並行異動 XML 文件內的元素資料

陳靖國
朝陽科技大學資訊管理系
jkchen@cyut.edu.tw

呂冠暐
朝陽科技大學資訊管理系
s9514637@cyut.edu.tw

摘要

在本論文中，我們提出適用於多使用者環境的 XML 文件存取的新增和刪除功能的兩個並行控制演算法。利用一條路徑，內含代表元素名稱的標籤名稱，新增(或刪除)演算法會依照路徑所指定的位置，新增(或刪除)指定的元素。當走訪 XML 資料索引樹時，使用廣度優先搜尋(breadth-first search)方式可避免向上回溯可能產生死結(dead lock)的問題。並行控制主要技術採用鎖定法(locking)，鎖的種類只使用兩種，即分享鎖（share lock）與排他鎖（exclusive lock），確保多人存取不會發生錯誤。以成對鎖定(lock-coupling)協定保證資料存取操作(operation)走訪的路徑上的節點都是正確無誤的。

關鍵詞：並行控制，XML，Locking，成對鎖定，廣度優先搜尋。

Abstract

In this paper, we propose an insertion and a deletion concurrency control algorithms for accessing the XML document in a multi-user environment. Given a path includes tag-names and/or attributes, the insertion/deletion algorithm can insert/delete the specific node into/from a specific location in the path. Breadth-first search is used to avoid the deadlock problem caused by the feeback propagation when visiting the XML index tree. The concurrency control method is locking. Two lock types, share lock and exclusive lock, is used to guarantee multiple users to concurrently access the same data without any error occurrence. The lock-coupling protocol is used to ensure the nodes in the path visited by the operation are correct.

Keywords: Concurrency Control, XML, Locking, Lock-coupling, Bread-first Search.

1.前言

配合 DTD (Document Type Definition)或者 XML Schema，XML (eXtensible Markup Language)[16]被使用在很多領域中，例如電子商務，資料交換，資料倉儲等等 [2, 9, 11, 12, 13, 14, 15]。由於 XML 文件的數量迅速增加，為了使 XML 資料的管理和存取更容易，使用資料庫管理系統來管理 XML 資料是有必要的[5, 11, 18, 19]，同時也需要一個並行控制機制來維護多使用者同時存取一份 XML 文件時做有效的控管。若一個 XML 被眾多操作同時存取而沒有適當地加以控管時，一些難以預期的錯誤，像是更新失誤、錯誤讀取與加總錯誤[4]，是可能會發生的。

在本文中我們提出兩個並行控制演算法，新增與刪除演算法，來更新 XML 文件內容。利用含有標籤名稱的參數路徑，新增（或刪除）演算法會新增（或刪除）一個指定的元素。為了達成並行控制目的，我們使用的鎖定方法（Locking），只需利用兩種鎖加以控管，其細節描述如下。共享鎖（share lock，簡稱 s-lock）[3]被使用在當操作要讀取節點資料之前所提出的許可要求。共享鎖可以跟其他的共享鎖共存相容，這表示一個元素可以被一個以上的操作用共享鎖加以鎖住。排他鎖（exclusive lock，簡稱 x-lock）[3]被使用在當操作要變動一個節點的內容之前所提出的許可要求。排他鎖與其他的鎖不相容，一個節點在任何時間只可以被一個操作用排他鎖加以鎖住，此時其他操作無法對該節點下任何鎖定。任何操作需要存取任何節點之前，必須要先鎖住該節點，當操作處理完該節點之後，必須儘快加以釋放以供其他操作存取。除了鎖定法與兩種鎖之外，我們也用了廣度優先搜尋技術與

