在資料倉儲中選擇實體化視域之研究
Selecting Materialized Views in Data Warehouses

陳耀輝
國立嘉義師範學院數理教育系
ychen@sun11.ncytc.edu.tw

劉宇昌 、 劉佳灝
國立屏東科技大學資訊管理研究所
m8556005@mail.npust.edu.tw

摘 要

資料倉儲是一種將資料聚集成資訊的方法，而資料方格 (data cube) 則是一種描述聚集性資料的架構。使用者的查詢往往僅用到資料方格內的部份視域，而受限於儲存空間，這些視域未必皆能被實體化，故本研究提出一種兩階段演算法來選擇被實體化的視域。這個演算法的第一階段以「減少儲存空間」爲目標，而第二階段則是在有限的儲存空間限制下，以增加實體化視域的方式來「降低整體查詢成本」。模擬程式的實驗結果顯示，本研究提出的兩階段演算法不僅可應用在較多的實例而且可顯著降低整體的查詢成本。

關鍵詞：資料倉儲、資料方格、實體化視域、啓發式演算法。

Abstract

A data cube is a framework of data warehouse. Users' queries only access parts of the views in the data cube. Because of the limitation of the storage space, not all views accessed by queries can be materialized. Therefore, we propose a two-phase algorithm to select views to materialize. In the first phase, we use dependency relations to decrease the number of materialized views in order to save the storage space. In the second phase, we add materialized views, under the storage space limitation, to reduce total query processing costs. The simulation results indicate that our approach can select materialized views in more cases and can reduce the total query processing costs.

Keywords: Data warehouse, data cube, materialized view, heuristic algorithm.

1 緒論

爲了要提供線上分析處理的功能，並且讓資料更容易地被轉變成有意義的資訊，就產生了資料倉儲 (data warehouse) 的概念[SSU96]。資料倉儲的資料模組是以 Kimball 所提出的多維資料模組 (dimensional data model) 來建構 [KS95]。多維資料模組亦稱爲星狀連結綱目 (star join schema)；它是以事實表格 (fact table) 與維度表格 (dimensional table) 所共同組合而成。除了原始定義之外，多維資料模組另有雪花 (snowflakes) 與銀河 (galaxies) 兩種變形 [McG96]。多維資料模組與關聯式資料模組最顯著的差異在於模組對象的改變。多維資料模組是針對將「企業流程」模組化而設計，所以非常適合資料倉儲的應用。

將資料聚集化 (aggregation) 是一種讓資料變成資訊的方法。所謂的聚集化是指將資料經過加總、平均等運算程序後變成資訊。資料方格 (data cube) 是一種將資料以二維、三維甚至多維的方式展示的架構 [HRU96]。在資料方格中，每一個儲存格 (cell) 表示使用者有興趣資料的測量值。圖 1-1 的資料方格是由兩個顧客、三個零件以及三個供應商構成的一個三維資料方格，每個儲存格代表交易的金額 (SP)。

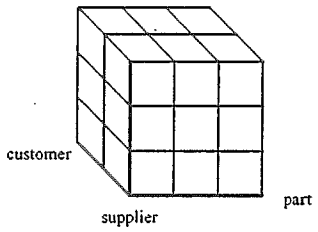

圖 1-1 顧客-零件-供應商之資料方格圖

當使用者查詢每位供應商所供應的各項零件的銷售情形時，必須對每一組供應商-零件組合的「所有的顧客」進行加總，亦即 $(c_1, p_1, s_1) + (c_2, p_1, s_1)$ 爲一組結果、$(c_1, p_2, s_1) + (c_2, p_2, s_1)$ 爲一組結果、、、餘此類推。Gray等人建議在每一個維度的定義域內增加一個"ALL"值 [GBLP95]，以對每一組供應商-零件組合的「所有」顧客進行加總，則查詢可改寫成 (ALL, p, s) 的形式。將前述的查詢寫成標準查詢語言(SQL)：

```
SELECT Part, Supplier, SUM(SP) AS TotalSales
FROM R
GROUP BY Part, Supplier
```

此處 R 是指原始的關連。

以圖 1-1 的資料方格爲例，將會產生八個不同組合的 GROUP BY 子句，所以便會產生八個不同的視域 (view)。圖 1-2 即爲此八個視域的架構，其中視域 cps 表示 GROUP BY 子句包含顧客 (c)、零件 (p) 以及供應商 (s)。 GROUP BY 子句所未包含的維度就沒有在視域名稱中列出。圖 1-2 的視域間存在一種相依性 (dependence)。查詢(c, p, ALL)是表示對供應商進行加總，即 $(c, p, s_1) + (c, p, s_2) + (c, p, s_3)$，而這些資料均可自視域 cps 取得。由此可知，在視域 cp 上可執行的查詢亦可在視域 cps 上取得相同的結果。由

