典藏數位化中異質性資料之對應與建構演算法

劉邦鑑　　張雅惠
國立台灣海洋大學資訊工程研究所
{ lbj , yahui }@cyber.cs.ntou.edu.tw

摘要：

數位典藏國家型計畫的目的，是希望將國家龐大的珍貴文物與檔案以數位化典藏，而目前個各機構大部分是將資料儲存於關聯式資料庫系統中。但是為了方便使用者查詢，資料必須轉換為延伸式標記語言 XML。所以，關聯式資料與 XML 資料之間，如何進行有效地管理、查詢、檢索等，已經成為重要的議題。本篇論文的研究，主要在研討關聯式資料和 XML 資料結構的差異性與如何解決兩類型資料的不一致性，以達到資料共享的目的。我們提出建立一個對應字典，以記錄表格、欄位、資料型態、限制條件等相關的對應資訊。我們並討論相關的建構演算法和其效率。

１．緒論

目前推動的數位典藏國家型計畫，企圖將國家龐大的珍貴文物與檔案以數位化典藏，其中包含的技術相當廣泛，如數位影像的描素、時空座標及語言座標系統的建構、後設資料的規劃與分析、聯合目錄的建置等。

隨著資料數位化的進行，整個計畫也逐漸面臨到異質性資料整合的問題。這是因為目前個個機構（如故宮博物院、國立自然科學博物館等）所使用的數位典藏系統，大多是建立在具有高穩定性、高成熟度特性的關聯式資料庫系統(Relational Database Management System ；簡稱 RDB) 中。但是在資料交換共享時，會因為所使用的資料庫系統不同，增加了資料管理上的複雜度，也造成了各資料庫的整合困難。

自從 1998 年 W3C 組織正式頒佈了可擴充式標註語言(Extensible Markup Language；XML)，由於該語言允許使用者針對所需要的資料，自訂標註(Markup) 來描述資訊內容意義，所以已經成為網際網路資料交換的格式標準。在數位典藏國家型計畫中，也設定專責的後設資料工作小組[9]，針對不同性質的文物、器皿及珍貴文史檔案等，訂定以 XML 描述的後設資料（Metadata），以便利資料的交換及查詢。而各典藏單位也會將資料以 XML 格式匯出，儲存於 OAI-based 聯合目錄[3]中，並發展 Archive Resourse Identifier[4] 的檢索技術，以更快速有效能的方式，在眾多分散式的資料庫主機中，擷取所要檢索的資料。

由於典藏資料分別以關連式格式和 XML 格式儲存，所以如何解決兩類型資料的不一致性，以達到資料共享的目的，將是未來重要的課題。本篇論文的研究，主要在研討具有固定結構的關聯式資料，和具有半結構化特性的（Semi-Structured ）XML 資料，兩者結構的差異性。然後我們提出建立一個對應字典（Mapping Dictionary），用以記錄表格、欄位、資料型態、限制條件等相關的對應資訊。利用對應字典提供的資料， 我們可以進一步進行查詢語言的轉換，以分享不同類型的典藏資料 [1]。

CREATOR 表格

Cid	Name	Identification

SUBJECT 表格

Sid	Type	Terms	Owner

CREATION 表格

Cid	Sid	Mod_date

圖 1： RDB Schema

本篇論文架構如下：在第 2 節，我們將探討 XML 文件和 RDB 資料的差異性與範例資料的介紹。於第 3 節中定義對應字典 Mapping Dictionary（以下簡稱 MD）的格式與所記錄的內容。於第 4 節說明我們所提出的 MD 建構方式及其演算法；並於第 5 節中探討其效能。最後在第 6 節提出本篇論文的結論與未來展望。

2．資料的差異性比較

在本節中，我們探討 XML 文件和 RDB 資料可能的差異性。圖 1 是依據故宮書畫類的後設資料所設計的精簡 RDB 資料範例，其中包含了三個表格：CREATOR 表格，依序記錄了作者的姓名及其他識別資料；SUBJECT 表格依序記錄的是作品的類別、主題及典藏單位；CREATION 表格則是依序記錄作者與作品的關聯性和最後修改日期。在此例中，CREATION 的 Cid 與 Sid 為外來鍵(Foreign key)，分別對應到 CREATOR 的主鍵 (Primary key）Cid，與 SUBJECT 的主鍵 Sid 。

