TR-81-006

分散式資料庫系統之詢問處理

研究報告

計畫編號：NSC70-0404-E001-01

執行限期：69.年 12 月 1 日至70.年 11 月 30 日

計畫金額：NT$ 280,800.

日　　期：中華民國　七十　年　十二　月

分散式資料庫系統之詢問處理

柯　志　昇

中央研究院資訊科學研究所

摘　要

本文旨在介紹分散式資料庫系統之詢問處理的意義、方法、與策略。文後並提出幾個將來有待進一步研究的相關問題。

1 引介

分散式資料庫系統(Distributed Data Base System, 簡稱 DDBS)是計算機網路和資料庫兩大技術的結合，前者代表資訊的廣泛分佈，後者代表資訊的整合。這種新科技趨勢起源於大企業機構經營管理上的需要。一個大企業機構往往有很多散佈在各地的分支機構，每一分支機構有類質相似或相關而來源不同的資訊。諸如，銀行系統、郵政系統、旅遊業系統、國防命令管制系統，....等等。

計算機與其週邊儲存器被用來作資料之收集、儲存與處理，時日已久；以往都集中於一地處理，對於遠地的資料則須利用郵電系統來傳達；近日，因計算機硬體和通訊網路的設備費用與日降低，使得代表資訊廣泛分佈的計算機網路逐漸普遍，對此廣泛分佈的資訊，更須要一個有效的管理系統來使用及管理分散在各地的資源—此即分散式資料庫管理系統。

大致上來講，分散式資料庫系統比起傳統集中式資料庫系統有下列之優點：

一、分散資料庫管理系統的負荷—一個種類繁複的資料可依其來源與使用情形歸類分別存在對整個系統效率最有利的地方。同時，詢問可化成一組子詢問，而這些子詢問可分別讓各地的資料庫管理系統作並行處理。

二、縮短詢問的反應時間—若詢問之內涵祇牽涉到本地的資料，本地的資料庫管理系統即可解決；較之集中式資料庫系統，可節省很多不必要的傳輸時間。若詢問之內涵牽涉到外地的資料，以計算機網路來傳遞資料是爲最快的方式。

三、可靠性與妥用性(Reliability & Availability) —— 因爲分散式資料庫系統是建立在分散式處理系統上，若某地之系統故障停機，並不影響其他地方之運轉。而且，在各地存放相同但常爲該地參考到的資料，可增強分散式資料庫系統的可靠性及妥用性。

四、易爲企業界所接受—因爲分散式資料庫系統之觀念適合具有各地分支機構的大企業之經營管理型態。

分散式資料庫系統的觀念新穎引人，但同時也產生了一些傳統資料處理所不曾發生的問題，其中之一就是詢問處理 (Query Processing) 。在分散式資料庫系統中，一個詢問往往會牽涉到儲存在兩個以上不同地方的資料，因此，其詢問處理有兩大特點：

一、散佈在不同地方的計算機可以作局部並行處理 (parallel local processing).

二、資料必須在各地之間搬運傳輸。以目前的傳輸技術，計算機處理資料的速率極高於資料傳輸的速率。因此，一個詢問的回覆時間主要是決定於資料傳輸量的多寡。

根據這兩個特點，吾人可利用各種不同的策略作詢問處理，以期回覆時間最短、處理費用最低。本文之目的即是在介紹兩種主要的詢問處理策略模型及討論其效應。

2 詢問處理的目的和評估

2.1 問題定義與假設條件

從一DDBS 中存取資料時，一個詢問往往要分解成一組相關的子詢問，而一個子詢問可能參考到兩個不同地方的資料；換言之，一組子詢問可能會在不同的地方作局部並行處理。問題是，執行這些子詢問的次序和地點往往會影響到回覆時間的長短以及整個系統的工作效率。如何安排執行各個子詢問的次序和地點，是 DDBS 詢問處理的核心問題。

爲了簡化問題的複雜性和讓吾人更容易瞭解問題，在不失一般性的原則之下，我們將做下列幾點假設：

一、關係模式資料庫——每一地的資料庫都是關係模式。

二、均勻的資料傳輸花費——對同一筆資料，在網路上任一地點傳到任一非本地點的費用一律相同。這種假設對於衛星通訊網路及完全連結網路(fully connected network) 是很合理的。