成對鎖定協定[3]來走訪 XML 文件的索引樹，以便快速正確地找出目標節點。

2.相關技術

XML 規格是 SGML [14]的子集合，一份 XML 資料有兩種類型的規範 [16]。第一個類型稱為良好格式(well-formed)，第二種類型稱為有效的(valid)。一份正確的 XML 文件至少必須是符合良好格式的要求，而有效的 XML 文件則必須通過 DTD 或 XML schema 的驗證。DTD 的範例與有關的 XML 資料請參考圖 1 的(a)和(b)。

```
<?xml version="1.0" encoding="Big5"?>
<!DOCTYPE booklist[
<!ELEMENT booklist (book_type+)>
<!ELEMENT book_type (book*)>
<!ATTLIST book_type type CDATA
#REQUIRED>
<!ELEMENT book
(title,author+,publisher+,price+,year+)>
<!ATTLIST book ISBN CDATA #REQUIRED>
<!ELEMENT author (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT year (#PCDATA)>
]>
```

圖 1(a)、XML 附屬 DTD

```
<?xml version="1.0" encoding="Big5"?>
<booklist>
    <book_type type="computer science">
        <book ISBN="115225722">
            <title> Database System </title>
            <author>Luce White</author>
            <publisher>Mcgraw Hill</publisher>
            <price>800</price>
            <year>2006</year>
        </book>
        <book ISBN="552489547">
            <title> information system management
            <author>Bill Wu</author>
            <publisher>Flag</publisher>
            <price>600</price>
            <year>2007</year>
        </book>
    </book_type>
    <book_type type="novel">
        <book ISBN="514569874">
            <title>Run</title>
            <author>Edwin King</author>
            <publisher>C.C.</publisher>
            <price>320</price>
            <year>2006</year>
        </book>
        <book ISBN="458474257">
            <title>Right now</title>
            <author>Jean Mear</author>
            <publisher>D.K.</publisher>
            <price>180</price>
            <year>2005</year>
        </book>
    </book_type>
</booklist>
```

圖 1(b)、XML 文件範例

並行控制提供一個可靠的機制，讓資料庫可以在多使用者環境下運作。並行控制主要方法有 3 種：鎖定法、時間戳記法和樂觀法[4]。鎖定法是用一個狀態表示一個資料單元可否存取，當資料單元被一個操作 T 鎖定時，T 可以存取該資料單元，其他操作可能不被允許存取該資料單元，當資料單元狀態為非鎖定時，任何操作都可以對它提出鎖定要求，通過之後再執行存取動作。此作法不允許兩個以上操作對同一個資料單元同時進行異動的動作，但允許同時讀取的動作。本論文使用兩種鎖，分享鎖(s-lock)、排他鎖(x-lock)，當節點被分享鎖鎖住時，表示允許其他操作也可以讀取這個節點的資料，但是不可以更動節點內容。當節點被排他鎖鎖定後，表示不允許其他操作去存取該節點內容。時間戳記法是利用資料庫管理系統對每一個操作給予唯一的識別碼，用來判斷操作的執行先後順序以決定該操作是否可以存取某個具有時間戳記的資料單元。樂觀法是允許任何操作都可以直接存取資料單元而沒有任何限制，只在操作結束的時候才作驗證工作。我們選用鎖定法來設計並行控制機制，是因為鎖定法是最受歡迎的和容易實作[6, 8, 10]。在資料庫中，常有建立索引裝置來提昇存取的速度，所以一份 XML 文件也可以建立一個輔助的索引裝置，例如圖 2 為圖 1(b)的 XML 文件的索引樹結構，採用 m-way 樹架構。

在索引樹中，每個節點包括 n+3 個指標單元，亦即標籤名稱（Tag_name）、內容（Content）、屬性（Attribute），與所有的子節點 Child[1…n]，n 代表某節點的子節點個數，”標籤名稱”指標是指向一個 XML 元素的標籤名稱。”內容”指標是指向一個 XML 元素的元素內容。”屬性”指標是指向一個 XML 元素的屬性與其值。Child[1…n]陣列指標是指向這個元素節點的所有子元素節點。

