XML 文件版本管理

Version Management of XML Documents

陳耀輝　蔡政霖
國立嘉義大學資訊工程學系
{ychen, s0910253}@mail.ncyu.edu.tw

摘要

XML 文件版本管理(Version Management)最主要的目的在於要對連續修改的 XML 文件作有效率地儲存，減少儲存的成本，並且能快速地擷取之前的 XML 文件版本和提供複雜查詢的功能。針對 XML 文件版本管理，給予 XML 樹狀結構上的每個節點時間戳記、階層值和一組權重值。藉由判別節點的時間戳記，可以直接地擷取之前的 XML 文件版本，並且利用權重值和階層值來維護節點之間的關係。權重值和階層值也能成為 XML 文件的索引資訊，作為查詢路徑比對的依據進而加快查詢效率。我們用關聯式資料庫儲存節點和其相關的權重值、階層值和時間戳記，以結合關聯式資料庫儲存大量資料的功能，及查詢處理和最佳化的技術。

關鍵詞

XML、版本管理、時間戳記、權重值

1. 簡介

近年來 XML (eXtenbsible Markup Language)已經成為網際網路上一個非常重要的資料交換標準[15]，其原因是 XML 具有獨特的資料自我描述性 (Self-describing) 與可擴充性 (Extensibility)，使得 XML 文件變成一份會說話的文件。XML 使用標籤(Tag)或元素(Element)來描述資料，例如 <年齡> 10 </年齡>，我們並不會將 10 誤解為 10 元或 10 公分，因為<年齡>標籤是被定義成隱含語意(Semantic)的資訊，很明顯地對於只著重在資料呈現的 HTML 來說有著相當大的不同。

在以前企業之間資料交換必須透過預先定義好的資料格式，彼此才能進行資料交換。現今的 XML 文件以純文字為基礎，沒有其它修飾性的標籤影響文件內容。所以當接收到一份 XML 文件時，可以利用 XML 解析器(Parser) [16]來解讀 XML 文件內容，並且轉換成符合自己所需的文件或給其它的應用程式使用。如此一來 XML 文件可以說是真正達到跨平台的目的。

一般可以大略地將 XML 文件分成交易性資料和文件型資料。交易性資料只單純地著重在資料的交換上，不會保留 XML 文件。例如異質性系統之間透過 XML 作資料交換，將資料擷取出來儲存在各自的資料庫中。而文件型資料類似書籍或公文。當我們使用 XML 格式編寫文件，而不使用廠商所提供的文書編輯器是因為 XML 的自我描述的功能並且以純文字為基礎，能確保文件在經過多年以後不會因為文書編輯器的昇級或消失而無法開啟。因此 XML 文件擁有其他文件格式所沒有的保值能力。

現今已有數種不同儲存 XML 文件的方法可供選擇[1][7][8][12]，概略地分為檔案系統、物件導向資料庫、關聯式資料庫[1][7][12]和特別的系統，例如 Lore 系統[8]。由於 XML 文件是有順序性的(Order)，而關聯表(Relation)的記錄(Record)是沒有順序的，如果要將 XML 文件儲存在關聯式資料庫中，則必須先將 XML 文件解析成樹狀結構，之後為每個節點(Node)作有順序的編碼[13]，再存入關聯表中，保持節點之間的順序關係。

雖然已有各種形式儲存 XML 文件的方法被提出，但仍然無法保留使用者對文件一連串修改的痕跡，我們把這個問題稱為 XML 文件版本管理。為了解決這個問題我們必須要考慮文件具有時間的價值性、文件可以持續再利用和文件的內容不能因為時間流逝而有所遺漏的三個特性。所以我們必須完整且持續地去記錄不同時間所修改的 XML 文件內容。直覺的方法是把每次修改後的 XML 文件後都另存成一份新的文件，如此一來在未來的日子裡將會有大量的 XML 文件被產生，而這些 XML 文件的管理勢必不能被忽視。

