一個 XML 文件上的節省空間索引方法*

廖宜恩　　張媛萱
中興大學資訊科學系　　中興大學資訊科學系
ieliao@nchu.edu.tw　　s9456052@cs.nchu.edu.tw

摘要

本研究有別於傳統以加快查詢速度為主的索引機制，提出一個可以有效壓縮 XML 文件的索引架構，提供使用者進行多元化的查詢。主要基於 B-tree 建立一個適用於 XML 資料結構的索引機制，在編碼方面可以避免 false hits 的問題產生。在實驗的效能評估方面，本研究的索引空間與字串儲存體空間的總和，與原始的 XML 文件相比較大約節省近百分之三十左右。在查詢的效能方面，也較傳統的 XSLT 方式快速，換言之，本研究所提出的索引架構所提供的查詢反應時間介於使用者的可接受範圍之內。相關應用方面，本研究的實驗環境以 SAX 的方式進行剖析(parser)與驗證，符合 XML stream 即時(real-time)的特性；因此，本研究可以應用於類似於影像串流等環境底下，以及適用於低運算或是有限空間的硬體平台使用。

關鍵字：XML、Index、B-tree、Space-efficient Indexing method, XQuery, XPath

1. 前言

現今網際網路資料交換的統一標準，包含可擴展標示語言(Extensible Markup Language, XML[3])及 HTML 等語言。在 XML 查詢的歷程中，包括以 XQuery[7]、XPath 來進行查詢之外，目前相關的研究已經朝向建立 XML 資料索引的方向；由此可知，XML indexing 已成為另一個重要的議題。

本研究採取 Wang ／Meng method[4]提出的編碼及字串概念。為了加快選擇性節點的查詢速度，在符合查詢節點字串的篩選方面，利用字串儲存體（String_value B-tree）來提升查詢的效率。由於本研究的索引結構特性，在 parent – child(PC)與 ancestor-descendant(AD)的查詢，也提供具有效率的查詢速度。總而言之，本研究提出一個可以有效壓縮 XML 文件的索引，提供使用者查詢。本研究基於簡單的資料結構與編碼方式，就可以呈現文件原始的內容提供查詢。此索引架構可以應用於嵌入式與影像串流的環境底下。並不需要高運算能力、龐大的儲存空間，就可以運行此索引結構，此為本研究的主要貢獻。

2. 相關研究

在 XML 索引方面，在相關研究[2]中將其分為三大類 summary indexes、 structure join indexes[5][8]、sequence-based indexes[4][6]。目前許多研究從事 sequence-based 的研究。其概念是將 XML 文件與查詢的字串視為字串，進行子字串的比對，避免額外 join 的動作。以下將簡單介紹相關研究所提出的編碼方式。根據 ViST(Virtual Suffix tree)[6] 所提出的 sequence 表示一個 XML 文件的方式，可表示為<(D, ε) (A, D) (T, DA) (W, D) (T, DW)>。根據 Wang ／Meng method 提出的編碼方式利用 DFS 的搜尋方式來找尋下一個節點所建立而成的字串，並且將 XML 樹狀結構表達為<D, DA, DAT, DW, DWT>，此字串可以代表獨一無二的文件內容。

在處理 XML 文件除了將其轉化為字串外，還必須給予一個標籤 label 注記其編號、子孫的個數或範圍、所屬的文件號碼等資訊。因此，許多研究學

* This research was partially supported by the National Science Council, Taiwan, under contract no. NSC95-2221-E-005-114.

者在這方面有著許多不同的編碼方式問世。如同 ViST 所提出的 label 為 <X, SIZEx>，代表節點本身的號碼及其子孫的個數。除此之外，Wang ／Meng method 提出的 label 為($n^{\vdash}$,$n^{\dashv}$), 代表 n 的編號及其最後一個子孫的編號，以方便辨別節點之間的父子關係。

由於 ViST 雖是以字串為基礎的索引架構，但其編碼方式會產生 false alarms 與 false dismissals 等問題產生；因此，學者再提出 Wang ／Meng method 的編碼方式來解決 ViST 所發生的誤判問題。因此，本研究使用 Wang ／Meng method 提出的編碼方式建立索引所需的 label 數值，並且利用 B-tree 的資料結構建立索引。

