具後設資料綱要變動適應能力的多版本後設資料管理系統

林宗德，施學琦
雲林科技大學資訊管理系
{g9123702, shihhc}@yuntech.edu.tw

摘要

後設資料有助於資源之檢索與利用，然而其客製化之過程往往需要許多的嘗試和摸索。每次後設資料綱要有所更動，使用者介面和後設資料庫便必須隨之改變，而頻繁的變更，經常會導致資訊系統的開發和維護成本高於當初所預期。為解決此問題，本研究由後設資料系統著手，嘗試在變動頻繁的情況下，仍然能夠以極低的成本快速地變動後設資料系統。本研究所提出的系統架構，以後設資料為核心，當使用者透過親和的維護介面自行修改後設資料綱要後，Web 介面和後設資料庫就會自動跟著改變，不需要系統發展者以人工方式修改任何程式。由於本研究的後設資料庫允許新舊版本並存，使用者不但可以查看最新版本的後設資料，也可以查看過去版本的後設資料，Web 介面會自動配合使用者的選擇而變動。在系統效能方面，本研究亦針對將 XML 文件儲存入關聯式資料庫時，採用不同的順序編碼所造成的影響，進行查詢與更新效能的評估。這個系統架構已成功地應用於運作中的系統，達到快速回應變動需求和降低系統變動成本的目的。

關鍵字：後設資料、多版本、動態系統、順序編碼、效能評估

1 簡介

1.1 研究背景與動機

後設資料(metadata)的起源是由於電子資源的數量和種類同時大增，傳統的圖書編目控制工具已經無法適用於網路出版的電子資源，因此必須發展新的後設資料資源描述格式，來提供電子資源的組織、整理、檢索與利用[13]。國際圖書館聯盟協會對於後設資料的定義為：有關資料的資料(data about data)，可用來協助對網路電子資源的辨識、描述、與指示其位置的任何資料[11]。

建立後設資料時，可以根據一些標準來建立。公眾檢索導向的後設資料標準以都柏林核心集(Dublin Core)為代表，所強調的是描述資料的通用性和簡單性。而另一類的後設資料標準則是以學科為導向的，著重於特定領域資訊的共同需求與著錄標準，如 CIDOC、MDA、CDWA、AAT、ULAN 等[14]。但是目前沒有一種後設資料標準可以完全符合所有的需求，縱使有一致的後設資料標準，其客製化仍有其必要。客製化後設資料的建立往往需要經歷許多的嘗試與摸索。換言之，在後設資料建立的過程中，由於後設資料及其綱要的經常變動，使得使用者介面和後設資料庫也會不斷地跟著變動，以符合其需求。

由於每次後設資料綱要有變動，使用者介面的程式和後設資料庫亦必須跟著變動，尤其當後設資料綱要非常複雜時，系統開發者必須花費大量的精力在使用者需求變動上。這些頻繁的後設資料綱要變動，導致後設資料系統的開發及維護成本高出原來所預期。本研究由後設資料系統著手，在需求變動頻繁的情況下，能夠以極低的成本快速地更改系統，並允許使用者追蹤後設資料的發展過程，及查看新舊版本的後設資料。

本研究是延續我們去年所開發的後設資料管理系統[16]，陸續加入許多功能。去年的系統只實作出前端的 Web 介面自動產生的部份，對於後設資料庫並未提出解決的方案，而且 Web 介面的單元只有 6 種。今年的系統將 Web 介面產生的單元增加到 14 種，分六大類。後設資料庫的部份，以彈性的資料庫結構來儲存，並加入多版本管理功能。另外，提出以唯一路徑編碼方式將 XML 文件儲存到關聯式資

料庫，並且評估了各種不同順序編碼方式對於 XML 文件的查詢和更新效能的影響。

1.2 研究目標

本研究的目的是加強去年所開發的後設資料管理系統，並發展出一個方法，來解決由於後設資料綱要的經常變動，使得後設資料管理系統的使用者介面和後設資料庫要隨之而經常變動，所造成的系統開發和維護成本增加的問題。本研究的目標為：(1)後設資料綱要由使用者自行定義及變動。(2)後設資料綱要變動之後，使用者介面自動隨之變動，不必更動程式，也不用重新編譯程式或產生中間碼，使用者介面可以動態產生。(3)發展一個彈性的資料庫結構，不管後設資料綱要如何變動，不需要變更資料庫綱要也能存入資料庫中。(4)允許使用者追蹤後設資料的發展過程，且可以查看舊版本後設資料。(5)針對以關聯式資料庫儲存 XML 文件的不同順序編碼方式，進行效能評估，以做為本研究及相關研究採用的參考。

