雜湊函數方法的 XML 查詢處理

XML Query Processing based on Hash Function

李建圖
南華大學資訊管理研究所
m5292@ms19.hinet.net

吳光閔
南華大學資訊管理研究所
gmwu@mail.nhu.edu.tw

厲彥廷
南華大學資訊管理研究所
yanteen_lee@yahoo.com.tw

摘　　要

XML 制定的主要目地是為了能在網際網路上傳送交換資料或處理文件資料，且由於 XML 著重在文件資料的結構性描述，所以在資料交換或呈現上有著一定的特色與優勢，但隨著所採用的人愈來愈普及，相對的資料量也愈來愈龐大，衍申出來的問題便是 XML 資料的檢索搜尋效率問題。本論文提出一個新的資料檢索技術，是透過將 XML DOM 結點加以編碼，計算出一個以位置為概念的編碼位置值，加以註記於節點中，在查尋檢索的過程中，透過提出的演算法，加以篩選過濾掉不必要的處理程序，本論文提出的方法皆與 2001 年由 Park and Kim[1]提出的查詢方式相比較，結果顯示本論文所提出之檢索方法的效率高於 Park and Kim 所提出之方法，且可在 XML 資料中快速的檢索，平均效率提昇了約 51%。

關鍵字：XML、檢索、查詢、 DOM

Abstract

The Extensible Markup Language (XML) has become the standard for information interchange on the Web. There are ever increasing applications; E-commerce application in particular, using XML as a mediator model. So, an XML digital signature for XML-based e-business systems is necessary. Several indexes are proposed for XML regular path expressions, i.e., [1]. In this paper, we propose a new signature-based query optimization technique, called mark-DOM, to minimize the number of nodes retrieved from the database. For any node's signature in XML data, mark-DOM uses a hashing function to coding it. We can reduce the number of nodes fetched by the signatures. Experimental results show that our mark-DOM achieves improvement of 51% in the number of queried nodes compared with the method in [1].

Keyword: XML, index, query, DOM.

壹、緒論

XML(eXtensible Markup Language)可擴展標示語言，是由W3C(World Wide Web)所提出，為一種描述語言。在XML未被提出前，傳統的HTML到處可見，像是線上的詩、詞、文章等，都可以看到HTML的採用，但HTML有著一些至目前為止，都無法突破的瓶頸。一般而言HTML所著重的是對於資料的呈現佈局而已，甚致有些資訊對使用者而言是毫無意義，而XML的提出卻解決了傳統HTML此類的問題。如我們可以容易的對我們的XML文件，做文件描述的能力，甚致可以輕易的改變它，但傳統的HTML卻不行。

XML之所以近幾年被廣泛的研究，就是因為他有著傳統標示語言(Markup Language)不一樣的特性與能力，像是使用者能自定標籤[2]，對資料上的文件格式定義DTD，及有著相關標準的跨平台特性。這些功能都能夠取代或延伸傳統的Cold Fusion或ASP的自訂標籤功能，也為企業上的資料交換帶來了解決的方案。

雖然XML與HTML都是SGML(Standard Generalized Markup Language)上應用的子集[2]，但XML與HTML最大的不同是在於資料的表達，不但可以表示處理大量的文字資料，且在標示(Markup)上有著更嚴謹的意義規範限制與儲存格式[3,4]，XML並不像傳統資料表示方法一樣，我們可以將XML視為樹狀結構(tree structure)，查詢(Query)運用規則路徑表示法(Regular Path Expression, RPE)。而在其間有許多學者也曾提出相關對XML資料查尋索引的技術，但在大多數的例子裡，在索引的過程中都會遺漏相關資訊或資料，或是無法包含全部的資料內容，這些方法都還存在著許多問題及改善的空間。

在XML的儲存也與一般的資料儲存方式有所差異，XML的資料儲存方式可以為檔案，也可以是所謂物件的方式，但在傳統的關聯式資料庫系統中(RDBMS)，是無法滿足儲存真實XML樹狀結構形式的資料。因些有許多學都也紛紛提出，如半結構化的資料庫儲存方式[5,6]及半結構化的資料庫系統，如Lore[7]、Xcelon[8]，及目前最流行的原生性資料庫(Native XML Database)[9]。如今XML的資料儲存與查詢也成為近來來流行的研究趨勢與方向。

而在這新興的網際網路語言中，不但可以讓使用都來自定其標籤(tag)來描述其相關資料內容的意義，且更進一步也解決以往在不同平台上資料交換的問題。竟然XML能解決在不同平台上大量資料交換或共享的問題，相對而言在XML資料的查詢也是一項極為重要的議題。有許多學都也提出了相關XML查詢語言，如XML-QL[10]、XML-GL[11]、XQuery[12]、Xpaht[13]等。在資料的查詢上也有學者提出相關的查詢索引方法[14]，而這些XML的查詢語言，絕大部分也都是依循著W3C所認定的規則路徑表示法。雖然W3C嚴然已成為所有XML研究者共同認定的標準單位，但就許

```
<?xml Version=”1.0”?>
<!DOCTYPE school>
<school>
   <teacher>
      <name>Litoto</name>
      <address>Taipei</address>
      <telephone>

<mobile>0933333333</mobile>
         <home>0222228885</home>
      </telephone>
   </teacher>
   <students>
      <name>T-C,Li</name>
      <address>Taipei</address>
      <degree>Three</degree>
   </students>
</school>
```

