國立交通大學

電機學院與資訊學院 資訊學程

碩士論文

XML 文件搜尋引擎的研究

Study on the Search Engine of the XML Documents

研 究 生：蕭遜文

指導教授：李素瑛　教授

中華民國九十五年八月

XML 文件搜尋引擎的研究
Study on the Search Engine of the XML Documents

研 究 生：蕭遜文　　Student：Hsun-Wen Hsiao
指導教授：李素瑛　　Advisor：Suh-Yin Lee

國 立 交 通 大 學
電機學院與資訊學院專班 資訊學程
碩 士 論 文

A Thesis
Submitted to Degree Program of Electrical Engineering and Computer Science
College of Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of
Master of Science
in
Computer Science
August 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年八月

XML 文件搜尋引擎的研究

學　　生：蕭遜文　　指導教授：李素瑛博士

國立交通大學 電機學院與資訊學院 資訊學程（研究所）碩士班

摘　　要

傳統的搜尋引擎主要以關鍵字查詢為主，雖然提供布林運算式的查詢，但無法查詢關鍵字在文件中的順序（order）關係。XML 文件搜尋引擎查詢時，除了必須具備傳統搜尋引擎關鍵字的查詢功能之外，必須考量 XML 文件中資料的階層關係，因此查詢時需透過由 W3C 所制定的 XML 查詢語言 XPath[5]的語法查詢關鍵字在 XML 文件中的順序關係，以彌補傳統搜尋引擎所欠缺的功能。

本論文主要在大量 XML 文件資料庫加速查詢，利用所謂 Begin-End-Level（BEL）[22]區間編碼方式，建立 XML 文件資料庫的索引結構。XML 文件經 BEL 編碼之後，將索引資料值儲存於關聯式資料庫系統（Relation DataBase Management System，RDBMS）。再利用 XPath 表示式轉換為 SQL Command 存取資料庫，重建（reconstruct）所得到的資料錄（records），可以獲得跟原先一致的 XML 文件內容。

為了加速 XML 搜尋引擎的查詢，引入 signature file 的索引機制，做為過濾機制，過濾掉不必要的資料庫查詢。

Study on the Search Engine of the XML Documents

Student：Shun-Wen Hsiao　　Advisor：Dr.Suh-Yin Lee

Degree Program of Electrical Engineering and Computer Science
National Chiao Tung University

ABSTRACT

Traditional search engine mainly query by keywords.Although Boolean operations are provided, it is unable to query the ordering of keywords or attributes in XML documents. In XML document serach engine, besides the keyword query function of the traditional search engine, the ordering of data or the hierarchical relation of data in the XML documents must also be considered. XML query strings expressed in Xpath, which is the W3C XML query language, can query the order of keywords,the structure of XML documents.

In this thesis, we are focused on the speed up of query operations in large XML documents database. We use Begin-End-Level (BEL) interval encoding method to build the index structure for each XML document.After the BEL coding of the XML documents, the indexes are saved into Relation Database. The query in the XPath expression is transformed into the SQL query Commands. The stored records can reconstruct the original and consistent contents of XML documents.

In order to speed up the query, the index mechanism of signature file is employed to filter out the unqualified documents first and avoid nonessential query operations.

誌謝

感謝指導教授李素瑛博士，口試委員許芳榮博士及陳正教授。也感謝摯友王絲品小姐，邱梅芳小姐以及小孩的媽。在我最困頓迷惘之時，幫助我完成學業。

目　　錄

中文摘要 ..... **I**
英文摘要 ..... **II**
誌謝 ..... **III**
目錄 ..... **IV**
表目錄 ..... **V**
圖目錄 ..... **VI**
一、 緒論 **1**
1.1 研究動機與目的 1
二、 背景 **3**
2.1 XML 3
2.1.1 XML的文件結構 4
2.1.2 XML的處理 5
2.1.2.1 文件物件模型（Document Object Model，DOM） 5
2.1.2.2 簡單XML應用程式介面（Simple API for XML，SAX） 6
2.1.3 XML的儲存 8
2.2 搜尋引擎 10
2.2.1 搜尋引擎的概念 10
2.2.2 XML的搜尋引擎 12
2.3 XPATH 12
三、 XML文件索引與查詢 **14**
3.1 XML文件的索引方法（INDEXING）探討 14
3.1.1 節點紀錄類索引 14
3.1.1.1 利用節點序號的索引 15
3.1.1.2 利用節點路徑訊息的索引 18
3.1.1.3 混合式索引 18
3.1.2 結構摘要類索引 20
3.2 簽章索引檔（SIGNATURE FILE INDEX） 20
3.2.1 簽章索引檔概念 21
3.2.2 簽章索引檔種類 22
3.2.3 雜湊函數的編碼 24
3.3 XML文件查詢 26
3.3.1 XPath expression 與SQL的轉換 26
四、 系統實作 **31**
4.1 系統架構 31
4.1.1 系統架構圖 31
4.1.2 系統環境 32
4.2 系統說明 32
4.3 系統展示 37
五、 結論 **40**
參考文獻 **42**

表目錄

表 1 STUDENTS.XML的元素混合編碼表 19
表 2 STUDENTS.XML的路徑編碼表 19
表 3 系統實作相關數據 37

圖目錄

圖 1 一個簡單的XML文件STUDENTS.XML 3
圖 2 文件STUDENTS.XML的樹狀結構 5
圖 3 DOM介面運作模式 6
圖 4 SAX介面運作模式 7
圖 5 SAX解析程式的事件觸發順序 8
圖 6 搜尋引擎架構 11
圖 7 符合XPATH查詢的子節點集合 13
圖 8 利用前序後序所取得節點序號 15
圖 9 BEL區間編碼及DEWEY局部編碼 16
圖 10 節點路徑訊息 18
圖 11 STUDENT.XML的結構摘要 20
圖 12 個別節點利用特殊HASHING FUNCTION 編碼轉成BITSTRING 21
圖 13 FILTER的概念應用 22
圖 14 WORD SIGNATURE方法 23
圖 15 SUPERIMPOSED CODING方法 24
圖 16 雜湊函數的應用－雜湊表 25
圖 17 利用BLOOM FILTER建立BIT STRING 25
圖 18 SUPERIMPOSED CODING的BLOCK對XPATH查詢字串的影響 26
圖 19 系統架構圖 31
圖 20 BEL編碼建立XML文件索引 33
圖 21 線上XML文件索引建立系統 34
圖 22 批次XML文件索引建立程式畫面 35
圖 23 關聯式資料表DOCUMENTS的SCHEMA 35
圖 24 關聯式資料表ELEMENTS的SCHEMA 35
圖 25 關聯式資料表TEXTS的SCHEMA 36
圖 26 關聯式資料表ATTRIBUTES的SCHEMA 36
圖 27 XPATH表示式查詢結果 37
圖 28 查詢中文資料結果 38
圖 29 未經由SIGNATURE FILE的失敗查詢 38
圖 30 經由SIGNATURE FILE的查詢結果 39

一、緒論

1.1 研究動機與目的

可擴展標示語言 XML（eXtensible Markup Language）[1]是全球資訊協會 W3C（World Wide Web Consortium）在 1996 年底所制定的標準，主要是用來克服超文字標示語言 HTML（HyperText Markup Language）[2]的限制，因為 XML 具備儲存與表達資料的結構意義，也可以訂定資料的型態，更重要的是 XML 是跨平台的，使不同平台資料的交換更為容易與方便。

近幾年來，受到網際網路盛行的影響，HTML 成為廣受歡迎的標示語言（Markup Language），大量的網頁都是根據 HTML 規格產生的，這些 HTML 文件透過 Web 經由網際網路將資訊傳送給遠端利用網頁瀏覽器（Web Browser）瀏覽網頁的使用者（end user）。而 Web 原本只是一個用來發佈科學文件的工具，如今已發展成當紅的媒體，地位與出版業及電視媒體不相上下，甚至有過之，因為 Web 提供了互動的應用，舉凡線上購物，電子商務，網路銀行，以及電子論壇等等。

HTML 受到如此熱烈歡迎和需求，幾年來也擴充不少，產生許多新的標籤（Tag），從 HTML 1.0 的數十個標籤，到目前的 HTML 4.0 的近百個標籤，這其中還不包括各家瀏覽器自訂的標籤在內。再加上其他林林總總的支援，如 Javascript，CGI， Java，Flash，MP3 等等，每一種支援都會產生屬於自身領域的標籤，無形中造成了 HTML 在應用上的負擔，這點可以從 Microsoft 的 Internet Explorer 從早期的 1 MB 左右，增加到目前的十幾 MB，即可了解箇中原因。當然，HTML 文件格式不夠嚴謹且過於彈性，也是造成類似 Microsoft 的 Internet Explorer 程式膨脹的原因。

此外，網路通訊平台多樣化，工作平台已不僅限於主流的桌上型個人電腦，資訊觸角延伸到更多元、更多樣的平台，進入了所謂後 PC 時代。在後 PC 時代，不只 PC，還包括形形色色的資訊家電 IA，如 PDA，WAP/3G 手機也開始在日常生活中扮演著資料通訊的角色。這些資訊家電或行動設備，為了便於攜帶，通常具備體積小、重量輕、耗電低，因此都使用簡單的處理器及少量的記憶體，在功能上並無法如 PC 般，能處理像 HTML 這樣複雜的語言。以往 HTML 所強調的美輪美奐的畫面及功能強大的多媒體支援，在這些僅擁有不到五英吋，甚至只有兩三英吋的液晶顯示幕上，顯得多此一舉。因為，這些資訊家電的使用者往往對資料的需求更甚於美輪美奐的畫面以及多媒體的展現。

網際網路的興起，電子商務也跟著發達，以往企業與企業間的資料交換，如今可以透過網際網路解決，變得更加方便。然而企業與企業間的工作平台及資料格式，未必相符。例如異質（heterogeneous）資料庫，即使兩個企業間使用同質（homogeneous）資料庫，資料格式也未必吻合，在交換過程中必須要有一種標準以取得正確的資料。這個標準必須能夠對資料具備自我描述（self-describing）的功能，方便雙方應用程式對資料的辨識及萃取。這一方面，一向注重資料顯示的HTML文件顯然無法提供相對的要求，因為HTML不具擴展性（extensibility），讓使用者自訂標籤，表達資料內容。當然，這些問題都可藉由SGML（Standard General Markup Language）[3]解決，只是SGML複雜的功能，造成使用上的不便，使得真正的應用並未像HTML那樣廣受歡迎，反而讓HTML成為SGML應用中最熱門的應用。

XML 的嚴格語法確保文件結構（nested）及內容的正確性（validation），標籤可擴充性提供了自我描述的能力（self-describing），解決前面 Html 文件結構不嚴謹及無法讓使用者自定標籤來描述資料的兩個問題，再加上其本身的文字檔內容，透過網際網路傳送之後，不同平台、不同應用程式也可加以處理。也因此，這幾年 XML 文件格式已成為一個相當重要而且廣泛應用的文件標準。

隨著大量的使用 XML 文件，XML 文件的儲存及搜尋已是重要研究課題。傳統以查詢字串（query string）為主的搜尋引擎[13]，透過查詢字串快速查詢相關文件的能力，發展出相當多成熟的文件索引技術[4]。不過，這些技術的目的僅止於對文件中字串的處理，對於像 XML 文件強調文件中字串與字串的巢狀（nested）結構關係，現行的搜尋引擎並未提供相關查詢機制，因此在查詢 XML 文件，即使相關查詢字串在 XML 文件中出現，該文件中相關字串間也未必具備相對應的結構關係，因而產生不正確的查詢結果。

本論文主要是針對上述傳統搜尋引擎在查詢功能及機制上，未能滿足XML文件的查詢需求，提出一個針對XML文件可以查詢文件結構的搜尋引擎，並對XML文件的索引技術做相關探討，利用BEL的索引技術來保存XML文件的結構關係，將索引資料值儲存於關聯式資料庫系統。再利用XPath[5]表示式轉換為SQL Command存取資料庫，重建（reconstruct）所得到的資料錄（records），可以獲得跟原先一致的XML文件內容。為了在查詢過程中避免不必要的繁複資料表的join動作，引入signature file的索引機制，做為過濾機制，過濾掉不必要的資料庫查詢。

以下本論文將在第二章介紹有關XML文件及搜尋引擎的相關背景，第三章討論XML文件的索引技術與查詢，第四章系統實作，最後一章結論。

二、背景

本章將針對 XML 文件結構、特性與相關技術資訊中與本論文有關的提出說明，並在第二節說明搜尋引擎的工作原理，其次說明 XML 文件的查詢語言 XPath 的規範。

2.1 XML

XML 與 HTML 都屬於文件標示語言（Markup Language），文件內容主要由一些標籤（tag）以及字元資料所構成，如圖 1。

```
<?xml version="1.0" ?>
<school>
        <student id="1">
                <name>
                        <firstname>John</firstname>
                        <lastname>Denver</lastname>
                </name>
                <sex>male</sex>
        </student>
        <student id="2">
                <name>
                        <firstname>Jimmy</firstname>
                        <lastname>Page</lastname>
                </name>
                <sex>male</sex>
        </student>
</school>
```