本研究的應用範圍，適用於任何後設資料複雜，但工作流程單純，且時常會更動後設資料綱要的後設資料系統。所謂複雜的後設資料，是指後設資料有很多欄位及階層、或很多種分類。而單純的工作流程是指，資料在輸入完畢後可以直接存入資料庫，而不用再經過許多額外的程序。

2 文獻回顧

2.1 快速度開發使用者介面

幫助系統開發者快速地開發應用程式的方法，已經有許多的研究者提出。例如，半自動資料資源出版(semiautomatic data resource publishing)被用來輔助快速開發網站應用程式[3]，後設資料表格(metadata table) 被用來產生動態的網站介面[12]，[7]以 XML schema 來描述資料庫綱要，並以此 XML schema 產生 Java Bean 和 JSP 的程式碼。這些方法在後設資料變動時，雖然可以在使用者介面部份節省修改時間，但是資料庫綱要的變動還是需要以人工的方式修改。

2.2 彈性的資料庫結構

本研究除了要達到使用者介面的快速變動外，在資料庫方面也要以一個很有彈性的結構來儲存後設資料，以達到後設資料變動的適應性(adaptability)。由於本研究是採用 XML 格式來儲存後設資料，因此如何能以一個較有彈性的方法來儲存 XML 文件，便成為一個重要的議題。

將 XML 文件儲存到資料庫的方法，可採用關聯式資料庫、XML 加強型資料庫(XML-enabled databases)和原生型 XML 資料庫(native XML databases)等不同的方式。考量資料庫技術的穩定性及普及性，仍有大量的應用程式採用目前資料庫的主流技術—關聯式資料庫來儲存 XML 文件。但由於關聯式資料庫中的 tuple 沒有順序的觀念，而 XML 文件中的節點間有先後順序的關係。所以存入關聯式資料庫中的資料必須加以順序編碼(order encoding)，以便取出時可以得到與原來順序相同的 XML 文件。

順序編碼的方式大致上可分為本地順序(local order)、整體順序(global order)和 Dewey 順序(Dewey order)三種[10]。本地順序編碼是在每一節點記錄其父節點，以及屬於同一父節點的兄弟節點間的順序。這種編碼方式在 XPath 的四個主要軸來回移動時，必須一層一層地找，查詢的效能不佳，但是更新效能最好。整體順序編碼是以整個 XML 文件的節點去做編號。[4]將 XML 文件以樹狀結構的前序(preorder)和後序(postorder)的方式來為每一個節點編碼。整體順序編碼的好處是可以很快地找到某個節點的所有子孫節點，查詢的效能佳。但是若要新增或刪除節點時，則需要重新編號，更新效能不佳。Dewey 編碼，Tatarinov 等人[10]提出 Dewey 編碼方法，讓查詢和更新兩個原本互斥的效能，得到一個平衡點。Dewey 編碼是把從 root 到該節點間所經過的路徑的本地編號全部以一個字串來表示。這種方式除了可以很快地在 XPath 的四個主要軸找到目標節點之外，在更新節點時需要重新編碼的範圍也會比整體順序編碼小。

以上三種編碼方式，都是將 XML 文

件拆解成許多 tuple，然後全部儲存在同一個表格當中。這種儲存方式，無論後設資料如何變動，資料庫綱要都不必變動，只需要變更資料庫表格的內容即可，將可以得到很有彈性的資料庫結構。

2.3 XML 版本管理

XML 版本管理的目地是可以追蹤 XML 版本的發展過程，讓使用者查看新舊版本的 XML 文件。XML 版本管理的目標是以最小的儲存空間儲存多個版本的文件，並以最快的速度取回指定的版本。