XML 文件版本管理最主要的問題在於如何對連續修改的 XML 文件作有效率地儲存，減少儲存的成本，並且能快速地擷取之前的 XML 文件版本和提供複雜查詢的功能。針對 XML 文件版本管理我們所提出的方法是給予 XML 樹狀結構上的每個節點時間戳記(Timestamp)、階層值(Level)和一組權重值，分別為最小權重值(Min-weight)與最大權重值(Max-weight) [20]。主要目的是藉由判別節點的時間戳記，可以直接地擷取之前的 XML 文件版本，並且利用權重值和階層值來維護節點之間的關係。權重值和階層值也能成為 XML 文件的索引資訊，作為查詢路徑比對的依據進而加快查詢效率[2]。最後將節點和其相關的權重值、階層值和時間戳記存入關聯式資料庫，以結合關聯式資料庫中現有的查詢處理和最佳化的技術。

本文的文章結構如下，第二節介紹相關研究，第三節簡單說明我們所使用的 XML 文件編碼方法，第四節則說明 XML 文件版本的儲存及擷取方式，第五節介紹查詢方法，最後第六節是我們的結論。

2. 相關研究

目前被提出的 XML 文件版本管理技術有 Edit-based [6][11][14] 、 Reference-based [5] 、 Timestamps-based [3][4]等方式。

Edit-based 方法的主要概念都是利用 Edit-script 語法(Insert、Delete、Update、Move)來記錄同一份的 XML 文件不同版本之間的改變。[6]使用 Page 的方式將 XML 文件版本儲存其中，並且使用 Page usefulness 概念將資料叢集在一起，減少擷取版本的 I/O 次數。其缺點是需要複製原有的資料來達成叢集資料的目的，以至於會增加多餘的儲存成本。[11]的做法是儲存最後一次修改的 XML 文件版本，利用 Forward completed deltas 來記錄兩個 XML 文件版本的改變，Forward completed deltas 讓我們獲取之前的 XML 文件版本，並且經反轉 Forward completed deltas 後，又可重新返回上一次修改的 XML 文件版本。[14]儲存第一次和最後一次修改的 XML 文件版本加上 Forward completed deltas 的特性，改進了擷取 XML 文件版本的速度，但是仍然無法突破直接任意地擷取 XML 文件版本的限制。

Reference-based 方法 [5]首先會完整地儲存最原始的 XML 文件版本，當 XML 文件版本被連續的修改時，只會儲存新插入文件的部份，而未改變的部分則是使用 Reference record 來指向在前一版相同資料的位置。因為每一個 XML 文件版本都儲存了部份文件資料，所以此一部份類似 Edit-based 方法的 Forward completed deltas。不同於 Edit-based 方法的是[5]可以直接地擷取特定的 XML 文件版本，主要是透過 Reference record 將共同未改變的資料，使用遞迴的方法恢復。此外，[5]也結合了 Page usefulness 的概念叢集資料，來減少擷取版本的 I/O 次數，但是會增加額外的儲存空間。

以 Timestamp-based 方法為基礎架構的[3]將所有的 XML 文件版本合併成一個階層式的結構，並且提出繼承時間戳記的想法。每個子節點或子孫節點都會繼承一組父節點或祖父節點的時間戳記。當擷取 XML 文件版本時，時間戳記相互之間發生繼承的衝突，此時解決的方

法是以子節點或子孫節點的時間戳記為基準，來決定此一節點是否要被擷取。

3. 資料模式和 XML 文件順序性編碼

在這節中，我們將提出一個數值範圍的編碼方法，以輔助用關聯式資料模式儲存有順序性的 XML 文件。此編碼方法還可以協助查詢和索引 XML 文件。

3.1 XML 資料模式

一份 XML 文件可以看成一個樹狀結構，其內部節點(Internal node)包含元素，葉節點(Leaf node)包含其值，節點與節點之間則存在父子(Parent-child)、兄弟(Sibling)和祖孫(Ancestor-descendant)關係。所以順序性(Order)對 XML 資料模式來說是很明顯的特徵，相對地 XML 的樹狀結構也隱含了順序。另外 DTD (Document Type Descriptor) [17]提供綱要(Schema)資訊來說明 XML 文件，而事實上不一定每份 XML 文件都要有 DTD。

3.2 關聯式資料模式

關聯式資料模式和 XML 資料模式最大的不同之處在於關聯表的記錄與記錄之間是沒有順序性。記錄可利用其主鍵值和其它的記錄來區分彼此，關聯表之間的關係則是透過外來鍵(Foreign Keys)來連繫。下面我們將說明如何替有順序的 XML 文件編碼。