圖 1 一個簡單的 XML 文件 Students.xml

XML 與 HTML 兩者不同之處主要有以下兩點：

（一）XML 並沒有預先定義任何標籤名稱，這部分可由 XML 文件出版者透過 DTD（Document Type Definition）[26]或 Schema[27]自行定義。

（二）XML 具備嚴謹的語法，具有文件驗證的功能。

根據上述兩項規範，使得 XML 文件具備資料描述能力（self describing），而且可以驗證（validation）文件的正確性，這一點企業對企業在實施資料交換過程時相當重要，也是目前使用相當廣泛 HTML 文件無法提供或是功能欠缺的部分。也因此，才有 XML 出現的機會，用來彌補 HTML 之不足。

2.1.1 XML 的文件結構

圖 1 呈現一個簡單的 XML 文件的內容，當儲存於檔案系統時，這些內容主要以文字檔的格式存在，因此可以由任何文字編輯器進行開啟與儲存，進行編修的動作。文字檔這種檔案格式，大多數的作業系統都有支援，如 Unix 、Microsoft Windows 等，因此達到跨平台的目的。

我們也從圖 1 中發現一個 XML 文件包含幾個組成單元（component）：

（1）元素（Elements）：每一個元素名稱都出現兩次，分別稱為起始標籤（start tag）與結束標籤（end tag），起始標籤由＜符號開始，以＞符號結束。結束標籤由＜/符號開始，並以＞符號結束。元素可以包含其他的元素或者文字（character data）。一份文件中只能有一個根元素（root），如圖 1 中的 school 即是根元素。根元素包含文件中所有元素。

（2）屬性（Attributes）：用來描述元素的額外資訊。如圖 1 中＜student id=”1”＞。屬性名稱與屬性值由＝符號連接，屬性值前後一定要有引號或是雙引號，並包含於元素的起始標籤內，與元素名稱以空隔分開，所以在 XML 文件的命名規則中元素名稱或屬性名稱不可以出現空隔。

（3）文字（Texts）：一般的文字資料（character data）XML 並未特別定義，但 XML 支援 unicode，所以也可以使用中文資料。

除了以上三種本文中主要使用的節點以外，尚有其他節點介面，例如處理指令 PI（processing instruction），CDATA 等。

XML 嚴格要求起始標籤與結束標籤成雙存在，但允許空元素存在，例如：

＜abc＞＜/abc＞

XML 為了簡化起見，空元素允許以＜abc／＞來彈性表示。

必須注意的是，除了元素與屬性的名稱不能有空隔以外，XML 所有內容是大小寫有區別的（case-sensitive），這一點與 HTML 明顯不同，也與 SQL 的查詢有所差異。因此，如何利用 SQL 查詢並保有 XML 的一些特性，也是本論文所強調的功能。

此外，XML 文件節點間的階層關係，為了文件處理的方便性，也常被表示成樹狀結構，如圖 2。

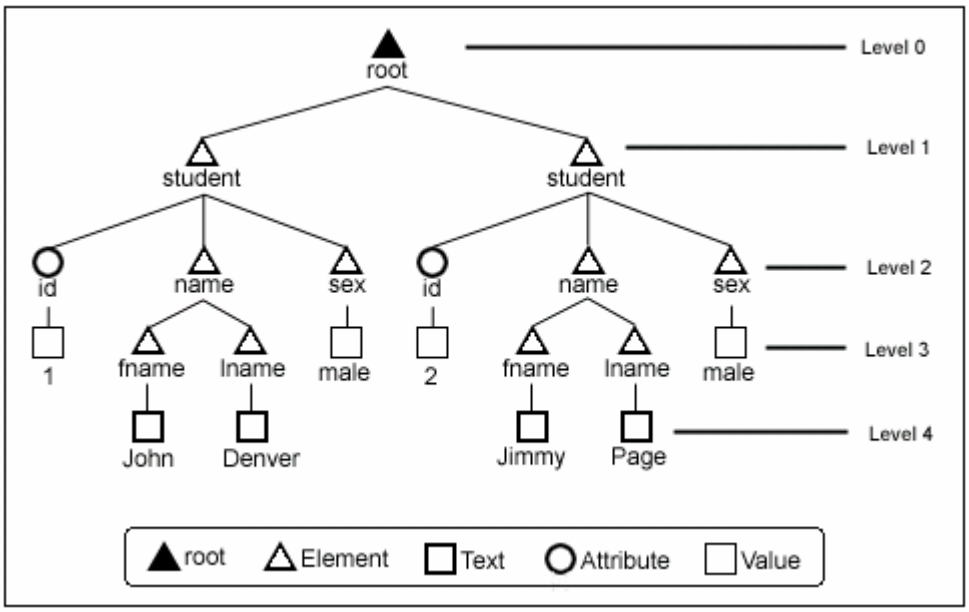

圖 2 文件 students.xml 的樹狀結構

2.1.2 XML 的處理

XML 文件，如同前面提到的特性，具備跨平台以及利用自訂標籤來達到資料描述的能力，如果單純以普通文字編輯器開啟 XML 文件，XML 文件本質上是一個內容包含許多特殊標籤的文字檔。因此，利用 XML 文件的特性來處理資料時，就必須依賴一些工具的協助，才能從這個單純的文字檔中取出相關的資訊，也才能發揮 XML 文件的功能。這類的工具一般稱為 XML 文件的解析器（Parser），主要有兩種介面，其中一種是由 W3C 所制定的介面（Interface）規格開發而成的文件物件模型（Document Object Model，DOM）[6]，另一種稱為簡單 XML 應用程式介面（Simple API for XML，SAX）[7]。SAX 和 DOM 兩者都提供程式設計師處理 XML 文件資訊。以下將就這兩個主要的文件處理介面提出說明。

2.1.2.1 文件物件模型（Document Object Model，DOM）

圖 3 表示 XML 文件、剖析器、DOM 物件樹和應用程式之間的關係。從圖中我們可以清楚的看到 XML 文件經由剖析器（Parser）剖析之後，形成一個 DOM 樹狀結構，如圖 2，此一樹狀結構將文件中的元素、屬性、資料...等以階層式節點方式加以儲存，應用程式則藉由存取樹中的節點而讀出資料或寫入資料。

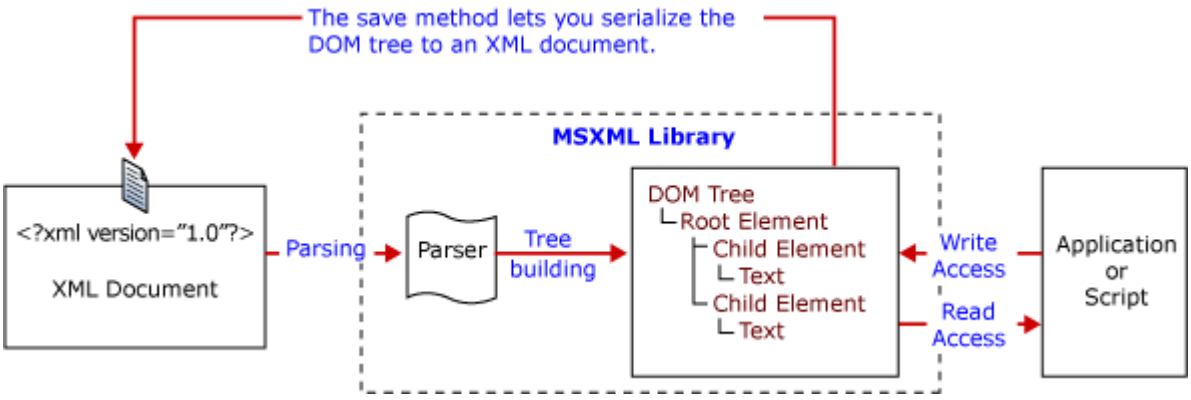

圖 3 DOM 介面運作模式

資料來源:http://msdn.Microsoft.com

使用 DOM 的優點如下：

- 可以任意擷取節點資訊
- 做 XSL 轉換比較容易

缺點如下：

- 需要額外記憶體，不利於大文件的解析
- 處理速度較慢，因為要預先在記憶體建構解析樹

2.1.2.2 簡單 XML 應用程式介面（Simple API for XML，SAX）

SAX 的全名是 Simple API for XML，不像 DOM 是 W3C 所制定出來的一個標準，而是由一個名為 XML-DEV 的非正式團體所發表。SAX 不僅不會在記憶體中特地根據 XML 文件的元素內容來建立樹狀結構，還可以為應用程式提供最有效率的資料儲存方式。

圖 4 中以事件為基礎的解析程式會將事件傳送給應用程式。這裡的事件和使用者介面所產生的事件（如視窗程式中按鈕的 onClick 事件）很類似。事件的功用是通知應用程式發生了某些狀況，應用程式則必須加以回應。在瀏覽器中，事件通常是為了回應使用者的動作而產生的，例如當使用者按下按鈕時，就會引發 ONCLICK 事件。

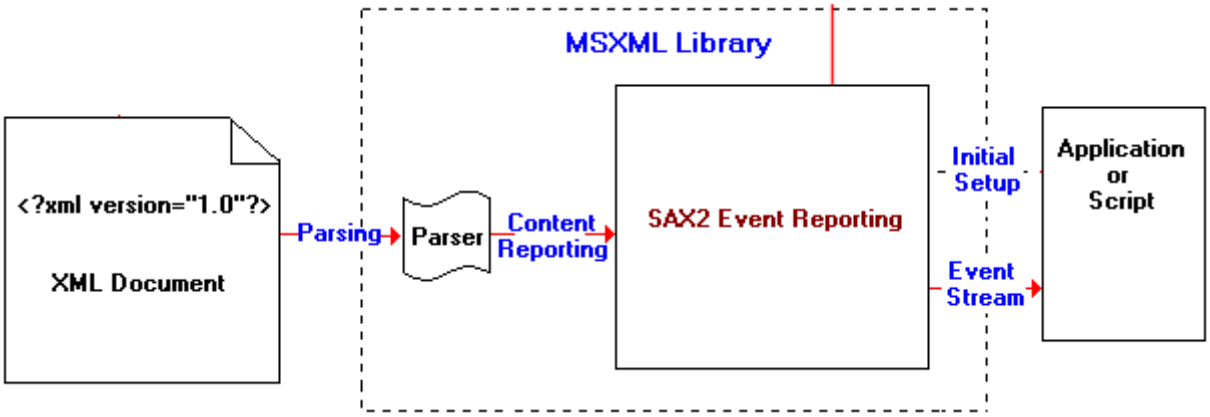

圖 4 SAX 介面運作模式

資料來源:http://msdn.Microsoft.com

但在處理 XML 解析程式時，事件和使用者介面無關，而是與從 XML 文件中讀取的元素有關。在此會出現的幾個主要事件包括：

- 文件的起始（startDocument()）
- 文件的結束（endDocument()）
- 元素的起始標記（startElement()）
- 元素的結尾標記（endElement()）
- 元素內容（characters()）
- 解析錯誤（warning()，error()及 fatalError()）

圖 5 所示為 SAX 解析程式的事件觸發順序。

當解析程式每遇到一個元素節點就會觸發一次 startElement()，所以圖 5 中遇到 school 節點時，觸發 startElement()事件。同樣地接下來的 student，name 以及 fname 都跟 school 一樣，觸發 startElement()事件,直到 John 這個 Text 節點，解析程式會觸發 characters()事件，然後接下來 fname 以及 lname 兩個結束標籤，解析程式會依序觸發兩次 endElement()事件。如此週而復始在整個文件中，不斷觸發一系列的 startElement()，endElement()以及 charactres()，直到文件結束最後會觸發唯一的事件 endDocument()為整個解析過程的結束。

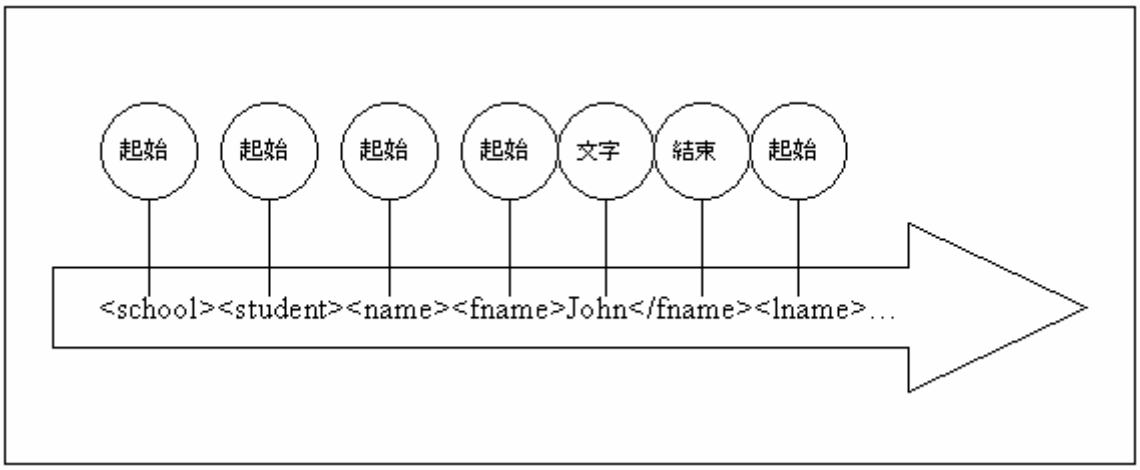

圖 5 SAX 解析程式的事件觸發順序

使用 SAX 的優點如下：

- 速度快
- 不需額外的記憶體，可以處理較大檔案

缺點如下：

- 無法隨意取得節點資訊
- 需要額外的物件維持解析過程中的節點資訊

基於以上優點本論文採用 SAX 解析程式，做為 XML 文件編碼過程的主要工具。

2.1.3 XML 的儲存

常見的 XML 文件儲存機制有兩種：第一種為檔案型態，第二種為儲存於資料庫中資料表的方式。儲存於資料庫中還可分為三種，第一種是儲存於關聯式資料庫，第二種是儲存在原生性 XML 資料庫（Native XML Database，NXD），第三種是儲存於所謂物件導向資料庫。以下分別就上述儲存機制與方法加以探討。

