|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ tên** | **MSSV** | **Hình ảnh** | **Ghi chú** |
| 1 | Hà Thị Phương Thảo | 0912430 | Description: 0912430 |  |
| 2 | Nguyễn Thị Thanh Thảo | 0912431 | Description: 0912431 |  |
| 3 | Trương Nguyễn Thủy Tiên | 0912463 | Description: 0912463 |  |
| 4 | Nguyễn Văn Tiến | 0912469 | Description: 0912469 | Nhóm trưởng |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Nội dung công việc** | **Thành viên 1**  **0912430** | **Thành viên 2**  **0912431** | **Thành viên 3**  **0912463** | **Thành viên 4**  **0912469** |
| 1 | Dịch tài liệu |  |  |  |  |
| 2 | Tổng hợp kết quả, viết tài liệu |  |  |  |  |

**XML AND INTERNET DATABASES**

Chương này bàn về vấn đề cơ bản trong việc truy cập và trao đổi thông tin trên Internet.

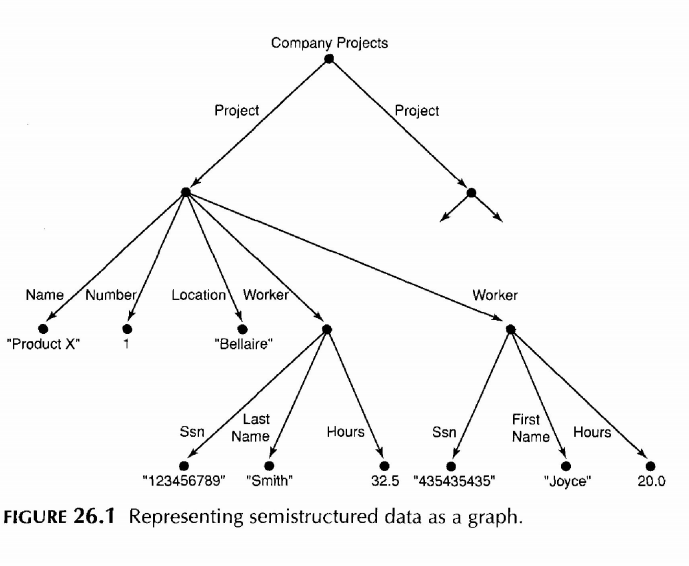
* Phần 1 nói về sự khác biệt giữa Web pages với cơ sở dữ liệu cấu trúc, và sự khác biệt giữa dữ liệu có cấu trúc, bán cấu trúc và không có cấu trúc.
* Phần 2: nói về chuẩn XML và mô hình dữ liệu cấu trúc cây.
* Phần 3: bàn về văn bản XML (XML documents) và ngôn ngữ dành cho chuyên môn hoá cấu trúc văn bản, XML DTD ( document Type Definition) và lược đồ XML (XML schema).
* Phần 4: đưa ra các hướng tiếp cận khác nhau cho việc lưu trữ văn bản XML, ở định dạng text, dạng nén, hoặc dạng quan hệ và các kiểu khác trong cơ sở dữ liệu.
* Phần 5: tổng quan về các ngôn ngữ được đề xuất trong việc truy vấn dữ liệu XML.
* Phần 6: tóm tắt nội dung của chương.

1. **Dữ liệu cấu trúc, bán cấu trúc và không có cấu trúc**

Các thông tin được lưu trữ trong cơ sở dữ liệu được xem như là dữ liệu cấu trúc vì nó được trình bày dưới 1 dạng rõ ràng. Ví dụ, mỗi record trong 1 bảng cơ sở dữ liệu quan hệ có cùng 1 dạng với nhau trong bảng. DBMS phải kiểm tra các dữ liệu phải có cấu trúc và các ràng buộc toàn vẹn trong lược đồ.

Tuy nhiên, không phải tất cả dữ liệu đều được thu thập và insert vào cơ sở dữ liệu cấu trúc được thiết kế một cách cẩn thận. ở 1 số ứng dụng, dữ liệu được thu thập vào 1 kiểu ad-hoc trước khi nó được biết nó sẽ được lưu và quản lý như thế nào. Dữ liệu này có thể có 1 cấu trúc nào đó, nhưng không phải tất cả các thông tin thu thập được sẽ có 1 cấu trúc đồng nhất. 1 số thuộc tính có thể được phân phối trong các thực thể khác nhau, nhưng các thuộc tính khác có thể chỉ tồn tại trong 1 số thực thể. Hơn nữa, các thuộc tính thêm vào có thể được giới thiểu trong 1 số các mục dữ liệu mới hơn ở bất kỳ khi nào, mà trước đó nó không được định nghĩa trong lược đồ. Kiểu dữ liệu này được biết như dữ liệu bán cấu trúc (semistructucted data). Nhiều mô hình dữ liệu được giới thiệu cho việc thể hiện dữ liệu bán cấu trúc, thường dùng cấu trúc dữ liệu dạng cây hoặc đồ thị hơn là cấu trúc mô hình quan hệ phẳng.