[6,8]把 XML 文件變動的過程記錄下來，若要取出前版本的文件，可以由變動過程的記錄往回或往前推算。[2]把第一個版本完整地保存下來，當文件變動時只另外儲存有變動的部份。對於沒有變動的部份則是用 Reference Record 指向前一版本的相同位置。這幾種方法都無法直接取出中間的某一個版本的內容。[1]以時間戳記(Timestamp)來做版本管理。記錄每一個節點的開始時間和結束時間。這個方法可以直接取得任何一個版本的 XML 文件。[15]將這個方法套用在關聯式資料庫儲存 XML 文件上，但其缺點是每個 tuple 都必須記錄結束時間，所以當任何 tuple 有變動時，整個新版本的 tuple 都必須更新結束時間。

3 系統架構概觀

本研究之系統架構如圖 1。此系統架構和去年的系統不同之處在於今年的系統增加了一個後設資料伺服器。它除了包含原來的使用者介面產生器之外，還增加了連結後端資料庫的功能、處理版本管理的機制，以及提供檔案上傳的功能。另外，新的架構是以彈性的資料庫結構儲存 XML 文件。接下來我們簡單地介紹此系統架構的運作情形。

首先，使用者在建立後設資料時，應依照我們去年之系統[16]所定義的後設資料描述規範，建立 XML Schema 和 Web 介面後設資料兩個文件。接著，本研究設計了一個後設資料伺服器，這個後設資料伺服器是建立在 J2EE 的架構上，其功能包括：使用者介面產生器(UI Generator)、Java Script、版本管理、資料庫連結及檔案傳輸。當使用者想要進入 Web 介面時，從客戶端送出請求給後設資料伺服器，由其中的使用者介面產生器負責產生 Web 介面(包括新增介面、修改介面、查看介面及搜尋介面)的工作。從傳入的參數中，使用者介面產生器可以得知應該讀取那個 Web 介面後設資料文件，由此文件中取得產生 Web 介面所需的資訊，動態地產生 Web 介面，並傳回客戶端。客戶端瀏覽器會讀取後設資料伺服器中經過一般化的 Java Script 模組，來控制使用者和系統間的互動。當使用者輸入資料完畢之後，Java Script 會把使用者所輸入的資料組合成一份 XML 文件，並將此 XML 文件傳回給後設資料伺服器。後設資料伺服器接著透過資料庫連結和版本管理功能把此 XML 文件送給後端的資料庫並決定那些元素應該存成新版本進行儲存的動作。版本管

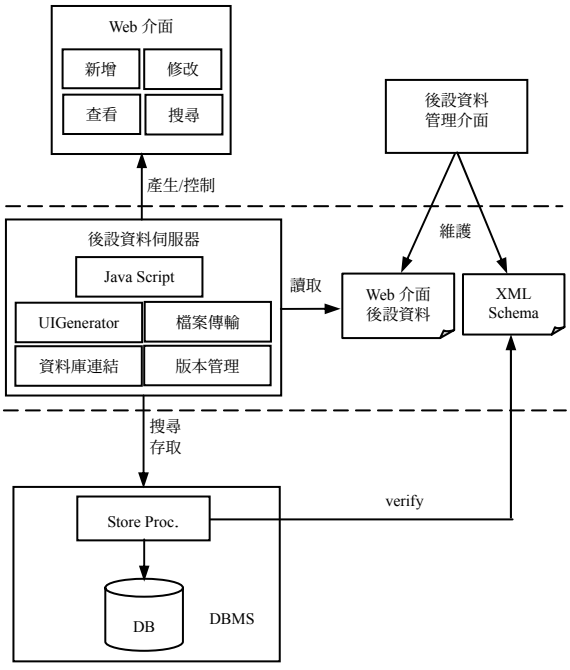

圖 1 系統架構圖

理的機制，還可以讓使用者追蹤後設資料的發展過程。使用者若指定要查看某個版本的文件，後設資料伺服器會讀取指定版本的 Web 介面後設資料文件及資料庫內容，產生該版本的 Web 介面和後設資料內容。另外，有許多的後設資料欄位存放的資料格式屬於各種類型的檔案，這些檔案

