XML應用於具有多對多關係的資料交換

陳靖國
朝陽科技大學資訊管理系
jkchen@cyut.edu.tw

劉景仁
朝陽科技大學資訊管理系
s9214623@cyut.edu.tw

摘要

資料交換是企業之間商業往來的重要工作之一。XML 因為具備階層結構而適合描述具有一對多關係的資料。然而，真實世界許多資料之間的關係是多對多型式。如何使用 XML 來有效地承載多對多關係的資料是相當重要的。以往一些資料交換研究都集中於 XML 與關聯式資料模型之間的資料轉換。然而有些企業或組織目前仍然還在使用網狀式資料模型來儲存多對多關係的資料。本論文提出一個資料轉換模型，利用兩個演算法模組來做網狀式資料模型與 XML 文件之間的資料轉換。

關鍵詞：資料交換、XML、網狀式資料模型、多對多關係。

1. 簡介

資料交換在商業交易中扮演非常重要的角色。若干文件媒介例如 SGML、HTML、XML 和 PDF [7, 12, 13]等都曾經被用來當作資料交換載具，其中以 W3C 的 XML 最有名也最受歡迎[1, 9, 11, 14, 15, 20-23, 25, 26, 31]。 主要因素是一份 XML 文件在結構與資料上具有自我完備的(self-contained)的特性。現在企業大都使用關聯式資料庫來儲存資料，資料交換工作容易處理，因為已有若干處理關聯式資料與 XML 資料轉換方法被提出[3, 6, 8, 9]。但是一些歷史悠久的機關行號、企業組織，例如行政院主計處、榮民總醫院、台灣銀行、台灣中小企銀等，考量資料模型的穩定性與資料轉換風險等因素，還一直在使用早期的資料庫系統。這些單位如果要與使用關聯式資料庫的其他機構交換資料就有困難。因此如何讓這些單位能夠順利地執行資料交換確實有其必要。XML 相當適合儲存具有一對多關係(One-to-many Relationship)的資料。然而，具有多對多關係(Many-to-many Relationship)的資料廣泛存在於真實世界之中。舉例來說，一個大學資料庫裡，課程資料與學生資料之間有著多對多關係，因為一門課可以允許多位學生選修，而一個學生也可以選修多門課來上。最早的資料庫模型，階層式資料庫模型(Hierarchical Data Model)，例如 IBM 公司的 IMS[5], [6]，的資料通常是一對多關係，適用於金融行業資料儲存，例如一家銀行可以有若干家分行、一個客戶可以開若干戶頭等。後來改良為第二代的網狀式資料模型[10]時又增加功能，可以建立資料之間的多對多關係。使用網狀式資料模型的企業無法直接交換資料，因為 XML 階層特性很難保存資料之間的多對多關係。雖然有論文[2]提出一個階層式資料庫與 XML 資料的轉換方法，但是該方法無法直接應用在網狀式資料模型與 XML 資料的轉換。

在這篇論文裡，我們提出一個資料轉換模型可以將網狀式資料庫的資料，具有多對多關係，轉換成 XML 文件格式。也可以將含有多對多關係資料的 XML 文件轉換成網狀式資料庫。資料轉換模型包括兩個演算法模組分別完成上述的資料轉換。最後我們舉一個實例說明兩家企業如何透過此資料轉換模型來達成具有多對多關係的資料交換。論文架構描述如下。第一節是論文簡介。第二節是文獻探討，簡要說明 XML 文件和網狀式資料模型。第三節是資料交換模型相關技術與演算法說明。第四節舉出一個實例介紹如何交換資料。最後是結論。

2. 文獻探討

2.1 XML 文件

XML (eXtensible Markup Language) [16]是W3C於1998年提出，由SGML[4]簡化許多繁雜的語法後產生的一種標記語言，目的是要在資料交換上扮演重要的角色，因此對格式有嚴謹的規定。XML自我描述(self-describing)[2]特性，可以在文件本身定義該文件的規格，也可以透過XML Schema或DTD檢驗不同的文件是否為同一規格。為了方便管理眾多文件和存取效率的提昇[21, 24, 29, 30]，使用XML文件作為儲存媒介逐漸風行。