Sự khác biệt rõ rệt giữa dữ liệu cấu trúc và dữ liệu bán cấu trúc liên quan đến lược đồ xây dựng (như tên thuộc tính, quan hệ, kiểu thực thể) được xử lý như thế nào. ở dữ liệu bán cấu trúc, thông tin lược đồ được trộn lẫn với các giá trị dữ liệu, vì mỗi đối tương dữ liệu có thể có các thuộc tính khác nhau mà không được biết trước. trên thực tế, kiểu dữ liệu này thỉnh thoảng được nhắc đến như là dữ liệu self-describing. Xét ví dụ sau. Ta muốn thu thập 1 danh sách các tài liệu tham khảo thư mục có quan hệ với 1 đề án nghiên cứu nào đó. Trong số đó có thể là sách hoặc báo cáo chuyên môn, những cái khác có thể là các bài báo nghiên cứu hằng ngày hoặc là tập công trình nghiên cứu. Một cách rõ ràng, trong số chúng có thể có các thuộc tính khác nhau và kiểu thông tin khác nhau. Thậm chí cho. Ví dụ 1 trích dẫn của 1 bài báo có thể khá hoàn chỉnh, nó chứa đẩy đủ thông tin tên tác giả, tựa đề, công trình nghiên cứu, số trang…, ngược lại 1 số trích dẫn lại không có các thông tin đó. Các kiểu tài nguyên thư mục mới có thể xuất hiên trong tương lai ( ví dụ: tài liệu tham khảo web) và chúng có thể có các thuộc tính liên quan đến chúng.



**Dữ liệu bán cấu trúc**: ( hình 26.1), đặc biệt mô hình bán cấu trúc:

* Thông tin lược đồ (tên thuộc tính, quan hệ, và lớp (kiểu đối tượng)) trong mô hình bán cấu trúc được trộn lẫn với các đối tượng và giá trị dữ liệu của chúng trong cấu trúc dữ liệu cùng loại.
* Không có yêu cầu cho một lược đồ đã được xác định trước đó các đối tượng dữ liệu phải phù hợp với lược đồ đó.



**Dữ liệu không có cấu trúc**: loại này có hạn chế chỉ ra kiểu của đối tượng. ví dụ điển hình về dữ liệu không cấu trúc: 1 văn bản kiểu text chứa thông tin trong nó. Trang web trong HTML chứa vài dữ liệu. ta xem thành phần của tập tin HTML ở hình 26.2. dữ liệu text xuất hiện giữa 2 dấu <..>, đó được gọi là HTML tag. 1 tag có 1 dấu chéo ngược </..> cho biết kết thúc 1 tag, có nghĩa là hết hiệu lực của 1 tag bắt đấu <…> tương ứng của nó. Tag mark up dùng để chỉ cho bộ xử lý HTML làm thế nào trình các dữ liệu text ở giữa tag bắt đầu và tag kết thúc. Thực tế, các tag chỉ rõ định dạng văn bản hơn là ý nghĩa của các thành phần dữ liệu khác nhau trong văn bản. các tag trong HTML chỉ rõ thông tin, như: kích thước và kiểu , màu, heading, trong văn bản… 1 vài tag cung cấp cấu trúc text trong văn bản, ví dụ như chỉ rõ danh sách số hoặc không số hoặc 1 bảng, ngay cả khi những cấu trúc tag này chỉ ra rằng dữ liệu nguyên văn được nhúng vào thì được trình bày theo một cách nào đó, chứ không phải chỉ ra kiểu dữ liệu được trình bày trong bảng.

HTML dùng số lượng lớn các tag được định nghĩa trước mà được dùng để chỉ ra tính đa dạng của các lệnh cho định dạng văn bản web. Start tag và end tag chỉ ra vùng text được định dạng bởi các lệnh. Ví dụ các tag trong hình 26.2:

* <html>… </html> chỉ ra phạm vi của văn bản.
* Thông tin header của văn bản nằm trong tag <head>…</head> chỉ ra các lệnh sẽ được dùng ở một nơi nào đó trong văn bản. ví dụ, nó có thể chỉ ra các script functions khác nhau trong 1 ngôn ngữ (ví dụ JAVA Script hoặc PERL) hoặc các kiểu định dạng nào đó. Nó cũng có thể chỉ ra nhan đề để cho biết tập tin HTML viết về điều gì.
* Phần thân của văn bản ( đặt ở trong tag <body>…</body>) chứa các text của văn bản và đánh các tag chỉ ra các text đó được định dạng và trình bày như thế nào. Nó cũng có thể chứa tham chiếu đến các đối tượng khác, ví dụ như ảnh, video….
* Tag <H1>…</H1> chỉ ra rằng text được trình bày ở heading 1. Có rất nhiều cấp heading (<H2>, <H3>, …).
* Tag <table>…</table> chỉ ra các text được trình bày dưới dạng bảng. mỗi dòng trong bảng kèm theo tag <TR>…</TR>, và dữ liệu text được trình bày trong tag <TD>…</TD>.
* Một số tag có thể có các thuộc tính, xuất hiện trong start tag và diễn tả cấu hình thêm của tag. ở hình 26.2 tag bắt đầu <table> có 3 thuộc tính miêu tả 4 đặc diểm của bảng.

Nhược điểm HTML: tài nguyên văn bản text HTML rất khó để làm rõ một cách tự động bởi lập trình vì lược đồ không chứa thông tin loại dữ liệu trong văn bản. XML sẽ giải quyết vấn đề này.

1. **Mô hình dữ liệu phân cấp XML**

Trong văn bản XML thì đối tượng cơ bản đó là XML. 2 khái niệm cơ bản được dùng để xây dựng 1 văn bản XML: phần tử và thuộc tính. Thuộc tính trong XML cung cấp thêm các thông tin để diễn tả các phần tử đó. Có thêm các khái niệm trong XML: thực thể, định danh, tham chiếu. nhưng trước tiên ta tập trung bàn về các phần tử và thuộc tính để làm rõ bản chất cốt lõi của mô hình XML.



