**Tìm hiểu XML**

1. **Giới thiệu:**

Ngoài cách lưu trữ dữ liệu trong các hệ quản trị CSDL ra thì có thể lưu trữ dữ liệu trong file TXT, file JSON hay file XML đều được. Tuy nhiên với những hệ thống lớn thì bắt buộc ta phải lưu trữ trong hệ quản trị CSDL bởi vì nó cung cấp những tính năng giúp quản lý dữ liệu tốt hơn. Còn đối với XML hay JSON thì ứng dụng lớn nhất của nó trong lập trình web đó là xây dựng các serveice và API (phần mềm trung gian cho phép hai ứng dụng giao tiếp với nhau), nghĩa là các API đó sẽ trả kết quả về dạng JSON hoặc XML để các hệ thống khác có thể hiểu được.

*\* Lịch sử hình thành:*

Vào năm 1986, ngôn ngữ đánh dấu tiêu chuẩn tổng quát (SGML) biểu thị dữ liệu trong những ứng dụng xử lý văn bản đa dạng và có cấu trúc tinh vi nhưng nó có nhược điểm là rất rắc rối, khó học, khó sử dụng,... Vào năm 1990, HTML ra đời để khắc phục những nhược điểm đó, là một phần nhỏ của SGML nhưng mọi người rất dễ dùng. Đến năm 1995, người ta bắt đầu thấy được sự giới hạn của HTML, mặc dù nó được Netscape, Microsoft cố gắng thêm thắt, bởi sự thịnh hành của Web. Do đó, nhiều người có ý định quay trở lại SGML nhưng lại ái ngại.

Tháng 11 năm 1996, bản đặc điểm kỹ thuật đầu tiên của XML (e**X**tensible **M**arkup **L**anguage) ra đời nhằm mục đích đơn giản hóa SGML để nó dễ dùng như HTML mà đồng thời mạnh mẽ, dễ dùng,...

Vào tháng 2 năm 1998, tổ chức mạng toàn cầu World Wide Web Consortium (W3C) phê chuẩn cho chính thức thi hành.

1. **XML là gì?**

***a. Ngôn ngữ XML:***

* XML là một ngôn ngữ đánh dấu (Markup Language), định nghĩa tập hợp các qui tắc để mã hóa các tài liệu trong một định dạng.
* XML không phải là một ngôn ngữ lập trình vì nó không thực hiện bất kỳ thuật toán hay kỹ thuật tính toán nào để tạo các chương trình máy tính.
* XML thường được lưu trữ trong một text của một tài liệu để phân ranh giới và gán nhãn các phần của tài liệu đó.

***b. Đặc trưng:***

* XML là một chuẩn chung: XML được phát triển bởi W3C và có sẵn như là một chuẩn mở.
* XML là có thể mở rộng: cho phép người dùng tự định nghĩa các thẻ theo qui ước riêng của mình để phù hợp với ứng dụng của bạn.
* XML mang dữ liệu, chứ không hiển thị nó: XML cho phép bạn lưu giữ dữ liệu, không quan tâm đến cái cách nó sẽ được hiển thị.

***c. Chức năng:***

* XML có thể núp phía sau để đơn giản hóa việc tạo ra các tài liệu HTML cho các website lớn
* XML có thể được sử dụng để trao đổi thông tin giữa các tổ chức và hệ thống
* XML có thể được sử dụng để Offload và Reload các cơ sở dữ liệu
* XML có thể được sử dụng để lưu trữ và sắp xếp dữ liệu, có thể tinh chế dữ liệu cần xử lý
* XML có thể dễ dàng để được hợp nhất với Style Sheet để tạo ra hầu như bất kỳ kết quả mong muốn nào.

Nói chung, bất kỳ kiểu dữ liệu nào có thể được biểu diễn như là một tài liệu XML.

1. **Tài liệu của file XML:**

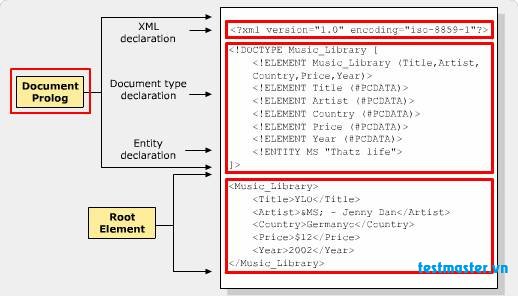
* Để tạo ra các tài liệu XML ta có thể sử dụng các trình soạn thảo đơn giản như: Notepad, Wordpad,.. hoặc chuyên nghiệp hơn như XML Writer.
* File XML sẽ có phần mở rộng là .xml. Tuy nhiên, vẫn hoàn toàn có thể sử dụng ngôn ngữ lập trình để thay đổi phần mở rộng cho nó.
* Một file XML gồm có 2 phần:
  + Document Prolog:

+ Khai báo XML

+ Khai báo kiểu dữ liệu trong XML

+ Khai báo thực thể

* + Root Element: chứa tất cả các phần tử và nội dung của file XML.