有關 XML 文件的基本結構可以參考文獻[16, 17, 18, 19, 27, 28]。XML 文件有兩種不同格式，第一種格式稱為良好格式(Well-Formed)，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)每一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起來。(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 或 XML Schema 中對於 XML 文件的規範。DTD 或 XML Schema 的功用是要讓 XML 文件在製作時有可以遵守的資料架構與屬性特徵。

```
<College>
  <Department name="IM">
    <Teacher tid="T001" name="Eric">
      <Student sid="S001" name="John"/>
    </Teacher>
  </Department>
  <Department name="IE">
    <Teacher tid="T101" name="Tom">
      <Student sid="S101" name="Peter"/>
      <Student sid="S102" name="Tim"/>
    </Teacher>
  </Department>
```

圖 1　XML 文件範例

圖 1 是一份 XML 文件的範例。此文件含有四個元素<College>、 <Department>、<Teacher>、和<Student>，其中<College>是根元素，包含兩個<Department>子元素。第一個<Department>子元素有 name 屬性(attribute)，其值是"IM"，包含一個<Teacher>子元素(name 屬性值是"Eric")，而且它又包含一個<Student>的子元素(name 屬性值是"John")。第二個<Department>子元素的 name 屬性值是"IE"，包含一個<Teacher>子元素(name 屬性值是"Tom")，而且它又包含兩個<Student>的子元素(name 屬性值，分別是"Peter"和"Tim")。這些元素之間以階層架構方式來呈現一對多的關係。

2.2 網狀式資料模型

IMS是一個階層式資料模型的資料庫。其內部資料之間的一對多關係，亦即一個父區段(parent segment)對應到多個子區段(child segment)，早期版本就有。後來IBM改良為IMS/2，藉由加入邏輯關係機制(Logical Relationship Mechanism)觀念[6]，增強建構資料之間的關係，由單純的一對多關係達到多對多關係之境界，變成網狀式資料模型開發平台而能夠實作網狀式資料庫。在這個模型中，每一個實體類型(Entity Type)實作為一個區段(Segment)[5]。透過邏輯關係機制，可以方便相同或不同資料庫的區段之間關係的建立。如果有兩個資料庫涉及，分別稱為實際(Physical)資料庫與邏輯(Logical)資料庫[6]。

邏輯關係機制有用到三種主要區段類型，描述如下。邏輯子區段(Logical Child Segment)是位於邏輯資料庫內的子區段，其父區段則是位於實際資料庫內。邏輯父區段(Logical Parent Segment)是位於邏輯資料庫內的父區段，其子區段則是位於實際資料庫內。實際父區段(Physical Parent Segment)是位於實際資料庫內的父區段，其子區段也是位於實際資料庫內。另有五種指標(Pointer)提供來實現邏輯關係機制，分別稱為階層向前(Hierarchical Forward, HF)、實際父母(Physical Parent,

PP)、邏輯父母(Logical Parent. LP)、邏輯第一小孩(Logical Child First, LCF)、和邏輯孿生向前(Logical Twin Forward, LTF) [6]。HF 指標在階層循序取用資料時用來指向下個區段。PP 指標用來指向一個實際父區段。LP 指標用來指向一個邏輯父區段。LCF 指標用來指向一個父區段的第一個邏輯子區段。LTF 指標用來從一個特定邏輯孿生區段指向另一個再它後面的邏輯孿生區段。圖二呈現一個資料具有多對多關係的網狀式資料庫。圖 2(a)表示資料庫的邏輯結構，圖 2(b)則顯示資料庫的實際結構。根區段“Department”包含兩個子區段“Teacher”和“Project”。 橋樑區段(Bridge Segment)“Participation”連結實際父區段“Teacher”和邏輯父區段“Project”。藉由區段“Participation”, 兩區段“Teacher”和“Project”之間可以建立起多對多的關係。一個老師可以參與多個計畫案，所以 Eric 有兩個計畫案“internet”和“database”。同理，一個計畫案可以讓多個老師參與，所以計畫案“database”有 Eric and Tom 參與。

