# 線上混合式關聯規則採掘之資料方體挑選
# Data Cube Selection for Online Hybrid Associations Mining

林文揚
義守大學資訊管理學系
wylin@isu.edu.tw

張耀升
義守大學資訊工程研究所
m893335m@stdmail.isu.edu.tw

## 摘 要

資料方體實體化的選取問題是資料倉儲研究中，近來廣受探討的問題。過去在這方面的研究都是針對一般的 SQL 查詢或 OLAP 分析，鮮少有針對資料採掘查詢，探討如何進行資料方體實體化的研究。本研究主要在探討如何利用資料方體技術，針對多維度關聯規則之混合式關聯規則，進行所謂的線上關聯規則採掘。針對此問題，我們明確定義利用資料方體來進行線上關聯規則採掘的模式，及其查詢成本的估算方式，並實作及比較幾種啟發式挑選方法，在有限的儲存空間下挑選出最佳的資料方體的組合。

**關鍵字：**資料倉儲，資料採掘，線上分析處理，資料方體，啟發式演算法

## Abstract

Data cubes selection is one of the most important problems in data warehouse community. In the past few years, most researches for this problem mainly focused on data cubes selection for SQL or OLAP queries. To our knowledge, there is no work addressing the data cubes selection issue for association query. In this paper, we explore how to use materialized datacubes to facilitate online hybrid association rules mining. To this end, we define a cost model for datacubes selection problem and elaborate the cost estimation for association query. Besides, we implement and compare various heuristic algorithms to select suitable datacubes subject to the space constraint.

**Keywords**: data warehousing, data mining, OLAP, data cube, heuristic method.

## 壹、簡介

(一)研究動機

資料採掘(data mining)是近年來極為熱門的研究領域，其範疇是探討如何從大量累積的資料中發掘有用的知識(可表示為規則或模式)，作為知識工作者或決策人員處理問題的參考。眾多的資料採掘技術中，如何由大量的資料庫中尋找並快速的產生對使用者有用的關聯規則(association rules)，一直是廣受矚目的問題。

所謂的關聯式規則可表示成 $A \Rightarrow B$，$A$，$B$ 表示商品的集合，稱為項目集(itemset)。例如從超市的交易資料庫中，找尋顧客購買的商品之間的關係，如"大多數購買桌上型電腦的使用者都會購買噴墨印表機"，此二商品間的相關性可以表示成 Desktop $\Rightarrow$ Ink-jet。此規則的可信度(confidence)若為 80%，即表示在購買桌上型電腦的客戶當中，有 80%會一併購買噴墨印表機。因此，關聯規則 $A \Rightarrow B$ 的可信度可表示為

$$\text{conf}(A \Rightarrow B) = \text{sup}(A \cup B) / \text{sup}(A),$$

其中 sup($A$)稱為項目集 $A$ 的支持度(support)。為避免項目集的支持度不高，卻產生可信度極高的關聯關係，因此訂定項目集的最小支持度(*minsup*)及最小的可信度(*minconf*)。當構成關聯規則 $A \Rightarrow B$ 的項目集 $A \cup B$ 的支持度達到最小支持度，且其可信度亦超過最小可信度，吾人才認為此規則具有參考的價值。

一般的關聯規則的採掘過程可分為兩個部份：1)產生頻繁項目集(frequent itemset)；2)產生關聯規則，其中又以步驟 1 最耗費時間，因此，過去的研究多集中在發展有效率的頻繁項目集的產生方法。這些方法雖然都強調能快速找出頻繁項目集，但實際上仍需要耗費許久的時間。此外，由於使用者並不知如何設定合適的支持度與可信度，才能得到有用的規則，因此，經常須不斷地變更支持度與可信度，進行反覆的採掘處理。如此一來，整個的採掘過程往往曠日費時。因此，一個完善的線上關聯資料採掘系統，應能提供良好的作業環境，允許使用者於線上交談的模式下，快速地變更其觀察的角度與相關的參數設定，如支持度與可信度，並能迅速地完成處理，以回應使用者的查詢[16][20]。

有鑑於此，以 J. Han 為首的加拿大 Simon Fraser 大學的研究團隊[14][15][16][17][22]首先提出結合線上分析處理(On-line Analytical Processing, 簡稱 OLAP)與資料採掘系統的概念，發展出利用資料方體(datacube)中的資料進行各種資料採掘的系統，包括關聯式規則、分類(classification)、預測(prediction)與分群(clustering)，稱為 DBMiner。

由於一般資料倉儲系統的儲存空間有限，無法儲存所有可用以回答使用者查詢的資料方體。因此，如何在有限的儲存空間限制下，選取適當的子方體加以實體化，以縮短查詢的時間，是近來廣受探討的問題。過去在這方面的研究都是針對一般的 SQL 查詢或 OLAP 分析，鮮少有針對資料採掘查詢，探討如何進行資料方體實體化的研究。本篇論文的主要目的即在探討在有限的儲存空間下，如何根據使用者所下達的關聯式規則查詢，挑選適當的資料方體加以儲存，以減少回答查詢所需的時間。我們將此問題稱之為線上混合式關聯規則採掘之資料方體挑選問題。