3.3 XML 文件順序性編碼方法

為了在關聯式資料模式中儲存和查詢 XML 文件，我們必須使用特殊的編碼方法來維持 XML 文件的順序性，並且要有良好的機制去維護編碼結構。選擇 XML 文件編碼的方法，主要考量的有三個因素，分別是維護的成本、重建文件的效率和查詢的速度。目前已有各種形式的編碼方式被提出[13][9]，這些方法主要的問題在於當編碼方法擁有低成本的維護機制時，相對地也會有較高的查詢時間。我們發現[20]的數值範圍表示法，能在這些問題上取得了一個平衡點。雖然[20]主要的目的是使用在 XML 文件索引的建立與維護上，但是其嚴謹的結構能確保 XML 文件的順序性，使得符合我們的需求。尤其是[20]能提供複雜查詢的功能，並且支援不確定路徑的查詢。接下來我們將簡單地介紹[20]的主要架構。

數值範圍給定的方式是給予樹狀節構上的每個非葉節點一組權重值，包含了最小權重值和最大權重值並且加上階層值，表示成(最小權重值：最大權重值, 階層值)，例如(1:4, 2)。葉節點只有最小權重值，所以葉節點的最大權重值則設為 Null 值，例如(1:Null, 2)，因此我們可以藉由數值的比對快速的決定節點間的關係。以 Figure 1 為例，針對所有的葉節點依據從左至右的順序給予一個由小至大的權重值，如 F_name 得到權重值 1，L_name 得到權重值 2，city 得到權重值 3，zip 得到權重值 4。非葉節點則取所有子孫節點的權重值範圍中，最小與最大權重值為其權重值範圍，如 name 的子孫節點中最小的權重值為 F_name 的權重值 1，最大的權重值為 L_name 的權重值 2，因此 name 的權重值範圍為 1 到 2，階層值為 2，因為 name 在 XML 文件樹裡位於高度 2 的位置，表示成(1:2, 2)。當我們利用上述方式分配給每個葉節點一個最小權重值及非葉節點一個權重值範圍後，只要利用其包含關係就能快速比對出兩點間的關係。如 name 的權重值範圍(1:2)被 people 的權重值範圍(1:4)所包含，因此 name 為 people 的子孫；葉節點 F_name 權重值為 1，被 name 的權重值範圍(1: 2)包含，因此 F_name 為 name 的子孫。此外，當某兩