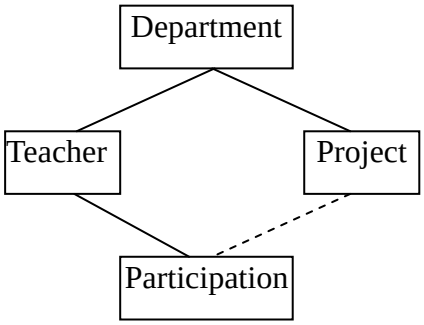

圖 2(a)資料庫的邏輯結構

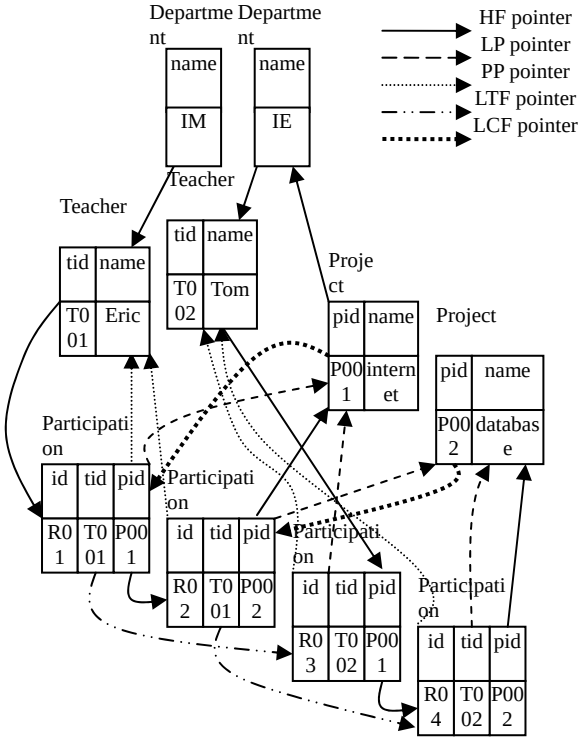

圖 2(b)資料庫的實際結構

3. 資料交換模型

以下介紹本文所提出的資料交換模型讓一家企業如何與另一家企業完成具有多對多關係的資料交換。首先由N2X模組摘取來源端資料庫內的資料，轉換成一份XML文件檔。然後將該XML文件經由網路傳送到接收端。再來是由*X2N*模組將XML文件內元素資料轉換成原本的多對多關係資料，最後存放到接收端的資料庫。圖3表示該資料交換模型的架構。N2X模組和X2N模組描述如下。

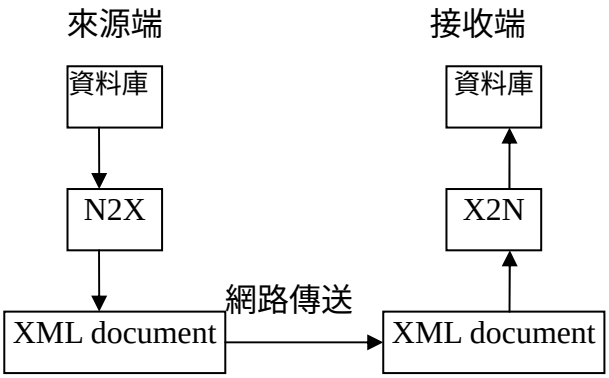

圖3 資料交換模型的架構

3.1 N2X 模組

此模組是用來將網狀式資料庫轉換成XML文件。網狀式資料庫的區段結構，如圖四所示，分為Prefix和Data兩部分。前者Prefix包含*level*和*pointer*兩欄位. Level欄位代表某個區段的等級(Level)。Pointer欄位由五個指標構成：HF, PP, LP, LCF, 和LTF, 用來完成邏輯關係機制的實行。後者Data部分儲存真實資料。一個區段*n*會被轉換成一個元素*n*。區段*n*的相關子區段會被轉換成元素*n*的相關子元素。區段*n*內的欄位名稱與值會被轉換成元素*n*的相關屬性名稱與值。部分元素會重複出現在XML文件，以便保存資料之間多對多關係的資訊，以下是N2X模組的實做演算法。