可以透過後設資料伺服器的檔案傳輸功能上傳到伺服器中，並在資料庫中儲存檔案名稱和儲存位置，以便取得檔案。由於使用者可能不懂 XML Schema 文件和 Web 介面後設資料文件的描述規則。所以本研究設計了一個後設資料管理介面(也請參考[16]之說明)，來幫助使用者很容易地建立和變動後設資料綱要。使用者透過後設資料管理介面變更後設資料綱要時，會同時變更 Web 介面後設資料文件和後設資料內容。以下就針對各個部份做說明。

4 使用者介面產生器與 JavaScript

本研究採用 XML Schema 來描述後設資料綱要，並在每個節點後面增加 Web 介面資訊，以提供使用者介面產生器動態地產生 Web 介面。使用者介面產生器是一支 Enterprise Java Bean 元件，用來產生 Web 介面。其詳細的後設資料描述規範與 Web 介面產生流程請參考[16]。

去年之系統可以動態產生的 Web 介面單元共有六種：沒有值、單值文字方塊(text box)、單值文字區塊(text area)、多值文字方塊、下拉式選單、複雜多值。

今年本研究則增加為 14 種，依照資料結構的不同將網頁內容單位分成六大類，分別為：(1)沒有值、(2)沒有限制值的範圍(單值文字方塊、單值文字區塊、多值文字方塊、多值文字區塊)、(3)有限制值的範圍(下拉式選單、下拉式選單加文字方塊、多值下拉式選單)、(4)相依的值(一對一、一對多)、(5)呼叫外部程式(ActiveX、查看圖片、檔案上傳)、(6)複雜多值。

以下我們針對今年新增的 Web 介面單元做介紹。在第二類：沒有限制值的範圍，這裡多了一種單元叫做多值文字方塊，如圖 2 所示。允許新增文字區塊，可放入多篇文章，並上下筆移動。在第三類，有限制值的範圍，新增二個單元。當下拉式選單中的選項沒有使用者想要的選項時，可以選擇“其他”，系統會帶出一個文字方塊讓使用者自行輸入，如圖 3 所示。圖 4 則是多值下拉式選單。 第四類，相依的值。某些欄位的值的範圍，是由其他欄位的值所決定。例如，使用者在縣市欄位中選擇“雲林縣”，那麼鄉鎮欄位中出現的選項應該是屬於雲林縣內的鄉鎮。以圖 5 為例，使用者在第一個下拉式選單所選擇的項目，決定了第二個下拉式選單的選項。第二個下拉式選單所選擇的項目，決定了第三個下拉式選單的選項，以此類

圖 2 多值文字方塊

圖 3 下拉式選單加文字方塊

圖 4 多值下拉式選單

圖 5 第四類，相依的值

圖 6 呼叫 ActiveX

圖 7 呼叫檢示圖片

圖 8 呼叫檔案上傳

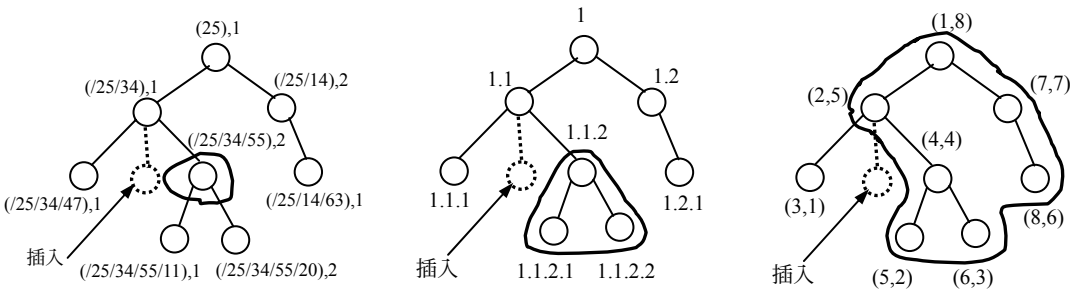

圖 9 唯一路徑編碼、Dewey 順序編碼和整體順序編碼在插入新節點的情況

推，這是一對一的情況。另一種相依的值是一對多，使用者在某個下拉式選單選擇某個選項之後，會同時決定二個或以上的其他下拉式選單的選項。 第五類，呼叫外部程式。某些欄位需要呼叫外部的程式來協助使用者完成輸入的工作。例如，呼叫 ActiveX 物件(圖 6，呼叫以 ActiveX 元件實作的中西曆對照表)、檢視圖片(圖 7)、呼叫後設資料伺服器的檔案上傳元件(圖 8)。