***a. Document Prolog:***

*a.1. Khai báo XML*

* Cú pháp:

<?xml

version="version\_number"

encoding="encoding\_declaration"

standalone="standalone\_status"

?>

Trong đó: + *version*: phiên bản XML

+ *encoding*: xác định mã hóa ký tự được sử dụng trong tài liệu

+ *standalone*: tình trạng liên kết của tài liệu XML và các tài liệu khác.

* Qui tắc:

+ Khai báo XML là tùy ý, nhưng khi được sử dụng, nó phải xuất hiện ở dòng đầu tiên trong tài liệu XML

+ Phải bắt đầu với “<?xml>” và kết thúc với “?>”

+ Thứ tự đặt tham số là quan trọng, thứ tự đúng là:

* + version: bắt buộc phải có
  + encoding: một HTTP Protocol có thể ghi đè giá trị của encoding
  + standalone: tham số này chỉ có 2 giá trị hợp lệ là “yes” hoặc “no” và giá trị mặc định là “no”

+ Tên và giá trị các tham số là phải phân biệt kiểu chữ

+ Tên luôn luôn ở dạng chữ thường

+ Các tham số có thể được chứa trong trích dẫn đơn hoặc kép.

*a.2. Khai báo kiểu dữ liệu XML*

*a.3. Khai báo thực thể*

***b. Root Element:***

* Phần này là các khối của XML. Chúng phân chia tài liệu thành các phần theo một cấu trúc có hệ thống, mỗi phần phục vụ cho một mục đích cụ thể. Các phần tử có thể là các Container, là một tổ hợp gồm text và các phần tử khác.
* Cú pháp cơ bản để viết một phần tử XML:

<element-name attribute1 attribute2>

....content

</element-name>

* Trong đó: + *element – name* là tên phần tử.

+ *attribute1, attribute2* là các thuộc tính của phần tử, được phân biệt bởi các khoảng trống trắng. Một thuộc tính được viết như sau:

name = "value"

*name* được theo sau bởi một dấu bằng (=) và một chuỗi *value* bên trong trích dẫn kép (“ ”) hoặc trích dẫn đơn (‘ ’)

* Phần tử trống trong XML:

Cú pháp:

<name attribute1 attribute2...>

</name>

Hoặc rút gọn thẻ đóng

<name attribute1 attribute2.../>

* Qui tắc thẻ:
  + Tên thẻ có thể chứa bất kỳ ký tự chữ - số và các dấu câu được cho phép xuất hiện trong tên phần tử: gạch nối (-), dấu gạch dưới (\_), dấu chấm (.) và dấu hai chấm (:)
  + Tên thẻ phải bắt đầu là một chữ cái hoặc dấu gạch dưới (\_)
  + Tên thẻ không được bắt đầu bằng XML (hoặc XMl, XmL,..) vì nó giống với thẻ khai báo header
  + Mỗi thẻ mở đều phải có thẻ đóng và các tên thẻ là phân biệt kiểu chữ
  + Các thẻ khi mở và đóng phải tuân theo đúng trình tự chính xác: một thẻ XML được mở trong phần tử khác phải được đóng trước khi phần tử bên ngoài đó được đóng
  + Các thuộc tính (atributes) của một thẻ luôn luôn tồn tại theo cặp theo quy ước: name = “value”; không nên đặt tên thuộc tính trùng nhau, và giá trị của thuộc tính phải đặt trong cặp dấu nháy kép hay nháy đơn. Tên của thuộc tính (atribute) sẽ theo qui luật đặt tên giống như đối với tên thẻ

1. **Một số thành phần khác trong XML:**
   * + 1. ***Comment*** *(ghi chú, giải thích)*

Các comment được bổ sung để thêm lời ghi chú hoặc giải thích phần code, giúp cho XML code dễ hiểu hơn. Comment có thể được sử dụng để bao các link, thông tin và các mục có liên quan.

Chúng là chỉ nhìn thấy chỉ trong source code; không trong XML code. Các comment có thể xuất hiện bất cứ đâu trong XML code.

* Cú pháp: <!-------Your comment----->

Một comment bắt đầu với “<!- -” và kết thúc với “- - >”. Có thể thêm phần ghi chú văn bản như là các comment ở giữa các ký tự.

