|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 |  |  |  |  |

### Topic #9a – XML và cơ sở dữ liệu Internets

Mục lục

[I. Dữ liệu cấu trúc, bán cấu trúc và không có cấu trúc 4](#_Toc309861610)

[II. Mô hình dữ liệu phân cấp XML.](#_Toc309861611) 8

[III. Văn bản XML, DTD, và lược đồ XML 10](#_Toc309861617)

[1. Văn bản XML và XML DTD đúng đắn và có giá trị 1](#_Toc309861619)0

[2. Lược đồ XML. 1](#_Toc309861620)3

[IV. Văn bản XML và cơ sở dữ liệu. 19](#_Toc309861618)

[1. Lưu trữ văn bản XML 19](#_Toc309861619)

[2. Khai triển văn bản XML từ cơ sở dữ liệu quan hệ. 20](#_Toc309861620)

[3. Xoá vòng để biến đồi sơ đồ thành cây.](#_Toc309861619) 23

[4. Các bước khác cho việc khai triển văn bản XML tử cơ sở dữ liệu. 26](#_Toc309861620)

[V. Truy vấn XML. 26](#_Toc309861621)

[1. Xpath: chỉ định đường đi các lệnh trong XML. 27](#_Toc309861622)

[2. Xquery. 27](#_Toc309861623)

[3. Kết luận. 28](#_Toc309861624)

**XML AND INTERNET DATABASES**

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

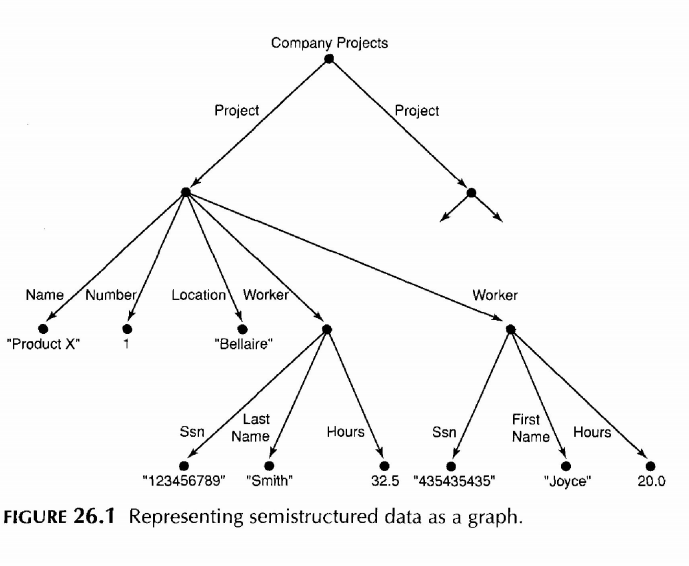
* Phần 1 nói về sự khác biệt giữa Web pages với cơ sở dữ liệu cấu trúc, và sự khác biệt giữa dữ liệu có cấu trúc, bán cấu trúc và không có cấu trúc.
* Phần 2: nói về chuẩn XML và mô hình dữ liệu cấu trúc cây.
* Phần 3: bàn về văn bản XML (XML documents) và ngôn ngữ dành cho chuyên môn hoá cấu trúc văn 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.

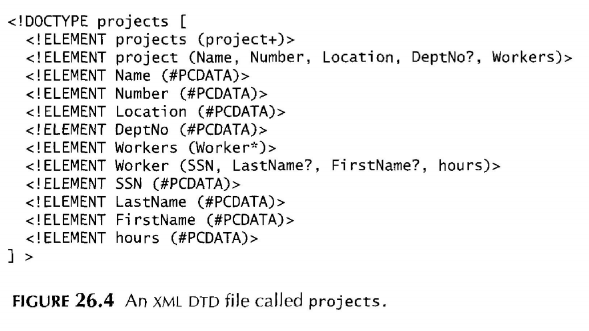
1. **Văn bản XML, DTD, và lược đồ XML**
2. Văn bản XML và XML DTD đúng đắn và có giá trị.



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.

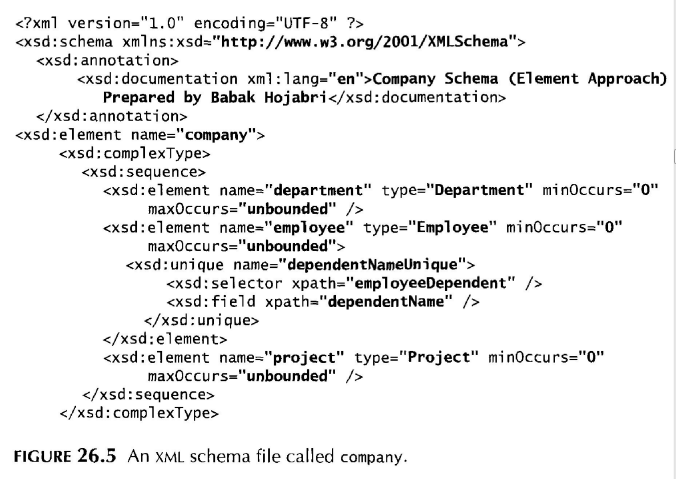
Ta có thể thấy cấu trúc cây torng hình 26.1 và văn bản XML trong hình 26.3 thích hợp với XML DTD trong hình 26.4. đề yêu cầu 1 văn bản XML được kiểm tra phù hợp với 1 DTD thì ta phải chỉ ra nó trong khai báo văn bản. ví dụ ta muốn thay đổi dòng 1 trong hình 26.3 thì làm như sau:



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.