ở hình 26.3 là 1 ví dụ của 1 phần tử XML gọi là <projects>. ở HTML, các phần tử được xác định bởi start tag và end tag của chúng. Tên tag được chứa trong dấu ngoặc: <…>, và end tag được xác định bởi móc </..>. các phần tử phức tạp được xây dựng từ các phần tử khác mang tính phân cấp, ngược lại các phần tử đơn giản thì mang giá trị dữ liệu. điểm khác biệt chính giữa XML và HTML là tên tag XML được định nghĩa để diễn tả ý nghĩa của dữ liệu phần tử trong văn bản, chứ không phải để diễn tả các text được trình bày như thế nào. Điều này có thể làm chương trình máy tính xử lý dữ liệu XML một cách tự động.

không khó để nhận thấy sự tương xứng giữa phép biểu diễn dữ liệu XML trong hình 26.3 với hình cấu trúc cây trong hình 26.1. ở cấy trúc hình cây, các node nội bộ tượng trưng cho các phần tử phức tạp, ngược lại các node lá tượng trưng cho các phần tử đơn giản. đó là lý do vi sao mô hình XML được gọi là mô hình cây hoặc mô hình phân cấp. trong hình 26.3, các phần tử đơn giản là tên tag <Name>, <number>, <Location>, <DeptNo>, <SSN>, <LastName>, <FirstName>, và <hours>. Các phần tử phức tạp là các tên tag: <projects>, <project>, và <Worker>. Nhìn chung, không có giới hạn nào về cấp bậc cho các phần tử.

nhìn chung, có 3 kiểu XML văn bản chính:

* Data-centric XML documents; các văn bản loại này có nhiều mục dữ liệu nhỏ theo một cấu trúc đặc trưng, do đó nó có thể được khai triển từ 1 cơ sở dữ liệu cấu trúc. Chúng được định dạng như văn bản XML để thay đồi hoặc trình bày chúng trên web.
* Document-centric XML documents: những văn bản này dùng khi có lượng lớn text, ví dụ như báo, sách. Các thành phần dữ liệu cấu trúc không có hoặc có rất ít trong văn bản loại này.
* Hybrid XML documents: loại văn bản này có thể có nhiều thành phần chứa dữ liệu có cấu trúc và các thành phần khác phần lớn chứa dữ liệu text nguyên bản, hoặc không có cấu trúc.

Lưu ý: văn bản XML data-centric có thể được xem như là văn bản bán cấu trúc, hoặc như là có cấu trúc. Nếu 1 văn bản XML đúng với 1 lược đồ XML hoặc DTD đã định nghĩa sẵn, thì văn bản đó có thể được xem như 1 dữ liệu có cấu trúc. Mặt khác, XML chấp nhận các văn bản mà không phù hợp với bất cứ lược đồ nào; và chúng được xem như là dữ liệu bán cấu trúc. Thứ 2, nó cũng được xem như là văn bản chemaless XML.

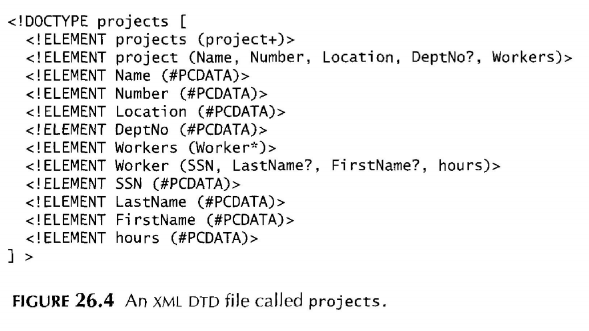
1. **Văn bản XML, DTD, và lược đồ XML**
2. Văn bản XML và XML DTD đúng đắn và có giá trị.



Văn bản XML đơn giản giống như trong hình 26.3. 1 văn bản XML đúng đắn nếu nó thoả một số điều kiện. nó phải bắt đầu với 1 lời khai báo XML (XML declaration) đề cho biết phiên bản XML đang dùng, nó thể hiện ở dòng đầu tiên trong hình 26.3. nó phải theo nguyên tắc cú pháp của quy tắc ở mô hình câu. Điều này có nghĩa là nên có 1 single root element, và mỗi phần tử phải chứa 1 cặp tag bắt đầu và tag kết thúc (start tag , end tag) của phần tử cha. Điều này chắc chắn rằng các phần tử được lồng vào nhau, chỉ rõ 1 cấu trúc cây đúng đắn.

1 văn bản XML đúng đắn phải đúng cú pháp. Điều này cho phép nó được xử lý bởi bộ xử lý chung và tạo ra 1 cây tượng trưng. 1 bộ các hàm API (application programming interface) chuẩn được gọi là DOM (Document Object Model) cho phép chương trình điểu khiển cây tượng trưng tương ứng với văn bản XML đúng đắn. tuy nhiên toàn bộ văn bản phải được phân tích ngữ pháp trước khi xử lý khi dùng DOM. 1 API khác được gọi là SAX cho phép xử lý văn bản XML trên chương trình xử lý nhanh cống bất cứ khi nào tag bắt đầu hoặc tag kết thúc được thấy. điều này làm cho dễ dàng xử lý văn bản lớn.

1 văn bản XML đúng đắn có thể có bất cứ tên tag nào cho phần tử bên trong văn bản. không có tập hợp phần tử ( tên tag) nào được định nghĩa trước mà chương trình xử lý văn bản chờ đợi. điều này cho người tạo văn bản thoải mái khi tạo 1 phần tử mới, nhưng có khả năng gây hạn chế cho thông dịch tự động các phận tử bên trong văn bản.