三、均勻的資料處理花費——對同一筆資料，在網路上任一地點作同樣的處理，其費用一律相同。這種假設可視爲網路上每一地所使用的計算機系統是一樣的。

四、計算機處理資料的速率極大於資料傳輸的速率。

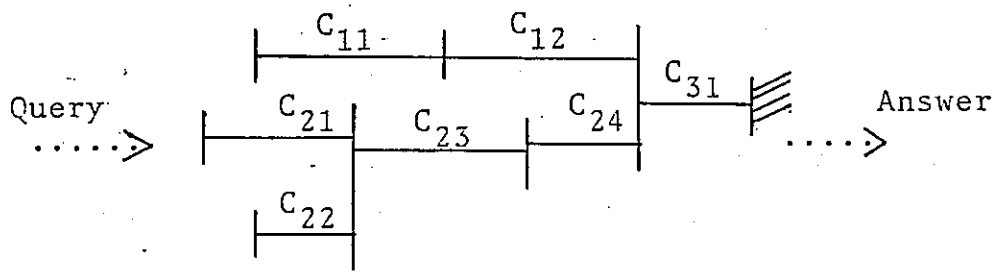
2.2 系統效率之測定標準

當吾人進行DDBS的詢問處理時，必須有一標準來衡量處理方式的好壞。在本文我們將以下列二點作爲測定詢問處理策略效率之標準：

一、平均回覆時間的長短——對於發出詢問的使用者而言，平均回覆時間愈短愈受歡迎。這種測定標準較適用於工作負載較輕且要求快速回覆的即時應用系統。

二、詢問處理的全部花費——詢問處理的全部時間花費往往直接影響到整個系統的工作效率。因此，對於工作負載較重的系統，我們將目標擺在減低全部的詢問處理時間和花費。

圖一表示處理一詢問（即一組子詢問）之時間花費圖，水平的線段 C_{ij} 是爲解決某子詢問其資料傳輸的費用，平行的水平線段代表可做局部並行處理的子詢問，於此可看出所謂〝局部〞的意義。垂直的線段表示執行子詢問的地點，因爲前面已假設計算機的處理速度極大於資料傳輸的速度，故其花費可忽略之。



〔圖一〕．詢問處理時間花費圖例

對於圖一這個詢問，其回覆時間是爲 $C_r = C_{21} + C_{23} + C_{24} + C_{31}$，而整個系統全部的時間花費爲 $C_t = \sum_{i,j} C_{ij} = C_{11}+C_{12}+C_{21}+C_{22}+C_{23}+C_{24}+C_{31}$. 運用不同的策略吾人可得到不同的 C_r 和 C_t。但是應注意，C_r 和 C_t 彼