XML 資料通常利用文件型別定義（Document Type Definition；DTD） 予以規範其文件格式，在本論文中我們將複雜的 DTD 定義以 DTD Graph 表示。其中單實線方塊表示內部結點（Internal Node）元素，雙實線方塊表示有值的（Value Node）元素，虛線方塊則表示元素的屬性（Attribute）。若元素在 XML 文件中可出現零次以上，則於方塊右上加上星號。連接線部份，實線表示元素及其子元素關係；虛線與箭頭由 IDREF(s)屬性指向包含與其相對應的 ID 屬性之元素。

圖 2 則是我們依據圖 1 所設計，對應的 XML DTD Graph 範例。其中 CREATOR 表格中的欄位名稱，表示於 DTD 結構中的 creator 元素下；而 SUBJECT 表格的欄位名稱，表示於 DTD 結構的 subject 元素下。至於 CREATION 表格是對應到 creator 元素下的 creation 元素，但是我們另外新增一個 record 元素，用來紀載資料表的更新紀錄；也就是，圖 1 中的 Mod_date 欄位，在 DTD 中是在 record 元素下以 M_date 元素表示。

至於表格 CREATION 與 SUBJECT 的 join 關係，對應於圖 2 DTD Graph 中，是以 creation 元素下的 sid 屬性和 subject 元素下的 sid 屬性

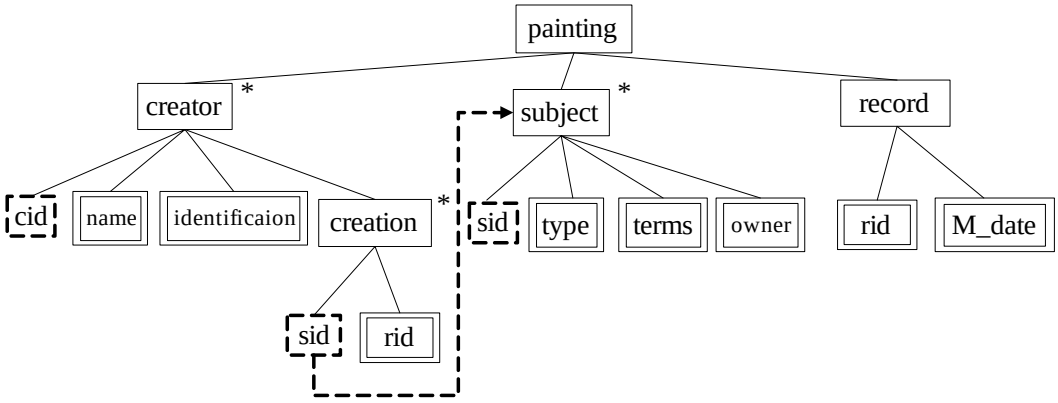

圖 2: XML DTD Graph 範例

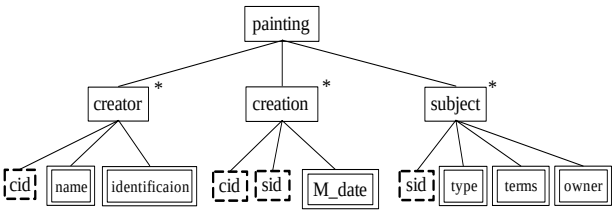

圖 3(a): 非巢狀結構的 DTD Graph 範例

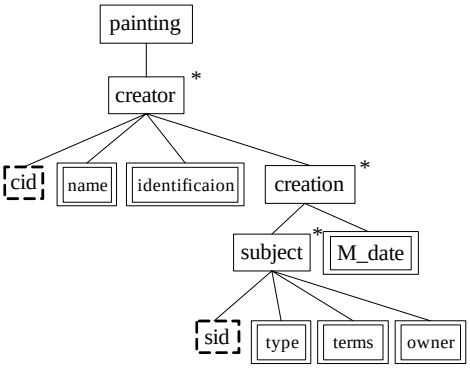

圖 3(b): 巢狀結構的 DTD Graph 範例

表示，也就是利用相同的屬性值來完成關連性的對應。而 M_date 與 ceration 間的關係，我們是以兩個相同的元素值 rid 來記錄其關聯性。

觀察雙方的資料表示，可以發現最大的差異性在於結構方面。這是因為，在關聯式資料庫中，儲存資料的結構是以扁平 (Flat) 的方式存在，而表格間彼此的關聯性僅能以 Primary key 、Foreign key 來定義 Join 限制。另一方面，XML 其對儲存資料的結構則有著較大彈性，基本上大致可分為下列幾種：