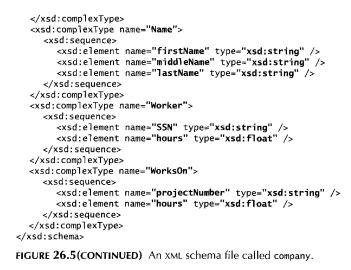
Mặc dù XML DTD hoàn toàn phù hợp cho việc chỉ định cấu trúc cây với các phần tử required, optional và repeating, thì nó có 1 vài giớn hạn. thứ 1 đó là, kiểu dữ liệu trong DTD thì không tổng quát. Thứ 2, DTD có cú pháp của riêng nó và yêu cầu xử lý đặc biệt. thứ 3, tất cả phần tử DTD luôn phải bắt theo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

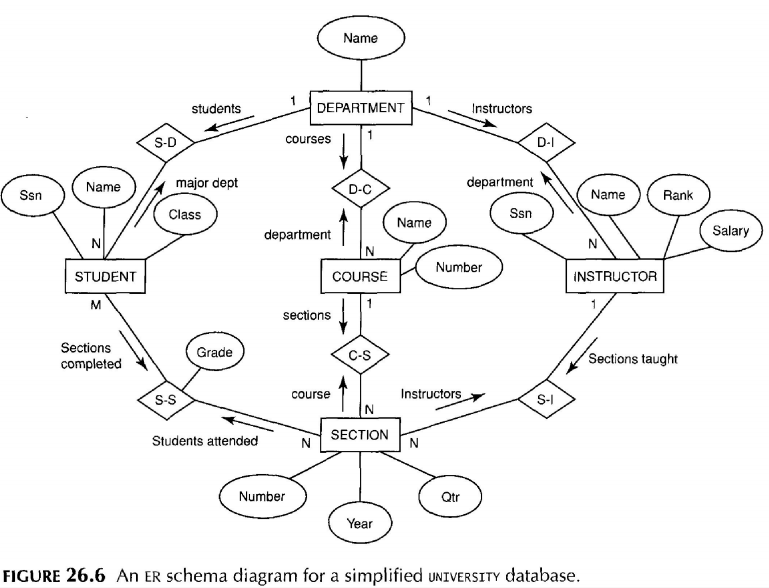
1. **Văn bản XML và cơ sở dữ liệu**
2. Lưu trữ văn bản XML

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

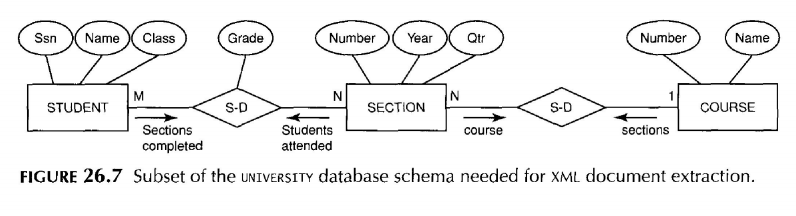
* Dùng 1 hệ quản trị để lưu trữ văn bản dưới dạng text: 1 hệ quản trị đối tượng hoặc hệ quản trị quan hệ có thể được dùng để lưu toàn bộ văn bản XML dưới dạng các trường text bên trong các record hoặc các đối tượng của hệ quản trị.
* Dùng hệ quản trị để lưu nội dung văn bản như các phần tử dữ liệu: cách 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.

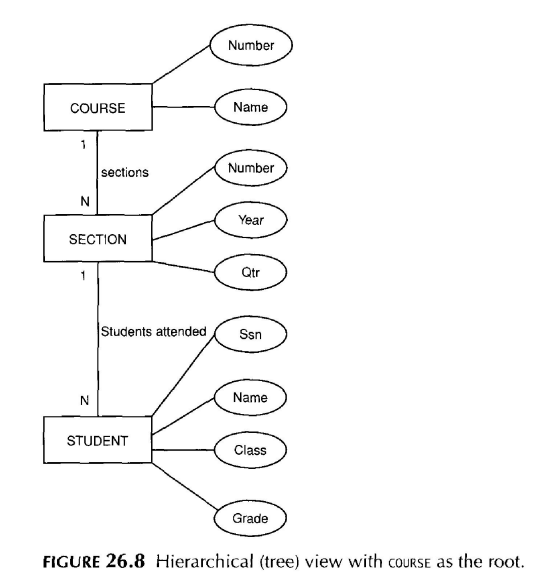
1. Khai triển văn bản XML từ cơ sở dữ liệu quan hệ

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

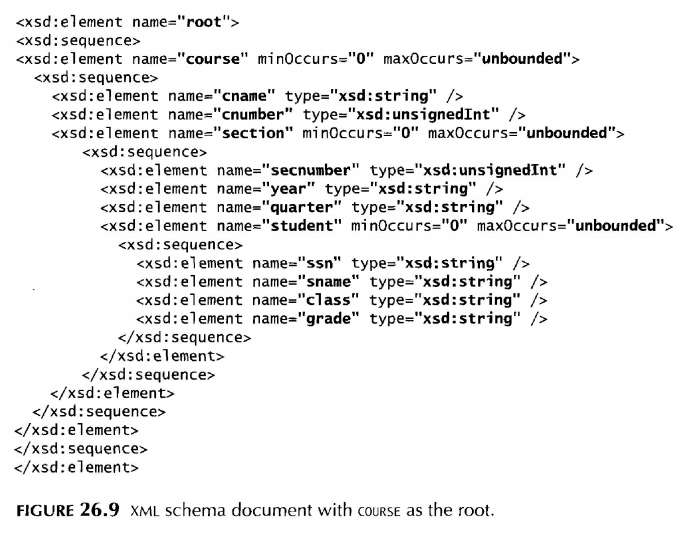


Ta sẽ dùng lược đồ ER UNIVERSITY trong hình 26.6 để minh hoạ vấn đề của chúng ta. 1 ứng dụng cần khai triển văn bản XML cho thông tin student, course, và grade trong cơ sở dữ liệu UNIVERSITY. Dữ liệu cần các văn bản này được chứ trong thuộc tính của thực thể kiểu COURSE, SECTION VÀ STUDENT trong cơ sở 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:





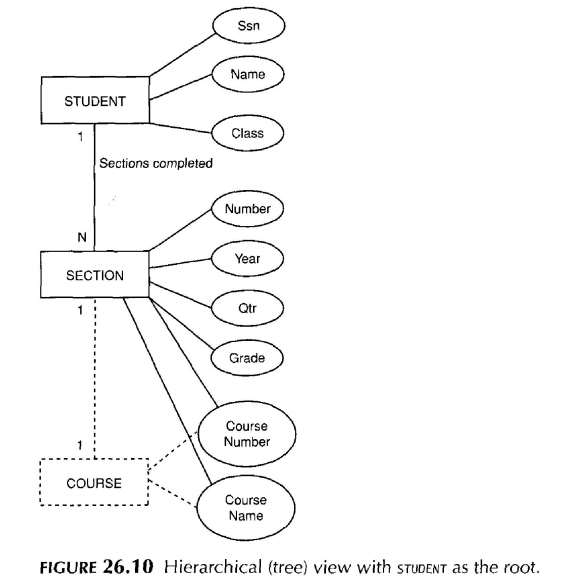
Có ít nhất 3 phân cấp văn bản được khai triển từ bộ con của cơ sở dữ liệu trong hình 26.7. trước tiên, ta có thể cho COURSE là gốc (trong hình 26.8). ở đây, mỗi thực thể course có tập hợp các section của nó như là các thành phần con, và mỗi section có các student như là các phần tử con. Ta có thể thấy 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:

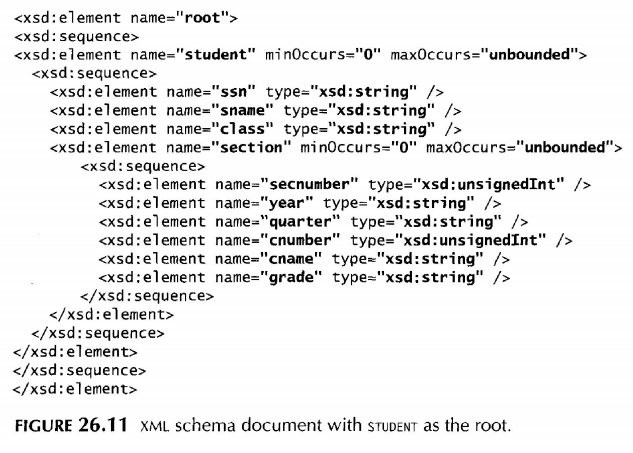


Vì STUDENT trở thành con của SECTION trong cây phân cấp, vì thế mỗi phần tử STUDENT được xếp dưới phần tử SECTION.

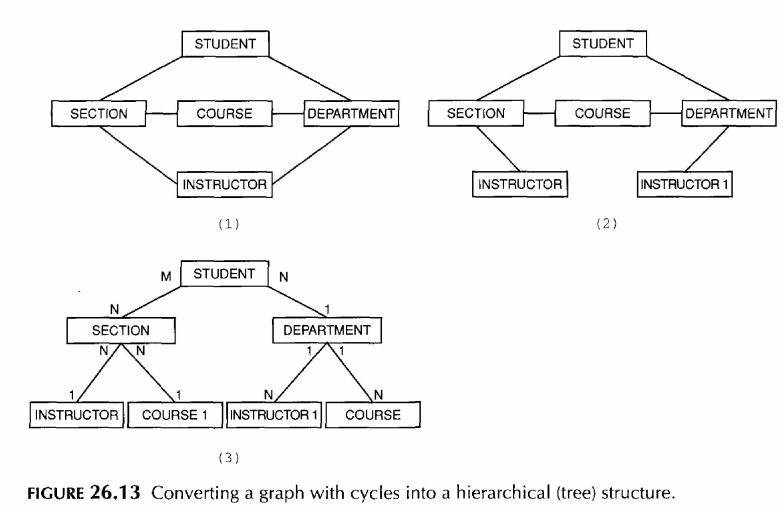
Trong cây phần cấp thứ 2 , ta có thể chọn STUDENT làm gốc (hình 26.10). trong cây này, mỗi student có 1 tập các section như là thành phần con của nó, và mỗi section quan hệ với 1 course như là con của nó, do quan hệ giữa SECTION với COURSE là n:1. Ta có thể gộp phần tử COURSE và SECTION.

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:

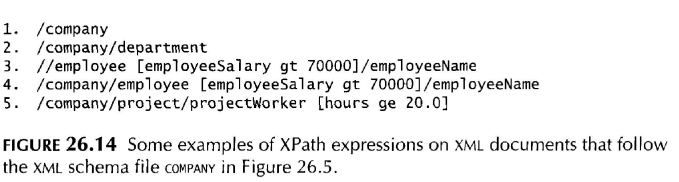
* Viết truy vấn SQL chính xác để thể hiện đúng trong văn bản XML.
* 1 lần truy vấn được thực thi thì kết quả của nó phải được cấu trúc hoá từ dạng quan hệ phẳng sang cấu trúc cấy XML.
* Câu truy vấn có thể được đặc chế thành select 1 đối tượng đơn hoặc đối tượng đa.