圖 1：XML 文件範例

多相關的 XML 領域技術而言，還沒有一定的標準可遵循，如 XML 查詢技術的效率等。這些研究還有著很大的改善空間。

在本論文研究中，提出一項查詢索引的技術，能夠將 XML 的樹狀結構視為物件，儲存於 XML 資料庫中，透過所提出的索引編碼方式，來達到效率的提昇。本論文其餘部分如下。第貳章將做有關的文獻及背景技術的探討，第參章我們說明 XML 的表示方法及如應用在本論文的實驗方法，第肆章我們深不探討本論文所提出之演算法，第伍章提出實驗結果，第陸章為結論。

貳、文獻探討

XML 的掘起，為資料交換上帶來了解決的方案，由此可知 XML 在市場上的潛力是無窮，這近兩年內，也有許多世界知名的資料庫軟體廠商也紛紛提出可支援 XML 資料儲存及處理的功能，打入市場。如 Oracle 9i 的 XDB、IBM DB2、微軟的 SQL server2000，皆有支援 XML 的儲存[15,16]，但這些資料庫軟體廠商的資料庫系統，並非真實的去儲存 XML 樹狀階層性的資料內容，目前有些學者也提出了有關的原生性資料庫(Native XML Database, NXD)，但在 NXD 導入市場卻又存在著與傳統 RDBMS 上資料轉換上的問題，這是一項值得再深入研究討論的問題，但在本論文不加以探討。

本論文主要的研究在 XML 查詢技術上，且透過編碼來達到檢索查詢上效率的改善提昇。早期編碼概念的提出有許多，較出名且目前較被廣泛採用的有雜湊方法(Hash)[17,18]或其它編碼方式的運用[19]。後來 XML 的興起，有許多學者也紛紛將其關念運用於 XML 的搜尋檢索技術上，如 Index 的運用[14]、符號編碼的處理概念[20]及半結構化資料庫的檢索[21]。當然絕大部分的用都是遵循 W3C 所認定的 XML 標準規範[22]。

2001 年 Sangwon Park 與 Hyourg-Joo Kim 所提出的 s-DOM[1]方法及 2002 年提出的方法[20]，皆是將 XML 文件資料視為樹狀結構，且每一結點視為一物件，並將之存於 XML 資料庫中，且以 XML 的 API 運用[23]來處理 XML 文件的搜尋或其它應用，而許多 XML 後端之處理也是運用此概念。但相信目前就 XML 的 Patten tree 查詢上還

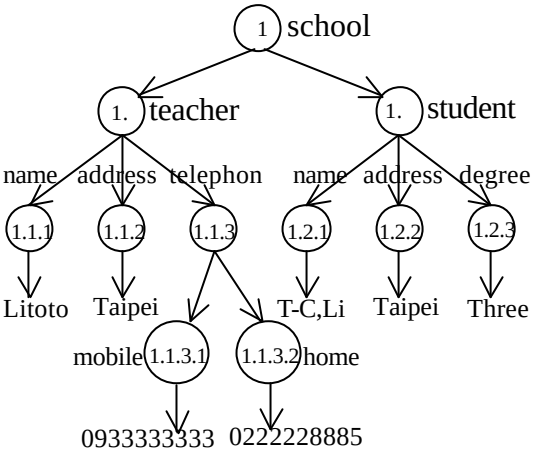


圖 2：DOM 表示法

有相當多的改善空間。目前仍有許多學者，依然紛紛提出改善方法或是自己的演算法，如[24,25,26,27]等，或引用一些新的關念於 XML 的查詢技術上，像是 Search filter[28]的運用。在本篇論文所提出的方法，在實驗結果皆與 s-DOM 的方法做比較，實驗結果顯示，本論文所提出的方法，較有效率，且具較優的可擴充性。

參、XML 的表示方法

本章節中，我們將探討 XML 文件的表示方法、相關的 XML 查詢方法技術，及如何將其運用於本論文所提出的方法。

一、XML 表示方法

XML 文件相較於一般的 HTML 文件，最大的不同是在結構上規範更嚴僅，且在每個文件上的標籤(tag)，可由使用者來自訂，且更具有意義。如圖 1 文件就是一個簡的 XML 文件範例，圖 1 的第一行<?xml Version=”1.0”?>代表 XML 板本宣告。而第二行之<!DOCTYPE school>為此文件 XML 的文件格式定義 DTD 檔(Document Type Definition)，這功能主要是從 SGML 繼承而來，相較於 SGML 而言，XML 的 DTD 比 SGML 更為簡潔好用多了，且 XML 文件也允許設計者不使用 DTD，但 SGML 卻一定要使用。

文件格式定義 DTD 可以是一個獨立的檔案或是直接撰寫於 XML 文件檔案中，只要在 XML 文件開頭註明此 XML 文件檔案所需依循的文件格式定義檔案名稱就行了，當然也可以在不同的 XML 文件中，使用相同的 DTD 檔案。而文件格式定義 DTD 主要是用來定義 XML 文件所有參考的資料元素屬性及文件的結構定義格式，透過文件定義的方式，只要不同端的平台，所要交換的資料參考同樣的文件格式定義 DTD，就可以很容易的達到資料交換的目的。