Một tiêu chuẩn quan trọng trong văn bản XML đó là có giá trị. Trong trường hợp này, văn bản phải được tạo ra một cách tốt. khi tạo phần tử, phải chú ý đến:

* Ký tự \* theo sau 1 tên phần tử có nghĩa là phần tử đó có thể được lập lại 0 hoặc nhiều lần trong văn bản. loại phần tử này được biết như là 1 optional multivalued element.
* Ký tự + theo sau 1 tên phần tử nghĩa là phần tử đó có thể lập lại 1 hoặc nhiều lần trong văn bản. loại này được biết như là 1 required multivalued element.
* Ký tự ? theo sau tên phần tử nghĩa là phần tử đó có thể được lập lại 0 hoặc 1 lần. laoị này là 1 optional single-valued element.
* 1 phần tử xuất hiện không có bất cứ ký tự nào đứng trước thì phải xuất hiện 1 lần trong văn bản. loại này là required single-valued element.
* Kiểu của phần tử được chỉ rõ qua dấu ngoặc đơn sau phần tử. nếu ngoặc đơn chứa tên của phần tử khác thì những phần tử này là con của nó. Nếu ngoặc đơn chứa từ khoá #PCDATA hoặc 1 trong các kiểu dữ liệu khác có trong XML, DTD, thì các phần tử là 1 node lá.
* Dấu ngoặc đơn được lồng vào khi đang chỉ các phần tử.
* Ký tự (e1 | e2) chỉ ra rằng e1 hoặc e2 có thể xuất hiện trong văn bản.

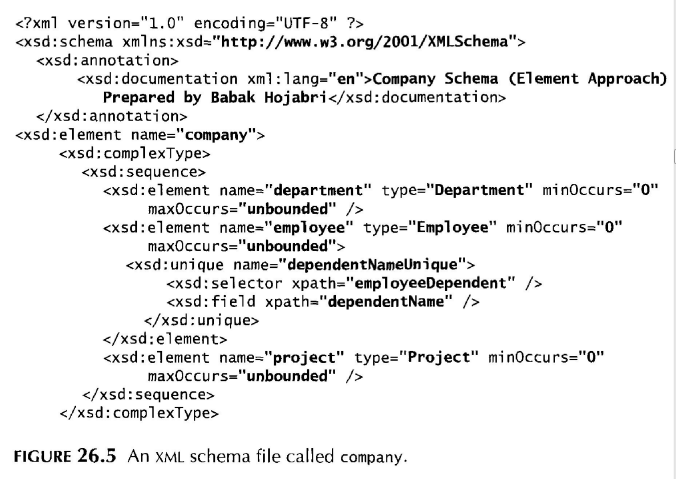
Ta có thể thấy cấu trúc cây torng hình 26.1 và văn bản XML trong hình 26.3 thích hợp với XML DTD trong hình 26.4. đề yêu cầu 1 văn bản XML được kiểm tra phù hợp với 1 DTD thì ta phải chỉ ra nó trong khai báo văn bản. ví dụ ta muốn thay đổi dòng 1 trong hình 26.3 thì làm như sau:



Khi giá trị thuộc tính standlone bằng “no” thì văn bản cần được kiểm tra dựa vào 1 văn bản DTD riêng lẻ. tập tin DTD trong hình 26.4 được lưu trong file system cùng chỗ với văn bản XML, và được đặt tên là “proj.dtd”. như một sự lựa chọn , ta có thể chứa văn bản text DTD ở phần đầu trong văn bảng XML của nó để kiểm tra cũng được.

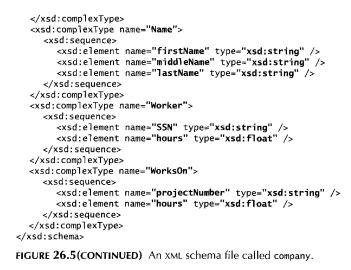
Mặc dù XML DTD hoàn toàn phù hợp cho việc chỉ định cấu trúc cây với các phần tử required, optional và repeating, thì nó có 1 vài giớn hạn. thứ 1 đó là, kiểu dữ liệu trong DTD thì không tổng quát. Thứ 2, DTD có cú pháp của riêng nó và yêu cầu xử lý đặc biệt. thứ 3, tất cả phần tử DTD luôn phải bắt theo yêu cầu được chỉ định của văn bản XML, các phần tử không được ra lệnh thì không được phép.

1. Lược đồ XML.









Ngôn ngữ lược đồ XML là 1 chuẩn cho việc chỉ định 1 cấu trúc văn bản XML. Nó dùng các nguyên tắc cú pháp giống như các văn bản XML thông thường, vì thế các bộ vi xử lý giống nhau có thể dùng cả 2. Để phân biệt 2 kiểu văn bản, ta sẽ dùng thuật ngữ văn bản minh hoạ XML (XML instance document) hoặc văn bản XML ( XML document) cho 1 văn bản XML thông thường, và văn bản lược đồ XML (XML schema document) cho 1 văn bản chỉ rõ lược đồ XML. Hình 26.5 thể hiện 1 văn bản lược đồ XML. Lược đồ trong hình 26.5 có thể phục vụ cho mục đích trong việc chỉ rõ cấu trúc của cơ sở dữ liệu COMPANY nếu nó được lưu trong hệ thống XML. Với lược đồ XML DTD được dựa trên mô hình dữ liệu cây thì các phần tử và thuộc tính giống như các khái niệm cấu trúc chính.tuy nhiên nó mượn thêm các khái niệm từ mô hình cơ sở dữ liệu và đối tượng, ví dụ như khoá, tham chiếu và định danh. ở đây ta miêu tả tính năng của lược đồ XML theo từng cách 1, việc tham khảo ví dụ văn bản lược đồ XML trong hình 26.5 là 1 minh hoạ. Ta giới thiệu và miều tả 1 số khái niệm của lược đồ để sử dụng chúng trong hình 26.5:

* Diễn tả lược đồ và XML namespaces: nó rất cần thiết khi xác định bộ các thành phần ngôn ngữ lược đồ XML được dùng cho việc chỉ định 1 tập tin được lưu ở 1 vị trí trong website. Dòng thứ 2 trong hình 26.5 chỉ ra tập tin được dùng trong ví dụ này, đó là “http:www.w3.org/2001/XMLSchema”. Đây là chuẩn chung nhất cho lược đồ các lệnh XML. Mỗi định nghĩa được gọi là 1 XML namespace, vì nó định nghĩa 1 bộ các lệnh có thể được dùng. Tên tập tin được đánh với biến xsd (CML schema description) dùng thuộc tính xmlns và biến này được dùng như 1 tiền tố với tất cả các lệnh . ví dụ: xsd: sequence.
* Chú giải, tra cứu và ngôn ngữ được dùng: 2 dòng kế tiếp trong hình 26.5 mình hoạ lược đồ các phần tử XML (tag) xsd:annotation và xsd:documentation, những cái này được dùng để ghi chú và miệu tả khác trong văn bản XML.
* Phần tử và kiểu: ta xác định phần tử gốc (root element) của lược đồ XML. Trong lược đồ XML, tên thuộc tính của tag xsd: element chỉ ra tên thuộc tính của phẩn từ gốc, trong hình 26.5 là company.
* Phần tử cấp 1 trong cơ sở dữ liệu company: có 3 cấp phần tử dưới phần tử gốc company trong hình 26.5. các phần tử này có tên: employee, department, và project. Mỗi phần tử được chỉ ra trong tag xsd: element. Chú ý: nếu 1 tag chỉ có 1 thuộc tính và không có phần tử con nào hoặc không có dữ liệu trong nó thì nó có thể được kết thúc bởi dấu ngoặc có ký tự (/>) thay vì phải có 1 tag kết thúc. Những phần tử này được gọi là phần tử rỗng (empty elements). Ví dụ: department.
* Chỉ rõ kiểu phần tử và sự kiện lớn nhất và nhỏ nhất: trong lược đồ XML, các thuộc tính type, minOccurs, và maxOccurs trong tag xsd:element chỉ ra kiểu và bội số của mỗi phần tử torng bất cứ văn bản nào mà thích hợp với lược đồ chi tiết hoá này. Nếu ta chỉ ra kiểu thuộc tính trong 1 xsd:element, thì cấu trúc của thuộc tính đó phải được diễn tả một cách riêng biệt, điển hình là dùng phần tử xsd:comlexType của lược đồ XML. Điều này được minh hoạ bởi các phần tử employee, department, và project torng hình 26.5. mặt khác, nếu không chỉ định kiểu thuộc tính thì cấu trúc các phần tử có thể được định nghĩa trực tiếp theo tag, như minh hoạ phần tử gốc company trong hình 26.5. tag minOccurs và maxOccurs được dùng để chỉ ra giới hạn số lượng sự kiện cửa 1 phần tử trong văn bản mà thích hợp với lược đồ chi tiết. nếu chúng không được chỉ định thì mặc định có 1 sự kiện. cái này giống với vai trò của các ký hiệu \*, +, ? trong XML DTD và giống với giá trị (min, max) trong ràng buộc của mô hình ER.
* Chỉ định khoá: trong lược đồ XML, có thể chỉ định các ràng buộc tương ứng với ràng buộc unique, khoá chính trong cơ sở dữ liệu quan hệ. tag xsd:unique chỉ ra các phần tử mà tương ứng với các thuộc tính unique trong cơ sở dữ liệu quan hệ mà nó không phải là khoá chính. Ta có thể gán cho mỗi ràng buộc duy nhất 1 tên và ta phải chỉ rõ tag xsd:selector và xsd: field cho nó để xác định kiểu phần tử mà chứa phần tử unique này và tên phần tử trong nó mà là unique bằng thuộc tính xpath. Ví dụ cho unique là phần tử departmentNameUnique và projectNameUnique trong hình 26.5. để xác định khoá chính, tag xsd:key được dùng thay vì dùng tag xsd:unique, ví dụ minh hoạ là các phần tử projectNumberKey, departemntNumberKey, và employeeSSNKey trong hình 26.5. để chỉ ra khoá ngoại thì tag xsd:keyref làm nhiệm vụ này, ví dụ trong hình 26.5 có 6 phần tử xsd:keyref.
* Chỉ định cấu trúc cho phần tử phức tạp và kiểu phức tạp: các phần tử phức tạp trong hình 26.5 đó là Department, Employee, Project, và Dependent, chúng dùng tag xsd;complexType để xác định phần tử phức tạp.mỗi phần tử phức tạp này ta chỉ định 1 dãy các thành phần con giống như các thuộc tính trong thực thể bằng cách dùng tag xsd:sequence và xsd:element trong lược đồ XML. Ta cũng cố thể xác định minOccurs và maxOccurs nếu ta cần thay đồi giá trị sự kiện mặc định của nó. Nếu các thuộc tính trong cơ sở dữ liệu được cho phép giá trị null , nên ta cần chỉ rõ giá trị minOccurs = 0, ngược lại các thuộc tính đa giá trị thì đặt maxOccurs = unbounded.
* Thuộc tính phức hợp: thuộc tính phức hợp giống như các kiệu phức tạp trong hình 26.5, ví dụ: Address, Name, Worker, và kiểu Workson phức tạp. những thuộc tính này được nhứng trong phần tử cha của chúng.