（1）以檔案系統儲存：XML 檔案（XML file）

XML 檔案雖然副檔名為.xml，實際上就是普通的文字檔內容，但經由瀏覽器的 XML 文件解析器（parser）解析之後，卻呈現階層式的結構，與文字檔所呈現的方式截然不同。雖然 XML 文件直接儲

存在檔案系統比起其他資料庫型態相對容易與方便，也絕對支援相關 XML 的應用技術，例如查詢，但也存在許多缺點。當資料量過於龐大，想要對文件進行搜尋，將大大影響搜尋效率，所以一般而言，對於 XML 文件的儲存，極少會以檔案方式為儲存的機制，即使有，大多數也都只是用於暫存。

（2） 以資料庫系統儲存

a. 關聯式資料庫系統（Relational Database Management System）

關聯式資料庫由於在技術成熟度及市場使用率兩個因素之下，成為目前儲存 XML 文件最廣泛的方法，如 Oracle、IBM DB2、Microsoft SQL、Sybase 等資料庫。許多論文[8,9,10]都紛紛針對 XML 文件如何儲存在關聯式資料庫，深入研究探討期可運用的方式與轉換技術。但對於 XML 文件的階層性樹狀資料結構，要將 XML 文件的資料內容儲存於關聯式資料庫的資料表中，必定有一定的困難與不相容。

XML 階層性資料結構與傳統的關聯式資料表儲存方法是有所差異，若要將 XML 文件資料儲存於資料表中，則必須經過一套轉換機制與標準，才能將 XML 資料內容或含有文件定義 DTD 的 XML 文件，儲存於關聯式資料庫的資料表中。要利用關聯式資料表來儲存階層性的資料內容，就必須要將這些階層性的資料分解（shredding）成獨立狀態（atomic），再透過這些轉換機制與標準所產生的資訊，來辨別儲存在資料表中的每筆資料彼此間的關係，而這套轉換機制與標準就是一套能夠保留 XML 文件內所有內容（如 element，attribute 以及 text）的階層關係的索引機制，藉由索引保持所儲存跟再重建回來的 XML 文件的一致性。這就是 XML 文件儲存在關聯式資料庫的主要運作模式。

此外，將 XML 文件儲存在關聯式資料庫也有另一項優點。就技術而論，資料庫系統算是相當成熟的，使用率也最廣泛，目前大部分的企業仍然以關聯式資料庫系統為主，將 XML 文件跟著現有資料一併儲存在現有資料庫中，不管在時間、成本以及人力等方面，都是不錯的考量，也是值得研究的方向。這也是本論文選擇以關聯式資料庫為後端儲存對象的原因。

b. XML 原生性資料庫（Native XML Database）

原生性資料庫[11]是近年來提出的一種針對 XML 文件的儲存機制，目前市面上有少數資料庫廠商研發相關的原生性資料庫系統，如 Oracle 所推出的 XMLDB、Software AG 的 Tamino。

原生性資料庫與關聯式資料庫，在儲存 XML 文件上有著極大的差異，原生性資料庫將 XML 文件以一種叫做 XML 資料型態，將整份 XML 文件是為一個儲存單位或者稱一個欄位，因此，不必如關聯式資料庫一樣，在儲存 XML 文件文件之前，需先將 XML 文件分解（shredding）成各個資料單元，存取時也不用經過特殊的處理，的確是更適合愈來儲存 XML 文件資料，在資料處理上也更容易與 XML 相關技術結合，不至於產生與原始資料不符合或不相容的情形。

在 XML 文件儲存方法的研究領域中，由於原生性資料庫的崛起到目前時間不長，並沒有太多的成果提出。但仍有部分針對這類資料庫的檢索查詢與系統建構的研究，利用原生性 XML 資料庫充當後端儲存架構，相信這方面還有相當的研究空間。

c. 以物件導向資料庫儲存 XML 文件，目前也有許多研究進行，技術上由於物件導向資料庫對於大型資料庫的複雜查詢，效能還有許多努力空間，再加上使用率較低，因此以此種方式儲存 XML 文件比起關聯式資料庫明顯少了很多。

2.2 搜尋引擎

搜尋引擎（Search Engine）[13]一詞，自從網際網路（Internet）發達之後，往往將透過 Web 存取遠端或後端資料庫資料的模式，稱之為搜尋引擎，例如 Google[1]。在這裡我們要對搜尋系統進一步了解並下一定義，才有助於本論文的發展。

2.2.1 搜尋引擎的概念

搜尋引擎可以是一支程式，例如 Unix 系統中的 grep 指令，也可以是一個所謂的搜尋系統，例如時下最為流行的 Google，後者就不只是一支程式可以完成整個搜尋作業。基本上兩者雖然在作業方式有所差異，但架構上還是有相同之處。整理如下：

（1）一個目標資料（文件）的集合，例如檔案系統、資料庫等。

（2）透過索引結構提供快速查詢指定文件。

（3）一種介面（如 Google 的 Web 介面）或是一個指令（如 Unix 系統的 grep 指令），提供使用者輸入查詢字串或是查詢命令的動作。

以 Unix 系統 grep 為例，它的目標資料就是所有文件，這些文件受到 Unix 檔案系統的管理，而它的介面就是透過 grep 指令，進行查詢字串與目標物件的樣式比對（pattern matching）。

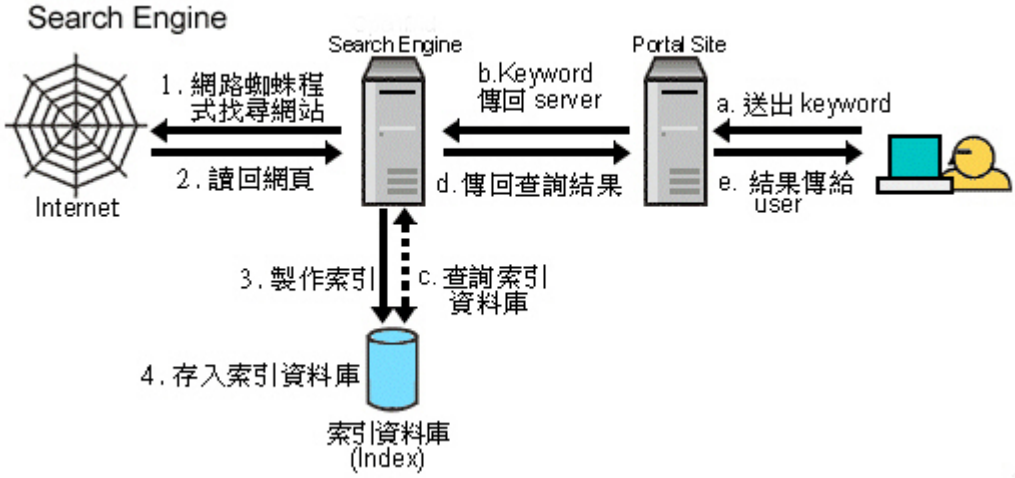

圖 6 搜尋引擎架構

資料來源：Openfind.com

Web 為架構的搜尋引擎，於文件的集合、索引的建立及搜尋的介面前者有所不同，如圖 6。網際網路的搜尋引擎除了處理 user 所下的查詢字串的 CGI 程式以外，還包含所謂的網路蜘蛛程式（spider）[13]或稱為網路蠕蟲程式（crawler）[13]。這類程式通常負責依照所指定的起始（initial）網站，利用 HTTP[24]通訊協定將網頁取回，除了將網頁中所要的資訊加以建立索引，將資料儲存在索引資料庫，還要根據網頁中的超連結持續地取回網頁，如此週而復始，於是就構成了一個所謂的搜尋引擎系統。

一個好的搜尋引擎，除了搜尋速度要快以外，還需搜尋資料的準確性要高。而搜尋速度要快關鍵就在於根據資料型態建立索引，靠著有效的索引技術，免除不必要的資料比對，來加速（speedup）資料的搜尋。在資料正確性方面，受制於 HTML 文件的先天不足，文件不具結構性描述，儘管在資訊檢索技術上應用了相當多的技術，例如布林運算，仍然還是無法做到精確地步，最後還是要靠人的篩選才能找到適當的資訊，尤其是資料量快速成長的今天。這點可由 Google 的搜尋結果印證得知，雖然該搜尋系統在準確度上已相當的成熟，做為一個 HTML 文件搜尋引擎，Google 還是難免搜尋一推累贅多餘的資訊需要由使用者來一一篩選。

2.2.2 XML 的搜尋引擎

XML 文件的的搜尋引擎，主要的架構也跟 2.2.1 節一樣，也是提供一個介面對 XML 文件的集合（Collection）搜尋符合的文件，本論文採用 Web 介面供使用者輸入查詢字串。另外，XML 文件必須經過索引建立的過程來加速（speedup）查詢效率並回應符合查詢要求的正確資料。我們將在第三章討論目前有關 XML 文件對應於關聯式資料庫（RDBMS）的索引的技術，最後利用目前儲存技術最成熟的關聯式資料庫，儲存相關的索引資料，提供查詢 XML 文件。

由於具備階層性的 XML 文件在分解（shredding）[16]之後，儲存到關聯式資料庫的資料表，相對於傳統 HTML 引擎，XML 文件的搜尋引擎就沒那麼順利，XML 文件的搜尋引擎必須利用索引建立時所獲得的位置資訊（position information）或者路徑資訊（path information），透過 SQL 操作，執行多重的 join 或者 self join 才可取得與原來 XML 文件相符的內容或是片段資訊。因此，取得資料過程中必需付出的關聯式資料表間的 join 運算相當繁複，本論文提出一個利用簽章索引檔（signature file）的過濾（filter）機制，來降低不必要的資料庫存取動作來加速查詢速度，並提供 XML 文件內容的文字資料大小寫有別的過濾機制，而不受 SQL 查詢關聯式資料庫時忽略大小的的影響。

2.3 XPath

XPath 是 XML 文件的查詢語言。

基本上，XPath 將一份 XML 文件視為一棵由節點構成的文件樹，如圖 2，這些節點主要是由 XML 文件中的元素（element）、屬性（attribute）及字元資料（text）所構成，XPath 定義游走（traverse）在這些節點之間的規則及語法，透過符合語法（syntax）規定的表示式（XPath Expression），可以找到想要查詢的資訊或者了解 XML 文件中是否存在該項資料。XPath 是 W3C 制定 XSLT 標準的主要要素，並且 XQuery 和 XPointer 也都建立在 XPath 表示式上。因此了解 XPath 是進一步利用 XML 的基礎。

XPath 表示式類似 Unix 系統中的目錄表達方式，可由好幾個步驟（step）串接而成，例如：/step/step/step。以圖 2 為例，想要知道所有 student 的 name 節點的內容可以表示成絕對路徑式（absolute location path）/school/student/name，也可以表示成相對路徑式（relative location path）student/name。

XPath 的表示式中每一個步驟（step）主要由三部份構成，分別是軸（axis）、節點測試以及：

（1）軸（axis）：定義所選取節點集合與目前節點在樹狀結構中的關係。例如：child::。通常 child 可以省略，像 name 即是 child::name。

（2）節點測試：指出在軸之下的哪一節點。

（3）零個或零個以上的 predicate：篩選經由節點測試後哪些是符合的節點。

經由上述 XPath 中對步驟的定義後，一個基本的路徑語法如下：

axisname::nodetest[predicate]

以圖 2 為例，要找子節點中 student 的 fname 是 Jimmy 的節點的表示式如下：

child::student[fname="Jimmy"]

執行上述 XPath 表示式之後，結果如下的節點被選出，如下圖 7：

```
<student id="3707">
    <name>
        <fname>Jimmy</fname>
        <lname>Page</lname>
    </name>
    <sex>female</sex>
</student>
```

圖 7 符合 XPath 查詢的子節點集合

XPath 表示式執行結果一個很重要的特性：任何 XPath 表示式的結果一定是一組符合路徑表示式的節點集合，包含所選取的節點的子節點所有集合。

此外，這種以文件處理導向（Document Processing oriented）的搜尋方式，暴露出 XML 經由 XPath 表示式查詢文件中相關節點的缺點，那就是為了找到 fname 是 Jimmy 的節點，XPath 必須一一比對每一個節點，才能找到符合的節點，即使沒有一個節點符合，XPath 也會對整個文件從頭比到尾。這樣的方式用在單一而且檔案不大的 XML 文件搜尋時還不成問題，如果要利用這樣的方式，對一個大量 XML 文件的集合做搜尋可能在效率上會大打折扣。因此，如何利用索引機制來剔除不必要的節點比對，成為 XML 文件儲存與查詢的討論重點。