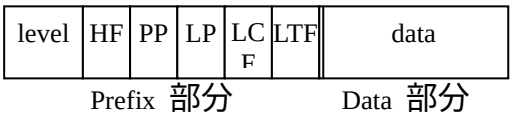

圖 4 網狀式資料庫的區段結構

```
Algorithm N2X(NDB_TREE N, XML_DOC X)
// Transform the data of a network database N
into an XML document X. //
input: N // a pointer to a network database //
output: X // an XML document //
begin
  STACK S; //a stack for saving temporary data//
  POINTER lt_ptr; //a pointer to a logical twin
  segment instance//
  if X is empty, then write the database name as
  the start tag of the root element to X;
  else
    get the name of N as the name of an element
    start tag and write to X;
    get the names and values of fields in N as the
    names and values of attributes of the element
    corresponding to N and write to X;
  endif;
    //check whether N is a logical child node or
    not //
  if LP pointer in N is not nil, then
    add PP_path as an attribute name to the
    element corresponding to N and get the
    absolute path of N.PP as the value of
    attribute PP_path, then write to X;
    add LP_path as an attribute name to the
    element corresponding to N and get the
    absolute path of N.LP as the value of
    attribute LP_path, then write to X;
      if LTF pointer in N is not nil, then
    add LTF_path as an attribute name to the
    element corresponding to N and get the
    absolute path of N.LTF as the value of
    attribute LTF_path, then write to X;
  endif;
endif;
//check whether N is a logical parent node or
not //
if LCF pointer in N is not nil, then
  add LCF_path as an attribute name to the
  element corresponding to N and get the
  absolute path of N.LCF as the value of
  attribute LCF_path, then write to X;
  assign N.LCF to lt_ptr;
  while lt_ptr is not nil, do
    get the name of lt_ptr as the name of an
    element start tag and write to X;
    get the names and values of fields in
    lt_ptr as the names and values of
    attributes of the element corresponding to
    lt_ptr and write to X;
    add PP_path as an attribute name to the
    element corresponding to lt_ptr and get
    the absolute path of lt_ptr.PP as the value
    of attribute PP_path, then write to X;
    add LP_path as an attribute name to the
    element corresponding to lt_ptr and get
    the absolute path of lt_ptr.LP as the value
    of attribute LP_path, then write to X;
    if LTF pointer in lt_ptr is not nil, then
      add LTF_path as an attribute name to
      the element corresponding to lt_ptr and
      get the absolute path of lt_ptr.LTF as
      the value of attribute LTF_path, then
      write to X;
    endif;
    assign lt_ptr.LTF to lt_ptr;
  end while;
  endif;
  if N.HF is not nil, then
    if N.level < N.HF.level, then push(N, S);
    else
      write the close tag of N to X;
        if N.level > N.HF.level, then
          for the number (N.level
          -N.HF.level) of top items in the
          stack S, do
            pop up an item i from S;
            write close tag of i to X;
          end for;
        endif;
    endif;
  else
    write the close tag of N to X;
    for all items in the stack S, do
```

```
        pop up an item i from S;
        write the close tag of i to X;
      end for;
      write the close tag of the root element to
      X;
      return X;
    endif;
  call N2X(N.HF, X); // recursive call N2X //
end N2X.
```

3.2 X2N 模組

此模組是用來將 XML 文件轉換成網狀式資料庫。XML 文件的根元素會被轉換成網狀式資料庫名稱。每一個非根元素會被轉換成一個區段，但是部分重複出現的元素除外。所有從 XML 文件摘取出來的區段會依其出現的順序儲存至資料庫。父元素與其子元素會被轉換成相對的父區段與子區段。相同父元素的子元素們所轉換成的區段會儲存在資料庫同樣 Level。XML 元素的屬性會被轉換成相對的區段內欄位，但是有四個屬性除外，分別是 PP_path, LP_path, LCF_path, 和 LTF_path，因為它們的值是用來建立多對多關係而非一般區段資料。以下是 X2N 模組的實做演算法。

```
Algorithm X2N(XML_DOC X , NDB_TREE N)
// Transform an XML document X into a network
database N. //
input: X // an XML document //
output: N // a pointer to a network database //
begin
  STRING data_string; //a string variable for
  saving a tag//
  STACK S; //a stack for saving a tag as a parent
  element//
  INTEGER level_count; //initial value is -1//
  read total data of the root element of X and use
  the start tag as the database name of N;
  while X is not empty, do
    read a tag from X to data_string;
    if data_string is a start tag, then
    push(data_string, S);
      if the current element instance is a
      sub-element of the element on the top of S,
      then
        push(the current element, S);
        increase the value of level_count by 1;
      endif;
      if the value of attribute LP_path in the
      current element instance is not nil and the
      value of attribute PP_path in the current
      element is the same with the absolute path
      of the parent of the current element, then
        create a current segment instance,
        pointed by the HF pointer of the previous
        created segment instance, for the
        corresponding current element instance;
        copy all the attributes in current element
        instance as corresponding fields to
        current segment instance excluding the
        four special attributes PP_path, LP_path,
        LTF_path, and LTF_path;
        assign the level number to the level field
        of current segment instance by counting
        the levels of the path in PP_path of
        current element instance;
        save the values of attributes PP_path and
        LP_path to the pointers of PP and LP,
        respectively, in current segment instance;
          if the value of attribute LTF_path in
          current element instance is not nil, then
            save the value of attribute LTF_path
            to the pointer LTF in current segment
            instance;
          endif;
        else
          create a current segment instance,
          pointed by N or the HF pointer of the
          previous created segment instance, for
          the corresponding current element
          instance;
          copy all the attributes in current
          element instance as corresponding
          fields to current segment instance;
          assign the value of level_count to the
          level field of current segment instance;
            if the value of attribute LCF_path of
            current element instance is not nil,
            then
              save the value of LCF_path to the
              LCF field of current segment
              instance;
            endif;
       else //data_string is a close tag//
        pop(S); decrease the value of level_count
        by 1;
      endif;
  end while;
  return N;
end X2N.
```