3.並行控制演算法與範例

3.1 新增演算法

在本文之前，作者曾發表過類似的新增並行控制演算法[1]，輸入的參數只包含標籤名稱，不包含新增所在位置的路徑資料，由於 XML 中，重複出現的標籤名稱是被允許的，造成搜尋目標節點時只會找到第一個符合條件的位置新增資料，但不一定是使用者想要的正確位置，因此本文改良前述的演算法，參數中加入路徑資料，如果路徑上有附加屬性值，則利用屬性值進一步來辨識具有相同標籤名稱的特定節點。如果沒有附加辨識用的屬性值，當符合標籤名稱之節點不止一個時，則以第一個找到的為目標。

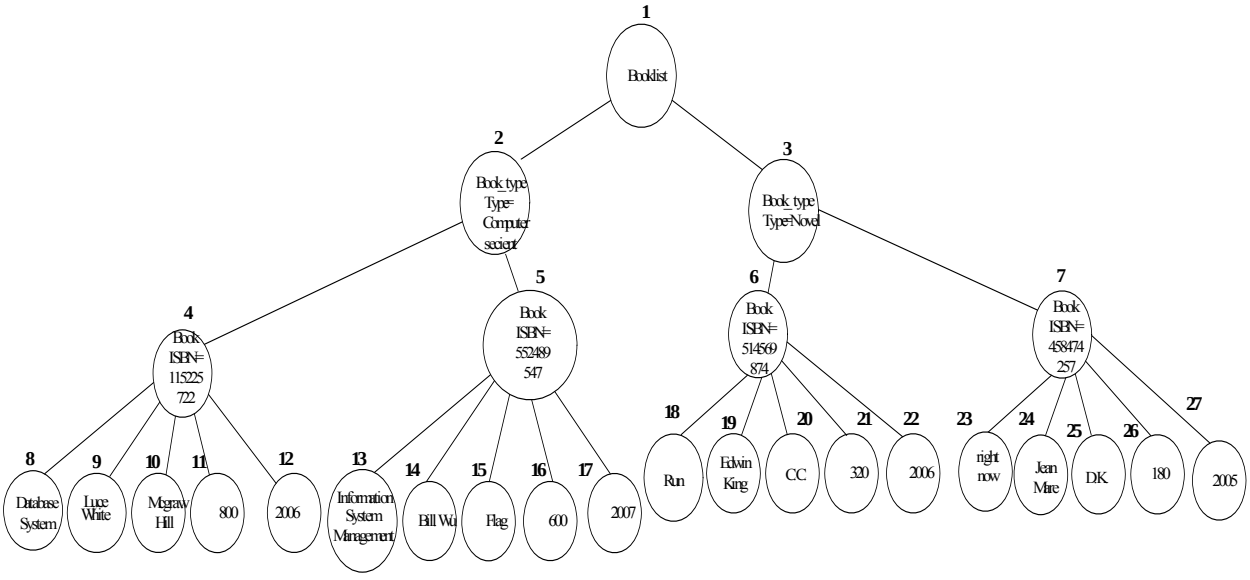

圖 2、XML 文件索引結構圖

演算法有三個參數，一為 XML 索引樹根節點 T_X，一為路徑參數 *Path*，一為新增元素節點 *Ins_Element*。*Path* 除了描述一條路徑的節點標籤名稱外，還可能包含辨識同名節點之間差異的屬性值資料，這些標籤名稱與屬性值我們用一個叫 *TNS(*a tag_name string*)*字串來表示，每一個 *TNS* 必須含有一個標籤名稱，屬性名稱與值則是選擇性的可有可無，其格式為：*tag_name* 或 *tag_name*[attribute_name=attribute value]。新增演算法將依照 *Path* 描述的路徑，在索引樹中找到目標節點，並將 *Ins_Element* 新增為目標節點的子節點。在走訪到目標節點之前一路上所經過的節點將會被依序用分享鎖鎖住，當走訪後，這些節點將被儘快地釋放，以提高系統並行度。當找到目標節點要執行新增動作前，目標節點上的分享鎖將被轉換成排他鎖，以防止目標節點被其他操作異動或讀取，並於新增動作結束後，釋放目標節點的排他鎖。圖 3 為新增演算法的流程圖，使用到的變數之用途說明如下。*Queue* 是一個佇列(queue)用來儲存廣度優先走訪時將被走訪的節點。N_{temp} 是目前要被走訪的節點，N_{target} 為一目的節點，當作新增節點的父節點。*Flag* 是一個旗標用來判別階層的變化，*Flag* 的起始值為 '0'，當走訪時，從上一層轉到下一層時，我們將 *Flag* 設定為 '1'，作為辨識用，並從 *Path* 中取出新的項目。詳細的新增演算法如下。