* Qui tắc cho Comment:
* Các comment không thể xuất hiện trước phần khai báo XML
* Các comment có thể xuất hiện ở bất cứ đâu trong một tài liệu
* Các comment không được xuất hiện bên trong các giá trị thuộc tính
* Comment này không được lồng trong comment khác.
  + - 1. ***Character Entity*** *(thực thể ký tự)*
* Chúng được khai báo trong phần Document Prolog hoặc trong một DTD - là một tập hợp các tờ khai đánh dấu để xác định một loại tài liệu cho các ngôn ngữ đánh dấu. Các thực thể ký tự được sử dụng để hiển thị các ký tự/biểu tượng đặc biệt mà không có sẵn để có thể gõ một cách trực tiếp từ bàn phím hoặc một số biểu tượng được dành riêng, các biểu tượng này không thể được sử dụng như là nội dung trong XML code (ví dụ: < và > được sử dụng cho các thẻ mở và đóng)
* Các loại thực thể ký tự:
* Thực thể ký tự được định nghĩa trước
* Thực thể ký tự dạng số
  + Tham chiếu thập phân: &# decimal number ;
  + Tham chiếu thập lục phân: &#x Hexadecimal number ;

| **Ký tự** | **Thực thể ký tự** | **Tham chiếu thập phân** | **Tham chiếu thập lục phân** |
| --- | --- | --- | --- |
| & | &amp | &#38; | &#x26; |
| " | &quot | &#34; | &#x22; |
| ' | &apos | &#39; | &#x27; |
| < | &lt | &#60; | &#x3C; |
| > | &gt | &#62; | &#x3E; |

* Thực thể ký tự được đặt tên

Để nhớ các ký tự số là khá khó khăn, thì loại thực thể ký tự thường hay sử dụng nhất là thực thể ký tự gọi tên. Ở đây, mỗi thực thể được chỉ định với một tên. Ví dụ: 'Aacute' biểu diễn ký tự chữ hoa https://vietjack.com/xml/images/namedentity_1.png.

'ugrave' biểu diễn ký tự chữ thường https://vietjack.com/xml/images/namedentity_2.png.

* 1. ***Character Data*** *(Khu vực CDATA)*

CDATA được định nghĩa như là các khối text mà không được phân tích bởi Parser (Parser có mục tiêu là biến đổi XML thành một code có thể đọc được), nhưng được nhận ra khi đánh dấu. Bằng việc sử dụng khu vực CDATA, bạn đang chỉ lệnh cho Parser rằng khu vực này trong tài liệu không cần đánh dấu, nên được bỏ qua (hay giữ nguyên), và nên được đối xử như là phần text thông thường.

* Cú pháp:

<![CDATA[

characters with markup

]]>

Cú pháp trên bao gồm ba phần:

* Phần bắt đầu CDATA: CDATA bắt đầu với 9 ký tự **<![CDATA[**
* Phần kết thúc CDATA: Khu vực CDATA kết thúc với 3 ký tự **]]>**
* Khu vực CDATA: Các ký tự ở giữa hai phần trên được thông dịch như là các ký tự, không đánh dấu. Khu vực này có thể chứa các ký tự đánh dấu (như <, >, và &), nhưng chúng được bỏ qua bởi bộ xử lý XML.
* Qui tắc cho CDATA:
* CDATA không thể chứa chuỗi “]]>” ở bất cứ đâu trong tài liệu XML
* Việc lồng nhau là không được cho phép trong khu vực CDATA.
  1. ***WhiteSpace*** *(khoảng trắng)*

Khoảng trắng là một tập hợp gồm các khoảng trống, tab và newline. XML chứa hai loại khoảng trắng:

* Khoảng trắng quan trọng:

Xuất hiện bên trong phần tử mà chứa text cùng với phần đánh dấu.

VD1: <name>TanmayPatil</name>

VD2: <name>Tanmay Patil</name>

Hai phần tử ở hai ví dụ trên là khác nhau bởi vì có khoảng trống ở giữa *Tanmay* và *Patil*. Bất kỳ chương trình nào khi đọc phần tử này trong một file XML sẽ duy trì sự khác nhau này.

* Khoảng trắng không quan trọng:

Các khoảng trắng đó chỉ có nội dung phần tử được cho phép.

VD1: <address category="residence">

VD2: <address category=" residence">

Hai phần tử ở hai ví dụ trên là giống nhau.

* Thuộc tính đặc biệt **xml:space** có thể được đính kèm với một phần tử

<!ATTLIST address **xml:space** (**default|preserve**) 'preserve'>

* Với giá trị default: chế độ xử lý khoảng trắng mặc định của một ứng dụng nên chấp nhận phần tử này
* Với giá trị preserve: chế độ duy trì tất cả khoảng trắng.
  1. ***Processing Instrucstions*** *(Chỉ thị xử lý)*

Chỉ thị xử lý cho phép mô tả thêm một số thông tin (liên quan đến xử lý) về tài liệu XML có ý nghĩa riêng với một công cụ xử lý nào đó. Đây là một phương pháp cho phép mở rộng, bổ sung các xử lý riêng vào một lớp tài liệu XML cùng thuộc một hệ thống phân lớp nào đó.

* Cú pháp: <?target instructions?>

Trong đó: + target: nhận diện ứng dụng mà chỉ thị hướng tới

+ instruction: nó là chỉ thị mà miêu tả thông tin cho ứng dụng để xử lý.