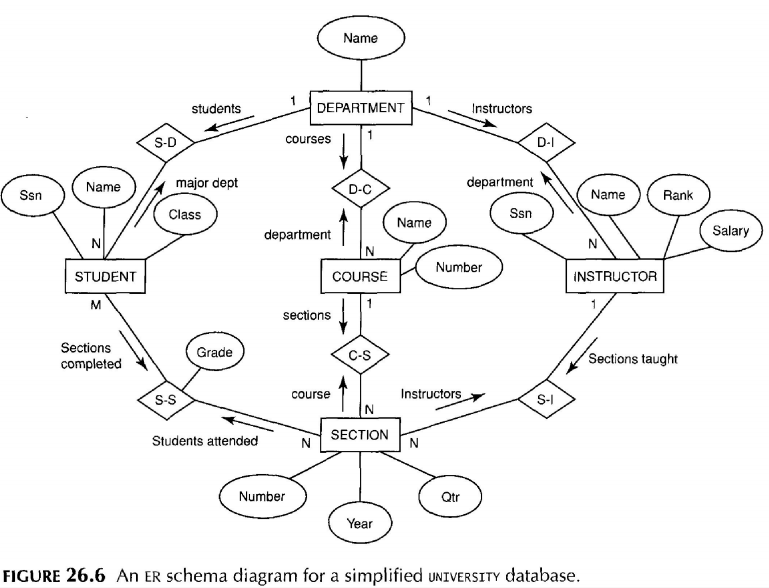
1. **Văn bản XML và cơ sở dữ liệu**
2. Lưu trữ văn bản XML

Có nhiều hướng tổ chức các nội dung của văn bản XML, dưới đây là 1 số các:

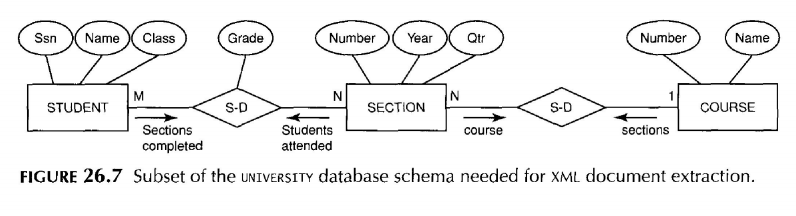
* Dùng 1 hệ quản trị để lưu trữ văn bản dưới dạng text: 1 hệ quản trị đối tượng hoặc hệ quản trị quan hệ có thể được dùng để lưu toàn bộ văn bản XML dưới dạng các trường text bên trong các record hoặc các đối tượng của hệ quản trị.
* Dùng hệ quản trị để lưu nội dung văn bản như các phần tử dữ liệu: cách này sẽ lưu 1 collection của văn bản mà theo 1 lược đồ XML DTD hoặc XML đặc biệt. Vì tất cả văn bản có cùng cấu trúc , nên có thể thiết kế cơ sở dữ liệu quan hệ để lưu trử các phần tử dữ liệu ở cấp lá trong văn bản XML. Cách này yêu cấu có thuật toán ánh xạ để thiết kế lược đồ cơ sở dữ liệu tương thích với cấu trúc của lược đồ XML như lược đồ XML , hoặc DTD của văn bản XML đó.
* Thiết kế 1 hệ thống đặc biệt để lưu trữ các dữ liệu XML thuần: hệ thống cơ sở dữ liệu dựa trên mô hình phân cấp được thiết kế và thi hành.hệ thống này có thể chứa các công nghệ chị mục và thiết kế, và có thể làm việc trên tất cả các loại văn bản XML.
* Tạo văn bản XML đặc chế từ cơ sở dữ liệu quan hệ tồn tại trước: vì đã có một lượng lớn dữ liệu đã có sẵn trong cơ sở dữ liệu quan hệ, nên các thành phần của dữ liệu này có thể cần được định dạng dưới dạng văn bản để trao đổi hoặc trình bày trên web. Cách này dùng 1 lớp phần mêm trung gian để xử lý các sự chuyển đổi qua lại giữa cơ sở dữ liệu quan hệ với văn bản XML.

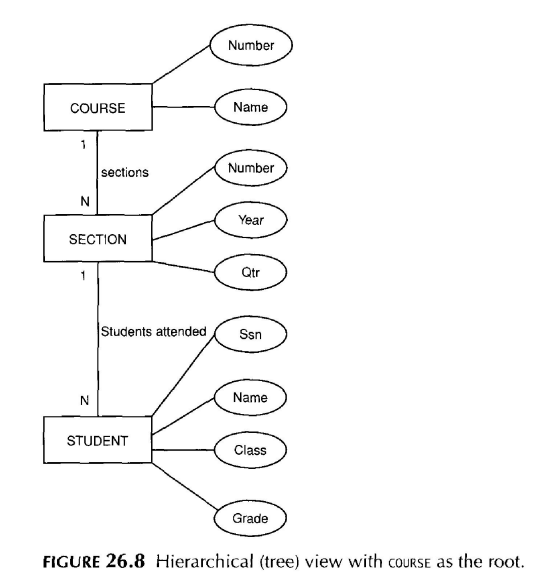
1. Khai triển văn bản XML từ cơ sở dữ liệu quan hệ

Phần này bàn đề các vấn đề gặp phải khi biến đổi từ cơ sở dữ liệu quan hệ sang văn bản XML.

