動態 XML 文件之並行存取

The Concurrency Access of Dynamic XML Document

陳靖國*
朝陽科技大學資訊管理系
jkchen@cyut.edu.tw

呂冠暲
朝陽科技大學資訊管理系
s9514637@cyut.edu.tw

摘要

XML 是一種簡單、有彈性的文字格式，近年來相當快速的發展，相關的應用不斷的推陳出新，促使 XML 資料管理需要一個有效的管理機制，以應付在資料交換、電子商務等領域的使用，提供多人同時使用熱門的 XML 文件。以往 XML 文件的單人使用存取方法並不適用於多人使用的環境，所以我們提出一個多人使用的動態 XML 文件資料存取方法，利用並行控制的技術，允許多人同時存取資料，能確保存取到正確的資料，而不會有錯誤的情形發生。

關鍵字：XML、並行控制、鎖定、存取方法

Abstract

XML is a simple and flexible text format. Relative developments of XML make the data management need an effective mechanism for data exchange and electronic commerce to support the XML document access of multiple users concurrently. The access method of single-user version for XML document is unsuitable in a multi-user environment. Therefore, we propose an access method of multi-user version for a dynamic XML document. With the techniques of concurrency control, our method allows multiple users to concurrently and correctly access the same data without any error occurrence.

Keyword： XML, Concurrency Control, Lock, Access Method

1.簡介

XML (eXtensible Markup Language) [15]文件支援多樣的資料格式，被廣泛地應用在各種商業交易，包括電子商務、資料交換、資料儲存等等[1,8,12,16,18]。XML 文件本身具有自訂規格的特性，透過 DTD 或 Schema 定義文件，可以檢驗 XML 文件的格式是否符合一定的規格。由於這些特色，XML 相當地受歡迎，由於文件的數量逐漸增加，為了方便文件管理和存取作適當的控管[6,11,13,17]，有必要用資料庫管理系統來管理 XML 文件，讓使用者可以方便又正確查詢或新增資料。所以發展一個適當的並行控制管理機制，讓多人同時對一份文件進行資料的讀取或新增是有必要的，以便維護 XML 資料的正確性與完整性。如果資料庫沒有適當控管機制來維護而讓多人隨意地存取資料，可能會導致一些不可預期的問題發生，如遺失更新（Lost update）、骯髒讀取(dirty

read)、或錯誤總和(incorrect Summary)[4]。

以下舉例說明，使用一份 XML 文件來紀錄書店的存貨資料，有兩個使用者在同一時間針對同一種雜誌資料進行不同的交易(transaction)處理，在沒有並行控制的機制下，如果兩個使用者的交易同時執行，有可能產生意想不到的錯誤。假設某雜誌的庫存有 10 本，使用者 A 的交易是因為要賣掉其中 3 本，所以庫存量要扣掉 3，使用者 B 的交易則是該雜誌剛進貨 5 本，所以庫存量要增加 5。使用者 A 的交易 T_A 包含了三個步驟，(1)讀取雜誌庫存量資料，(2)將庫存量減 3，(3)將庫存量寫入資料庫。使用者 B 的交易也是包含三個步驟，(1)讀取雜誌庫存量資料，(2)將庫存量加 5，(3)將庫存量資料寫入資料庫。假設交易執行的順序是先 T_A 後 T_B，則 T_A 將庫存量變更為 7 後，T_B 再將庫存量變更為 12，結果是正確的，如表 1。如果先 T_B 後 T_A，則 T_B 將庫存量更改為 15 後，T_A 再將庫存量變更為 12，其結果也是正確的，如表 2。但是當 T_A、T_B 同時執行，也就是交易內的指令交叉執行如表 3 時，可能會出現錯誤結果。首先 T_A 讀取庫存量為 10，接著 T_B 讀取庫存量為 10，T_A 將庫存量減去 3 得到 7，T_B 也將庫存量加 5 得到 15，最後 T_A 將庫存量變更為 7，T_B 也馬上將庫存量變更為 15 並結束交易。當這兩個交易執行完畢後，資料庫發生了遺失更新的錯誤，亦即 T_A 的結果在 T_A 尚未完成並 COMMIT 之前就被 T_B 的結果覆蓋。我們期待的是 T_A 與 T_B 的指令即使交叉執行，其執行順序不應影響執行結果的正確性，因此我們可以瞭解到並行控制對多使用者同步資料的存取是多麼重要。