三、XML 文件索引與查詢

3.1 XML 文件的索引方法（Indexing）探討

在 2.3 節所提及 XPath 的缺陷主要是因為 XML 查詢處理時，必須經由標簽路徑尋找節點這一限制造成的，想要得到符合查詢路徑的節點，必須順序地依次拜訪標簽路徑對應的節點才行。那麼，如果能夠改變 XML 資料管理的方式，就可避免必須通過路徑找節點的限制，進而在某種輔助資訊的幫助下，直接針對節點集合的管理模式得到符合的節點集合，這就是節點記錄類 XML 索引的基本思想。

將 XML 看成樹狀結構，針對實際的處理要求，將 XML 資料單元（這裡所謂的資料單元是指標簽名稱（element）、屬性名稱（attribute name）、屬性的值（attribute value）、文字（text）等，作爲記錄保存;同時在資料單元的記錄中保存有助於確定節點間結構關係的位置資訊，表示為(資料單元標識,資料單元在 XML 中的位置資訊)。

位置資訊通常分爲兩類,一類是節點經由某種搜尋方法（如樹狀結構中的 preorder 及 postorder）所得節點序列中的序號，另一類是節點在 XML 文檔中的路徑資訊。

實際索引資料的儲存可以採用不同的型態，如 2.3 節所述，可以儲存爲原生性(native storage)、關聯式資料型態或者物件導向資料型態。其中最常見的就是將索引記錄儲存爲關聯式的資料型態。

以下主要針對這種方式進行說明。

3.1.1 節點紀錄類索引

節點記錄類索引本質上即是將 XML 資料分解爲資料單元的記錄集合，同時在記錄中保存該單元在 XML 資料中的位置資訊。主要有兩種獲取位置資訊的方法，一種是節點序號方法(node numbering method，有時也稱爲節點標簽方法，node labeling method)，另一種是節點路徑方法(node path method)。對於 XML 資料處理研究中佔了相當大的部分。根據路徑資訊表現形式的不同，節點記錄類索引分爲 3 大類：分為節點序號的索引、節點路徑資訊的索引和二者相結合的混合索引。

3.1.1.1 利用節點序號的索引

這類索引中的位置資訊是節點根據某種樹狀搜尋（traverse）序列中的序號。一個 XML 文檔經由樹狀搜尋方法,搜尋的結果是一個由節點組成的序列；相對地，節點的標簽在序列中就有一個序號，該序號呈現出節點間的次序關係，也能夠反映節點間的結構關係，進而，就可以該序號資訊獲得節點間的結構關係。

節點序號類 XML 索引的基本想法是：對 XML 文檔採取某種節點遊走方式,得到由元素組成的序列，節點的標簽在序列中就具有唯一的次序，將序列與某指標集（最常用的就是自然數）建立一一映射的關係，對應序列中某個標簽就有唯一的序號；對任意兩個具有節點序號資訊的節點，可以構建出某種運算，該運算的結果可以顯現節點間的結構關係，即{(獨立單元屬性,位置資訊)}→{結構關係}映射。

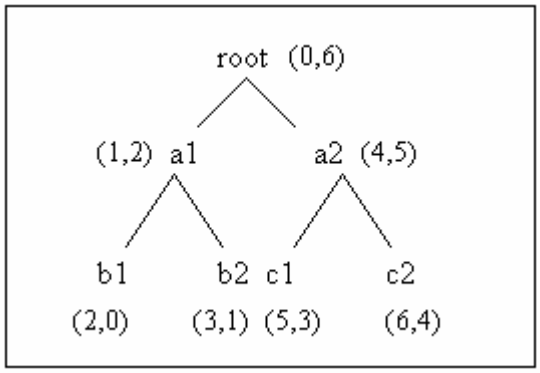

圖 8 利用前序後序所取得節點序號

位置資訊中最爲常見的是所謂的區間編碼[25]（interval encoding）爲序號對的形式，如圖 8。其基本做法是：將節點在先序 traverse 和後序 traverse 中的序號對組成的區間(pre，post)作爲該節點的位置資訊，而其基本索引形式為（s，pre，post），其中的 s 為 XML 文件樹中的節點標籤，利用這樣的索引形式將滿足以下的關係：

(1) 如果 s1 和 s2 分別是 XML 文件中具有祖先－子孫關係的節點，則相對應的（s1，pre1，post1）及（s2，pre2，post2）必然滿足 pre1＜pre2，post2＜post1。

(2) 如果 s1 是 s2 的兄弟節點，那麼相應的(s1,pre1,post1)和(s2,pre2,post2)必然滿足 post1＜pre2。

根據上面的關係，可以由 pre1<pre2,post2<post1 不等式確定對應的節點 s1 和 s2 是祖先-子孫的關係；這個不等式反方向未必成立。s1 和 s2 之間是否屬於父親-孩子關係卻無法確定，而且，僅由 post1＜pre1 不能確定 s1 和 s2 是否就是兄弟關係。爲解決無法確定父親-孩子關係的問題，引入節點在 XML 樹狀結構中”層”的數值資訊，叫做 level。相應的四元索引記錄形式爲(*s*,pre,post,level)。

另外一種常用且與區間編碼有類似性質的編碼稱為 BEL（Begin-End-Level）。其基本做法是：將標記的起始、終止標簽在 XML 字元流（stream）中出現的序號組成數對，作爲該標記的位置資訊。其中的 B 表示標記的起始標簽出現的次序，E 表示同一標記的終止標簽出現的次序，而 L 表示標記距離根標記的層數。由於這樣的做法很適合利用 2.1.2.2 的 SAX 來處理，因此 BEL 也是本論文主要的 XML 索引建立方法。如圖 9，R1 是 XML 文件的根節點，它的 BEL 編碼為（0，13，0），其中第一碼 0 為 R1 的起始標籤位置編號，第二碼 13 為 R1 的終止標籤位置編碼，至於第三碼 0 為 R1 的在 XML 文件中的第 0 層，因為是根節點。

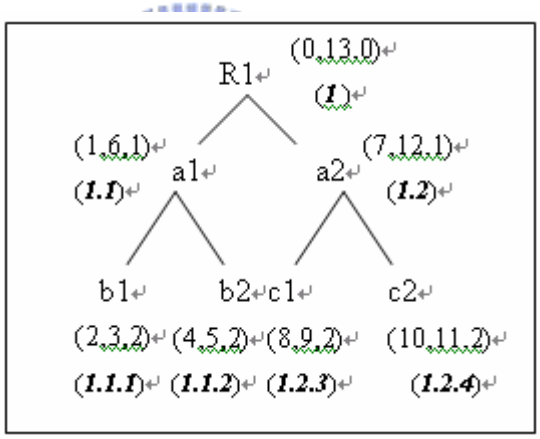

圖 9 BEL 區間編碼及 Dewey 局部編碼

例如 students.xml 經由 BEL 編碼後得到如下：

<school><student id="1"><name><fname>John.....</student></school>

Position：	0	1	2	3	4	13	27
Level：	0	1	2	3	4	1	0

並利用關聯式資料庫儲存，其中元素資料表的 schema 為 Element=<ElementName, DocId, begin,end, level>，

Element=< "school" , 154 , 0 , 27 , 0>
Element=< "student" , 154 , 1 , 13 , 1>

Element=< "name" , 154 , 2 , 9 , 2>
Element=< "fname" , 154 , 3 , 5 , 3>
Element=< "lname" , 154 , 6 , 8 , 3>
Element=< "sex" , 154 , 10 , 12 , 2>
Element=< "student" , 154 , 14 , 26 , 1>
Element=< "name" , 154 , 15 , 22 , 2>
Element=< "fname" , 154 , 16 , 18 , 3>
Element=< "lname" , 154 , 19 , 21 , 3>
Element=< "sex" , 154 , 23 , 25 , 2>

文字節點資料表的 schema 為 Text=<Textm, DocId,Wordno,level>，

Text=< "John" , 154 , 4 , 4>
Text=< "Denver" , 154 , 7 , 4>
Text=< "male" , 154 , 11 , 3>
Text=< "Jimmy" , 154 , 17 , 4>
Text=< "Page" , 154 , 20 , 4>
Text=< "male" , 154 , 24 , 3>

屬性節點資料表的 schema 為
Attribute=<AttributeName,AttributeValue, DocId, begin >，
其中 begin 是該屬性所屬的元素的開始標籤序號：

Attribute=< "id","1" , 154 , 1 >
Attribute=< "id" , "2", 154 , 14 >

經過解析以及 BEL 編碼所得上述資料，分別存入關連式資料庫中的元素、文字及屬性的資料表中。由於 BEL 方法能確保 XML 文件的節點位置訊息，編碼過程與 SAX 解析 XML 文件方式相當吻合，加上 2.1.2.2 節中所列 SAX 在解析過程中有記憶體使用上的優勢，速度快等優點。因此本論文採取 BEL 做為 XML 文件的索引機制。

在節點位置資訊產生過程中,考慮該節點在兄弟節點中的次序,並保留父節點的序號資訊,稱爲局部編碼方式,相應的位置資訊形式爲“父節點序號資訊.本節點在兄弟節點集合中的序號”。

另一種 Dewey[28]編碼，如圖 9 粗體字部分，其方法爲：對每一個子節點,根據該節點在兄弟節點中的位置，給於一個序號。例如,某節點在兄弟節點中處於第 4 個位置,那麼,它的編碼就應該是“L(u).4”,其中,L(u)爲父節點的編碼。Dewey 編碼規定,根節點的編碼爲 0。以圖 9 中 C2 為例，其父節點 B4 的節點的節點紀錄為”1.2”，所以 C2 的節點序號資訊為”1.2.1”，因為 C2 是 B4 唯一的子

節點。這種方式的好處，就是可以輕易取得從根目錄到該節點的所有節點資訊。

3.1.1.2 利用節點路徑訊息的索引

節點的路徑資訊同樣蘊含節點在 XML 資料中結構的資訊：如果給定兩節點的路徑資訊，同時預知兩節點存在結構關係的情況下，就必然可以獲知它們之間的結構關係。即節點 A 的標簽路徑包含節點 B 的標簽路徑，那麼，在 XML 樹中 A 和 B 之間一定具有祖先-子孫的結構關係，且 B 是 A 的祖先。所以，基於路徑資訊來獲取節點的結構關係就成爲另一組 XML 資料處理技術的思路來源,並演化出以節點路徑為基礎的 XML 索引。這類索引的核心技術是字串的模式匹配。因而，這類索引記錄資料的管理方法，有許多是來自於資訊獲取（information retrieval）領域的。例如，Trie，Patricia trie 以及 Suffix tree，甚至 Suffix array 等結構。索引記錄的基本模式爲（資料單元標識,路徑資訊），如圖。路徑索引並無法保留節點位置資訊，須配合節點序號類索引而形成混合式索引。

No	Element	path
0	school	school
1	student	school/student
2	sex	school/student/sex
3	name	school/student/name
4	fname	school/student/name/fname
5	lname	school/student/name/lname

圖 10 節點路徑訊息

3.1.1.3 混合式索引

混合式索引同時保留節點的位置資訊（position information）以及節點的路徑資訊做為索引紀錄，這類的索引採用多個模式的組合，如（資料單元，路徑 ID，序號位置訊息）和（路徑 ID，標籤路徑）的組合，以 students.xml 為例：表 1 主要是利用 BEL 編碼來保存節點的位置資訊，表 2 則是每個節點所對應的路徑資訊。

混合式的優點是，除了具有路徑資訊的簡潔，並能獲取想對的節點位置資訊。

Term	DocID	begin	end	Level	pathID
school	1	0	27	0	0
student	1	1	13	1	1
name	1	2	9	2	3
fname	1	3	5	3	4
lname	1	6	8	3	5
sex	1	10	12	2	2

表 1 students.xml 的元素混合編碼表

Path expression	PathID
School	0
School/student	1
School/student/sex	2
School/student/name	3
School/student/name/fname	4
School/student/name/lname	5

表 2 students.xml 的路徑編碼表

3.1.2 結構摘要類索引

結構摘要索引，主要是以 XML 結構樹中的節點路徑資訊為基礎，採取某種簡化方式，使得簡化後的結構樹只保留不同的路徑，而不會出現相同路徑的節點。如圖 11。

結構摘要索引方式的實作有 Standford 大學的 DataGuide[21]，雖然可以滿足簡單路徑的查詢要求，但是在將 XML 結構樹簡化後，無法避免的將會失去節點間的相關位置資訊，因此後來的 F&B[23]索引就結合了 BEL 編碼機制，以加快查詢。