此一觀點看來，資料倉儲是由實體化視域 (materialized views) 所組合而成的 [Gup97, QGMW96]。

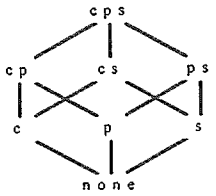

圖 1-2 顧客-零件-供應商的八種視域組合

如果資料方格中所隱含的所有視域都在處理查詢時才產生，查詢處理的速率將會很慢。所以如何選擇視域來實體化 (materialized) 以增進查詢處理的速率，是一個值得探討的課題。由於相依性的關係，並非所有視域皆需實體化。但在實際建置系統時，又有以下三個限制：(1) 實體化視域必須能處理所有查詢。 (2) 實體化視域所需的儲存空間總和必須在系統的儲存空間限制之內。 (3) 應盡可能地降低實體化視域之整體查詢成本。

針對視域實體化的問題，本研究採兩階段的處理方式。首先在第一階段以「減少儲存空間」爲目標，運用視域間相依性關係來減少不一定要實體化的視域。第二階段則是在有限的儲存空間限制下，以增加實體化視域的方式來「降低整體查詢成本」。本篇論文其餘章節如下：第二節介紹資料方格查詢成本模式及貪進視域選擇法。第三節闡述本研究的處理方法。第四節以模擬程式評估並分析查詢成本。第五節則做一總結並探討未來發展的方向。

2 資料方格

資料方格的查詢成本可用以下的線性成本模式來表示 [HRU96]：

$$T = m * S + c$$

T 表示查詢所花費的時間（成本），S 表示查詢視域的筆數， m 爲 T 與 S 之比率， c 爲查詢單一筆資料所需時間。此線性成本模式以實驗證明 m 約略爲固定值。我們以範例 2-1 說明貪進視域選擇法 [HRU96] 的內容。

【範例 2-1 】一由顧客、零件、供應商以及時間所構成的四維資料方格，其格狀圖如圖 2-1 所示。假設使用者總共使用到視域 cpt 、 pst 、 cp 、 cs 、 pt 、 st 、 c 、 p 以及 s 等九個視域來處理查詢。其中 (cpt, 70) 表示視域名稱爲 cpt ，視域有 70 筆資料。各視域的使用情形如圖 2-2 ，其中 (cpt, 0.08) 表示視域名稱爲 cpt ，使用頻率爲 0.08 。我們用貪進視域選擇法再選取三個視域來實體化。

首先起始狀態頂端視域 cpst 必須實體化（演算法基本假設），如此便保證能處理範例 2-1 中的九個視域。另外定義一個效益函數 (benefit) 來代表若實體化單一視域時的效益。例如：若選擇視域 cp 實體化，對視域 cp 、 c 以及 p 的查詢而言，將發生查詢重新配置的情形。以視域 cp 爲例，查詢成本由原來在 cpst

