國立中山大學電機工程學系

碩士論文

一種發掘週期性關聯規則之演算法

An algorithm for discovering periodical association rules

研究生：江忠益 撰

指導教授：李錫智 博士

中華民國九十三年七月

摘要

本論文主要內容有兩個部分，第一部分我們設計一個新的、更有效率的演算法來採掘資料庫中具有多層次關係時間週期性質的 calendar-based association rules。不同於一般使用 apriori-like 的方法，我們的方法利用最多掃瞄資料庫兩次的架構，可避免多次掃瞄資料庫，以節省大量的資料庫掃瞄時間，並且利用時間週期的特性來減少掃瞄資料庫過程中所要搜尋的 candidate calendar patterns 數目以增進程式執行的速度。以上這兩個特點使我們的方法能夠更有效率的採掘出整個資料庫中所有具有時間週期的 association rules。

在第一部分中，我們所考慮的是具有嚴格限制，必須在週期上固定時間點循環出現的 calendar-based association rules，沒有考慮到異位（非同步）的情形，但是在真實世界所收集到的資料中，很可能具有非同步週期出現的規則存在，而這些規則可能是有用的。因此，我們在論文的第二部份透過 membership function 來定義所要找尋的非同步週期 fuzzy calendar pattern，並進一步搜尋出資料庫中符合 fuzzy calendar pattern 的 association rules，以得到 fuzzy periodical association rules。

由實驗結果得知，我們的方法能更有效率的從資料庫中挖掘出 calendar-based 及具有非同步週期的 fuzzy periodical association rules。

關鍵字：temporal association rules, calendar-based association rules, fuzzy periodical association rules.

Abstract

There are two main contributions in the thesis . Firstly, we design a novel and efficient algorithm for mining calendar-based association rules which have multilevel time granularities in temporal databases. Unlike apriori-like approaches , our method scans the database twice at most. By avoiding multiple scans over the database , our method can reduce the database scanning time.

Secondly, we use membership functions to construct fuzzy calendar patterns which represent asynchronous periods. With the use of fuzzy calendar patterns, we can discover fuzzy periodical association rules which are association rules occurring in asynchronous periods.

Experimental results have shown that our method is more efficient than others, and we can find fuzzy periodical association rules satisfactorily.

Keywords: temporal association rules, calendar-based association rules, fuzzy periodical association rules

目錄

圖表目錄

1.圖

2.流程圖

3.表

第一章、論文內容簡介

本論文概分為三個部分，在第二章中我們先簡單介紹一般採掘 temporal association rules 的幾個方向，再開始進入採掘 calendar-based association rules 的工作。首先從 association rules 的定義開始介紹，再循序漸進的介紹 calendar-based patterns 與一般 calendar-based periodical association rules 的定義。透過以上的定義可以清楚瞭解所要研究的對象，也方便之後採掘 calendar-based periodical association rules 演算法的描述。在第二章敘述演算法時，先簡單介紹之前別人所提出的方法，概要的敘述其主要架構，然後再描述我們所設計的演算法。

在第二章中，所採掘的對象是之前別人所提出的 calendar-based periodical association rules，本論文設計一個新的演算法來增進其執行效率。而在第三章中，本論文提出一個新的採掘對象---fuzzy periodical association rules。Fuzzy periodical association rules 與 calendar-based periodical association rules 的不同之處在於 fuzzy periodical association rules 利用 membership function 來建構 calendar patterns，我們稱之為 fuzzy calendar patterns，以此來描述所要找尋的非同步週期，使得所要找尋的週期可以具有非同步（異位）的性質，可以容許 association rules 所出現的週期有些許的位移，即不一定非得準確的出現在每個週期的固定時間點上，而容許稍微提前或延後出現。如此一來，將可以更有彈性的依照使用者需求針對所想要知道的週期，來設計此週期的非同步性質，再根據這樣的週期特性來找出具有此種非同步週期的 association rules。

第四章利用數個實驗來支持我們的方法。其中，前三個實驗以第二章中所描述的兩個演算法來做比較，證明我們所設計的演算法可以增進採掘 calendar-based periodical association rules 時的執行效率。在第四個實驗中，以經過設計的資料來測試所提出的 fuzzy calendar pattern 是否真的可以成功的採掘具有非同步週期的 association rules。

第二章、 Mining calendar-based periodical association rules

1. Introduction

近年來，隨著資訊爆炸時代的來臨，資訊已經越來越方便取得，其數量也日趨龐大，人們不再滿足於隨處可見，原始而沒有經過分析整理的資訊，開始渴求經過分析、過濾、擷取與整理之後的知識。因此，如何在資料庫中找尋未知及令人感興趣的知識，受到越來越多的注意與研究。其中，在 data mining 的各種領域中，找尋描述資料庫中各種物件之間存在的 association rules 在 1993 年由 Agrawal 提出[1]，帶動在資料庫中找尋物件之間的關聯性之研究。

Association rules 是存在於一群資料中，物件與物件之間的關聯性規則，例如在禮品店中我們可能找到像〝客人在購買皮夾時常會一起購買皮帶〞這樣的規則。而透過從資料庫之中找出類似的規則，使用者可以憑著這些規則來安排商品的擺設或選擇促銷的組合，使用這些策略來提高獲利。在 association rules mining 的方法中，由於資料庫的資料量越來越大，因此如何以最快的速度發掘出所需要的 association rules，成為做此項研究時一項非常重要的問題。為了增進在資料庫中找尋 association rules 的效率，有很多方法已經被提出及研究[2][3][8][9][10]。

上述的方法中，所發掘的 association rules 雖然可以找到物件之間的關聯性，但是其存在的價值是以整個資料庫而言，這一類的方法所找出的 association rules 所考慮的範圍包含了整個資料庫所存在的時間。但是在現實生活中，我們人類會依照季節的不同，星期的循環，或是特定的節日來安排我們的活動。因此，如果將時間的特性

也考慮進去時，將可以發現更多令人感興趣且有用的 association rules，例如〝在每年的 2 月 13 日（情人節前一天）消費者通常會一起購買花跟巧克力〞這般加入時間描述的規則。透過這樣的發現，市場管理者可以在 2 月 13 日的時候加強花跟巧克力的促銷，或者將兩者做搭配來販售，將可以提高更多的營業額，而且可以在不同的日子根據不同的規則加以改變銷售策略，例如在聖誕節前改促銷圍巾跟手套組，如此一來將可以更貼近消費者的需要進而得到更高的利潤。

然而，在一般 association rules mining 的方法中由於沒有考慮到時間屬性，所以沒有辦法直接用來找尋有時間規律的 association rules。因此，為了發掘這些跟時間相關的規則，temporal association rules mining 開始被討論及研究。

Temporal association rules mining 考慮到資料時間屬性的特點，並且利用這些時間特性來發展適合的演算法。在這些研究中，包括找出 association rules 存在於資料庫中的時段 [12][14]，例如圍巾與外套從 10 月到 1 月之間常常一起被買的規則；找出 items 出現的先後順序的規則[4][13]，例如人們總是先看完哈力波特第一集接著才看哈力波特第二集然後第三集類似這樣的順序關係；以及我們所要討論的，找出在資料庫的時間單位中具有循環出現特性的 periodical association rules[5][6][7][11]等等。

在 temporal association rules mining 中，認為在資料庫的物品中隱含著具有時間規律出現的 association rules。而為了找出這些具有時間規律的 association rules，有一些方法已經被提出來。其中有一些方法是在資料庫中找出具有先後順序關係的 sequential patterns[4][13]。[4]中提出，在資料庫中存有顧客所購買的商品並以先後購買的次序排列而成 sequences，其中不同的顧客有各自的一

條 sequence。Sequential patterns 則是代表在這資料庫的 items 中，有哪些 items 之間具有先後出現的次序關係。另外還有一些方法是在找尋具有週期性的 association rules[5][6][7][11]，這些方法是以固定的時間單位來分割資料庫中的資料，並且在這些時間單位中找尋具有固定時間間隔循環出現的 association rules。

在[5]中，提出找尋在資料庫中具有 cyclic association rules 的方法，這個方法是在 association rule sequences 資料庫中（該 sequence 長度為所有的時間單位數量且 sequence 中的每個成員是以 binary 型態表示此 association rule 是否出現在該時間點內。）找出在週期長度 l 的第 o 個時間單位循環出現的 association rules，例如會在每 10 天中的第五天出現，並且嚴格規定 association rules 在每個週期的第 o 個時間單位都必須存在。這樣的方式使用者必須對資料庫中的資料有一定的瞭解才能知道要找尋什麼樣的週期，否則必須不斷嘗試，且不一定可以得到合適的結果。而考慮到現實世界中有些週期並不是如此完整的循環出現，因此，為了找到更多有用的 rules，在[11]中提出一個 association rule 只要在時間序列的週期裡出現夠多的次數就能成立的方法，可以找到更多有用的 periodical association rules。

在現實世界裡我們所遵守的時間規則是有多種的、層級式的形式。例如年、月、日、星期…等等。並且有些時間單位並沒有固定的週期間隔，像是月有分大小月，那麼以找尋在特定的時間間隔循環的方法，就沒有辦法找出像是〝每個月的第一天〞這樣的週期，但這樣的週期確實存在我們的生活當中並且可能含有重要且有利用價值的 association rules 在裡面。

在[6]中，讓使用者可以透過 calendar algebra 定義他所想要找尋的週期形式，像是〝每個月的第一個工作天〞這樣的週期，然後找出在該週期出現的 association rules。雖然[6]提供給使用者方便的工具來定義所要找尋的週期，但是使用者自己必須清楚的知道什麼週期才是他想要找的，才能有效的利用這個方法。但是有時候面對一個資料庫，使用者對裡面的資料並沒有明確的概念或不清楚該找什麼樣的目標週期。

因此，[7]中提出，讓使用者自己定義 calendar schema 則此 calendar schema 內所描述的所有週期 calendar-based patterns(或簡稱 calendar patterns)，就可以被找出來。一個 calendar schema 是一種層級式的日期觀念，如年、月、星期、日…等等。例如，使用者可以定義一組 calendar schema R = (year : {1995,1996,…,1999},month:{1,2,…,12},day:{1,2,…,31}) 。

在[7]中提出的 calendar-based temporal association rules mining 方法〝 Temporal-apriori 〞，他把具有時間屬性的資料庫依照 calendar schema 所定義的最小時間單位 basic time interval 做分割，並參考 Apriori[2]的架構來進行 calendar-based temporal association rules mining，他同樣具有必須多次掃瞄資料庫的特性，而且在掃瞄過程中，必須一層一層的在每一個 basic time interval 中，根據前一次掃瞄資料庫得到的結果來產生 itemsets 的 candidates 來計算它們出現在這個 basic time interval 的次數是否滿足成為 large itemsets 的標準，還要計算 large itemsets 出現在涵蓋到該 basic time interval 的 calendar patterns 的次數，因而造成時間上的損失。

我們認為這種使用 calendar schema 的方式比起在固定 time interval 的 sequence 裡找尋的時間週期模式更具完整性跟實用性，可以找到具有真實世界時間規則的 association rules。因此，本論文以此週期模式來做採掘 temporal association rules 方法的研究。我們的方法利用把掃瞄資料庫的步驟限制在最多兩次，來減少存取資料庫的時間。另外，在第一次掃瞄資料庫的時候，我們利用 calendar patterns 的特性來減少所要搜尋的 candidate calendar patterns 的數目，並且藉由一次產生全部的 candidate itemsets 以及它們的 candidate calendar patterns 來避免在每個時間區間產生 candidate itemsets，藉此增進整體的效率。

從實驗結果得知，我們的方法的確可以更快速的找出所有 large itemsets 的 calendar patterns。

2. Problem definition

2.1 Association rule

Association rule的觀念首先是在[1]中提出，假設I是一群items的集合，即$I=\{i_1, i_2 \ldots i_N\}$(假設共有N個items)，其中每個item各異(即$i_k \neq i_j$，if $k \neq j$)，使D為一個transaction database，一筆transaction t在資料庫D中表示一組items的子集合，即$t \subseteq I$，如果一組I的子集合$x \subseteq t$，稱為交易t內含有x，並且定義一組itemset的support代表在資料庫D裡有包含此itemset的transactions對於所有的transactions的比例。

一條 association rule 的表示如下：

$$x \rightarrow y \qquad (1)$$

代表兩個相異的 itemset x 跟 y, 在 itemset x 出現的情況下，itemset

y會伴隨著出現的規則，其中x∩y＝ϕ。以O_i代表在資料庫D中包含集合i 的transactions數量，｜D｜代表整個資料庫的transaction數目，則集合i在資料庫出現的support $S_i = O_i / |D|$，我們稱一條association rule x -> y具有support $S_{x \cup y}$

$$S_{x \cup y} = O_{x \cup y} / |D| \qquad (2)$$

表示在資料庫中 x 跟 y 同時出現的機率。而稱 association rule x -> y 具有 confidence $C_{x->y}$

$$C_{x->y} = S_{x \cup y} / S_x \qquad (3)$$

代表在資料庫中，已經出現 x 而伴隨著出現 y 的機率。

如果一組 itemset 的 support 大於使用者所定義的 support threshold，稱之為 frequent itemset 或 large itemset，我們稱一條 association rule x → y 在資料庫 D 中成立表示 support $S_{x \cup y}$ 大於 support threshold，而且 confidence $C_{x->y}$ 大於使用者所定義的 confidence threshold。在圖（II-2.1）中舉例說明上述各項定義。在這個例子中，假設資料庫D共有五筆transactions，且設定support threshold 為 0.6，confidence threshold 為 0.7。則我們說一組itemset在這個資料庫中為large ，表示該itemset至少要在這個資料庫所有的 transactions 中出現 5*0.6＝3 次。且我們說有一組association rule x->y 在這個資料庫中成立，表示除了 x 與 y 一起出現在這個資料庫的 transactions 次數要大於 support threshold 之外，x 與 y 一起出現在 transactions 的次數除以 x 出現在transactions的次數比例至少要0.7才可以。

Minimal support　s= 0.6 , Minimal confidence=0.7, |D|=5 , 5*0.6=3 最少要出現3次

D =

A B C D
A B C E
A C F
A B E
C D F

Large 1-itemset	count≧3
A	4
B	3
C	4

Large 2-itemset	count≧3
AB	3
AC	3

Association rule	$S_{x\cup y} = O_{x\cup y} / \lvert D \rvert$	$C_{x->y} = S_{x\cup y} / S_x$
A → B	3/5 = 60%	3/4 = 75%
A → C	3/5 = 60%	3/4 = 75%
B → A	3/5 = 60%	3/3 = 100%
C → A	3/5 = 60%	3/4 = 75%

圖（II-2.1）舉例說明 association rule 的各項定義

2.2 Calendar-based pattern

Calendar-based pattern 是在[7]中所定義的 association rules 所存在的週期型態。首先必須定義 calendar schema，一組合法的 calendar schema R 可定義為 $R = (f_n : D_n, f_{n-1} : D_{n-1}, ..., f_1 : D_1)$ 其中每個屬性 f_i 是一種時間單位如年、月、日、星期、小時…等等。每個值域 D_i 則是一組有限的正整數集合，其中 f_i 是一種階層式的架構，每一個 f_i 皆可以由 f_{i-1} 中的所有成員合起來組成，且 f_i 的每一個成員之間所包含的範圍不可以重疊。舉例如下：