4. 一個實際資料交換劇本

此節舉出一個實際資料交換劇本來說明我們所提的資料交換模型的應用與可行性。假設有 A 與 B 兩所學校要分享系所資

訊，兩所學校都使用網狀式資料庫儲存管理資料並同意用 XML 文件來交換資料。A 學校首先將資料庫，如圖 2(b)所示，使用 N2X 模組轉換成相對的 XML 文件，如圖 5 所示，並透過網路傳送給 B 學校。B 學校再利用 X2N 模組將 XML 文件還原成原始的網狀式資料庫儲存起來，資料交換工作即告完成。同理，B 學校也可以使用同樣模式將資料分享給 A 學校。

```
<Department name=”IM” >
  <Teacher name=”Eric” tid=”T001”>
    <Participation id=”R01” tid=”T001” pid=”P001”
      PP_path=”Department/Teacher/T001”
      LP_path=”Department/Project/P001”
      LTF_path=”Department/Teacher/Participation/R03”>
    </Participation>
    <Participation id=”R02” tid=”T001” pid=”P002”
      PP_path=”Department/Teacher/T001”
      LP_path=”Department/Project/P002”
      LTF_path=”Department/Teacher/Participation/R04”>
    </Participation>
  </Teacher>
  <Project name=”internet” pid=”P001”
   LCF_path=” Department/Teacher/Participte/R01” >
    <Participation id=”R01” tid=”T001” pid=”P001”
      PP_path=”Department/Teacher/T001”
      LP_path=”Department/Project/P001”
      LTF_path=” Department/Teacher/Participation/R03”>
    </Participation>
    <Participation id=”R03” tid=”T002” pid=”P001”
      PP_path=”Department/Teacher/T002”
      LP_path=”Department/Project/P001”>
    </Participation>
  </Project>
</Department>
<Department name=” IE” >
  <Teacher name=”Tom” tid=”T002”>
    <Participation id=”R03” tid=”T002” pid=”P001”
      PP_path=”Department/Teacher/T002”
      LP_path=”Department/Project/P001”>
    </Participation>
    <Participation id=”R04” tid=”T002” pid=”P002”
      PP_path=”Department/Teacher/T002”
      LP_path=”Department/Project/P002”>
    </Participation>
  </Teacher>
  <Project name=”database” pid=”P002”
   LCF_path=” Department/Teacher/Participation/R02” >
    <Participation id=”R02” tid=”T001” pid=”P002”
      PP_path=”Department/Teacher/T001”
      LP_path=”Department/Project/P002”
      LTF_path=”Department/Teacher/Participation/R04”>
    </Participation>
    <Participation id=”R04” tid=”T002” pid=”P002”
      PP_path=”Department/Teacher/T002”
      LP_path=”Department/Project/P002”>
    </Participation>
  </Project>
</Department>
```