儲蓄部　　營業部

客戶資料 T

ID	NAME	DEPOSIT
C01	CHENG	1,000,000
C05	WONG	150,000
C20	HONG	200,000

分行客戶資料 C

ID	NAME	DEPOSIT
C01	CHENG	300,000
C04	CHANG	200,000
C10	CHEN	100,000

分行客戶資料 H

ID	NAME	DEPOSIT
C01	CHENG	200,000
C02	LIN	20,000
C10	CHEN	50,000

分行客戶資料 K

ID	NAME	DEPOSIT
C01	CHENG	150,000
C05	WONG	100,000

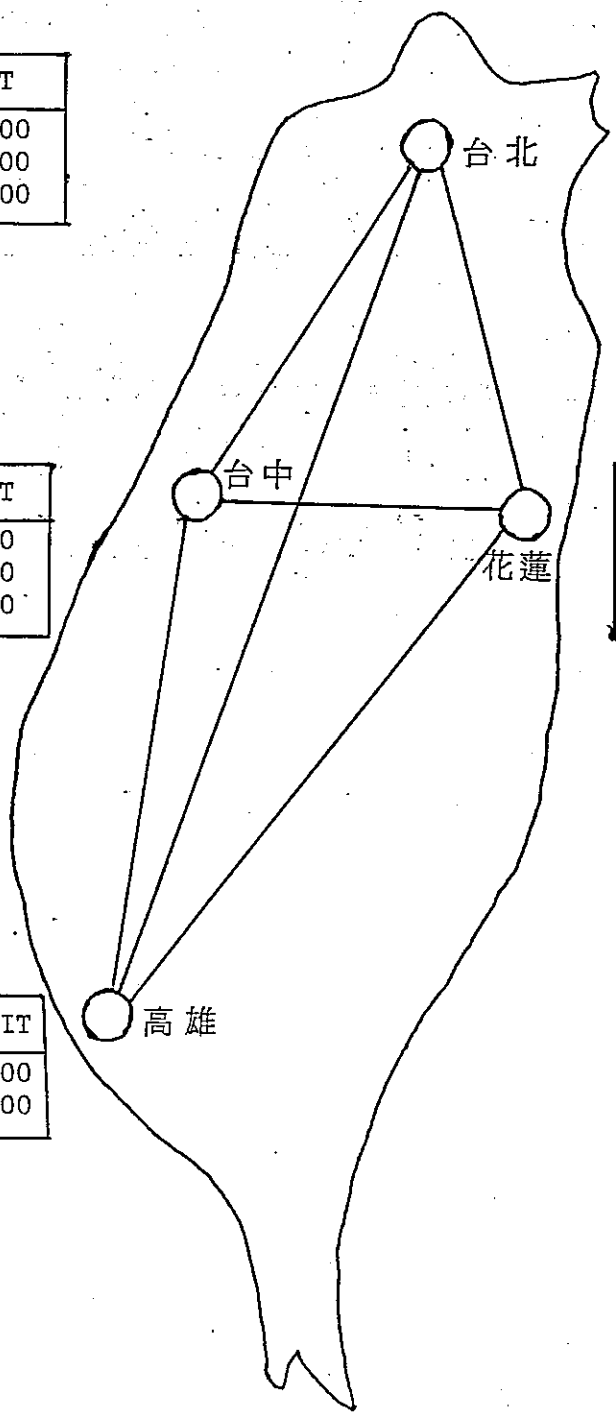


圖　二

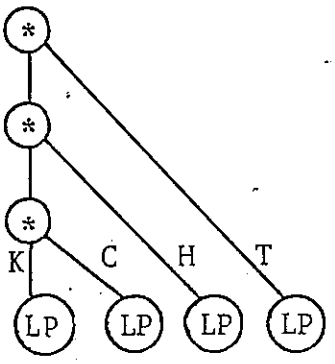
此間並沒有絕對的正比關係；換言之，C_r 最小時，其 C_t 未必也是最小；反過來，C_t 最小時，也未意味著 C_r 是最小。

3. DDBS 詢問處理之概觀

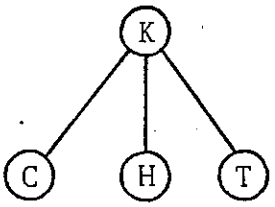
當使用者由某地發出一個詢問 Q 時，首先，詢問處理系統必須將該詢問解析成一組較簡單且相關的子詢問，然後先作祇參考到一地資料的子詢問，於是餘下的子詢問都是參考到不同兩地資料者。這些子詢問的關係可以一個詢問圖 G_Q 來表示。詢問圖可分爲二大類（參見圖三）。其一，節點(node) 代表運算子而支邊 (edge) 是表資料集；第二類則是以節點代表資料集而支邊代表運算子。不論是那一類，對一詢問 Q 而言，由於詢問解析時之遞移律封閉〔2〕將使得詢問圖並非唯一。

以下吾人將利用一個實例來說明。如圖二所示，假設某銀行在台北、台中、高雄、花蓮都有分行。今有某位先生想知道他的太太是不是在這四家分行都有存款。對於這個詢問吾人可以圖三.A和圖三.B兩個不同類型的詢問圖表示之。

Q = Find the customers who have deposits in all branches.



〔圖三.A〕 G_Q- type 1 ;
"*" 代表 join
L.P 即 Local Processing



〔圖三.B〕 G_Q-type 2

〔圖三〕 二種不同類型的詢問圖 K. C. H. T.分別表示在高雄、台中、花蓮、台北的關係檔。

吾人依詢問圖三.A 可得一運算次序如下：

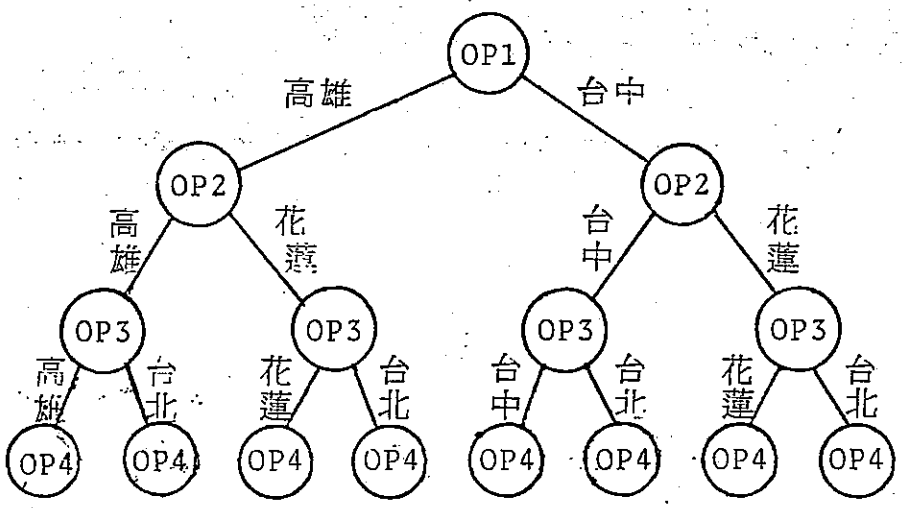
OP1 : join K and C into R1

OP2 : join R1 and H into R2

OP3 : join R2 and T into R3

OP4 : move R3 to query requesting site

我們可以作一決策樹來描述對於圖三.A之各種不同解詢法：



node : Operation

edge : site that performs operations 〔圖四〕

在此，我們要再強調一點，G_Q 不是唯一的，操作次序不是唯一的，操作地點的選擇也不是唯一的。就這個簡單的例子，吾人至少有幾十種不同的解詢方式可以得到同一答案，問題是那一種方式才是最好的？下節我們將歸納出兩個不同的策略模型，可用來求一較好的解詢法。

4. 詢問處理之策略分析

在關係模式資料庫系統中，主要的運算有Selection, Projection和 join（前兩者是屬於一元運算(unary operation)，後者是屬二元運算 (binary operation)。在不失一般性的原則下，吾人暫不考慮其他的運算）。一元運算每次祇作用於一個關係檔 (relational file or relation)，不會牽涉到資料的搬運問題。因此，我們的詢問處理策略評估時可以將它們忽略。join須作用於兩個關係檔，若兩個關係檔分別存在不同的地點，則必須將其中一個檔案搬到另一檔案的儲存地以便執行join 的運算。在此先介紹何謂選中率 (selectivity)。設 A和 B 爲兩個關係檔，其記錄數 (cardinality) 各爲 ||A|| 和 ||B||，若經過join 後，A 被選中的記錄數爲 ||A'||，B被選中的記錄數爲 ||B'||，則吾人稱 $\frac{\|A'\|}{\|A\|}$ 爲 B 對 A之選中率，$\frac{\|B'\|}{\|B\|}$ 爲 A對B 之選中率。在本節中，吾人將詢問處理策略分爲兩類；其一是假設已知運算選中率，第二類是假設未知運算選中率。

4.1 策略模型一：已知運算選中率

一般而言，吾人可以將上節所言之第一類詢問圖化成一個二元詢問樹 (binary query tree) T_Q，如圖五所示。

在圖五中，虛線所代表的部份皆屬於本地運算 (local operation)；換言之，運算所需的資料都存在執行運算的地方，沒有資料搬運的必要。最上面的三個運算則屬於未定地運算 (non-localized operation)；換言之，吾人尚未知這三個運算被執行的地點，而詢問處理的目標就是要對這些未定地的運算之執行地點作最適當的選擇，使得資料搬運量爲最小，而整個系統之效率爲最佳。

以下吾人將在已知運算選中率的條件下，定出一個數學模式，用來評估選擇詢問策略。這個模式是我們修改參考資料〔3〕所提出的模式。

令 $X_{ij} = \begin{cases} 1 & \text{if } f_i \text{ is an input to operation } \beta_j,\ i=1,2,\ldots,n, j=1,2,\ldots,m. \\ 0 & \text{otherwise} \end{cases}$

$Y_{ij} = \begin{cases} 1 & \text{if } f_i \text{ needs to be transmitted for operation } \beta_j, \\ & i=1,2,\ldots,n,\ j=1,2,\ldots,m. \\ 0 & \text{otherwise} \end{cases}$

$$Z_f = \begin{cases} 1 & \text{if final operation } \beta_f \text{ is not at the requesting site .} \\ 0 & \text{otherwise} \end{cases}$$

則全部費用 $C = \sum_{j=1}^{m} P_j \sum_{i=1}^{n} S(f_i)\, X_{ij}$ /* Processing cost */

$$+ \sum_{j=1}^{m} C_j \sum_{i=1}^{n} S(f_i)\, Y_{ij}$$ /* communication cost */

$$+ Z_f\, C_f\, S(f_n)$$ /* final moving cost */

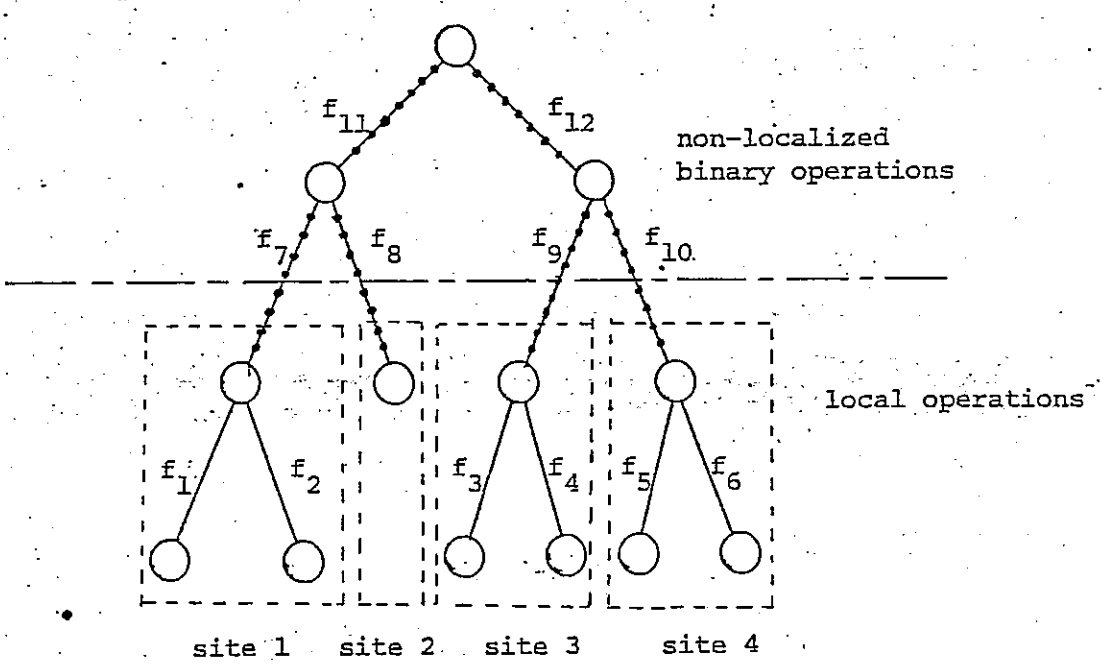
Where, P_j is unit data processing cost

C_j is unit data transmission cost

$S(f_i)$ is size of f_i (depends on selectivities)

決策：選擇一個俱有最小C 值的策略。

在這個策略模型中，$S(f_i)$ 的大小決定於每次操作的運算選中率。因爲我們已假設運算選中率是爲已知，故 C 值很容易可以算出來。然而預知選中率並不是一件很容易的事。畢竟選中率和資料分佈(data distribution)有密切的關係，而資料分佈亦可能隨著操作而變。因此，這個模式祇能粗略地做預測。目前一般作這方面研究的人都假設資料是均勻分佈 (uniform distribution)，這種假設與實際情形相差太大，不夠客觀。因此，如何預知選中率是這種策略模式的關鍵和難題。



○ : relational operation β

f_i : relational file

•••• : non-localized intermediate file

——— : local operation file

〔圖五〕 T_Q

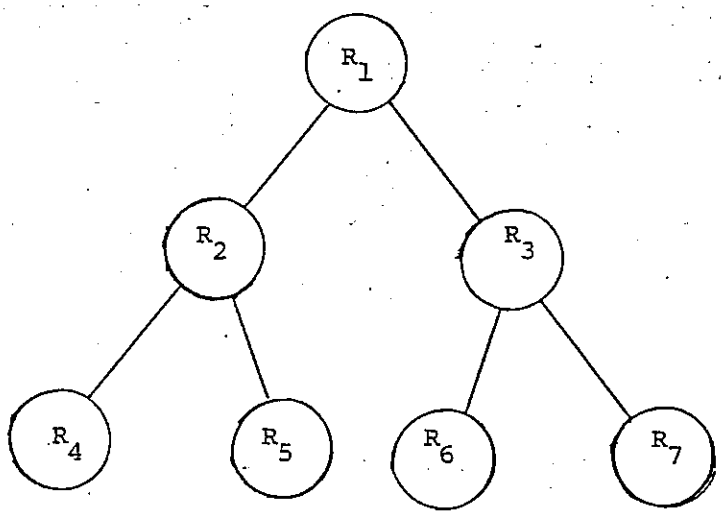
4.2策略模式二：未知運算選中率

在這個策略模式中，我們假設事先未知運算選中率，而採用semi-join [4,5]的觀念。簡言之，先利用semi-join 來縮減(reduce)有關的關係檔之記錄數 (cardinality; that is, no. of tuples) 即減少其不必要之記錄，最後才搬運經過縮小的關係檔去眞正的作join 。

設若 R, S 兩個關係檔（分別在甲地和乙地）欲對一結合值域 (join domain) 作 join，寫成 $R_{A1} \bowtie_{A2} S$；而 semi-join $R_{A1} \ltimes_{A2} S$ 意爲，先對乙地 S 關係檔作 Projection on A2, 得 S'[A2]，然後撥運 S'[A2] 至甲地，再對 R 關係檔作 (R.A1=S'.A2) 之 restriction, 得 R'且 $\|R'\| \leq \|R\|$（反之，$S_{A2} \ltimes_{A1} R$ 亦可得一 $\|R'\|$ 使得 $\|S'\| \leq \|S\|$）。就處理整個詢問而言，若有關的原始關係檔之集合爲 $\{R_1, R_2, \ldots, R_n\}$，我們欲先利用 Semi-joins 將其變成 $\{R_1', R_2', R_3', \ldots R_n'\}$，使得 $\|R_i'\| < \|R_i\|$, $i=1,2,3,\ldots n$. 最後才利用 $\{R_1', R_2', \ldots R_n'\}$ 做眞正的 joins 下面我們以一例子說明如何利用 semi-join 來縮減一詢問所參考到的關係檔。

$$Q : ((R_1 \bowtie R_2) \wedge (R_2 \bowtie R_4) \wedge (R_2 \bowtie R_5) \wedge (R_1 \bowtie R_3) \wedge (R_3 \bowtie R_6) \wedge (R_3 \bowtie R_7))$$

G_Q :



R_i : relational file

edge ; join operation

〔圖六〕 T_Q

圖六是屬第二類的詢問圖。假設該詢問所參考到的關係檔分別儲存在不同的地點。吾人作縮減的工作可分爲兩大步驟：

第一步是爲由下而上 (bottom-up) 作 semi-joins，其 semi-join Program 爲 $\sigma_{Q,1}$ 〔4〕。

$$\sigma_{Q,1} = \langle R_2 \ltimes R_4, R_2 \ltimes R_5, R_3 \ltimes R_7, R_1 \ltimes R_2, R_1 \ltimes R_3 \rangle$$

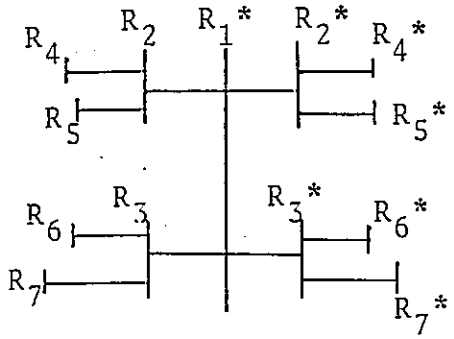
由左而右依序作semi-join, $R_2 \ltimes R_4$ 縮減 R_2 成 R_2', $R_2 \ltimes R_5$ 中之 R_2 係指被 $R_2 \ltimes R_4$ 縮減過的 R_2',於是 R_2 又再被縮減一次。同樣，$R_3 \ltimes R_6$ 與 $R_3 \ltimes R_7$ 縮減 R_3 二次。而 $R_1 \ltimes R_2$ 與 $R_1 \ltimes R_3$ 中 R_2, R_3 係指被縮減過的關係檔。經過這二個 semi-join,所得縮減過的 R_1 記爲 R_1^*, 此 R_1^* 中的記錄均是眞正作join時會被選中的記錄，換言之，吾人再也無法進一步縮減 R_1^*。求得 R_1^* 後，吾人再由 R_1^* 開始，作縮減的第二步驟。

第二步驟是爲由上而下 (top-down) 作semi-joins ，其 semi-join program 爲 $\sigma_{Q,2}$。

$$\sigma_{Q,2} = \langle R_2 \ltimes R_1, R_3 \ltimes R_1, R_4 \ltimes R_2, R_5 \ltimes R_2, R_6 \ltimes R_3, R_7 \ltimes R_3 \rangle$$

同理，由左而右依序作 semi-join, 吾人可一一得 $R_2^*, R_3^*, \ldots R_7^*$. 這時，此詢問所參考到的關係檔都已被完全地縮減 (fully reduced). 然後，我們再搬運被完全縮減過的關係去眞正做 join，以求得最後的答案。

綜合以上兩個步驟，很明顯地我們可以看出，若一樹狀詢問 (Tree query，參見下一節) 參考到 n 個不同地的關係檔，則由下而上的 semi-join program 包含了 n-1 個 semi-joins ；同樣，由上而下也包含了 n-1 個 semi-joins ，故整個 σ_Q 有 2(n-1)個semi-joins。在本例 $\|\sigma_Q\| = 2x(7-1) = 12$ 個 semi-joins 吾人可給一 semi-join program 之時間圖，如圖七所示，semi-join program 也是可以作局部平行處理。



圖七：semi-join program $\sigma_Q = \langle \sigma_{Q,1}, \sigma_{Q,2} \rangle$ 之時間圖

在此，吾人必須强調一點，雖然semi-join 可減少將來join時資料傳輸量，但semi-join 本身也要有資料傳輸的花費。於是，semi-joins的策略模式是否應完全採納，是值得商榷的問題。我們不在這裡申論這個問題，有興趣的讀者可以參考〔6〕。

5. 樹狀詢問之鑑別

在4.2節中所討論的策略模式，都假設詢問Q其詢問圖是爲樹狀，但是由一個使用者所寫的詢問，其對應的詢問圖有可能是爲非樹狀。對於大部分的情形，非樹狀的詢問圖可以轉化成樹狀。本節即是描述一方法來鑑定這種俱有非樹狀詢問圖的詢問是否存在一等效的樹狀詢問圖。

首先，吾人定義一些名詞。

詢問圖(Query Graph)：$G_Q(V_Q,E_Q)$ 表一詢問圖，V_Q爲節點所成的集合，$V_Q=\{R_i|R_i$ 爲Q 所參考到的關係檔$\}$；E_Q爲支邊(edge) 所成的集合，$E_Q=\{(R_i,R_j)|$ i≠j 且Q 中某子詢問同時參考到R_i 和 $R_j\}$ 。

樹狀詢問 (Tree Query)：一詢問（或是其等效詢問*）之詢問圖是爲樹狀。

設TQ爲所有樹狀詢問的集合。

環狀詢問(Cyclic Query)：　不屬於TQ的詢問皆爲環狀詢問。

設CQ爲所有環狀詢問的集合；由TQ與CQ之定義可看出 $TQ\cap CQ=\phi$，且 $TQ\cup CQ=\{Q(t,q)|t$ is target of Q, q is the qualification$\}$.