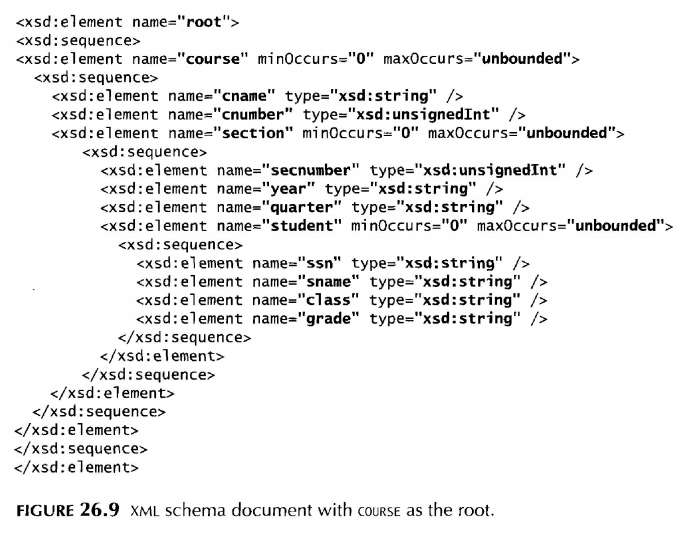


Ta sẽ dùng lược đồ ER UNIVERSITY trong hình 26.6 để minh hoạ vấn đề của chúng ta. 1 ứng dụng cần khai triển văn bản XML cho thông tin student, course, và grade trong cơ sở dữ liệu UNIVERSITY. Dữ liệu cần các văn bản này được chứ trong thuộc tính của thực thể kiểu COURSE, SECTION VÀ STUDENT trong cơ sở dữ liệu ở hình 26.6, và các mối quan hệ s-s và c-s giữa chúng. Thường hầu hết các văn bản được khai triển từ 1 cơ sở dữ liệu chỉ dùng 1 tập con của các thuộc tính, kiểu thực thể và các quan hệ trong cơ sở dữ liệu. ví dụ, trong hình 26.7:





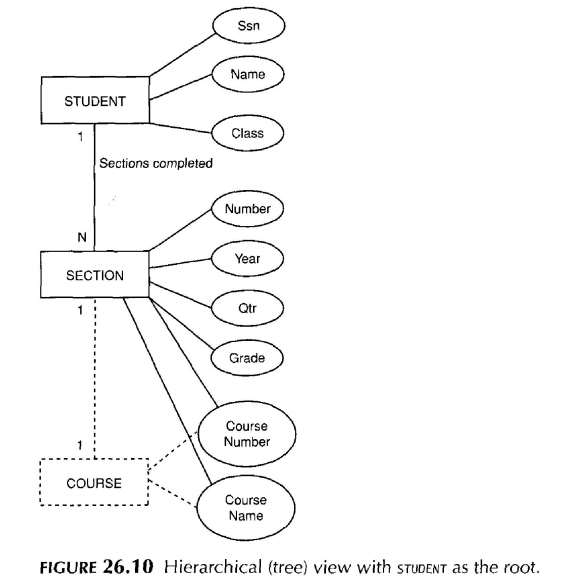
Có ít nhất 3 phân cấp văn bản được khai triển từ bộ con của cơ sở dữ liệu trong hình 26.7. trước tiên, ta có thể cho COURSE là gốc (trong hình 26.8). ở đây, mỗi thực thể course có tập hợp các section của nó như là các thành phần con, và mỗi section có các student như là các phần tử con. Ta có thể thấy kết quả của việc mô hình hoá thông tin trong 1 cấu trúc cây phân cấp. nếu 1 studen có nhiều section, thì thông tin của student đó sẽ xuất hiện nhiều lần trong văn bản ( mỗi 1 section là 1 lần). 1 lược đồ XML đơn giản diễn tả hình 26.8 là hình 26.9:

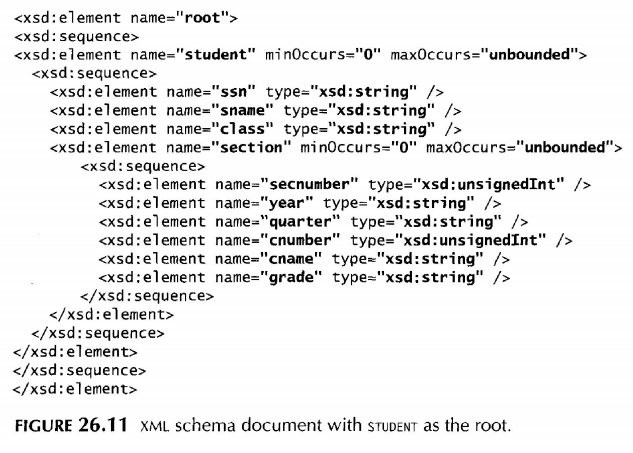


Vì STUDENT trở thành con của SECTION trong cây phân cấp, vì thế mỗi phần tử STUDENT được xếp dưới phần tử SECTION.

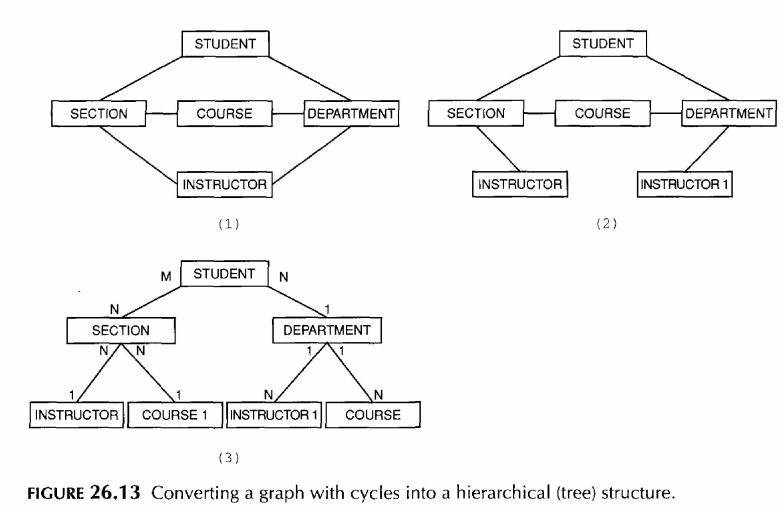
Trong cây phần cấp thứ 2 , ta có thể chọn STUDENT làm gốc (hình 26.10). trong cây này, mỗi student có 1 tập các section như là thành phần con của nó, và mỗi section quan hệ với 1 course như là con của nó, do quan hệ giữa SECTION với COURSE là n:1. Ta có thể gộp phần tử COURSE và SECTION.