由圖 1 我們可察覺，其實 XML 文件的嚴謹，我們能將之視為一樹狀結構(tree structure)，此方法我們可稱之為 XML DOM 表示法[29]，將每個 XML 文件的元素、屬性和內容，都視為一個節點(node)，且儲存在 XML 資料庫中。因此我們能將圖 1 之 XML 文件轉換對應到圖 2 二樹狀結構的 DOM 圖表示法，每個標籤視為一個元素(element)

```
/school/*/address[@caption="Taipei"]
```

圖 3

節點，將葉節點視為一個屬性(attribute)節點。而這樣的表示方法就如同物件(Object)的資料，在本篇論文中我們也將運用這樣的觀念，以拜訪節點的方法，加以過濾篩選掉資料，達到最佳的檢索效率。

二、查詢方法

當然有了資料的儲存與表示方法，相對而言也會有資料的查詢方式產生。一般而言，XML 查詢表示上，大多所提出的方法都是依循著規則路徑表示法(Rgular Path Expression)的原則[30,31,32]，目前支援規則路徑表示法之查詢語言有 XML-QL、XQL、Xquery、XSL 等。這些方法都提供了一些簡易懂的語法，可透過一些符號或是條件式的文字，來達到擴大範圍的搜尋或查詢。圖 3 就是一個簡單的 Xquery[33]語法的查詢句子，主要目地是要找出 school 下所有 address，且必須是住在 Taipei。此查詢的 school 與 address 是元素(element)，caption 為屬性(attribute)。當然在所有的 XML 查詢語言表示法中，都有些許的差異，不見得完全相同，也有少部分不依循 RPE 的規則，但本論文不加以討論。

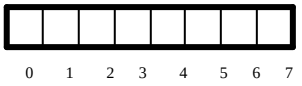


圖 4(a)

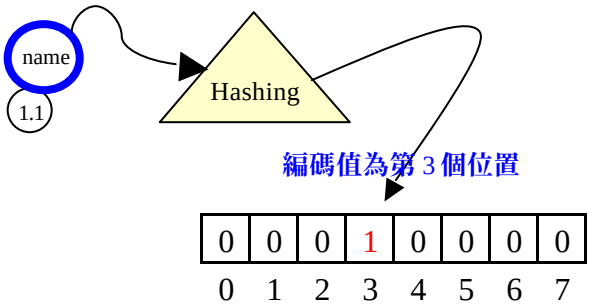


圖 4(b)

肆、以空間概念為主的編碼查詢處理

在本章節中，我們詳細描述我們演算方法與一些演算法的效率改善方法，並探討 s-DOM[1,20]的編碼方式可能產生的缺點。

一、傳統編碼的缺點

在 s-DOM 的編碼方法中，若在資料量龐大或編碼位元不足情形下，可能會導至下列二項嚴重的缺點(請參閱 s-DOM 編碼方式[1,20]。

(一)重覆性太高

若資料編碼的位元不足或不夠長，再加上資料量龐大的情況下，則會產生不同的編碼來源值，經過編碼後產生相同的編碼結果，且重覆性相當高，效率自然下降。

(二)搜尋時容易產生錯誤的判斷

若因重覆性太高，則可能導致系統在做搜尋處理時，產生錯誤判斷的例子，系統斷定存在但事實並不存在於目前搜尋中，導至時間的浪費。

節點編號	節點內容	編碼值
1	School	00100000
1.1	Teacher	10000000
1.2	Students	00000010
1.1.1	Name	01000000
1.1.2	Address	10000000
1.1.3	Telephone	00000100
1.1.3.1	Mobile	00010000
1.1.3.2	Home	00000001
⋮		

表 1：節點編碼值 H_i

編點編號	編碼值
1	11110111
1.1	11010111
1.2	11100110
1.1.1	01000000
1.1.2	10000000
1.1.3	00010101
1.1.3.1	00010000
1.1.3.2	00000001
:	