本文的目的就是要提出一套並行控制演算法來控管資料的存取，提供多使用者在同一份 XML 文件進行元素搜尋或新增的功能。利用鎖定(Lock)技術，使用讀取鎖定(Read-lock)[3]單元讓一個交易需要讀取某個元素資料時也允許其他交易可讀取該元素資料，提高交易執行並行度，但不允許該元素資料被更改。使用寫入鎖定(Write-lock)[3]單元讓交易在新增資料時排除其他交易鎖定該元素的企圖，因而不會受到其他交易干擾。在交易存取某一個 XML 元素資料之前必須先鎖定(Lock)該元素，在處理完該元素後，且不妨礙到後續工作的情況下，儘速釋放(Unlock)該元素資料以便讓其他交易能立即使用該元素，提高系統並行度。另外使用成對鎖住協定(Lock-coupling protocol)在鎖住一個元素 E 並處理完之後，確定已鎖住下一個要處理的目標元素之後，才能將元素 E 給釋放掉，用此協定來搜尋 XML 資料索引樹之元素節點以確保正確的路徑走訪。

表一、交易結果正確（先 T_A 後 T_B）

時間順序	交易名稱	指令	庫存量 X
1	T_A	Read x	10
2	T_A	x=x-3	10
3	T_A	Write x	7
4	T_A	Commit	7
5	T_B	Read x	7
6	T_B	x=x+5	7
7	T_B	Write x	12
8	T_B	Commit	12

表二、交易結果正確（先 T_B 後 T_A）

時間順序	交易名稱	指令	庫存量 X
1	T_A	Read x	10
2	T_B	Read x	10
3	T_A	x=x-3	10
4	T_B	x=x+5	10
5	T_A	Write x	7
6	T_B	Write x	15
7	T_A	Commit	15
8	T_B	Commit	15

表三、交易結果發生錯誤

時間順序	交易名稱	指令	庫存量 x
1	T_B	Read x	10
2	T_B	x=x+5	10
3	T_B	Write x	15
4	T_B	Commit	15
5	T_A	Read x	15
6	T_A	x=x-3	15
7	T_A	Write x	12
8	T_A	Commit	12

2.概念與技術探討

2.1 XML 文件簡介

XML 是由 SGML[15]語法衍生的一種標記語言，簡化了許多 SGML 中繁雜的規定，目的是要讓使用者進行資料交換時，有一個雙方認同的文件樣式的標準[14]，因此，XML 與眾人熟知的 HTML 最大的差異是 XML 對格式有更嚴格的要求。W3C [14] 對 XML 文件的格式要求有兩種，第一種文件格式稱為良好格式(Well-Formed)，其規定如下。(1)文件的開頭必需宣告文件版本、編碼等資訊，(2)文件中必須有一個唯一的根元素，(3)每個元素中的資料被一組對稱的起始與結尾標籤(tag)包括起來，(4)任一個元素的起始與結尾標籤不可與其他元素的起始與結尾標籤交錯出現，(5)所有元素的屬性值都必須使用雙引號括住，(6)已被 XML 指定為保留字元的資料要呈現時必須使用替代符號，例如”>”用’&pt’代替。第二種文件格式稱為驗證過的(valid)，一份通過驗證的 XML 文件不但必須滿足良好格式條件，而且還要符合一份 DTD (Document Type Definition) 或 XML Schema 中對於 XML 文件的規範。DTD 或 XML Schema 的功用是要讓 XML 文件在資訊交換時有可以遵守的資料架構與屬性特徵，如圖一(a)的範例，DTD 規定在 XML 文件中一本書至少要有一個 title 元素，每個 book 中一定要有包含 ISBN 屬性等規則，滿足 DTD 規範的文件可稱為驗證過的 XML 文件。圖一(b)為一份驗證過的 XML 文件，這份文件描述一份藏書清單，清單中以主題作為分類，每本書的資料包含書名、作者、出版商、價格以及出版年。