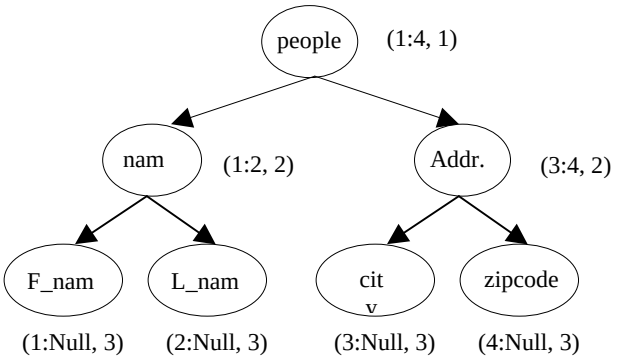

Figure 1. XML 文件順序性編碼

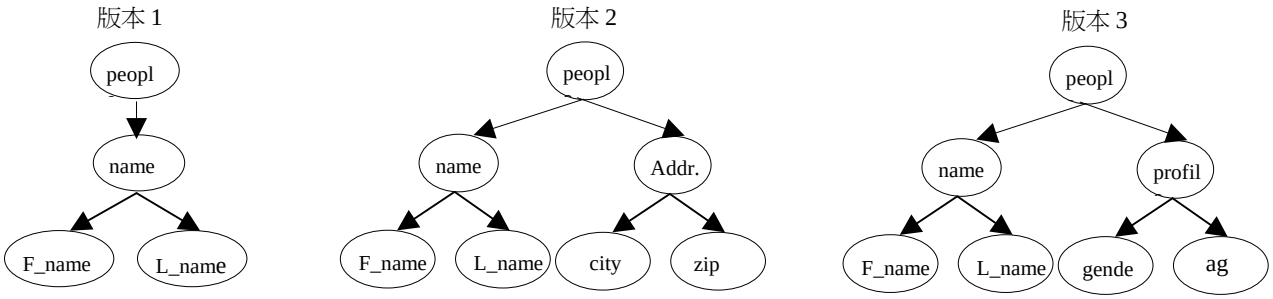

Figure 2: 連續性修改的 XML 文件版本

點間有包含的關係且階層數差 1，則此兩點為父子關係，若階層數差 2 以上，則兩點為祖孫關係。藉著以上的編碼方法，我們可以快速的決定兩點間的關係。

之前提到一個好的編碼方法，必須擁有低成本的維護。在此[20]對於新增、刪除 XML 文件樹時，只須更動一小部份的資料，且能很方便地利用其它節點來決定被新增、刪除節點的權重值或子孫權重值範圍。如此，使得新增及刪除的問題更容易處理，在此我們不加以說明，詳細內容請參考[20]。

4. 時間戳記版本模式

接下來我們將介紹如何把連續性修改的 XML 文件版本合併成一份 XML 文件。並且利用時間戳記和上一節所提到的數值範圍表示法，存入關聯式資料庫中，並且有效率地擷取任一時間的 XML 文件版本。

4.1 時間戳記(Timestamp)

時間戳記的基本想法與數值範圍表示法一樣，都是為了確認節點的位置，只不過範圍表示法指的是空間上的位置，時間戳記則是時間軸上的位置。Figure 2 顯示出 3 個不同時間點的 XML 文件版本，分別本是版本 1、版本 2 和版本 3。版本 2 的產生是由版本 1 新增了 Addr. 節點和子節點 city 及 zip，而版本 3 的產生則是版本 2 刪除了 Addr.節點和子節點和新增 profile 節點和子節點 gender 和 age。在往後當我們提到新增、刪除 Addr.節點或新增 profile 節點時，將隱含其子節點。

我們發覺到在 XML 文件版本管理的問題裡每個節點都具有空間性和時間性，所以為 XML 文件版本分配範圍值來定位每個節點的順序時，也能同時地為節點記錄生命週期。也就是說當 XML 文件經過連續性地修改後，有些資料是不變的，有些則是被新增、刪除或是更改。因此我們可以為節點設立一個時間戳記來表示資料的時間範圍。要特別注意的是在 XML 文件版本的問題裡，刪除只是邏輯上的形式，假如真正地把資料給刪除後，則永遠找不回來。例如 Figure 2 裡顯示版本 3 的產生是由版本 2 刪除了 Addr.節點和新增 profile 節點所得到的。如果我們沒將版本 2 的 Addr.節點記錄下來，則在恢復版本 2 時將會找不到 Addr.節點的資料。

綜合了時間性與空間性的觀念後，我們只要將節點原有的數值範圍的資訊，再加上時間戳記後，就能充份地了解資料在全部的 XML 文件版本裡所在的位置。我們將數值範圍加上時間戳記表示為（最小權重值：最大權重值, 階層值, 版本起始值：版本終止值）。Figure 3 指出（版本 1）的 XML 文件，每個節點上除了原本的數值範圍外，還加上了時間戳記，例如根節點 people 的權重數值範圍是從 2 到 4，階層為 1，版本起始值是 1，版本終止值也是 1，表示為(2:4, 1, 1:1)。版本 1 的權重值給定

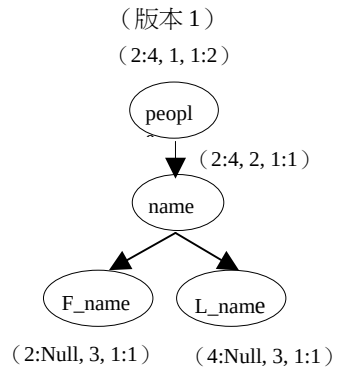

Figure 3: 整合性 XML 文件版本（1）

方式是依據第 3 節所提到的編碼方法，而時間戳記的給定方式則是依照資料出現在 XML 文件版本的時間來確認的。在版本 1 時，我們給定每個節點的時間戳記都是 1 到 1，因為這是第一次建立的 XML 文件。