一、非巢狀結構：譬如在圖 3(a)中，creator、subject 及 creation 三個元素，都是在 painting 元素下，沒有互相包含的關係。所以元素間的關連，就必須透過相同的元素值或屬性值來建立，如同之前所討論。

二、巢狀結構：利用階層式的巢狀結構，來表示元素間的關係。譬如在圖 3(b)中，作品”subject 元素”，是巢狀表示在作者“creator 元素”的階層之下。所以元素間的關連，可直接透過如同 Xpath 的路徑表示法取得。

3．對應字典格式及範例

我們設計對應字典儲存關聯式資料與 XML DTD 間彼此的對應關係，在本章節中，我們將說明對應字典的格式和範例資料。

3.1 對應字典格式說明

我們用較具彈性的 XML 格式儲存對應資料，其 DTD 結構如圖 4 所示。為了方便說明，本論文是從 Relational Schema 對應至 XML DTD 的方向進行解析與建立。

基本上 MD 記錄了表格和欄位的對應路徑、表格與表格之間的 Join 關係和 XML DTD 的結構式限制。其中，Rdb 元素表示關聯式資料的目標資料庫名稱；Xdoc 表示 XML 文件檔案來源的儲存路徑。

Table 元素記錄了關聯式資料表格名稱 Tname 與欄位名稱 Fname 和其所對應的路徑 Txpath 與 Fxpath；以 RXfun 與 XRfun 分別記錄 RDB Schema 與 XML Schema 相互轉換時，欄位屬性的轉換關係。

此外，以 RStructure 元素來記錄關聯式資料端主表格 PTable 與其它相關聯表格 FTable 間的整合性限制，PF 與 FK 分別表示 primary key 與 foreign key，並給予每個 join 條件限制式一個 Xid 編號。以 Xstructure 元素來表示上述於 XML 端的結構性限制，其中屬性 Xid 用以表示與 RStructure 元素對應的流水編號。至於 ForPath，是用在當 XML 資料是利用巢狀結構；而 WherePath，是用在當 XML 資料是利用非巢狀結構。

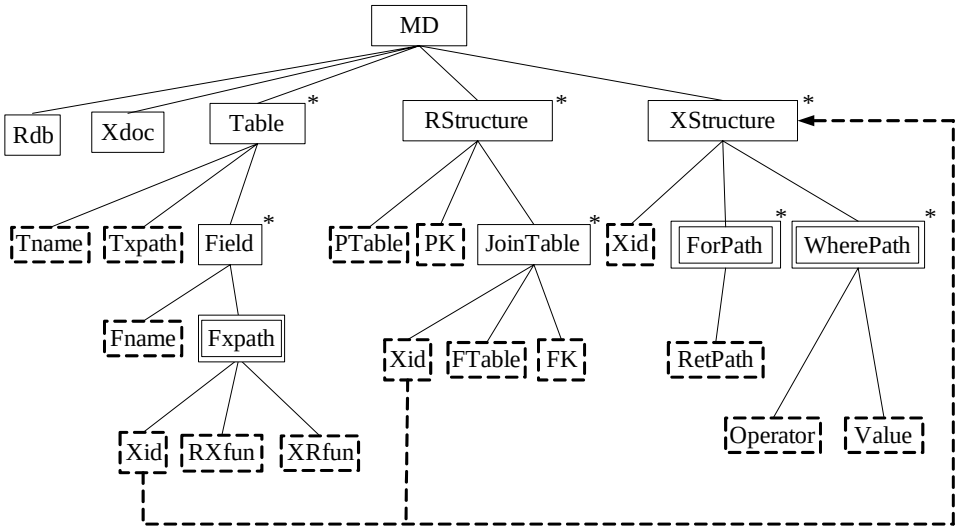

圖 4：對應字典 MD 之 DTD Graph

3.2 Mapping Dictionary 範例

針對圖 1 的 Relational Schema，假設其對應的 XML DTD Tree 如圖 2，我們所建立的 MD，會如表格 1 所示。舉例來說，在此 MD 範例中 L06 行以 TXPath 來記錄圖 2 中，CREATION 表格在 DTD 中的對應路徑；針對 Mod_date 欄位，我們以 RXFun 及 XRFun 記錄欄位屬性轉換，加上它所對應到的路徑如 L08 所示。