5 資料庫設計與版本管理

5.1 唯一路徑編碼

為了改善 Dewey 編碼的缺點，本研究提出另一種編碼方式，以唯一路徑(unique path)加上本地順序編碼的方式來儲存。所謂的唯一路徑，是事先賦予每個節點一個唯一的數字，由 root 開始到目前節點位置所經過的路徑，把所有經過的節點的唯一數字連接起來用"/"隔開。這樣在搜尋子孫節點和祖先節點時，只要比對唯一路徑的前段是否相同即可，一樣可以在短時間內搜尋出結果。但是唯一路徑無法表示出子節點的順序，所以我們再加上本地順序編碼來區分出子節點間的順序。

以圖 9 為例，左圖是唯一路徑編碼，括號中的字串為唯一編碼，逗點右邊為本地順序編碼。中間的圖為 Dewey 順序編碼。右圖為前序-後序編碼，括號內左邊為前序，括號內右邊為後序。Dewey 順序編碼在搜尋 1.1 節點的子孫節點時，可以在 SQL 中以 dewey LIKE '1.1.%'的方式找出所有的子孫節點。而唯一路徑也可以用相同的方式 uniquepath LIKE '/25/34/%'，找出所有的子孫節點。前序-後序編碼則是用 pre > 2 and post < 5，找出其子孫節點。

在插入一個新節點的情況，Dewey 順序編碼應給圖中新節點的編碼為 1.1.2，另外需要變動編碼的範圍為圖中圈起來的三個節點。前序-後序編碼則需要把所有 pre 大於 4 的節點，其 pre 都加 1；和所有 post 大於 2 的節點，其 post 都加 1。而在唯一路徑編碼中插入一個新節點的情況時，對於新的節點可以給任意的唯一數字，例如 66。所以新節點的編碼應為 (/25/34/66),2，需要變動編碼的節點只有一個，只需把圖中圈起來的節點的本地順序編碼改為 3 即可。

5.2 版本管理與後設資料存取

為了可以追蹤後設資料的發展過程，及查看舊版本的資料，本研究加入了版本管理的機制。在後端資料庫我們增加二個表格，schema_version_table 記錄後設資料綱要的每一次變動時間；data_version_table 記錄後設資料內容的每一次變動時間。在 Uniquepath_table 部份，除了原來的資料表格欄位之外，另外增加兩個欄位 tstart 和 tend。tstart 是此節點的開始版本，tend 是此節點的結束版本。最新版本的 tend 欄位為 null，故只要此節點沒有變動過，系統完全不用去更新這個 tuple，只需針對有變動的節點去做更新即可，減少後設資料變動時，需要更新的 tuple 數。這部份改善了[15]之缺失。

當使用者想要新增或更新資料內容時，假設後設資料綱要沒有變動，使用者在 Web 介面中按下存檔按鈕，會啟動一支 JavaScript 程式。此程式的功能是把使用者在 Web 介面中所輸入的資料組合成一個 XML 文件，並將此 XML 文件傳回給後設資料伺服器。後設資料伺服器先至 data_version_table 確認最後的版本，接著把此 XML 文件送給後端資料庫的內儲程

序(Store Procedure)，內儲程序收到此XML文件之後，先讀取 XML Schema，確認此 XML 的有效性。再利用 DOM 剖析器剖析此 XML 文件，依照拆解後的節點至 Uniquepath_table 對照此節點最後版次的內容是否已經變更。如果是變動過的節點，須先將原來記錄的 tend 欄位加上前版次，再新增一筆更新過的記錄，其 tstart 為最新版次，tend 為 null。最後在 data_version_table 中加入最新版次記錄。