2.2 並行控制

並行控制的目的是提供資料庫在多人環境下，提供有效率、可靠的機制，用來保證交易中的指令不論如何交叉執行其結果都是正確的。交易是一個邏輯上不可切割的執行單元，包含了數個指令，在前文所描述的庫存更新即是一個交易，包括讀取資料、處理資料、更新資料指令等等。在並行控制的環境中，不同交易的指令被執行的順序是不一定的。一個交易必須被完整的執行，不可只做部份的指令就結束，假若交易半途終止，已執行過的指令所產生的結果必須被取消，受影響的資料必須復原到執行前的狀態，例如更新庫存資料失敗時，庫存資料應該要回復到更新前的狀態。

常用的並行控制方法有三種，亦即鎖定法

(Locking)、時間戳記法(Time stamping)、樂觀法(Optimistic)[4]。鎖定法的概念是用是一個鎖來表示一個資料單元可否存取的狀態，分別是鎖定(Lock)與非鎖定(Unlock)兩種，當資料單元被一個交易 T 鎖定時，T 可以存取該資料單元，但是其他交易不行，當資料單元狀態為非鎖定時，任何交易都可以對他作鎖定要求，這是基本的二元鎖定法(Binary Lock)只用一種鎖定類型(Lock type)。此方法的缺點在於兩個交易無法對同一個資料單元同時進行讀取的動作。雖然二個交易都不會改變資料內容，但還是必須等前一個交易釋放出該資料單元後另一個交易才能讀取該資料單元。後來演變為兩種鎖定型態，分享鎖定(Share lock)、互斥鎖定(Exclusive lock)，當狀態為分享時，表示允許其他交易只能讀取這個資料，但是不可更動。互斥鎖定表示一個資料單元被鎖定後，不允許其他交易去存取該資料單元。用鎖定法鎖定資料單元處理資料時有一個協定，叫做兩階段鎖定協定(Two phase locking protocol 簡稱 2PL)，分為兩個階段，成長（Growing）與縮減（Shrinking）階段。交易在成長階段時，必須將所有需要的資料鎖定住，而且不能釋放已鎖定的鎖，所有需要的資料都鎖定後才開始資料處理工作，資料處理工作完成後，交易就進入所謂的縮減階段，這時交易不可在鎖定任何資料單元，只能釋放成長階段中鎖定的資料單元，這是為了確保死結現象不會發生，這種機制讓較早被鎖定的資料單元，要等到交易結束被釋放後，才能讓其他交易使用該資料，因而嚴重降低交易執行的並行度。此外，2PL 不適用在樹狀索引結構上，因為事先無法得知有那些節點要鎖住。另有一種鎖定的協定叫做成對鎖住協定(Lock-coupling protocol)非常適用在樹狀結構的索引上，當交易需要依序鎖定兩個相鄰的節點時，第一個已被鎖住的節點必須在第二個節點鎖定之後才可以被釋放。交易鎖住資料單元後就可以對其做處理資料，完畢後應立刻釋放該資料單元，以減少資料單元被鎖定的時間。這樣一來就可以提昇交易之間的執行並行度，所以我們將採用這個協定作為資料鎖定與釋放的原則。