表 2：向上記錄編碼值 S_i

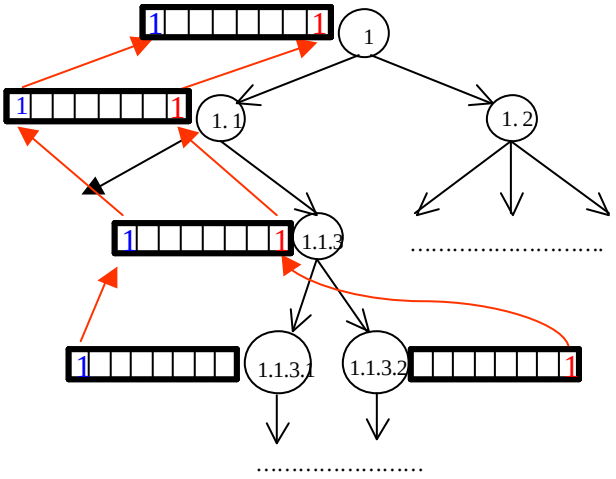


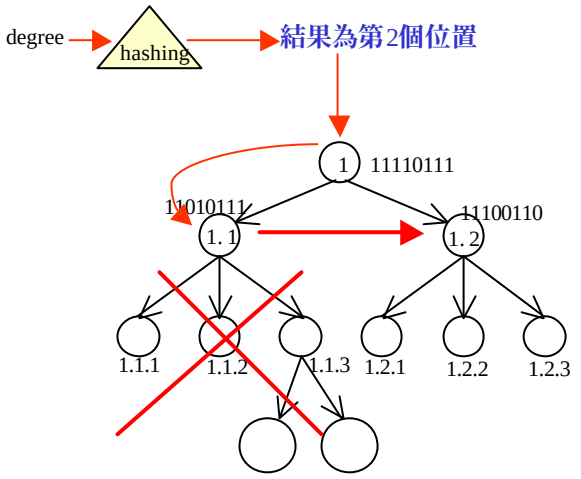
圖 5

二、雜湊函數為主的空間編碼

本論文提出的方法，我們假設所有的 XML 文件樹狀結構上的節點，皆儲存為一物件，如 DOM[34,35,36]。圖 2 就是一個 XML 文件 DOM 圖，文件內容請參考至圖 1 的 XML 文件。且將所有的 DOM 節點儲存於 XML 資料庫中。

首光我們要探討說明的是何謂空間編碼的概念，假設我們以 8 個 bit 為單位來舉例。圖 4(a)就是一個 8 個 bit 的儲存空間，我們將雜湊函數(hash function)將 DOM 結點內容加以編碼，且編碼後的值不會像 ASCII 碼一樣，在 8 bit 中會有一個以上的 1 產生，如(10010010)；而只會是一個位置。換句話說，在這 8 個 bit 的儲存空間中只會有一個 bit 為 1，其餘皆為 0，如圖 4(b)。

而此 8 個 bit 的儲存空間代表著 8 個不同的位置，在這 8 個位置中唯一的一個 1(其餘皆 0)，是指此 DOM 節點內容的編碼的位置代表。



圖六 6

我們以此概念將所有 DOM 節點加以編碼，並且在每個個別 DOM 節點編碼產生後，再***向上記錄***。所謂向上記錄是指目前 DOM 節點經過 hash function 編碼計算出位置後，不但要將此結果記錄於目前 DOM 節點自己本身的 n-bit 儲存空間中，並且還必須將此編碼後所代表的位置，向上記錄於此 DOM 節點的所有祖先節點中，如圖 5 為例。當編號 1.1.3.1 的 DOM 節點經過 hash function 編碼後的位置(預設皆為 0)，假設結果為第 0 個位置，它不但要將此結果記錄在編碼 1.1.3.1 自己本身中，還要將此結果向上記錄於 1.1.3.1 的祖先節點 1.1.3、1.1 及 1 的 DOM 節點中。以此類推，我們便能完成所有 DOM 節點的編碼，演算法 1 為本文的空間概念編碼演算法。

表 1 為圖 2 之 DOM 圖的部分節點個別編碼 H，表 2 為圖 2 之 DOM 節點編碼之向上記錄後的編碼結果 S。由表 2 我們可以看出，愈是在 DOM 樹上層的 DOM 節點編碼 S，其 S 不只一個位置是 1。而在 DOM 樹的最下層葉結點之編碼值 S，只會有一個位直會是 1，其餘皆為 0。這就是向上整合的結果，如表 2，因此由上述我們可以定出一則：

法則 1

$S_i \subset S_i'$

S 為表 2 之向上整合值

i 為 DOM 節點編號

S_i'為 S_i 的子樹

由上法則我自可得知，

$S_{(1)(1.3)} \subset S_{(1.1.3.1)(1.1.3.2)}$、

$S_{(1.1)} \subset S_{(1.1.1)(1.1.2)(1.1.3)}$、 $S_1 \subset S_{(1.1)(1.2)}$。

三、查詢處理

本論文所提出之空間概念的編碼查詢，最主要的優點是可透過上一段落所介紹之雜湊函的編碼，然後在查詢時直接裁減掉不必要的查詢過程，本章節是探討如來運用此編碼來做查詢的處理。

我們以一個簡單的概念來判斷使用者所要查詢的條件，是否存在於目前 DOM 節點的子樹之中。假設存在於目前 DOM 節點的子樹中，則繼續向其子樹搜尋；若不存在於目前 DOM 節點