Figure 4 的（版本 1-2）所表達的意思是版本 1 與版本 2 合併在一起，我們只要經過簡單的比對就能知道版本 1 與版本 2 之間的差別[19]。在比對版本 1 與版本 2 後發現，在版本 1 的 name 節點的右邊新增了節點 Addr.，其他則無任何節點被刪除或修改。所以先將原先版本 1 而在版本 2 裡沒有變動的資料，修改時間戳記的版本終止值為 2，版本起始值則維持不變。例如根節點 people 的時間戳記已經被修改成 1 到 2。至於新插入的節點 Addr.及子節點，我們必須使用第二節提到的方法維護 XML 文件樹，[20]的論文中提到 XML 文件樹的新增可以區分成 10 種情形，而所有的處理方式則有四種方式。（版本 1-2）是屬於情況 6，因為 Addr.插入根節點的最右邊，並且使用方法 1 的方式來編碼。對照（版本 1-2）的 name 節點最大權重值為 4，所以我們只要選擇比 4 還大的數值，來對葉節點 city 和 zip 作最小權重值的給定，分別為 10 和 12，階層值為 3，時間戳記為 2 到 2，各自表示為(10:Null, 3, 2:2)、(12:Null, 3, 2:2)。Addr.節點的最小權重值和最大權值，則取 city 和 zip 的 10 與 12，層階為 3，時間戳記一樣為 2 到 2，表示為(10:12, 2, 2:2)。不一樣的是我們必須去修改根節點 people 的最大權重值為 12，因為這樣才能符合父節點的數值範圍包含子節點數值範圍，因此 people 節點從原來的(2:4, 1, 1:2)被表示為(2:12, 1, 1:2)。而 name 節點與 Addr.節點則左兄弟節點的最大權重值，不能大於右兄弟節點的最小權重值。

（版本 1-3）表示整合版本 1、版本 2 和版本 3，首先將版本 2 從（版本 1-2）擷取出來（擷取方法會在下一節中介紹）與版本 3 比對，比對後發現 Addr.被刪除和新增了 profile，所以先把原先版本 2 而在版本 3 裡沒有變動的資料，修改時間戳記的版本終止值為 3，版本起始值則維持不變，Addr.節點及子節點的版本終止值更改為 2。比較要注意的是在版本 3 裡，profile 節點是被新增在 name 節點的右方，如果要將版本 3 合併至（版本 1-2）

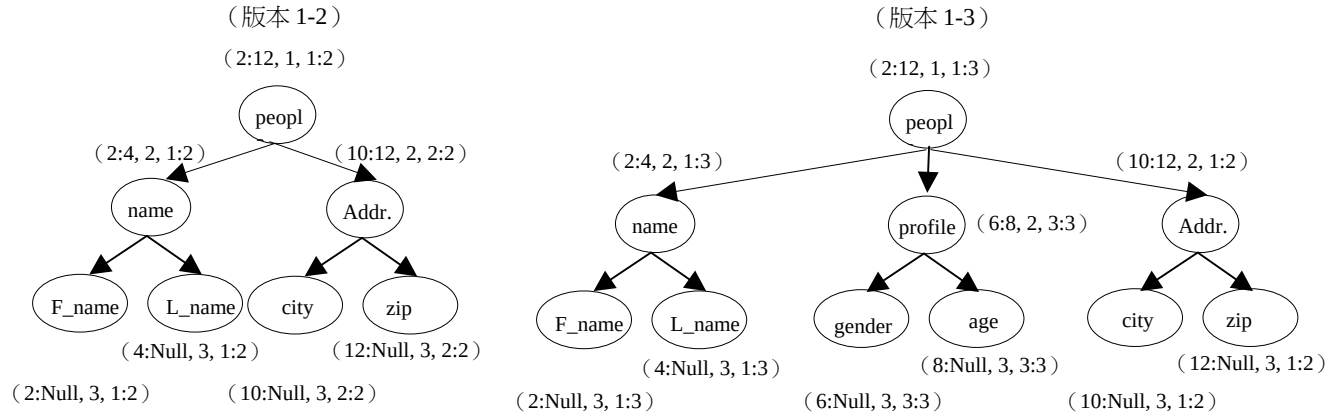

Figure 4: 整合性 XML 文件版本（2）

時，profile 節點要選擇插入在 Addr.節點的左邊或者是右邊呢？其實這個問題很簡單，我們只要確保 profile 節點，是在 name 節點的右方就可以了，因為 Addr.節點是不會出現在版本 3，所以並不會影響擷取 XML 文件版本的順序性。在這裡我們是選擇插入在 name 節點和 Addr.節點中間。（版本 1-3）的維護方法是[20]的情況 1，處理方式也是方法 1，分配時只要注意 profile 節點的最小權重值不能大於 name 節點的最大權重值，而最大權重值不能超過 Addr.的最小權重值。profile 節點及子節點 gender 和 age 的時間戳記則為 3 到 3，分別表示成(6:8, 2, 3:3)、(6:Null, 3, 3:3)、(8:Null, 3, 3:3)。