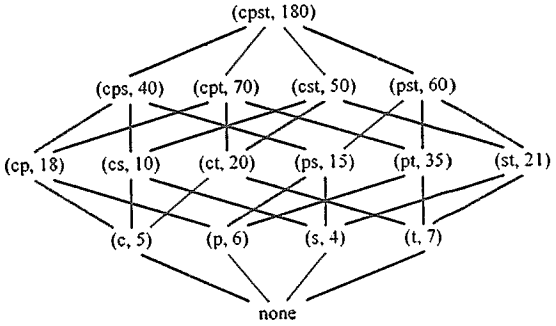

圖 2-1 顧客-零件-供應商-時間之格狀圖

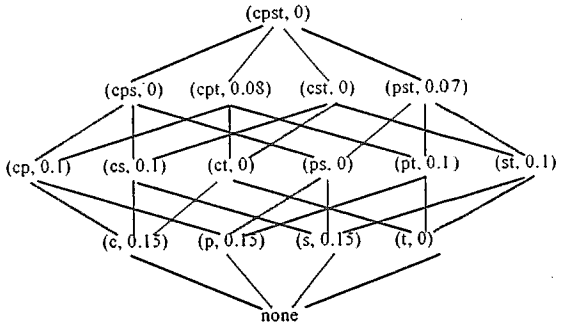

圖 2-2 查詢使用情形（頻率）

處理的 18 (= 180 * 0.1) 降低爲在 cp 處理的 1.8 (= 18 * 0.1)，故效益爲 16.2 (= 18 - 1.8)。對視域 c 或 p 而言，查詢成本由原來在 cpst 處理的 27 (= 180 * 0.15) 降低爲在 cp 處理的 2.7 (= 18 * 0.15)，效益爲 24.3 (= 27 - 2.7)。所以視域 cp 、 c 以及 p 的總合效益爲 64.8 (= 16.2 + 24.3 + 24.3)。表 2-2 爲範例 2-1 效益提昇之情形，橫軸表示第幾次選擇，縱軸表示加入單一視域的名稱。如此繼續計算增加其它視域的情形，在第一次選擇將產生十四種不同的組合，其中以視域 cpst 與 cps 的組合最佳，所以下次選擇便以此組合出發，繼續再挑選下一個視域。經過三次選取後，結果是選擇實體化視域 cpst 、視域 cps 、視域 pst 以及視域 ct。

	1st selection	2nd selection	3rd selection
cps	140(.1*2+.15*3)=91	selected	selected
cpt	110(.08+.1*2+.15*2)=63.8	110(.08+.1)=19.8	110*.08=8.8
cst	130(.1*2+.15*2)=65	130*.1=13	10*.1=1
pst	120(.07+.1*2+.15*2)=68.4	120(.07+.1*2)=32.4	selected
cp	162(.1+.15*2)=64.8	22(.1+.15*2)=8.8	22(.1+.15*2)=8.8
cs	170(.1+.15*2)=68	30(.1+.15*2)=12	30(.1+.15*2)=12
ct	160*.15=24	160*.15=24	160*.15=24
ps	165(.15*2)=49.5	25(.15*2)=7.5	25(.15*2)=7.5
pt	145(.1+.15)=36.25	145*.1+5*.15=15.25	25*.1+5*.15=3.25
st	159(.1+.15)=39.75	159*.1+19*.15=18.75	39*.1+19*.15=6.75
c	175*.15=26.25	35*.15=5.25	35*.15=5.25
p	174*.15=26.1	34*.15=5.1	34*.15=5.1
s	176*.15=26.4	36*.15=5.4	36*.15=5.4

表 2-2 範例 2-1 效益提昇之情形

[HRU96] 所提出的貪進視域選擇法有以下兩個問題值得探討。第一個問題是起始狀態的選擇。在 [HRU96] 中嚴格定義頂端視域 (top view) 必須實體

化。這是爲了找出一個可以處理所有查詢的視域。而在範例 2-1 中此視域恰巧爲頂端視域 cpst。同樣是圖 2-1 之格狀圖，若假設是使用視域 cs 、 ct 以及 st 三個視域處理查詢的情形，則可實體化視域 cst 便同樣地能處理所有的查詢。再者，貪進視域選擇法是一次增加一個實體化視域，所以儲存空間會不斷增加。若起始狀態所選擇的實體化視域的儲存空間已經超過系統儲存空間的限制，則貪進視域選擇法便無用武之地。實體化頂端視域除了有儲存空間的限制之外，它的整體查詢成本也是最差的。貪進視域選擇法的第二個問題是其本身的瑕疵。在圖 2-1 中若實體化視域 cpt 及 pst ，將導致視域 cpst 並未處理任何查詢，所以並無實體化的需要。但在貪進視域選擇法中並未考量刪除這種未處理查詢的視域。另外增加實體化視域時，考量的應是「儲存空間」而非實體化視域的「個數」。

3 針對查詢選擇實體化視域

是否選擇實體化某一個視域牽涉到兩個問題，一是「儲存空間」，二是「查詢成本」。但想要同時減少「儲存空間」與「查詢成本」卻相當不容易。一般來說，增加一個實體化視域，可預見地會增加儲存空間而降低查詢的執行成本，故兩者是相互衝突的。由於不易清楚的界定兩者的關係，本研究採取兩階段的處理方式將「儲存空間」與「查詢成本」的問題分開處理。首先在第一個階段以「儲存空間」最小化爲目標，然後將結果輸出給第二階段做「查詢成本」最小化的處理。由於篇幅的限制，本篇論文只以範例說明這個兩階段演算法，詳細內容請參閱[Liu97]。

3.1 階段一：減少儲存空間

在第一個階段以減少儲存空間爲目標，並且必須符合「實體化視域的集合必須處理所有查詢」的限制。

【範例 3-1 】一由顧客、零件、供應商以及時間所構成的四維資料方格，其格狀圖如圖 2-1 。假設使用者總共使用到視域 cpt 、 pst 、 cp 、 cs 、 pt 、 st 、 c 、 p 以及 s 等九個視域來處理查詢。