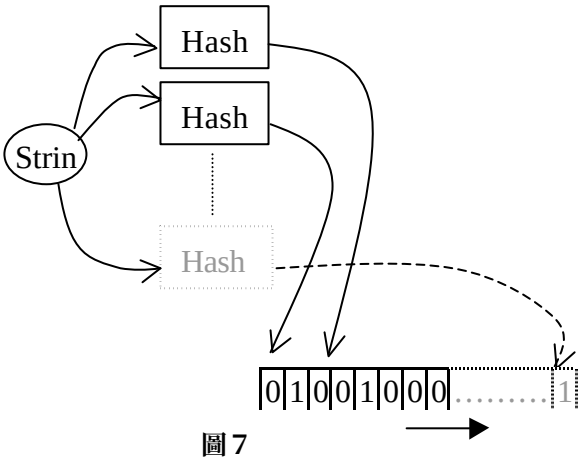


圖 7

的子樹中，我們可斷定，此 DOM 節點之子樹不會存在使用者所要查尋的相關訊息，因此我們可以直接略過。

以圖 6 為例，假設有一個 DOM 節點，其節點內容為 degree，我們將此 DOM 節點內容，透過我們所設計的 hash function 加以編碼，求出此 DOM 節點內容所代表的位置值。換句話說，也就是將 degree 加以編碼。假設編碼後為第二個位置，其值就為 00100000，我們可由此判斷第二個位置就代表其 degree 本身，然後我們便可以進入實際搜尋的步驟。如圖 6 為例，我們將 degree 編碼後的值由編號 1 的 DOM 節點開始搜尋，我們發現此 DOM 節點的 S_1 的第二個位置為 1，也就是 true，代表我們悔須再向下搜尋。我們就再往其子樹搜尋。接下來便是編號為 1.1 的 DOM 節點，我們發現其節點 $S_{1.1}$ 的第二個位置不為 1 而是為 0，也就是 false，代表著我們已不必要去再乎此節點的子樹了，我們便可直接裁掉此子樹的搜尋，轉向其兄弟之節點，如圖 6 所示。由上述訴我們可將查詢步驟分為兩個階段：

一、使用者查詢的條件編碼

將使用者下達之查詢句的條件內容，加以分解後且將條件編碼。

二、樹的比對查詢處理

以編碼後的條件，實際進入資料中搜尋。

上述的查詢處理過程中，我們可定下另一法則：

法則 2

假設 $H_i \not\subset S_i$ 則 S_i' 必不存在於 H_i

S 表 2 之向上記錄編碼值

H 表 1 之個別節點編碼值

i DOM 節點編號

S_i' S_i 的子樹

五、效率的改善

由於資料量可能非常龐大，本論文所提出的方法，可有多種不同的改善方法效率的方法，可避免因編碼結果相同所導至的結果。

(一)、增加編碼長度：

將編碼長度增加，例如從 1byte 擴增至 2bytes 或更多。可減少編碼結果的重覆性。

(二)、以多個 hash function 來編碼：

以 n-hash，對同一節點內容編碼，減少因單一 hash 編碼驗証所造成的位置重覆，以 n-hash 編碼來驗証，可降低重覆的機率，如圖 7 所示。

(三)、儲存位置的不同：

將不同的雜湊函數所計算之結果，存放於不同儲存空間，可預防因資料量龐大，在同一位置可能會有多筆不同內容的編碼值相同的資料重疊發生圖 7。

	節點總數	檔案大小
Shakespeare(D1)	537,621	7.5Mbytes
The book of Mormon(D2)	142,751	6.7Mbytes

表 3：XML 資料檔案

Q1	D1	PLAY.*[2].PERSONA
Q2	D1	*.TITLE
Q3	D2	tstmt.*[1].(title|ptitle)
Q4	D2	*.chapter

表 4：4 個查詢子句

伍、實驗估計

一、整體比較分析

本論文實驗程式撰寫採用 Java 語言，我們將所有資料儲存如物件，且查詢作業採用提取物件的方式作業。本實驗利用文獻[1]中實驗所用資料庫，表 3 為本論文實驗所之 XML 檔案及每個檔案之節點數，這些檔案是來自於 Shakespeare 與 The book of mormon 文章中，我們將這些檔案轉換為 XML。表 4 為本論文實驗所採用的查詢。在本論之實驗估計，共用了 4 個查詢來實驗。在每個查詢實驗中都與 s-DOM 方法來相比較。表 5 為在本論文實驗估計上所用之 Hashing function，表 5(a)為與 s-DOM 比較之函數，表 5(b)為與表 5(a)比較之函數。實驗結果顯示出本論文所提出之方法，效率高過於 s-DOM。

$$\left\lfloor \frac{\sum_{i=1}^{a} ASC(S_i)}{N} \right\rfloor \qquad \left\lfloor \frac{\sum_{i=1}^{a} asc(S_i)}{N} \right\rfloor$$

(a) (b)

S 字母
a 單字數目
Si 目前單字的第 i 個字母
ASC(asc) 將 Si 轉換為應的 ASCII 碼
大寫編碼(小寫編碼)

表 5：hashing function

圖 8 是利用 Q1 查詢子句來對 D1 檔案做資料的查詢，結果顯示出，mark-DOM 與 s-DOM 皆於 2bytes 時就成隱定的成長，但 mark-DOM 的效率卻高於過 s-DOM 非常多，s-DOM 所提取的節點高於 mark-DOM，但不會相差太大，本論文所提的方法效率約提昇了 85%。圖 9 之實驗結果，是採用 Q2 查詢子句來對 D1 檔案做查詢，由此結果可得知，本論文所提方法(mark-DOM)與 s-DOM，在以 1byte 為編碼大小時，mrak-DOM 所提取的節點數遠遠小於 s-DOM，且於 2bytes 為編碼大小時就趨近於隱定的成長；而 s-DOM 卻要在以 4bytes 為

編碼大小時才達到隱定的成長我們的方法提昇 50.6%的效率。圖 10 是採用 Q3 查詢子句對 D2 檔案做查詢，此結果顯示 mark-DOM 與 s-DOM 並無很大差異，不論編碼大小如何，成長皆不會影響太大，都成隱定性的成長，但是由圖可得知，s-DOM 效率雖然不至於很差，但仍舊是不如 mark-DOM，效率約改善了 25.3%。而圖 11 是採用 Q4 查詢子句對 D2 檔案做查詢，結果顯示 mark-DOM 於 2bytes 時就成隱定成長，且平均所捉取節點數皆小於 s-DOM 約 4 萬個節點左右，效率約成長 44.2%。