參考圖 2，因為 M_date 欄位對應於 DTD Tree 中，是利用同值相等元素 rid 和 creation 下的 rid 元素來完成於 XML 端的結構對應，所以我們將此結構限制式記錄於 XID= “XC1”的 Xstructure 元素中，其中 XC1 為此 XML 結構限制式的流水編號。

其次，關聯式資料表格 CREATION 的外來鍵 Cid 參考到 CREATOR 中的主鍵 Cid，在 MD 的結構中，我們以 RStructure 元素來記錄此 join 關係，並將此兩表格於 DTD 中的對應路徑記錄於 XID= “XC2”的 XStructure 元素中，其中 XC2 為此 XML 結構限制式的流水編號。

表格 1： Mapping Dictionary 範例

```
L01  <?xml version="1.0"?>
L02  <MD>
L03    <Rdb>Painting and Calligraphy </Rdb>
L04      <XDoc> </XDoc>
L05      <Tables>
L06        <Table Tname="CREATION"
             TXPath="/painting/creator*/creation*">
L07          <Field Fname="Mod_date">
L08            <Fxpath RXFun="Date2String"
                 XRFun="String2Date" XCID=”XC1”>
                 /painting/record/M_date </FXPath>
L09          </Field>……略
L10        </Table>……略
L11      </Tables>
L12      <RStructures>
L13        <RStructure PTable="CREATOR"
             PK="Cid">
L14          <JoinTable FTable="CREATION"
               FK="Cid" XID="XC2">
L15          </JoinTable>
L16        </RStructure>…略
L17      </RStructures>
L18      <XStructures>
L19        <XStructure XID="XC1">
L20          <ForPath>/painting/creator*/creation*</ForPath>
L21          <ForPath>/painting/record</ForPath>
L22          <WherePath Operator="="
             Value="/painting/record /rid">
             /painting/creator*/creation*/rid</WherePath>
L23        </XStructure>
L24        <XStructure XID="XC2">
L25          <ForPath>/painting/creator*</ForPath>
L26          <ForPath>/painting/creator*/creation*</ForPath>
L27        </XStructure>
L28      </XStructures>
L29  </MD>
```

4. 對應字典之建構

在本節中我們說明如何建立對應字典。首先我們介紹輔助的資料定義，然後介紹建立的演算法。

4.1 定義

在建立 MD 時，我們必須瞭解表格與表格之間的 Join 關係。為了清楚的表示該類關係，和提供有系統的建立 MD 的依據，我們將每一個關聯式表格以 RDB Node 形態表示，並根據表格之間的關係，表示成一棵 RDB Tree，如圖 5 所示。其中，每個 Relational Table 視為一個 RDB 節點 V，且 V＝(tn, pk, fk, f, m)：

- tn 為表格的名稱。
- pk 為表格的 Primary Key，可以為欄位集合或空集合。
- fk 為表格的 Foreign Key，可以為欄位集合或空集合。
- f 為表格的欄位集合且 f＝ {(fi, di) }，i＝1…n，其中 fi 為欄位名稱，di 為資料型態。
- m 為布林值，初始值為 0，用來標註示該節點是否已經被程式處理過，若未處理過為 0，反之為 1 (下次程式走訪時，便不會重複處理)。

一個 RDB schema 以一個 RDB Tree G = (V, E)表示，其中 V 為 RDB Node 的集合。若 Vj 中 foreign key 對應到 Vi，且(i != j) ，也就是 Vj Table 參考到 Vi Table，則建立一條有向線從 Vi 指到 Vj，並記錄於集合 E 中。

【Example】

以圖 1 為例，CREATION 表格分別以屬性 Cid 參考到 CREATOR 表格的 Cid 屬性，以屬性 Sid 參考到 SUBJECT 表格的 Sid 屬性；則相對應的 RDB Tree 如圖 5 所示，注意到本範例中有兩個 Root Node。