在範例 3-1 中，若沒有儲存空間的限制條件，或者九個視域儲存空間的總和在系統儲存空間的限制條件之內，則只要直接實體化此九個視域便可得到最佳的查詢執行成本。然而在儲存空間不足的情況時，我們可利用視域間相依性的關係來減少儲存空間。首先是不實體化部份視域：例如，在視域 s 上處理之查詢，同樣地也可以在視域 st 上完成。所以若實體化視域 st ，則視域 s 便不一定要被實體化。其次是實體化沒有被查詢的視域：例如爲了處理視域 cp 、 cs 、 c 、 p 以及 s 上的查詢，若將五個視域皆實體化，則儲存空間的需求爲 18 + 10 + 5 + 6 + 4 = 43 。但因爲視域 cps 可以處理上述五個視域的查詢，即表示若實體化視域 cps ，則上述五個視域便毋需實體化。視域 cps 的儲存空間僅需 40 ，較上述五個視域儲存空間總和 43 爲低。這兩種情況均是利用視域間的相依關係來取代視域。這些視域皆是被其祖先視域 (ancestor view) 所取代。

我們定義初始實體化視域集合 (initial materialized view set) 爲查詢使用到的所有視域。而祖先視域集合 (ancestor view set) 是指不含本身視域的所有祖先視域的集合。在圖 2-1 的格狀圖中，除視域 cpst 外，其餘視域皆有祖先視域集合。例如：視域 cp 的祖先視域集合爲視域 cpst 、 cps 以及 cpt 等三個視域。將初始實體化視域集合中所有視域的祖先視域集合做聯集的處理，其結果稱爲總祖先視域集合 (total ancestor view set)。由於視域只能被其祖先視域所取代，所以只有總祖先視域集合中的視域才有可能取代初始實體化視域集合中的視域。

在範例 3-1 中，假設以視域 cp 取代視域 c 及 p ，則僅須實體化視域 cpt 、 pst 、 cp 、 cs 、 pt 、 st 以及 s 等七個視域即可處理所有查詢。但因爲以視域 cs 來取代視域 c 的查詢執行成本會較用視域 cp 來取代視域 c 爲佳，故我們選擇視域 cs 來處理視域 c 上的查詢。這個步驟就稱爲查詢重新配置。另一種查詢重新配置的情形如下：假設第一輪測試選擇的視域集合爲視域 cpt 、 pst 、 cp 、 cs 、 pt 、 st 以及 s 等七個視域，其中視域 cp 處理視域 cp 與 p 上的查詢，視域 cs 處理視域 cs 與 c 上的查詢，其餘五個視域均處理本身視域之查詢。進入第二輪測試後，首先找出總祖先視域集合。若選擇實體化視域 ps ，則產生的實體化視域集合爲視域 cpt 、 pst 、 cp 、 cs 、 ps 、 pt 以及 st 等七個視域。經過前述之第一種查詢重新配置的處理，僅視域 s 的查詢由視域 ps 轉配置至視域 cs 上處理。但對視域 p 的查詢而言，當視域 ps 未被選擇實體化時，視域 p 的查詢在視域 cp 上處理，然而當視域 ps 被實體化後，視域 p 的查詢將於視域 ps 上處理。重新配置的目的就是讓所有的查詢都能夠在一個合理（最佳查詢執行成本）的視域上處理。

我們將第一階段的所有程序整理如下：首先必須將初始實體化視域集合內的九個視域皆實體化方能處理所有查詢，其次找出總祖先視域集合，然後自其中揀選一個視域來取代初始實體化視域集合內的子孫視域，完成取代動作將需要重新配置的查詢逐一調整，重新配置後刪除不必要的視域，所產生的結果稱爲候選實體化視域集合 (candidate materialized view set)。候選實體化視集合亦可處理所有查詢。在一輪測試中，必須將總祖先視域集合中所有視域均揀選處理後才算完成一輪測試。以範例 3-1 爲例，在第一輪測試中將會產生十一組候選實體化視域集合。然後在所有候選實體化視域集合中，選擇一組最少儲存空間的候選實體化視域集合作爲下一輪測試的初始實體化視域集合。進入下一輪測試時，總祖先視域集合將會逐漸地減少，直到總祖先視域集合減少至空集合，第一階段便終止。