時間戳記法(Timestamps)利用資料庫管理系統對每一個交易產生一個唯一的識別碼的時間戳記，用來判斷交易的執行先後順序。每當一個交易要讀取或寫入資料到某一個資料單元時，先比較交易的時間戳記和資料單元的時間戳記，如果前者比後者大則可以作讀/寫工作，並將資料單元的時間戳記更新為交易的時間戳記。如果前者比後者小，交易將被終止，重新執行並取得新的時間戳記。樂觀法作法是任何交易都可以直接存取資料單元不需任何鎖定或戳記比對，只在交易結束的時候才作驗證工作，交易是在資料項目副本(Local copy)存取資料，當最後驗證通過確定交易的結果是正確的之後，再將副本更新到資料庫中[4]。

論文[9]中的方法 XLP 是對 XPath 量身訂做的並行控制方法。XPath 是一套針對 XML 查詢的語言，用來為 XSLT 尋找文件中元素的位置。XLP 將 XML 視為一個樹狀結構，在查詢時由根節點開始到目標節點所經過的路徑上，所有的節點均使用 P-lock(Pass_by)鎖定，當搜尋的動作結束後，將找到的目標節點上的 P-lock 升級成操作所需要的某種 lock 類型，例如刪除、新增、讀、寫等鎖。

論文[10]的概念是將一份文件中會重複出現的結構作切割，將原本較大的資料單元切割成數個較小的資料單元，其方法名為 *-factoring，在 DTD 中*代表同樣標籤名稱的元素資料可能出現多個。當 XML 文件切割完畢後，使用 list locking protocol 進行並行控制，針對各個不同的資料單元作新增、刪除的功能，該協定是一個遵循 2PL 規則的並行控制處理機制。在走訪子節點前需先對該子節點下 traverse lock。當一個父節點有若干子樹時，若要刪除其中任何一棵子樹，則必須先對父節點

下 list lock，當要刪除一棵子樹時，必須先對該子樹的根元素下 delete lock，intentional lock 則是存取子樹中根元素的任一後代節點時使用。

論文[7]的 NO2PL、OO2PL 方法主要概念如下。一個節點除了紀錄標籤名稱、內容等資料外，為了滿足針對某節點的第 n 個子節點作新增或查詢的需求，節點中的指標紀錄了節點自己左右兄弟節點、節點第一個與最後一個子節點的指標，共四個指標，根據走訪、修改兩種操作的狀況下，依照走訪的節點關係與不同的操作給予不同的鎖定，合計有八種鎖定，其中走訪的操作使用的鎖定為 S-LOCK 而修改的操作使用的鎖定為 X-LOCK。

3.方法介紹

綜觀前文所介紹的方法，XLP 的缺點有兩點，首先是 XPATH 缺乏新增、修改等指令，只能針對資料作查詢，第二是 XLP 針對專業人員而設計，使用 XPATH 必須先花費時間學習，並了解 XML 文件架構，故一般使用者難以使用 XPATH 作為查詢工具，NO2PL、OO2PL 方法中使用了太多的鎖定類型，過於複雜，在節點指標的設計上，目標節點若是為第一子節點與最後子節點的中間時，需要鎖定的節點數量將會相當龐大，在交易進行前，需要花費大量的時間與資源進行檢查。有鑑於上述的理由，本文提出一套適用在 XML 文件的搜尋與新增的並行控制演算法，給予搜尋的若干關鍵字，利用搜尋演算法可以快速找出符合條件的適當元素資料。給予一個元素資料，利用新增演算法，可以很快找出一個適當父元素，再將新增元素當成子元素插入父元素裡。

3.1 索引樹資料結構

本文將為 XML 文件建立一個索引樹狀結構來加快資料存取速度，選擇 m-way 樹作為 XML 的索引結構，這是因為 XML 資料呈現的架構與 m-way 樹的結構相似，而且 m-way 樹限制較少，不像有些樹狀結構的節點有限制子節點的數量，或是節點的鍵值(Key value)必須有順序性，例如 B-tree[2]、R-tree[5]。這棵索引樹中的每個節點結構如圖 2，包含(1)標籤名稱(tag_name)欄位：一個指標指向節點所對應 XML 元素標籤名稱的起始位置。(2)元素內容(content) ：指向 XML 元素內容的起始位置，若元素內容無實際的資料，則此欄位值 Null。欄位($child_1$, $child_2$, …,$child_n$)則是指向每個子節點的指標。