Một chỉ thị xử lý với một thẻ đặc biệt “<?” và kết thúc với “?>”. Việc xử lý nội dung kết thúc ngay sau khi gặp chuỗi “?>”.

* Ví dụ: Chỉ thị xử lý được sử dụng để liên kết tài liệu XML tới một Style Sheet:

<?xml-stylesheet href="tutorialspointstyle.css" type="text/css"?>

Trong đó: + target là xml-stylesheet href="tutorialspointstyle.css"

+ instuction là type="text/css"?>

Trong trường hợp này, thẻ chỉ thị xử lý định dạng thể hiện tài liệu XML với “chương trình định dạng” theo ngôn ngữ css được lưu trữ bên trong tập tin tutorialspointstyle.css

* Quy tắc chỉ thị xử lý:

Một chỉ thị xử lý có thể chứa bất kỳ dữ liệu nào ngoại trừ tổ hợp “?>”, mà được thông dịch như là delimiter đóng.

* 1. ***Encoding*** *(Mã hóa ký tự)*

Là tiến trình chuyển đổi các ký tự Unicode thành biểu diễn nhị phân tương đương của chúng. Khi bộ xử lý XML đọc một tài liệu XML, nó mã hóa tài liệu phụ thuộc vào kiểu mã hóa. Vì thế, chúng ta cần xác định kiểu mã hóa trong khai báo XML. Có hai kiểu mã hóa ký tự chính: UTF-8 và UTF-16.

Cú pháp: Kiểu mã hóa được bao trong khu vực XML Prolog của tài liệu XML:

* + UTF-8: <?xml version="1.0" encoding="UTF-8" standalone="no" ?>
  + UTF-16: <?xml version="1.0" encoding="UTF-16" standalone="no" ?>
  1. ***Validation*** *(Xác nhận)*

Validation là một tiến trình xác nhận một tài liệu. Một tài liệu XML được xem như là hợp lệ nếu nội dung của nó kết nối với các phần tử, các thuộc tính và được liên kết với DTD, nếu tài liệu tuân theo các ràng buộc được trình bày trong nó. Validation được xử lý theo hai cách bởi XML Parser:

* Tài liệu XML là Well-formed:
* Không có DTD.
* Chỉ có duy nhất có một phần tử thuộc cấp cao nhất trong tài liệu, còn gọi là nút gốc (root element)
* Mỗi một thẻ mở đều phải có thẻ đóng và tên thẻ là phân biệt hoa thường
* Các thẻ khi đóng phải theo đúng trình tự (mở sau đóng trước)
* Tên thẻ không nên có khoảng trắng, không nên bắt đầu bằng “xml”
* Các thuộc tính (atributes) của một thẻ luôn luôn tồn tại theo cặp theo quy ước: <tên> = “<giá\_trị>”; không nên đặt tên thuộc tính trùng nhau, và giá trị của thuộc tính phải đặt trong cặp dấu nháy kép hay nháy đơn. Tên của thuộc tính (atribute) sẽ theo qui luật đặt tên giống như đối với tên thẻ
* Các thẻ (tag) trong XML có thể lồng nhau (Thẻ này có thể chứa nhiều thẻ khác ở bên trong)
* Tài liệu XML là hợp lệ:
* Một tài liệu XML hợp lệ là tài liệu Well-fomed có một DTD liên kết.

1. **Ứng dụng XML:**

* Mô tả cấu hình của 1 Website hay một ứng dụng thông qua các tập tin web.config, faces-config.xml, web.xml,...
* Cung cấp thông tin, dữ liệu cho các hệ thống khác nhau để có thể khai thác, sử dụng
* Cung cấp hệ thống thông tin từ các nhà máy con cho hệ thống chính quản lý.
* Ứng dụng XML mang ý nghĩa cho biết một tập các thẻ hay tập con XML hoạt động riêng trong một lĩnh vực nào đó:

- MathML: định dạng các biểu thức, kí hiệu toán học

- CML: ngôn ngữ định dạng hóa học

- CDF: khuôn dạng định nghĩa kênh

- SMIL: ngôn ngữ tích hợp multimedia đồng bộ

- XHTML: dùng mở rộng và định nghĩa lại ngôn ngữ định dạng HTML

- XUL: ngôn ngữ cấu hình giao diện người dùng

- VML: ngôn ngữ định dạng Vector

- WML: ngôn ngữ định dạng mạng không dây

- SOAP: giao thức truy cập đối tượng đơn giản.