表 3-1 爲範例 3-1 各輪測試候選實體化視域集合總儲存空間的變化情形，縱軸表示自總祖先視域集合揀選的視域名稱，「-」表示視域在此回合不爲總祖先視域集合中之視域。

	1st	2nd	3rd	4th	5th
cpst	180	180	180	180	-
cps	226	170	170	-	-
cpt	165	140	-	-	-
cst	239	198	180	-	-
pst	163	-	-	-	-
cp	218	158	-	-	-
cs	220	158	-	-	-
ct	224	158	-	-	-
ps	234	-	-	-	-
pt	223	-	-	-	-
st	225	-	-	-	-

表 3-1 範例 3-1 候選實體化視域集合總儲存空間

在第一輪測試將產生十一組候選實體化視域集合，其中以視域 pst 的候選實體化視域集合儲存空間最少。接下來在第二、三輪發現選取視域 cpt 及 cps 的候選實體化視域集合儲存空間最少。在總候選實體化視域集合內以 cpt（處理視域 cpt 與 cp 的查詢）、pst（處理視域 pst 、 pt 、 st 以及 p 的查詢）以及視域 cs（處理視域 cs 、 c 以及 s 的查詢）所組成的候選實體化視域集合的儲存空間最少。故以此組候選實體化視域集合爲下一輪測試的初始實體化視域集合。

3.2 階段二：減少查詢成本

第二階段的目是要在有限的儲存空間下獲得較佳的查詢成本。本階段僅對儲存空間最小的基本實體化視域集合做處理。另外，系統的儲存空間限制爲一任意指定值。

【範例 3-2 】一四維資料方格由顧客、零件、供應商以及時間所構成，其格狀圖如圖 2-1 。假設使用者總共使用到視域 cpt 、 pst 、 cp 、 cs 、 pt 、 st 、 c 、 p 以及 s 等九個視域來處理查詢。圖 2-2 表示各視域使用之情形。假設一開始選擇實體化視域 cpt 、 pst 以及 cs 處理查詢。儲存空間以 200 爲限。

如上述，開始時視域 cpt 處理視域 cpt 與 cp 上的查詢，視域 pst 處理視域 pst 、 pt 、 st 以及 p 上的查詢，視域 cs 處理視域 cs 、 c 以及 s 上的查詢。我們的處理方法是一次增加一個視域，使查詢轉配置至新增加的視域上處理。例如：增加視域 cp 後，視域 cp 與 p 的查詢便需要重新配置處理。但如同在第一階段取代處理一般，並非隨意地增加一個視域便會發生查詢重新配置的情形。

一個查詢除了可在其原始的視域上處理外，亦可在它的祖先視域上處理。所以一組查詢只可以在其總祖先視域集合與查詢視域集合之聯集上處理。另外已實體化的視域亦毋需考慮。最後我們還要刪除儲存空間過大的視域。經過以上的篩選，我們僅需測試視域 cps 、 cst 、 cp 、 ct 、 ps 、 pt 、 st 、 c 、 p 、 s 等十個視域。當然，我們也可以逐一測試格狀圖內所有的視域（十五個），但如此做會增加演算法的複雜度。

在進行的過程中，首先揀選一個視域並將其加入實體化之視域集合，接著做查詢重新配置的動作，然後再刪除未處理查詢之視域，最後重新計算整體實體化視域之儲存空間與整體執行成本。在一輪測試中，必須將所有的視域皆揀選處理，然後自其中選擇一組整體儲存空間仍在限制之內、整體查詢執行成本最佳的實體化視域集合，以做爲進入下一輪測試的基礎。

表 3-2 爲範例 3-2 整體查詢成本之變化。橫軸表示第幾輪測試，縱軸表示單獨加入之實體化視域，數值爲單獨加入此實體化視域後的整體查詢成本。「-」表示在某輪測試未改變整體查詢成本者（未發生查詢重置者或整體儲存空間超過儲存空間限制者）。「 m 」代表已被實體化的視域。以第三輪爲例，開始時已被實體化的視域爲 cpt 、 pst 、 cp 、 cs 以及 st 。單獨計算加入各實體化視域後的整體查詢成本以 pt 的 23.9 爲最低。但因爲實體化視域 pt 會造成整體儲存空間超過儲存空間限制 (200)，所以選擇實體化視域 p 。在第六輪測試時已無增加任何一個視域之可能，故已完成增加實體化視域的工作。最後輸出的實體化視域集合爲視域 cpt 、 pst 、 cp 、 cs 、 st 、 c 、 p 以及 s ，其中僅視域 pst 處理視域 pst 與 pt 上的查詢，其餘視域皆處理本身查詢。整體儲存空間爲 194 ，整體查詢執行成本爲 22.95 。