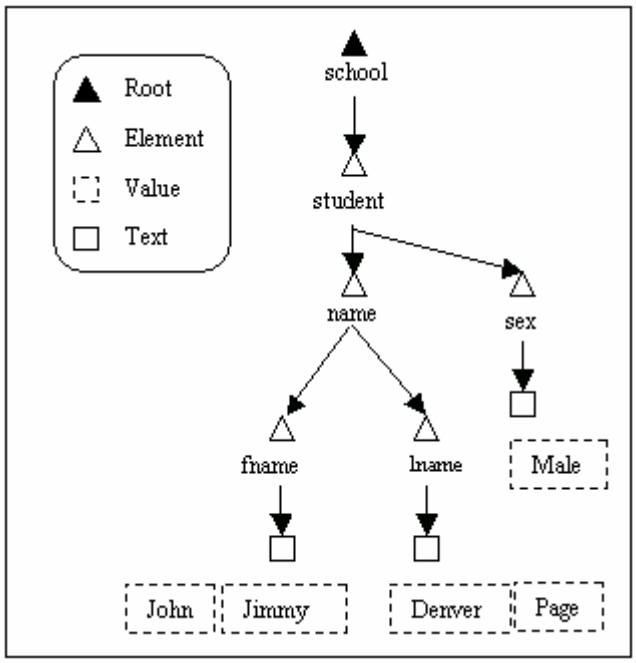

圖 11 student.xml 的結構摘要

3.2 簽章索引檔（Signature file index）

簽章索引檔（Signature file index）[4][14]是一種相當特殊的索引方法，尤其利用在無結構（non-structured）的資料的索引建立，例如應用在多媒體檔案[15]。相較於反向索引方法（inverted index）[4]，其索引檔案所佔的空間小很多，在比對上也由於結構上（位元運算）的關係，有其速度優勢。以下將就簽章索引檔的一些概念、方法並且就與簽章索引方法相關的雜湊函數做一說明。

3.2.1 簽章索引檔概念

簽章 Signature 利用一串 bit pattern 或者稱為 bit string。Bit string 如圖 12，是利用特殊設計的 hash function 對字串產生一個 hash value，再根據這 hash value 對固定長度（固定位元數）的 bit string 中某幾個位元（bit）設為 1（true），其他的位元設為 0（false），而被設為 1 的位元在每個 bit string 中的個數也是固定的。例如：student 這字串經過 hash function 的運算之後,取得一個整數值 3，如果 bit string 的長度為 8 的話，而每一個 bit string 只能出現一個位元是"1"的話，這個代表 student 的 signature 就是 00001000，這就是 signature 的產生過程。

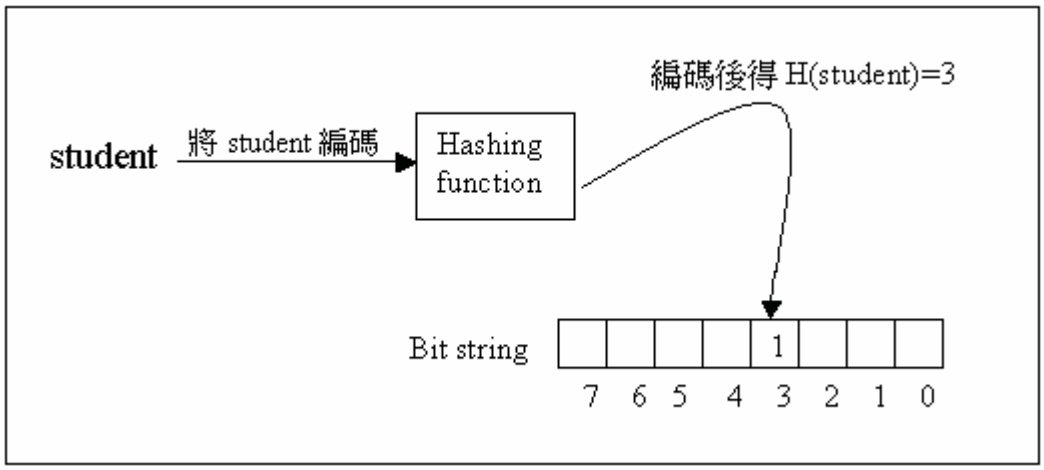

圖 12 個別節點利用特殊 Hashing function 編碼轉成 bitstring

而 signature file 就是儲存這些 pattern 的 binary file，只不過存的不是字串或文件本身，而是一連串由上述方法產生的簽章（signature）。

以一個文字檔為例，來說明 signature file。假設該文字檔包含四個字串，分別是 school、student、name 和 fname，這四個字串先透過上述 hash function 的計算取得三個 hash value，在此假設分別是 3，2，6 跟 7。經由這四個整數值轉換成 bit string 後，分別為 00001000,00000100， 01000000 和 10000000。接著將上述四個 bit string 做某種位元的運算之後，得到該文件的 bit string，這個就是代表該文字檔的 signature，其實跟前面單一字串產生的 signature 沒兩樣，唯一的差別在於 signature file 中含"1"位元不是固定數目，甚至連 bit string 的長度也會不長短不一。這點將再後一節討論。

有了 signature file 的索引檔之後，如果要找哪些檔案中含有 student 跟 name 兩個字串的話，只要將 student 跟 name 兩個字串分別先經過上述的 hash function 的計算，取得它們的 signature，接著將兩個 signature 做位元的運算，取得一個 query bitstring，再到 signature file 中逐一的跟 signature

block 做位元的 and 運算後，結果跟 query bit string 相等的話，那麼這個 signature block 所代表的文件就是符合查詢條件的文件。

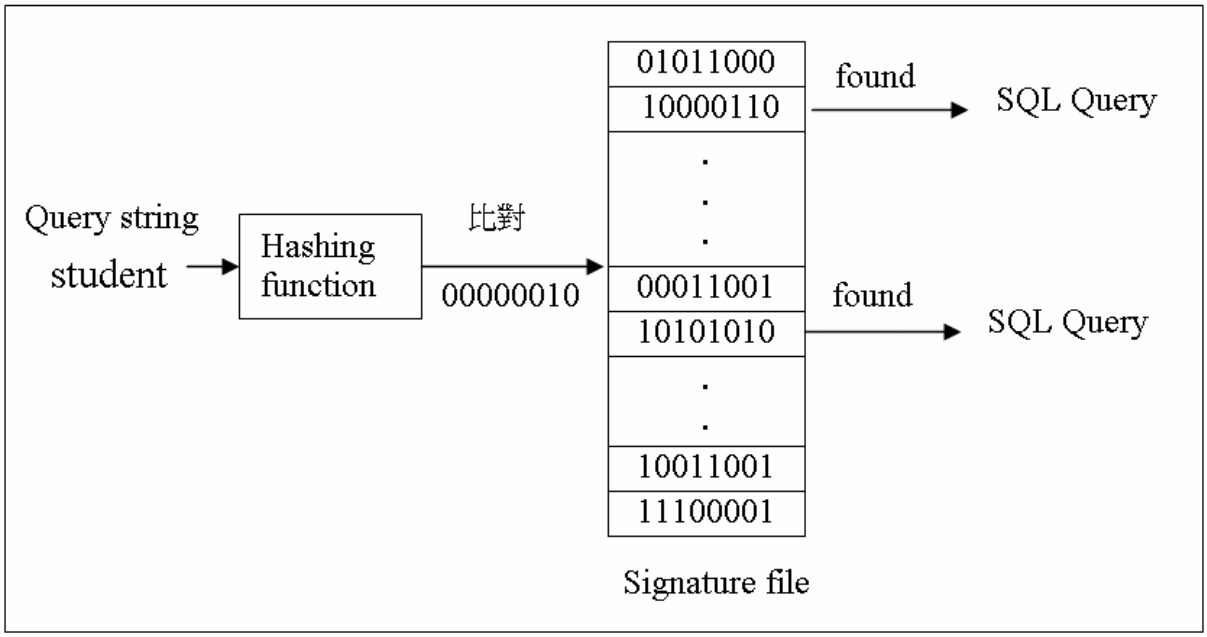

圖 13 Filter 的概念應用

從以上可以發現 signature file 利用連續儲存的位元比對降低字元的查詢時的 I/O 成本（I/O cost），快速判斷查詢字串或是字串組合存在與否，這點可用來輔助將 XML 文件儲存在關聯式資料庫查詢時，充當 filter 功能，如圖 13。首先，當所有查詢路徑的節點字串所產生的 bitstring 經過位元運算之後，所得的 signature 沒有在 signature file 中時，就不必再進行繁瑣的資料庫操作。另一方面，可以用來改進資料庫查詢時大小寫不分的缺點，因為當你要查 book 跟 Book 時，透過 SQL command 的查詢，它們都會被選取的，而 XML 文件本身所有名稱及文字節點是有大小寫之分的。

3.2.2 簽章索引檔種類

本節主要說明文件的 block signature 產生方式，主要有 Word signature 及 Superimposed coding 這兩種：

（1）Concatenation Signature（WS）方法

如圖 14 所示，假設該文件僅包含四個字串，此法主要是將整個文件中的每個字串，經由特殊的雜湊函數取得固定長度的 word signature ，這裡的 bit pattern 的長度為 f=4，每個 bit pattern 只能出現一

個”1”。再將這些位元樣式（word signatures）連接（concatenate）起來而形成文件的簽章（signature）。這個方法可以保留字串在文件中的位置資訊（positioning information），不過會造成每一筆文件的簽章長度不一樣，增加處理的複雜度。

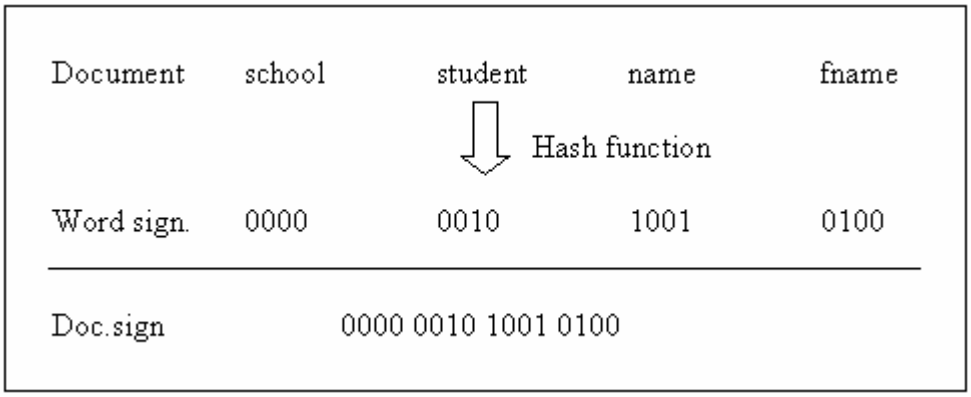

圖 14 Word signature 方法

（2）Superimposed Coding（SC）方法

第二種簽章方法如圖 15 所示，與 word signature 同樣的文件內容，一樣透過特殊的雜湊函數分別取得文件中每個字串的雜湊值，再利用這雜湊值獲得每個字串固定長度的 bit pattern，在此例 bit pattern 的長度 f=20，而每個 bit patten 只能出現一個”1”。所不同的是，在建構文件的簽章時，採用不同的方式取得文件的簽章，也就是將文件分成若干個 block，每個 block 包含固定數量的字串，再將 block 中每個字串取得簽章，每個字串的簽章執行位元 or（bitwise）運算，即得到所謂的 block signature。接著連接（concatenate）所有 block signature 就是文件的簽章。在這 block signature 中，只要第 13 位元被設為”1”（on），即表示 block signature 包含了 student 字串。

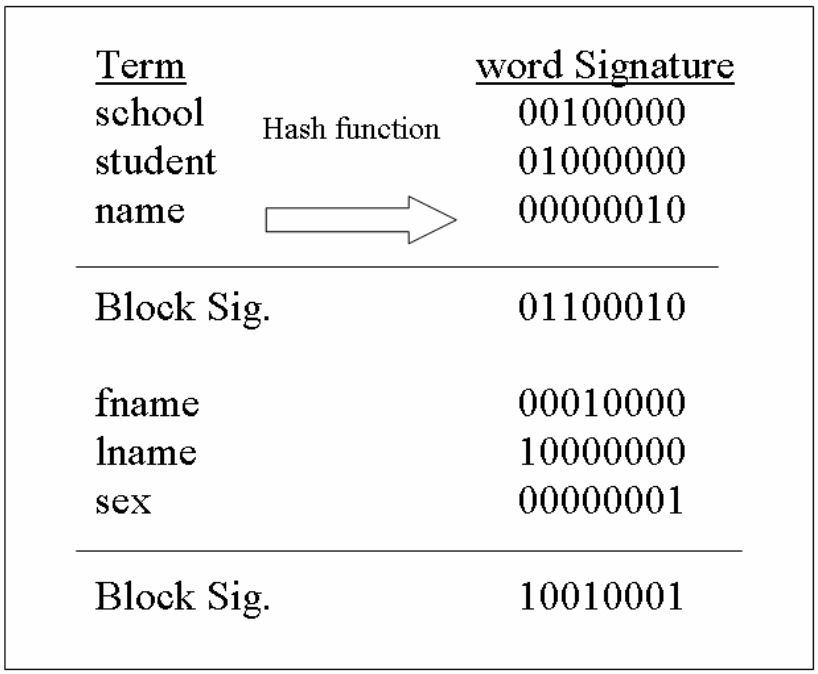

圖 15 Superimposed Coding 方法

嚴格來講，Word signature 是 Superimposed Coding 的一個特例，當 Superimposed Coding 方法中 block 的字數（words）等於 1 時，Superimposed Coding 方法就跟 Word signature 方法一樣。Superimposed Coding 利用邏輯區塊主要是節省比對次數，進而縮短搜尋時間。

3.2.3 雜湊函數的編碼

在 3.2.2 中所提到的兩個簽章索引方法，其中雜湊函數（hash function）扮演著相當重要的角色，本節將就 hash function 以及本論文所採用的 hash function 如何對 XML 文件編碼做一說明。