如果是後設資料綱要有變動的狀況下，本研究定義了「新增節點」、「新增末子節點」、「刪除節點」、「複製節點」、「貼上節點」、「貼上末子節點」等六個動作。當使用者對後設資料綱要有所變動時，首先系統會到 schema_version_table 尋找最後的綱要版本，把舊版本的 Web 介面後設資料檔名後面加上版本編號另存檔案，如 xxx.ui~1，xxx.ui~2…，而將最新版本的檔名存成 xxx.ui。採用這種簡單的作法是因為 Web 介面後設資料的資料量並不是很大，而且這樣可以減少剖析舊版本節點的時間。接著系統必須更動後設資料庫的內容。如果是新增節點，則在 Uniquepath_table 中每一 XML 文件的相同節點位置新增一筆記錄，tstart 記錄為最新版次。如果是刪除節點，則在 Uniquepath_table 中每一 XML 文件的相同節點位置的 tend 記錄為前版次。最後再分別新增新版本記錄到 data_version_table 和 schema_version_table 中。

當使用者要求讀取最新版本的 XML 文件時，使用者介面產生器應該讀取無版本編號的 Web 介面後設資料檔名，如 xxx.ui。接著到 Uniquepath_table 中搜尋該文件中所有 tend 為 null 的 tuple(tend 為 null 的 tuple 即為最新版本)，重新組合成 XML 文件送回後設資料伺服器。若使用者要求查看過去某個時間點的版本時，要在條件式中加入 tstart<=n and (tend>=n or tend is null)，n 為 data_version。取出之後重新組合成 XML 文件送回後設資料伺服器，此時使用者介面產生器應讀取的後設資料檔名應為，xxx.ui~n，n 為 schema_version。

另外，此系統是以全文搜尋的方式供使用者搜尋資料。系統會從 node_name 和 node_value 兩個欄位來搜尋是否有符合關鍵字的記錄。

6 不同順序編碼方式之效能評估

以不同的順序編碼儲存 XML 文件到關聯式資料庫中，在查詢和更新的效能會有不同的表現。故本研究針對不同的順序編碼方式做效能之評估。本研究所需要之評估標竿的特性應為：(1)必須要有更新效能的評估標竿；(2)必須是資料導向的評估標竿；(3)不能是針對某個應用程式來做評估，應該是要能觀察資料庫本身的效能，也就是微細層面的評估標竿。我們針對目前已被提出的數個評估標竿(Xbench 標竿、Xmach-1 標竿、Xmark 標竿、X007 標竿、Michigan 標竿)進行調查，結果只有 Michigan 標竿[9]符合本研究的需求。故本研究以 Michigan 標竿來進行實驗的評估標竿。

本研究的實驗環境如下，中央處理器 2.4GHz，隨機記憶體 1Gb，作業系統 Windows 2000，資料庫 Oracle 9i 第二版。關於實驗對象的挑選，由於本地順序編碼的查詢需要一層一層地往下搜尋，很明顯地會得到一個極差的效能，故本研究不將它納入實驗對象。在整體順序編碼部份，延伸出許多的方法，如 (preorder, postorder)、(order, size)和相對區域座標。這些延伸出來的方法基本上和前序-後序編碼 (preorder, postorder) 的本質是一樣的，所以我們就以前序-後序編碼方式為代表當作整體順序編碼的實驗對象。Grust[4]曾經提出前序-後序編碼方式的增進效能的方法。為了得到更佳的查詢效能，除了對 pre 和 post 兩個欄位以 b-tree 做索引之外，還可以對 pre 欄位做叢集(cluster)加快讀取資料的速度。或者是以 r-tree 對 pre 和 post 做索引，用空間的觀念來取得目標節點，也可以得到很好的效能。所以我們加上這兩種方法當做實驗對象。Grust 等人在最近發表了另一篇論文[5]，針對前序-後序編碼方式的效能再做改進。文中提到將前序和後序的編碼數字拉長(stretch)，讓

pre(v) < pre(v') < post(v)而且 pre(v) < post(v') < post(v)；v'為 v 的所有子孫節點。這樣只要以 pre 欄位或 post 欄位來搜尋，不必同時使用兩個欄位比對，加快搜尋的速度。例如在 where 子句中加入 pre(v') > pre(v) and pre(v') < post(v)。所以本實驗再加上這個編碼方法當作實驗的對象。最後，加上 Dewey 編碼和本研究提出來的唯一順序編碼為實驗對象。因此本研究的實驗的對象有：(1)原始的前序-後序編碼、(2)經過叢集的前序-後序編碼，(3)以 r-tree 索引的前序-後序編碼，(4)經過拉長(stretched)編碼的前序-後序編碼、(5)Dewey 編碼、以及(6)唯一路徑編碼。