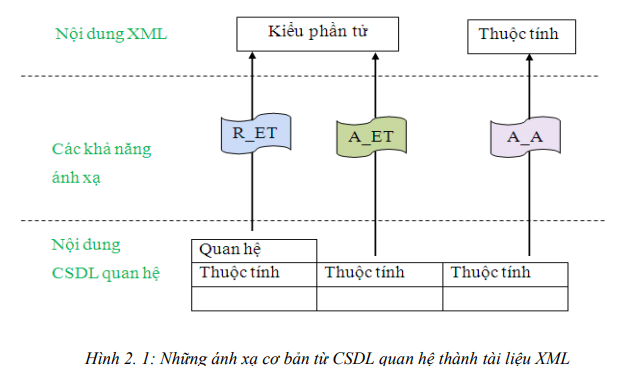
**6. Các phương pháp chuyển đổi dữ liệu quan hệ sang dữ liệu XML:**

Nhìn chung XML và CSDL là những công nghệ lấy dữ liệu làm trung tâm. XML thích hợp cho việc truyền thông dữ liệu, trong khi đó CSDL thích hợp cho việc lưu trữ và truy vấn dữ liệu.

***a/ Ánh xạ một CSDL quan hệ thành một tài liệu XML***

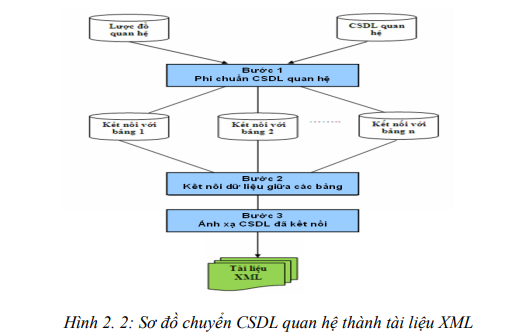
Các CSDL được kết nối lại và được ánh xạ vào trong một DOM chính, sau đó nó được ánh xạ thành một tài liệu XML.

Dựa vào cấu trúc của một tài liệu XML cũng như mối quan hệ giữa một tài liệu XML và một CSDL quan hệ chúng ta có một số ánh xạ cơ bản từ một CSDL quan hệ sang một tài liệu XML



* R\_ET: một quan hệ R được ánh xạ thành một kiểu phần tử ET.
* A\_ET: một thuộc tính của quan hệ (A) được ánh xạ thành một kiểu phần tử (ET) nhờ vào quan hệ của thuộc tính đưa ra quan hệ của kiểu phần tử.
* A\_A: thuộc tính của một quan hệ được ánh xạ thành thuộc tính XML.

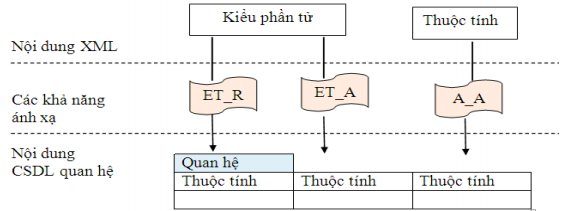
Sau khi thực hiện kết nối các bảng cơ sở dữ liệu quan hệ thành công. Các bảng cơ sở dữ liệu liên kết được hình thành và được ánh xạ thành DOM chính theo các nguyên tắc ánh xạ cơ bản và sau đó được chuyển thành một tài liệu XML tương ứng, được thể hiện qua sơ đồ sau:



***b/ Phương pháp ánh xạ một tài liệu XML thành CSDL quan hệ***

Ánh xạ một tài liệu XML sang một CSDL quan hệ được thực hiện thông qua việc ánh xạ trực tiếp các phần tử thành các thuộc tính dựa trên một số quy tắc ánh xạ sau:

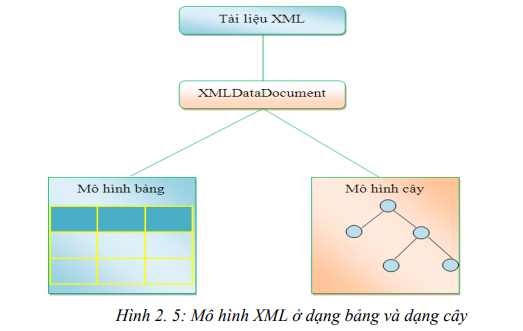
* ET\_R: một kiểu phần tử (ET) của tài liệu XML được ánh xạ thành một quan hệ R.
* ET\_A: một kiểu phần tử (ET) của tài liệu XML được ánh xạ thành một thuộc tính A của quan hệ.
* A\_A: một thuộc tính của tài liệu XML được ánh xạ thành một thuộc tính của quan hệ.



Hình 2.3: Những ánh xạ cơ bản từ tài liệu XML thành CSDL quan hệ

***c/ Thuật toán chuyển đổi giữa CSDL quan hệ và XML***

Dựa vào phương pháp luận tích hợp các hệ thống cơ sở dữ liệu quan hệ và XML, thuật toán chuyển đổi giữa các hệ thống cơ sở dữ liệu quan hệ và XML được xây dựng theo mô hình tổng quát sau:



Để xây dựng thuật toán chuyển đổi giữa CSDL quan hệ và XML ta thực hiện theo các bước sau:

* Bước 1: Biểu diễn tài liệu XML theo mô hình DOM
* Bước 2: Biểu diễn tài liệu XML theo dạng bảng của CSDL quan hệ

=> Chuyển một tài liệu XML sang một CSDL quan hệ.