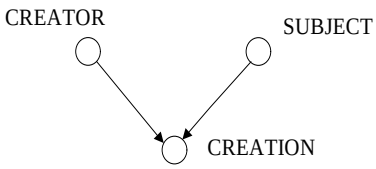

圖 5: RDB Tree 範例

為了自動化的建立 MD，我們還必須把一些輔助的對應資料先建立好。首先，在找出 RDB 欄位可能對應的 DTD 路徑部分，我們會先使用 Clio[5]的方法，若有不足的部份，使用者亦可自行填入正確的對應路徑。另外，在 DTD 結構方面，包含了巢狀結構、利用 IDREF 連接或用同值相等元素對應來表示。所以我們也記錄了 ID 和 IDREF 對應路徑，以及同值相等元素對應路徑的外部參考對應資料。

4.2 建構流程及演算法

MD 的建立步驟主要包含三大部分：(1)首先輸入一個 RDB Tree 的資料結構 (2)由 MDS 函式建立表格與欄位對應 (3)由 MDR 函式建立表格間的 Join 關係；經由上述三個步驟完成後，便輸出所產生的對應字典（Mapping Dictionary），如表格 1。

4.2.1 建立資料字典

建立 MD 的演算法列舉如表格 2，主程式 BuildMD 會分別由每個 RDB Tree 的根節點開始，並呼叫副程式 TraverseRDBT 繼續處理。TraverseRDBT 函式則是以遞迴的 Depth-First-Search 方式去走訪每一個 RDB 節點。其中若有子節點且沒有被標註過 (t.m=0)，就由該子節點開始繼續向下走訪，直到找到葉節點且沒有被標註，便呼叫 MDS 函式建立表格與欄位的對應。之後由葉節點返回，再對其他節點做建立表格和欄位對應

表格 2： BuildMD 演算法

```
【Algorithm BuildMD】
輸入：RDB Tree g
輸出：Mapping Dictionary md
L01 procedure BuildMD(RDB Tree g)
L02     {    md = ( );    //初始化的md為空白的xml文件
L03          for (each root node t in g)
L04            {    md = md + TraverseRDBT(t);
                //將每次走訪所建立出的 md 片段加入
L05            }
L06          return md;
L07     }

L08  procedure TraverseRDBT(RDB Node t)
L09  {    md' = ( );
L10     if (t.m == 1)    //代表該節點資訊已加入md中
L11          return md';
L12     if (t is a leaf node)
L13     {    md' = md' + MDS(t);
            //MDS 函式建立表格與欄位的對應
L14          t.m = 1;    //已走訪處理過，標示此節點為1
L15          return md';
L16     }
L17    for (each child c in t)
L18     {    if (c.m == 0) //節點尚未被標註
L19          md'= md'+ TraverseRDBT(c);
L20          md'= md'+ MDR(t, c);
L21     }       //MDR 函式建立表格間的 Join 關係
L22    md' = md' + MDS(t);
L23    t.m = 1;
L24    return md';
L25 }
```

部份，同時呼叫 MDR 函式，將父節點與子節點的 Join 關係結構建立起來。在這個過程中，將所新增的對應資訊加入 MD 中，最後輸出結果。

特別注意的是，L13 和 L22 呼叫的 MDS 函式是建立表格和欄位對應，該函式除了建立對應路徑外，亦會判斷欄位所對應的 DTD 路徑與表格的 DTD 路徑是否有巢狀結構；若沒有，就必須從 ID 和 IDREF 或同值相等元素的對應路徑中，找出對應的路徑。

表格 3： MDS 演算法

```
【Algorithm MDS】
輸入：RDB Node t
輸出：部份 Mapping Dictionary md'
外部參考：表格欄位對應 tm, ID 和 IDREF
          對應 idm, 同值相等元素對應 eqm
區域變數：表格路徑 t_path, 欄位路徑 f_path,
          IDREF 路徑 idref_path
Procedure MDS(RDB Node t)
Setp1: If (t_path==FindEquNodePath(t.tn, tm))
       then 將表格名稱、所有欄位的名稱與型態在 XML
            DTD 中的對應路徑，記錄於 MD 中;
Step2: If (!IsNest(t_path, f_path))
       then 新增一個限制式編號 Xid 於 Xstructure 元素中;
        If (idref_path==IDREFLink(t_path, f_path, idm))
        then 將表格與欄位在 DTD 中，利用 IDREF 來對應
             的路徑資訊，記錄於 Xstructure ForPath 中;
        else    // 找同值相等元素對應路徑
Step3: If equ_path1==EquNodeLink(t_path, eqm))
       If equ_path2 = EquNodeLink(f_path, eqm)
        then 將此用同值相等元素的關係，以 Xstructure
             中的 WherePath@Operator = "=" 連結並記錄;
Step 4: return 此單一 RDB Node 所建立出的 md';
```