圖 12、13、14、15 是利用 2 個 hash function(2-hash) 計算編碼與 1 個 hash function(1-hash)做查詢，分別的比較，在這四個圖中，我們可得知，利用 2-hash 與 1-hash 比較，可將效率再改善約 30.8%。

圖 12 是 1-hash 與 2-hash 兩個不同的編碼方式， 利用 Q1 查詢 D1 的資料，的結果，利效可從 1-hash 的方式再改善約 37.9%。圖 13 也是利用 1-hash 與 2-hash 兩個不同的編碼方式，而利用 Q2 查詢 D1 的資料。效率可再改善約 33.8%。圖 14 是 1-hash 與 2-hash 兩個不同的編碼方式， 利用 Q3 查詢 D2 的資料的結果，率效可從 1-hash 的方式再改善約 20.7%。圖 15 也是根據 1-hash 與 2-hash 兩個不同的編碼方式， 利用 Q1 查詢 D1 的資料，的結果，利效可從 1-hash 的方式再改善約 31.1%。

由以上的實驗中我們可得知，在本論文所提之 mark-DOM 的效率是高過於 s-DOM，以穩定性的觀點來看，mark-DOM 通常只要以 2 個 bytes 的編碼儲存空間，就能夠讓結果達到一定的效率水準，且成穩定性的成長。就編碼的方面來觀查，本論文所提出的方法，只要用一個 bit 來註記編碼後的結果，相較於其它類似 ASCII 的多個 bit 的編碼來比較，重覆性是更低，更容易達到以較小的儲存空間來表示，而且低重覆性及高效率的成果。

二、編碼數增加與傳統方法的比較說明

本論文所提出方法與傳統之 s-DOM 在編碼上的差異，有著些許的不同，在 s-DOM 的編碼是以一類似 ASCII 的編碼方式。舉個例子，若有一個節點為 student，以 8 - bit 為編碼空間大小，經過 s-DOM 的編碼後值為 10011101；而本論文方法 mark-DOM 編碼後值則為 00100000。本論文提出之方法，在 8-bit 的編碼空間中，只有其中一個 bit 會是 1，其餘皆為 0，相反的 s-DOM 會產生多個 1。因此對於 8-bit 的編碼大小空間中，在向上記錄的步驟裡(參考第四章第二節)，s-DOM 當然會比 mark-DOM 更容易將全部的空間都填滿(全部都為 1，如 11111111)，這是無庸置疑，尤其當子樹中，其中一個子節點，其編碼值有 1 產生，且佔了 n-bit 的編碼空間的 3 分之 2 以上，其效率大大的下降(如 11101011，編碼值產生 1 的部分為 8bit 的 4 分之 3)。

若在編碼空間數提高時，s-DOM 仍然存在上述容易填滿的問題，因為在 s-DOM 的編碼方法上，還是會產生多個 1(最差狀況可能皆為 1)，不會因為編碼大小來改變編碼方式。相反，Mark-DOM 還是只會有一個 1 存在這編碼中，其餘皆為 0。因此在效率上遠遠大於 s-DOM。

陸、結論

在所有的 XML 文件中，我們都可以樹狀結構來看它，而一般而言，我們可以把它的節點視為一物件，且將它儲存於半結構化的資料庫中，當我們要查詢 XML 資料時，我們可以透過從資料庫中擷取節點，且當索引(index)無法使用時，效果更是彰顯。

在本論文中，使用了以雜湊函數為基礎的編碼註記查詢方法，讓我們在擷取節點時，能以過慮的方法，來檢驗我們所要查詢的條件內容，是否存在於目前節點下的子樹之中，此方法不但可以大大的裁減掉不必要的拜訪過程，且可以提昇查詢的效率及時間。

XML 為當今興新的研究方向，在查詢的技術上更是一項很大的研究空間，目前為止仍無一定的標準，眾說紛紜，且本論文採 W3C 所提之規則路徑表示法，再透過本論文所提出之 mark-DOM 方法來達到最佳化的查詢，在效率上也具有高度的發展空間。實驗結果顯示，我們所提的方法比其它方法來的快速且有效。

演算法一

```
1   node-目前捉取的節點
2  signature (XML tree)
3   {
4      fetch node (root_node)
5    while (node not null)
6    if (node is edge)
7     {
8        If (node no brother)
9          return to parent node
10        else
11          fetch (brother node)
12    }
13    else
14       fetch (child node)
15  }
```