1. Xoá vòng để biến đồi sơ đồ thành cây.





Trong 1 số trường hợp các bộ con có thể có nhiều vòng. Trong trường hợp này, rất phức tạp trong tạo cây phân cấp. ta lấy ví dụ lược đồ ER trong hình 26.6.Với phần tử gốc STUDENT cần thông tin ở tất cả các kiểu thực thể và quan hệ của hình 26.6 trong văn bản XML.



Hình 26.13 minh hoạ 1 cấu trúc cây được tạo thành cho văn bản này như thế nào. Trong hình (1) 26.13 nó không phải là cấu trúc cây vì có vòng. 1 cách để phá vỡ vòng này là lặp lại kiểu thực thể gây ra vòng. Trước tiên, ta tái tạo 1 bản sao INSTRUCTOR như trong hình (2), đặt là INSTRUCTOR1. Sau đó tạo COURSE như torng hình (3).

1. Các bước khác cho việc khai triển văn bản XML tử cơ sở dữ liệu

Để tạo thêm 1 cây phân cấp XML thích hợp và tương ứng với lược đồ XML, các bước khác cần khi khai triển:

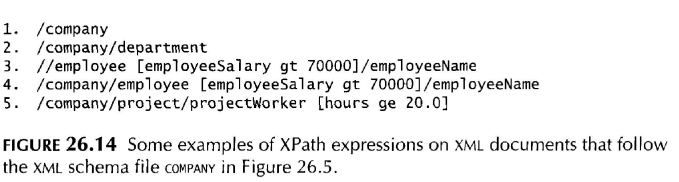
* Viết truy vấn SQL chính xác để thể hiện đúng trong văn bản XML.
* 1 lần truy vấn được thực thi thì kết quả của nó phải được cấu trúc hoá từ dạng quan hệ phẳng sang cấu trúc cấy XML.
* Câu truy vấn có thể được đặc chế thành select 1 đối tượng đơn hoặc đối tượng đa.

1. **Truy vấn XML**

Có một số đề xuất cho ngôn ngữ truy vấn XML, nhưng có 2 chuẩn nổi bật. 1 là Xpath và Xquery.

1. Xpath: chỉ định đường đi các lệnh trong XML

1 lệnh Xpath trả về tập hợp các node phần tử thoả mãn các mẫu nào đó trong lệnh. Tên của lênh Xpath là tên node trong cây văn bàn XML (tên tag hoặc tên thuộc tính). Có 2 ký hiệu gạch chéo chính trong path: / và //. Gạch / trước 1 tag cho biết tag đó phải xuất hiện như 1 con của tag trước, ngược lại gạch // chỉ rằng tag có thể xuất hiện như 1 node cháu của tag trước. ví dụ 26.14.



Lệnh đầu tiên trong hình 26.14 trả về nod gốc company và tất cả các node cháu của nó, có nghĩa là trả về toàn bộ văn bản XML.

Lệnh thứ 2 trả về tất cả node department và các cây con của nó.

Lệnh 3 dùng // thì thuận lợi cho việc dùng nếu ta không biết tên path đầy đủ mà ta cần tìm, nhưng biết tên của 1 số tag trong văn bản XML. Câu lệnh trả vể tất cả node employeeName mà là node con trực tiếp của 1 node employee, node employee có các node con có employeeSalary lớn hơn 70000.

Lệnh 4 trả về cùng kết quả như câu 3 nhưng biết đầy đủ tên path. Lệnh 5 trả về các node projectWorker và node cháu của nó dưới path /company/project và có 1 node con có thời gian > 20 giờ.

1. Xquery

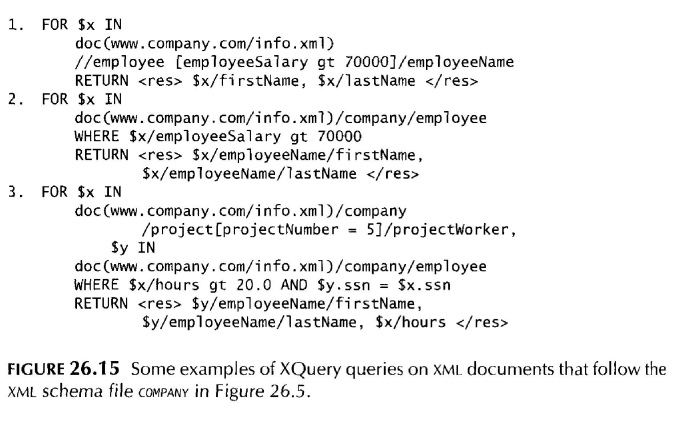
Xquery cho phép nhiều câu truy vấn chung trên nhiều văn bản XML. Dạng chung của truy vấn Xquery được biết như 1 lệnh FLWR, có 4 mệnh đề chính:

FOR <biến liên kết với 1 node (phần tử)>

LET<biến liên kết với tập hợp các node (phần tử)>

WHERE <điều kiện>

RETURN <giá trị trả về>



1. **Kết luận**

Chương này cho ta cái nhìn tổng quan về chuẩn trình bày và trao đổi dữ liệu trên Internet. Ta bàn luận về sự khác biệt giữa dữ liệu cấu trúc, bán cấu trúc, và không có cấu trúc. Và ta đã bàn vì sao cần phải có ngôn ngữ XML. Ta đã trình bày các dạng chuẩn XML và mô hình dữ liệu cấu trúc cây của nó. Và đi qua tổng quan về cách lưu trữ XML, chuyển đổi từ mô hình quan hệ. và cuối cùng là câu truy vấn XML.