基於 Wang ／Meng method 所轉換的子字串建立的樹狀結構及索引連結，缺乏考量節點重覆性的問題，當大量的 XML 文件轉換出來的字串(sequence)重覆性低的情況下，容易產生的索引架構的水平連結(head of horizontal links)的數目隨著 XML 文件的數量上升。因此，本研究考量節點重覆性的問題，提出以節點來儲存字串的編碼數值，將數值集中儲存以減少大量的連結空間產生。

3. 系統架構與演算法

3.1 XML 文件編碼方法

本研究使用的編碼方式是由 Wang ／Meng method 所提出的編碼方式。本研究應用其編碼方式，提出必須符合下列 **Definition 1: Labeling Constraint** 的 labeling schema。此概念主要為若兩個節點存在父子關係，則子節點的 label 區間必定被包含於其父節點的 label 區間。

Definition 1: Labeling Constraint

XML 文件的樹狀結構為 D，在 D 中包含 n 個節點。每一個節點給予三個數值，分別為節點本身的編號、節點的最後一個子孫節點的編號以及節點所屬 XML 文件的編號。假設 R_j, R_k 屬於 D 中的節點，並且 R_j 為 R_k 的父節點或祖先節點；則 R_j 的 label 數值為<$R_j^{\vdash}$, $lastR_j^{\dashv}$,docID>、R_k 的 label 數值為<$R_k^{\vdash}$, $lastR_k^{\dashv}$, docID>。R_j 與 R_k 的 label 數值必定存在一個關係，即為 $R_k^{\vdash} \in [R_j^{\vdash}, lastR_j^{\dashv})$、$lastR_k^{\dashv} \in (R_j^{\vdash}, lastR_j^{\dashv}]$。

除了上述的 labeling schema 之外，由於建立 XML 文件的索引是藉由轉化為字串的形式來處理，必須注意在轉化後的字串是否能獨一無二的表示 XML 文件的內容。因此，本研究的編碼節點的追蹤順序是採取深度優先搜尋(DFS) 來轉換 XML 文件內容的搜尋方法。

3.2 索引的建構方式

前文已說明相關的編碼方式與限制，本節將利用已編碼完成的資訊及關鍵值進行索引主體的建立。以下說明主要分為三部份來詳細說明整個索引架構。第一部份包括索引的關鍵值的選擇，第二部份包括說明節點的編碼順序與方法，最後，將前二部份的資訊一併放入 B-tree 中建立出索引架構。

本研究考量節點重覆性的問題，提出以節點來儲存字串的編碼數值，將數值集中儲存以減少大量的連結空間產生。因此，從 XML 檔案中搜尋各節點，建立一個 head of label <node , level>，見圖 1，<dblp,1>, <www, 2>, <article, 2>, <title, 3>, <url, 3>, <year, 3>, <author, 3>, <pages, 3>。其中，author, title, year 是重覆的，因此，只要建立一個即可。

由於轉換的對等性的考量，目前採用 DFS 的搜尋方法，分析(parser)XML 檔案，給予每個節點其 number。見圖 1，此 XML 文件的 sequence 為 <d, dw, dwt, dwu, dwy, dwa, da , daa, dat, dap, day> ，每個節點給予 label 為 <(1, 11,1) (2, 6,1) (3, 3,1) (4, 4,1) (5, 5,1) (6, 6,1) (7, 11,1) (8, 8,1) (9, 9,1) (10, 10,1) (11, 11,1) > 。

將上述所提及的 head of label、label of node 連結在一起，只要 label 的節點名稱相同的，雖然字串

(sequence)不同，但可以放在相同的 head of label 裡。見圖 1 的 www, article 其分支皆有 author，因此，將這兩個分支下的節點的 label 與 head of label <author, 3>連結(linking)在一起。將上述所述的 label 與其 head of label 連節在一起後，取 head of label 為 key，取 label 為資料數值(value)，放入 B-tree 中，見圖 2 所示。