至於在L20行呼叫的MDR函式則是處理表格間 Join 關係，如同 MDS 函式，也會判斷兩個表格之間是否有巢狀結構的對應路徑，若無巢狀關係則會去尋找是否有 ID 和 IDREF 對應路徑。上述將在稍後的演算法中詳加介紹。

4.2.2 MDS 函式建構之演算法

針對 RDB Tree 中的每一個 Node，我們要對其表格與欄位建立與 XML DTD Tree 的對應關係，在執行此演算法之前，我們必須建立好 RDB Node 與 DTD Tree 中的部分對應資訊，以便將適當的資訊加入至 MD 之中。其中 tm 表格記錄表格與欄位於 DTD Tree 中的路徑對應、idm 表格記錄 id 和 idref 對應路徑，另外 eqm 表格則記錄同值相等元素的路徑關係。演算法 MDS 列舉如表格 3，所會用到的函式簡介如下：

- FindEquNodePath(table | field ,tm)：輸入表格名稱或欄位名稱，參考表格欄位對應表，傳回 DTD 中的對應路徑。
- IsNest(path1 , path2)：傳入兩條路徑，判斷是否有包含關係（巢狀結構）。

- IDREFLink(t_path, f_path, idm)：傳入表格路徑、欄位路徑，參考 ID 和 IDREF 的對應表，回傳 IDREF 路徑。
- EquNodeLink(t_path, eqm)：傳入該 RDB Node 路徑，參考同值相等元素對應表，傳回同值相等元素的路徑。

當走訪每個 RDB Node 時，會與上述之外部參考表格比對，取出表格名稱與其位於 DTD Tree 中的路徑，並將之記錄於 MD 中。

接下來，會判斷該表格與欄位間是否有巢狀關係。若沒有巢狀關係時，再判斷是否有 ID 和 IDREF 對應，或是同值相等元素對應這兩種情況，並將相關資訊記錄於 MD 中，如 3.2 節中所述。

4.2.3 MDR 函式建構之演算法

針對 RDB Tree 中，表格與表格間的 join 關係，我們用 MDR 函式來建立，其中 RStructure 元素用以表示關聯式資料端主表格 Ptable 與其它相關聯表格 Ftable 間的整合性限制，演算法如表格 4 所列。

MDR 函式會將輸入的父子兩個節點，分別以主表格 Ptable 與外部參考表格 Ftable 記錄在 RStructure 元素下，並給此 join 關係式一個編號；另將此兩表格位於 DTD Tree 中的對應路徑，記錄於 XStructure 元素中，我們以 ForPath 元素來表示。如 3.2 節中所述，對應於表 1 的 MD 範例，相當於 L13 至 L16 及 L24 至 L27 行所示。當兩個 RDB Node 不為巢狀結構時，則利用前述之 ID 和 IDREF 對應路徑，來建立相關對應片段。

表格 4： MDR 演算法

```
【Algorithm MDR】
輸入：RDB Node t, RDB Node c
輸出：部份 Mapping Dictionary md'
外部參考：表格欄位對應 tm, ID 和 IDREF 對應 idm
區域變數：表格路徑 t_path, c_path
procedure MDR(RDB Node t, RDB Node c)
Step1: 記錄兩個表格 join 的資訊(表格名稱與鍵值)
       於 MD 的 Rstructure 中;
Step2:  t_path = FindEquNodePath(t.n);
        c_path = FindEquNodePath(c.n);
       // 找尋此兩表格於 DTD 中的路徑
Step3: 給予此 join 關係一個 Xid 編號，並記錄於
       XStructure 元素中;
Step4: If (IsNest(t_path, c_path))
     {  分別記錄此兩表格與 DTD 中的路徑於
        Xstructure 下的 ForPath 元素中;
     else    // 不為巢狀結構時
       {  idref_path = IDREFLink(t_path, c_path, idm);
          將表格與表格利用 IDREF 來對應的路徑資訊,
          記錄於 Xstructure ForPath 中;
       }
     }
```