1. Calendar schema R＝（week:{1, 2, 3, 4}, day:{1, 2, 3, 4, 5, 6, 7}）是一個合法的 calendar schema，其中每個星期皆由星期日到星期

六這 7 天所組成，且每個星期的星期日到星期六皆不會與其他星期重疊。

2. Calendar schema R＝(month:{1,2,3,4},week:{1,2,3,4,5})為一個不合法的 calendar schema，因為有可能 1 月的第五個星期正好也是二月的第一個星期,所以這是一個不合法的calendar schema。

當一組 calendar schema R 被定義為 $R=(f_n:D_n, f_{n-1}:D_{n-1},..., f_1:D_1)$ 之後，[7]定義在此組 calendar schema 上的 calendar-based pattern(或簡稱為 calendar pattern)為一組組合（$d_n,d_{n-1},...,d_1$），其中 $d_i \in D_i$ 或者是萬用字元〝＊〞， 其中 $d_i \in D_i$ 為 calendar pattern 中的固定值，萬用字元〝＊〞代表能出現在〝＊〞所在位置的每一個成員。下面以例一說明上述定義。

例一、對於 calendar schema R=（year:{1996,…,2000}，month:{1,…,12}，day:{1,…,31}）來說，calendar pattern（＊,1,10）代表每一年的一月十號，calendar pattern （1996,＊,10）代表 1996 年每個月的 10 號。

我們使用布林函數 δ_A 來描述 calendar pattern A，在此

$$\delta_A : T_i \rightarrow \{0,1\} \qquad (4)$$

每一個 time interval $T_i \in D_n \times D_{n-1} \times .... \times D_1$，此函數 $\delta_A(T_i)$ 的值代表 time interval T_i 是否包含在 calendar pattern A 裡面，其函數可以表示成

$$\delta_e(T_i)=\begin{cases}1, T_i \in e\\ 0, otherwise\end{cases} \qquad (5)$$

一個最基本的 calendar pattern 為一個 basic time interval e_0=（$d_n,d_{n-1},...,d_1$），其中 $d_i \in D_i$。則

$$\delta_{e_0}(T_i)=\begin{cases}1, T_i = e_0\\ 0, otherwise\end{cases} \qquad (6)$$

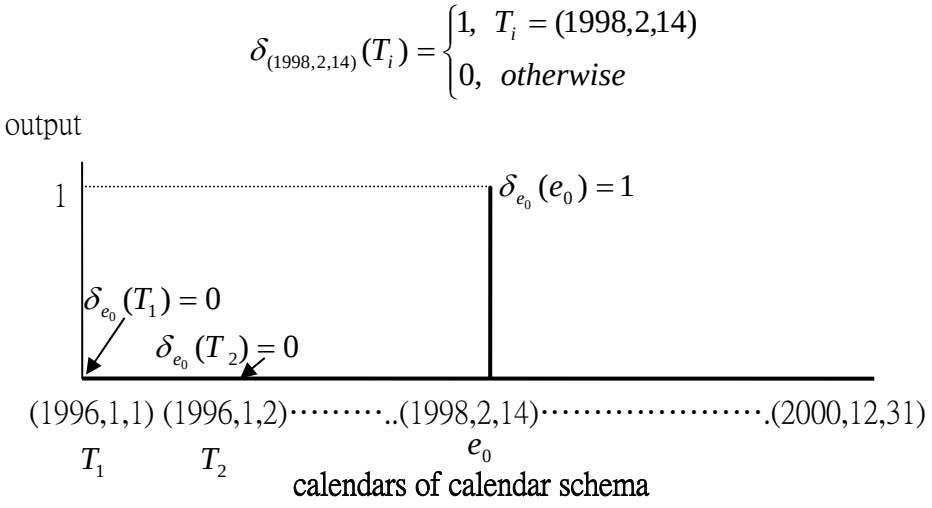
以例一來說，一個 basic time interval e_0=（1998,2,14），則 δ_{e_0} 可以下圖（II-2.2）來表示。亦可以用圖（II-2.3）的各個時間單位的布林函數組合來表示。而一個 calendar pattern e = (*,2,14) 則可以利用圖（II-2.4）來描述。

我們稱一個 calendar pattern e 涵蓋一個 basic time interval e_0，列式表示：

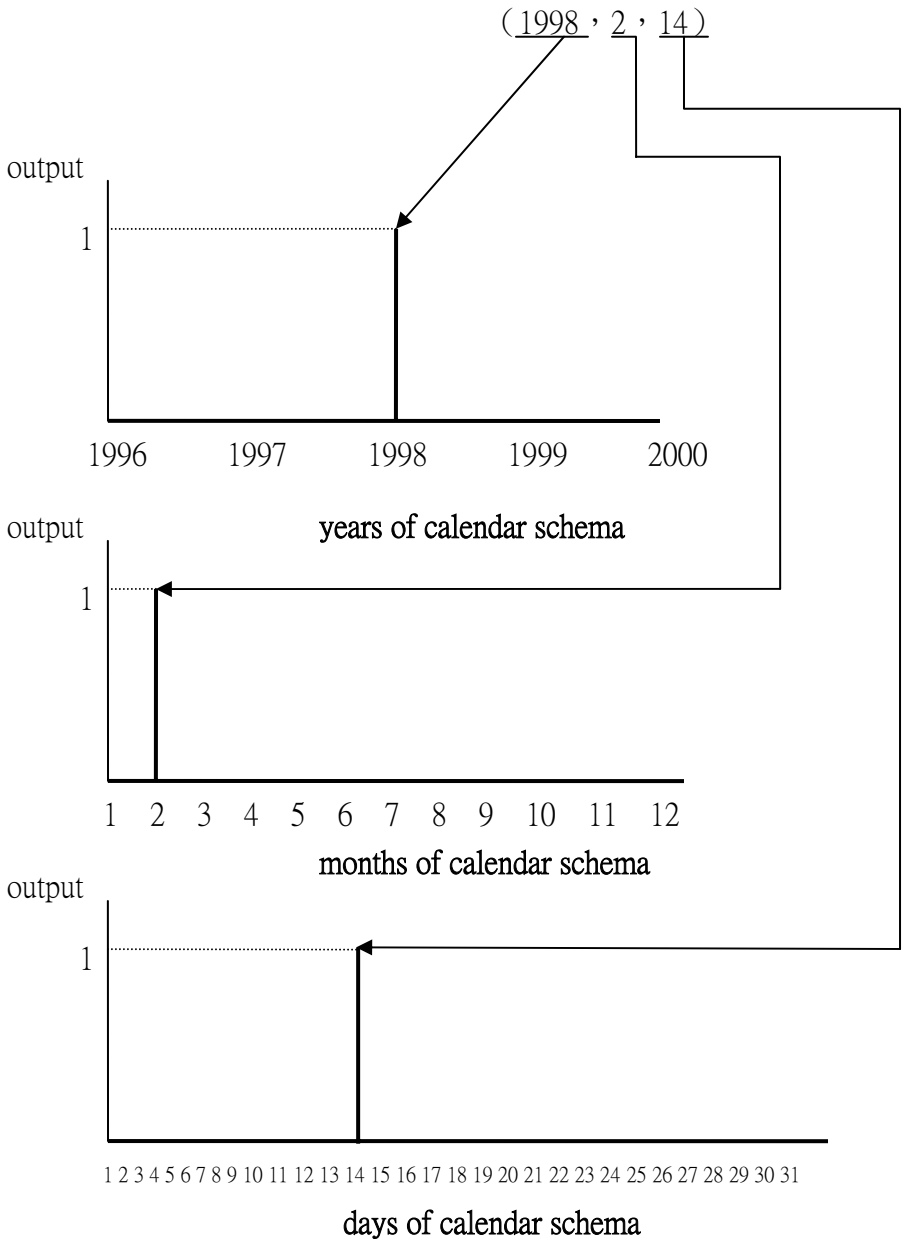
if $\delta_e(e_0)=1$ then e cover e_0

即 e_0 為週期 e 內的一個時間區間。

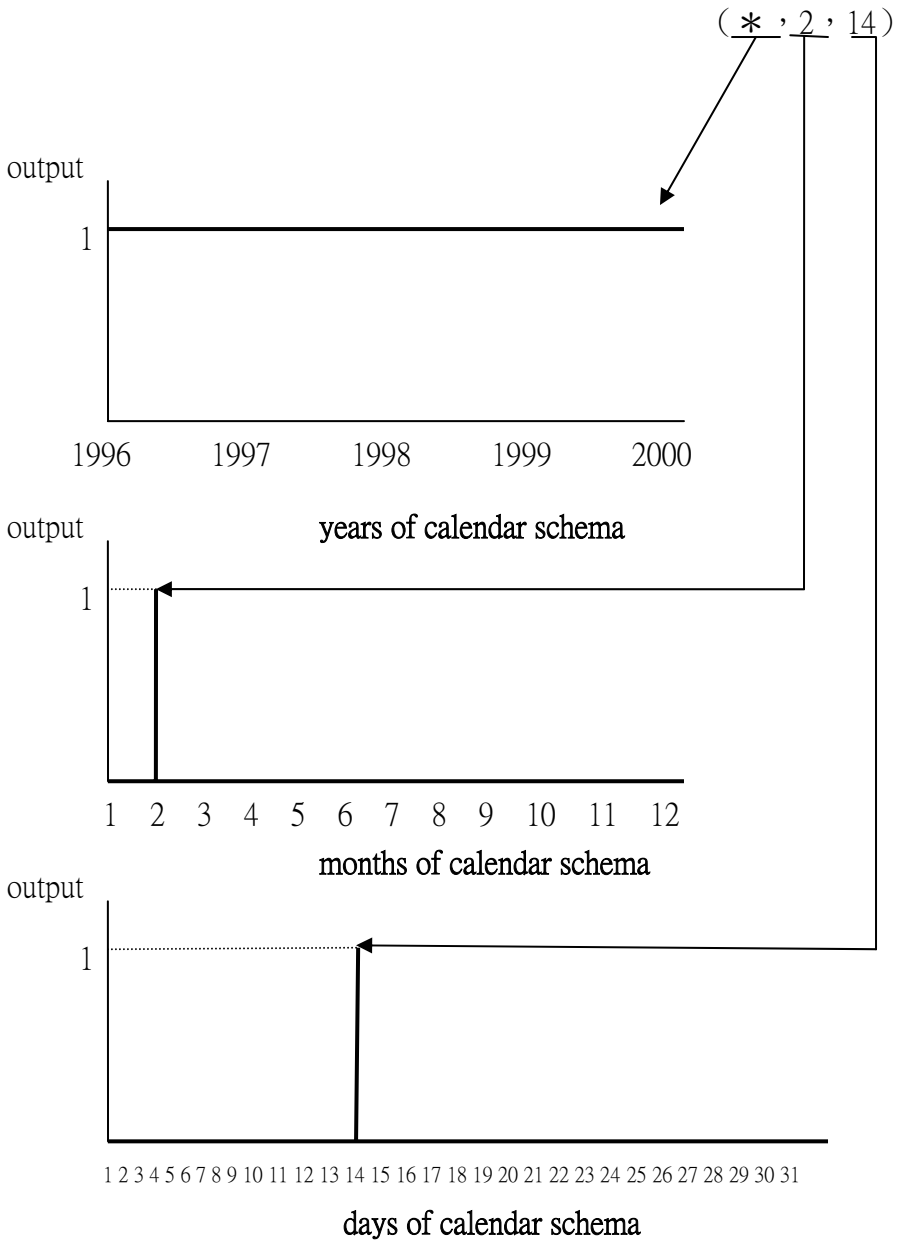
為了表示上的方便，[7]中稱有 k 個萬用字元〝＊〞的 calendar pattern 為 k-star calendar pattern(表示為 e_k)，並且稱沒有包含任何萬用字元〝＊〞的 calendar pattern 為在此 calendar schema 內的 basic time interval。



圖（II-2.2）以布林函數表示（1998，2，14）



圖（II-2.3）以布林函數的組合來表示（1998，2，14）



圖（II-2.4）以布林函數的組合來表示（＊，2，14）

2.3 Calendar-based periodical association rules

假設每筆transaction都有一時間屬性，該屬性紀錄此transaction發生的時間，而整個資料庫的資料則由calendar schema所定義的basic time interval所分割，令一個basic time interval e_0所具有的資料分割P(e_0)，其transaction數量為|P(e_0)|，其中O_i代表在P(e_0)中包含集合i 的transactions數量，則集合i在e_0內的support $S_i = O_i / |P(e_0)|$。我們說一條association rule x->y存在於e_0表示此association rule在此basic time interval的support

$$S_{x\cup y} = O_{x\cup y} / |P(e_0)| \geq \text{support threshold} \quad (7)$$

且confidence

$$C_{x->y} = S_{x\cup y} / S_x \geq \text{confidence threshold} \quad (8)$$

給定一個 calendar schema $R = (f_n : D_n, f_{n-1} : D_{n-1}, ..., f_1 : D_1)$，則依照此 calendar schema 下存在於資料庫的一條 calendar-based periodical association rule 表示為：

$$x \xrightarrow{e} y \quad (9)$$

代表 x -> y 具有 calendar pattern e 這樣的週期，或者說在 calendar pattern e 這樣的週期中，具有 x->y 這樣的 association rule 存在。

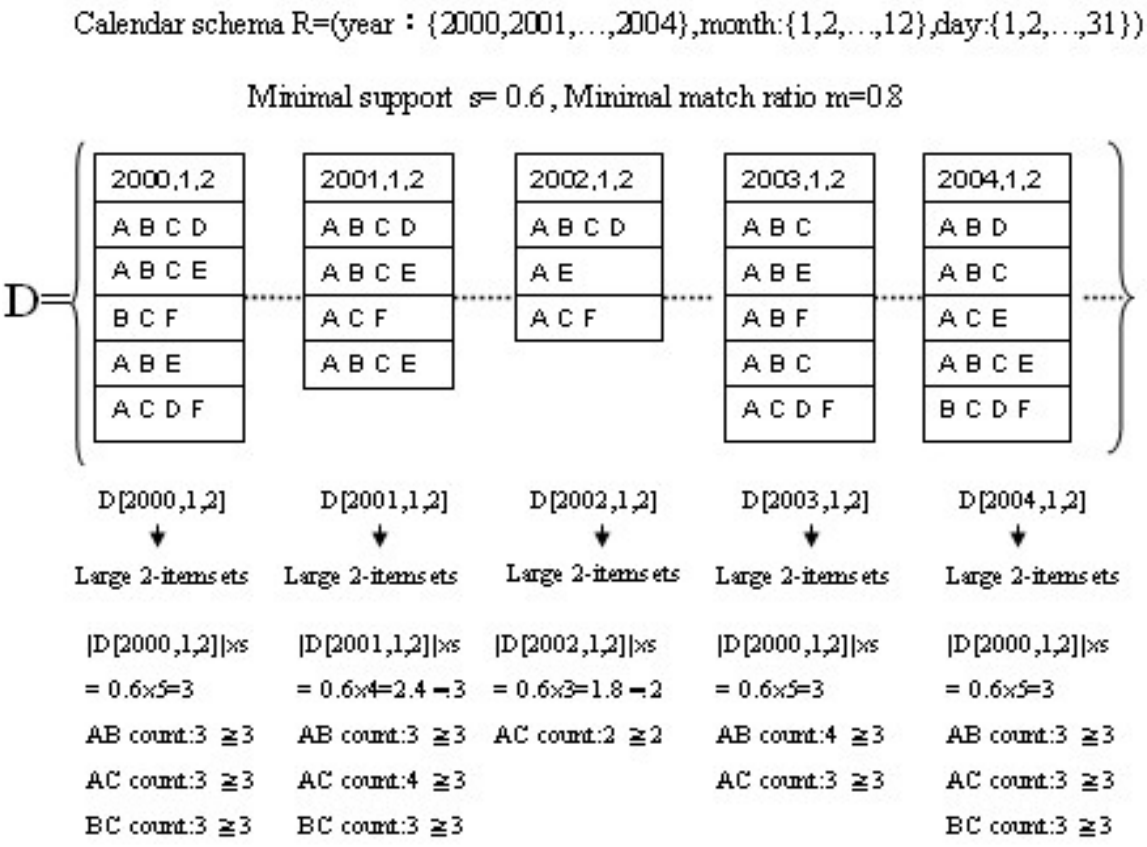
假設 calendar pattern e 所涵蓋的 basic time intervals 數量為|e|，x -> y 在這些 basic time intervals 中出現的次數為$|e_{x->y}|$，則定義 x -> y 具有 e 此週期的 match ratio $m_{x \xrightarrow{e} y}$

$$m_{x \xrightarrow{e} y} = |e_{x->y}| / |e| \quad (10)$$

一條 association rule x -> y 在 calendar pattern e 所涵蓋的 basic time intervals 中出現夠多的次數，即其 match ratio 大於 match ratio threshold 時，則稱$x \xrightarrow{e} y$這條 calendar-based periodical association rule 成立。為了區別一般的 calendar

pattern 與滿足使用者設定的 match ratio threshold 的 calendar pattern，我們稱滿足使用者設定的 match ratio threshold 的 calendar pattern 為 frequent calendar pattern。