圖 5 轉換成相對的 XML 文件

4.結論

XML 文件適合記錄具有一對多關係的資料。然而要用 XML 文件來記錄具有多對多關係的資料並不是一件容易事。在這篇論文中，我們提出一個資料交換模型來解決這個艱難問題，主要關鍵點在於新增若干個特別屬性到每一個特別元素中，這些特別元素是從邏輯父/子區段轉換得來的。藉由此資料交換模型，企業組織之間可以分享彼此的資料。此外資料也可以讓使用其他類型的資料庫，例如關聯式資料庫、原生型 XML 資料庫，的企業組織分享，因為 XML 文件是一種標準媒介載具。所提的資料交換模型不僅可以讓網狀式資料庫交換資料，也能夠讓其他類型的資料庫交換資料，如果對 N2X 或 X2N 模組做適當修訂的話。

參考文獻

[1] E. Bertino and E. Ferrari, “XML and data integration,” *IEEE Internet Computing*, Vol. 5, Issue: 6, pp. 75-76, 2001.

[2] J. K. Chen and M. J. Liu, "A Model for Data Exchange between XML document and Hierarchical Databases," *proceedings of the 2002 International Computer Symposium*, Dec. 18-21, Hualien, Taiwan, ROC, Vol. 5, Session 8, E8-1, 2002

[3] J. Fong, F. Pang, and C. Bloor, “Converting relational database into XML document,” *Proceedings of 12th International Database and Expert Systems Applications*, 2001.

[4] E. R. Harold, *XML Bible*, John Wiley & Sons, 2001.

[5] IBM, *IMS Primer, http://www.redbooks.ibm.com*

[6] IBM, *IMS/ESA V5 Admin Guide: DB, http://www-306.ibm.com/software/data/ims/v5pdf/DFSA10C6.PDF*

[7] G. Kappel, S. Rausch-Schott, S. Reich, and W. Retschitzegger, **“**Hypermedia document and workflow management based on active object-oriented databases,” *Proceedings of the Thirtieth Hwaii International Conference, System*

Sciences, Vol. 4, pp. 377-386, 1997.

[8] J. S. Kim, W. Y. Lee, and K. H. Lee, "The cost model for XML documents in relational database systems," *ACS/IEEE International Conference, Computer Systems and Applications*, pp. 185-187, 2001.

[9] T. Kudrass, "Management of XML documents without schema in relational database systems," *Information and Software Technology*, Vol. 44, Issue: 4, pp. 269-275, 2002.

[10] T. William Olle, *The Codasyl Approach to Data Base Management*, John Wiley & Sons, 1978.

[11] J. Singh, "XML for power market data exchange," *IEEE Power Engineering Society Winter Meeting*, Vol. 2, pp. 755-756, 2001.

[12] M. B. Spring, "Reference model for data interchange standards," *Computer*, Vol. 29, Issue: 8, pp. 87-88, 1996.

[13] R. Summers, J. J. L. Chelsom, D. R. Nurse, and J. D. S. Kay, "Document management: an Intranet approach," *IEEE the 18th Annual International Conference, Engineering in Medicine and Biology Society, Bridging Disciplines for Biomedicine*, Vol. 3, pp. 1236-1237, 1997.

[14] M. Sundaram and S. S. Y. Shim, "Infrastructure for B2B exchanges with RosettaNet," *The third International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems*, pp. 110-119, 2001.

[15] S. Wegener and D. Davis, "XML TPS data exchange," *IEEE AUTOTESTCON proceedings of Systems Readiness Technology Conference*, pp. 605-615, 2001.

[16] W3C, Extensible Markup Language (XML) 1.0 (second Edition) W3C Recommendation, 6 October 2000, *http://www.w3.org/TR/2000/REC-xml-20001006*, 2000.

[17] W3C, XML Schema Part 0: Primer W3C Recommendation, 2 May 2001, *http://www.w3.org/TR/2001/REC-xmlschema-0-20010502*, 2001.

[18] W3C, XML Schema Part 1: Structures W3C Recommendation, 2 May 2001,*http://www.w3.org/TR/2001/REC-xmlschema-1 -20010502*, 2001.

[19] W3C, XML Schema Part 2: Datatypes W3C Recommendation, 2 May 2001, *http://www.w3.org/TR/2001/REC-xmlschema-2-20010502*, 2001.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

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

[28] W3C, Standard Generalized Markup Language,

http://www.w3.org/MarkUp/SGML/

[29] 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.

[30] 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.

[31] 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.