雜湊函數運用很廣泛，像 ORACLE 就利用雜湊函數在它的資料庫軟體上。雜湊技術主要是利用特殊設計的函數來計算關鍵值（key），直接存取該關鍵值所代表的相關資料，這裡所謂的「直接」意味著不需透過繁複的鍵值比對而能直接取得資料的意思。如此即能達到直接存取，降低搜尋時間，如圖 16 的雜湊表的應用。

雖然雜湊函數能提供直接存取資料的優點，但不能避免的資料過載（overflow）的問題。為了解決 overflow 或是 bucket split 問題，也發展出 over chaining、open address 等新的解決技術。當然也有許多針對 hashing 技術的新發明，如 Dynamic hashing[18]、Extendible Hashing[19]及 Linear Hashing[20]等。

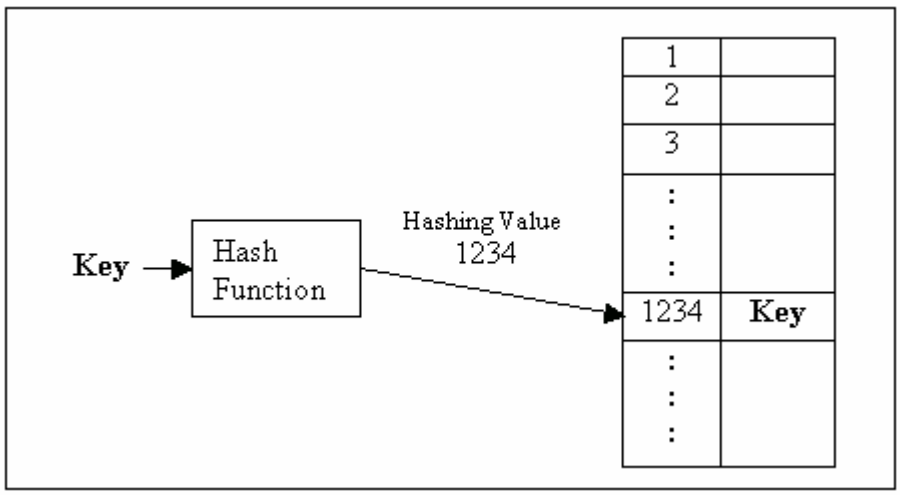

圖 16 雜湊函數的應用－雜湊表

雜湊函數應用方式也很多樣，如用於雜湊表的String Hashing，用於使用者驗證及授權的Cryptographic Hashing、用視覺辨識的Geometric Hashing及Bloom Filter[17]等。本論文主要是應用其中的Bloom Filter。

Bloom Filter 方法主要是利用多個雜湊函數（multiple hash functions）對每個鍵值（key）產生雜湊值，再利用這些雜湊值將某個固定長度的 bit string 中相對位置設為“1”，如圖 17。

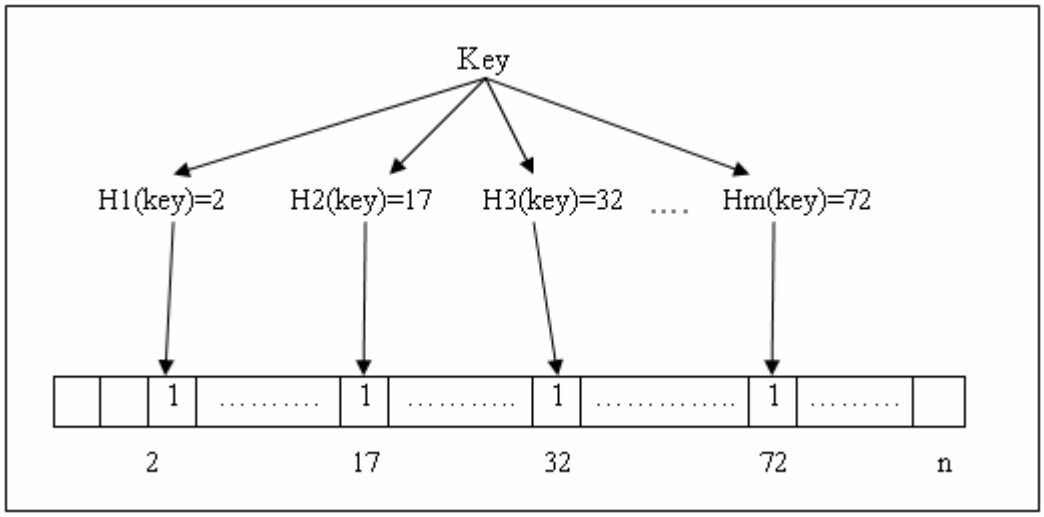

圖 17 利用 Bloom Filter 建立 bit string

利用簽章索引方法建立索引時，要考慮 false drop 問題，Bloom Filter 可以產生稀疏（sparse）的 bit string，如此可降低 false drop 的機率。

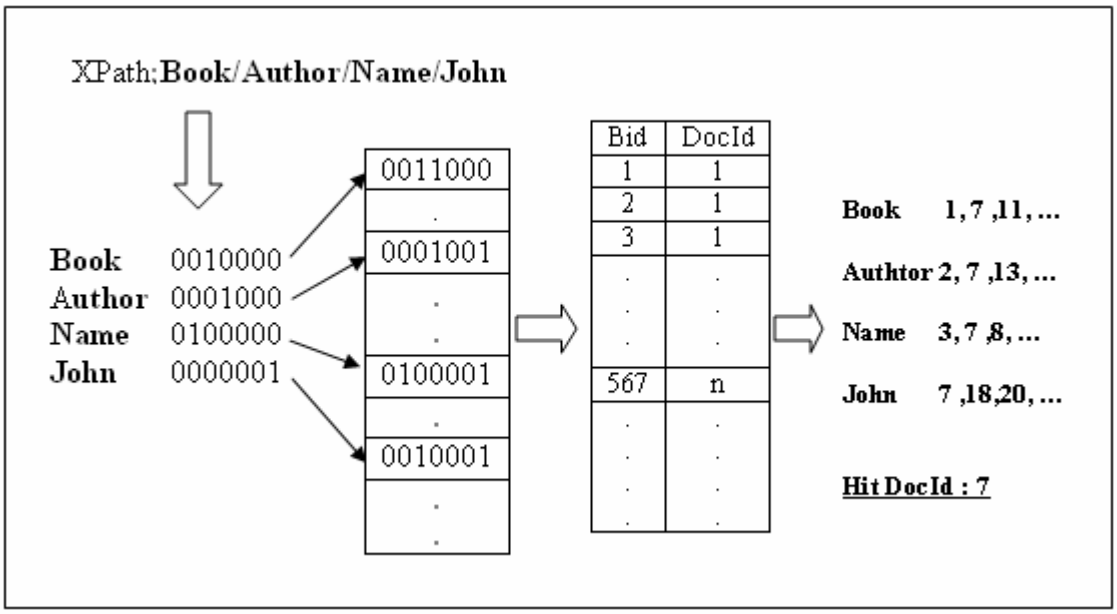

圖 18 Superimposed coding 的 block 對 XPath 查詢字串的影響

本論文所採用類似 superimposed coding 的編碼方式時，主要是將整個 XML 文件中的元素（element）、屬性名稱（attribute name）、屬性值（attribute value）及文字（text）的字串值，利用 Bloom Fliter 方法加以編碼。這裡要考量到如果以 superimposed coding 的編碼方式編碼，可能在 XPath 查詢時，因為 block 的關係而發生有該路徑及節點存在，卻資料編碼在不同 block 中，造成查詢結果失敗，如圖 17，分別以 Bloom Filter 求出 XPath 查詢字串中的 Book、Author、Name 及 John 四個字串的 bitstring 為 0010000，0001000，0100000，0000001，分別對簽章索引檔比對，如圖 18，本論文另外利用一個文件 table 儲存每個 block 所對應的文件 Id，也就是 DocId。每個查詢關鍵字經由 signature file 比對之後，如果存在該樣式，即取得該 block id 透過 block Id 到上述 table 查出該 block 所屬文件的 ID，接著比較各個查詢關鍵字經由上述過程所取得的文件 Id，如果存在相同文件 Id 則表示在資料庫中有某一文件包含這幾個關鍵字，可以進一步透過資料庫中，所儲存的位置資料進一步驗證是否該文件具備 XPath 表示式的階層關係。

3.3 XML 文件查詢

3.3.1 XPath expression 與 SQL 的轉換

XPath 表示式可以表現出 XML 文件中 element 間的直接關係(Direct)，如 school/student，以及間接關係(indirect)，如 school//firstname。這種一個 element 包含另一個 element 的查詢即所謂 containment query[12]。containment query 本身則可利用 containment join 的觀念來構建 sql

command，而經過 BEL 編碼存在關聯式資料庫的任意兩個 term（括號中的數據為該 term 經由 BEL 編碼後的位置資訊）：

T1(D1，begin1，end1，level1)

T2(D2，begin2，end2，level2)

具備以下關係：

（1）Containment Property.T1

包含 T2 必須滿足

a. D1=D2
b. begin1<begin2 ,end1>end2。

例如：(1,1,23,0)包含(1,9,13,2)

（2）Direct Containment Property

T1 direct contains T2 必須滿足

a. D1=D2
b. begin1<begin2,end1>end2
c. level1+1=level2。

例如：(1,1,23,0)直接包含(1,2,7,1)

（3）Tight Containment Property

T1 tight contains T2 必須滿足

a. D1=D2
b. begin1<begin2，end1>end2。

例如：(1,14,21,2) tight contains(1,15,20,3)。

因為 XML 文件的巢狀結構特性，tight containment 意味著也是 direct containment，但反之不成立。

根據上述幾個關係，可以歸納出以下幾個 XPath expression 轉換成 SQL COMMAND 的基本準則：

a. E//"T"

```
select *from ELEMENTS e，TEXTS t
where e.term='E' and t.term='T'
and e.docno=t.docno
and e.begin>t.wordno
and t.wordno<e.end
```

b. E/"T"

```
select *from ELEMENTS e，TEXTS t
where e.term='E' and t.term='T'
and e.docno=t.docno
and  e.begin<t.wordno
and  t.wordno<e.end
and  e.level=t.level-1
```

c. E="T"

```
select *
from ELEMENTS e，TEXTS t
where e.term='E' and t.term='T'
and  e.docno=t.docno
and  t.wordno= e.begin + 1
and  e.end=t.wordno + 1
```

d. E1/E2

```
select *
from ELEMENTS e1，ELEMENTS e2
where e1.term='E1' and e2.term='E2'
and  e1.docno=e2.docno
and  e1.begin> e2.begin
and  e1.end<e2.end
and  e1.level+1=e2.level
```

e. E1//E2

```
select *
from ELEMENTS e1，ELEMENTS e2
where e1.term='E1' and e2.term='E2'
and  e1.docno=e2.docno
and  e1.begin> e2.begin
and  e1.end<e2.end
and  e1.level<e2.level
```

例一、school/student；

查詢父節點是 school 的 student 節點，利用轉換準則 d 可以得到下面的 SQL 查詢命令：

```
SELECT e2.Eid
FROM
Elements e1 , Elements e2
WHERE e1.term="school"  AND e2.term="student"
AND e2.docno=e1.docno
AND e2.begin>e1.begin
AND e2.end<e1.end
AND e2.level=e1.level+1
```

例二、school//student；

查詢祖先節點是 school 的 student 節點，利用轉換準則 e 可以得到下面的 SQL 查詢命令：

```
SELECT e2.Eid
FROM Elements e1 , Elements e2
WHERE e1.term="school"
AND e2.term="student"
AND e2.docno=e1.docno
AND e2.begin>e1.begin
AND e2.end<e1.end
AND e2.level>e1.level
```

例三、Path 的 predicate，例如 school[fname="Jimmy"]//name；

查詢祖先節點是 school 而且 fname 節點的值等於 Jimmy 的 name 節點，透過轉換準則 c 跟準則 e 合併使用可以得到下面的 SQL 查詢命令：

```
SELECT e4.Eid
FROM Elements e1 , Elements e2 ,Texts t3 , Elements e4
WHERE e1.term="school"
AND e2.term="fname"
AND e2.docno=e1.docno
```

```
AND e2.begin>e1.begin
AND e2.end<e1.end
AND e2.level>e1.level
AND  t3.term="Jimmy"
AND t3.docno=e2.docno
AND t3.wordno=e2.begin+1
AND t3.wordno+1=e2.end
AND e4.term="name"
AND e4.docno=e1.docno
AND e4.begin>e1.begin
AND e4.end<e1.end
AND e4.level>e1.level
```