*c.1/ Biểu diễn tài liệu XML theo mô hình DOM*

Một cây được tạo bởi từ nhiều nút, mỗi nút lại là một cây chứa những nút khác. Nút lá thì không có con, vì thế nút này được dùng để hiển thị văn bản.

Lớp XmlDataDocument kế thừa lớp XmlDocument vì thế nó cũng có một số phương thức giống lớp XmlDocument.

Với mô hình DOM, một tài liệu XML được tổ chức dưới dạng một sơ đồ hình cây có cấu trúc tương tự như một cây thư mục trong Windows. Vì vậy mục đích chính của phần này là biểu diễn tài liệu XML theo mô hình DOM như minh họa sau:

Document

Alan

computer scientist

1912

profession

name

born

person

people

last\_name

first\_name

Turing

<?**xml** version="1.0"?>

<**people**>

  <**person** born="1912">

       <**name**>

         <**first\_name**>Alan</**first\_name**>

         <**last\_name**>Turing</**last\_name**>

       </**name**>

       <**profession**>computer scientist</**profession**>

  </**person**>

</**people**>

Thuật toán:

Tài liệu XML được đọc thông qua hình thức đọc một tập tin dữ liệu và được xây dựng theo dạng mô hình DOM qua các bước sau:

Bước 1: Truy cập và lấy phần tử gốc của tài liệu.

Bước 2: Lấy danh sách tất cả các phần tử con của phần tử gốc.

Bước 3: Mỗi phần tử con tìm được sẽ được gán một nhãn đặc trưng.

Bước 4: Tìm các phần tử con được thực hiện theo phương pháp đệ quy.

Bước 5: Nếu phần tử đang xét là nút lá thì số các phần tử con của nó bằng 0.

Bước 6: Xây dựng và lưu trữ đường dẫn tương ứng tới mỗi phần tử con trong tài liệu XML.

Bước 7: Lấy tên các phần tử con và các giá trị thuộc tính của nó.

Bước 8: Chuyển đổi thông tin về đường dẫn, nội dung của các phần tử sang dạng chuỗi để in ra màn hình dưới dạng một cây tài liệu

*c.2/ Biểu diễn tài liệu XML theo dạng bảng của CSDL quan hệ*

Như đã trình bày ở trên, tài liệu XML là một cây (theo mô hình DOM) bao gồm các nút (node). Những nút này nội dung có thể chứa phần tử (element), dữ liệu (text), các thuộc tính (attribute),… và các nút con khác. Hình sau minh họa việc biểu diễn tài liệu XML theo dạng bảng sau:



Hình 2.25: Biểu diễn tài liệu XML theo dạng bảng

Để chuyển một tài liệu XML dưới dạng cây sang dạng bảng ta sử dụng thuật toán sau:

Thuật toán:

Bước 1: Duyệt qua tất cả các nút của cây theo mô hình DOM, bắt đầu từ nút gốc theo phương pháp đệ quy.

Bước 2: Xây dựng một hàm để kiểm tra và xử lý nút của cây tài liệu DOM. Đối với mỗi nút hiện hành mà có nút con thì hàm đệ quy tìm sâu xuống các nút bên dưới và hình thành một DOM tương ứng cho nút.

Bước 3: Với mô hình DOM thu được, thực hiện một ánh xạ các phần tử, các thuộc tính với giá trị tương ứng thành các quan hệ, thuộc tính và giá trị tương ứng của bảng CSDL quan hệ.

*c.3/ Chuyển một CSDL quan hệ sang một tài liệu XML*

Để chuyển một CSDL quan hệ thành một tài liệu XML, ta thực hiện tạo nguồn dữ liệu ODBC từ các CSDL quan hệ đã phi chuẩn (sử dụng công cụ Administrative Tools của Windows), sau đó thực hiện thuật toán sau:

Thuật toán:

Bước 1: Xây dựng lớp kết nối với ODBC để kết nối dữ liệu được xây dựng trong môi trường Microsoft Access, MySQL hay Microsoft SQL Server.

Bước 2: Tạo các kết nối trên các CSDL quan hệ cần chuyển đổi thành tài liệu XML thông qua các truy vấn SQL.

Bước 3: Chuyển đổi dữ liệu đã kết nối sang tài liệu XML thông qua các nguyên tắc ánh xạ đã trình bày.

***d/ Cài đặt thuật toán***

Thuật toán được cài đặt bằng ngôn ngữ lập trình C#.

Các CSDL quan hệ được sử dụng trong chương trình được tạo lập từ môi trường CSDL Microsoft Access hay Microsoft SQL Server. Các tài liệu XML có thể được tạo lập trong môi trường EditPlus, WordPad, NotePad, XML Editor,…

*d.1/ Đọc tài liệu XML dưới dạng DOM*

Một tài liệu XML được đọc vào chương trình được hiển thị dưới dạng một cấu trúc hình cây.