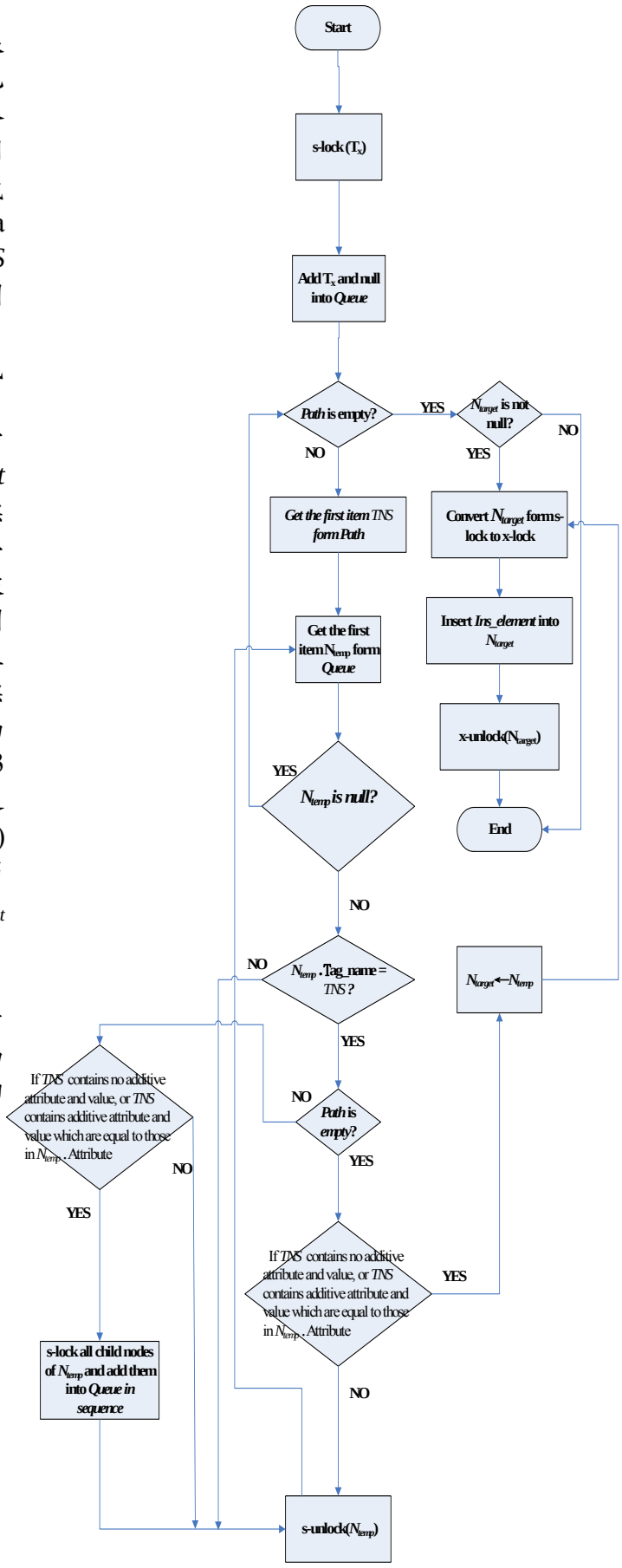

圖 3、 新增演算法流程圖

```
Algorithm Insert(Tx, Ins_Element, Path)
Input:
NODE POINTER Tx;
ELEMENT Ins_Element;
STRING Path;
Begin
QUEUE Queue;
NODE POINTER N_temp;
STRING TNS; //An tag_name string abstracted form Path//
NODE POINTER N_target;
INTEGER Flag;
01.  Flag ← 0; //0: incorrect i/p tag name//
02.  N_target ← null;
03.  s-lock(Tx);
04.  add_node (Tx, Queue);
05.  add_node (null, Queue); // null is a dummy node, to separate nodes at different levels.//
06.  while Path is not empty, do
07.  TNS ← get_tagname(Path); //Get the first item form Path//
08.  N_temp ← get_node(Queue); //Get the first item form Queue//
09.   if N_temp = null, then // level changing //
10.     if Flag = 0, then
11.       print("Invalid tag name or path data");
12.       for each node i in Queue, do
13.           s-unlock(i); //unlock the nodes being locked//
14.       end for;
15.       return;
16.     end if;
17.     add_node(null, Queue);
18.     Flag ← 0; //reset Flag value//
19.   else
20.     if N_temp · Tag_name = TNS, then
21.       Flag ← 1; //1:correct i/p tag name//
22.       if path is empty, then
23.         if TNS contains no additive attribute and value, or TNS contains additive attribute and value which are equal to those in N_temp · Attribute, then
24.           N_target ← N_temp; //find target node //
25.           break;
26.         end if;
27.       else // path is not empty//
28.         if TNS contains no additive attribute and value, or TNS contains additive attribute and value which are equal to those in N_temp · Attribute, then