4.2 儲存結構及版本擷取

在前面的章節中，我們已經討論了如何替有順序性的 XML 文件編碼，好讓 XML 文件可以存入關聯式資料庫中。此外也討論了有關時間戳記用在 XML 文件版本的問題上，準確且忠實地記錄資料在每個 XML 文件版本時間上的位置。有了這兩個空間與時間的位置資訊後，我們就能很輕易地從關聯式資料庫中直接擷取任何的 XML 文件版本。更重要的是，在查詢 XML 文件版本的內容時，能在不需要恢復版本的情況下就取得結果。接下來將介紹我們如何定義關聯式資料庫的欄位。

XML 文件樹裡，一個節點擁有了數值範圍資料和時間戳記，其內容分別為（最小權重值：最大權重值, 階層值）和（版本起始值：版本終止值），並且加上節點所擁有的元素值。我們將這些項目分別對映到資料表的 6 個欄位。Table 1 顯示了這 6 個欄位，另外資料表記載了 10 筆記錄，這些記錄是從 Figure 4 的（版本 1-3）XML 文件樹所有節點的資料。例如資料表的第一筆記錄是節點 people 的資料，而空間上的資料有最小權重值、最大權重值、階層值分別為 2、12 和 1，時間上的有版本起始值、版本終止值分別為 1 和 3。請注意到有些記錄的時間戳記的資料是 1 到 3、1 到 2 和 3 到 3，這是因為每一筆記錄在不同時間的 XML 文件版本，被新增、刪除、修改的結果，並且也符合我們之前所提：資料刪除只是邏輯上的刪除，資料並不會真正的消失。另外資料表的主鍵是最小權重值、最大權重值、階層值、版本起始值和版本終止值所共同組成的。因為當我們修改任一節點資料時，新資料會繼承原來資料的數值範圍，並產生新的時間戳記和元素，成為一筆新記錄。所以如果以最小權重值、最大權重值和階層值為主鍵，將違反關聯式資料模式唯一性原則。

Table 1. 儲存（版本 1-3）關聯表

最小權重值	最大權重值	階層	版本起始值	版本終止值	元素
2	12	1	1	3	people
2	4	2	1	3	name
8	Null	3	3	3	age
10	Null	3	1	2	city
12	Null	3	1	2	zip
6	Null	3	3	3	gender
4	Null	3	1	3	L_name
6	8	2	3	3	profile.
10	12	2	1	2	Addr.
2	Null	3	1	3	F_name

只用簡單的規則就能利用時間戳記來擷取任何一時間的 XML 文件版本，將 XML 文件版本找出。但是資料是存放在關聯式資料模式中，而記錄之間是沒有順序性的，所以當我們利用時間戳記將資料找出後，需執行排序命令，才能恢復原來的順序。

被擷取的 XML 文件版本要符合版本起始值 <= 擷取的版本 <= 版本終止值的規則，只要記錄在此一範圍內，都是被擷取的對象。例如我們想要擷取版本 2 的 XML 文件，將上述規則對照 Table 1 的版本起始值與版本終止值後，符合條件的有 people、name、F_name、L_name、Addr.、city 和 zip 共 7 筆記錄。請注意，這些資料的順序並不是 Figure 2 版本 2 的順序。解決的方法是先對最小權重值作排序後，當有相同的最小權重值時，再對階層值作

排序就能比較出緊鄰的關係。Table 2 說明了執行的結果。

Table 2. 擷取版本 2 關聯表（排序後）

最小權重值	最大權重值	階層	版本起始值	版本終止值	元素
2	12	1	1	3	people
2	4	2	1	3	name
2	Null	3	1	3	F_name
4	Null	3	1	3	L_name
10	12	2	1	2	Addr.
10	Null	3	1	2	city
12	Null	3	1	2	zip