// Tạo một tài liệu XML theo mô hình DOM đã có sẵn dữ liệu

XmlDocument xmlDoc = new XmlDocument();

xmlDoc.Load(Server.MapPath("HocVien.xml"));

// Khởi tạo đối tượng điều khiển

TreeView TreeView1.Nodes.Clear();

TreeView1.Nodes.Add(new TreeNode(xmlDoc.DocumentElement.Name));

TreeNode tNode = new TreeNode(); tNode = TreeView1.Nodes[0];

// Hiển thị lên TreeView theo mô hình DOM

AddNode(xmlDoc.DocumentElement, tNode);

TreeView1.ExpandAll();

// Thêm Node

XmlNode xNode;

TreeNode tNode;

XmlNodeList xNodeList;

int i;

// Kiểm tra tập tin XML

// Thêm các Node vào TreeView trong quá trình lặp

if (xmlNode.HasChildNodes)

{

xNodeList = xmlNode.ChildNodes;

// Lấy danh sách các phần tử con của Node

for (i = 0; i <= xNodeList.Count - 1; i++)

{

xNode = xmlNode.ChildNodes[i];

treeNode.ChildNodes.Add(new TreeNode(xNode.Name));

tNode = treeNode.ChildNodes[i];

AddNode(xNode, tNode);

}

}

Else

{

// Điền giá trị lấy được ở tập tin xml vào node ở cây TreeView

treeNode.Text = (xmlNode.OuterXml).Trim();

}

*d.2/ Chuyển đổi một tài liệu XML thành CSDL quan hệ:*

Một tài liệu XML được tạo lập theo đúng cấu trúc yêu cầu sẽ được chuyển đổi và hiển thị dưới dạng một bảng CSDL quan hệ.

// Khởi tạo và tải tài liệu XML

XmlDocument xmlDocument = new XmlDocument();

xmlDocument.Load(fileName);

// Lấy danh sách các Node của tài liệu XML

XmlNodeList xmlNodeList = xmlDocument.GetElementsByTagName(nodeNa me);

// Khởi tạo đối tượng Table

Table table = new Table();

// Duyệt qua các Node của XML để thêm vào Table

foreach (XmlNode xmlNode in xmlNodeList)

{

TableRow tableRow = new TableRow();

TableCell tableCell = new TableCell();

tableCell.Text = "HocVienID";

tableCell = new TableCell();

tableCell.Wrap = false;

tableCell.BorderWidth = 1;

tableCell.Text = xmlNode.Attributes["HocVienID"].Value;

tableRow.Cells.Add(tableCell);

foreach (XmlElement xmlElement in xmlNode.ChildNodes)

{

tableCell = new TableCell();

tableCell.Wrap = false;

tableCell.BorderWidth = 1;

tableCell.Text = xmlElement.InnerText;

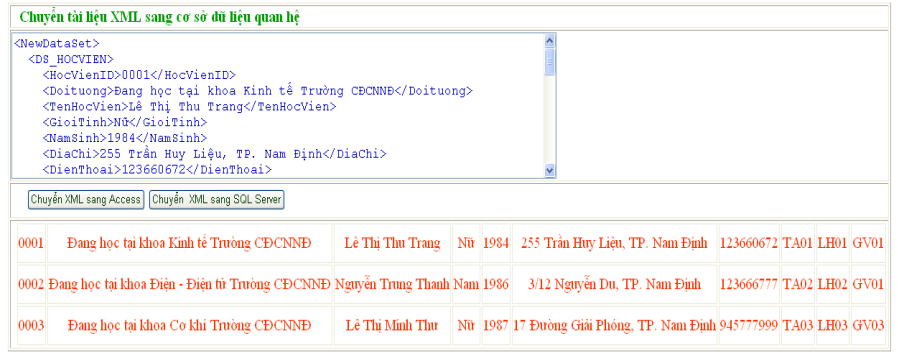
tableRow.Cells.Add(tableCell);

}

table.Rows.Add(tableRow);

}

Kết quả thực hiện thuật toán chuyển đổi tài liệu XML sang cơ sở dữ liệu quan hệ như hình minh họa sau:



*d.3/ Chuyển một CSDL quan hệ thành một tài liệu XML*

Để chuyển một CSDL quan hệ được tạo lập từ môi trường Microsoft SQL Server, MySQL, Microsoft Access,... thành một tài liệu XML. Cụ thể, chúng ta thực hiện cài đặt thử nghiệm trong các môi trường như sau:

*d.3.1) Cài đặt trên môi trường Microsoft SQL Server:*

// Chuyển CSDL được tạo trong môi trường Microsoft SQL Server sang tài liệu XML như sau:

protected void btnCSDLSQLSERVER\_Click(object sender, EventArgs e)

{

// Tạo thông tin kết nối

string strConnection = "server=lan\_ttts;uid=sa;database=quanlyhv;password=lannt ";

// Tạo thông tin và đối tượng kết nối

SqlConnection conn;

