主從式資料庫系統中三種以位元表爲基礎的連結演算法*

Three Bitmap-based Join Algorithms for Client-server Database Systems

劉佳灝

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

陳耀輝

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

摘要

在主從式資料庫系統中，利用位元表能有效提昇快取資料的再使用性。本研究以此爲基礎，重新設計巢狀迴圈、排序合併及雜湊連結等三種演算法。這些方法讓用戶端與伺服器能根據快取資料的狀況，完成所需負責處理的部份，並藉由平行處理有效地降低連結運算的時間成本。我們建立了這些演算法的成本模式，並經由對實驗結果的分析發現：這些演算法可以有效利用快取資源，以降低連結運算處理成本，提昇系統效率。

關鍵字：主從式資料庫、快取、連結運算、位元表

Abstract

In client-server database systems, a bitmap of the cached data can be used to improve the usability of the cached data. This research devises new bitmap-based nested-loop, sort-merge, and hash join algorithms. The algorithms let the clients and server simultaneously process a query using the cached data. The parallelism introduced in the methods can effectively reduce the time costs of the join operation. In addition, we provide cost models for the algorithms in the paper. The experiment results show that the algorithms can use the cached data to reduce the join costs, and thus improve the performance of the system.

Keywords: Client-server database, Cache, Join, Bitmap

1. 緒論

資訊科技的進步改變了資訊系統的架構。在資源共享的理念下，經由網路技術的發展使得主從式(client/server)架構成爲資訊系統的主流。在資料庫的領域中，由於網路技術日漸成熟，資料庫藉由主從式架構將資源共享提昇至另一個更高的層次。主從式資料庫系統[Car91,Rou91,Del92,Del93]的運用與發展，亦成爲研究上的一個重要範疇。

在主從式資料庫中，資料庫存放於伺服器端(server)，而使用者在用戶端(client)經由網路對伺服器上的資料庫發出查詢(query)需求，以取得所需的資料。於此架構下，我們可以利用快取(cache)的技術，來有效地降低因網路傳輸所造成的瓶頸，以提昇系統的效能[Kel96]。這一部份的相關研究，在過去是以限制式爲基礎(predicate-base)的快取技術[Kel96]、資料傳送(data-shipping)[Fra96]與查詢傳送(query-shipping)[Fra96]爲主要的處理方式。

就查詢傳送的方式而言，用戶端接收到使用者的查詢命令時，直接將查詢命令傳送到伺服器端，伺服器於處理完成後再將結果回應給用戶端。在此種處理模式下，對於查詢的處理動作完全由伺服器負責，因此，於伺服器上將會造成極大的工作負載。在資料傳送的處理模式下，用戶端將查詢處理命令傳送至伺服器，伺服器將所需的相關資料傳送回用戶端，而由用戶端自行處理。於此種方式下，若查詢只要擷取某一小部份資料，伺服器仍然需傳送所有相關的資料，對於傳輸成本而言，將會是一沈重的負擔。雖然以限制式爲基礎的資料庫系統在過去是被廣爲採用的方式，但由於以此種模式來實現快取功能時，必須經過複雜度極高的限制式比對，使得相關應用的表現成效不盡理想。

爲了改善以上處理方式的缺點及效率，近來學者又提出了一種以位元表爲基礎的快取及查詢技術[Wen97] 並與查詢傳送及資料傳送在處理成本上作一詳細的評估與分析，以驗證以位元表爲基礎的快取及查詢技術的可行性與優越性。經由實驗的結果顯示：以位元表爲基礎的快取及查詢技術經由用戶端有效地再使用(reuse)既有的資料，更能夠改善因網路傳輸所造成的瓶頸及減輕伺服器的工作負載，而提昇系統的效能。在以位元表爲基礎的主從式架構下，本研究將連結運算演算法重新設計。藉由快取功能，讓用戶端能如同伺服器一般具有處理連結運算的能力。經由主從之間適當的分工與平行處理的方式，來降低處理連結運算的時間成本。

*本研究之部份經費由國家科學委員會專題計畫 NSC87-2213-E-023-001 補助。

為了進一步對『以位元表為基礎的快取及查詢技術』研究與改善，本研究將針對此一系統架構在處理連結運算的方法上作更進一步的評估。對於以位元表為基礎的系統重新發展出『巢狀迴圈連結』(Nested Loop Join)、『排序合併連結』(Sort Merge Join)與『雜湊連結』(Hash Join)連結運算演算法，並對這三種連結方式在處理成本上作一評估，並分析演算法對系統效率所造成的影響。

本篇論文的架構為: 第二節簡介位元表的架構及特性。第三節中將舉例說明本研究中所設計的三種連結運算的演算法。在第四中將以雜湊連結為例，描述演算法的成本模式。第五節為系統模擬的結果與分析。第六節則是本研究的結論。

2. 位元表的架構及特性

2.1 位元表的架構

在以位元表為基礎的系統下，資料庫中的每一關聯(relation)都有一相對應的位元表(bitmap table)，於位元表中以一個位元來表示一屬性資料是否存在於用戶端的快取區中。若一屬性資料已存在於用戶端的快取區中，則該屬性的位元值為1；反之，則位元值為0。

假設關聯R的資料存在於用戶端的狀況為圖1(a)所示，圖1(b)表示對應於(a)的位元表。其中值組編號(tuple id)為"001"的"姓名"屬性資料存在用戶端，因此，在關聯R的位元表中，值組編號為"001"的該屬性位元值為1；而該值組的"部門"屬性資料並未存在於用戶端，所以，與其相對應的位元值為0。就值組編號為"002"的值組而言，該值組的"姓名"屬性資料未存在於用戶端，所以其位元值為0；而"部門"屬性資料已存在於用戶端，故其位元值為1。

值組編號	姓名	部門
001	李小寶	
002		資訊部門

(a)用戶端快取區中的關聯 R

值組編號	姓名	部門
001	1	0
002	0	1

(b)對應於關聯 R 的位元表

圖 1.用戶端快取區中關聯 R 與其位元表之對應關係

2.2 主從式架構及其基本運作模式

在以位元表為基礎的查詢處理系統中，由於用戶端與伺服器端都存有對應於每個關聯的位元表，透過對位元表的查詢，可迅速地判斷出有哪些資料存在於快取區中。

如圖 2 所示，在用戶端接收到使用者查詢(query)後，須先將此查詢傳至伺服器，然後檢查與該查詢相關的位元表，以決定出用戶端可自行處理的部份並加以處理。若用戶端未能擁有全部與查詢有關的關聯資料，則用戶端處理的結果將只是整個查詢結果的一部份，此時必須等待伺服器將另一部份的結果傳送至用戶端，整合之後的結果才是最終查詢結果。然而整個查詢處理的結果也將會被儲存於用戶端的快取區中，成為用戶端的資源，之後用戶端可以隨時再使用這些資源。如果用戶端已擁有所有查詢所需的資料，則用戶端可自行完成整個查詢處理，不需再透過伺服器的服務。

伺服器收到由用戶端所傳送過來的查詢命令後，會去檢查相關的位元表，以找出那些用戶端因缺乏資料而無法完成的部份，伺服器經過處理後再將此一部份的結果傳送給用戶端，用戶端整合出完整的查詢處理結果，輸出給使用者或應用程式。在查詢處理完成時，用戶端與伺服器亦必須更新相關的位元表。

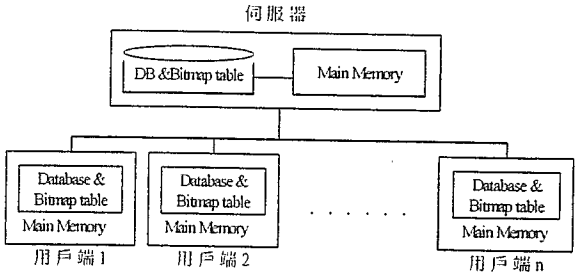


圖 2 系統架構圖

2.3 以位元表為基礎的特性

對於伺服器查詢處理的結果，其中不存在於用戶端的部份將會被寫入到用戶端的快取區中。因此，隨著查詢次數的增加，伺服器所要傳送的資料量將隨之減少，對於資料傳輸成本的降低有顯著的改善。若用戶端擁有所需的所有資料時，用戶端即可獨自完成查詢處理動作，不但能將傳輸成本降至最低，並可減輕伺服器的工作負載。綜合上述特點，採用以位元表為基礎的方式，在主從式架構的環境下來實現快取功能，將可以歸納出以下幾項優點：快取資料再使用率的提昇、平行處理的特性、降低傳輸成本、減低伺服器的工作負載及提昇查詢處理的整體效率。

3.以位元表為基礎的連結演算法

巢狀迴圈連結 (Nested Loop Join) [Val84,Gra93,Lu94]是利用雙重迴圈的方式，來完成連結值組的比對動作。這是一種基本的連結處理方式，但其成本也相對較高。藉由排序及合併的技巧可以減少連結過程中值組比對的成本，此種方式我們稱之為排序合併連結(Sort Merge Join) [Eva96,Lu94]。另一種常見的處理方式為雜湊連結 (Hash Join) [Bra84,Dew84,Omi91]，在此一方式中，一關聯藉由雜湊函數將值組分置於儲存區(bucket)中，另一關聯值組僅與所對應之儲存區中的值組進行比對。此一方式更能夠降低處理成本，並提昇效率。

3.1 巢狀迴圈連結演算法

巢狀迴圈連結是利用兩關聯中的值組，根據連結屬性值一一進行比對後再行值組的合併。我們以

值組編號及連結屬性分別對兩關聯建立輔助表，而連結處理的比對動作將作用在所建立的輔助表上。

值組編號	姓名	部門	職等
001	李小寶	會計部	7
002	徐嘉伶	資訊部	5
003	張忠衛	工程部	4
004	陳明道	資訊部	1

(a) 關聯 R

值組編號	職等	加給
101	1	1000
102	7	7000
103	4	4000
104	9	9000

(b) 關聯 S

圖 3. 關聯 R 及關聯 S

值組編號	姓名	部門	職等
001	李小寶	會計部	7
002			
003	張忠衛	工程部	4
004	陳明道		1

(a1) 快取區中的關聯 R

值組編號	職等	加給
101	1	1000
102	7	7000
103		
104		9000

(b1) 快取區中的關聯 S

值組編號	姓名	部門	職等
001	1	1	1
002	0	0	0
003	1	1	1
004	1	0	1

(a2) 關聯 R 的位元表

值組編號	職等	加給
101	1	1
102	1	1
103	0	0
104	0	1

(b2) 關聯 S 的位元表

圖 4. 用戶端快取區中存在之關聯 R 及關聯 S

值組編號	職等
001	7
003	4

(a)外部關聯輔助表

值組編號	職等
101	1
102	7

(b)內部關聯輔助表

圖 5. 用戶端執行巢狀迴圈連結演算法示意圖

假設關聯 R 與 S 的狀況如圖 3 所示。圖 4(a1)、(b1)表示用戶端擁有兩關聯資料情形，與其對應的位元表如圖 4(a2)、(b2)所示。用戶端在接收到使用者查詢命令後，則將此查詢命令傳送至伺服器，並經由對關聯 R 位元表的判斷後，發現關聯 R 中僅有值組編號爲"001"及"003"兩值組的所有屬性皆存在於快取區中，則取出值組編號及連結屬性值(職等)建立關聯 R 的輔助表，如圖 5(a)所示。同理，關聯 S 中的"101"及"102"兩值組亦會被記錄，如圖 5(b)所示。經由巢狀迴圈方式，將只進行(001,101)、(001,102)、(003,101)及(003,102)等四組比對動作。若兩值組符合連結條件，則自快取區中取出值組資料，並將其合併後送至輸出緩衝區中。用戶端處理結果爲：

姓名	部門	職等	加給
李小寶	會計部	7	7000

伺服器於接收到用戶端所傳送過來的查詢後，經由對位元表的檢查，發現外部關聯 R 中的"001"及"003"兩值組的所有屬性資料都存在快取區中，此二值組的狀態旗標值將設爲 1 ；相反的，"002"與"004"則爲 0 。以值組編號、連結屬性值及狀態旗標值建立外部關聯輔助表。對於關聯 S 而言，將不完全存在於快取區的"103"及"104"值組記錄在輔助表 0 中，而存在快取區中的"101"及"102"則記錄在輔助表 1 中。如圖 6 所示。

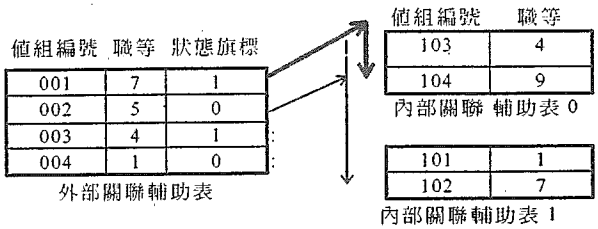


圖 6. 伺服器端執行巢狀迴圈連結示意圖

對於外部關聯輔助表中狀態旗標值爲 1 的值組，僅與內部關聯輔助表 0 的部份進行比對。這是由於其與輔助表 1 比對的部份，已於用戶端完成(圖 5)。因此，此一架構可以有效達成主從間的分工而避免重複。而狀態旗標值爲 0 的值組(如"002")，需與內部關聯輔助表 0 及輔助表 1(亦即整個內部關聯)進行比對。因此，伺服器將處理(001,103)、(001,104)、(002,103)、(002,104)、(002,101)、(002,102)、(003,103)、(003,104)、(004,103)、(004,104)、(004,101)及(004,102)等 12 組用戶端無法處理的比對。對於連結屬性值的比對，符合連結條件的值組，我們將其所屬的關聯分頁載入，在取得兩值組的資料後將其合併。以此模式運作，可避免重複執行用戶端已經處理的部份。如此，伺服器的工作負載將可藉由快取技術及用戶端處理連結運算的能力而獲得改善。伺服器進行巢狀迴圈連結處理的結果爲：

姓名	部門	職等	加給
張忠衛	工程部	4	4000
陳明道	資訊部	1	1000

對於處理的結果，伺服器透過位元表的檢查可以找出其中尚未儲存於快取區的屬性資料，並加以傳送。如此，可以減少傳輸的成本。因此，經由伺服器端對結果的檢查後，兩值組中實際傳送的資料分別爲："**4000**" 與 "資訊部" 。 於傳送時，伺服器會賦予各資料項一個索引，以便用戶端能正確地將屬性資料寫入快取區中。用戶端接受到伺服器傳送過來的結果後，自快取區取出與伺服器執行結果的相關屬性資料，整合後輸出最終連結運算結果如下。最後，用戶端與伺服器端都必須將相關的屬性位元值修改爲 1 。

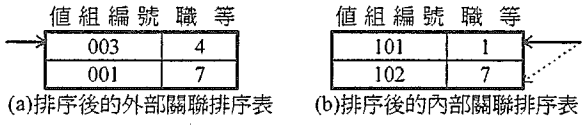
姓名	部門	職等	加給
李小寶	會計部	7	7000
張忠衛	工程部	4	4000
陳明道	資訊部	1	1000

3.2 排序合併連結演算法

將值組根據連結屬性排序後再將其合併，可以減少連結處理過程中的比對次數。於本研究所設計的排序合併連結演算法中，用戶端僅對於關聯 R 與 S 中存在快取區的值組建立排序表；而於伺服器端上主要以用戶端所無法處理的部份爲工作目標，因此我們將其中的一關聯區分成存在用戶端與不存在用戶端兩個部份。以期在合併的過程中，能夠盡量避免重複處理用戶端已執行過的比較動作。

以 3.1 節中相同的假設爲例(圖 4)。用戶端經由對位元表的判斷後，將關聯 R 可處理的部份，以"值組編號"及"連結屬性"建立外部關聯排序表，將值組編號爲"001"及"003"的值組編號及連結屬性值記錄於排序表中，經由排序表的建立可以降低排序時資料移動的成本；同理，建立內部關聯 S 的排序表。兩排序表根據連結屬性排序後，結果如圖 7 。合併過程中，對於符合連結條件的值組自快取中取出，並加以連結。用戶端處理結果爲：

姓名	部門	職等	加給
李小寶	會計部	7	7000



(a)排序後的外部關聯排序表　(b)排序後的內部關聯排序表

圖 7. 用戶端之排序合併連結處理示意圖

伺服器端分別建立外部關聯排序表與兩個內部關聯排序表，內部關聯排序表 0 中記錄不存在快取區的值組，存在快取區的則記錄在內部關聯排序表 1 。對連結屬性進行排序後其結果如圖 8 所示。在連結的過程中，外部排序表需先檢查狀態位元，若其值爲 0 ，表該值組不存在於快取區中，此時比對的動作需同時考慮內部關聯排序表 0 與內部關聯排序表 1 ；若狀態位元值爲 1 ，僅需與內部關聯排序表 0 進行合併的比對。合併比對的動作爲：連結屬性質若符合連結條件，則載入關聯分頁並將其連結；否則，其值較小者將指到下一筆記錄。

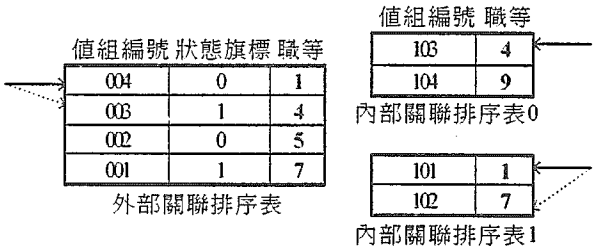


圖8. 伺服器端之排序合併連結示意圖

就外部關聯排序表而言，指標往下移動的兩種狀況爲：(1)若狀態旗標值爲 1 ，且內部關聯排序表 0 之連結屬性值較大時，(2)狀態旗標值爲 0 ，且兩內部關聯排序表之連結屬性值皆比外部排序表的連結屬性值大時。將內部關聯 S 之排序表切割成兩部份，用意是在減少伺服器部份不必要(用戶端會處理)的比對，以縮短處理時間。伺服器端處理結果爲：

姓名	部門	職等	加給
陳明道	資訊部	1	1000
張忠衛	工程部	4	4000

同樣的，伺服器僅傳送不存在於快取區的部份屬性資料。經由用戶端彙整結果後，將可以得到與巢狀迴圈連結相同的結果。最後，用戶端與伺服器端都必須將相關的屬性位元值修改爲 1 。

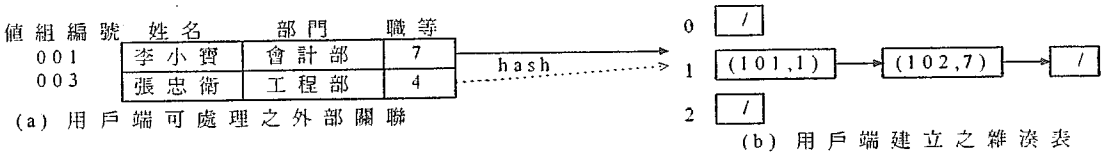
3.3 雜湊連結演算法

藉由雜湊表的方式，可以縮小連結中比對的範圍，減少不必要的比對。於此一演算法中，用戶端僅對快取區中的值組建立雜湊表並進行比對的動作；於伺服器上爲了避免重複用戶端已處理的部份，因此我們將雜湊表分成兩個部份，分別記錄存在快取區與不存在快取區的值組，使另一關聯在與雜湊表進行比對時能夠只處理用戶端未能完成的部份。

使用 3.1 節中的例子來以說明，用戶端利用對內部關聯 R 位元表的檢查，發現其中可以處理的值組爲編號"101"及"102"兩值組。將此二值組的連結屬性值經由雜湊函數計算後(假設雜湊函數爲 mod 3)，於進入點(entry)所指到的儲存區(bucket)中記錄該值組編號及其連結屬性值，建立用戶端雜湊表，如圖 9(b)所示，此二值組恰好都記錄於儲存區 1 中。於外部關聯中，用戶端可處理值組編號爲"001"及"003"兩值組。此二值組的連結屬性值經由雜湊函數計算後，與其所指到儲存區中的記錄進行比對動作，如圖 9(a)所示。編號爲"001"的值組於此過程中發現可以與編號"102"的值組連結，自快取區中取出該值組資料，並於連結後送至輸出緩衝區中。用戶端此一部份處理結果爲：

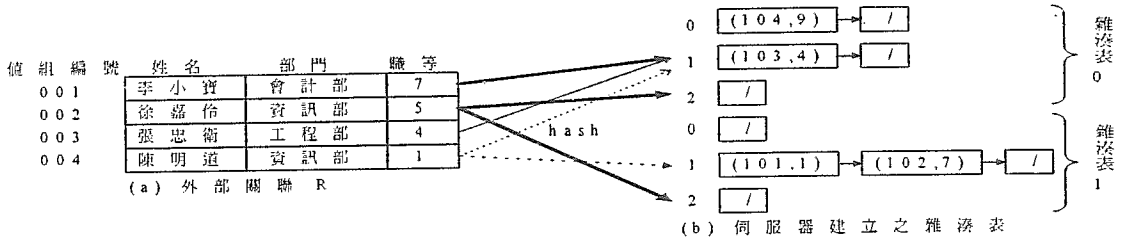
姓名	部門	職等	加給
李小寶	會計部	7	7000

伺服器端對內部關聯 S 建立雜湊表，將其中不存在於快取區的"103"及"104"兩值組編號及其連結屬性值記錄於雜湊表 0 中，而存在於快取區的"101"及"102"則記錄於雜湊表 1 中，如圖 10(b)所示。伺服器依序讀取外部關聯分頁，並進行連結處理動作。如圖 10 所示，伺服器讀取"001" 值組，經由位元表的判斷發現該值組已存在快取區，此值組經由雜湊函數運算後，僅與雜湊表 0 的部份進行連結比對，因爲此值組與雜湊表 1 進行比對的部份已由用戶端執行(圖 9)，伺服器不需再重複處理，如此將可於主從間獲得有效的分工。就編號"002"的值組而言，由於此值組未存在於快取區，因此，該值組經過雜湊函數運算後，需與雜湊表 0 及雜湊表 1 進行比對。



(a) 用戶端可處理之外部關聯　　(b) 用戶端建立之雜湊表

圖 9. 用戶端執行雜湊連結示意圖



(a) 外部關聯 R　　(b) 伺服器建立之雜湊表

圖 10. 伺服器端執行雜湊連結示意圖

若比對結果發現兩值組符合連結條件，則將雜湊表所記錄值組的關聯分頁載入，並進行連結處理。對於伺服器端處理結果的眾屬性中，伺服器只傳送其中不存在於快取區中的屬性資料。用戶端彙整結果後，將可得到連結處理的最終結果。最後，用戶端與伺服器端都必須將相關的屬性位元值修改爲1。伺服器執行雜湊連結之結果爲：

姓名	部門	職等	加給
張忠衛	工程部	4	4000
陳明道	資訊部	1	1000

4.成本模式

礙於篇幅的限制，無法詳述整個成本模式，於此一部份我們將僅對雜湊連結的成本模式作介紹，對於巢狀迴圈與排序合併連結成本模式之說明請參閱[Liu97]。

4.1 研究環境的假設

本研究中分別對用戶端與伺服器端的環境作了基本的假設，除了減低複雜度外，也力求整體的環境假設能屬於合理的範圍內。研究中假設用戶端的資料庫系統爲一主記憶體資料庫(Main Memory Database)系統，快取區亦即存在於主記憶體中。每一關聯都有其相對應的位元表，且儲存於主記憶體中。對於伺服器端而言，假設伺服器端的資料庫存在於磁碟上，透過分頁(page)的方式讀取，且每一個關聯一次僅能載入一個分頁。伺服器擁有與用戶端相同資料的位元表，其存取亦採用分頁的方式。另外，對於雜湊連結演算法中，假設儲存區(bucket)的數量爲16。

4.2 雜湊連結成本模式

在以位元表爲基礎的系統架構下，雜湊連結運算之成本模式包含了連結處理(t[process])與後續資料處理動作(t[data])兩階段的成本。整個雜湊連結運算的成本(t[join_H])將等於兩者的總和(變數說明請參照表1)。雜湊連結演算法的成本模式表示如下：

t[join_H]=t[process] + t[data]
t[process]=max{t[client_H],t[server_H]}
t[client_H]=t[creat_hashtab_c]+t[process_c,R]+t[mark_c]
t[creat_hashtab_c]=t[check_bitmap,S] +t[set_pairtobk]
t[process_c,R]=t[check_bitmap,R] +t[hash_join_c]
t[hash_join_c]=t[join_c]+t[write_c]
t[server_H]=t[query_flow]+t[read_bitmap,RS]
+t[creat_hashtab_s] +t[process_s,R] +t[tran]
t[creat_hashtab_s] = t[read_relation,S] + t[check_bitmap,S]
+t[set_pairtobk0] + t[set_pairtobk1]
t[process_s,R]=t[read_relation,R]+t[check_bitmap,R]
+t[hash_join_R]+t[read_bkpage]+t[check]+t[jn_out]
t[data] = t[data_write] + t[client_access] + t[output]
+t[update_bitmap]
t[data_write]= (jn_out_size/attr_size)*t[write_attr]
t[client_access]=(result_attr_num-tran_attr_num
-client_res_attr_num) * t[load_attr]
t[update_bitmap]=
max{ t[update_bitmap_c] ,t[update_bitmap_s]}

變數	說明
t[join_H]	雜湊連結演算法的總時間成本
t[process]	連結處理的時間成本
t[data]	資料處理的時間成本
t[client_H]	用戶端連結處理的時間成本
t[server_H]	伺服器連結處理的時間成本
t[creat_hashtab_c]	用戶端建立雜湊表的時間成本
t[creat_hashtab_s]	伺服器建立雜湊表的時間成本
t[process_c,R]	用戶端處理關聯 R 的時間成本
t[set_pairtobk]	記錄值組編號及屬性值的成本
t[hash_join_c]	用戶端進行雜湊計算與比對之成本
t[query_flow]	伺服器接收查詢命令所需的成本
t[read_bitmap,RS]	伺服器讀取兩位元表的時間成本
t[tran]	伺服器傳送資料至用戶端之成本
t[data_write]	將資料寫入快取區所需的時間成本
t[client_access]	自快取區擷取相關資料的時間成本
t[output]	將結果輸出所需的時間成本
t[update_bitmap]	更新位元表所需的時間成本

表 1. 成本模式使用變數之說明

5. 實驗與分析

本研究的實驗共分爲兩個部份，以成本模式爲基礎來進行實驗的模擬。實驗中所採用的重要參數如表2所示。

參數名稱	參數預設值
attr_size	8 (bytes)
I/O_speed	40M (bytes/sec)
network_speed	10M (bits/sec)
page_size	4096 (bytes)
commit_size	50 (bytes)
index_size	1 (byte)
t_attr	64 (個)
z_R	10000 (個)
z_S	10000 (個)
t[cpu]	20 (ns/instruction)
t[latency]	5.6(ms)
t[seek]	9.5(ms)

表 2. 參數預設值

5.1 實驗一

本實驗主要研究當連結因子(join selectivity factor)等於0.000001及0.01的環境下，用戶端所擁有的資料的百分比與連結運算處理在時間成本上的關係。經由對圖形的分析發現：連結運算的時間成本隨著用戶端存在資料比例的提昇而減小，表示用戶端使用快取區中既有的資料，分擔部份連結運算工作，能有效地提昇整體處理連結運算的效率。

由於本研究假設用戶端的資料庫爲一主記憶體資料庫，因此當用戶端的快取區中已擁有所有連結運算所需的資料時，用戶端將可完全自行處理，且運算處理的過程完全於主記憶體中完成，不需負擔磁碟輸出入(disk I/O)的成本，同時亦可免除透過伺服器遠端服務所需的成本。因此於圖11中，當用戶端存在資料比例從0.9變化到1(完全擁有)時，圖形呈現明顯的下降曲線。當連結因子較大時(0.01)，載入關聯分頁的磁碟輸出入成本，將對連結運算的效率造成極大的負擔，但隨著用戶端快取資料量的增加，用戶端所能處理的能力亦隨著增加，對於連結效率能有顯著的改善。三種演算法中又以雜湊連結的方式爲最佳，而巢狀迴圈與排序合併連結的曲線幾乎重疊，這是由於此兩種演算法中對於關聯分頁的讀取數相同，在連結因子較大時，磁碟輸出入的時間將成爲決定連結處理所需成本的關鍵因素。其結果如圖12所示。

5.2 實驗二

實驗二，將用戶端存在資料比例(f)定爲20%的狀況下，連結因子(於0.000001到0.01之範圍內)與連結運算處理在時間成本上的關係。當f=20%時，連結運算處理在時間成本上之實驗結果如圖13所示。由於本研究所設計的演算法中，在被檢查的兩值組符合連結條件時，於伺服器端將會進行關聯分頁的讀取動作，以便將兩值組合併。因此，在用戶端存在資料量固定的情況下，若連結因子較小時，可表現出極爲理想的執行效率；若連結因子相當大時，將會對處理成本造成極大的負擔。在巢狀迴圈及排序合併連結演算法中，當兩值組可以合併時，必須分別載入兩值組的關聯分頁；而於雜湊連結演算法中，由於關聯R的值組已存在主記憶體中，因此在可以連結的情況下，僅需載入雜湊表所記錄值組的一個關聯分頁。因此，當連結因子較大時，其在連結處理上的時間成本將會與巢狀迴圈及排序合併連結產生相當大的差距。

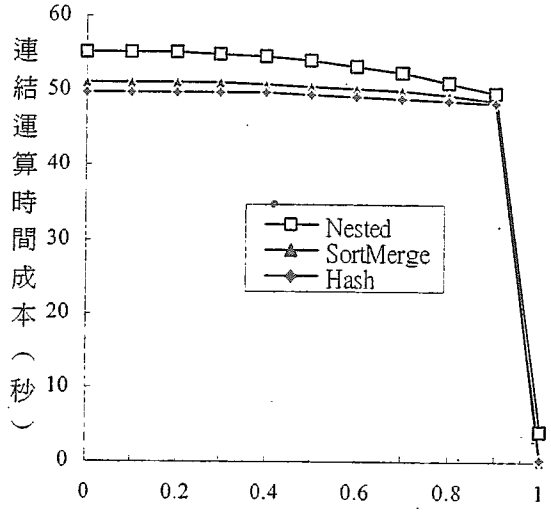


圖 11. 用戶端存在資料比率與連結處理成本之關係 (連結因子=0.000001)

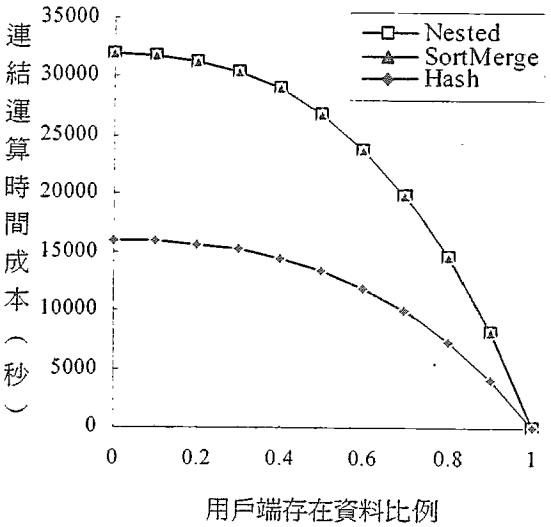


圖 12. 用戶端存在資料比率與連結處理成本之關係 (連結因子=0.01)

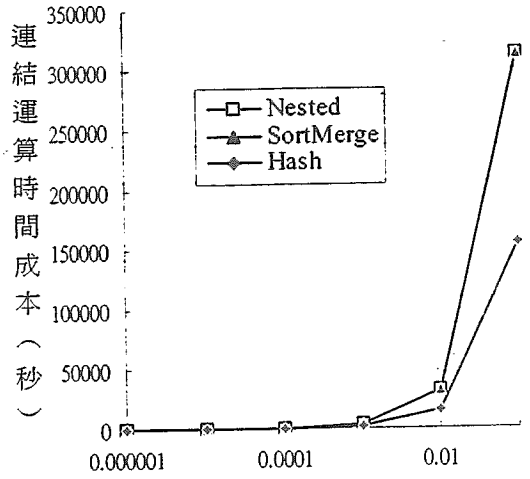


圖 13.連結因子與連結處理成本之關係
(用戶端資料存在比率=0.2)

6.結論

在以位元表爲基礎的主從式架構資料庫系統中，利用位元表的方式來實現快取資料的功能，更能落實對快取資料的再使用性，有效地減少資料的傳送量，降低傳送的時間成本。本研究根據位元表設計的架構及特性，重新對巢狀迴圈、排序合併及雜湊連結等三種演算法加以設計，以期用戶端與伺服器端能夠根據快取資料狀況明確分工，藉由平行處理的方式減低連結處理的時間成本。經由實驗，本研究對各演算法做了基本的評估與分析。藉由本研究，亦驗證了巢狀迴圈、排序合併及雜湊等連結方式在以位元表爲基礎的主從式資料庫系統中的可行性。本研究主要的後續發展工作爲：

1.在此架構下，深入探討用戶端與伺服器端之間工作負載平衡的問題並尋求解決方案。
2.研究在Multi-processor的環境下的平行處理方式及對有關Multi-join的運算處理。

參考文獻

[Bra84] K. Bratbergesngen, "Hashing Methods and Relational Algebra Operations". In Proc. Int'l Conference on Very Large Data Bases. VLDB Endowment, pp.323-333, 1984.

[Car91] M. Carey, M. Franklin, M. Livny, and E. Shekita, "Data Caching Tradeoff in Client-Server DBMS Architecture," In Proc. ACM SIGMOD Int'l Conference on Management of Data, pp.357-366, June. 1991.

[Del92] A. Delis and N. Roussopoulos, "Performance and Scalability of Client-Server Database Aechitecture," In Proc. 18th Int'l Conf. on Very Large Data Bases, pp.610-623, 1992.

[Del93] A. Delis and N. Roussopoulos, "Performance Comparision of Three Modern DBMS Architecture," IEEE Transactions on Software Engineering, 19(2):120-138, Feb. 1993.

[Dew84] D.J. DeWitt, R. Katz, F. Olken, L. Shapiro, M. Stonebraker and D. Wood, "Implementation Techniques for Main Memory Database Systems". In Proc. ACM SIGMOD Int'l Conference, ACM, New York, pp.1-8, Jan. 1984.

[Eva96] P. Evan Harris and Kotagiri Ramamohanarao, "Join algorithm costs revisited," The VLDB Journal, (5):64-84, 1996.

[Fra96] M. Franklin, B.T. Jonsson and D. Kossmann "Performance Tradeoffs for Client-Server Query Processing." In Proc. ACM SIGMOD Int'l Conf. on Management of Data, pp.149-160, 1996.

[Gra93] G. Graefe, "Query Evaluation Techniques for Large Databases". ACM Computing Surveys, 25(2):73-170, June 1993.

[Kel96] A. Keller and J. Basu, "A Predicate-Base Caching Scheme for Client-Server Database Architecturees", The VLDB Journal, (5):35-47, Jan. 1996.

[Liu97] 劉佳灝，"以位元表爲基礎之主從架構下的連結演算法設計與分析"，技術報告，國立屏東科技大學， 資管所， 1997.

[Lu 94] H. Lu, B. C. Ooi, K. L. Yan, Query Processing in Parallel Relational Database Systems , IEEE Computer society, 1994.

[Omi91] E. Omitecinski, "Performance analysis of a load balancing relational hash-join algorithm for a shared-memory multiprocessor," In Proc. Int'l Conference on Very Large Data Bases (Barcelona, Spain). VLDB Endowment, 375-385, 1991.

[Rou91] N. Roussopoulos, and A. Delis, "Modern Client-Server DBMS Architectures", ACM SIGMOD Record 20(3):52-61, 1991.

[Val84] P. Valduriez and G. Gardarin , "Join and Semijoin Algorithms for a Multiprocessor Database Machine" , ACM Trans. Database Systems, 9(1):133-161 , Mar. 1984.

[Wen97] 溫盛威，陳耀輝，"主從架構下的快取及查詢技術"，一九九七分散式系統技術及應用研討會論文集， pp.263-271, 1997.