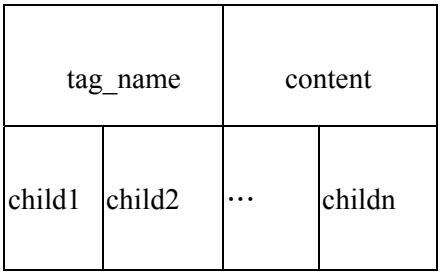

圖 2.XML 資料索引樹的節點結構

3.2 鎖定模式

本文使用的鎖包含兩種類型 share-lock 和 exclusive-lock。這兩種鎖的相容與否狀態請參考圖 3，其中 s-lock 表示 share-lock， x-lock 表示 exclusive-lock，O 表示兩者相容，X 表示兩者不相容。節點的鎖定與釋放順序遵守成對鎖定協定作法。share-lock（s-lock）使用在交易需要讀取節點中的資料之前，必須先鎖住該節點成功然後才能讀取，當節點被某個交易 s-lock 時，其他交易只允許再對它做 s-lock，而不可以作 x-lock。Exclusive-lock（x-lock）使用在交易需要修改節點中的資料之前，交易必須先 x-lock 該節點成功之後才能修改，當節點被交易 x-lock 時，其他交易的 s-lock 或 x-lock 要求將被暫時中止，直到該節點被 x-unlock 後才能生效。

	s-lock	x-lock
s-lock	O	X
x-lock	X	X

圖 3.兩種 lock 之相容與否描述

3.3 搜尋動作(Search operation)

搜尋的操作目的是搜尋節點中的標籤名稱、內容或屬性值包含了關鍵字的節點，搜尋時使用廣度優先搜尋法(breadth-first search)由上而下由左至右的順序，走訪並檢查每一個節點，以下列出在搜尋過程中會用到的一些名詞解釋。我們將 R 定義為 XML 文件的索引根節點、Q 定義為一個佇列，用來儲存待檢查的 XML 索引節點、r 是一個佇列，用來儲存符合搜尋條件的 XML 索引節點、最後 KEY 是由 n 個搜尋關鍵字(K_1, K_2, K_3, …, K_n)所組成的集合。搜尋程序一開始將 R s-lock 住，並將 R 加入 Q 中，然後不停地從 Q 中取出節點作比對工作，直到 Q 空了為止。當一個節點內含與關鍵字比對時，若節點中屬性、標籤名稱含有所有關鍵字就將此節點存入 r 中，再將該節點底下所有子節點依序 s-lock 後再加入 Q 中，最後將該節點 s-unlock。當索引樹每個節點都找過之後，有符合的節點即存於 r 中，即可將其對應的元素資料傳回給使用者參考。以下是搜尋演算法的主要步驟和詳細演算法描述。

Step1. s-lock(R)。

Step2. 將 R 加入 Q。

Step3. 由 Q 中取出一個節點 N。

Step4. 如果 N 的屬性、標籤名稱包含所有關鍵字 KEY，將 N 存入 r 中。

Step5. N 所有子節點依序 s-lock 後加入 Q。

Step6. s-unlock(N)。

Step7. 持續 Step 3-6 的動作直到佇列 Q 為空，傳回 r。

```
Algorithm SEARCH(R, KEY )
Input : char KEY[1..n]; // 關鍵字 K1 ,K2 ,… ,Kn//node R; //根節點//
Output：queue r; //儲存符合搜尋條件的 XML 節點佇列//
Begin
  node N; //進行關鍵字比對的目標節點//
  queue Q;
  //儲存待檢查的 XML 節點佇列//
  queue r; //儲存符合搜尋條件的 XML 節點佇列//
  s-lock(R);//將根節點鎖定//
  Add R to Q;
  while Q is not empty, do
     N← get the first element form Q;
     if N's tag_name, content, or attributes
     include KEY, then
        add N to r;
        //將符合條件的節點加入 r//
     end if;
     for each child i of N, do
        s-lock(i);
        //將 N 的所有子節點鎖定//
        add i to Q;
     end for;
     s-unlock(N);//完成檢核後釋放 N//
  end-while;
  return r;//傳回搜尋結果//
End SEARCH.
```