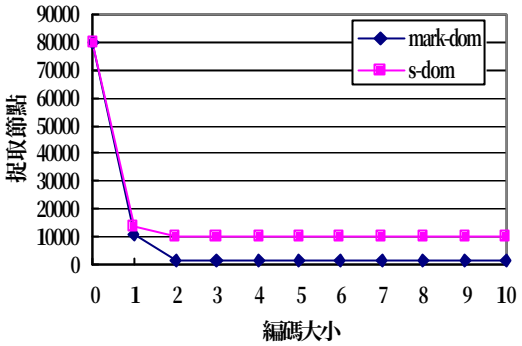


圖 8：利用 Q1 來查詢 D1 資料

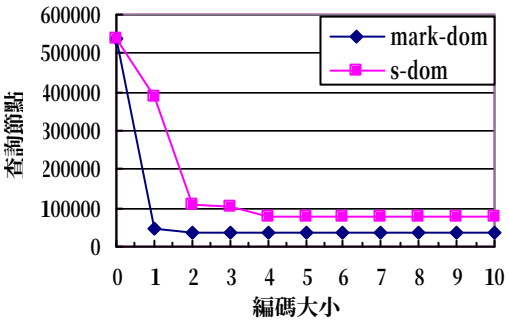


圖 9：利用 Q2 來查詢 D1 資料

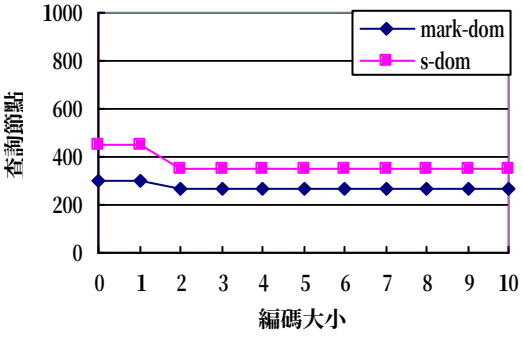


圖 10：利用 Q3 來查詢 D2 資料

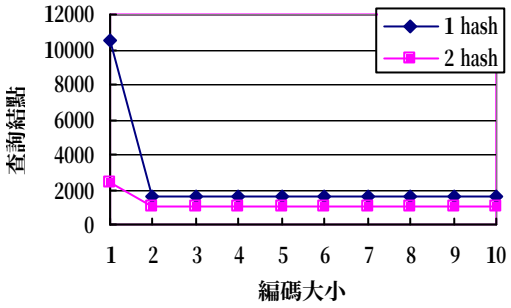


圖 12：利用 1-hash 與 2-hash 來比較，以 Q1 來查詢 D1 資料

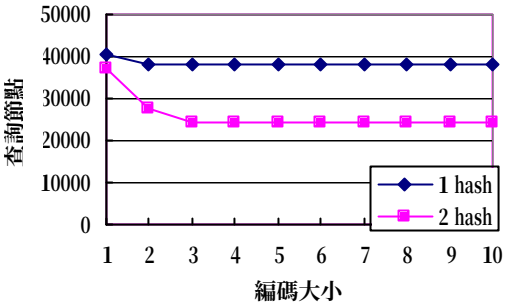


圖 13：利用 1-hash 與 2-hash 來比較，以 Q2 來查詢 D1 資料

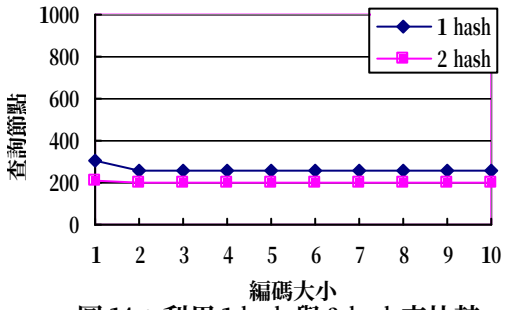


圖 14：利用 1-hash 與 2-hash 來比較，以 Q3 來查詢 D2 資料

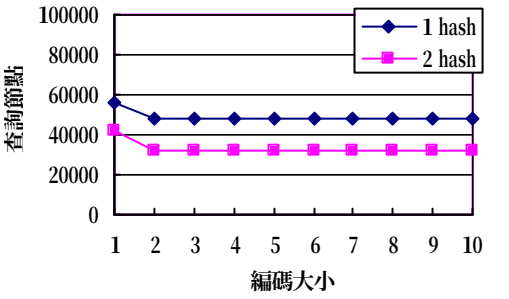


圖 15：利用 1-hash 與 2-hash 來比較，以 Q4 來查詢 D2 資料

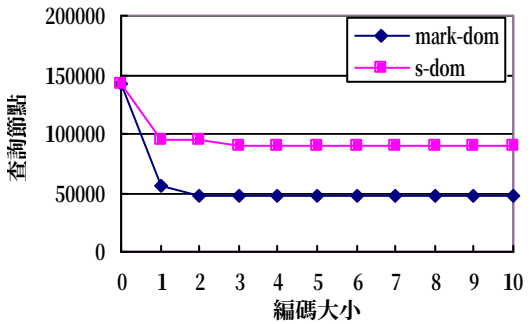


圖 11：利用 Q4 來查詢 D2 資料

參考文獻

[1] Sangwon Park, Hyoung-Joo Kim, “A New Query Processing Technique for XML Based on Signature”, IEEE. 2001.

[2] A. Zisman, “An overview of XML”, Computing and Control Engineering Journal, August 2000.

[3] http://www.w3.org/TR/2000/REC-xml-20001006

[4] For technical reports, working drafts, recommendations, and specifications of XML and related technologies, see the World Wide Web Consortium(W3C)Web

[5] S. Abiteboul. "Querying Semistructured Data. International Conference on Database Theory", Jan. 1997.

[6] P. Buneman, "Semistructured Data", ACM SIGACT-SIGMODSIGART Symposium on Principles of Database Systems, May 1997.

[7] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, and J. WiDOM, "Lore: A Database Management System for Semistructure Data", SIGMOD Record, 26(3), 9 1997.