29.           if N_temp is not a leaf node, then
30.             for each child[i] of N_temp, do
31.               s-lock(child[i]); // lock-coupling//
32.               add_node(child[i], Queue);
33.             end for;
34.           end if;
35.         end if;
36.     end if;
37.     s-unlock(N_temp);
38.     goto Line08;
39.   end if;
40.  end while;
41.  s-unlock(all nodes in Queue);
42.  if N_target is not null, then
43.     convert(s, x, N_target); //convert the Lock on N_target form s-lock to x-lock//
44.     add Ins_Element to N_target as a child node;
45.     x-unlock(N_target);
46.  end if;
End insert.
```

以下範例說明新增演算法的運作過程。當輸入參數 *Path* 為 "Booklist\Booktype(Nove)" 與 *Ins_Elemen* 為 "Book ISBN=9579598010" 後，將 T_X (根節點， "Booklist") s-lock 後，與一個 NULL 一起加入佇列 *Queue*。從 *Path* 中取出第一個 *TNS*，"Booklist"，由佇列 *Queue* 中取出第一個元素 N_{temp} 為 "Booklist"。比對 N_{temp} 是否符合 *TNS*，因為兩者相同，所以將 N_{temp} 所有的子節點 Book_type(2)、Book_type(3) s-lock 後依序加入 *Queue*，釋放 Booklist 的 s-lock，再由 *Queue* 中取出新節點 N_{temp} 為 NULL。由於 NULL 表示索引樹的階層已經變化，因此必須取得新的 *TNS* "booktype(novel)"，並加入新一個 NULL 到 *Queue*，表示此時

Queue 中的節點全是同一個階層。接著由 *Queue* 中取出新的節點 N_{temp} 為“Book_type(Computer Science)”，此時比對 N_{temp} 與 *TNS* 兩者的個別標籤名稱與屬性值都必須是一樣才行。由於結果不同，比對完畢後馬上釋放 N_{temp} 的 s-lock，然後取出新的節點 N_{temp} 為”Book_type(Novel)”，比對後發現節點 N_{temp} 完全符合 *TNS* 的描述，且此時的 *TNS* 又是 *Path* 內最後一個，這表示尋找目標節點的工作已經完成，N_{temp} 就是符合搜尋條件的 N_{target}，將 *Queue* 中所有節點的 s-lock 釋放掉。最後 N_{target} 的 s-lock 轉換成 x-lock 後，將 *Ins_Elemen* 新增為 N_{target} 子節點後，釋放 N_{target} 上的 x-lock，結束新增工作。