(二)相關研究

目前在探討資料方體選取的問題上，大都是針對一般的 SQL 查詢或 OLAP 分析。這個問題已知是屬於非多項式時間完成問題(NP problem) [18]。目前解決此一問題主要有三種方法：第一種是窮舉法(exhaustive) [22][25]，第二種為利用一些最佳化方法，如遺傳演算法[1][23][29]，第三種則以啟發式(heuristics)為主[2][4][9][11][12][13][18][28]。

Harinarayan 等人首先定義針對處理 OLAP 的多維度分析時，如何選取實體化的子方體的問題[18]，並提出一貪婪演算法來解決此問題；Ezeife [11] 和 Gupta [12]則進一步將索引(index)的選取納入考量，但二者皆未考量維護這些實體化子方體的成本[13]；近來則有 Shim 等[25]及 Smith 等[26]探討動態的選取方法。陳耀輝等[4]則針對此問題提出一種兩階段的演算法，先減少資料倉儲的儲存空間，再改進 Harinarayan 等所提的貪婪法來挑選合適的子方體。我們亦針對此問題，提出一植基於遺傳演算法的挑選方法，能找出比一般貪婪法更佳的解[1][23]。

從資料庫的角度來看，資料方體實際上可視為是針對 OLAP 查詢的實體化視域。Gupta [12]於是將資料倉儲可回答的查詢種類擴大為所有可表示成 AND-OR 圖的查詢，探討在有限的儲存空間下，如何選取適當的實體化視域及其索引，以減少查詢處理及視域維護的成本。Theodorates [27]亦探討了此一問題，其研究進一步假設所選取的實體化視域必須能回答所有的查詢問題，或者使用者的查詢經由改寫(rewritting)後，仍然可用選取的實體化視域來回答，惟其研究不考慮儲存空間的限制。Yang 等人[28]也針對同一問題提出不同的處理模式，其考慮多個查詢間通常存在共同的部分表達式(common subexpressions)的現象[21]，但未考慮索引的選取問題。

另外，最近也有學者根據關聯規則採掘所需的頻繁項目集，將之定義為所謂的冰山方體(iceberg cube)[6][10]，用以儲存出現次數大於等於門檻值的項目集。此種資料方體的最大的優點是可以很快地產生出符合最小支持度的關聯規則，缺點是無法處理當使用者所定義的最小支持度比冰山方體的門檻值還小的情況。因此，我們認為在類似 DBMiner 等須結合 OLAP 與資料採掘分析的環境下，冰山方體的方法並不適用。

## 貳、理論架構與問題定義

(一)多維度關聯規則

資料倉儲中的資料皆以多維度的型態儲存，因此，由此種多維度資料所採掘出的關聯規則必須能反應不同維度間的關聯性，稱之為多維度關連規則。多維度的關聯規則可表示成：

$$A_1 = v_1 , A_2 = v_2 , \dots , A_m = v_m \Rightarrow B_1 = u_1 , B_2 = u_2 , \dots , B_m = u_n$$

其中 $A_i$，$1 \le i \le m$，及 $B_j$，$1 \le j \le n$，皆表示資料的屬性，而 $v_i$ 和 $u_j$ 則分別為屬性 $A_i$ 及 $B_j$ 的值。根據各屬性間的組合種類，多維度資料關聯規則大致上可分為三種類型[30]：

1、單一維度關聯規則(Single-dimensional association rules)：亦稱為維度內關聯規則(Intra-dimensional association rules)，當構成關聯規則的屬性只有一種稱之，即 $A_1 \equiv A_2 \equiv \dots \equiv A_m \equiv B_1 \equiv B_2 \equiv \dots \equiv B_n$。以表一為例，下列即為一維度內關聯規則。

Product = “CarryBags” ⇒ Product = “T-shirt”

2、維度間關聯規則(Inter-dimensional association rules)：當構成關聯規則的屬性皆不相同時稱之，即 $A_1 \ne A_2 \ne \dots \ne A_m \ne B_1 \ne B_2 = \dots \ne B_n$。以表一為例，下列即為一維度間關聯規則。

Supplier = “FILA”, Customer = “Peter” ⇒ Product = “T-shirt”

3、混合式關聯規則(Hybrid association rules)：此種關聯規則為本論文主要的研究對象，其結合了維度間關聯規則和維度內關聯關則，以表一為例，下列即為混合式關聯規則。

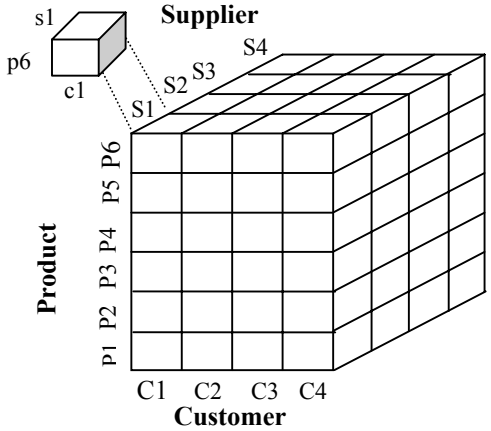
Supplier = “FILA”, Product = “CarryBags” ⇒ Product = “T-shirt”