一般來說，進行 association rules mining 時，在找出 large itemsets 之後，rules 就可以很容易的由這些 large itmesets 產生。同樣的，在進行 calendar-based periodical association rules mining 時，我們必須先找出所有的 large itemsets 以及他們所存在的 frequent calendar patterns，有了這些資訊將可以很容易的由這些資訊產生 calendar-based periodical association rules。在圖（II-2.5）中我們以 2-itemset 為例，說明如何利用上述的定義透過 minimal support 及 minimal match ratio 找出 large 2-itemsets 是否具有 frequent calendar patterns。在圖（II-2.5）中，假設我們要找出具有在每一年的 1 月 2 日會出現的 large 2-itemsets，換句話說，即這些 large 2-itemsets 具有每一年的 1 月 2 日會出現的週期性質。首先，必須先找出在各年的 1 月 2 日中有哪些 2-itemsets 出現的次數滿足 minimal support 成為 large 2-itemsets。然後再由這些資訊去判斷在這些 large 2-itemsets 當中，有哪些具有週期出現的特性。而用來判斷是否具有週期出現的特性標準，即為 minimal match ratio，也就是在全部年裡的 1 月 2 日中此 large 2-itemset 是否出現足夠的次數來決定。以圖（II-2.5）的例子來說全部有 5 年有 1 月 2 日這一天，因此要滿足 minimal match ratio = 0.8，即至少要在此五年中的 1 月 2 日的其中 5*0.8＝4 天中出現。如此一來，〝在每一年的 1 月 2 日出現〞這樣的週期規則才會成立，如圖（II-2.5）中的 AB、AC。



(*,1,2) has 5 days , m=0.8

An itemset has the frequent calendar pattern(*,1,2) if it occurred in at least 5×0.8= 4 days

AB :[2000,1,2],[2001,1,2],[2003,1,2],[2004,1,2]

count:4 days ≧ 4 days , so AB has the frequent calendar pattern(*,1,2)

AC :[2000,1,2],[2001,1,2],[2002,1,2],[2003,1,2],[2004,1,2]

count: 5 days ≧4days , so AC has the frequent calendar pattern(*,1,2)

BC :[2000,1,2],[2001,1,2],[2004,1,2]

count: 3 days ≦4days , so BC doesn't has the frequent calendar pattern(*,1,2)

圖（II-2.5）Calendar-based periodical association rules 的一個例子

3. Algorithms

以下介紹的algorithms將焦點放在找出所有 large itemsets 以及它們所存在的 frequent calendar patterns 上面。由於我們主要的比較對象為[7]中所提出的 temporal apriori 此方法，因此，在 3.1 中將簡單的介紹 temporal apriori 之演算法架構，並在 3.2 中介紹我們所提出的方法。

3.1 Temporal apriori

在[7]中 Li 使用一個 apriori-like 的方法來找尋存在於 frequent calendar patterns 中的 large itemsets，稱之為 temporal apriori。

Temporal apriori 沿用 apriori 的基本架構，採用一層接一層掃瞄資料庫得到 large itemsets，再以得到的 large itmests 產生下一層掃瞄時的 candidates，直到沒有 large itemsets 產生或沒有 candidates 產生為止。在產生 itemset 的 candidates 時，應用到集合的觀念：

〝一個 large itemset 的所有子集合都是 large itemsets〞

得到在產生 candidate itemsets 時的一個重要原則：

〝當一個 itemset 不是 large 時，他的任何 superset 都不應該產生成為一個 candidate itemset〞

上述的原則在各種類的 association rules mining 中被廣泛的用來減少 candidates 的產生，在 apriori 中利用此原則來產生 candidate itemsets 的子程式稱之為 apriori_gen，其敘述如下：

假設上一層的 large itemsets 為 L_{k-1}，其中兩兩 itemset 皆相異，並且這些 itemsets 內的 items 按照由小到大的順序排列而成。

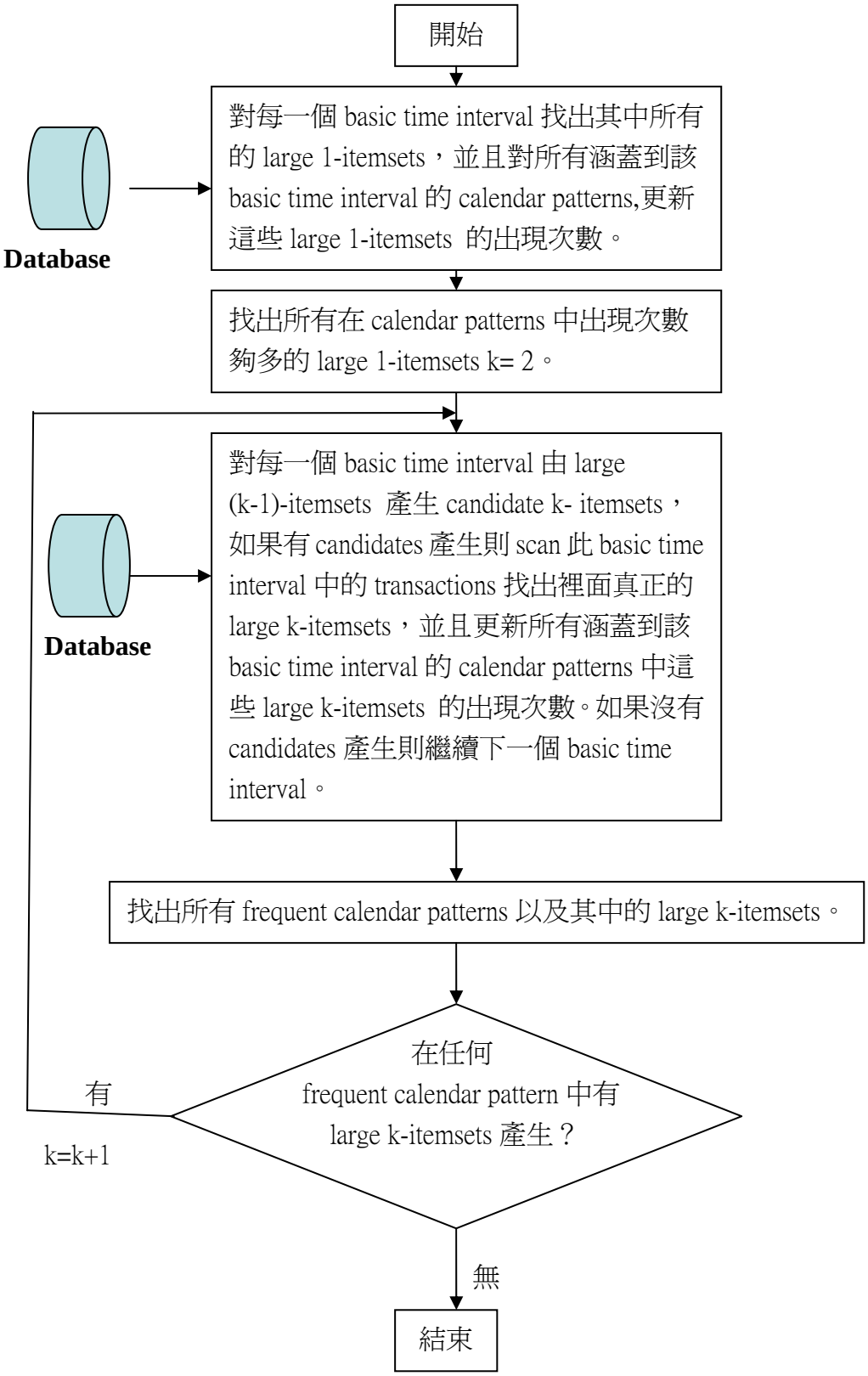
一、對於每一個 $x \in L_{k-1}, y \in L_{k-1}$ ，$x = (x_1, x_2, ..., x_{k-1}), y = (y_1, y_2, ..., y_{k-1})$ 找出其中 $x_1 = y_1, x_2 = y_2, ..., x_{k-2} = y_{k-2}$ 但 $x_{k-1} < y_{k-1}$ 的組合，組成新的 k-itemset $z = (x_1, x_2, ..., x_{k-1}, y_{k-1})$，即 z 為 x 與 y 的聯集。

二、對每一個 $x \in L_{k-1}$ 計算 $x \subset z$ 的個數 m，如果 m=k，則 z 為 large k-itemset 的一個 candidate。

三、將所有符合上面兩步驟的 z 全部找出來則為 large k-itemset 的所有 candidates。

舉例來說，假如 $L_2 = \{AB, AC, AD, BC\}$，則可以由 $L_2 * L_2$ 從 AB、AC 產生 ABC，從 AB、AD 產生 ABD，從 AC、AD 產生 ACD，但是由於 BD 不存在 L_2 中，所以 ABD 不能成為 candidate。同理，CD 不存在 L_2 中，所以 ACD 不能成為 candidate。

流程圖（II-3.1）為〝temporal apriori〞的演算法概述，其中一個 basic time interval 為分割資料庫的最小時間單位。Temporal apriori 採用 apriori 的架構，一層一層逐漸產生 frequent calendar patterns 所包含的 large itemsets。首先第一次掃瞄資料庫，針對每個 basic time interval 找出所有的 large 1-itemsets，然後對存在於涵蓋到此 basic time interval 的 calendar patterns 裡面的這些 large 1-itemsets 加以統計出現在 calendar patterns 的週期中是 large 的次數。例如在一個 calendar schema R=（year:{1996,…,2000}，month:{1,…,12}，day:{1,…,31}）中，items A、B、C 在（2000，10，20）裡面是 large，而涵蓋到（2000，10，20）的 calendar patterns 有（＊，10，20）、（2000，＊，20）、（2000，10，＊）、（＊，＊，20）、（＊，10，＊）、（2000，＊，＊）六種 calendar patterns，因此這六種 calendar patterns 裡的 A、B、C 在掃瞄資料庫到（2000，10，20）時，其次數得以累加。最



流程圖（II-3.1）Temporal apriori 的演算法

後在第一次掃瞄資料庫之後，計算完每個 calendar patterns 裡面所有 1-itemsets 出現在該週期內的次數，再把滿足使用者所設定的 match ratio threshold 的 1-itemsets 保留下來。則這些被保留下來的 large 1-itemsets 即是具有所存在的 frequent calendar patterns 這些週期的 large 1-itemsets。

在第二層之後，每一層在每個 basic time interval 中分為三個部分。1.產生 candidate itemsets。2.計算 candidate itemsets 在此 basic time interval 中出現的 support。3.以出現的 large itemsets 更新包含到此 basic time interval 的 calendar patterns 裡面這些 large itemsets 的出現次數。在第二層之後一直重複相同的動作，直到沒有任何 calendar pattern 有新的 large itemset 產生為止。

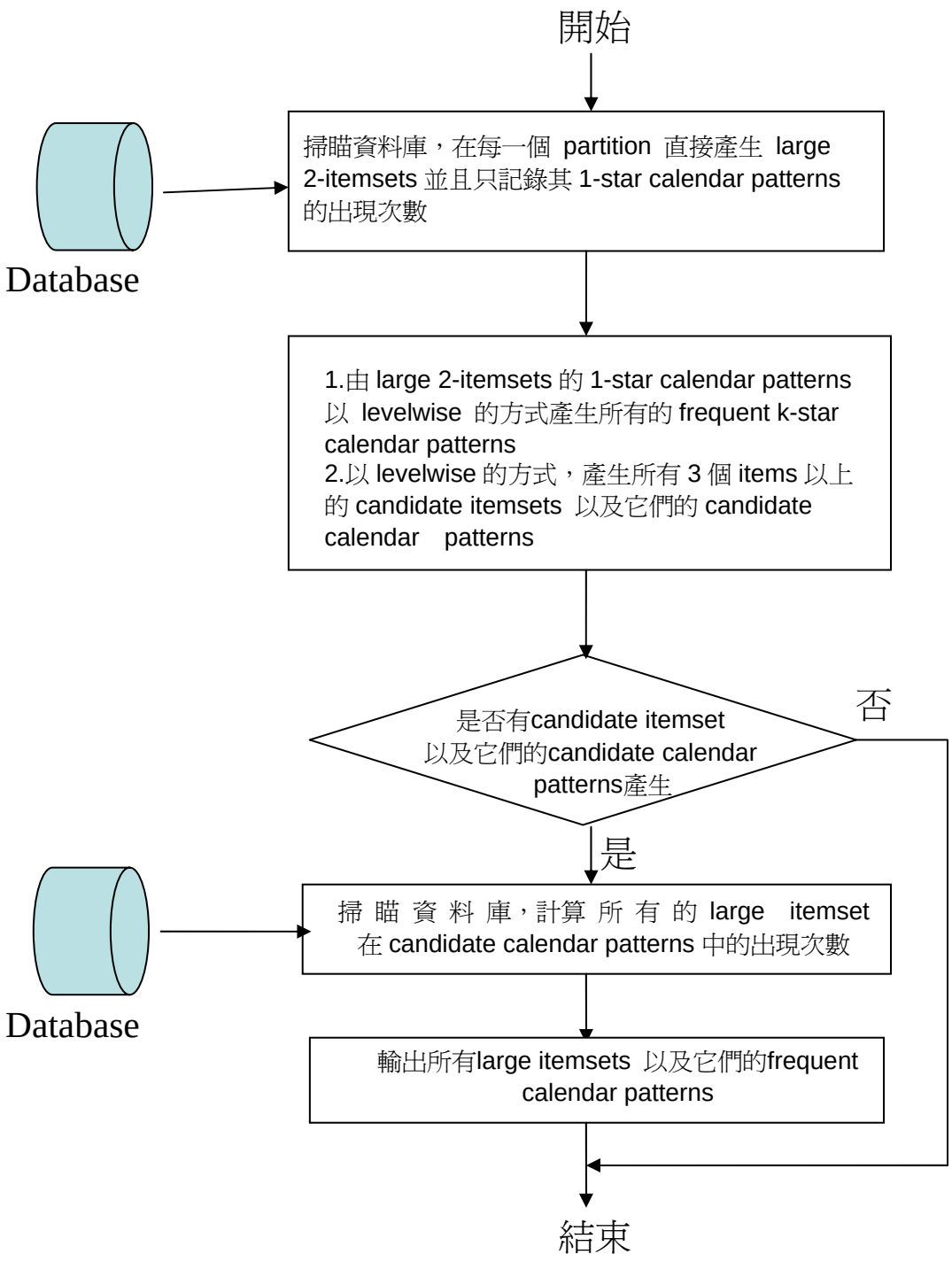
Temporal apriori 以提供 calendar schema 的方式來同時找尋多種不同的週期，讓使用者不需要一個一個週期去嘗試尋找其中的 association rules，可以一次得到許多具有不同週期的 association rules。但是 temporal apriori 在執行效率上還有可以改善的空間，第一、與 apriori 相同，掃瞄資料庫的次數決定於最大的 large itemset 的長度，可能需要花費大量掃瞄資料庫的時間。第二、在每個 basic time interval 中為了產生 candidates，必須記錄前一次所產生的 large itemsets，這些數量會隨著 basic time intervals 的數量成長，可能會非常的龐大。第三、在掃瞄資料庫的過程中，必須保留所有的 calendar patterns 並且對其做更新的動作。因此，為了能夠更有效率的達成目的，在下一節，我們設計一個新的演算法來進行 calendar-based association rules mining。

3.2 Our method

在一般的情況下，一條 association rule 至少要由 2 個 item 以上的 large itemset 產生。而且，在 apriori-like 的方法中，我們可以發現在比較產生的 candidates 跟得到的 large itemsets 的時候，往往在 2-itemset 的 candidates 中具有最多多餘的 candidates，而且當 support threshold 越低的時候，相對的多餘的 2-itemset 的 candidates 就會越多。因此，我們的演算法直接從 large 2-itemset 開始找起，而不是從 large 1-itemsets 產生 2-itemset 的 candidates 再產生 large 2-itemsets。

我們設計一個新的演算法來達到有效的從資料庫中找出所有 large itemsets 的 frequent calendar patterns 的目的。流程圖（II-3.2）是我們演算法的概述，我們的演算法一開始先進行第一次資料庫的掃瞄，在這個步驟中我們要找出所有出現在 basic time interval e_0 所包含資料分割 $P(e_0)$ 中的 large 2-itemsets L_2，以及 L_2 所具有的 1-star calendar patterns $L_2.E_1$。在這個階段中，最初的 L_2 與 $L_2.E_1$ 裡面沒有任何的物件，在每一次從一個 basic time interval e_0 所包含的資料分割 $P(e_0)$ 中，對每筆 transaction 內的 items 兩兩組合來計算所有 2-itemsets 在 $P(e_0)$ 中出現的次數，找出在 e_0 內所有的 large 2-itemsets $L_2^{e_0}$，此時假設所有涵蓋 e_0 的 1-star calendar patterns 為 $E_1^{e_0}$。然後，考慮以下情形：

第一，一個 large 2-itemset $l_2 \in L_2^{e_0}$，但是 $l_2 \notin L_2$，此時將 l_2 加入 L_2 中，即 $L_2 = L_2 \cup l_2$，並且將涵蓋到此 basic time interval e_0 的 1-star calendar patterns $E_1^{e_0}$ 加入 l_2 的 1-star calendar patterns 集合 $l_2.E_1$ 中，即 $l_2.E_1 = l_2.E_1 \cup E_1^{e_0}$，令這些新加入的 1-star calendar



流程圖（II-3.2）我們的方法之演算法

patterns 的計數值為 1。

第二，一個 large 2-itemset $l_2 \in L_2^{e_0}$ 而且 $l_2 \in L_2$ ，但是存在 $e_1^{'} = E_1^{e_0} - l_2.E_1$ ，$e_1^{'} \neq \phi$ ，“-”在此為差集。即存在有涵蓋 e_0 的 1-star calendar patterns $e_1^{'}$ 不存在於該 l_2 所具有的 1-star calendar patterns $l_2.E_1$ 中。此時將 $e_1^{'}$ 加入 $l_2.E_1$ 中，即 $l_2.E_1 = l_2.E_1 \cup e_1^{'}$ ，並且初始這些 1-star calendar patterns 的計數值為 1。

第三，一個 large 2-itemset $l_2 \in L_2^{e_0}$ 而且 $l_2 \in L_2$ ，且存在 $e_1^{'} = E_1^{e_0} \cap l_2.E_1$ ，$e_1^{'} \neq \phi$ ，即存在有涵蓋 e_0 的 1-star calendar patterns $e_1^{'}$ 亦存在於該 l_2 所具有的 1-star calendar patterns $l_2.E_1$ 中。此時將 $l_2.E_1$ 中的 $e_1^{'}$ 這些 1-star calendar patterns 的計數值累加。

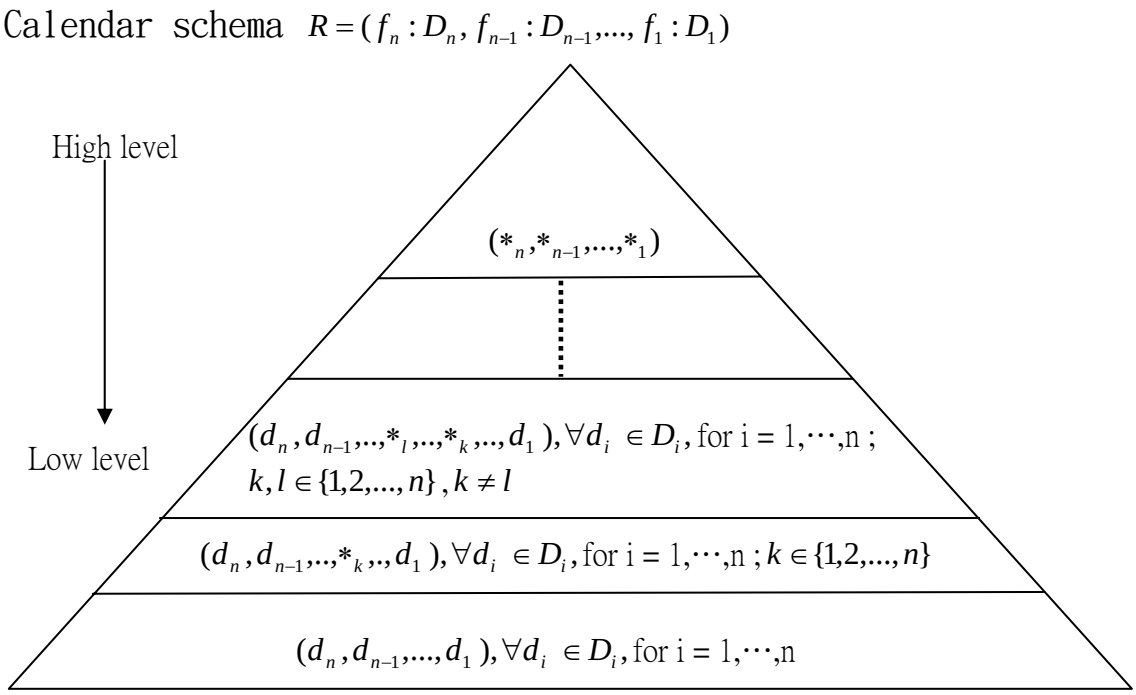
在第一次掃瞄資料庫並且在每一個 basic time interval 做上述的處理之後，將可以得到所有 large 2-itemsets L_2 以及它們所具有的 1-star calendar patterns $L_2.E_1 = \bigcup l_2.E_1$, $\forall l_2 \in L_2$ 。

由於我們所使用的 calendar schema 是一種階層式的構造，如圖（II-3.1），在這個構造中我們可以由較低階層的 calendar patterns 來組成較高階層的 calendar patterns。列式如下：

$$\begin{aligned} &\sum_{j=1}^{r}(*_n,*_{n-1},.....,*_{m+1},d_m^j,d_{m-1},....,d_1) \\ &=(*_n,*_{n-1},.....,*_{m+1},*_m,d_{m-1},....,d_1) \\ &\forall d_m^j \in D_m, |D_m| = r \ , \ d_i \in D_i, i = 1,....,m-1 \end{aligned} \tag{11}$$

既然我們知道 2-star calendar patterns 的 count 可以從 1-star calendar patterns 的 count 加總產生（如圖（II-3.2）中 AB（＊,＊,2）的 count 即由 AB（2000,＊,2），AB（2001,＊,2），AB（2002,＊,2），AB（2003,＊,2）的 count 加總產生）。那麼就沒有必要在掃瞄資料庫的過程中紀錄這些 2-star calendar patterns 的資訊，以減少在更新 calendar patterns 的 count 時所要搜尋的數目，達到

增進效率的目的。因此，我們可以藉L_2所具有的 1-star calendar patterns $L_2.E_1$來產生所有L_2的 frequent calendar patterns。



圖（II-3.1）Calendar schema的階層式架構

在第一次的資料庫掃瞄之後，我們得到了所有 large 2-itemsets 以及它們的 frequent calendar patterns，為了在第二次掃瞄資料庫時可以更快的得到所有的 large itemsets 跟它們的 frequent calendar patterns，我們要在掃瞄資料庫之前由式（12）透過 apriori_gen的方式產生所有的candidate itemsets，其中candidate 3-itemsets 由 large 2-itemsets 產生，而 3 以上的 candidate itemsets則由前一層的candidate itemsets產生。並由式（13）透過交集產生這些itemsets的candidate calendar patterns，式(13)為apriori_gen原理的的衍生，即

〝若一集合的子集合不具有 frequent calendar pattern e，則此集合亦不具有 frequent calendar pattern e〞

所以我們可以用式（13）來減少 candidate calendar patterns 的產生。

$$C_k = \begin{cases} apriori\_gen(L_2) & if \quad k = 3 \\ apriori\_gen(C_{k-1}) & if \quad k > 3 \end{cases} \tag{12}$$

$$c_k(e) = \bigcap(\{c_{k-1}(e) \mid \quad \forall c_{k-1} \subset c_k\}) \tag{13}$$

其中 apriori_gen 以[2]裡面的方式產生，可參考前一節的描述。c_k 為 C_k 內一個 candidate k-itemset，$c_k(e)$ 為 k-itemset c_k 的 candidate calendar patterns。

但是由於我們有 large 2-itemsets 的 1-star calendar patterns 的完整資訊，所以可以利用這些已知的訊息來減少多餘的 candidates。我們在產生 3-itemsets 的 2-star calendar patterns 的 candidates 時，由於我們已經知道 large 2-itemsets 的 1-star calendar patterns 的 count 數 $e_1^i.count$，其中 i 表示＊出現的位置。可以利用他們來估計 large 3-itemsets 在相同的 1-star calendar pattern e_1^i 中會出現的最大值，假設 c_3 為 C_3 中的一個 3-itemset，則它在 e_1^i 中會出現的最大值 $\hat{c}_3.e_1^i.count$ 如式（14）：

$$\hat{c}_3.e_1^i.count = Min\{c_2.e_1^i.count \mid \forall c_2 \subset c_3\} \tag{14}$$

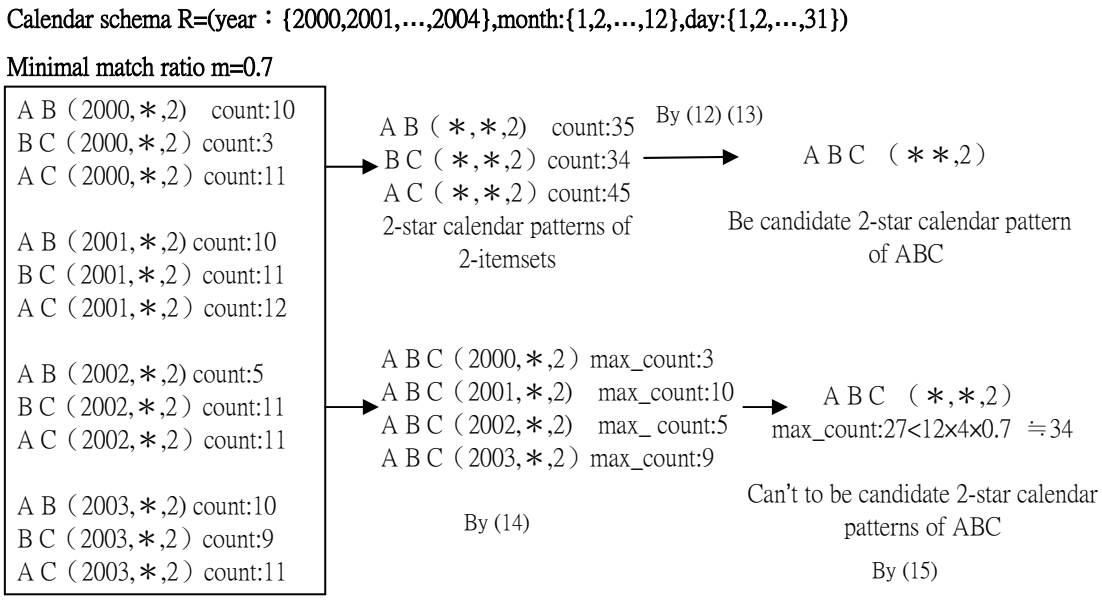
因為如果一個 3-itemset 在一個 basic time interval 中會出現，則其組成它的所有 2-itemsets 皆會出現，反之，如果有組成它的 2-itemsets 中的任何一個沒有出現的話，該 3-itemset 便不會出現在該 basic time interval，因此可以從 2-itemset 出現在 calendar pattern 中的最少次數來推斷由此 2-itemset 組成的 3-itemset 可能出現在 calendar pattern 中的最大值。所以 c_3 在 1-star calendar

pattern e_1^i 出現的最大次數為其全部的子集合會一起出現的最大值，亦即其子集合中出現次數最少的集合之e_1^i計數值。而c_3的 2-star calendar pattern e_2^{mi}所可能出現次數的最大值可以由式（11）衍生之式（15）來計算，其中 m,i 表示＊出現的位置，列式如下：

$$\sum_{j=1}^{r}(d_n, d_{n-1}, ..., d_m^j, ..., d_{i+1}, *_i, d_{i-1}, ..., d_1)$$
$$= (d_n, d_{n-1}, ..., d_{m+1}, *_m, d_{m-1}, ..., d_{i+1}, *_i, d_{i-1,}, ..., d_1) \quad (15)$$
$$\forall d_m^j \in D_m, |D_m| = r,\ d_i \in D_i, i = 1, ..., n$$

所以我們可以透過式（14）及（15）來計算出 3-itemsets 的 calendar pattern 計數值可能的最大值，並去除此最大值還不能滿足 match ratio 的 candidate calendar patterns 來減少 candidates 的數量。如此一來，我們可以找到更接近真正 large 3-itemset 的 calendar patterns 的 candidates。圖（II-3.2）是一個以式（14）（15）減少式（12）（13）有可能產生多餘的 candidates 的一個例子。

在得到所有的 candidates 之後我們要再一次從頭掃瞄資料庫一次，計算所有 large itemsets 出現在其 calendar patterns 的次數。當我們在掃瞄資料庫時，在每一個時間區間中，由於我們已經在掃瞄之前將全部的 candidates 找出來，所以不需要像[7]一樣還要再透過 apriori-like 的方式由前一次掃瞄後的結果來產生新的 candidates，只要直接從第二部分產生的 candidates 中提出涵蓋到現在的 basic time interval 的 itemsets，就可以對這些 candidate itemsets 計算其出現的次數，看是否在此區間內是 large，如果是 large，則將此 large itemset 所具有涵蓋到此區間的 calendar patterns 的 count 做累加。



圖（II-3.2）Calendar candidate patterns pruning 的例子

最後將所有符合 minimal support 以及 minimal match ratio 要求的 large itemsets 和它們的 frequent calendar patterns 輸出。即為達到使用者要求，具有週期 frequent calendar patterns 的 large itemsets。

我們的演算法透過限制掃瞄資料庫的次數在最多兩次以減少資料庫存取所需花費的時間，並且利用時間週期上的特性來減少在掃瞄資料庫時所要搜尋的 candidates 的數量，以此來提高執行的速度。在第 III 部分中，我們將以實驗的結果來佐證這些想法。

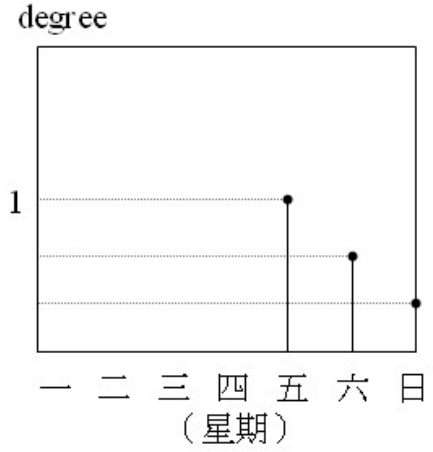
第三章、 Mining fuzzy periodical association rules

1. Introduction

Calendar-based periodical association rules 雖然可以提供使用者採掘具有週期性的 association rules，但是這些 association rules 必須準確的發生在該週期的正確時間點上，完全不考慮該週期以外的時間點所發生的 association rules 對於此週期的影響。這樣的作法雖然可以找到一些令人感興趣的具有週期性的 association rules，但是卻也因此少了一些彈性。在現實世界中，我們可以發現，往往有很多的週期性行為會有某個程度的時間位移產生。例如，雖然某甲喜歡看每個星期五出刊的某雜誌，但是他可能有時候準時在該雜誌星期五出刊的時候去購買，也有可能是星期六或者是星期日才想要去購買。但是對於某甲會在該雜誌出刊後去購買的這個行為，由於分散在星期五、星期六跟星期日發生，所以透過 calendar-based periodical association rules mining 的方法可能無法發現具有這樣的規律的 rules。

因此，考慮到 association rules 在時間週期中存在的位移現象（非同步性質），本論文發展新的 periodical association rules---fuzzy periodical association rules--- 一種透過 membership function 所定義的 fuzzy calendar pattern 來決定所要找尋的 periodical association rules 的週期表現形式及所涵蓋到的週期範圍。例如，可以利用稱之為 〝靠近〞的 membership function 來找出發生在該週期附近，靠近該週期的 association rules，並且可以設定〝靠近〞的範圍。以上述某甲買在星期五出刊

的雜誌的行為來說，可以定義〝靠近每個星期五〞此 fuzzy calendar pattern 的範圍為星期五到星期日，其 membership function 如圖（III-1.1）：



圖（III-1.1）Membership function 一個例子

如此一來，我們就可以透過這樣的 membership function 來計算符合〝靠近每個星期五時購買該雜誌〞這樣的非同步週期行為的程度，並且透過設定 threshold 來找出具有相當程度的這種非同步週期行為。在本論文中將提供一個初步找尋這種 fuzzy periodical association rules 的演算法。

2. Problem definition

2.1 Fuzzy calendar pattern

我們使用 fuzzy calendar pattern 來表示在週期時間中具有非同步情形的 calendar pattern。在前面的部分，在描述一般的 calendar pattern 的時候，我們使用布林函數來表示該 calendar

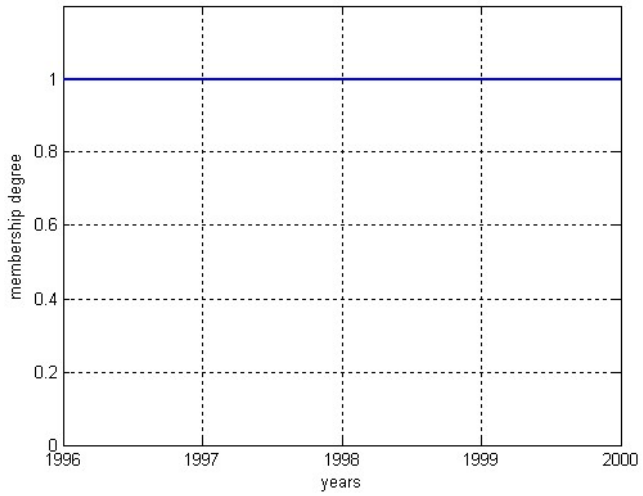
pattern 出現的範圍，有出現時即為 1，沒有出現的 calendar 即為 0，且根據定義，描述其各時間單位的布林函數只會是一個 singleton（$d_i \in D_i$) 或者是輸出恆為 1（$d_i = *$) 的函數。而為了找尋 fuzzy periodical association rules ，我們使用 membership function 來描述 fuzzy calendar patterns，其函數值在 0 到 1 之間，以其值的大小來表示一個 time interval 對於此 fuzzy calendar pattern 的貢獻度大小。而在一個 calendar schema $R = (f_n : D_n, f_{n-1} : D_{n-1}, ..., f_1 : D_1)$ 中，使用者定義在 calendar schema 下的 fuzzy calendar pattern FC 的 membership function F ＝$(F_n, F_{n-1}, ..., F_1)$，其中 F_n 為對應到 calendar schema 中時間單位 f_n 的 membership function。而一個 basic time interval $(d_n, d_{n-1}, ..., d_1), d_i \in D_i, i = 1, ..., n$，其對於 fuzzy calendar pattern 的貢獻度則由下式計算：

$$\deg ree_{FC}(d_n, d_{n-1}, ..., d_1) = F_n(d_n) \times F_{n-1}(d_{n-1}) \times ..... \times F_1(d_1) \qquad (16)$$

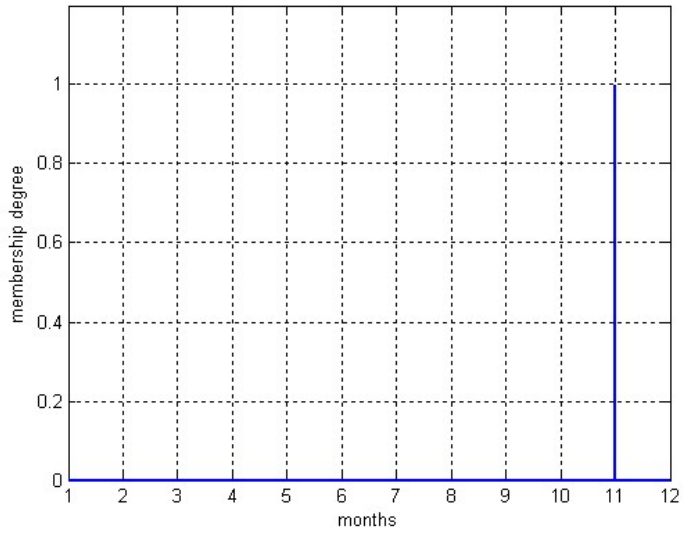
在圖（III-2.1）中我們以一個 fuzzy calendar pattern 〝close to（＊，11，25）〞為例，以 membership function $F = (F_{year}, F_{month}, F_{day})$ 來描述此 fuzzy calendar pattern。在此我們假設在每一年的 11 月 25 日前後兩天的範圍內才具有『靠近每一年的 11 月 25 日』這樣的關係。

2.2 Fuzzy periodical association rule

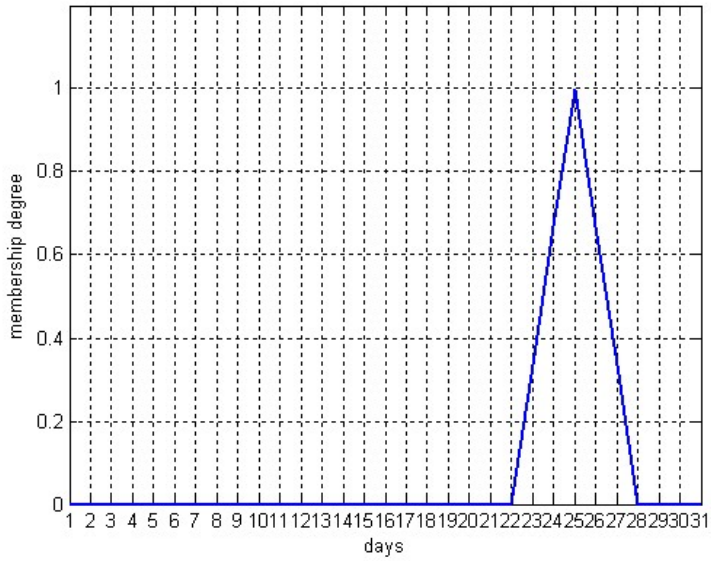
fuzzy periodical association rule：給定一個 calendar schema $R = (f_n : D_n, f_{n-1} : D_{n-1}, ..., f_1 : D_1)$ 跟一組以 basic time interval 分割的資料庫 D，還有一個 fuzzy match ratio threshold fm(0<fm≦1)，以及使用者定義在 calendar schema 下的 fuzzy calendar



Membership function F_{year}



Membership function F_{month}



Membership function F_{day}

圖（III-2.1）以 membership function 來表示 close to（＊，11，25）

pattern FC ＝ 〝close to（$d_n,d_{n-1},....,d_1$）″（其中 $d_i \in D_i \cup *, i=1,2,....,n$，即（$d_n,d_{n-1},....,d_1$）為一個一般的 calendar pattern）的 membership function F＝$(F_n,F_{n-1},....,F_1)$，其中 F_n 為對應到 calendar schema 中 f_n 的 membership function。透過 membership function 計算對於 fuzzy calendar pattern 的貢獻度來推論一條 association rule 是否具有 fuzzy calendar pattern FC 這樣的一組在 calendar schema R 內的非同步週期。

為了知道一條 association rule 是否存在 fuzzy calendar pattern FC ＝ 〝close to（$d_n,d_{n-1},....,d_1$）″ 這樣的非同步週期，我們利用 fuzzy rule-based inference 的步驟來進行推論。首先定義在一個 basic time interval t_i =（$d_n^{'},d_{n-1}^{'},....,d_1^{'}$）其中 $d_j^{'} \in D_j, j=1,2,....,n$，推論其〝close to（$d_n,d_{n-1},....,d_1$）″ $d_j \in D_j \cup *, j=1,2,....,n$ 的 fuzzy if-then rules 如下：

〝If $d_n^{'}$ is close to d_n AND $d_{n-1}^{'}$ is close to d_{n-1} AND …AND $d_1^{'}$ is close to d_1 THEN t_i is close to（$d_n,d_{n-1},....,d_1$）″

透過使用者定義在 calendar schema 下的 fuzzy calendar pattern FC＝〝close to（$d_n,d_{n-1},....,d_1$）″ 的 membership function F ＝$(F_n,F_{n-1},....,F_1)$，可以得到 $d_j^{'}$ is close to d_j 的 fuzzy matching degree $F_j(d_j^{'})$，$j=1,2,...n$。並利用 AND 運算以乘法對 t_i 內每個成員的 fuzzy matching degree 值連乘來計算 t_i 對於非同步週期 FC 的 fuzzy matching degree 值（如式 16）。

當所有的 basic time interval 對於 FC 此非同步週期的 fuzzy matching degree 值都計算出來之後，我們計算 association rule 存

在於 FC 此週期的 fuzzy match ratio，看這樣的 fuzzy periodical association rule 是否能滿足使用者所設定的 fuzzy match ratio threshold，即它具有此非同步週期的程度是否夠強烈來決定其是否成立。

假設每個 basic time interval t_i 透過 fuzzy matching 得到對於 FC 的 degree 值 $\deg ree_{FC}(t_i)$，代表此 basic time interval 對於屬於非同步週期〝close to P〞（其中 P=($d_n, d_{n-1}, ..., d_1$)，$d_i \in D_i \cup *, i = 1,2,....,n$，即 P 為一個一般的 calendar pattern）的貢獻度，則一個 association rule 具有此非同步週期的 fuzzy match ratio FM 以下式計算

PD = $\sum \deg ree_{FC}(t_i) \times existed \ / \sum \deg ree_{FC}(t_i)$, where t_i is covered by p

FD = $\sum \deg ree_{FC}(t_i) \times existed \ / \sum \deg ree_{FC}(t_i)$, where t_i is not covered by p

FM = PD + αFD, 其中 $0 \leqq \alpha \leqq 1$ (17)

其中 existed 代表該 association rule 是否在 t_i 中出現，是則 existed 值為 1，否則為 0，PD 為出現在一般的 calendar pattern 週期中的比例，即為一般的 match ratio，FD 為具有異位特徵的時間區間出現的程度，α 為使用者希望異位情況對於整個 fuzzy calendar pattern 所佔的權重，α 值越大則異位情況的影響就越大，如果 α 值等於零，則異位情況將不列入考慮，FM 即為一般的 calendar pattern 週期的 match ratio。如此一來，將可以利用 PD 來保留一般 calendar pattern 所能找到的 association rules，另外再加上 FD 來找出具有異位情形的 association rules，進而達成我們所要的目的。

如果推論值 FM 大於使用者所設定之 fuzzy match ratio threshold fm，則我們稱在 calendar schema $R=(f_n:D_n, f_{n-1}:D_{n-1},..., f_1:D_1)$ 的一個 fuzzy periodical association rule 表示如下

$$x \xrightarrow{FC} y \qquad (18)$$

表示 association rule x->y 具有 FC(即〝close to $(d_n,d_{n-1},....,d_1)$〞)這樣的非同步週期存在。

舉例來說，在 calendar schema R=（year:{1996,…,2000}，month:{1,…,12}，day:{1,…,31}）之下我們可以定義〝close to（＊，11，25）〞的 membership function $F=(F_{year},F_{month},F_{day})$ 如圖（III-2.1）所示，其中定義具有〝close to（＊，11，25）〞的範圍為每一年 11 月 25 日的前後兩天，再經過 fuzzy matching degree 的運算後，只有（＊，11，23）、（＊，11，24）、（＊，11，25）、（＊，11，26）、（＊，11，27）這些天才具有非零的 degree，也就是說除了這些日期之外的其他日期都不具有〝close to（＊，11，25）〞的關係，因此在掃瞄資料庫時可以跳過這些不具有〝close to（＊，11，25）〞的資料。例如（1998，11，20）其〝close to（＊，11，25）〞的 degree = $(F_{year}(1998)\times F_{month}(11)\times F_{day}(20))=1\times 1\times 0=0$，因此不用考慮（1998，11，20）裡面的資料。

假設我們要找具有〝close to（＊，11，25）〞這樣週期的 association rules，在掃瞄過資料庫之後發現，association rule〝火雞→南瓜派〞存在（1996，11，23）、（1996，11，24）、（1996，11，26）、（1997，11，25）、（1997，11，26）、（1998，11，24）、（1998，11，26）（1998，11，27）、（1999，11，23）、（1999，11，25）、（1999，11，26）、(2000，11，25)、（2000，11，26）、

（2000，11，27）這些日期中，我們就可以得到〝火雞→南瓜派〞具有 〝close to（＊，11，25）〞的 fuzzy match ratio 為

$$PD=\{F_{year}(1997)\times F_{month}(11)\times F_{day}(25)+F_{year}(1999)\times F_{month}(11)\times F_{day}(25)+F_{year}(2000)\times F_{month}(11)\times F_{day}(25)\}/\{F_{year}(1996)\times F_{month}(11)\times F_{day}(25)+F_{year}(1997)\times F_{month}(11)\times F_{day}(25)+F_{year}(1998)\times F_{month}(11)\times F_{day}(25)+F_{year}(1999)\times F_{month}(11)\times F_{day}(25)+F_{year}(2000)\times F_{month}(11)\times F_{day}(25)\}$$

$$=(1+1+1)/5=0.6$$

$$FD=\{F_{year}(1996)\times F_{month}(11)\times F_{day}(23)+F_{year}(1996)\times F_{month}(11)\times F_{day}(24)+F_{year}(1996)\times F_{month}(11)\times F_{day}(26)+F_{year}(1997)\times F_{month}(11)\times F_{day}(26)+F_{year}(1998)\times F_{month}(11)\times F_{day}(24)+F_{year}(1998)\times F_{month}(11)\times F_{day}(26)+F_{year}(1998)\times F_{month}(11)\times F_{day}(27)+F_{year}(1999)\times F_{month}(11)\times F_{day}(23)+F_{year}(1999)\times F_{month}(11)\times F_{day}(26)+F_{year}(2000)\times F_{month}(11)\times F_{day}(26)+F_{year}(2000)\times F_{month}(11)\times F_{day}(27)\}/\{F_{year}(1996)\times F_{month}(11)\times F_{day}(23)+F_{year}(1996)\times F_{month}(11)\times F_{day}(24)+F_{year}(1996)\times F_{month}(11)\times F_{day}(26)+F_{year}(1996)\times F_{month}(11)\times F_{day}(27)+F_{year}(1997)\times F_{month}(11)\times F_{day}(23)+F_{year}(1997)\times F_{month}(11)\times F_{day}(24)+F_{year}(1997)\times F_{month}(11)\times F_{day}(26)+F_{year}(1997)\times F_{month}(11)\times F_{day}(27)+F_{year}(1998)\times F_{month}(11)\times F_{day}(23)+F_{year}(1998)\times F_{month}(11)\times F_{day}(24)+F_{year}(1998)\times F_{month}(11)\times F_{day}(26)+F_{year}(1998)\times F_{month}(11)\times F_{day}(27)+F_{year}(1999)\times F_{month}(11)\times F_{day}(23)+F_{year}(1999)\times F_{month}(11)\times F_{day}(24)+F_{year}(1999)\times F_{month}(11)\times F_{day}(26)+F_{year}(1999)\times F_{month}(11)\times F_{day}(27)+F_{year}(2000)\times F_{month}(11)\times F_{day}(23)+F_{year}(2000)\times F_{month}(11)\times F_{day}(24)+F_{year}(2000)\times F_{month}(11)\times F_{day}(26)+F_{year}(2000)\times F_{month}(11)\times F_{day}(27)\}=\{0.33+0.67+0.67+0.67+0.67+0.67+0.33+0.33+0.67+0.67+0.33\}/\{0.33+0.67+0.67+0.33+0.33+0.67+0.67+0.33+0.33+0.67+0.67+0.33+0.33+0.67+0.67+0.33+0.33+0.67+0.67+0.33+0.33+0.67+0.67+0.33\}=6.01/10=0.601$$

$FM = PD + \alpha FD$

假設令 α=0.3，則 FM = 0.6 + 0.3*0.601 = 0.6 + 0.1803 =0.7803 ，因此，當 fuzzy match ratio threshold 設定的值比 0.7803 小的話，那麼〝火雞→南瓜派〞這條規則具有 〝close to（＊，11，25）〞

這樣的非同步週期，表示成

$$\text{火雞} \xrightarrow{close\_to(*,11,25)} \text{南瓜派}$$

即在靠近每一年的 11 月 25 日時，存在著購買火雞的人常常會伴隨著購買南瓜派，這樣的一條非同步週期規則。

3. Algorithm

本論文提供一個方法讓使用者可以以查詢的方式，針對其感興趣的週期定義 fuzzy calendar pattern 的 membership function 來進行 fuzzy periodical association rules mining。

首先，使用者可以在 calendar schema 中設計他感興趣的 fuzzy calendar pattern 的 membership function，例如在 calendar schema R=(year:{1996,…,2000}，month:{1,…,12}，day:{1,…,31}) 中，〝close to （＊，11，25）〞此週期的 membership function 可以設計如圖（III-2.1），其中定義具有〝close to（＊，11，25)〞的範圍為前後兩天，即為（＊，11，23）、（＊，11，24）、（＊，11，25）、（＊，11，26）、（＊，11，27）這些天才具有〝close to（＊，11，25)〞此週期的關係，並依照靠近的程度有不同的 fuzzy matching degree。

我們利用類似在找尋 calendar-based periodical association rules 時使用的架構，使用最多掃瞄資料庫兩次的方法，在開始掃瞄資料庫之前先根據使用者對於感興趣的 fuzzy calendar pattern 定義的 membership function，找出所有具有 fuzzy matching degree 程度的 basic time intervals，然後在掃瞄資料庫時只要對這些具有 fuzzy matching degree 程度的 basic time intervals 做處理即可。

在每次掃瞄資料庫時，我們只考慮具有 fuzzy matching degree 貢獻度的 basic time intervals，首先在第一次掃瞄資料庫的時候，從這些 basic time intervals 找出 large 2-itemsets，然後看 basic time interval 是屬於一般的 calendar pattern 或屬於具有異位的 time interval，然後累加這些 large itemsets 對於此 fuzzy calendar pattern 的 fuzzy matching degree 值到 PD 或 FD，如果一個 large itemset 在一個 basic time interval 中，則他的 count 將加上此 basic time interval 對於此 fuzzy calendar pattern 的 fuzzy matching degree 值，如果此 large itemset 不存在此 basic time interval 中，則 count 不變。

在第一次掃瞄完資料庫之後，可以將 large itemset 的 PD 與 FD 個別除以他們具有的 fuzzy matching degree 總和，並且以式 (17) 來做 fuzzy match ratio 的運算，留下滿足使用者要求的 large 2-itemsets。這樣就把具有非同步週期的 large 2-itemsets 找出來了。

在第二次掃瞄資料庫之前，以 apriori_gen 的方式產生 3 以上的 candidate itemsets(如式 (12))。然後進行第二次資料庫的掃瞄，與第一次掃瞄時相同，只要考慮具有 fuzzy matching degree 程度的 basic time intervals，並對這些的 PD 與 FD 做 fuzzy matching degree 的累加。

最後再以式 (17)來做 fuzzy match ratio 的運算，留下滿足使用者設定的 fuzzy match ratio threshold 的 large itemsets。則這些 large itemsets 與前面產生的 large 2-itemsets 即為具有使用者查詢的 fuzzy calendar pattern 週期的 large itemsets。

第四章、 Experiments

在本節中，我們藉由幾個實驗來比較我們的方法跟[7]的方法〝Temporal_apriori〞以及觀察 fuzzy periodical association rule mining 是否真可以找出具有非同步週期性質的 large itemsets。我們使用機器配備 CPU2.20Hz,1.0G memory,使用的作業系統是 Windows 2000，程式軟體為 Matlab 6.5。在實驗中我們使用跟[2]相同產生方法的 synthetic data 來當作輸入的資料。在實驗一中使用四種不同的資料 T10.I3.D600K,T10.I4.D600K,T10.I5.D600K,T10.I6.D600K，透過調整 support threshold 來比較在不同的 support threshold 下，兩種方法在執行時間上的差異。在實驗二中，使用四種大小 T10.I4.D400K，T10.I4.D600K，T10.I4.D800K，T10.I4.D1000K 的資料，比較在不同的 support threshold 下，這兩個方法在 data size 變動時相對執行效能的變化。其中 T 代表平均的 transaction 大小，I 代表 potential maximal large itemsets 的平均大小，D 代表 data 所含的 transaction 數量，K 為代表 1000 的單位，資料量從 47MB 到 117MB，使用 calendar schema R=(year : {1,2,3,4},month:{1,2,…,12},day:{1,2,…,31})。

在實驗三中，我們比較在本論文中提出的方法裡，在第一次掃瞄資料庫的時候，只保留 1-star calendar patterns 與保留所有的 calendar patterns 數量上面的差異。在實驗四中，我們使用設計過的 data 來說明 fuzzy periodical association rule mining 可以找到一般的 periodical association rule mining 所無法找出的具有非同步週期的 frequent itemsets。

Experiment 1

我們使用四種不同的資料 T10.I3.D600K，T10.I4.D600K，T10.I5.D600K，T10.I6.D600K 在每個時間單位內平均含有 1000 筆 transactions，針對不同的 support threshold（0.06，0.07，0.08，0.09，0.1）來看兩種方法在 support threshold 改變時執行效率上的差異，在此實驗中所設定的 match ratio threshold 為 0.8。

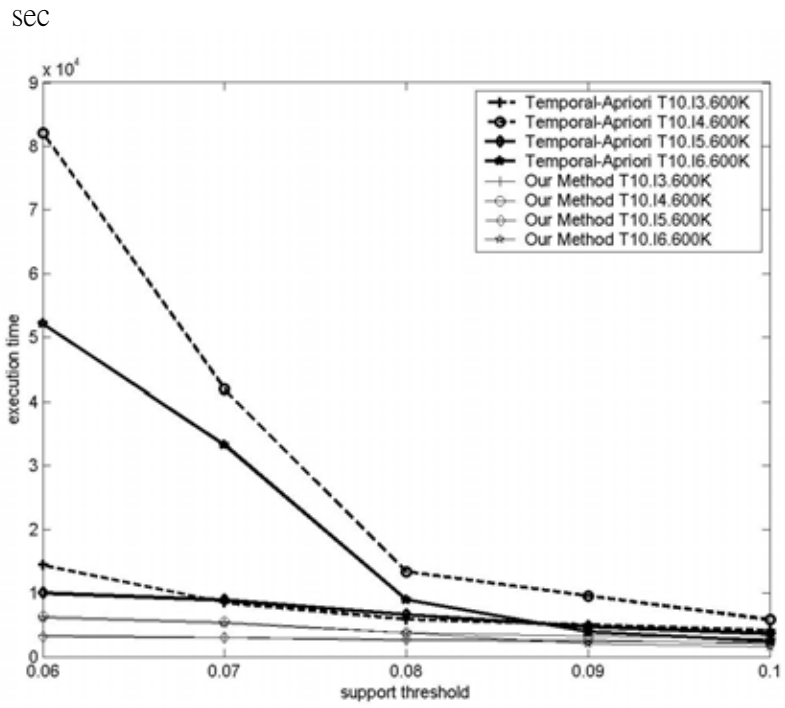
實驗結果

表（IV-1.1）為在不同的 support threshold 下，兩種方法在四個不同的資料庫產生的 candidate calendar patterns 與所採掘出的 frequent calendar patterns 的數量以及所產生的 frequent itemsets 的最大長度，圖（IV-1.1）為在不同的 support threshold 下，兩種方法在四個不同的資料庫的執行時間。

對照表(IV-1.1)及圖(IV-1.1)之後我們可以發現，在 frequent itemsets 的最大長度越大時，temporal apriori 的方法其執行速度會隨著 candidate calendar patterns 的增加而急遽增加，而在 frequent itemsets 的最大長度越小時，其執行速度的增加便趨於緩慢（由 T10.I4.D600K 與 T10.I5.D600K 比較可以明顯的看出）。因為當 frequent itemsets 的最大長度越大時，temporal apriori 的方法必須掃瞄資料庫越多次，而當平均每次要掃瞄資料庫時所要檢查的 candidate calendar patterns 增加時，其影響就會越明顯。而我們的方法由於頂多只掃瞄資料庫兩次，因此在執行時間上受 frequent itemsets 的最大長度的影響並不明顯，對於 candidate calendar patterns 增加的影響也沒有像 temporal apriori 的方法那麼劇烈。可以說我們的方法在這些情況下皆可以有良好的執行效率。

			Support threshold				
			0.06	0.07	0.08	0.09	0.1
T10.I3.600k	Candidate calendar patterns of	Temporal-Apriori	65511	40023	28333	22485	15878
		Our Method	72992	46105	29990	24935	17839
	Discovered frequent calendar patterns		64844	39596	28071	22206	14727
	Maximal length of frequent itemsets		7	6	6	5	5
T10.I4.600k	Candidate calendar patterns of	Temporal-Apriori	205886	133932	73784	48726	29277
		Our Method	212943	174203	89915	55214	34681
	Discovered frequent calendar patterns		198454	128495	71432	46618	27756
	Maximal length of frequent itemsets		8	8	8	6	6
T10.I5.600k	Candidate calendar patterns of	Temporal-Apriori	56017	45429	34215	18191	6146
		Our Method	55465	46083	36561	25585	9430
	Discovered frequent calendar patterns		52942	43431	31732	16540	5412
	Maximal length of frequent itemsets		6	6	6	6	6
T10.I6.600k	Candidate calendar patterns of	Temporal-Apriori	180697	122007	48047	9787	1313
		Our Method	192262	138016	83417	21123	2407
	Discovered frequent calendar patterns		162640	105666	42128	8968	1068
	Maximal length of frequent itemsets		9	8	8	6	4

表（IV-1.1）Experiment 1 兩種方法產生的 calendar candidate patterns 與所採掘出的 frequent calendar patterns 的數量以及所產生的 frequent itemsets 的最大長度



圖（IV-1.1）Experiment 1 兩種方法的執行時間比較圖

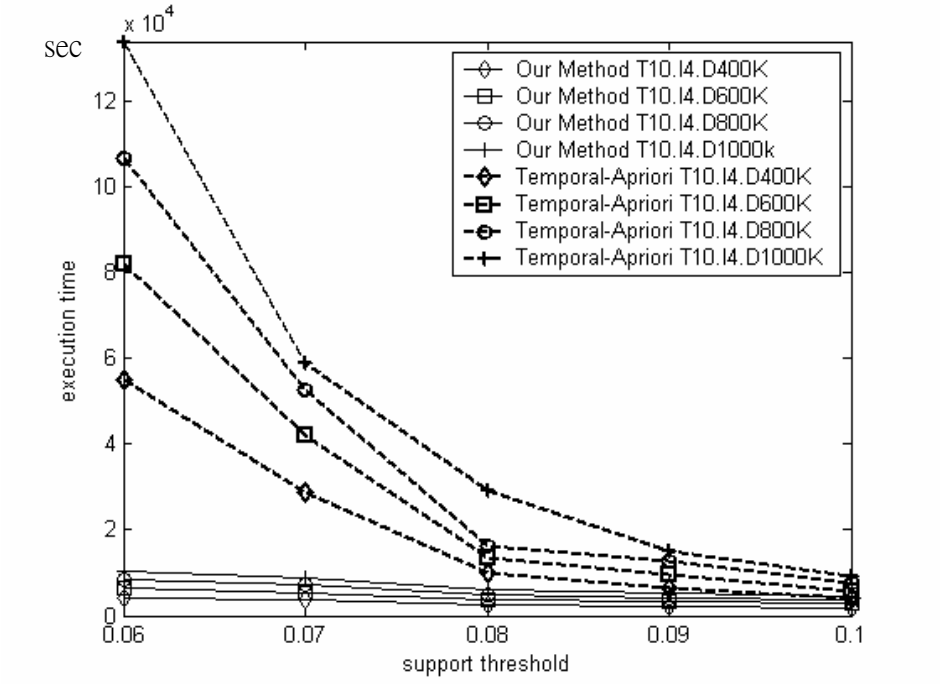
Experiment 2

我們使用四種不同大小 T10.I4.D400K，T10.I4.D600K，T10.I4.D800K，T10.I4.D1000K 的資料，在每個時間單位內平均含有 1000 筆 transactions，針對不同的 support threshold（0.06，0.07，0.08,0.09,0.1）來看兩種方法在 data size 改變時執行效率上的差異，在這個實驗中我們使用的 match ratio threshold 為 0.8。

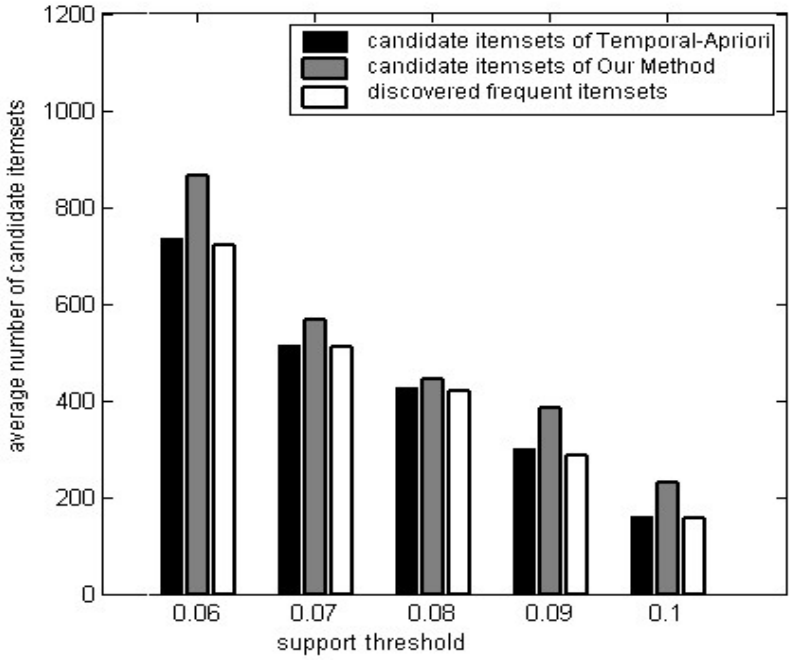
實驗結果

圖（IV-2.1）為在四種不同大小的資料庫大小下，兩種方法對於 support threshold 的變化，其中虛線部分是 temporal apriori 的執行時間，而實線部分是我們的方法的執行時間。圖（IV-2.2）為在不同的 support threshold 下，兩種方法在四種不同大小的資料庫中

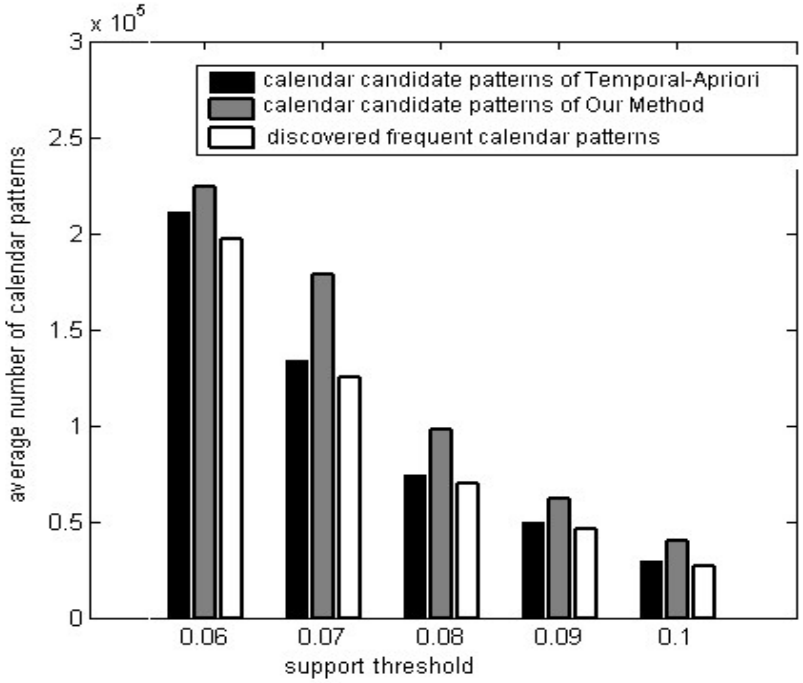
的 candidate itemsets 與所採掘出的 frequent itemsets 的數量。圖（IV-2.3）為在不同的 support threshold 下，兩種方法在四種不同大小的資料庫中的 candidate calendar patterns 與所採掘出的 frequent calendar patterns 的數量。從圖中可以看出在 database size 越大的時候，由於 temporal apriori 的方法掃瞄資料庫的次數是根據 frequent itemset 的最大長度來決定，所以當資料庫越大的時候，相對的需要掃瞄資料庫較多次的 temporal apriori 方法，花費在掃瞄資料庫上的執行時間會隨著資料庫變化而增加。而這樣的變化，在 support threshold 越小時，由於每一次掃瞄資料庫所需檢查的 candidates 越多，因此在每一次掃瞄越大的資料庫時，其所花費的執行時間就增加越多，所以其在執行時間上因為掃瞄資料庫的次數增加所產生的變化將會越明顯。從圖（IV-2.1）很明顯的可以看出，在不同的 data size 之下，我們的方法都可以得到較好的執行效率。



圖（IV-2.1）Experiment 2 兩種方法的執行時間比較圖



圖（IV-2.2）Experiment 2 兩種方法產生的 calendar itemsets 及採掘出的 frequent itemsets 平均數量圖



圖（IV-2.3）Experiment 2 兩種方法產生的 candidate 與 frequent calendar patterns 平均數量圖

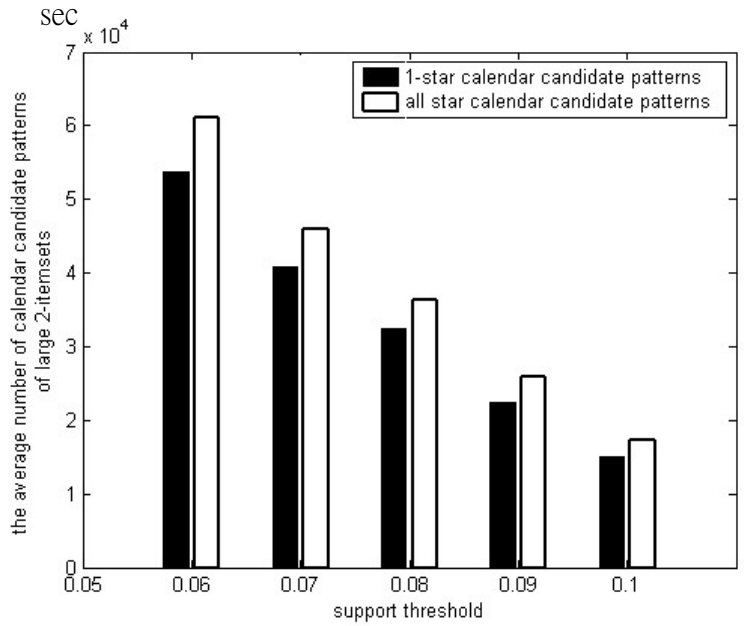
3. Experiment 3

我們使用 T10.I4.D400K， T10.I4.D600K，T10.I4.D800K，T10.I4.D1000K 四種不同大小的資料，以每個 basic time interval 1000 筆 transactions 來分割，針對不同的 support threshold 來比較在第一次掃瞄資料庫時，只單單保留 1-star calendar patterns 與保留全部的 calendar patterns 對於執行速度上面的影響。

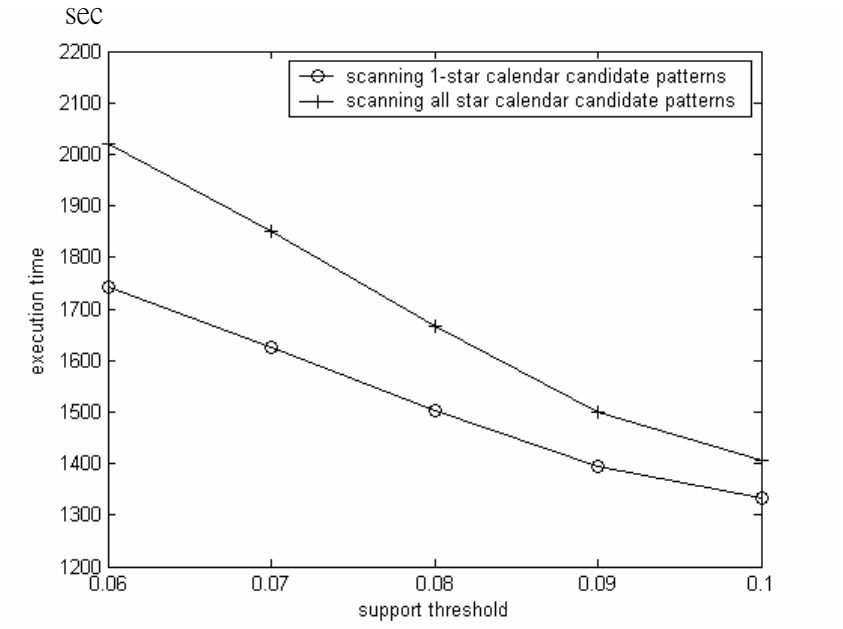
實驗結果

圖（IV-3.1）表示在不同的 support threshold 之下，四種不同大小的資料庫經過第一次掃瞄之後得到的 calendar patterns 平均數量。

我們可以清楚的知道，既然 all k-star calendar patterns 包含 1-star calendar patterns，1-star calendar patterns 的數量勢必小於 all k-star calendar patterns 的數量，以我們在此實驗所使用的 calendar schema 來說, 1-star calendar patterns 的數量大概是 all k-star calendar patterns 的 90%左右。所以使用我們的方法大概可以減少約 10%calendar patterns 數量，而這個比例將隨著所使用的 calendar schema 的複雜度提高而變小，例如有更多的時間單位，或者是同一個時間單位內的值域有更多成員。從圖（IV-3.2）中，我們可以看到利用只保留 1-star calendar patterns 的方法與保留 all k-star calendar patterns 之間執行時間的程度的差異，可以清楚的看到這樣的方法的確可以提高部分執行的效率。



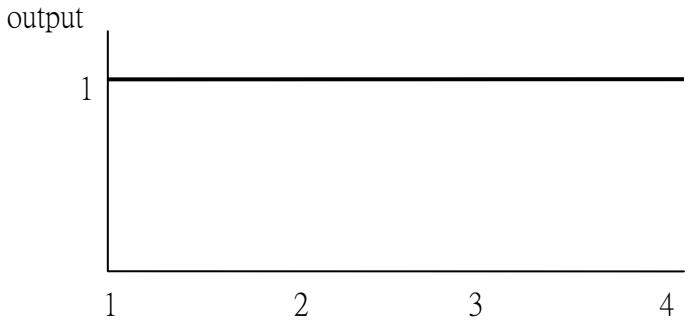
圖（IV-3.1）Experiment 3 第一次掃瞄資料庫時只保留 1-star calendar patterns 與保留全部的 k-star calendar patterns 的數量比較

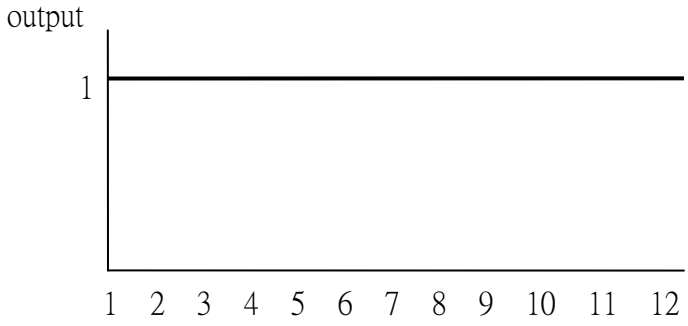


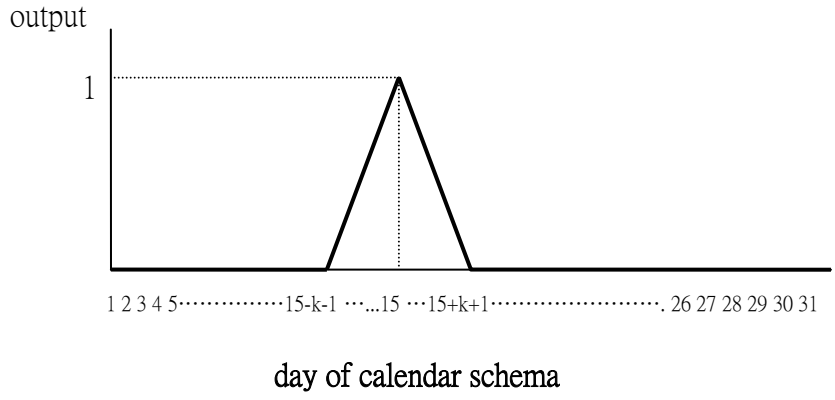
圖（IV-3.2）Experiment 3 第一次掃瞄資料庫時只保留 1-star calendar patterns 與保留全部的 k-star calendar patterns 的執行時間之比較

4. Experiment 4

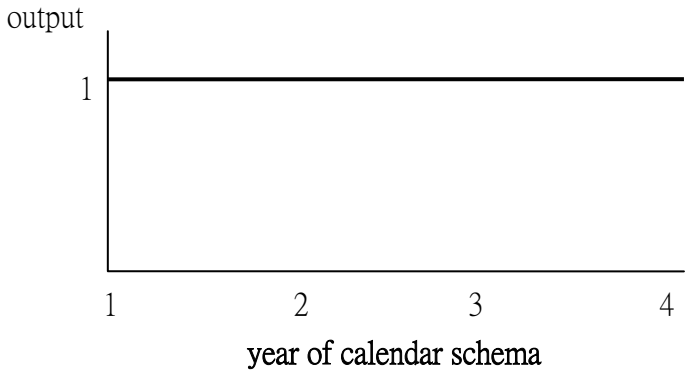
在此實驗中我們使用 T10.I4.D1M 的資料，以每個 basic time interval 1000 筆 transactions 來分割。進行在不同的 fuzzy match ratio threshold 下，產生的 large itemset 個數的變化與一般的 calendar pattern 方法做比較。在此實驗中我們使用的 fuzzy calendar pattern 為〝close to （＊，＊，15）〞 圖（IV-4.1）與圖（IV-4.2）分別定義兩組 fuzzy calendar patterns FC1 與 FC2，其中 FC1 的非同步性質為越靠近 15 日的日期，其對此週期的貢獻度就越大的特性，而 FC2 則是只要在 15 日附近 k 日的範圍內的所有日期，它們的貢獻度差異比較小。因此，在 FC1 中只要越靠近 15 日的日期中有出現 large itemsets，則此週期就越可能成立，而在 FC2 中，只要在範圍之內出現夠多的日期具有 large itemsets，則此週期就會成立。其中我們令靠近的範圍 k 為前後 3 天，而位移部分的加權值 α（式（17））設為 1-fuzzy match ratio threshold，因為一般來說，在做採掘的工作時會由較大的 threshold 開始，然後在找出的 patterns 無法滿足使用者時，才慢慢的調降。而將 α 設為 1-fuzzy match ratio threshold 時，當 fuzzy match ratio threshold 越大時，受正確週期時間點的影響越大，此時找到的 periodical association rules 較符合正確的週期時間，此時若找不到合適的 patterns 表示資料中符合正確週期時間的 patterns 沒那麼多，所以需要調降 fuzzy match ratio threshold，這時可以考慮提高非同步部分的比重，來採掘出合適的 patterns。因此，當 fuzzy match ratio threshold 越小時，則 α 值越大，受非同步部分的影響也就越大。support threshold 設為 0.07，而當我們將範圍 k 設為 0 時，即等於做一般的 calendar pattern mining。