3.2 刪除演算法

刪除演算法有三個參數，一為 XML 索引樹根節點 T_X，一為路徑參數 *Path*，一為刪除目標節點 *Del_Element*。*Path* 路徑的最後一個 TNS 是目標節點，也就是欲刪除節點的父節點，刪除演算法將按照這條路徑找到目標節點，接著在從這個節點中找出 *Del_Element* 子節點並刪除之。在走訪的過程中，被走訪過的節點將被依序用共享鎖鎖住，當走訪後，節點將被儘快地釋放。當找到目標節點執行刪除動作時，目標節點上的分享鎖將被轉換成排他鎖，以防止目標節點被其他操作讀取或異動。圖 4 為刪除演算法的流程圖，所使用的變數用途與新增演算法用的相同。

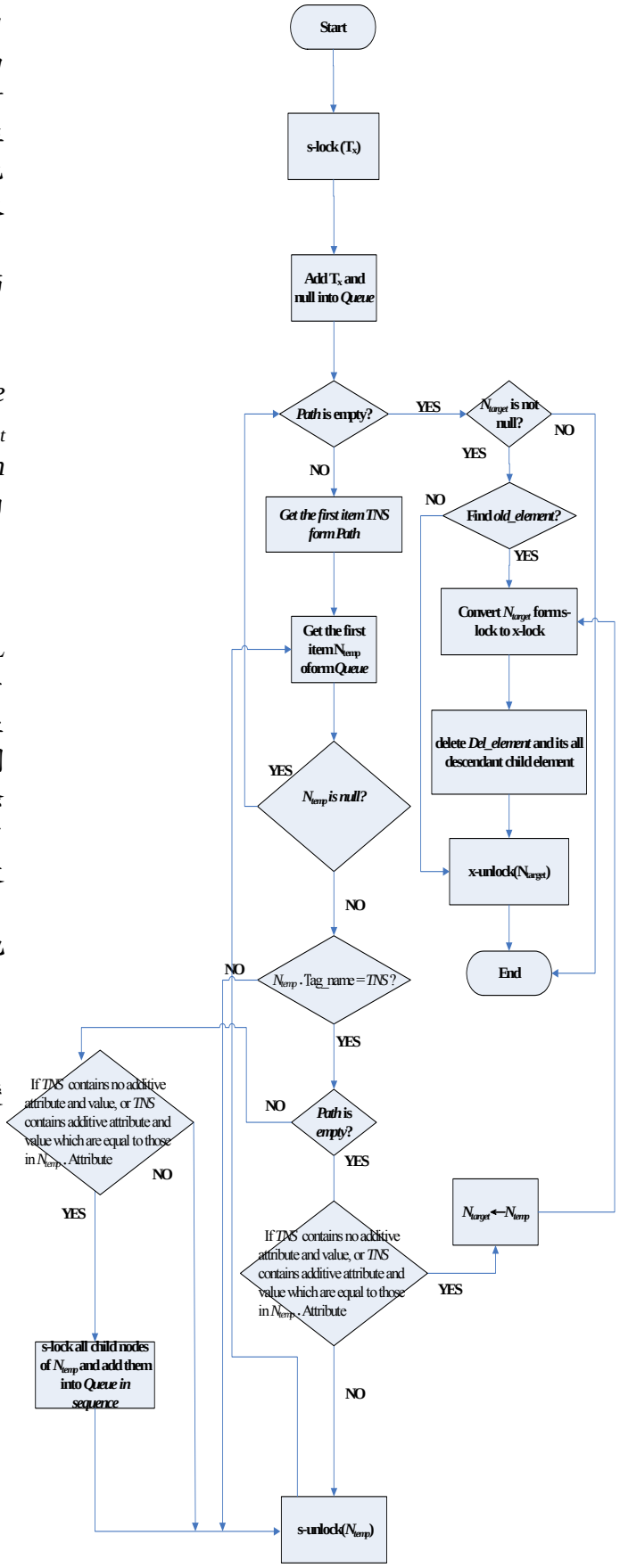

圖 4 刪除演算法流程圖

```
Algorithm Delete(Tx, Del_Element, Path)
Input:
NODE POINTER Tx;
ELEMENT Del_Element;
STRING Path;
Begin
QUEUE Queue;
NODE POINTER N_temp;
STRING TNS;
NODE POINTER N_target;
INTEGER Flag;
01.  Flag ← 0;
02.  N_target ← null;
03.  s-lock(Tx);
04.  add_node (Tx, Queue);
05.  add_node (null, Queue);
06.  while Path is not empty, do
07.   TNS←get_tagname(Path);
08.   N_temp ← get_node(Queue);
09.   if N_temp = null, then
10.     if Flag = 0, then
11.       print("Invalid tag name or path
                          data");
12.       for each node i in Queue, do
13.         s-unlock(i);
14.       end for;
15.       return;
16.     end if;
17.     add_node(null, Queue);
18.     Flag ← 0;
19.   else
20.     if N_temp.Tag_name = TNS, then
21.       Flag ← 1;
22.       if path is empty, then
23.         if TNS contains no additive
              attribute and value, or TNS
              contains additive attribute
              and value which are equal to
              those in N_temp.Attribute,
              then
24.           N_target ← N_temp;
25.           break;
26.         end if;
27.       else
28.         if TNS contains no additive
              attribute and value, or TNS
              contains additive attribute
              and value which are equal to
              those in N_temp.Attribute,
              then
29.           if N_temp is not a leaf node,
                then
30.             for each child[i] of N_temp,
                  do
31.               s-lock(child[i]);
32.               add_node(child[i],