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

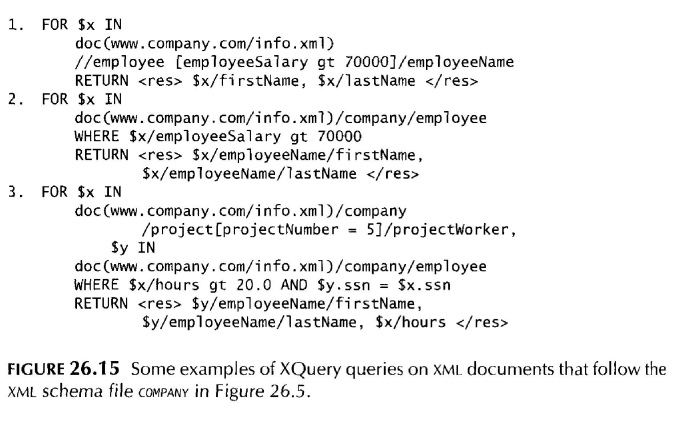
Xquery cho phép nhiều câu truy vấn chung trên nhiều văn bản XML. Dạng chung của truy vấn Xquery được biết như 1 lệnh FLWR, có 4 mệnh đề chính:

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.

### Topic #9b – XML và cơ sở dữ liệu Internets

Mục lục

[I. World Wide Web 30](#_Toc309861610)

[1. Khái quát về HTML: 3](#_Toc309861619)0

[II. Kiến trúc.](#_Toc309861611) 30

[1. Server ứng dụng và Java server-side 32](#_Toc309861619)

[III. Xa hơn HTML (beyond HTML) 33](#_Toc309861617)

[1. Khái quát về XML 34](#_Toc309861619)

[2. XML DTD 35](#_Toc309861619)

[3. DTD đặc tả lĩnh vực (Domain – Specific DTD). 37](#_Toc309861620)

[4. XML –QL: truy vấn dữ liệu XML. 38](#_Toc309861620)

[5. Mô hình dữ liệu bán cấu trúc. 3](#_Toc309861620)9

[6. Các vấn đề cài đặt dữ liệu bán cấu trúc 41](#_Toc309861620)

[IV. Chỉ mục cho tìm kiếm chữ 41](#_Toc309861618)

[1. File nghịch đảo 42](#_Toc309861619)

[2. File ký số 43](#_Toc309861620)

[V. Xếp hạng các từ khóa tìm kiếm trên web. 26](#_Toc309861621)

[1. Thuật toán xếp hạng trang Web. 27](#_Toc309861622)

**XML VÀ CƠ SỞ DỮ LIỆU INTERNET**

# World Wide Web

Web giúp truy xuất file ở bất kỳ đâu trên Internet. Một file được xác định bởi universal resource locator (URL):

*http://www.hcmus.edu.vn/view/main/index.html*

URL trên xác định một file gọi là index.html, được lưu trữ trong thư mục view/main ở máy chủ [www.hcmus.edu.vn](http://www.hcmus.edu.vn). File này được định dạng sử dụng HyperText Markup Language (HTML) và chứa một vài link đến các file khác được xác định thông qua URL

Những đoạn lệnh có định dạng được thông dịch bởi web browser để hiển thị tài liệu cho người dùng và người dùng có thể gọi đến các tài liệu khác bằng cách chọn các link. Khi người dùng nhấp vào link, Web browser connect vào webserver ở điểm đến là website sử dụng protocol gọi là HTTP và submit URL của link. Tập hợp những tài liệu như vậy được gọ là webssite và được qản lý bằng cách sử dụng một chương trình gọi là Web server, nó sẽ chấp nhận URL và trả về các tài liệu tương ứng. World Wide Web lad tập hợp các website có thể truy xuất trên internet.

## Khái quát về HTML:

HTML là một ngôn ngữ đơn giản được sử dụng để miêu tả một tài liệu. Nó còn được gọi là markup languague vì HTML làm việc bằng cách bổ sung các đoạn text thông thường với mark có ý nghĩa quan trọng cho web browser xử lý tài liệu.

*<head>*

*<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />*

*<title>Tin nhanh VnExpress - Đọc báo, tin tức online 24h</title>*

*</head>*

Các đoạn lệnh trong HTML được gọi là tag và chúng bao gồm tag bắt đầu (start tage) và tag kết thúc (end tag) <TAG> và </TAG> theo thứ tự. HTML định nghĩa một tập các tag giống như ý nghĩa của tag, ví dụ <TITLE> là một tag đánh dấu tiêu đề của tài liệu

# Kiến trúc

Để thực thi một chương trình ở web server, server tạo một tiến trình mới và giao tiếp với tiến trình bằng cách sử dụng giao thức common gateway interface (CGI). Kết quả của chương trình có thể được sử dụng để tạp một file HTML phản hồi cho người yêu cầu. Các trang được tính toán theo cách này tại thời gian chúng được yêu cầu được gọi là trang động (dynamic page, các trang tồn tại và được phân phối một cách đơn giản đến Web browser được gọi là trang tĩnh (static page).