SqlDataAdapter dad;

try

{

conn = new SqlConnection(strConnection);

// Tạo câu lệnh T-SQL

string sql = "SELECT tblLICHHOC.NgayHoc, tblLOP.TenLop, tblTHOIKHOABIEU.phongHoc FROM tblLOP INNER JOIN tblTHOIKHOABIEU ON tblLOP.LopID = tblTHOIKHOABIEU.LopID INNER JOIN tblLICHHOC ON tblTHOIKHOABIEU.LichHocID = tblLICHHOC.LichHocID";

// Tạo đối tượng câu lệnh T-SQL

dad = new SqlDataAdapter(sql, conn);

conn.Open();

DataSet ds = new DataSet();

dad.Fill(ds, "DS\_LICHHOC");

// Ghi sang tập tin XML

ds.WriteXml(Server.MapPath("DS\_LICHHOC.xml");

conn.Close();

}

catch (Exception)

{

Page.Controls.Add(new LiteralControl("<script language=’javascript’> window.alert(\” Lỗi kết nối CSDL Microsoft SQL Server !\” <” + “/script>”));

}

} //Kết thúc btnCSDLSQLSERVER\_Click()

*d.3.2) Cài đặt trên môi trường MySQL*

// Chuyển CSDL được tạo trong môi trường MySQL sang tài liệu XML như sau:

protected void btnCSDLMYSQL\_Click(object sender, EventArgs e)

{

// Tạo thông tin kết nối.

string dsr = "Driver={MySQL ODBC 5.1 Driver};Server=localhost;Database=quanlyhv; uid = root; pwd=;option=3";

// Tạo câu lệnh T-SQL

string sql = "SELECT HocVienID, LopID, TenHocVien, GioiTinh, NamSinh, DiaChi, DienThoai, DoiTuong, LoaiHinhID, GiaoVien\_ID FROM tblHocVien";

// Tạo thông tin và đối tượng kết nối

try

{

OdbcConnection con = new OdbcConnection(dsr);

OdbcDataAdapter dad = new OdbcDataAdapter(sql, con);

con.Open();

DataSet ds = new DataSet();

dad.Fill(ds, "DS\_HocVien");

dg.DataSource = ds.Tables["DS\_HocVien"].DefaultView; dg.DataBind();

ds.WriteXml(Server.MapPath("DS\_HocVien.xml", XmlWriteMode.IgnoreSchema);

// Hiển thị tài liệu vào trong TextBox

TextBox1.Text = ds.GetXml();

con.Close();

}

catch (Exception)

{

Page.Controls.Add(new LiteralControl("<script language=’javascript’> window.alert(\” Lỗi kết nối CSDL MySQL !\” <” + “/script>”));

}

} //Kết thúc btnCSDLMYSQL\_Click()

*d.3.3) Cài đặt trên môi trường Microsoft Access 2003*

// Chuyển CSDL được tạo trong môi trường MySQL sang tài liệu XML như sau:

protected void btnCSDLMSA\_Click(object sender, EventArgs e)

{

// Tạo thông tin kết nối

string dsr = "Provider=Microsoft.Jet.OLEDB.4.0; data source=" + Server.MapPath("quanlyhv.mdb");

// Tạo câu lệnh T-SQL

string sql = "SELECT HocVienID, LopID, TenHocVien, GioiTinh, NamSinh, DiaChi, DienThoai, DoiTuong, LoaiHinhID, GiaoVien\_ID FROM tblHocVien";

// Tạo thông tin và đối tượng kết nối

try

{

OleDbConnection con = new OleDbConnection(dsr);

con.Open();

OleDbDataAdapter dad = new OleDbDataAdapter(sql, con);

DataSet ds = new DataSet();

dad.Fill(ds, "DS\_HocVien");

dg.DataSource = ds.Tables["DS\_HocVien"].DefaultView;

dg.DataBind();

ds.WriteXml(Server.MapPath("DS\_HocVien.xml"), XmlWriteMode.IgnoreSchema);

// Hiển thị tài liệu vào trong TextBox

TextBox1.Text = ds.GetXml();

con.Close();

}

catch (Exception)

{

Page.Controls.Add(new LiteralControl("<script language=’javascript’> window.alert(\” Lỗi kết nối CSDL Microsoft Access !\” <” + “/script>”));

}

} //Kết thúc btnCSDLMSA\_Click()

*d.3.4) Hiển thị nội dung tài liệu XML*

// Hiển thị tập tin XML vào đối tượng GridView

void showXmlData(string fileName)

{

DataSet dataSet = new DataSet();

dataSet.ReadXml(fileName);

if (dataSet != null)

{

GridView1.DataSource = dataSet.Tables[0];

GridView1.DataBind();

}

}} // Kết thúc showXmlData(string)

Kết quả thực hiện thuật toán chuyển đổi cơ sở dữ liệu quan hệ sang tài liệu XML như hình minh họa sau:

