適用於物件導向資料庫中類別繼承架構的索引法之研究

李建億
國立台南師範學院
資訊教育研究所
leeci@ipx.ntntu.edu.tw

陳凰姬
國立台南師範學院
資訊教育研究所

摘　要

由於在物件導向資料庫（Object-Oriented Database）中，利用索引技術將是一種適切的方法。然而如何來設計索引方法，將是目前於類別繼承架構中一個值得研究的議題。對於使用者的查詢需求大致可分爲兩類：一是單一類別查詢，一是類別繼承架構查詢。目前已提出的方法中，如 CH-tree、CG-tree 和 hcC-tree 索引法，都不能呈現出類別之間的相關性，而 H-tree 索引法則因巢狀指標建置複雜且不能全面呈現整個類別架構，也就是說，目前的方法都無法有效的提供使用者的各種查詢需求。所以，在本論文中，我們將提出類別編碼索引法（Class-coding Tre ），其主要的構想是將類別採用 Huffman code 的方式予以編碼，如此，不但可以清楚得知類別之間的關係及其在整個類別架構的位置，而且可以快速又有效率地回答上述兩種查詢需求。從實驗分析的結果比較，驗證我們所提出的類別編碼索引法在各種查詢問題上皆能有較好的查詢效率。

1. 緒論

在新一代的電腦應用上，例如電腦輔助設計和製作（CAD/CAM）、多媒體資料庫（MMDB）和軟體發展環境（SDEs）等等，都需要強而有力的技術去產生和處理大量的資料，但是在一般以記錄（record）爲單位的關聯式資料模組（relational data model）中，卻無法透過模組直接對複雜的資料做詳盡的描述，而且資料和資料之間複雜的關係，也無法利用關聯式資料模組來明確定義。例如：學校是由學生、教職員以及一些硬體設施所組成（aggregation）、而學校中的教職員又可細分成行政人員和教師，他們都繼承（inherit）教職員的共同屬性，例如：教職員編號、服務單位等等；這些資料若只運用關聯式資料模組之關聯（relation）來解釋將變得複雜而難懂。物件導向資料模組（Object-oriented data model）不但清楚地描述了資料的特性，甚至可以清楚地解釋資料和資料間複雜的關係。在物件導向資料模組中，任何真實世界的實體（entity）都可以用物件來模組（modeling），每一個物件都有一個唯一的辨識碼（object identifier,OID）和若干個屬性（attribute），而屬性就是代表該物件的狀態，可以單純只是字串、整數或布林函數（string、integer、boolean），也可以是一組屬性的類別（a set of attributes）。擁有相同屬性的物件就會被群組成一個類別（class），而類別和類別之間的關係有兩種：

(1) 類別組合關係（class aggregation relationship）。例如：學校由學生、教職員以及一些硬體設施所組成；學校是一個類別，而其他三個類別：學生教職員以及硬體設施則和學校類別之間的關係即爲類別組合類別。

(2) 類別繼承關係（class inheritance relationship）。例如：學校中的教職員可細分成行政人員和教師；也就是說行政人員和教師這兩個類別都是同屬於學校的教職員，也都繼承教職員的屬性，此時，教職員類別和其他兩個類別（行政人員和教師）之間的關係即爲類別繼承關係。

當我們要查詢資料時，爲了快速而直接的讀取我們所需的資料，最常使用的技術是索引法（indexing）[2,9]。在一般關聯式資料模組中是利用一個屬性當索引鍵值（key），將一個關聯（relation）中所有記錄（record）依索引鍵值排序好，若查詢兩個以上的關聯，則需先將兩個關聯合併（equal join）後再做查詢。而在物件導向資料模組中，因類別的關係有組合（aggregation）與繼承（inheritance）兩種，因此架構索引方法也有所不同。

在此論文中，我們將針對於類別繼承架構中，研究提出一個更爲有效率的索引方法。目前已提出的方法有：Ch-tree index[8]、H-tree index[10]、CG-tree index[7]、hcC-tree index[11]。由於類別的繼承關係（inheritance relationship），可將類別區分爲上屬類別和下屬類別的關係（super-class/subclass），也就是下屬類別將繼承上屬類別所有的屬性。當利用某一個屬性來查詢某一類別的範圍時，會包括兩種方面的問題：第一，查詢該類別中符合此屬性值的所有物件（instances）；例如查詢學校教職員的薪水，將可知道在學校類別中所有教職員的薪水；第二，以該類別爲起始類別，查詢該類別本身以及位於其下之類別繼承架構中所有下屬類別的物件；例如查詢學校教職員的薪水，除了學校類別的人員，還將包含行政人員和教師兩類別中所有人員。一個物件導向資料模組的索引法（indexing）必須能夠有效支援這兩種範圍的查詢。

在已提出的索引法 CH-tree、CG-tree 和 hcC-tree 中，都不能明顯地看出類別(class)與類別之間的相關性，而 H-tree 雖然有巢狀指標指向其下屬類別，但也只能看出其下一層的下屬類別，不能全面知道整個類別關係的架構。如果我們想針對特定類別做查詢，例如想知道某一類別以及其類別架構的左邊子樹（left subtree）的物件，或者想知道在整個類別架構中位於同一層次（level）類別的物件等等，這些索引法都無法很快速的找到答案，所以我們提出一個新的方法稱之爲類別編碼索引法（Class-coding Tree）。我們的作法是先將類別編碼，透過編碼不但可以很清楚的知道類別之間的關係，也可以知道類別在整個資料模組中的地位，並且在類別目錄中存放了類別編碼，如此便能很容易的回答上述的問題。而從實驗分析的結果比較，我們驗證了所提出的類別編碼索引法在各種查詢問題上比起其他的方法皆能有較好的查詢效率。

我們將於第二章簡要說明目前文獻針對類別繼承架構所提出的索引方法；第三章則提出我們類別編碼索引法的構想及作法；在第四章將說明分析使用的變數並列出我們對各索引方法所做的成本分析及效能結果比較；最後，我們將於第五章敘述結論與未來的研究方向。

2. 文獻探討

針對類別繼承關係架構上如何建立索引，目前已提出的方法大致可歸納成下列四種，基本上這些方法均是植基於 B+-tree 的方法上：

(1) *CH-tree* 索引法：只針對根類別（root class）的一個屬性當成索引值 建立一個索引，將所有類別中具有相同屬性值的物件串在一起。

(2) *H-tree* 索引法：則針對一個屬性分別於各個類別建立單一索引，稱之爲 *H-tree*，爲了加強前述第二類的查詢問題，各 *H-tree* 間建立巢狀指標（nesting pointer），用來降低上屬類別與下屬類別間 *H-tree* 查詢的時間。

(3) *CG-tree* 索引法：在實際儲存的末端節點（leaf nodes）也是採用單一索引的方式，但是將 *CH-tree* 索引中的索引值目錄（key directory）獨立出來成爲類別目錄（class directory）。

(4) *hcC-tree* 索引法：則儲存了單一類別和類別繼承架構的索引內容，並在中間節點（internal nodes）中存放位元圖（bit map），用來顯示某一類別是否具有符合該屬性值的物件，若位元圖中對應某類別的位元爲“0”，則表示該類別不具有此屬性值的物件，所以可以省略讀取該類別的內容之後，才得知該類別不具符合條件的物件。

在物件導向資料模組中，除了前述對某個類別（class）或是這一類別以及其下所包含的所有下屬類別做查詢外，查詢的範圍也可以分成兩類：單一查詢（point queries）、範圍查詢（range queries）；於是我們可將所有的查詢分類成四種型式：

CHP （Class Hierarchy Point）queries：
對某一個類別以及其下所有類別做單一查詢。

SCP （Single Class Point）queries：
對某一個類別做單一查詢。

CHR （Class Hierarchy Range）queries：
對某一個類別以及其下所有類別做範圍查詢。

SCR （Single Class Range）queries：

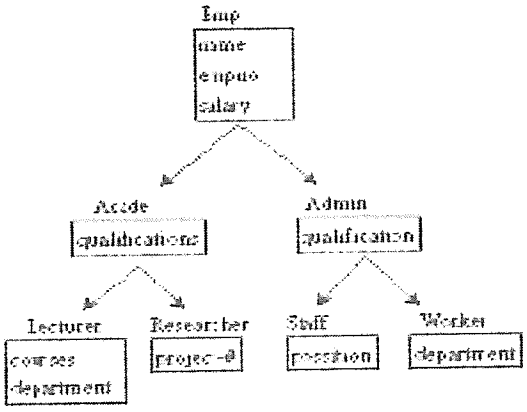
對某一個類別做範圍查詢。

3. CC-tree 索引法

在目前已提出的索引方法中（*CH-tree*、*H-tree*、*CG-tree*、*hcC-tree*），都不能明顯的看出類別與類別之間的相關性，換句話說，類別和類別的關係對索引而言是獨立的。以圖 1 的資料模組來說，如果我們想知道類別 Emp 以及其左邊子樹（left subtree）中薪水是 35,000 元的員工，或者我們想知道在資料模組中位於同一層次（level）的類別（例如： Lecturer、Researcher、Staff、Worker），其薪水是 35,000 元的員工，所有的方法都必須讀取許多不必要的資料，然後再做資料篩選的工作；然而，物件導向式資料模組的一個重要特性，即是能清楚而有效率的定義各類別物件以及其間之關係，有鑑於此，我們覺得一個好的索引法也應能表現出這樣的特性，然後可針對特定類別做有效地查詢。

因此，我們所提出的方法，稱之爲類別編碼索引法（*Class-coding Tree*，*CC-tree*），其構想是先以 Huffman code 的編碼方式[5]將類別編碼，以圖 1 的資料模組爲例，編碼的方式即如圖 2。

由圖 2 所示，透過編碼我們可以清楚知道每一個類別在資料模組中的地位，例如：類別Emp 的編碼（code ID）是(0)，那就表示這是一個根類別(root class)；類別 Acade 和類別 Admin 的編碼分別爲（00）和（01），由於這兩個編碼都只有兩個位元（bit），可以知道這兩個類別位圖



1、一個具有繼承關係的物件導向資料模組的例子

於資料模組的同一個層次（level），而且這兩個類別編碼只有第二個位元不同，第一個位元是相同的，也就是他們彼此是同一層兄弟類別（brother），更分別爲類別 Emp 的左子樹（left subtree）和右子樹（right subtree）；同理，類別 Lecturer、Researcher、Staff、Worker 的編碼依次爲（000）、（001）、（010）、（011），由編碼的個數可以知道他們位於同一層次；而類別 Lecturer 和 Researcher 編碼的前兩碼都是（00），也就是繼承類別 Acade 的編碼，所以很清楚的知道其分別爲類別 Acade 的左子樹和右子樹；而類別 Staff 和 Worker 編碼的前兩碼都是（01），也就是繼承類別 Admin 的編碼，所以很清楚的看出其分別爲類別 Admin 的左子樹和右子樹。當然，在現實世界中一個類別也許可能被許多類別所繼承，但是只要增加編碼的位元數即可，例如圖 3；其增加的位元數依子類別的個數而定，若有 X 個子類別，則需 $\log_2 X$ 個位元數來表示這些子類別的編碼。

接著，我們架構出類似 *CG-tree* 索引結構，因爲 *CG-tree* 索引的好處是可以有效地支援範圍查詢（range query），而且沒有如 *H-tree* 索引般建置巢狀指標的複雜性，也沒有如 *hcC-tree* 索引般雙倍儲存單一類別鏈及類別繼承組織鏈的空間浪費。

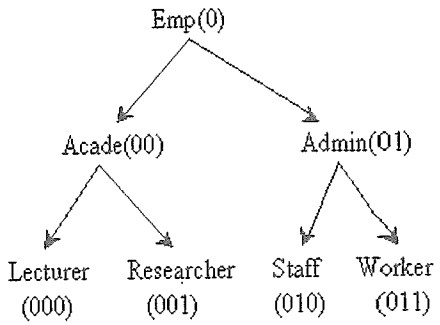


圖 2、簡單類別關係編碼方式

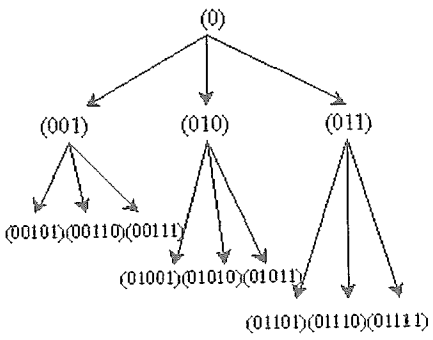


圖 3、複雜類別關係編碼方式（X = 3）

因此，類別編碼索引法和 *CG-tree* 索引法的結構其中間節點（internal nodes）和末端節點（leaf nodes）型式相同；

不同的地方在類別目錄（set directory）的結構。我們在類別目錄中多加一個編碼（*code ID*）的欄位，其結構將圖示於圖 4。

爲了減少類別目錄儲存空間的使用效率，我們也採用動態的儲存非空指標的類別目錄。其邏輯結構與實際儲存結構說明於圖 5。

圖 6 說明 *CC-code tree* 的架構圖。利用類別編碼索引法我們可以輕易地回答上述提到的問題：

(1) 當我們想知道類別 Emp 以及其左邊子樹（left subtree）中薪水是 35,000 元的員工；我們只要讀取編碼是（0）、（00）以及（00）其下屬類別（000）、（001）的類別指標就可以了。

(2) 當我們想知道在資料模組中位於同一層次（level）的類別（例如： Lecturer、Researcher、Staff、Worker），其薪水是 35,000 元的員工；我們只需讀取編碼都是三位元的類別（000）、（001）、（010）、（011）。

當然，類別編碼會需要額外的儲存空間，例如：會需要一個表格儲存類別編碼與類別辨識碼（*class ID*）的參照表，以及類別目錄中多出的類別編（*class code*）碼欄位；但是，這些空間不會佔據太多的記憶體（例如：255 個類別僅需額外記憶體空間 255bytes），而且對提昇索引的效率來說是值得的。

4. 效能分析

在這部份，我們將對所提出的類別編碼索引法（*CC-tree*）作成本分析，並和 *CH-tree*、*H-tree*、*CG-tree* 及 *hcC-tree* 索引法於儲存成本及查詢成本等方面作分析比較(因篇幅限制，這些方法之成本分析請參閱[20]。

4.1 成本分析模組

爲了讓我們的實驗和成本分析模式易於操作，以及爲了很公平地比較每一種索引方法，針對我們的分析模組提出以下幾點假設：

1. 所有的索引值（key value）都是相同的長度。
2. 每個索引值的個數皆相同。
3. 每個非末端節點（nonleaf node）擁有相同的 fanout（*f*）。

而整個分析所用到的控制參數如表 1 以及導出參數如表 2 所示。除此之外，我們考慮索引值分佈的三種狀況：

1. Disjoint（互斥）：D = ΣD_i，Nc = 1。
2. Total Inclusive（完全包含）：$D = MAX(D_i)$，$Nc = Tc$。
3. Partial Inclusive（部份包含）：$D = \Sigma(D_i)/Nc$，$1 < Nc < Tc$。

在我們的成本模組中，在不影響成本分析結果的公平與正確性下，我們將明確給定數值於表 3 的幾個參數：此外，我們給定一個儲存頁的大小 $P = 4096$ (bytes)，而一個物件辨識碼（*OID*）的空間佔 8 bytes。

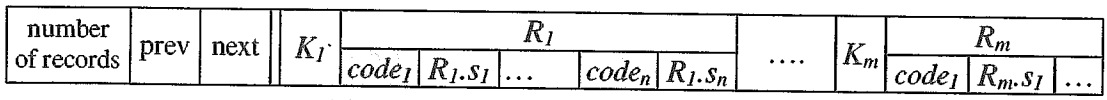
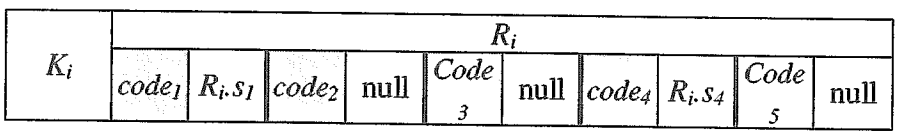
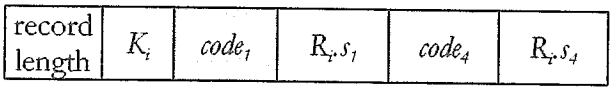


圖 4、類別編碼索引法之類別目錄的表示法



Conceptual Representation



Physical Representation

圖 5、類別編碼索引法之類別目錄中一筆資料項的表示法

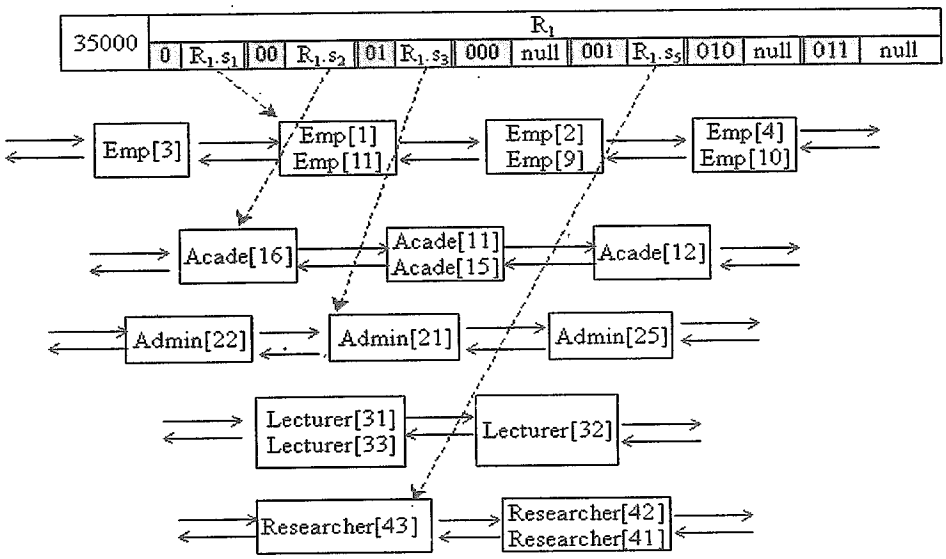


圖 6、一個類別編碼索引法的例子

4.2 儲存成本

一個索引的大小即是所有索引節點（index node）所佔用儲存空間的總合。在這節當中，我們將說明 *CC-tree* 索引法之成本分析模組及各索引法分析比較之結果。

4.2.1 CC-tree 索引法之儲存成本分析模組

因爲 *CC-tree* 具有類別目錄，所以計算一筆資料項（*NE*）時是以類別目錄（*Dir*）中一個索引值的類別目錄爲單位（*XD*），但在其中增加儲存每個類別的編碼（*code ID*），由索引樹的高度（*Nch*）來決定編碼所需的長度；另外，*CC-tree* 同 *hcC-tree* 索引法具有位元圖（bit map）的對應功能（mapping），所以有這兩項額外的儲存量。整個索引的儲存空間（*NN*）亦爲中間節點（*NI*）、末端節點（*NL*）和類別目錄（*Dir*）的加總。

表 1、分析使用之控制參數及敘述

Control Parameters	
Labels	Descriptions
D_i	no. of unique values in the index of class *i*.
D	no. of unique values in the domain of the indexed attribute.
N	total no. of objects in the class hierarchy.
N_i	no. of object in class *i*.
K	total no. of OIDs per key in an attribute of a class hierarchy.($\mathbb{K} = \lceil \mathbb{N}/\mathbb{D} \rceil$)
K_i	average no. of OIDs per key in an attribute of a class *i*.($\mathbb{Ki} = \lceil \mathbb{Ni}/\mathbb{Di} \rceil$)
F	average fanout from a nonleaf index node.
P	size of an index page in bytes.
Tc	total no. of classes in the class hierarchy.
Nc	no. of classes that contain objects with an indexed value.
NRQ	no. of key values in the range specified for a given query.

4.2.2 分析結果與比較

在我們的實驗中採取每一個類別擁有的物件量爲（Ni = 50,000）、索引值域爲（Di = 2000），整個類別架構（class hierarchy）包含 10 個類別（Tc = 10），然後從變化索引值的分佈情況（*Nc*）來比較各索引法在儲存空間的需求。

由圖 7 可明顯的看出，在索引資料的儲存上，集合類別架構中所有類別而建立唯一索引的方法（*CH-tree*）會比針對單一類別建立索引的方法（*H-tree*、*CG-tree*、*CC-tree*）需要較多的儲存空間；而 *hcC-tree* 因爲具有這兩項儲存特質，所以其儲存空間幾乎是各索引方法的兩倍。

其次，圖 7 的橫座標（*Nc*）就是索引值分佈情況，當 *Nc* = 1、*Nc* = 10 是兩個極端的例子，前者爲互斥（disjoint）、後者爲完全包含（total inclusive），介於中間的數值則是部份包含（partial inclusive）；由圖可知在儲存空間上，針對單一類別建立索引的方法（*H-tree*、*CG-tree*、*CC-tree*）對於索引空間的需求並沒有很大的變化；而對於集合類別架構中所有類別而建立唯一索引的方法（*CH-tree*、*hcC-tree*）則在資料爲完全包含的情況下（*Nc* = 10）有相當大的需求空間。

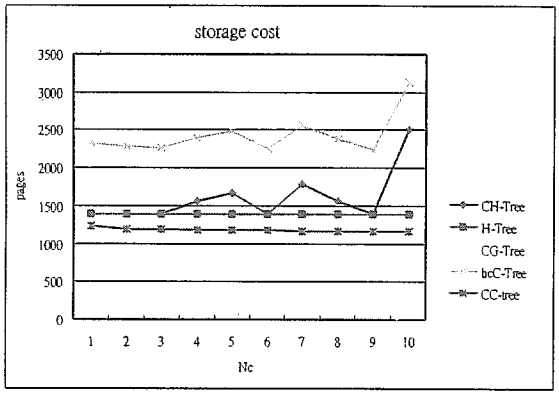


圖 7、儲存空間成本

另外，圖 7 中，*CH-tree* 和 *hcC-tree* 這兩種索引法在資料部份包含的情況下對儲存空間的需求有高低起伏的狀況，原因是我們在計算儲存空間時，是先計算末端節點的一個儲存頁可以包含的平均資料項（*NE*）爲多少，亦即一儲存頁能存放的物件含有多少個不同的索引值域（D_i），其中索引值域的定義域爲（$D = \Sigma(D_i) / Nc$），因此當其儲存之索引值域（*NE*）相同，而資料部份包含的情況增加（亦即 *Nc* 增大）時會使得索引值域的定義域減少（亦即 *D* 減少），那麼末端節點的空間就會相對減少，繼而減少中間節點的需求，所以整個儲存空間的需求就會減少。

4.3 查詢成本

在這一節當中，我們將分析各個索引法在四種查詢型式下的查詢存取成本。此四種型式分別爲單一索引值之單一類別查詢（SCP）、類別架構查詢（CHP）、單一類別之範圍查詢（SCR）以及類別架構之範圍查詢（CHR）。

4.3.1 CC-tree 索引法之查詢成本

單一索引值的查詢成本，就是索引樹的高度，也就是計算中間節點（*NI*）時，將末端節點（*NL*）遞迴地除以（fanout）的次數，（當然如果有溢位時還要加上額外存取的頁數）；至於範圍查詢，除了索引樹的高度外，還要以查詢多少個索引值（*NRQ*）來決定需要讀取多少頁

表 2、分析使用之導出參數及敘述

Derived Parameters	
Labels	Descriptions
XC	average length of a nonleaf-node index record in bytes for a class-hierarchy.
XS_i	average length of a nonleaf-node index record in bytes for a single class.
XS	average length of a nonleaf-node index record in bytes for all classes.
XD	average length of a directory-node index record in bytes for CG-tree.
NE_i	no. of entries in a leaf node for a single class.
NE	no. of entries in a leaf node for all classes.
OID	no. of oid node in hcC-tree index .
Dir	no. of directory node in CG-tree index .
NL_i	no. of leaf node in index *i*.
NL	no. of leaf-level index pages.
NI_i	no. of internal node in index *i*.
NN_i	no. of nodes (leaf and internal) in index *i*.

表 3、給定之參數值

1.在非末端節點（nonleaf node）中，	
索引值的長度 (key length)	2 bytes
索引值 (key value)	8 bytes
指向下一層的指標(next-level-page pointer)	4 bytes
2.在末端節點（leaf node）中，	
紀錄的長度 (record length)	2 bytes
索引值的長度 (key length)	2 bytes
索引值 (key value)	8 bytes
溢位的指標 (overflow-page pointer)	4 bytes
3.在索引值目錄 (key-directory) 中，	
類別的數量 (number of class)	2 bytes
類別辨識碼 (class ID)	2 bytes
移位 (offset)	4 bytes
物件辨識碼的數量 (number of *OID*)	2 bytes

的末端節點。因爲 *CC-tree* 具有類別目錄，所以存取成本還要加上這一層，另外，末端節點的存取，由於無法預期是哪些類別擁有符合索引值的物件，所以我們採取平

均值。但在其中因具有同 *hcC-tree* 索引法之位元圖（bit map）的對應功能（mapping），可以較早得知物件存在與否，所以是以平均高度來計算查詢成本：

$$\mathbf{SCP} \Rightarrow ave\_height(CC-tree)+1$$
$$\mathbf{CHP} \Rightarrow ave\_height(CC-tree)+Nc$$
$$\mathbf{SCR} \Rightarrow ave\_height(CC-tree)+\left\lceil \frac{NRQ}{P_{XS}} \right\rceil$$
$$\mathbf{CHR} \Rightarrow ave\_height(CC-tree)+\left\lceil \frac{NRQ}{P_{XC}} \right\rceil * Nc$$

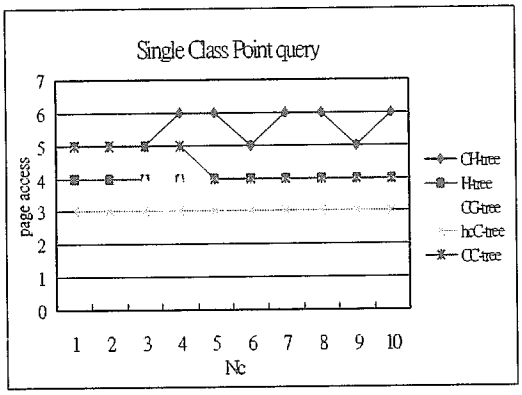


圖 8、SCP query

4.3.2 分析結果與比較

在單一類別之單一查詢問題（SCP）中，所有索引方法讀取儲存頁的次數，都是決定於索引樹的高度，由圖 8 可知，其值介於 3 至 6 次；由圖 8 並可看出利用單一類別建構的索引法（*H-tree*、*CG-tree*、*hcC-tree*、*CC-tree*）其高度遠較集合類別架構中所有類別而建立的索引法（*CH-tree*）爲低，所以在回答查詢問題如（Q1:請找出 Class Admin 中薪水爲 50,000 元的員工）時，*CH-tree* 索引法的效率最差；但是在類別架構之單一查詢問題（CHP）中，如回答查詢問題（Q2:請找出薪水爲 50,000 元的所有員工）時，由圖 9 可知，集合類別架構中所有類別而建立的索引法（*CH-tree*、*hcC-tree*）卻有較好的效能。*hcC-tree* 索引法因同時具有這兩種建構方法的特性，所以在這兩項查詢問題上擁有最佳的效率。

從圖 9 和圖 10 會發現，我們的實驗和 Kilger & Moerkotte（Kilger & Moerkotte, 1994）說明兩種建置索引方法的效能差異不謀而合，也就是在類別架構之單一查詢問題（CHP）中，針對屬性值將物件群組在一起（grouping by key）的索引法（*CH-tree*、*hcC-tree*）有較佳的效能，但是在類別架構之範圍查詢問題（CHR）中，如回答查詢問題（Q3:請找出薪水爲 50,000 到 65,000 元的所有員工）時，另外一種針對類別將屬性值相同的物件群組在一起（grouping by set membership)的索引法（*H-tree*、*CG-tree*、*hcC-tree*、*CC-tree*）則有較佳的效能。

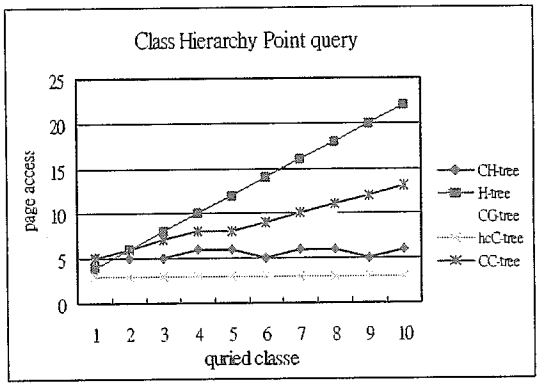


圖 9、CHP query

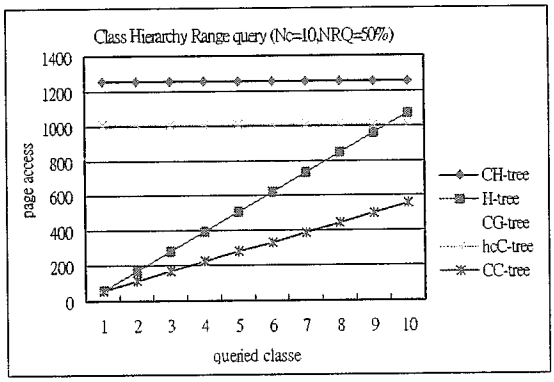


圖 10、CHR query (*Nc*=10, *NRQ*=50%)

5. 結論

在物件導向資料模組中，因類別的關係有組合（aggregation）與繼承（inheritance）兩種，因此架構索引的方法也從這兩方面著手。在本論文的第二章詳細介紹了在類別繼承架構的資料模組中，目前已提出的索引方法（*CH-tree*、*H-tree*、*CG-tree* 以及 *hcC-tree*），然而由於這些索引方法皆無法清楚地表示類別間的關係，因此在一些查詢問題的效能（performance）上有其限制。因此，本論文在第三章中提出了一種更爲有效率的方法，稱之爲類別編碼索引法（*Class-coding Tree, CC-tree*），其作法主要是藉由類別編碼來記錄類別間的繼承關係；而第四章中的成本分析則進一步地證明我們提出的 *CC-tree* 索引法在任何的查詢問題上都比其他索引法擁

有最佳的查詢效率。再則，如何來改進本索引法以滿足類別與類別之間的關係同時存在著組合關係與繼承關係，將是我們未來研究的方向。

參考文獻

[1]Bertino, E., & Kim, W., Indexing Technique for queries on Nested Objects. IEEE Trans. on Knowledge and Data Engineering. Vol 1, No 2, pp.196-204, 1992.

[2]Bertino, E., Index Configuration in Object-Oriented Databases. The VLDB Journal, Vol 3, No 3, pp.355-399, 1994.

[3]Date, C. J. , An Introduction To Database Systems. 6th-ed, Addison-Wesley Publishing Inc., American ,1995.

[4]Ehud Gudes , A uniform Indexing Scheme for Object-Oriented Database. IEEE Transactions on Knowledge and Data Engineering,, pp.238-246 , 1996.

[5]Huffman, D. A. , A Method for the Construction of Minimum Redundancy Codes. Proc., IRE 40, Sep. 1952.

[6]Kemper, A., & Moerkotte, G., Access Support Re lations: an Indexing Method f or Object Based, Informations Systems, Vol 17, No 2,pp117-145,1992.

[7]Kilger, C., & Moerkotte, G. , Indexing Multiple sets. The MIT Press, Proceedings, 20th VLDB conference, Santiago,Chile.pp.180-191,1994.

[8]Kim, W., Kim, K.C., & Dale, A. , Indexing technique for Object-Oriented Database. Object-Oriented Concepts,Database, Object-Oriented Concepts,Database, and Applications. ACM Press, pp.371-394,1989.

[9]Lee, Chien-I, Chang, Y.-I., W.-P. , A New Indexing organization for a class-aggregation hierarchy in object-oriented databases Journal of Information Science and Engineering, Vol 15,No2 , pp.217-241,1999.

[10]Low, C. C.,Ooi B. C., & Lu, H., H-Trees: a Dynamic Associative Search Index for OODB. The MIT Press,.Proceedings, ACM SIGMOD. pp. 134-143,1990.

[11]Sreenath, B. & Seshadri, S. , The hcC-tree: An Efficient Index Srcture For Object Oriented Database. VLDB conference, Proceedings. pp. 203-213,1994.