*<HTML><HEAD><TITILE> The database bookstore </TITLE></HEAD>*

*<BODY>*

*<FORM action=”find\_books.cgi” method=post>*

*Type an author name:*

*<INPUT type=”text” name=”authorName” size=30 maxLength=50>*

*<INPUT type=”submit” value=”Send it”>*

*<INPUT type=”reset” value=”Clear form”>*

*</FORM>*

*</BODY></HTML>*

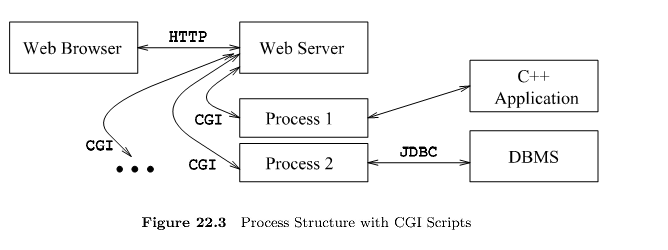
Hãy xem xét ví dụ ở trên. Trang web này chứa một biểu mẫu người dùng điền vào tên của tác giả. Nếu người dùng nhấn vào button “Send it”, Perl script “find\_books.cgi” sẽ thực thi như một tiến trình độc lập. Giao thức CGI định nghĩa giao tiếp của biểu mẫu và đoạn script được thể hiện như thế nào.

Hình 22.4 là một ví dụ của CGI scipt. Chúng tôi đã loại bỏ đoạn lệnh kiểm tra lỗi để đơn giản hóa. Perl là một ngôn ngữ thông dịch thường được sử dụng cho CGI và có nhiều thư viện Perl được gọi là module cung cấp interface cấp cao cho giao thức CGI. Chúng ta sử dụng 2 thư viện trong ví dụ: DBI và CGI. DBI là một API độc lập cơ sở dữ liệu cho Perl cho phép trừu tượng hóa từ DBMS được sử dụng – DBI đóng vai trò trong Perl như JDBC trong Java. Ở đây ta sử dụng DBI như cầu nối đến ODBC driver quản lý kết nối thật đến cơ sở dữ liệu. CGI module là một tập hợp các hàm tạo ra CGI script. Trong phần 1 của scipt mẫu, chúng tôi đã trích ra nội dung của biểu mẫu HTML như sau:

*$authorName = $dataIn -> param(‘authorName’);*

Chú ý rằng tên tham số authorName được sử dụng trong Hình 22.2 để gọi tên field input đầu tiên. Trong phần 2 chúng ta xây dựng một câu lệnh SQL thật trong biến $sql. Trong phần 3, ta bắt đầu dịch mã của trang để trả về cho trình duyệt. Ta muốn hiển thị các dòng kết quả của truy vấn nhhư một những entry trong một list không sắp xếp, chúng ta dùng tag <UL>. Các entry độc lập trong list được bao quanh bởi tag <LI>. Protocol CGI sẽ tóm tắt quá trình xử lý làm sao một trang web được trả về trình duyệt. Vì vậy mọi thứ mà đoạn script viết trong câu lệnh write sẽ là một phần của cấu trúc động của trang web được trả về cho trình duyệt. Phần 4 sẽ thiết lập một kết nối cơ sở dữ liệu và chuẩn bị và thực thi câu lệnh SQL được lưu trong biến $sql ở phần 2. Trong phần 5, chúng ta fetch kết quả của câu truy vấn, mỗi lần một dòng, chèn mỗi dòng vào output. Phần 6 sẽ đống kết nối đến hệ thống cơ sở dữ liệu, và ta kết thúc ở phần 7 bằng cách chèn tag đóng cho trang kết quả.

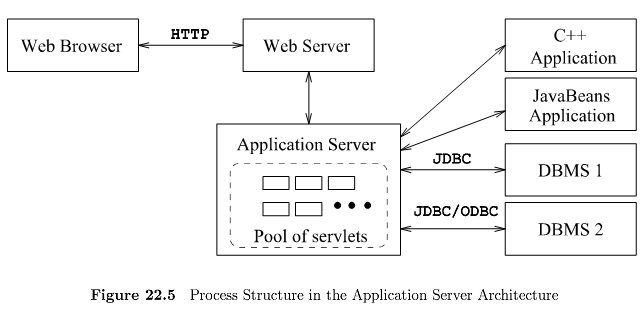
Các protocol thay thể mà trong đó chương trình được gọi lên bằng một yêu cầu, được thực thi nội trong tiến trình Web, và được đề nghị bởi Microsòt (Internet Server API (ISAP)) và Netscape (Netscape ServerAPI (NSAPI)).



## Server ứng dụng và Java server-side

Trong phần trước, chúng ta đã thảo luận cách giao thức CGI có thể được sử dụng để dịch mã tự động một trang web với nội dung được tính toán theo yêu cầu. Tuy nhiên, mỗi yêu cầu trang web đều dẫn tới tạo ra một tiến trình mới, phương án này không tốt đối với số lượng yêu cầu cùng lúc lớn. Vấn đề hiệu suất này đã dẫn đến sự phát triển của một chương trình đặc biệt gọi là server ứng dụng (application server). Một server ứng dụng sẽ tiền phân nhánh các thread hoặc tiến trình và do vậy tránh được chi phí khởi tạo một tiến trình mới cho mỗi yêu cầu. Application server liên quan đến package của tier giữa cung caáp nhiều chức năng khác nhau để loại bỏ chi phí tạo tiến trình:

* Tích hợp các nguồn dữ liệu không đồng nhất.
* Giao tác liên quan đến nhiều nguồn dữ liệu
* Bảo mật
* Quản lý các session

Một kiến trúc Website với application được thể hiện ở hình 22.5. Client giao tiếp với web server thông qua giao thức HTTP. Web server phân phối trang HTML hoặc XML tĩnh trực tiếp đến client. Để dịch mã dynamic page, web server gửi một request đến application server. Application server liên hệ với một hoặc nheièu nguồn dữ liệu để lấy ra các dữ liệu cần thiết hoặc gửi các request cập nhật đến nguồn dữ liệu. Sau khi quá trình giao tiếp với nguồn dữ liệu được hoàn thành, application server sẽ dịch mã trang web và báo cáo kết quả với web server để nó lấy ra page và phân phối đến client. 