為了讓實驗的結果更符合實際的情況，我們做了一些變數的控制。(1)為了不讓我們所寫的 SQL 影響到實驗的正確性，我們使用 Oracle 所提供的基於成本的最佳化處理器(Cost-based Optimizer，簡稱 CBO)，它會幫我們做 SQL 的最佳化動作。(2)在進行「經過叢集的前序-後序編碼」這個實驗對象時，需要建立一個叢集。為了觀察不同的叢集因子(clustering factor)會不會造成不同的效能結果，我們也對這個部份做了簡單的實驗。實驗後發現對於效能不會有太大的影響，所以使用內定值產生叢集。(3)在進行「以 r-tree 索引的前序-後序編碼」實驗對象時，我們是以點的型態儲存前序和後序編碼，也就是 Oracle 的 MDSYS.SDO_POINT_TYPE，以加快查詢的速度較快。 (4)實驗中若測試單元需要多個 SQL 指令才能完成，我們把它寫成一個預儲程序，讓工作送出後一次做完才傳回結果，不會使來回傳送時間影響真正的效能表現。(5)本研究所有的測試單元都是單獨執行，單獨的資料庫連結。讓每個測試單元間互相影響的變數降到最低。

測試單元共有 53 個，可分成七類。簡單搜尋(qs1-qs14)，結構上的搜尋(qs15-qs27)，複雜型態的搜尋 (qs28-qs35)，傳回的結構(qr1-qr4)，結合查詢(qj1-qj4)，聚合查詢(qa1-qa5)，更新(qu1-qu5)。括號中的代號為 Michigan 標竿的測試單元代號。每一個測試單元都測試 5 次，去掉最高和最低值，並平均後得該測試單元的結果。為了節省篇幅我們再將 53 個測試單元的實驗結果依照七類加以平均，得到表 1 的結果。表中 *prepost* 表示原始的前序-後序編碼， *prepostC* 表示經過叢集的前序-後序編碼，*prepostR* 表示以 r-tree 索引的前序-後序編碼，*prepostS* 表示經過拉長編碼的前序-後序編碼，*Dewey* 表示 Dewey 編碼，*uniquepath* 表示唯一路徑編碼。單位時間是毫秒(milli second)。

表 1 實驗結果

	qa1~5	qj1~4	qr1~4	qs1~14	qs15~27	qs28~35	qu1~5
prepost	2,125	1,343	11,205	337	4,563	25,319	183,057
prepostC	14,331	4,722	19,495	1,101	15,670	52,427	801,697
prepostR	2,365	1,524	11,981	381	36,252	13,169	9,155,505
prepostS	2,088	1,317	8,681	330	4,432	24,755	188,400
Dewey	1,593	948	2,038	323	1,885	5,489	75,070
uniquepath	1,748	1,154	2,420	343	2,411	7,423	56,563

我們觀察實驗的結果，查詢的部份平均來講以 Dewey 編碼和唯一路徑編碼的效能最好。這兩者的查詢方式事實上是相同的，但由於此資料組的元素數目非常的大(超過 60 萬)，使得唯一路徑的長度比 Dewey 的長度還長，在資料比對的速度上造成些微的差距。前序-後序編碼的查詢效能在簡單搜尋的測試單元中和 Dewey 的表現差不多。但是在其他的查詢測試單元中出人意料地比 Dewey 和唯一路徑的效能差。追究其原因，prepost 查詢其所有子孫節點時，在 SQL 的 where 子句中會有類似這樣的語法 *child.pre > parent.pre AND child.post < parent.post*。若在運算子的兩邊都出現 column 名稱，則會自動忽略索引。我們也曾經在 SQL 中加入暗示(hint)，讓資料庫強迫使用索引來查詢，但是效能反而更差。

觀察「經過叢集的前序-後序編碼」，如果查詢的結果剛好都是聚集儲存在附近的區塊，效能就特別的好。反之則效能特別差。「以 r-tree 索引的前序-後序編碼」在某些測試單元中其效能則明顯變差，其原因我們則無法得之。而「經過拉長編碼