5. 評估與討論

我們已經將第 4 節描述的演算法實做出來，使用的程式語言是 MS-Visual C++ 6.0。我們並設計四組不同的範例，於 Pentium-4 CPU 2.4GHz 配有 512MB DDR-Ram 的電腦上進行我們的實驗，測試演算法效能；其中，DB1 是第二節中圖 2 的範例資料、DB3 是我們針對學生選課所設計的 Schema、DB2 及 DB4 則分別是 DBLP 及 TCP-H 的 Schema，相關定義請參考[1]。

實驗的數據如表格 5 所示。綜合我們的實驗而言，建立 MD 的時間都小於 1 秒，所以執行的效率是可以接受的。

表格 5： 測試數據

內容資訊 / 實驗編號	RDB 表格數	RDB 欄位數	XML DTD Node 數	MD 檔案大小(Kb)	轉成MD所需時間(秒)
DB1	3	10	16	2.35	0.172
DB2	3	10	13	2.10	0.109
DB3	3	7	15	1.76	0.094
DB4	7	17	19	3.95	0.205

進一步分析這四組實驗數據。其中 DB1 與 DB2 在關連式方面的表格數和欄位數相同，差異在於 XML DTD 方面。DB2 為五層的巢狀結構，表格的 Join 資訊僅需記錄兩表格的路徑關係；而 DB1 是較扁平的四層巢狀結構，需另外建立 IDREF 參考資訊於 XStructure 元素中, 所以建立 MD 的需時間較久。

另外，DB4 具有最多的表格。與 DB1 至 DB3 比較，我們可以發現當表格的個數增多，表格間的 Join 關係越多，所需的建立時間也越長。原因在於我們必須建立更多的結構關係。由表 5，我們也可觀察相對於 DB4 的 MD 檔案是最大的。所以，由實驗的結果得知，當 RDB 或 XML 的結構越複雜，所建立的資料量越大，則建立 MD 的時間越多。

6．結論與未來展望

本論文主要在探討關聯式資料庫與 XML 資料，在資料表示方面的差異性，並實作建構出一個對應字典，用來記錄彼此之間的對應關係，包含表格與欄位的對應路徑、表格與表格間的 join 關係所需的資訊。未來，我們期望與數位典藏機構合作，實地測試對應字典的完整性；並研究如何利用對應字典實際達到在數位典藏計畫中資料共享的目的。

■ 誌謝：此計畫由國科會贊助。
編號 NSC93-2422-H-019-001

7．參考文獻

[1] 邱豐傑*“轉換 SQL 為 XQuery 之研究與實作”* M.S. Thesis 海洋大學資訊系 2003.

[2] 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，第27卷・第2期 2001

[3] 陳昭珍、陳雪華、陳亞寧 *”數位典藏國家型科技計畫 OAI-based 聯合目錄建置規畫”* 第一屆數位典藏技術研討會論文集 2002

[4] Chao-Chin Chou, Jan-Ming Ho*”ARI – The Archive Resource Identifier”* 第二屆數位典藏技術研討會論文集 2003

[5] Renée J. Miller, Mauricio Hernández, Laura M. Haas, Ling-Ling Yan,C. T.Howard Ho, Ronald Fagin, Lucian Popa. “*The ClioProject: Managing Heterogeneity*.” SIGMOD 2001.

[6] Joo Kyung-Soo, “ *A design of iddleware components for the connection between XML and RDB*”, Proceedings of IEEE International Symposium on Industrial Electronics. June 2001.

[7] Masatoshi Yoshikawa, Toshiyuki Amagasa, Takeyuki Shimura, Shunsuke Uemura , *”XRel: a path-based approach to storage and retrieval of XML documents using relational databases”* ACM Transactions on Internet Technology (TOIT), Volume 1, Issue 1, August 2001.

[8] Wang-Chien Lee, Gail Mitchell, Xin Zhang, *“Integrating XML Data with Relational Databases”*, Proceedings of 2000 ICDCS WORKSHOP, page(s): 47-F53, April 2001.

[9]中央研究院後設資料工作組
http://www.sinica.edu.tw/~metadata/news/news-frame.html