	1st	2nd	3rd	4th	5th	6th
cpst	-	-	-	-	-	-
cps	35.8	-	-	-	-	-
cpt	m	m	m	m	m	m
cst	40.8	29.3	-	-	-	-
pst	m	m	m	m	m	m
cp	30.3	m	m	m	m	m
cs	m	m	m	m	m	m
ct	-	-	-	-	-	-
ps	35.05	29.85	25.95	-	-	-
pt	35.55	27.8	-	-	-	-
st	37.9	26.4	m	m	m	m
c	41.05	29.55	25.65	23.85	m	m
p	33.7	28.5	24.6	m	m	m
s	40.9	29.4	25.5	24.9	22.95	m
t	-	-	-	-	-	-

表 3-2 範例 3-2 整體查詢成本之變化

4 實驗結果與分析

本研究之模擬程式以 C 語言撰寫，在 Sun 工作站的環境下執行。 4.1 節實驗單一資料方格中不同查詢數目間的關係。 4.2 節實驗不同維度的資料方格的關係。

4.1 單一資料方格

本實驗以五維資料方格（如圖 4-1 ）爲實驗對象，其中 (0, 1500) 表示視域的編號爲 0 ，儲存空間爲 1500 。依使用者查詢的數目區分爲四組，每組實驗五次，每次的查詢均以隨機的方式產生。表 4-1 至 4-4 爲四組實驗結果。表格最右欄爲無儲存空間限制之查詢成本，右邊數來第二欄爲以實體化所有被查詢

視域爲儲存空間限制，然後儲存空間限制由右至左逐次遞減 5%。

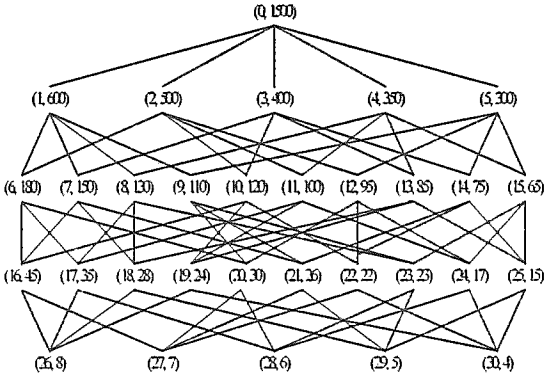

圖 4-1 五維資料方格格狀圖

第一組實驗設定查詢個數爲 10 。在查詢數目過少的情況下，儲存空間限制小於實體化頂端視域，貪進法不適用。

一	貪進法	-	-	-	-	-	-	-	112
	兩階段	-	-	-	181	148	137	114	112
二	貪進法	-	-	-	-	-	-	-	126
	兩階段	-	-	-	261	135	135	127	126
三	貪進法	-	-	-	-	-	-	-	94
	兩階段	231	155	143	122	120	116	95	94
四	貪進法	-	-	-	-	-	-	-	60
	兩階段	-	191	146	133	86	75	60	60
五	貪進法	-	-	-	-	-	-	-	123
	兩階段	-	198	172	160	133	129	125	123

表 4-1 第一組實驗之查詢成本（ 10 個查詢）

第二組實驗設定查詢個數爲 15 。在本組實驗中，儲存空間限制變化相當大，其中第二次實驗與第一組的情況相似。第三次實驗則有部分情況貪進法仍可應用。其餘三次因本研究第一階段輸出之基本實體化視域集合爲頂端視域，故當儲存空間限制逐漸縮小時，發生相等現象。

一	貪進法	-	1307	944	837	898	800	583	131
	兩階段	-	1307	266	202	164	147	132	131
二	貪進法	-	-	-	-	-	-	-	70
	兩階段	-	151	118	108	87	78	71	70
三	貪進法	-	-	-	-	1116	732	783	127
	兩階段	349	296	227	197	148	136	128	127
四	貪進法	-	1162	861	735	608	500	374	132
	兩階段	-	1162	197	164	151	140	134	132
五	貪進法	-	912	883	707	489	434	392	162
	兩階段	-	912	883	236	201	171	163	162