3.4 新增動作(Insert operation)

新增的操作目的是將一個新的節點新增於指定的位置。以下列出在新增過程中會用到的一些名詞解釋，R 定義為 XML 文件的索引樹、N 定義為新增的子節點、PATH 紀錄一條路徑由根節點到預定插入 N 的父節點所經過的節點。新增一開始要找出插入 N 的位置，透過 PATH 內容，可以到達 N 的父節點，叫做節

點 M。在走訪的過程中，為了防止其他應用程式交易更動 PATH 中的任一節點，沿途用成對鎖定方法將 PATH 中的所有走訪節點一一s-lock，當走訪至 M 時改用 x-lock，然後新增 N 到 M 中當其子節點，再來 x-unlock M 並且依序 s-unlock 所有 PATH 中的節點，結束新增的動作。從根節點到 M 節點之間的節點，我們使用 2PL 協定鎖定主要理由是將此路徑當成一個邏輯"大節點"，讓 N 插入 M 時，這個大節點內容不會受到其他交易變更而影響。其插入路徑正確性。以下是新增演算法的主要步驟和詳細內容描述。

Setp1.用成對鎖住方式，走訪 PATH 中節點，並依序 s-lock 這些節點

Setp2. x-lock(M)。

Setp3.新增 N 為 M 的子節點。

Setp4.x-unlock(M)。

Setp5.依序 s-unlock PATH 中被 s-lock 的所有節點。

```
Algorithm INSERT （R,,N,PATH）
Input：m-way tree R;node N;
        //新增的子節點//
         queue PATH[1..n];
         //佇列存有根節點到 M 所經過的節點群，不包含 M//
Begin
  for each node i, form top to bottom in PATH, do
    s-lock(i);//將 PATH 的所有節點鎖定//
  end for;
  x-lock(M);//將節點 M 鎖定//
  add node N to M as M's child node;//新增節點//
  x-unlock (M);//釋放節點 M//
  for each node i, form top to bottom in PATH, do
    s-unlock(i); //釋放將節點 M 鎖定//
  end for;
End INSERT.
```

4.實例說明

使用一個案例描述搜尋、新增時執行的狀況，圖 4 為圖 1XML 文件的邏輯結構，每個圓圈代表一個元素，節點中紀錄一個 XML 元素中標籤名稱、屬性、內容、所有子節點的指標，下面的說明中以標籤名稱代表一個 XML 元素，標籤名稱(屬性)代表一個含有屬性的元素，由於本範例中包含同樣標籤名稱的 XML 元素，圖中會加上編號作為辨識，在範例中以標籤名稱(編號)表示，編號依照由上而下，由左向右的順序依序編號由 1 開始。

當使用者輸入關鍵字為"system"並開始搜尋後，根節點 Book list 先加入佇列 Q，s-lock 後，開始比對關鍵字是否被包含在 Book list 的標籤名稱或屬性中，由於 Book list 不包含關鍵字，所以不作任何處理，將 Book list 的子節點 Book_type(2)、Book_type(3) s-lock 後加入 Q，再由 Q 中取得第一個節點 Book_type(2)，將節點 Book_type(2) s-lock 後，對該節點執行比對的工作，由於該節點的標籤名稱、屬性不包含關鍵字，故 Book_type(2) un-slock，將 Book_type(2)的子節點 Book (4)、Book(5)放到 Q 中，由 Q 中取出 Book(4)節點，檢查 Book(4)後發現其中的屬性包含了關鍵字，因此將 Book(4)紀錄到 r 中，Book(4) un-slock，由 Q 中取出下一個節點 Book(5)，重複上述的步驟直到佇列為空，最後的結果就是 Book(4)、Book(5)這兩個節點為符合搜尋條件的節點。當對書籍 Book(6)新增一個 author 的元素，PATH 中會紀錄使用者事先輸入的路徑 Booklist(1)、Book_type(3)，依序將 Booklist(1)、Book_type(3) s-lock，Book(6)節點 x-lock，當上述的鎖定依序取得後，開始新增元素"author"，新增操作完成後，x-unlock Book_type(3)，s-unlock Book(6)、Booklist(1)後結束新增。