四、系統實作

4.1 系統架構

4.1.1 系統架構圖（圖 19）

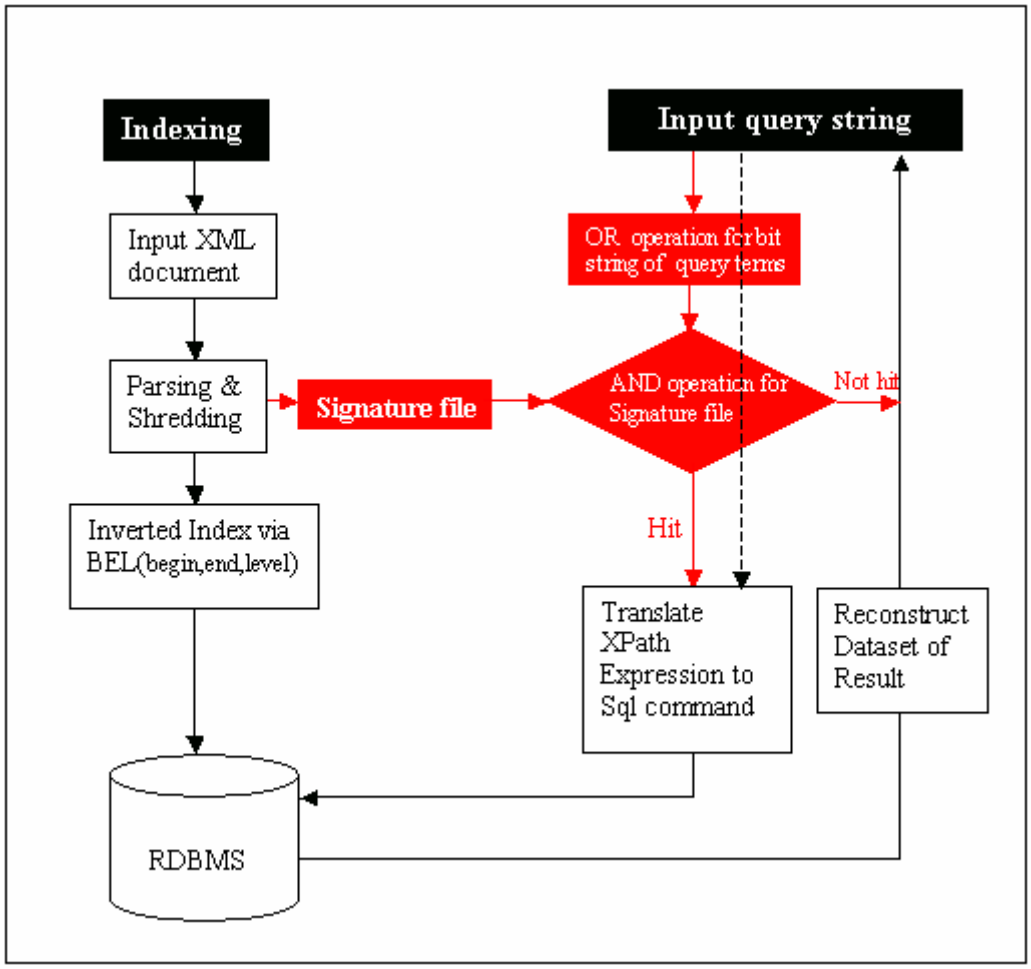

圖 19 系統架構圖

主要分成兩部分：

- 索引建立模組，如圖 19 左半部份：

  索引建立模組同時產生兩種索引資料，分別是由 BEL 編碼產生的階層索引資料，另一個為簽章索引檔。

- 查詢模組，如圖 19 右半部份：

主要是提供 user 利用 XPath 表示式查詢 XML 文件。

4.1.2 系統環境

- Operation system：Microsoft Windows 2000 Server SP4
- Web server ：Microsoft IIS 4.0
- CGI：Delphi 7.0 Enterprise
- XML parser：MSXML 3.0

4.2 系統說明

本系統提出一個以 Web 為介面，可以儲存（Store）並查詢（Query）XML 文件的解決方案，而這方案主要的工作如下：

（1）選擇 indexing 方法，將 XML 文件 shredding，取得 Elements，Attributes 和 Texts 在 XML 文件中的相關位置資訊（position information）。

（2）建立關聯式資料表（tables）來儲存 XML 文件。將保留原文件中順序（order）的 shredding 資料，存入關聯式資料表的列（rows）中。

（3）提供以 XPath Expression 為查詢語句的查詢功能。

（4）轉換 XPath Expression 為 SQL 命令，進行關聯式資料庫查詢動作。

（5）將查詢結果按照原 XML 文件的順序，重構（Reconstruct）XML 文件或片段。

首先，在 indexing 方法方面，系統採用 BEL（Begin, End, Level）方法（類似傳統 IR 的 inverted index），透過 SAX 將 XML 文件資料 shredding，以獲取其中元素（element）、屬性、屬性值以及文字（text）節點的位置資料（position information）做為 BEL 的索引值。實際運作如圖 20：

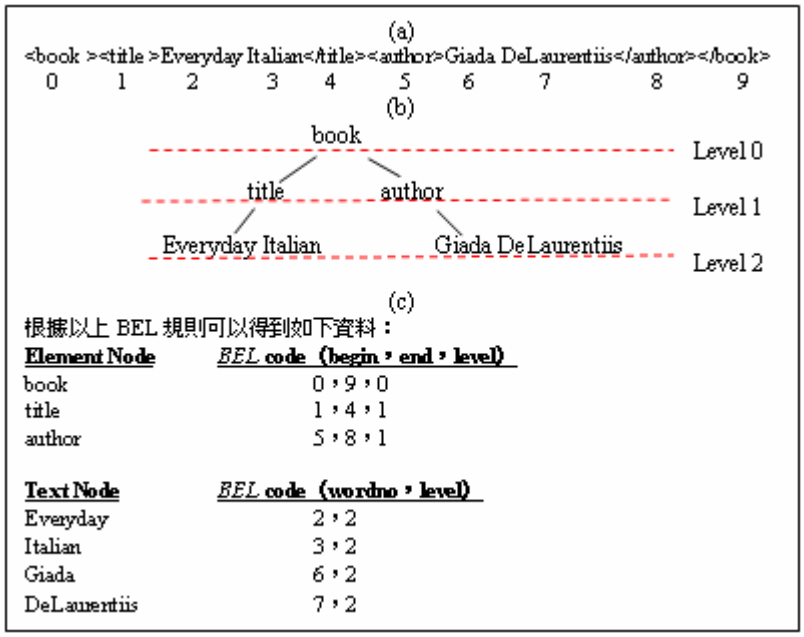

圖 20 BEL 編碼建立 XML 文件索引

圖 20(a)說明 XML 文件經由 SAX 解析後，取得每個節點，包含文字節點及元素節點序號（number）資訊透過這些序號資訊再加上圖 20(b)的 level，即可完成圖 20(c)中元素（element）、文字（text）、屬性（Attribute）以及屬性值（value）的 BEL 編碼。

不過，實際上 BEL 編碼並未針對屬性及屬性值編碼做說明，本文以（屬性,屬性值,begin）作為屬性的編碼格式，因為屬性在 XML 文件中並無次序性，與之相關的僅與其所屬的元素的 begin 序號，因為按照 XML 文件規定，屬性僅出現在元素的開始標籤，因此只要獲得該屬性的 begin 值以及其文件 id 就能決定該屬性屬於哪一個文件的哪一個元素的屬性。

在建立索引資料庫時，本論文提供兩種介面供使用：

一種是 Web 介面一次解析單一 XML 文件，並將取得的索引資料存入關連式資料庫，如圖 21。

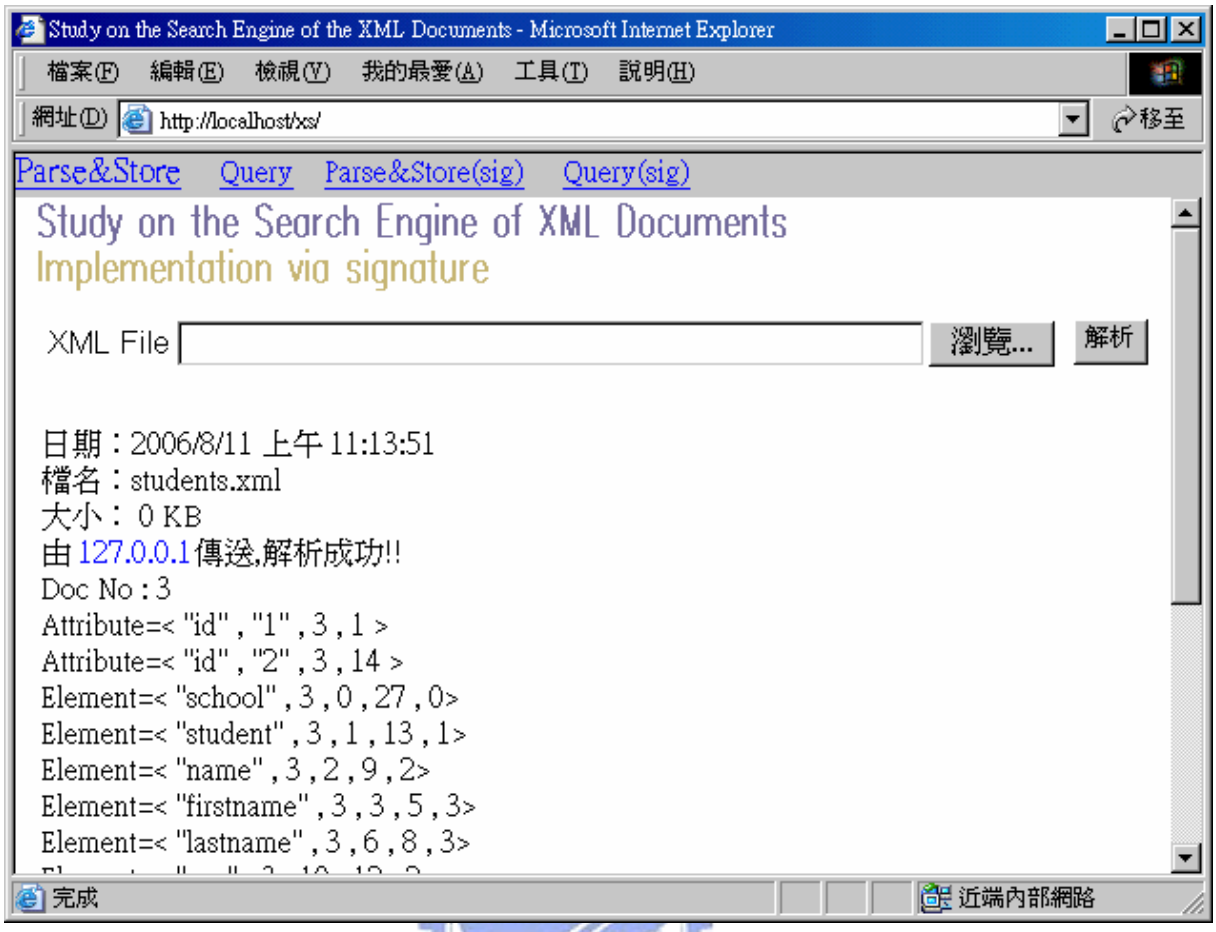

圖 21 線上 XML 文件索引建立系統

另一種方式是 console 端的 GUI 應用程式，主要是批次處理大量的 XML 文件的索引，如圖 22。BEL Index Builder 可以快速且方便的進行大量 XML 文件的索引建立工作。

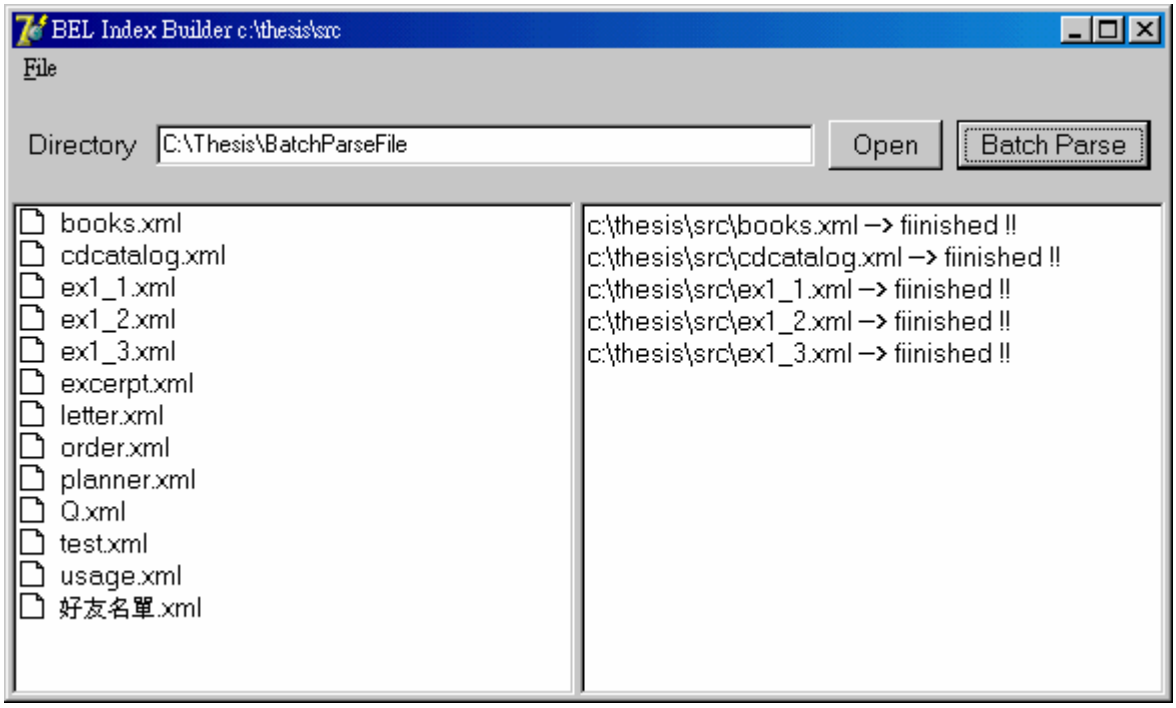

圖 22 批次 XML 文件索引建立程式畫面

第二步建立關聯式資料庫的資料表 schema 如下圖 23，圖 24，圖 25，圖 26：

```
Documents table{    docno(pk)    integer;
                    DocName      character;}
```