表 4-2 第二組實驗之查詢成本（ 15 個查詢）

第三組實驗設定查詢個數爲 20 。在本組實驗中，儲存空間限制相當大，貪進法大致上均可應用。本研究第一階段輸出之基本實體化視域集合爲頂端視域的情況亦增加許多。

一	貪進法	-	921	603	648	452	350	290	94
	兩階段	-	921	165	130	117	105	95	94
二	貪進法	-	1067	850	766	611	518	553	108
	兩階段	-	1067	164	145	128	117	108	107
三	貪進法	-	801	554	452	324	296	425	107
	兩階段	-	801	554	180	143	118	107	107
四	貪進法	-	-	1070	790	649	660	484	87
	兩階段	150	134	128	112	102	98	88	87
五	貪進法	-	810	630	544	466	313	503	107
	兩階段	-	811	629	159	132	117	108	107

表 4-3 第三組實驗之查詢成本（ 20 個查詢）

第四組實驗設定查詢個數爲 25 。本組實驗幾乎查詢所有視域(25/30)，兩階段法的第一階段輸出之基本實體化視域集合均爲頂端視域。而當儲存空間限制很大時，貪進法的頂端視域幾乎不處理任何查詢，故查詢成本與兩階段演算法相等。

一	貪進法	553	480	458	400	134	127	114	114
	兩階段	553	480	157	143	133	124	114	114
二	貪進法	499	413	369	370	329	281	131	131
	兩階段	499	413	209	173	156	141	131	131
三	貪進法	555	438	450	350	343	269	111	117
	兩階段	555	171	155	143	133	122	111	117
四	貪進法	818	756	572	458	504	371	413	97
	兩階段	818	756	143	127	119	106	97	97
五	貪進法	479	472	560	432	320	132	120	120
	兩階段	479	193	167	153	143	128	120	120

表 4-4 第四組實驗之查詢成本（ 25 個查詢）

各組實驗中，貪進法不一定能夠運用，故僅擷取貪進法可運用的情況下比較兩個方法的查詢成本。表 4-5 至 4-7 爲後三組實驗的查詢成本比較。數值 =(兩階段法 / 貪進法) * 100%。

第一次	100	28.1	24.2	18.3	18.4	22.6	35.3
第三次	-	-	-	13.3	18.6	16.4	16.1
第四次	100	22.9	22.4	25.0	28.0	35.8	39.0
第五次	100	100	33.4	41.2	39.5	41.5	59.3
平均							37.4

表 4-5 第二組實驗查詢成本比較

第一次	100	27.3	20.0	25.9	30.1	32.6	39.3
第二次	100	19.3	19.0	21.0	22.6	19.6	33.6
第三次	100	100	39.8	44.2	39.8	25.2	58.2
第四次	-	11.9	14.2	15.7	14.8	18.1	15.0
第五次	100	100	29.3	28.4	37.5	21.5	52.8
平均							39.8

表 4-6 第三組實驗查詢成本比較

第一次	100	100	34.2	35.9	99.5	97.7	100	81.0
第二次	100	100	56.6	46.7	47.5	50.2	100	71.6
第三次	100	39.1	34.6	40.8	38.7	45.6	100	57.0
第四次	100	100	25.1	27.7	23.5	28.6	23.6	46.9
第五次	100	40.9	29.8	35.4	44.8	97.1	100	64.0
平均								64.1

表 4-7 第四組實驗查詢成本比較

從表 4-5 至 4-7 可知，同組實驗內每次實驗的查詢成本改善情況差異均不小。就平均而言，表 4-5 與 4-6 查詢成本平均改善約略相等。表 4-7 在所有查詢幾乎皆使用的情況下，查詢成本平均改善情況雖不大，但兩階段法仍只有貪進法 64.1%的水準。

4.2 不同維度之資料方格

爲比較不同維度資料方格的影響，我們對四、五、六以及七維度的資料方格各做一組實驗，每組實

驗仍重覆做五次。查詢的個數約爲資料方格所含視域個數的三分之一。儲存空間以實體化所有查詢視域爲限制，逐次遞減 1%。依據前四組的結果顯示，由於貪進法未必皆能運用，故查詢成本比較值改爲（空間限制下的查詢成本 / 無空間限制下的查詢成本）。各組實驗結果如表 4-8 至 4-11 。

第一次	1.833526490	1.833526490	1.014454087
第二次	-	1.152678616	1.005669047
第三次	2.720396728	2.720396728	1.007922784
第四次	1.812026652	1.812026652	1.014399028
第五次	-	-	1.005531873
平均	2.121983290	1.879657121	1.009595364

表 4-8 第五組實驗（四維資料方格）查詢成本比較

第一次	2.71094553	1.813926715	1.000281867
第二次	-	3.262223242	1.000383768
第三次	-	3.439755289	1.009877582
第四次	-	-	1.000199569
第五次	-	-	1.000342960
平均	2.71094553	2.838635082	1.002217149