                            Queue);
33.             end for;
34.           end if;
35.         end if;
36.       end if;
37.       s-unlock(N_temp);
38.       goto Line08;
39.   end if;
40.  end while;
41.  s-unlock(all nodes in Queue);
42.  if N_target is not null, then
43.    convert(s, x, N_target);
44.    for each child[i] in N_target, do
45.      if child[i] = Del_Element , then
46.        delete Del_element and its all
             descendant child elements;
47.        x-unlock (N_target);
48.      end if;
49.    end for;
50.  s-unlock (N_target);
51.  for each node i in Queue, do
52.    s-unlock(i);
53.  end for;
54.  end if;
End Delete.
```

以下範例說明修刪除演算法的運作過程。當輸入參數 *Path*=Booklist\booktype[Computer sciences] 與 *Del_Element* "Book ISBN=115225722" 後。將 T_X (根節點，"Booklist") s-lock 後，與一個 NULL 一起加入佇列 *Queue*。從 Path 中取出第一個 *TNS* "Booklist"，由佇列 *Queue* 中取出第一個元素 N_{temp} 為 "Booklist"，開始比對 N_{temp} 是否符合 *TNS*，因為相同所以將 N_{temp} 所有的子目節點 Book_type(2)、Book_type(3) s-lock 後依序加入 *Queue*，釋放 Booklist 的 s-lock，再由 *Queue* 中取出新節點 N_{temp} 為 NULL，節點執行比對的工作，由於 NULL 表示索引樹的階層已經變化，因此必須取得新的 *TNS*" booktype[Computer seciense]"，並

加入新一個 NULL 到 *Queue*，表示此時 *Queue* 中的節點全是同一個階層。接著由 *Queue* 中取出新的節點 N_{temp}"Book_type(Computer Science)"，比對後發現節點 N_{temp} 完全符合 *TNS* 的描述，且此時的 *TNS* 又是 *Path* 內最後一個，這表示尋找目標節點的工作已經完成，N_{temp} 就是符合搜尋條件的 N_{target}，將 *Queue* 中所有節點的 s-lock 釋放掉。N_{target} 的 s-lock 轉換成 x-lock 後，並找出 N_{target} 的子節點中符合 *Del_Element* 描述的節點，進行刪除動作，並於刪除之後，釋放 N_{target} 的 x-lock。

4.結論

本文提出二個針對 XML 文件特性而設計的並行控制演算法，包括新增與刪除兩種操作，不像其他 XML 存取並行控制演算法採用多種類型的鎖[6,8,10]，我們只使用兩種鎖，分享鎖與排他鎖。新增與刪除時由根節點開始依序鎖定到目標節點，當任何節點的相關資料處理完之後，馬上將其鎖釋放，讓該節點可以立刻被其他操作存取，提高系統的操作並行度。新增或刪除之節點不限為葉節點，可以是非葉節點。

參考文獻

[1] 陳靖國、呂冠暐，"*動態 XML 文件之並行存取"2008 資訊科技國際研討會*,2008

[2] M. ARENAS, L. LIBKIN, "XML data exchange: Consistency and query answering" *Journal of the ACM (JACM) Vol. 55 Article No. 7, May 2008*.