最後 Figure 5 顯示了 XML 文件版本管理的流程圖。一開始從關聯式資料庫利用擷取規則取出版本 N-1；和版本 N 作比對，對於未更動的資料將版本終止值加 1；至於新增和修改的資料，給予適當的數值範圍和目前時間的時間戳記；刪除的資料無須更動數值範圍和時間戳記，將這些資料定位後存入關聯式資料庫。

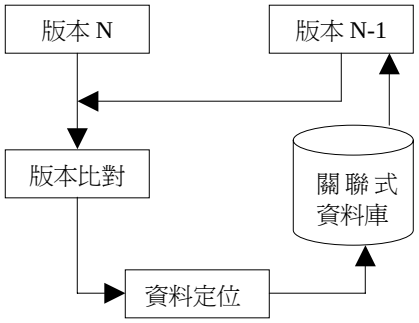

Figure 5: XML 文件版本管理流程圖

5. 查詢

目前 XML 查詢語言基本上都是以正規的路徑表示式或 XPath [18]的方式來說明所要查詢的資料。所以在說明如何執行查詢前需要先討論如何處理路徑問題。在關聯式資料庫裡所記錄的都是元素在 XML 文件的位置，而不是 XML 文件樹的完整路徑資訊。所以在查詢處理上，我們必須將查詢路徑切斷，以 2 個元素為一組的方式分段來處理，並且利用每個元素的最小權重值和最大權重值來判斷父子或祖孫之間的關係，而階層值則為判斷是否有緊鄰關係。並且利用結構結合(structural join) [2]的方式處理，經過多次結合處理後所得到的結果即為符合的查詢路徑。執行結合處理時，比對數值範圍就可以得知兩筆資料所代表的兩個元素是否在同一路徑上。以版本 2 為基礎的查詢路徑 people/name/F_name，分段為 people/name 和 name/F_name。我們必須先從關聯式資料庫找出 people、name 和 F_name 的數值範圍，但先決條件是要在版本 2 有出現的。這個問題可以使用時間戳記配合擷取 XML 文件版本的規則來解決。假設在資料庫中找到 2 筆 name 元素，時間戳記分別為 1 到 2 和 3 到 4，但我們的查詢條件是版本 2 的 name 元素，所以時間戳記為 3 到 4 的 name 元素是不被取用的。

根據上述的查詢路徑，經由 Table 1 的找尋得知 people 的範圍值為(2:12, 1)，name 為(2:4, 2)，F_name 為(2:Null, 3)。當執行查詢時，我們將先尋找名稱為 F_name 的節點，因此得知有 1 個 F_name 節點其數值範圍為(2:Null, 3)，在此 Null 值只是幫助我們來分辨此元素是否為葉節點。接著再比對名稱為 name 的節點；數值範圍中是否有包含 2 的，由 name 的數值範圍得知(2:4, 2)包含 2，接著再利用階層值的比對證明兩者階層數差 1，則該 name 節點為 F_name 節點的父節點，依相同方式從 Table 1 中尋找 people 的數值範圍中有包含 name 節點(2:4, 2)的，我們找到 people 的數值範圍為(2:12, 1)，並透過階層的比對得知為 name 節點的父節點，因此我們找到一條符合 people/name/F_name 的查詢條件路徑，其數值範圍依序為(2:12, 1)/(2:4, 2)/(2:Null, 2)，Figure 6 為此查詢條件流程圖。

其實上面的查詢還隱藏著另一層意義，一般 XML 文件版本管理的問題裡，當我們想要查詢某個版本的內容時，必須先恢復其原有版本，再作資料找尋的動作。而我們的方法不需恢復 XML 文件版本就能直接地對資料作查詢。其原因是時間戳記和數值範圍擁有時間位置和空間位置的資訊，可以讓我們不用恢復