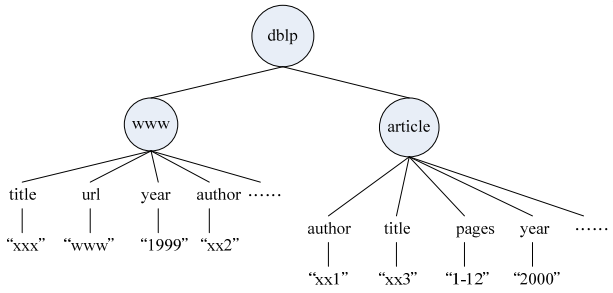


圖 1：XML File

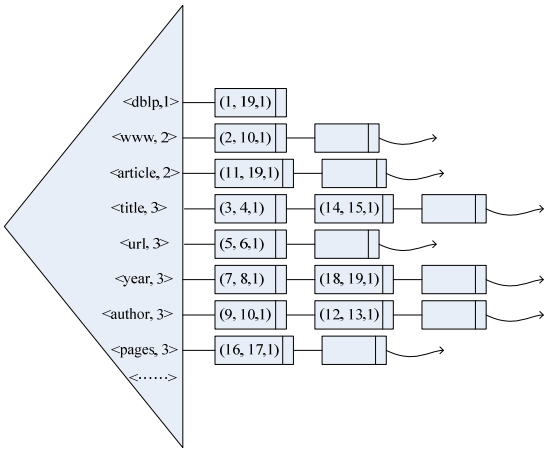


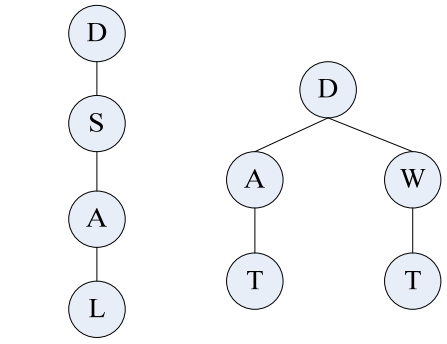
圖 2：Index structure example

3.3 查詢比對方法與限制

本研究處理 XML 檔案查詢的部份，主要提供選擇性節點、PC、AD 等分支查詢。當使用者輸入符合 XQuery 與 XPath 的查詢字串，例如：查詢字串為 //D//S//A//L，見圖 3 (a)。或者是 //D[.//A//T]//W//T，見圖 3 (b)。當查詢語句為查詢分支時，見圖 3 (b)，其父子關係必須符合 **Definition 2: Constraint Match** 的條件。

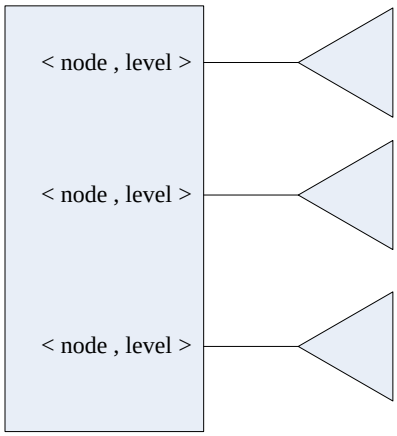
先從 leaf 的節點的 label 數值中開始搜尋符合的父節點，再從分支的另一個子節點的 label 數值比對已符合另一分支的父節點，最後再將符合 twig tree 的父節點等資訊傳回，以避免 join 兩個查詢條件的父節點，以節省搜尋的時間。倘若，查詢字串為圖 3 (a)，即為 root 至 leaf 的查詢字串，只需要從 leaf 節點進行比對，最後再將符合的傳回即可。

為了處理選擇性節點的查詢，會產生大量的字串比對。本研究利用 B-tree 結構建立 String_value trees，見圖 4，將字串內容放入 B-tree 中，再儲存對應的 label 數值，以減少不符合的節點比對次數。因此，本研究提出的索引架構再與 String_value trees 中所儲存的 content-text()字串進行對應的動作，加速選擇性節點的查詢速度。



(a) supply chain item (b) twig

圖 3：查詢字串 twig tree



String_value tree

圖 4：String_value trees

Definition 2: Constraint Match

在 XML 文件檔案為 D 及查詢語句為 Q 之間，對應相關連的查詢節點，給予一個比對模組為 m(·)。在 m(w)中所代表的是，當在索引關鍵值 w 節點對應出來的 w'為在比對查詢分支中的 w，即代表 w=w'。此比對模組 m(·)是存在於條件式 f 中，滿足條件式 f 中包含的 **Definition 1** 的條件，則可證明其父子關係。當兩對父子關係存在相同的父節點 label 數值時，即代表其父節點是相同的。亦即下列條件成立：

1. m(w) = w' $\Rightarrow$ w = w' ;
   m(t) = t' $\Rightarrow$ t = t' ;
   m(a) = a' $\Rightarrow$ a = a' ;
   m(w) = w" $\Rightarrow$ w = w" ;
2. f (w , t) = f (m(w), m(t)) = f (w', t')
   f (w , a) = f (m(w), m(a)) = f (w", a')

Based on **Definition 1.** f (w', t') 及 f (w", a')比對其 label 數值

3. f (w', t')中 w'的 label 數值 = f (w", a')中 w"的 label 數值

4. 系統實作與實驗結果

4.1 實驗資料集

DBLP 資料集是一個廣為人知的資訊科學的參考書目資料庫，此資料庫被大量的運用在比較 XML index 方法的基準點。每一筆在 DBLP 的資料記錄皆為一個簡單的樹狀結構。本研究下載的 DBLP 資料 134MB 內含有 3332130 個節點(elements)、404276 個屬性(attributes)、最大深度為 6。XMark 資料集是結合許多子結構的文件，如 item, person…等。本研究下載的 XMark 資料 113MB 內含有 1666310 個節點(elements)、381870 個屬性(attributes)、最大深度為 13。

4.2 索引空間評估

在索引空間耗用的衡量方面，主要經由 XML 索引架構、字串儲存體及原始的 XML 文件等三個指標來進行空間耗用的評估。首先，XML 索引架構基於 B-tree 的資料結構所建立，此索引結構在此稱為 Label_Index。本研究將 XML 索引架構輸出為 .txt 檔，取得其所佔空間大小，並將此空間大小稱為 Label_Index_Size。其次，字串儲存體主要儲存節點的內容(content)字串，本研究將其以 B-tree 的結構儲存，此字串儲存體在此稱為 String_value B-trees。本研究將字串儲存體輸出為 .txt 檔，取得其所佔空間大小，並將此空間大小稱為 String_repository_Size。最後，由於本研究所使用的 XML 檢索系統，並不需要原始的 XML 文件，是將原始的 XML 文件轉化為 XML 索引架構與字串儲存體二部份，因此，利用原始的 XML 文件的空間大小與上述所提及的 XML 索引架構、字串儲存體進行節省空間的比較。

衡量節省空間的方法是利用上述所提及的三個指標，配合公式(4-1)進行相關的計算，進而了解其節省的比例，見(4-1)

$$\left(XML\_file\_Size - LIS - SRS\right) \Big/ XML\_file\_Size \ * \ 100\%$$

$$LIS : Label\_Index\_Size$$
$$SRS : String\_repository\_Size \qquad (4\text{-}1)$$

本研究在資料集 XMark 所建立的 Label tree 的索引大小，見圖 5 所示，當文件內含 82000 個節點其索引大小為 1.15MB，當文件內含 521000 個節點時，其索引大小為 7.62MB 及字串儲存體(String repository)其空間約為 17MB，兩者相加大約為 25MB。由此可知，本研究的索引空間(Label index)大約佔原本的 XML 文件 (XML file size)的百分之二十三左右。除此之外，索引空間與字串儲存體的空間相加，經由空間節省比例的公式(4-1)計算得知，與原始的 XML 文件相比較大約節省近百分之三十左右。見圖 6，可知本研究提出的索引架構建立的時間是以線性成長，並且在文件內含 521000 個節點時，建立索引的時間，需要 6.32 分鐘。在資料集 DBLP 方面，如圖 7、圖 8 所示。

4.3 查詢效能評估

本節對於建立的索引架構及檔案系統，進行的查詢效能測試。查詢效能的測試包括 Xmark、DBLP 等二種資料集的測試，對於 Xmark 的資料集是

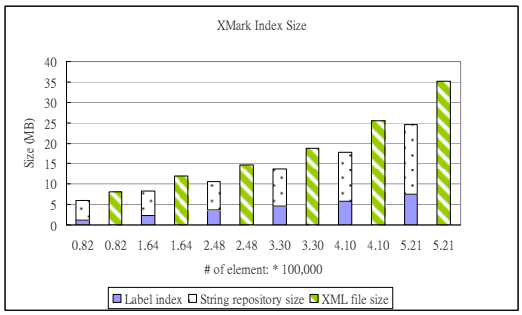


圖 5：Index Size on XMark

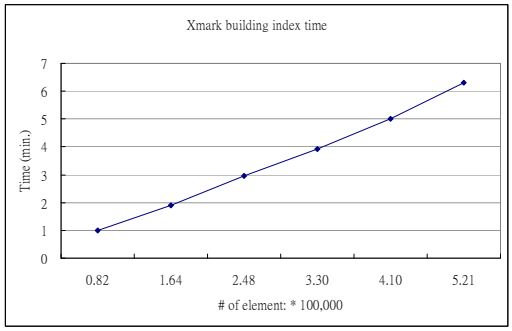


圖 6：Time of Building index on XMark

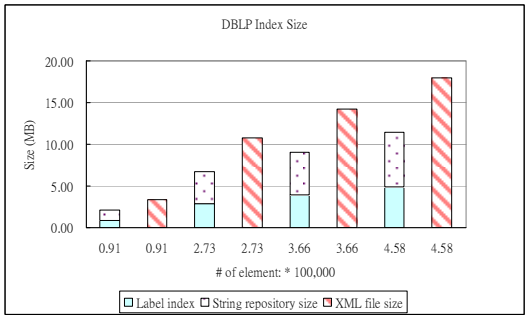


圖 7：Index Size on DBLP

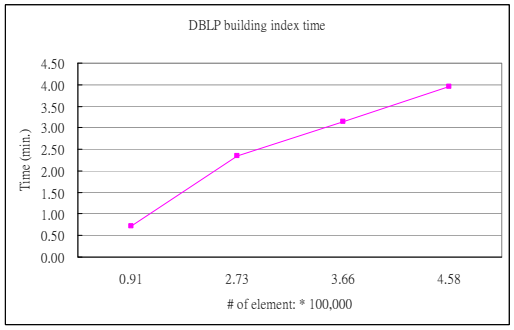


圖 8：Time of Building index on DBLP

以 Q1~Q5 的查詢條件進行測試，對於 DBLP 的資料集是以 Q6~Q8 的查詢條件進行測試，見表 1。

在查詢效能方面，必須遵循的評估方法，進行以下的查詢條件等實驗測試。經由本研究提出的索引結構進行查詢，查詢效能的測試結果見表 2。

本研究所提出的 XML 檢索系統以三個項目呈現每次查詢的情況，見中所示前十筆的平均搜尋時間、回應時間及符合搜尋條件結果的數量。由於有時符合結果的數量會十分龐大，因此搜尋時間會較長，因此，取前十筆的搜尋時間為其平均搜尋時間。最後，中的回應時間為符合單一查詢條件的結果，包括搜尋完畢且傳回檔案系統中實際的資料的總時間，見表 2。

表 1：Sample queries over DBLP and XMark datasets

Query	**DataSet**
Q1 : //item[location='United States']/mail/date[text='08/05/1999'] Q2：//item//date[text='05/27/1999'] Q3：//item/location Q4：//item//mailbox//mail//text//bold Q5： //people//person/city[text=“Portland”]	XMark
Q6：//inproceedings/title Q7： //inproceedings/author[text='Hyeong In Choi'] Q8：//author[text='David']	DBLP

由於 XSLT 可以用來顯示及轉換 XML 文件。利用 XSLT 擷取 XML 文件中的資料，將擷取出來的資料重組成為另一個檔案。利用 XSLT 配合 XPath 的語法將的查詢條件進行轉換為符合查詢條件的 XML 文件。因此，利用本研究提出的索引架構及檔案系統與 XSLT 進行比較。

在平均搜尋時間的比較方面，見圖 9，以 Q2、Q3、Q7、Q8 的查詢條件下，本研究提出的方法優於傳統的 XSLT 查詢方法，換言之，本研究所提出的索引架構所提供的查詢反應時間介於使用者的可接受範圍之內。

表 2：Our approach on Q1 ~ Q8 (Time in sec.)

Our approach			
	Avg. top 10 search time (sec.)	Response Time (sec.)	Result number
Q1	0.031	16.187	4
Q2	0.031	16.016	4
Q3	0.047	109.984	3156
Q4	10.782	6.266	4
Q5	0.891	17.906	21
Q6	2.016	847.735	78
Q7	0.002	0.016	2
Q8	0.031	46.093	10

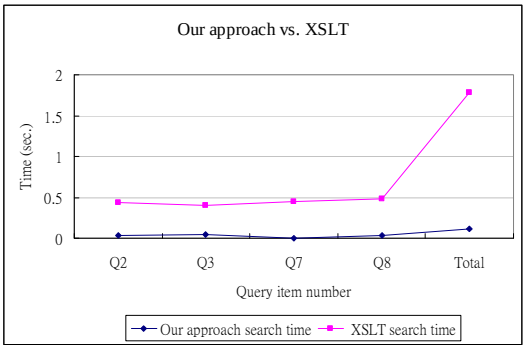


圖 9：Our approach vs. XSLT (Time in sec.)

5. 結論

本研究提出一個可以有效壓縮 XML 文件的索引，提供使用者進行選擇性節點的查詢與 parent – child (PC) 與 ancestor-descendant (AD) 的查詢。在實驗的效能評估方面，本研究的索引空間與字串儲存體空間的總和，與原始的 XML 文件相比較大約節省近百分之三十左右。在查詢的效能方面，也較傳統的 XSLT 的查詢方式快速，換言之，本研究所提出的索引方法之查詢反應時間是在使用者可接受範圍之內。因此，此索引架構可應用於嵌入式與影像串流的環境底下。

由於本研究提出的 XML 檢索系統，目前在查詢方面尚未支援如 OR、AND 的查詢語句以及更新索引的部份。未來可朝向此方向研究，以利增加此索引機制的完整性。除此之外，本研究也可以結合串流即時的特性與 MPEG-7 的資料格式，延伸出適用於 MPAG-7 Stream[1][9]的索引架構等新議題。

參考文獻

[1] Andrea Kofler-Vogt, Harald Kosch, Joerg Heuer, and Andr´e Kaup, “BeTrIS—An Index Systemfor MPEG-7 Streams,” Hindawi Publishing Corporation EURASIP Journal on Applied Signal Processing, Volume 2006, pp. 1–11.

[2] Barbara Catania, Anna Maddalena, Athena Vakali, "XML Document Indexes: A Classification," IEEE Internet Computing, vol. 9, no. 5, Sept/Oct, 2005, pp. 64-71.

[3] Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation 6 October 2000, http://www.w3.org/TR/2000/REC-xml-20001006 .

[4] Haixun Wang and Xiaofeng Meng, “On the Sequencing of Tree Structures for XML Indexing,” in Proc. of ICDE, 2005, pp.372-383.

[5] H. Jiang, H. Lu, W. Wang, and B. C. Ooi, “XR-tree: Indexing XML Data for Efficient Structural Joins,” in Proc. of IEEE ICDE, 2003, pp. 253-264.

[6] H. Wang, S. Park, W. Fan, and P. S. Yu, “ViST: A Dynamic Index Method for Quering XML Data by Tree Structures,” in Proc. SIGMOD, 2003, pp. 110 – 121.

[7] S. Boag, D. Chamberlin, M. F. Fernandez, D. Florescu, J. Robie, and J. Simeon, “XQuery 1.0: An XML Query Language,” http://www.w3.org/TR/xquery/, 2003.

[8] Simon Sheu and Nigel Wu, “XCut: Indexing XML Data for Efficient Twig Evaluation,” in Proc. of ICDE, 2006, pp.127.

[9] Yi Chen, Susan B. Davidson, Yifeng Xheng , “An Efficient XPath Query Processor for XML Streams,” Proceedings of the 22nd International Conference on Data Engineering (ICDE’06), IEEE, 2006 ,pp.1-12.