(二)使用於關聯規則採掘之資料方體

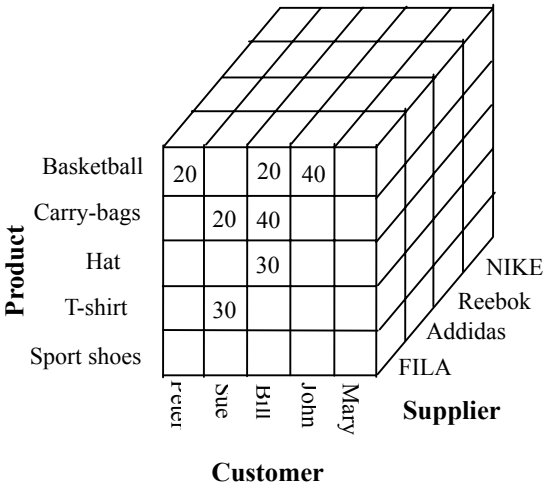
早期的資料倉儲皆以提供 OLAP 分析為主要的訴求。所謂的 OLAP 分析[8]係指一種能讓使用者在線上交談模式，以不同的主題

和角度觀察資料彙總的情形。為了能快速變換並展示使用者所觀察的資料的彙總結果，通常將所儲存的資料加以篩選、分類、彙總產生許多極小的資料方體，形成所謂的多維資料方體。例如圖一所示為一包含消費者，供應商及產品三個維度的銷售資料方體，其中每一儲存格(cell)儲存了賣給某一客戶由某供應商所提供的某一產品的銷售量。

由上述的例子可以發現，若將資料方體中所儲存的銷售量改為相對組合的統計次數，則資料方體所儲存的資訊即可成為線上關聯規則採掘時，所需要的各種項目組合的支持度。圖二所示即為根據表一的關聯式資料庫所產生的資料方體。




圖一、OLAP 資料方體的例子。




圖二、用於線上關聯規則採掘的資料方體。

(三)問題的定義

已知一個資料方體格構 $L$，其中包含 $n$ 個資料方體 $D = \{d_1, d_2, \ldots, d_n\}$，使用者所查詢的關聯規則問題集合 $Q = \{q_1, q_2, \ldots, q_m\}$，每個問題查詢的頻率為 $F = \{f_{q_1}, f_{q_2}, \ldots, f_{q_m}\}$，空間限制 $S$，上述問題可表示成 $\theta = \{L, D, Q, F, S\}$，如何在空間限制下，從 $D$ 中找到一組最佳的資料方體集合 $M$ 進行實體化，使得回答所有的查詢所需的時間為最低，即 $\min \sum_{i=1}^{m} f_{q_i} E(q_i, M)$，其中 $E(q_i, M)$ 表示在實體化的集合 $M$ 中回答每個查詢問題 $q_i$ 所花費的時間。

## 參、混合式關聯查詢的成本估算

在本章中，我們針對前一章節中的成本估算函數 $E(q_i, M)$，說明其計算方式。欲瞭解成本估算函數的計算方式，我們必須先明白：(1)關聯規則查詢與資料方體的對應關係，即對任一關聯規則查詢，可用以回答的資料方體為何；(2)如何利用對應的資料方體回答我們定義每一使用者所下達的多維度關聯規則查詢。

(一) OLAP 資料方體與關聯規則的對應關係

為了方便討論，我們先定義下列的資料採掘查詢語言來表達每一種不同的多維度關聯規則。

**mine hybrid associations**
**trans** *dim.attr* [, *dim.attr*]}
**item** *dim.attr*[, *dim.attr*]
**from** *schema*
**with** *minsup* = s%, *minconf* = *c*%

**定理一**：給定一個關聯規則查詢 $q$，則任何可用以回答某一關聯規則查詢 $q$ 的資料方體 $D$ 的條件為 $Attr(q) \subseteq Attr(D)$，即構成關聯查詢 $q$ 的維度屬性集必須都在構成資料方體 $D$ 的維度屬性集合中。

**例一**：已有實體化之 OLAP 資料方體 $D_1$= (product, supplier, customer)、$D_2$ = (product, supplier, customer, city)及 $D_3$ = (product)，現有一關聯規則查詢如下:

**mine hybrid associations**
**trans** 產品.*product*
**item** 供應商.*supplier*，顧客.*customer*
**from** *S*
**with** *minsup = s %, minconf = c%*

則 $Attr(q)$ = { product, supplier, customer}。根據定理一，$Attr(q) \subseteq Attr(D_1)$、$Attr(q) \subseteq Attr(D_2)$ 但 $Attr(q) \not\subset Attr(D_3)$。故可以回答關聯規則查詢 $q$ 的 OLAP 資料方體有 $D_1$ 和 $D_2$。

(二)以OLAP資料方體採掘混合式關聯規則

假設查詢的資料屬性集為 *item*，交易屬性集為 *trans*，且 $|trans| \geq 1$，$|item| \geq 2$，而用以回答此查詢的資料方體為 Cube[$A_1$, $A_2$,…, $A_r$]，且 $trans \cup item \subseteq \{A_1, A_2, \ldots, A_r\}$。以下是

表一、包含四個維度之關聯資料表。

| 城市<br>city | 供應商<br>supplier | 產品<br>product | 顧客<br>customer | 次數<br>count |
|---|---|---|---|---|
| Chicago | NIKE | Hat | John | 10 |
| Dallas | NIKE | Hat | John | 10 |
| New York | NIKE | Hat | John | 20 |
| Chicago | NIKE | Hat | Mary | 20 |
| New York | NIKE | Sport shoes | John | 20 |
| Dallas | NIKE | Sport shoes | John | 10 |
| Chicago | NIKE | Sport shoes | John | 10 |
| Chicago | Reebok | Basketball | Peter | 10 |
| Seattle | Reebok | Basketball | Peter | 10 |
| Chicago | Reebok | Carry Bags | Peter | 80 |
| Seattle | Reebok | Carry Bags | Peter | 30 |
| Dallas | Reebok | Carry Bags | Peter | 30 |
| New York | Reebok | Carry Bags | Peter | 20 |
| Seattle | Reebok | Hat | Peter | 10 |
| Chicago | Reebok | Basketball | Bill | 10 |
| New York | Reebok | Carry Bags | Bill | 10 |
| New York | Reebok | Hat | Bill | 10 |
| Seattle | Reebok | T-shirt | Sue | 10 |
| Dallas | Reebok | Hat | Sue | 10 |
| Seattle | Addidas | Carry Bags | John | 20 |
| Dallas | Addidas | Carry Bags | John | 10 |
| New York | Addidas | Carry Bags | John | 40 |
| Dallas | Addidas | Hat | Peter | 30 |
| Seattle | Addidas | Hat | John | 30 |
| Dallas | Addidas | Hat | John | 20 |
| New York | Addidas | Hat | John | 30 |
| Seattle | Addidas | Sport shoes | Bill | 20 |
| New York | Addidas | Hat | Peter | 10 |
| Dallas | Addidas | Carry-bags | Peter | 10 |
| New York | Addidas | Carry-bags | Peter | 30 |
| Chicago | Addidas | Carry Bags | Peter | 10 |
| Dallas | FILA | Carry Bags | Sue | 20 |
| Dallas | FILA | Hat | Bill | 10 |
| Chicago | FILA | Hat | Bill | 20 |
| Seattle | FILA | T-shirt | Sue | 20 |
| Chicago | FILA | T-shirt | Sue | 10 |
| Chicago | FILA | Basketball | John | 30 |
| New York | FILA | Basketball | John | 10 |
| New York | FILA | Basketball | Peter | 20 |
| Dallas | FILA | Carry-bags | Bill | 20 |
| Chicago | FILA | Carry-bags | Bill | 20 |
| New York | FILA | Basketball | Bill | 20 |

其產生頻繁項目集的演算法。

**演算法一**、產生混合式之頻繁項目集

輸入：

- 查詢交易屬性集 *trans*，及資料屬性集為 *item*
- 資料方體：Cube[$A_1, A_2, \ldots, A_r$] , $r \geq 3$
- 最小支持度門檻值(minimum support threshold)：*min_sup*

輸出：混合式頻繁項目集 *F*

步驟：

1. $k = 1$；$F = \varnothing$;
2. $C_1 = \bigcup_{A_i \in item} Domain(A_i)$;
3. $F_1 = gen\_frequent(1, C_1)$;
4. **while** $F_k \neq \varnothing$ **do**
5. $k = k + 1$;
6. $C_k = gen\_candidate(k, F_{k-1})$;
7. $F_k = gen\_frequent(k, C_k)$;
8. $F = F \cup F_k$;

9. **endwhile**

**function** gen_frequent($k$,$C_k$)
輸入：長度 $k$ 之候選項目集 $C_k$
輸出：長度 $k$ 之頻繁項目集 $F_k$
步驟：

1. $F_k = \varnothing$ ;
2. **for** each candidate $k$-itemsets, $I$ = $\{i_1,i_2,\ldots,i_k\} \in C_k$ **do**
3. $I.count = 0$; //出現之次數
4. **for** each $t \in Domain(trans)$ **do**
5. $All$ = TRUE;
6. **for** each item $i \in I$ **do**
7. accumulate the count saved in each cell partially indexed by ($t$, $i$);
8. **if** (count = 0) **then**
$All$ = FALSE;
9. **break**;
10. **endif**
11. **endfor**
//同一交易 $t$ 中，所有項目皆出現在此項目集，則次數累加 1
12. **if** ($All$ = TRUE)$I.count$ ++;
13. **endfor**
14. $I.supp = I.count$ / $totalcount$;
15. **if** ($I.supp > min\_sup$ ) $F_k = F_k \cup I$;
16. **endfor**
17. **return** $F_k$;

**function** gen_candidate($k$, $F_{k-1}$)
輸入：長度 $k$-1 之頻繁項目集 $F_{k-1}$
輸出：長度 $k$ 之頻繁項目集 $C_k$
步驟：

1. $C_k = \varnothing$ ;
2. **for** each itemset $I_1 \in F_{k-1}$ **do**
3. **for** each itemset $I_2 \in F_{k-1}$ **do**
4. **if** (first $k$–2 items in $I_1$ and $I_2$ are same but the last item are different) **then**
5. $c = I_1 \bowtie I_2$ ;
6. **if** $\exists$($k$–1)-subset $s$ of $c$, $s \notin F_{k-1}$ **then**
7. delect $c$;
8. **else**
9. add $c$ to $C_k$;
10. **endif**
11. **endif**
12. **return** $C_k$

(三) 查詢成本估算

眾所週知，處理關聯規則的時間絕大部分都花在產生頻繁項目集，因此，我們僅估計此部份所花費的時間。如前所述，我們所提出產生頻繁項目集所用的方法為類似 Apriori 演算法[6]，因此，所需花費時間可分為兩大部分，一為掃描資料庫的時間($t_{scan}$)，另一則為計算各項目集出現次數的時間($t_{count}$)。

假設所使用的資料方體為 $D$，回答的查詢為 $q$，首先我們分析掃瞄資料方體的時間。根據 Apriori 式的方法，掃瞄資料庫的次數等於產生之最大的頻繁項目集的長度。因此，若已知最大的頻繁項目集長度為 $k_{max}$，則 $t_{scan} = |D| \times k_{max}$。

接下來討論計算項目集出現次數的間。根據演算法二，此部份的計算方式可視為在進行第 $k$ 次的資料庫掃瞄，將每一個可能的 $k$-項目集與每一筆交易資料進行比對，若出現於此交易中，則將此 $k$-項目集的次數加 1。因此，所需的比對次數為 $\binom{d_i}{k} \times d_t$。其中，$d_i = |Domain(item)|$，$d_t = |Domain(trans)|$，$\binom{d_i}{k}$ 表組合函數。

另外，比對每一個 $k$-項目集與交易資料的方式可以下列說明。考慮一個三維 ABC 的資料方體，假設所欲回答關聯規則查詢的交易維度為 A，$Domain$(A) = {$A_1$, $A_2$, $A_3$, $A_4$}，而項目維度為 B，$Domain$(B) = {$B_1$, $B_2$}。現假設要比對的項目集為 {$B_1$, $B_2$}，而交易資料為以 $A_1$ 所組成的資料。要確定{$B_1$, $B_2$}是否出現在此資料中，我們必須分別比對 $B_1$ 及 $B_2$，以 $B_1$ 為例，須需掃瞄($A_1$, $B_1$, $C_1$) 、($A_1$, $B_1$, $C_2$) 、($A_1$, $B_1$, $C_3$) 及 ($A_1$, $B_1$, $C_4$)，計算其加總的值，由此可知，每比對一個 $k$-項目集與一筆交易資料所需的計算次數為 $k * d_r$，此處 $d_r$ 表非交易和非項目維度的屬性值之數目，即 $d_r = |Domain(Attr(D) - Attr(q))|$。

綜合以上的分析，可得到

$$t_{count} = \sum_{j=1}^{k_{max}} \binom{d_i}{j} \times d_t \times j \times d_r 。$$

上述式子中 $k_{max}$ 的值會隨著 $min\_sup$ 的值而改變，但 $min\_sup$ 的值是由使用者下達，事先無法預知，因此，我們必須以其他方式來估算合理的 $k_{max}$ 值。一個合理的估算是每個交易的平均長度，計算方式如下：$D_{trans}$ 為以其交易屬性集 group-by 所對應的資料方體，則$|D_{trans}|$為實際交易筆數。我們分別估算每筆交易包含項目屬性集 $item$ 中某一屬性 $A_i$ 所構成的項目集數目，即$|D_{trans} \cup A_i|/|D_{trans}|$；由此推得各項目屬性之項目集數目，最後將其累加，即為我們估算平每筆交易所包含之項目長度

$$k_{max} = \sum_{A_i \in item} \frac{|D_{trans} \cup A_i|}{|D_{trans}|} 。$$

**例三**：承例二，交易屬性與各項目屬性所對應的資料方體各為 $D_{(Customer,\ Supplier)}$和 $D_{(Customer,\ Product)}$，如表二和表三，而交易屬性對應的資料方體 $D_{(Customer)}$ 如表四，故

$$k_{max} = \frac{|D_{(Customer,Product)}| + |D_{(Customer,Supplier)}|}{|D_{Customer}|} = 6.$$

表四是由表一以顧客為交易代號所得到的交

易資料表。我們加總各筆交易的項目集長度再予以平均，可求得(8+2+6+6+5)/5 = 6，發現與前面所估計的 $k$ 相同。

最後，假設掃描一筆資料與出現次數加 1 的時間比為$\alpha$，則以資料方體 $D$ 回答查詢 $q$ 所需的時間可表示為

$$E(q, D) = t_{count} + \alpha \times t_{scan} = \sum_{j=1}^{k_{\max}} \binom{d_i}{j} \times d_t \times j \times d_r + \alpha \times |D| \times k_{max}$$

表二、由表一產生顧客及產品屬性之資料方體表

| 顧客 Customer | 產品 Product | 次數 Count |
|---|---|---|
| Peter | Basketball | 40 |
| Peter | Carry Bags | 240 |
| Peter | Hat | 20 |
| John | Hat | 120 |
| John | Sport shoes | 40 |
| John | Carry Bags | 70 |
| John | Basketball | 40 |
| Bill | Basketball | 30 |
| Bill | Carry Bags | 50 |
| Bill | Hat | 40 |
| Bill | Sport shoes | 20 |
| Sue | T-shirt | 40 |
| Sue | Hat | 10 |
| Sue | Carry Bags | 20 |
| Mary | Hat | 20 |

表三、由表一產生顧客及供應商屬性之資料方體表

| 顧客 Customer | 供應商 Supplier | 次數 Count |
|---|---|---|
| Peter | Reebok | 190 |
| Peter | Addidas | 90 |
| Peter | FILA | 20 |
| John | NIKE | 80 |
| John | Addidas | 150 |
| John | FILA | 40 |
| Bill | Reebok | 30 |
| Bill | Addidas | 20 |
| Bill | FILA | 90 |
| Sue | Reebok | 20 |
| Sue | FILA | 50 |
| Mary | NIKE | 20 |

## 肆、實驗分析

在本章中，我們選擇了目前最具代表性的三種啟發式演算法:包括 Harinarayan 等人[18]所提的出的正向貪婪挑選法(Forward greedy selection, FGS) 、我們[2]提出的反向貪婪挑選法(Backward greedy selection, BGS) 、以及 Shukla 等人[24]所提出的體積挑選法(Pick by size，PBS)，進行實驗分析比較，以了解這些方法運用於針對混合式線上關聯規則查詢的資料方體挑選問題的適用狀況。有關這三個方法的細節，請讀者參考文獻[2][18][24]。

表四、由表一產生顧客屬性之資料方體

| 顧客 Customer | 次數 Count |
|---|---|
| Peter | 300 |
| John | 270 |
| Bill | 140 |
| Sue | 70 |
| Mary | 20 |

為了解這三個挑選法實際的效能，我們由 MicroSoft SQL 7.0 和其含的 FoodMarket 資料庫以顧客(Customer, c)、產品(Product, p)、及日期(Date, d)等維度中各屬性，如表六，組成測試用的資料方格的來資料進行一連串的實驗；而 Product 有 3 種可能，Date 有 3 種可能，其中 Customer 維度內的屬性因無階層關係，所以多了一個 ec 的可能組合，故 Customer 有 4 種可能，最後所有屬性的組合數為 35，如表六。至於各個子方體的使用頻率的組合，我們考慮了二種模式：第一種是所有的子方體的使用頻率皆相同，且都大於零；第二種是由隨機產生介於 0~1 間的亂數所組成的組合。另外我們將儲存空間的限制，分別設為子方體總合的 5%、10%、20%、30%、40%、50%、60%、70%、及 80%。此外，若發生當某一查詢所使用的子方體皆未實體化的情況，我們則將其視為需回到資料倉儲的基底關聯進行查詢，在此我們將其時間設為資料方體中最大之子方體查詢時間的 3 倍，而α值為 100。

表五、customer，product，date 維度屬性表。

| | 維度 | | |
|---|---|---|---|
| | Product | Customer | Date |
| 屬性 | product | education | day |
| | type | city | month |
| | - | - | - |

表六、由表五之各屬性所組成之資料方體

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| p-- | 135 | p-d | 7545 | -cm | 898 | ted | 8542 |
| t-- | 45 | p-m | 3124 | pce- | 7412 | tem | 1579 |
| -e- | 5 | te- | 225 | tce- | 1543 | tcd | 9784 |
| -c- | 78 | tc- | 3208 | -ced | 1741 | tcm | 4103 |
| -ec- | 300 | t-d | 7541 | -cem | 1114 | pecd | 5699 |
| --d | 150 | t-m | 540 | ped | 4781 | pecm | 6741 |
| --m | 12 | -ed | 1551 | pem | 1974 | tecd | 9994 |
| pe- | 350 | -em | 60 | pcd | 4531 | tecm | 1023 |
| pc- | 9568 | -cd | 4464 | pcm | 4563 | | |

(一)查詢成本比較

我們首先比較此三個方法所求得的實體化資料方體集的優劣。比較的指標式使用其求得的實體化資料方體集回答所有的查詢所需要的時間，稱為查詢成本。由圖三我們可以很清楚的發現，不管頻率或空間限制為何，正向與反向貪婪法查詢成本上都優於體積挑選法，且當空間限制在20%時，可以找到一組較佳的近似解。而體積挑選法則在空間限制 70%時，才找到最佳近似解。