的前序-後序編碼」的效能的確比原始的前序-後序編碼有所改善，因為它只需用一個 column 去做比對，但改善的幅度並不大。

接下來，我們觀察更新的效能。由於四種前序-後序編碼都是屬於整體順序編碼，更新效能如預期中表現得不好。而在 Dewey 和唯一路徑編碼兩者之間，則以唯一路徑編碼較佳，因為唯一路徑在更新時需要更動到的節點是最少的。

7 結論

本系統架構有很好的擴充性。除了前面六大類剖析狀況所提到的元件外，其他如 check box、radio button…等元件也都可以任意地加入；或是在文字方塊中限制使用者輸入的值的範圍(domain)。另外，也可以引用其他資源的權威檔，例如，人名、地名等。本研究的系統架構已成功地應用在數位典藏國家型科技計畫的國立歷史博物館數位典藏系統的後設資料建置子系統，以及經濟部工業局的創意家具設計輔導與推廣計畫的資料庫建置子系統。本研究證明這個架構可以減少因為後設資料綱要的時常變動而產生的維護成本增加。並且可以讓使用者有更多的自由來進行後設資料變動，及追縱後設資料的發展過程。

參考文獻

[1] Buneman, P., S. Khanna, K. Tajima and W.C. Tan, Archiving Scientific Data, Proceedings of the 2002 ACM SIGMOD international conference on Management of data, 2002, pp.1-12

[2] Chien, S.Y., V.J. Tsotras and C. Zaniolo, Efficient Management of Multiversion Documents by Object Referencing, Proc. Of the 27th VLDB, Rome, Italy, 2001

[3] Fraternali, P., Tools and Approaches for Developing Data-Intensive Web Applications: A Survey, ACM Computing Survey, 1999, vol.31, no.3, pp.227-263

[4] Grust T., Accelerating Xpath Location Steps, ACM SIGMOD, 2002, June 4-6, pp.109-120

[5] Grust, T., J. Teubner and M. van Keulen, Accelerating XPath Evaluation in Any RDBMS, in ACM Transactions on Database Systems (TODS) , March 2004 (to appear)

[6] Marian, S., G. Abiteboul, L. Cobena and Mignet, Change-centric management of version in an XML warehouse, Proc. Of the 27th VLDB, Rome, Italy, 2001

[7] Milosavljević, B., M. Vidaković and Z. Konjović, Automatic code generation for database-oriented web applications, ACM International Conference Proceeding Series: In Proceedings of the inaugural conference on the Principles and Practice of programming, 2002, pp.59-64

[8] Raymond, K. and N.L. Wong, Managing and querying multi-version XML data with update logging, Proceedings of the 2002 ACM symposium on Document engineering, McLean, Verginia, USA, 2002

[9] Runapongsa, K., J.M. Patel, H.V. Jagadish and S. Al-Khalifa, “The Michigan Benchmark: Towards XML Query Performance Diagnostics”, http://www.eecs.umich.edu/db/mbench/bmShortV.pdf, Feb. 2002

[10] Tatarinov, I. et. al., Storing and Querying Ordered XML Using a Relational Database System, ACM SIGMOD, 2002, pp.204-215

[11] Weibel, S., J. Godby and E. Miller, OCLC/NCSA Metadata Workshop Report, http://www.oclc.org/oclc/research/conferences/Metadata/dublin_core_ report.html, 1997

[12] Weiner, M., M. Sherr and A. Cohen, Metadata Tables to Enable Dynamic Data Modeling and Web Interface Design: The SEER Example, International Journal of Medical Informatics, 2002, vol. 65, issue 1, pp.51-58

[13] 王麗蕉，Metadata 初探，東吳大學圖書館通訊，1999，第 8 期

[14] 陳亞寧、陳淑君，Metadata 在數位博物館之發展與分析，圖書館學與資訊科學，2001，第 27 卷，第 2 期，pp.52-71

[15] 陳耀輝、蔡政霖，XML 文件版本管理，第二屆數位典藏技術研討會，台北、台灣，2003，pp.149-156

[16] 林宗德、陳宏儒、施學琦，由後設資料驅動的動態著錄系統，第二屆數位典藏技術研討會，台北，2003，pp. 181-188.