圖 23 關聯式資料表 Documents 的 schema

```
Elements table{     eleid (pk)   integer;
                    docno (fk)   integer;
                    term         character;
                    begin        integer;
                    end          integer;
                    level        integer;}
```

圖 24 關聯式資料表 Elements 的 schema

Texts table{	tid (pk)	integer ;
	docno (fk)	integer ;
	term	character ;
	wordno	integer ;
	level	integer ; }

圖 25 關聯式資料表 Texts 的 schema

Attributes table{	attid (pk)	integer ;
	docno (fk)	integer ;
	begin (fk)	integer ;
	attName	character ;
	value	character ; }

圖 26 關聯式資料表 Attributes 的 schema

根據前面 BEL 所得的 position 資料無損失地（lossless）存入上述資料表中。

經過前面兩個步驟後，系統提供以 XPATH 表示式的查詢介面，透過 3.3.1 的轉換模組，將 XPath expression 轉換成 SQL command 之後，透過 SQL command 來存取關聯式資料庫，所得到的資料集(dataset)將經由兩個階段完成 reconstruct 回原來 XML 文件。

第一個階段是選取（Select）階段，在這階段所有符合 XPath expression 的元素節點（element）是所謂的 inter-element order，由資料庫傳回的資料集（dataset），只是一筆一筆的元素（element）紀錄（record），不管到底是哪一文件的，為了在使用者介面更清楚的表達出文件的區別，因為這也是搜尋引擎的目的，在這個階段只要傳回 docno 即可。

接著是第二個階段，重建（reconstruct）階段，是所謂的(intra-element order)，這個階段利用第一階段所獲得的 docno 查出該文件中所有符合的節點，然後進行文件或是文件片段的（reconstruct）。

4.3 系統展示

(1)相關數據

檔案數	節點名稱		
	元素（Element）	字元（Text）	屬性（Attribute）
308	8221	8075	1528

表 3 系統實作相關數據

(2)查詢結果

- XPath 表示式查詢結果，如圖 27：

圖 27 XPath 表示式查詢結果

- 中文資料的查詢結果，如圖 28：

圖 28 查詢中文資料結果

- 未經過 signature file 的失敗查詢結果，如圖 29：

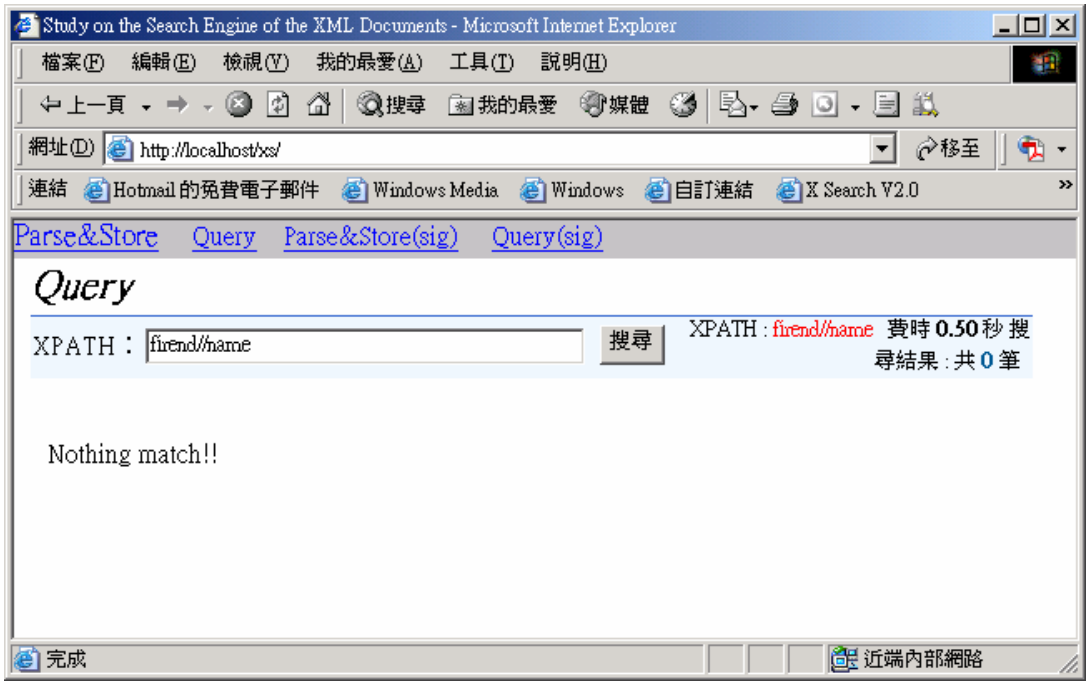

圖 29 未經由 signature file 的失敗查詢

- 經由 siganture file 的查詢結果，經由過濾機制回應查詢失敗時間明顯縮短，如圖 30：

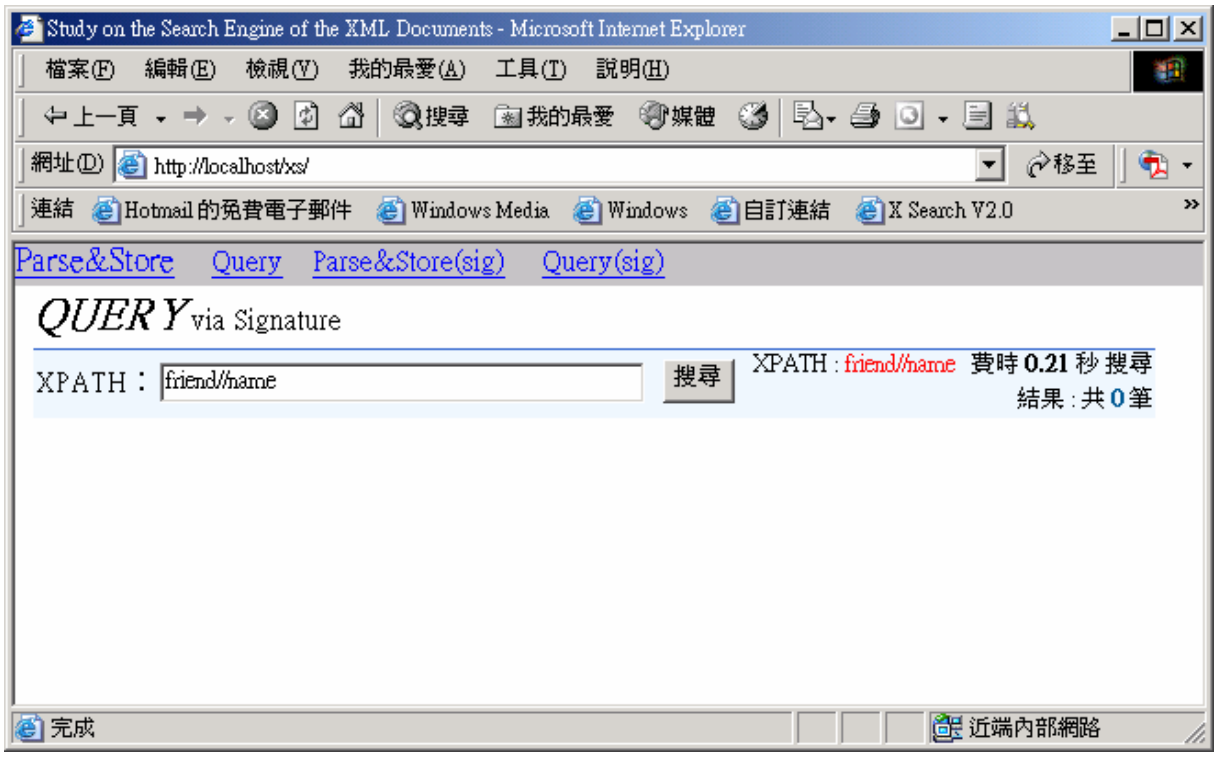

圖 30 經由 signature file 的查詢結果

五、結論

本論文建立一個以 XPath 表示式為查詢基礎的系統，並利用 BEL 編碼機制及簽章索引檔建立索引，查詢大量 XML 文件資料。此系統提供以下功能：

(1) 可以對大量文件進行 XPath 進行查詢產生該文件符合 XPath 表示式的元素集合，並重建出跟 XPath 表示式查詢單一 XML 文件一模一樣的文件片段。

(2) 對於文字節點與屬性值中文部份亦能提供相對應的查詢。

(3) Web 介面的查詢方式，方便 Internet 查詢。

(4) GUI 介面的 Index Builder 批次處理 XML 文件，取得索引資料，另外提供 Web 介面的單一 XML 文件索引建立機制。方便分散式的處理。

未來工作將朝向以下幾點進行：

(1) 支援中文標籤名稱的查詢。

(2) 大型檔案查詢速度的改進。

(3) XPath 表示式轉換成 SQL 查詢命令更加完整。例如：支援布林代數（and、or），萬用字（*）等。

(4) 建立更友善的查詢介面，降低需透過 XPath 學習才能操作查詢系統的瓶頸。

參考文獻

[1] W3C web site,http://www.w3.org/XML.

[2] W3C web site,http://www.w3.org/MarkUp.

[3] Google web site,http://www.google.com.

[4] Christos Faloutsos,"Access Method for Text",ACM Computing Surveys,17(1), 49-74,1985.

[5] W3C web site,http://www.w3.org/TR/xpath.

[6] DOM web site,http://www.w3.org/DOM.

[7] SAX project web site,http://www.saxproject.org.

[8] Daniela Florescu,Donald Kossmann,"Store and Querying XML Data using an RDMBS",Bulletin of IEEE Computer Society Technical Committee on Data Engineering,1999.

[9] P. Bohannon et al., "From XML Schema to Relations: A CostBased Approach to XML Storage," Proc. 18th Int'l Conf. Data Engineering (ICDE) ,pp. 64-75, Morgan-Kaufmann, 2002.

[10] Jayavel Shanmugasundaram,et al.,"Relational Databases for Querying XML Documents: Limitations and Opportunities",VLDB,1999.

[11] Jagadish HV,Al Khalifa S,"Timber: A native XML database", VLDB, 11(4), pp.274-291,2002

[12] Chun Zhang,et al.,"On Supporting Containment Queries in Relational Database Management Systems", In Proceedings of the ACM SIGMOD International Conferenece on Management of Data,Santa Barbara, California,May 2001.

[13] Bohdan O. Szuprowicz,Search Engine Technologies for the World Wide Web and Intranets,Computer Technology Research, Charleston,S.C.,1997.

[14] C. Faloutsos,"Signature files: Design and performance comparison of some signature extraction method.",In Proc. SIGMOD'85 Conference,pp.63-82,Austin, Texas, May 1985.

[15] Yannis Manolopoulos, Alexandros Nanopoulos, Eleni Tousidou, Advanced signature indexing for multimedia and Web applications, Kluwer Academic, Boston, 2003.

[16] Mark Graves, Designing XML Database, Prentice Hall PTR, Upper Saddle River, NJ, 2002.

[17] B.H.Bloom, "Space/time trade offs in hash coding with allowable errors ", Communications of the ACM, 13(7), pp.422-426, July, 1970.

[18] M.Scholl, "New file organizations based on dynamic hashing", ACM Transactions on Database Systems, 6(1), pp.194-211, March 1981.

[19] R. Fagin, J. Nievergelt, N. pippenger and H. R. Strong, "Extendible Hashing-A Fast Access Method for Dynamic Files", ACM Transactions on Database Systems, 4(3), pp.315-344, September 1979.

[20] P.A.Larson, "Linear Hashing with partial Expansions", Proc.6 Conference on VLDB, pp.224-232, Montreal, October 1980.

[21] R. Goldman, J. Widom, "Dataguides: Enabling query formulation and optimization in semistructured databases", Proc.of the 23rd Int'l Conf. on Very Large Data Bases (VLDB), pp.436-445, Athens: Morgan Kaufmann, 1997.

[22] Beverly Yang, et al., "Virtual Cursors for XML Joins", CIKM'04. pp.523-532, Washing-ton, D.C., USA, November 8-13 2004.

[23] R.Kaushik, et al.," Covering indexes for branching path queries", ACM SIGMOD'2002 Int'l Conf. on Management of Data (SIGMOD), pp.133-144 , Madison, Wisconsin, USA, 2002.

[24] W3C web site, http://www.w3.org/Protocols.

[25] T.Grust, J. Teubner, "Accelerating XPath evaluation in any RDBMS", ACM Transactions on Database Systems, 29(1), pp.91-131, March 2004.

[26] W3C web site, http://www.w3.org/2002/xmlspec.

[27] W3C web site, http://www.w3.org/XML/Schema.

[28] Igor Tatarinov, et al., "Storing and querying ordered XML using a relational database system", ACM SIGMOD'2002 Int'l Conf. on Management of Data (SIGMOD), PP.204-215, Madison, Wisconsin, USA, 2002.