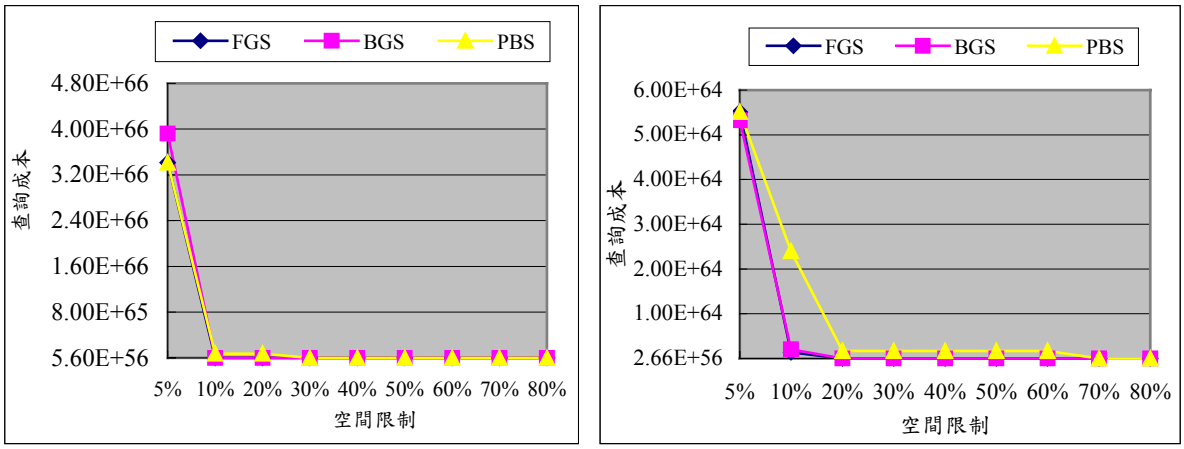
(二)挑選時間比較

接著我們比較此三個演算法的執行效率。由圖四可以得知，正向與反向貪婪挑選法的速度較慢，而依空間限制的不同，其執行時間各為升冪及降冪的曲線。由於正向貪婪法是由空集合開始，一次挑選一個資料方體，直到已挑選的資料方體大小總和超過空間限制為止，因此當儲存空間愈大，則需執行的挑選次數愈多；相反地，反向貪婪挑選法是假設所有的資料方體皆已挑選，一次刪除一個資料方體，直到剩下的資料方體總和不超過空間限制為止，故儲存空間愈大，則需執行挑選的次數就愈少；最快為體積挑選法，其挑選的依據純粹是根據資料方體的大小，故其速度不受空間限制的影響。

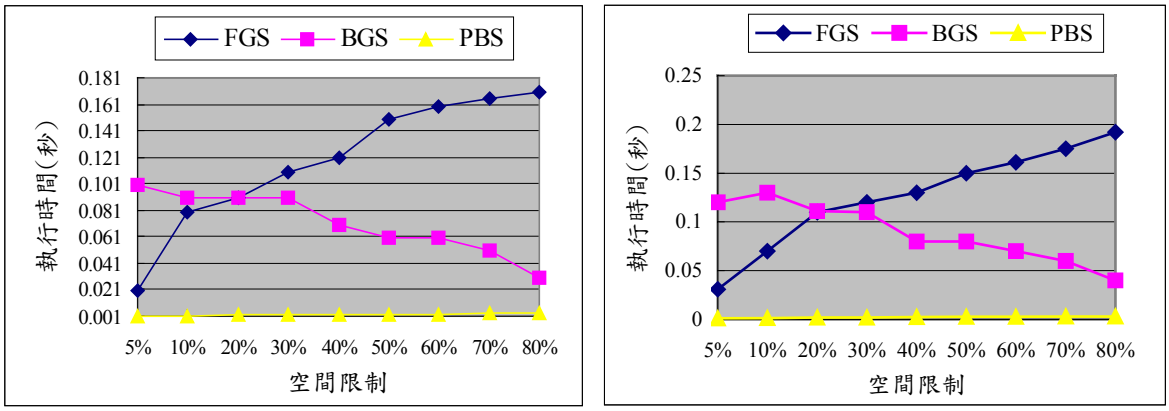
## 伍、結論

資料倉儲與資料採掘的結合已成為目前知識發掘與管理研究的主流趨勢。雖然在這方面的研究目前已有不少的研究成果，但有關如何針對資料採掘等查詢，建立適當的資料方體與挑選方法仍在啟蒙階段。在這篇論文中，我們針對利用 OLAP 資料方體進行線上關聯規則採掘的模式，明確定義利用資料方體線上關聯規則採掘的成本估算方式，以估算每個關聯規則查詢成本，作為資料方體挑選的依據。此外我們也說明如何運用啟發式挑選方法，在空間限制條件下，依據我們定義的成本估算模式，進行資料方體的挑選。

未來我們將考慮資料方體的維護成本，並探討同時存在 OLAP 查詢與關聯規則查詢時的資料方體構建問題。




圖三、混合式關聯規則之資料方體挑選，當(a)各子方體的使用頻率皆相同時，及(b)各子方體的使用頻率為亂數時，各挑選法在查詢成本上的比較




圖四、多維度關聯規則之資料方體挑選，當(a)各子方體的使用頻率皆相同時，及(b)各子方體的使用頻率為亂數時，挑選時間的比較

## 誌謝

本論文為國科會補助之研究計畫成果 NSC 90-2213-E-214-040。

## 參考文獻

[1] 林文揚與郭義中，“遺傳演算法於資料倉儲中構建資料方體之應用”，八十八年全國計算機會議論文集，第一輯，頁 241-248，1999。

[2] 林文揚與郭義中，“應用於線上分析之資料方體的雙向貪婪挑選法”，第五屆資訊管理實務研討會，高雄，臺灣，2000。

[3] 林文揚與張耀升，“線上關聯規則資料方體之挑選方法”，中華民國人工智慧學會，台中，臺灣，2002。

[4] 陳耀輝、劉宇昌與劉佳灝，”在資料倉儲中選擇實體化視域之研究”，八十六年全國計算機會議論文集，第一輯，頁 72-77，1997。

[5] R. Agrawal and R. Srikant, “Fast algorithms for mining associations rules,” in *Proc. VLDB*, pp. 487-499, 1994.

[6] K.S. Beyer and R. Ramakrishnan, “Bottom-up computation of sparse and iceberg cubes,” in *Proc. ACM SIGMOD*, pp. 359-370, 1999.

[7] E. Baralis, S. Paraboschi, and E. Teniente, “Materialized view selection in a multi-dimensional database,” in *Proc. VLDB*, pp. 156-165.

[8] S. Chaudhuri and U. Dayal, “An overview of data warehouse and OLAP technology,” *ACM SIGMOD Record*, Vol. 26, pp. 65-74, 1997.

[9] P. M. Deshpande, K. Ramasamy, A. Shukla, and J. F. Naughton, “Caching multi-dimensional queries using chunks,” in *Proc. ACM SIGMOD*, pp. 371-382, 1999.

[10] M. Fang, et al, “Computing iceberg queries efficiently,” in *Proc. VLDB*, pp. 299-310, 1998.

[11] C.I. Ezeife, “A uniform approach for selecting views and indexes in a data warehouse,” in *Proc. IDEAS’97*, pp. 151-160, 1997.

[12] H. Gupta, “Selection of views to materialize in a data warehouse,” in *Proc. ICDT’97*, pp. 98-112, 1997.

[13] H. Gupta and I.S. Mumick, “Selection of views to materialize under a maintenance cost constraint,” in *Proc. ICDT’99*, pp. 453-470, 1999.

[14] J. Han, “OLAP mining: An integration of OLAP with data mining,” in *Proc. IFIP Conf. Data Semantics*, pp. 1-11, 1997.

[15] J. Han, “Toward on-line analytical mining in large databases,” *ACM SIGMOD Record*, Vol. 27, No. 1, pp. 97-107, 1998.

[16] J. Han, L.V.S. Lakshmanan, and R.T. Ng, “Constraint-based, multi-dimensional data mining,” *IEEE Computer*, Vol. 32, No. 8, pp. 46-50, 1999.

[17] J. Han, S. Chee, and J. Chiang, “Issues for on-line analytical mining of data warehouses,” in *Proc. DMKD’98*, pp. 21-25, 1998.

[18] V. Harinarayan, A. Rajaraman, and J.D. Ullman, “Implementing data cubes efficiently,” in *ACM SIGMOD‘96*, pp. 205-216, 1996.

[19] C. Hidber, “Online association rule mining,” in *Proc. ACM SIGMOD*, pp. 145-156, 1999.

[20] W.H. Inmon and C. Kelley, *Rdb/VMS: Developing the Data Warehouse*, QED Publishing Group, Boston, Massachussetts, 1993.

[21] M. Jarke, “Common subexpression isolation in multiple query optimization,” *Query Processing in Database Systems*, pp. 191-205, 1984.

[22] M. Kamber, J. Han, and J.Y. Chiang, “Metarule-guided mining of multidimensional association rules using data cubes,” in *Proc. ACM KDD*, pp. 207-210, 1997.

[23] W.Y. Lin and I.C. Kuo, "OLAP data cubes configuration with genetic algorithms," in *Proc. IEEE SMC’00*, pp. 1984–1989, 2000.

[24] A. Shukla, P.M. Deshande, and J.F. Naughtion, “Materialized view selection for multi-dimensional datasets,” in *Proc 24th VLDB*, pp. 156-165, 1998.

[25] J. Shim, P. Scheuermann, and R. Vingralek, “Dynamic caching of query results for decision support systems” in *Proc. ICSSDM*, pp. 254-263, 1999.

[26] J.R. Smith, et al, “Dynamic assembly of views in data cubes,” in *Proc. ACM PODS*, pp. 274-283, 1998.

[27] D. Theodoratos and T. Sellis, “Data warehouse configuration,” in *Proc. 23rd VLDB*, pp.126-135, 1997.

[28] J. Yang, K. Karlapalem, and Q. Li, “Algorithm for materialized view design in data warehousing environment,” in *Proc. 23rd VLDB*, pp. 136-145, 1997.

[29] C. Zhang and J. Yang, “Materialized view evolution support in data warehouse environment,” in *Proc. ICDSAA*, pp. 247-254, 1999.

[30] H. Zhu, *On-line Analytical Mining of Association Rules*, Master Thesis, SIMON FRASER University, 1998.