XML 文件版本，而直接且快速地查詢。由上

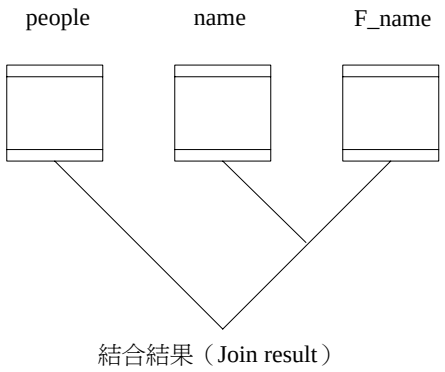

Figure 6: 查詢處理流程圖

面的查詢處理可以看出，我們只要對元素增加時間戳記的判斷就能達成此一目的。

6. 結論

我們利用關聯式資料庫能有效率地處理大量資料的優勢，來儲存有順序性的 XML 文件版本，並且加入時間與空間的概念定位元素的位置，且在不需要恢復 XML 文件版本的情形下，就能直接地查詢任意版本的內容。此外數值範圍除了扮演定位元素的角色外，也能成為 XML 文件版本的索引資訊，加速查詢處理的速度。

誌謝

本研究部分經費由國科會 NSC 92-2422-H-415-007 專案計畫補助。

參考文獻

[1] S. Abiteboul, S. Cluet, et al., “Querying and updating the file,” *VLDB*, 1993.

[2] S. Al-Khalifa, H. V. Jagadish, N. Koudas, J. M. Patel, D. Srivastava, Y. Wu, “Structural Joins: A Primitive for Efficient XML Query Pattern Matching,” *Proceedings of the International Conference on Data Engineering*, 2002.

[3] Peter Buneman, Sanjeev Khanna, Keishi Tajima, Wang-Chiew Tan, “Archiving Scientific Data,” *Proceedings of the ACM SIGMOD International Conference on Management of Data*, June 2002.

[4] Shu-Yao Chien, Vassilis J. Tsotras, Carlo Zaniolo, Donghui Zhang, “Storing and Querying Multiversion XML Documents using Durable Node Numbers,” *Proc. of The 2nd International Conference on Web Information Systems Engineering (WISE)*, Kyoto, Japan, Dec. 2001.

[5] Shu-Yao Chien, Vassilis J. Tsotras, Carlo Zaniolo, “Efficient Management of Multiversion Documents by Object Referencing,” *Proc. of the 27th VLDB*, Rome, Italy, Sep. 2001.

[6] Shu-Yao Chien, Vassilis J. Tsotras, Carlo Zaniolo, “Version Management of XML Documents,” *The World Wide Web and Databases*, 2000.

[7] D. Florescu, D. Kossman, “Storing and Querying XML Data using an RDMBS,” *IEEE Data Engineering Bulletin*, 1999.

[8] R. Goldman and J. Widom, “DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases,” *Proceedings of the Twenty-Third International Conference on VLDB*, pages 436-445, Athens, Greece, August 1997.

[9] T. Grust, “Accelerating XPath Location Steps,” *ACM SIGMOD*, June 2002.

[10] C. Kanne, G. Moerkotte, “Efficient storage of XML data,” *ICDE*, 2000.

[11] Marian, S. Abiteboul, G. Cobena and L. Mignet, “Change-centric management of versions in an XML warehouse,” *Proc. of the 27th VLDB*, Rome, Italy, Sep. 2001.

[12] J. Shanmugasundaram, K. Tufte, et al., “Relational Databases for Querying XML Documents: Limitations and Opportunities,” *VLDB*, 1999.

[13] I. Tatarinov, Stratis D. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, C. Zhang, “Storing and Querying Ordered XML Using a Relational Database System,” *ACM SIGMOD*, June 2002.

[14] Raymond K. Wong, Nicole Lam, “Managing and querying multi-version XML data with update logging,” *Proceedings of the 2002 ACM symposium on Document engineering*, 2002, McLean, Virginia, USA.

[15] World Wide Web Consortium, “Extensible Markup Language (XML),” See http://www.w3.org/TR/REC-xml , October 2000.

[16] World Wide Web Consortium, “Document Object Model (DOM) Level 1 Specification,” See http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929/, September 2000.

[17] World Wide Web Consortium, “Datatypes for DTDs (DT4DTD) 1.0,” See http://www.w3.org/TR/dt4dtd, January 2000.

[18] World Wide Web Consortium, “XML Path Language (XPath) Version 1.0,” See http://www.w3.org/TR/xpath.html, November 1999.

[19] XML TreeDiff, http://www.aphaworks.ibm.com/formula/xmltreediff, 2002.

[20] 李銘桀, *針對多重路徑查詢建構 XML 文件索引機制*, 碩士論文, 國立嘉義大學資訊工程研究所, 2002.