[8] eXcelon, "An XML Data Server For Building Enterprise Web Applications", http://www.odi.com/products/white-papers.html, 1999.

[9] P. Valduriez, "Join indices," ACM Transactions on Database Systems (TODS), 1987, pp. 218-246.

[10] Alin Deutsch, Mary Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu, "Aquery language for XML", In proceedings of the 8th International World Wide Web Conference, pages 77-91, Toronto, Canada, May 1999.

[11] Stefano Ceri, Sara comai, Ernesto Damiani, Piero Fraternali, Stefano Paraboschi, and Letizia Tanca, "XML-GL: A graphical langeage for querying and restructuring XML documents", In proceedings of the 8the International World Wide Web Conference, page 93-109, May 1999.

[12] GMD-IPSI. GMD-ISPI XQL Engine, http://xml.darmstadt.gmd.de/xql, 2000.

[13] James Clark and Steve DeRose, "XML Path Language(Xpath) version 1.0 w3c recommendation", Technical Report REC-xpath-19991116, World Wide Web Consortium, November 1999.

[14] Quanzhong Li, Bongki Moon, "Indexing and Querying XML Data for Regular Path Expression", Proceedings of the 27th VLDB Conference, Roma, Italy, 2001.

[15] Sandeepan Banerjee, Vishu Krishnamurthy, Muralidhar Krishnaprasad, Ravi Murthy, and Oracle Corporation. "Oracle8i-The XML enabled Data Management System", IEEE, 2000.

[16] Josephine Cheng, Jane Xu, amd IBM Santa Teresa Laboratory, "XML and DB2", IEEE, 2000.

[17] Litwin. W, "Virtual Hashing: A Dynamically Changing Hashing", Proc. 4th Conference. On VLDB, West Berlin, Sept. 1978, PP.417-523.

[18] Fagin. R, Nievergelt J, Pippenger N, and Strong H.R, "Extendible Hashing-A Fast Access Method for Dynamic Files", ACM Trans. Database System 4,3(Sept. 1979), PP.315-344.

[19] IBM Alomaden Research Center, "A Signature Access Method for the Starburst Database System".

[20] Sangwon Park, Hyoung-Joo Kim, "SigDAQ: an enhanced XML query optimization technique".

[21] Michael Barg, K.Wong, "Structural Proximity Searching for Large

Collections of Semi-Structured Data ", CIKM'01 November 5-10,2001, Atlanta, Georgia, USA.

[22] W3C Extensible Markup Language(XML) 1.0, http://www.w3.org/TR/1998/REC-xml-19980210.

[23] A. Silberschatz, H. Korth, and S. Sudarshan, Database System Conecepts, McGraw-Hill, 2001.

[24] Patel, D. Srivastava, Y. Wu, S. Al-Khalifa, H. V. Jagadish, N. koudas, "Structural Join: A primitive for Efficient XML Query Pattern Matching," In Proc. Of ICDE 2000.

[25] S. Al-Khalifa and H. V. Jagadish, "Multi-level Operator Combination in XML Query Processing," In Proc. Of ICDE 2001.

[26] H. Jiang, H. Lu, W. Warg and J. Xu-Yu, "Path Materialization Revisited: An efficient Storage Model for XML data," In Proc. Of ICDE 2001.

[27] A. Schmidt, M. Korsten, H. Windhouwer, F. waas, "Efficient Relational Storage and Retrieval of XMl documents", Proceedings of the 27th VLDB Conference, Roma, Italy, 2001.

[28] B.H. Boolm, "Space/time trade offs in hash coding with allowable errors", CACM,1970.

[29] W3C Document ObjectModel(DOM), http://www.w3.org/DOM/,2 ,2000.

[30] S. Abiteboul, D. Quass, J. McHugh, J. WiDOM and J. Wiener. "The Lorel Query Language for Semistructured Data", International Journal on Digital Library, 1(1),4 1997.

[31] P. Buneman, S. Davidson, G. Hillebrand and D. suciu. "A Query Language and Optimization Techniques for Unstructured Data", SIGMOD, 1996.

[32] V. christophides, S. Abiteboul, S. Cluet and M. Scholl, "From Structured documents to Novel Query Facilities", SIGMOD, 1994.

[33] DOM Chamberlin, Daniela Florescu, Jonathan Robie, Jrme Simon, and Mugur Stefanescu, "XQuery: A query Language for XML W3C working draft", Technical Report WD-xquery-20010215, World wide Web Consortium, February 2001.

[34] D. Florescu and D.Kossmann, "Storing and Querying XML Data using an RDBMS", Data Engineering Bulletin, 22(3), Setp. 1999.

[35] J.Mchugh, S. Abiteboul, R. Goldman, D. Quass and J. WiDOM, "Lore: A Database Management System for Semistructured Data". SIGMOD Record, 26(3), 9 1997.

[36] T. Shimura, M. Yoshikawa, and S. Uemura", "Storage and Retrieval fo XML Documents Using Object-Relational Database", DEXA, 1999.

作者簡介

吳光閔，國立交通大學資訊科學博士，現任南華大學資管理學系(所)副教授。

李建圖，南華大學資訊管理研究所碩士班研究生。

厲彥廷，南華大學資訊管理研究所碩士班研究生。