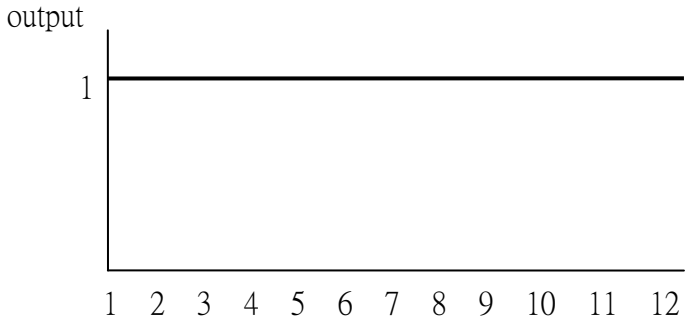


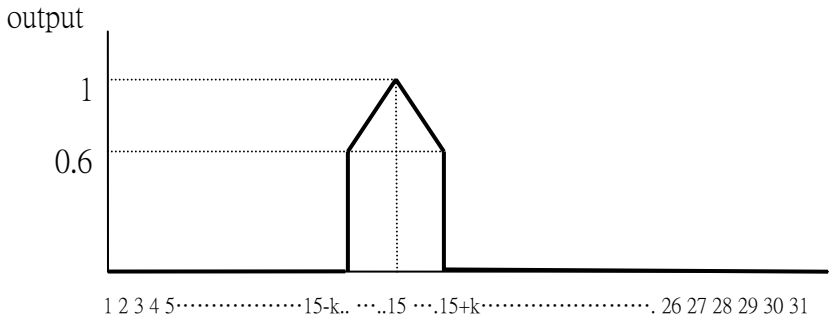




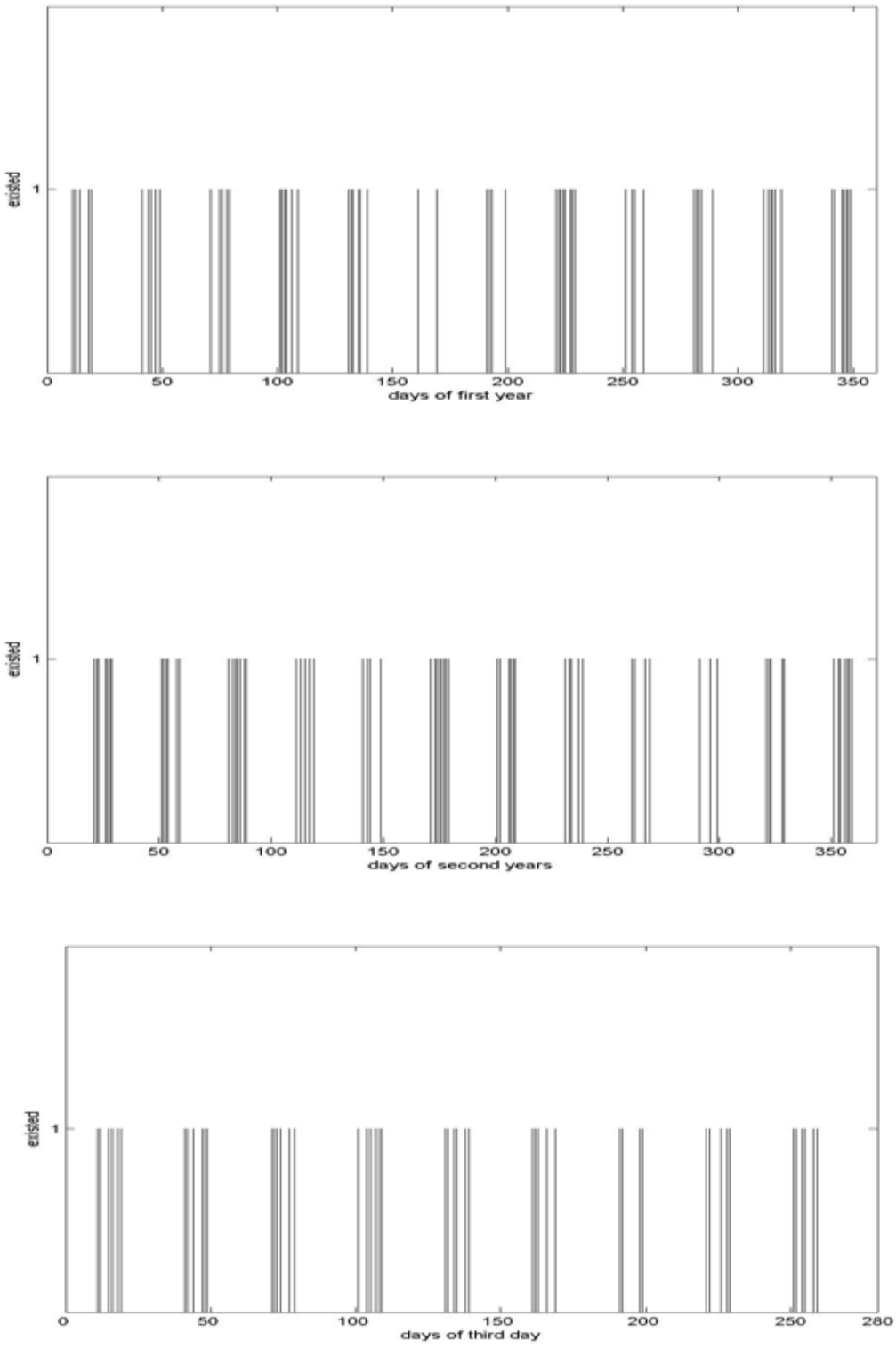
圖(IV-4.1)Experiment 4 close to (＊，＊，15)的 membership function FC1







圖（IV-4.2）Experiment 4 close to （＊，＊，15）的 membership function FC2



圖（IV-4.3）Experiment 4 在資料庫中（＊，＊，15）附近加入特定 patterns 的日期

在這個實驗中，為了說明加入 fuzzy 之後的效果，我們在所查詢的 calendar pattern 附近穿插加入特定的 large itemsets，然後比較在一般 calendar pattern 與 fuzzy calendar pattern 所找到的 large itemsets 是否有出現我們所加入的 large itemsets，來證明使用 fuzzy calendar pattern 的確可以找到出現非同步週期的 large itemsets。我們加入兩個長度 7 的 large itemsets，由其所衍生出的 large itemsets 的數量 Q 為

$$Q = 2\times(C_2^7 + C_3^7 + C_4^7 + C_5^7 + C_6^7 + C_7^7) = 2\times(21+35+35+21+7+1) = 240$$

我們以隨機的方式在每年每個月的 15 日前後 5 天的範圍內加入特定的 large itmesets，圖（IV-4.3）以布林函數圖形表示在三年的資料中加入特定 large itemsets 的日期分佈，其中 existed 為 1 代表有特定 large itemsets 在該日期中。

實驗結果

表(IV-4.1）為 fuzzy match ratio threshold（Fm）從 0.5 到 0.8 時，使用一般 calendar pattern 的方法（即 General 列），與使用 fuzzy calendar pattern 的方法（即 FC1 與 FC2 列）所得到的 large itemsets（total）與其中具有所加入特定 large itemsets(target)的數量。其中，FC1 與 FC2 列中的數量為範圍從前後兩天到前後四天所得到的 large itemsets 數量的平均值。其中 target 內的數量表示我們所加入的特定的 large itemsets 的數量。

從表（IV-4.1）中我們可以看到，在 fuzzy calendar pattern 之後，的確可以找出在資料庫中出現的具有非同步週期的 patterns，其中 FC1 與 FC2 又根據其 fuzzy calendar pattern 的特性而採掘出不同的 patterns。

	Fm=0.5		Fm=0.6		Fm=0.7		Fm=0.8	
	total	target	total	target	total	target	total	target
General	216	0	180	0	143	0	125	0
FC1	556	240	488	240	434	240	381	240
FC2	564	240	502	240	209	0	145	0

表(IV-4.1)Experiment 4 calendar pattern mining 的方法與 fuzzy calendar pattern mining 的方法所得到的 large itemsets 的數量之比較

我們從表(IV-4.1)中亦可以發現,當我們使用不同的membership function 時所得到的結果亦不相同，例如在相同的 fuzzy match ratio threshold Fm≧0.7 的情況之下，FC2 找不到我們所加入的large itemsets，但FC1 卻可以找到。因此，使用者應該知道什麼樣的非同步性質是他所想要找出的週期，以此來使用 membership function 來設計 fuzzy calendar pattern。

第五章、Conclusion

1.結論

在本論文中我們設計一個新的演算法來加速發掘資料庫之中的 calendar-based periodical association rules，我們利用掃瞄資料庫最多兩次，而且由 large 2-itemsets 以及它們的 frequent calendar patterns 一次產生全部 k-itemsets 以及它們 calendar patterns 的 candidates，以減少多次掃瞄資料庫所要花費的時間。另外，我們還利用 calendar pattern 的特性由只收集 1-star calendar patterns 的資訊來減少掃瞄資料庫中所要搜尋的 candidate calendar patterns，以提高整體的執行效率。在實驗中也的確顯示出我們所設計的演算法擁有較快速且穩定的執行速度。

另外，我們還設計了新的 fuzzy calendar pattern，可以用來採掘出現在資料庫中的非同步週期，並且提供一個演算法讓使用者可以查詢存在資料庫中的非同步週期規則---fuzzy periodical association rule。我們在這個部分中強調，使用 fuzzy calendar pattern 可以採掘出使用一般 calendar pattern 所無法採掘到之具有非同步週期的 association rules，而這些非同步週期的 association rules 在真實世界所蒐集到的資料中是可能存在且具有可利用的價值。

2.改進空間與未來工作

在發掘資料庫之中的 calendar-based periodical association rules 時，雖然我們的演算法比之前的方法可以得到較好的效率，但是由於我們是直接從 large 2-itemsets 開始產生，所以在第一次掃

瞄資料庫時,從transactions中對items做兩兩組和產生2-itemsets時，可能會產生過多多餘的 2-itemsets，而造成在第一次掃瞄資料庫時花費較多的時間。未來如果可以對此部分再加以改進，相信我們的演算法可以得到更佳的執行效率。

目前我們的方法是以查詢的方式，對特定的 fuzzy calendar pattern 做 association rules 的採掘。未來我們希望可以將此演算法擴展至不限制要某一個週期,而是能夠直接找出在資料庫中所有的非同步週期以及具有非同步週期的 association rules，並且還可以擁有不錯的執行效率之方法。如此一來，將會對採掘時間性資料庫中的非同步週期性關聯規則有更大的貢獻。

參考文獻

[1]R. Agrawal, T. Imielinski, A. N. Swami, “Mining association rules between sets of items in large data bases”, in : Proceedings of the 1993 ACM-SIGMOD International Conference on Management of Data, pages 207-216, Washington , DC, USA, May 1993.

[2]R. Agrawal, R. Srikant, “Fast algorithm for mining association rules in large databases”, in : Proceedings of the 20th International Conference on Very Large Data Bases, pages 487-499, Santiago, Chile, September 1994.

[3]R. Agrawal, J. C. Shafer, “Parallel mining of association rules”, IEEE Transactions on Knowledge and Data Engineering ,Vol. 8, NO. 6,pages 962-969 , December 1996.

[4] R. Agrawal, R. Srikant, “Mining sequential patterns”, in: Proceedings of the 11th International Conference on Data Engineering, pages 3-14, Taipei, Taiwan, March 1995.

[5]B. Özden, S. Ramaswamy, A. Sillberschatz, “Cyclic association rules”, in : Proceedings of the 14th International Conference on Data Engineering, pages 412-421, Orlando, Florida, USA , February 1998.

[6]S. Ramaswamy, S. Mahajan, A. Sillberschatz, “On the discovery of interesting patterns in association rules”, in : Proceedings of the 1998 International Conference on Very Large Data Base, pages 368-379, New York, USA, August 1998.

[7]Y. Li, P. Ning, X. S. Wang, S. Jajodia, “Discovering calendar-based temporal association rules”, Data and Knowledge Engineering , Vol. 44, NO. 2, pages 193-218, 2003.

[8]J. S. Park, M. -S. Chen, P. S. Yu, “Using a hash-based method with transaction trimming for mining association rules”, IEEE Transactions on Knowledge and Data Engineering, Vol. 9, NO. 5, pages 813-825, September 1997.

[9]C. -H. Lee, C. -R. Lin, M. -S. Chen, “Sliding-window filtering : an efficient algorithm for incremental mining”, in : Proceedings of the ACM 10th International Conference on Information and Knowledge Management(CIKM01), pages 263-270, Atlanta, Georgia, USA, October 2001.

[10] J. Han, J. Pei, Y. Yin, “Mining frequent patterns without candidate generation”, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, Vol.29, NO. 2, pages 1-12, Dallas, Texas, USA, May 2000.

[11]J. Han, G. Dong, Y. Yin, “Efficient mining of partial periodic patterns in time series database”, in: Proceedings of the 15th International Conference on Data Engineering, pages 106-115, Sydney, Australia, March 1999.

[12]J. M. Ale, G. H. Rossi, “An approach to discovering temporal association rules”, in: Proceedings of the 2000 ACM Symposium on Applied Computing, pages 294-300, Como, Italy, March 2000.

[13] J. Ayres , J. Flannick , J. Gehrke , T. Yiu, “Sequential Pattern mining using a bitmap representation”, in : Proceedings of the 8th ACM SIGKDD international conference on Knowledge Discovery and Data mining, pages 529-534, Edmonton, Alberta, Canada, July 2002.

[14]G. Zimbrão, J. M. de Souza, V. T. de Almeida, W. A. da Silva, “An algorithm to discover calendar-based temporal association rules with item's lifespan restriction”, in: Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - 2nd Workshop on Temporal Data Mining, Vol.8, pages 701–706, Edmonton, Alberta, Canada, July 2002.

[15]J. Yen, R. Langari, Fuzzy logic: intelligence, control, and information, Prentice-Hall, Inc. Upper Saddle River, New Jersey , 1999.

[16]R. Chandra, A. Segev, M. Stonebraker, “Implementing calendars and temporal rules in next generation database”, in : Proceeding of 10th International Conference on Data Engineering, pages 264-273, Houston, Texas, February 1994.

[17] W.-J. Lee, and S.-J. Lee, “Constructing Fuzzy Calendars for Mining Temporal Rules”, in: Proceedings of the 8th Conference on Artificial Intelligence and Applications, pages 147-152, December 2003.

[18]C. M. Kuok, A. Fu, M. H. Wong, “Mining fuzzy association rules”, in: Proceedings of the sixth international conference on Information and Knowledge Management, pages 209-215, Las Vegas, Nevada, United States, 1997.