結合圖形(Join Graph)：$J_Q(V,E)$對應詢問 Q之詢問內涵q，$V=\{R_i.A|A$ 是 joining domain,且$R_i.A$爲q 所參考到$\}$；$E=\{(R_i.A,R_j.B)|i\neq j$ 且q 中某子詢問同時參考到$R_i.A$和$R_j.B\}$ 。

---

*等效詢問意指，設不同描述的詢問Q1與Q2，若Q1和Q2對任一資料庫系統詢問所得之結果都相同，稱Q1與Q2爲等效詢問。

因爲對一詢問 Q 之內涵 q，其 G_Q 是唯一決定*。同樣，Q1 也是唯一決定，所以 G_Q 與 J_Q 是一對一對應，是故接著下面所描述之詢問圖轉化法是合理的。

詢問圖轉化法：

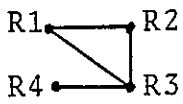
[step 1] 對詢問 Q，繪其詢問圖 G_Q。
若 G_Q 爲樹狀，則 $Q \in TQ$；結束。
[step 2] 若 G_Q 不爲樹狀，則繪其結合圖 J_Q。
[step 3] 由結合圖，找出其 minimal spanning trees J^c.
[step 4] 由 J^c，繪出 G^c(normalized query graph).
若 G^c 爲樹狀，則 $Q \in TQ$;
若 G^c 不爲樹狀，則 $Q \in CQ$。結束。

以下，我們以實例來說明之。

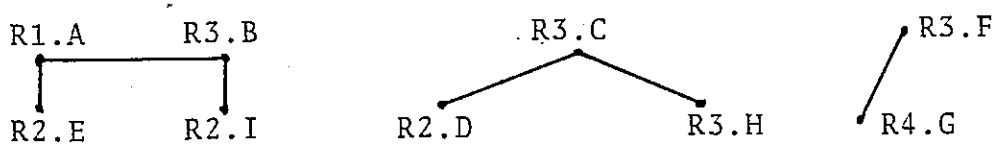
例一：

Q : ((R1.A=R3.B)^(R3.C=R2.D)^(R2.E=R1.A)^(R3.F=R4.G)
^(R3.C=R3.H)^(R3.B=R2.I))

G_Q:



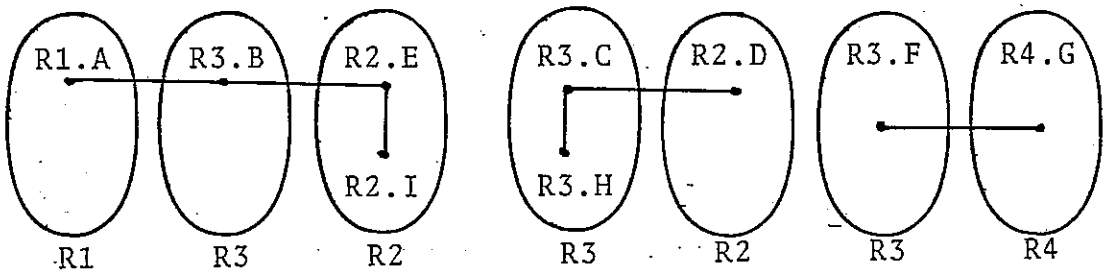
J_Q: 連結 $R_i.X$ 與 $R_j.Y$，若 $R_i.X=R_j.Y$ 存在於 Q。



---

* 前面幾節中，所謂對一詢問可能有不祇一個詢問圖是指由 transitivity closure 所引出的等效詢問。在此所謂的唯一決定，係指對一詢問之內涵 q 所繪出的 $G_Q(V_Q, E_Q)$ 由定義亦可看出是唯一決定的。

J^C: 對於每一J_Q，將R_i相同的各資料元放於同一組並相連之。再將各組以一綫連結起來。



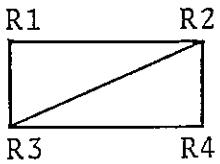
G^C: 將J^C內相連的各關係檔案連結起來。

由此，吾人得到相當於環形G_Q的樹形G^C.

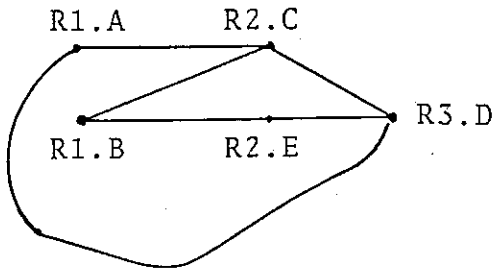
例二：

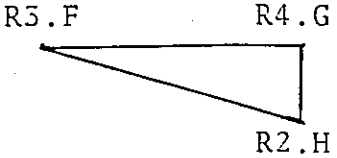
Q : (R1.A=R2.C)^(R2.C=R3.D)^(R2.E=R3.D)^(R2.E=R1.A)
(R1.A=R3.D)^(R3.F=R4.G)^(R4.G=R2.H)^(R3.F=R2.H)

G_Q:

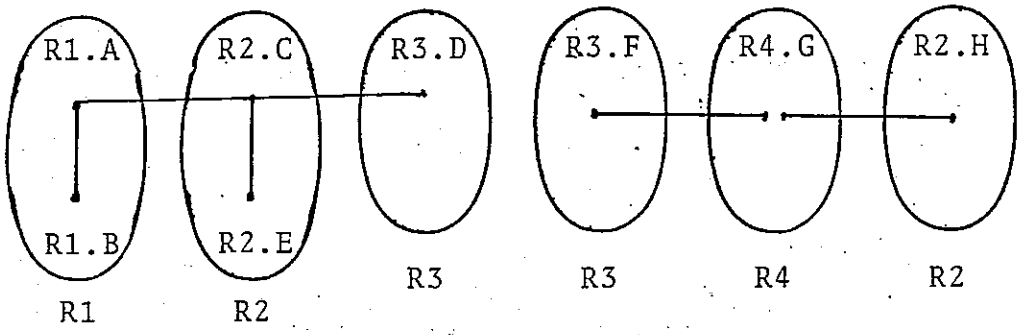


J_Q:

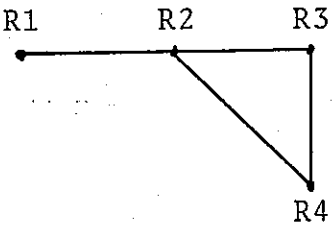




J^C:



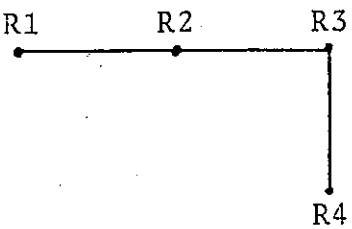
G^C:



Q 不是一個 tree query

若我們從 Q 中移去 R4.G = R2.H 的條件，
則

G^C 變爲



Q是一個 tree query

6. 結論

在這篇文章裡，我們先爲分散式資料庫系統詢問處理的問題設下定義，闡述一般詢問處理的過程，並提出兩個詢問處理的基本策略模式。爲了方便描述起見，我們做了許多假設，也省略了一些較複雜而實際的問題。在實際的DDBS中，至少有下列幾個問題有待進一步的研究。

問題一：複詢問處理 (Multiple Queries) [7]

通常一個 DDBS內，在同一時刻，會有兩個以上的詢問等待處理，這些詢問也許會用到部分相同的資料。因此，安排它們的處理順序必定是比單詢問的情形要複雜。

問題二：外在工作負載

在DDBS中，系統除了處理詢問以外，往往還要作其他的資料處理的工作，因此，資源分配(resource sharing)和程式執行安排 (process scheduling) 將使得計算機處理時間不得被忽略。換言之，資料傳輸時間不再是唯一主要的時間因素。Queueing Theory可用來解決部分的問題。

問題三：網路圖形 (Network Topology) 和多份資料 (Mutiple Copies)

當資料傳輸費用不是均勻（即必須考慮網路圖形）時，資料傳輸費用與其所傳輸的路徑有關，且不同地可能存有相同的資料；如此，都將使得計算傳輸費用和其最佳化更形複雜。Dynamic Programming 可用來幫助解決這個問題。

問題四：資料分佈和運算選中率

在前面所提的兩個基本策略模式裡，爲了使效率提高，或多或少，都有必要知道資料分佈的情形。然而資料庫內的資料是隨時在改變的，而且資料被運算選中的情形也很難預知。因此，如何用 Approximation Theory 來描述資料分佈情形，進而預測運算選中率，是一個很有意義的研究題目。

這四個問題都是很實際而難以克服的。在分散式資料庫詢問處理的研究題目是非常豐富的，我們希望在最近的將來能提出一些有實質貢獻的策略來解決DDBS的詢問處理。

感謝詞：

本文承張品燦先生加以整理並提供許多寶貴意見；陳瓊音小姐和方鳳梅小姐細心打字，特此申謝。

參考文獻：

1. Alan R. Hevner & S. Bing Yao, "Query Processing on A Distributed Database," Proceedings of the Second Berkerley Workshop on Distributed Data Management and Computer Networks, May, 1977.

2. Eugene Wong and Computer Corporation of America, "Retrieving Dispersed Data from SDD-1 : A System for Distributed Database," Proceedings of the Second Berkerley Workshop on Distributed Data Management and Computer Networks, May, 1977.

3. Wesley W. Chu and Paul Hurley, "A Model for Optimal Query Processing for Distriubted Data Bases," COMPCON 79, Spring.

4. Philip A. Bernstein and Dah-Ming W. Chiu, "Using Semi-joins to Slove Relational Queries," Journal of the ACM, Vol.28, No.1, January 1981, pp.25-40.

5. Dah-Ming W. Chiu, Philip A. Bernstein & Y. C. Ho, "Optimizing Chin Queries in A DDB System," TR-01-81, Center for Research in Computing Technology, Harward University, Cambridge, MA.

6. J. S. Ke and P. T. Chang, "An Approximation Technique for Determining Semi-Join Strategies in Distributed Database Systems," to be appeared.

7. Wojciech and Daniel Meyer, "A Multi-Query Approach to Distributed Processing in A Relational Distributed Database Management System," in "Distributed Data Bases" edited by C. Delobel and W. Litwin, North-Holland, 1980.