[3] J.K.CHEN,Y.F.HUANG, Y.H.CHIN,"A Study of Concurrent Operations on R-Trees" *Information Sciences, Volume 98, Number 1*, pp. 263-300, 1997.

[4] R. Elmasri and S. B. Navathc, "Fundamentals of Database Systems, 4th Education." *Addison Wesley*, 2003.

[5] G. Governatori, B. Stantic , and A. Sattar1, "Handling of Current Time in Native XML Databases", *17th Australasian Database Conference Vol. 49*, pp. 175-182,2006.

[6] S. Helmer, C. Kanne, and G. Moerkotte, "Evaluating Lock Based Protocols for Cooperation on XML Documents", *ACM SIGMOD Record Vol. 33, Issue 1*, pp.58-63, 2004.

[7] A. Heuer, H. Meyer, A. C. Schering, "Managing Highly Correlated Semi-Structured Data" *Proceedings of the ACM first Ph.D. workshop in CIKM*, pp 101-108 , 2007.

[8] K. F. Jea and S. H. Chen, "A High Concurrency XPath-Based Locking Protocol for XML Databases", *Information and Software Technology Vol. 48*, pp. 708–716, 2006.

[9] M. Kudo, J. Myllymaki, H. Pirahesh, N. Qi, "A function-based access control model for XML databases", *Proceedings of the 14th ACM international conference on Information and knowledge management*, pp 115 – 122, 2005.

[10] E. Lee, "Multi-Granularity Locks for XML Repetitive", *IEEE Fourth Annual ASIC International Conference on Computer and Information Science*, pp. 222-227, 2005.

[11] E. J. Lu, R.H. Tsai, and S.H. Chou," An Empirical Study of XML/EDI", *Journal of Systems and Software Volume: 58, Issue: 3*, pp. 271-279, 2001.

[12] T. Milo, S. Abiteboul, B. Amann, O. Benjelloun, and F. D. Ngoc, "Exchanging Intensional XML data", *ACM Transactions on Database Systems Vol. 30, Issue 1*, pp. 1-40, 2005.

[13] S. Natu and J. Mendonca "Digital Asset Management Using A Native XML Database Implementation" *Proceedings of the 4th Conference on Information Technology Curriculum CITC4 '03*, pp. 237-241, 2003.

[14] R. Rajugan, E.Chang, T.S. Dillon, L. Feng,"A layered view model for XML repositories & XML data warehouses" *Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference*, pp206-213, 2005.

[15] N. Wiwatwattana, H.V. Jagadish, L.V.S. Lakshmanan, D. Srivastava, "X^ 3: A Cube Operator for XML OLAP" *Data Engineering, 2007. ICDE 2007. IEEE*

23rd International Conference, pp916-925, 2007

[16] W3C, ExtensibleMarkupLanguage(XML)1.1,

[17] http://www.w3.org/TR/2006/REC-xml11-20060816/

[18] W3C, Standard Generalized Markup Language, http://www.w3.org/MarkUp/SGML/

[19] X. Yin and T. B. Pedersen, "Evaluating XML-Extended OLAP Queries Based on a Physical lgebra",*7th ACM International Workshop on Data Warehousing and OLAP*, pp.73-82, 2004.

[20] Boyi Xu, Lihong Jiang, Fanyuan Ma "On the new B to B E-business Enabling platform: cnXML in China", ACM International Conference Proceeding Series; Vol. 113 Proceedings of the 7th international conference on Electronic commerce, pp. 681 – 684, 2005.