈時，以平均數為基準會有 99%的數據落在而三倍標準差範圍內。因此，我們可以在支持度與 0 之間再取一個值做為「準支持度」(quasi-support)，當 sp-BuildItemSet 產生 ItemSet'時會產生通過準支持度的項目集，我們以下稱這些項目集為「準項目集」 (quasi-Itemset)，因為依照常態分配與統計經驗法則的概念，關聯規則的支持度在各別文件中的值有 99%的機率會落在以平均值為基準的三個標差之內，再由準項目集中找出關聯規則，此方法可以同時改善演算法的執行效率與關聯規則遺失的情形。但值得注意的，由於支持度在部份文件中可能有極端值 (outlier) 的出現，因此仍舊有可能發生遺失關聯規則的情況，這在目前的實作上是無法避免的，將來會尋求改善的方法。

5. 結論

本研究本論文針對 XML 資料庫提出一個關聯規則漸進式探勘方法，以論文[6]所提出的 XQuery 探勘 XML 關聯規則方法為基礎，加入漸進式探勘的概念予以改良，並以 XML Enabled 資料庫為作業環境，儲存 XML 交易文件、歷次探勘結果和探勘相關的資訊於資料庫中。當資料庫發生增加或刪除交易資料等異動時，只需針對掃描異動的部份進行計算或讀取的歷史資料，隨之調整項目集結構，使它能符合更新後資料庫中的交易資料內容。因此便能從調整後的項目集中探勘出關聯規則，不需重新掃描整個更新後資料庫。我們提出了調整 Apriori 演算法規則及步驟，組合設計出 IXARM 演算法，可適用於資料庫新增交易資料或刪除交易資料，以及同時有新增及刪除時之關聯規則漸進式探勘，並實驗結果驗證其可行性與優越性。

參考文獻

[1] R. Agrawal, and R. Srikant, "Fast Algorithms for Mining Association Rules," *Procedings of the 20th VERY LARGE DATA BASE (VLDB) Conference Santiago*, 1994.

[2] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules Between Sets of Items in Large Databases," *In Procedings of the ACM SIGMOD Conference on Management of Data*, pp. 207-216, *1993*.

[3] Don Chamberlin, Jonathan Robie, and Daniela Florescu, "Quilt: An XML Query Language for Heterogeneous Data Sources," *Proceedings of WebDB 2000 Conference, in Lecture Notes in Computer Science*, Springer-Verlag, 2000.

[4] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," *in Procedings of the ACM SIGMOD Int. Conferences on Management of Data*, pp. 1-12, Dallas, Texas, USA, 2000.

[5] Suh-Ying Wur and Yungho Leu,"An effective Boolean Algorithm for Mining Association Rules in Large Databases," *Database Systems for Advanced Applications (DASFAA 99)*, 1998

[6] Jacky W.W. Wan, and Gillian Dobbie, "Mining Association Rules XML Data using XQuery," *The Australasian Workshop on Data Mining and Web Intelligence* (DMWI 2004) , 2004.

[7] Jacky W.W. Wan, and Gillian Dobbie, "Extracting association rules from XML documents using XQuery," *In Proceedings of Fifth International Workshop on Web Information and Data Management*, New Orleans, LA, USA, 2002.

[8] IBM Almaden Research Center, Quest Synthetic Data Generation Code, http://www.almaden.ibm.com/software/projects/iis/hdb/Projects/data_mining/datasets/syndata.html#instructions.

[9] KDnuggets, Polls : Data Mining Methods, http://www.kdnuggets.com/polls/2006/data_mining_methods.htm, Apr 2006.

[10] World Wide Web Consortium, XML, http://www.w3.org/XML .

[11] X-Hive/DB. http://www.x-hive.com.