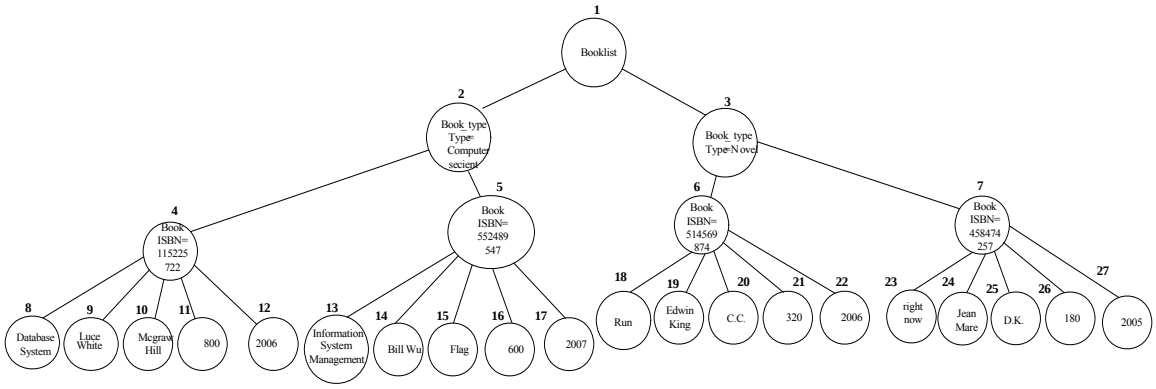

圖 4、XML 文件邏輯結構圖

五.結論

本文提出一個針對 XML 文件特性而設計的搜尋與新增的並行控制存取方法，索引的設計使用 m-way 樹，使用兩種鎖定模式 share 與 exclusive，在搜尋時，使用廣度優先方式搜尋標籤名稱、屬性、內容中包含所有關鍵字的節點，新增時由根節點開始依序鎖定到目標節點，將新節點加入後，將之前鎖定的節點釋放，本文提出的方法提供資料的查詢與新增的工具，使用簡單的鎖定來避免資料存取時系統的判斷問題，增加系統的效能同時確保資料執行的正確性。

參考文獻

[1]A. Bonifati, S. Ceri, and S. araboschi, "Pushing Reactive Services to XML Repositories Using Active Rules", ***The 10th International Conference on World Wide Web***, pp. 633-641, 2001.

[2]R. Bayer and M. Schkolnick, Concurrency of Operations on B-trees, ***Acta Inforrnatica*** pp. 1-21, 1977.

[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]A. Guttman, "R-trees: A dynamic index structure for spatial searching" ***Proc.ACM SIGMOD Annual Meeting***, pp. 47-57, 1984.

[6] 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.

[7]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.

[8]W. Hümmer, A. Bauer, and G.r Harde, "XCube – XML for Data Warehouses", ***the 6th ACM International Workshop on Data Warehousing and OLAP***, pp. 33-40, 2003.

[9]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.

[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]W3C,ExtensibleMarkupLanguage(XML)1.1, http://www.w3.org/TR/2006/REC-xml11-20060816/.

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

[16]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.

[17]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.

[18]D. C. Yen, S. M. Huang and C.Y. Ku "The Impact and Implementation of XML on Business-to-Business Commerce", ***Computer Standards & Interfaces, Volume 24, Issue 4***, pp 347-362, 2002.

```
<?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 </title>
            <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