表 4-9 第六組實驗（五維資料方格）查詢成本比較

第一次	1.808640000	1.808640000	1.000000029
第二次	2.061343230	2.061343230	1.000000242
第三次	1.494591037	1.596957852	1.000004517
第四次	1.237421188	1.237421188	1.000001438
第五次	1.613023976	1.240811675	1.000006723
平均	1.643003886	1.589034789	1.000002590

表 4-10 第七組實驗（六維資料方格）查詢成本比較

第一次	1.338074465	1.338074465	1.000000001
第二次	-	-	1.000000002
第三次	1.859533211	1.301639365	1.000000007
第四次	1.560799576	1.291605412	1.000000017
第五次	1.751338378	1.751338378	1.000000186
平均	1.627436407	1.420664405	1.000000043

表 4-11 第八組實驗（七維資料方格）查詢成本比較

然後將各組的平均值整理於表 4-12 。

四維資料方格	2.121983290	1.879657121	1.009595364
五維資料方格	2.710945530	2.838635082	1.002217149
六維資料方格	1.643003886	1.589034789	1.000002590
七維資料方格	1.627436407	1.420664405	1.000000043

表 4-12 不同維度間資料方格的查詢成本比較

除第五組實驗外，在同組實驗內，平均查詢成本隨儲存空間增加而遞減。同樣地，在相同儲存空間限制比率下，較高維度的平均查詢成本亦較低。這是因爲一方面選取的查詢個數較多，另一方面較高維度的資料方格變化較複雜，因此產生的組合較多，故兩階段的方法可以得到較佳的成效。

5 結論與未來研究方向

資料倉儲結合資料庫與決策支援的概念，可及時提供使用者有用的資訊。因爲受限於硬體設備，在建置系統時經常無法實體化使用到的所有視域。本研究提出一種啓發式演算法來選擇被實體化的視域，以達到在有限的儲存空間下盡可能地降低查詢成本的目的。

本研究提出之兩階段演算法的第一階段以減少儲存空間爲目標，在起始狀態的選擇上較爲嚴謹，也以更具彈性的方式處理各種可能的情況。另外，在第二階段也修正了貪進視域選擇法的一些缺點，讓我們更能夠充分的利用儲存空間來降低查詢成本。最後經由實驗證實，隨著使用查詢的不同，結果有相當大的差異。但由於彈性的處理方式，使兩階段演算法仍能夠找尋到較佳的結果。但此演算法仍有一些需要改善之處。 1. 資料方格架構：目前實驗的資料方格僅包含維度間的階層，我們仍需提供方法處理含有維度與屬性階層關係的資料方格。 2. 演算法的改進：實驗結果顯示，當所有視域幾乎皆被使用時，第一階段的方法未發揮功能，導致改善查詢成本的效果不大。這是需要改進之處。另外，後續的研究有必要繼續探索其他的啓發式演算法。

參考文獻

[GBLP95] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh, "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals," Microsoft Technical Report No. MSR-TR-95-22.

[Gup97] H. Gupta, "Selection of Views to Materialize in a Data Warehouse," *Proceedings of the International Conference on Data Theory*, Athens, Greece, Jan. 1997.

[HRU96] V. Harinarayan, A. Rajaraman, and J. D. Ullman, "Implementing Data Cubes Efficiently," *Proceedings of ACM SIGMOD Conference*, Montreal, Canada, June 1996.

[KS95] R. Kimball and K. Strehlo, "Why Decision Support Fails and How to Fix it," *ACM SIGMOD Record*, Vol. 24, No. 3, Sep. 1995.

[Liu97] 劉宇昌，「在資料倉儲中針對查詢選擇實體化視域之研究」，碩士論文，國立屏東科技大學資訊管理研究所， Jun. 1997 。

[McG96] F. McGuff, "Data Modeling for Data Warehouse," http://members.aol.com/fmcguff/dwmodel.html, OCT. 18, 1996.

[QGMW96] D. Quass, A. Gupta, I. S. Mumick, and J. Widom, "Making Views Self-Maintainable for Data Warehousing," *Proceedings of the Conference on Parallel and Distributed Information System*, Miami Beach, FL, Dec. 1996.

[SSU96] A. Silberschatz, M. Stonebraker, and J. Ullman, editors, "Database Research: Achievements and Opportunities into the 21st Century," *ACM SIGMOD Record*, Vol. 25, No. 1, Mar. 1996.