Quá trình thực thi của business logic ở Web server site, hay xử lý phía server, đã trở nên mô hình chuẩn cho việc cài đặt các xử lý business phức tạp hơn trên Internet.

# Xa hơn HTML (beyond HTML)

Trong khi HTML thích hợp để biểu diễn cấu trúc của tài liệu phục vụ cho mục đích hiển thị, tính năng của ngôn ngữ không thích hợp cho việc đại diện cho cấu trúc của dữ liệu trong một tài liệu cho các ứng dụng tổng quát hơn là chỉ hiện hiển thị.

Extensible Markup Language (XML) là ngôn ngữ đánh dấu đươc phát triển để bù đắp những thiếu sót của HTML. Ngược lại với HTML có tập các tag có ý nghĩa bắt buộc, XML cho phép người dùng định nghĩa tập hợp các tag có thể được sử dụng cho bất kỳ cấu trúc dữ liệu hoặc tài liệu nào người dùng muốn truyền tải. XML là một cầu nối quan trọng giữa khung nhìn hướng tài liệu của dữ liệu trong HTML và khung nhìn hướng lược đồ của dữ liệu vốn là trung tâm của một DBMS. Nó có tiềm năng làm hệ thống cơ sở dữ liệu dễ dàng tích hợp vào trong ứng dụng web hơn.

XML nổi lên từ sự hợp lưu của hai công nghệ, SGML và HTML. SGML (Standard Generalized Markup Language) là metalanguage cho phép định nghĩa của dữ liệu hoặc tài liệu xen kẽ ngôn ngữ như HTML. Chuẩn SGML được phát hành năm 1988 và nhiều tổ chức quản lý số lượng lớn tài liệu phức tạp đã áp dụng nó. Dù có sự tổng quát, SGML khá phưc tạp và đòi hỏi chương trình tinh vi để khai thác tiềm năng đầy đủ của nó. XML được phát triển để có nhiều sức mạnh của SGML trong khi vẫn giữ được sự đơn giản. Dù sao thì XML vẫn giống SGML cho phép định nghĩa ngôn ngữ đánh dấu tài liệu mới.

Mặc dù XML không ngăn cản người dùng thiết kế code để mã hóa các hiện thỉ dữ liệu trong trình duyệt web, vẫn có một loại ngôn ngữ style cho XML được gọi là XSL (Extensible Style Language). XSL là một chuẩn để miêu tả cách một tài liệu XML - tham gia vào từ vựng của tag - nên được hiển thị

## Khái quát về XML

* *Element:* Element hay còn gọi là tag, là một khối xây dựng cơ bản của tài liệu XML. Điểm bắt đầu nội dung của một element ELM được đánh dấu với <ELM> - gọi là tag bắt đầu (start tag), và kết thúc của nôi dung được đánh dấu với </ELM>, được gọi là end tag. Trong ví dụ element BOOKLIST bao gồ toàn bộ thông tin trong ví dụ. Element BOOK phân ranh giới tất cả dữ liệu với một quyển sách đơn. XML element phân biệt hoa – thường, <BOOK> khác <Book>. Element phải được lồng nhau một cách đúng đắn. Tag bắt đầu xuất hiện trong nội dung của tag khác phải có tag kết thúc tương ứng.

Ví dụ:

*<BOOK>*

*<AUTHOR> <FIRSTNAME>Richard</FIRSTNAME><LASTNAME>Feynman</LASTNAME>*

*</AUTHOR>*

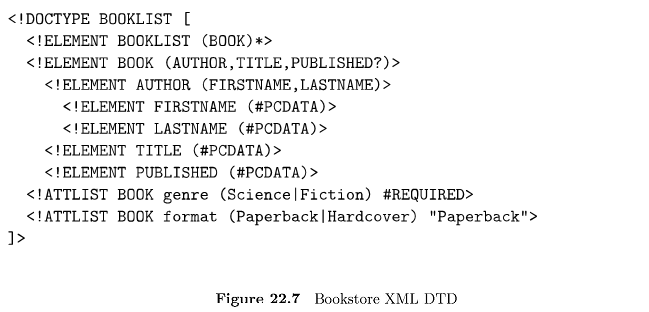
*</BOOK>*

* *Thuộc tính:* Một element có thể có các thuộc tính mô tả cung cấp thông tin mở rộng về element. Giá trị của thuộc tính được set trong tag bắt đầu. Ví dụ <ELM att = “value”> Tất cả giá trị trích dẫn phải nằm trong dấu ngoặc kép.
* *Tham chiếu đối tượng:* đối tượng là những shortcut đối với một phần của text thông thường hoặc nôi dung của các file mở rộng và và chúng ta gọi cách sử dụng của một đối tượng trong tài liệu XML là tham chiếu đối tượng. Ở bất kỳ nơi đâu tham chiếu đối tượng xuất hiện trong tài liệu, nó được thay thế theo đúng nguyên văn bằng nội dung của nó. Tham chiếu đối tượng sẽ bắt đầu với “&” và kết thúc với “;’. Có 5 đối tượng được định nghĩa trong XML là thay thế cho các kí tự đặc biệt trong XML. Ví dụ: “<” - lt, “&” – amp, “>” – gt, “ “ “ – quot, “ ‘ “ – apos. Ví dụ: đoạn text ‘1<5’ đươc viết lại là &apos;1&lt;5&apos.
* *Comment (chú thích)* : chúng ta có thể chèn chú thích vào bất kỳ nơi đâu trong tài liệu XML. Chú thích bắt đầu với <!- và kết thúc -.. Chú thích có thể chứa các đoạn text tùy ý trừ chuỗi “- -“

Khai báo loại tài liệu (DTD), trong XML chúng ta có thể định nghĩa ngôn ngữ đánh dấu cho riêng mình. Một DTD là một tập các rule cho phép ta xác định tập element, thuộc tính và đối tượng cho chính mình. Vì vậy, DTD là ngữ pháp căn bản chỉ định tag nào được cho phép, theo thức tự nào chúng có thể xuất hiện và được lồng nhau ra sao.

## XML DTD

Một DTD là tập các qui tắc cho phép ta xác định tập element, thuộc tính và đối tượng của chính mình. DTD xác định yếu tố nào chúng ta có thể sử dụng và ràng buộc trên các element đó,… các element có thể lồng vào nhau như thế nào và element có thể xuất hiện ở đâu trong tài liệu. Chúng ta sẽ gọi một tài liệu có hiệu lực nếu có một DTD liên quan với nó và tài liệu đươc cấu trúc theo các tậ qui tắc của DTD.



Một DTD được bao quanh bởi <!DOCTYPE name [DTDdeclaration]>, trong đó name là tên của tag bao quanh ngoài cùng, và DTDdeclaration là đoạn text qui tắc của DTD. DTD bắt đầu với element ngoài cùng, còn được gọi là element gốc (root element) – là BOOKLIST trong ví dụ

*<!ELEMENT BOOKLIST (BOOK) \*>*

Qui tắc này cho biết BOOKLIST element chứa 0 hoặc nhiều BOOK element. Dấu “\*” sau từ BOOK ám chỉ có bao nhiêu BOOK element có thể xuất hiện trong BOOKLIST element.

* \*: 0 hoặc nhiều
* +: một hoặc nhiều
* ?: 0 hoặc 1

*<!ELEMENT LASTNAME <#PCDATA)>*

Đến đây chúng ta chỉ mới xem xét các element chứa các element khác. Qui tắc trên cho biết LASTNAME là một element không chứa các element khác, nhưng chứa đoạn text thật. Element chỉ chứa các element khác được gọi là có nội dung element, trong khi #PCDATA được gọi là nội dung hỗn hợp. Tóm lại, một khai báo loại element có cấu trúc sau:

*<!ELEMENT (contentType)>*

Có 5 loại nội dung:

* Chứa các element khác
* #PCDATA: ám chỉ dữ liệu kí tự (charater)
* EMPTY: ám chỉ element không có nội dung, element không có nội dung không bắt buộc có tag đóng
* ANY: cho biết bất kỳ loại nội dung cũng được cho phép. Loại nội dung nên tránh bất kỳ khi nào có thể bỏi vì chúng làm vô hiệu mọi kiểm tra cấu trúc tài liệu bên trong element.
* Một biểu ngữ thông thường (regular expression) xây dựng từ 4 lựa chọn trên. Một regular expression là một trong những nội dung sau:
  + Exp1, exp2, exp3: danh sách các regular expression
  + Exp\*: một expression mở rộng (không hoặc nhiều sự xuất hiện)
  + Exp?: một expression mở rộng (không hoặc nhiều sự xuất hiện)
  + Exp+: một expression bắt buộc (một hoặc nhiều sự xuất hiện)
  + Exp1 | exp2: exp1 or exp2

Thuộc tính được khai báo bên ngoài element. Ví dụ, hãy xem xét khai báo thuộc tính dưới đây:

*<!ATTLIST BOOK genre (Science | Fiction) #REQUIRED>*

Phần XML DTD này xác định thuộc tính genre, là một thuộc tính của element BOOK. Thuộc tính có thể lấy 2 giá trị: Science hoặc Fiction. Mỗi BOOK element phải được mô tả trong tag bắt đầu của nó bằng thuộc tính genre vì thuộc tính này bắt buộc được chỉ định bằng từ khóa #REQUIRED. Dưới đây là cấu trúc tổng quát của một khai báo thuộc tính DTD:

*<!ATTLIST elementName (attName attType default) + >*

Từ khóa ATTLIST xác định sự bắt đầu của một khai báo thuộc tính. Chuỗi elementName là tên của element theo sau là định nghĩa thuộc tính liên quan. Những yếu tố theo sau là định nghĩa của một hoặc nhiều thuộc tính. Mỗi thuộc tính có một tên được xác định bởi attNam và một loại – attType. XML định nghĩa một số loại có thể có cho một thuộc tính. Ở đây, chúng ta chỉ nói về loại string và enumerated. Một thuộc tính kiểu string có thể sử dụng bất kỳ chuỗi nào làm giá trị. Chúng ta có thể khai báo một thuộc tính bằng cách thiết lập loại của nó là CDATA. Ví dụ:

*<!ATTLIST BOOK edition CDATA “1”>*

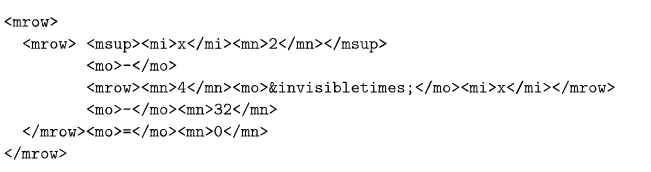
Nếu một thuộc tính thuộc kiểu enumerated, chúng ta liệt kê toàn bộ các giá trị có thể trong khai báo. Trong ví dụ, thuộc tính genre là một kiểu enumerated, giá trị có thể là “Science” hoặc “Fiction”

Phần cuối cùng của khai báo thuộc tính được goi là đặc tả mặc định. XML DTD trong hình 22.7 thể hiện 2 đặc tả mặc định khác nhau: #REQUIRED và chuỗi “Paperback”. Đặc tả mặc định #REQUIRED cho biết thuộc tính đó là bắt buộc và bất cứ khi nào các element liên quan tới nó xuất hiện trong tài liệu XML, một giá trị cho thuộc tính phải được xác định. Ngược lại, đặc tả mặc định được xác định bởi chuỗi “Paperback” ám chỉ thuộc tính đó không bắt buộc.

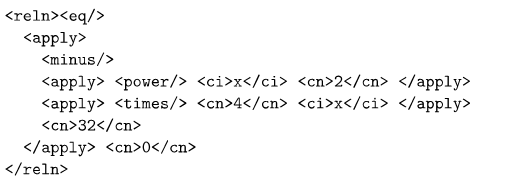
## DTD đặc tả lĩnh vực (Domain – Specific DTD)

Gàn đây, DTD đã được phát triển cho một số lĩnh vực chuyên biệt – bao gồm phạm vi rộng của lĩnh vực thương mại, kĩ thuật, tài chính, công nghiệp, khoa học – và nhiều điều thú vị về XML có bắt nguồn từ sự tin tưởng rằng sẽ có nhiều chuẩn DTD hơn nữa được phát triển.

Ví dụ. MathML (mathematical markup language) được phát triểnn cho việc giải mã các tài iệu toán học trên Web. Ví dụ biểu thức sau: được diễn tả như sau với element hiển thị như sau:



Với element nội dung, biểu thức được biểu diễn:

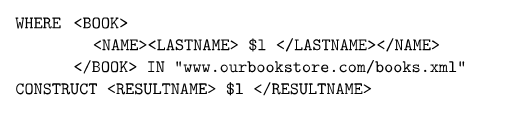


## XML –QL: truy vấn dữ liệu XML

Dữ liệu được mã hóa bằng cách phản ánh cấu trúc của một tài liệu XML, chúng ta có cơ hội để sử dụng ngôn ngữ bậc cao khai thác cấu trúc này một cách thuận lợi để lấy dữ liệu ra. Loại ngôn ngữ này sẽ mang quản lý dữ liệu XML gần với quản lý cơ sở dữ liệu hơn là các mẫu hướng text của tài liệu XML. Loại ngôn ngữ này cho phép ta dễ dàng chuyển đổi dữ liệu XML giữa các DTD khác nhau, điều này là bắt buộc đối với việc tích hợp dữ liệu từ nhiều nguồn.

Trong chương này, chúng ta sẽ tìm hiểu về một ngôn ngữ truy vấn xác định dành cho XML được gọi là XML – QL - có sự giống nhau manh mẽ với các ngôn ngữ truy vấn khác đã được phát triển trong cộng đồng cơ sở dữ liệu.

Hãy xem xét lại tài liệu XML được thể hiện trong hình 22.6. Truy vấn ví dụ dưới đây trả về last name của tất cả tác giả, với giả thiết tài liệu XML này được đặt ở [www.ourbookstore.com/books.xml](http://www.ourbookstore.com/books.xml)

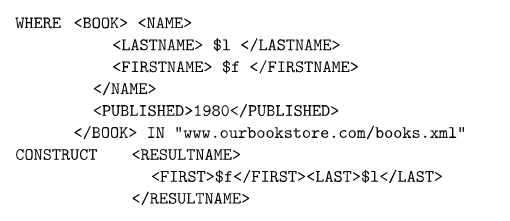


Câu truy vấn này rút trích dữ liệu từ tài liệu XML bằng cách xác định pattern của đánh dấu. Chúng ta quan tâm đến dữ liệu được lồng trong BOOK element, NAME element và LASTNAME element. Đối với mỗi phần của tài liệu XML trùng khớp với cấu trúc xác định bởi câu truy vấn trên, biến l gắn với nội dung của LASTNAME element. Để phân biệt tên biến từ text thông thươnd, ta sử dụng tiền tố $. Nếu như câu truy vấn này được áp dụng cho dữ liệu trong hình 22.6, kết quả trả về sẽ là:

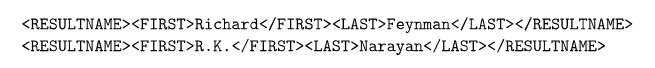
*<RESULTNAME>Feynman</RESULTNAME>*

*<RESULTNAME>Narayan</RESULTNAME>*

Sự chọn lọc được diễn tả bằng cách thay thế đoạn text trong nội dung của một element. Và kết quả của một truy vấn không giới hạn một element đơn. Ví dụ với truy vấn sau:



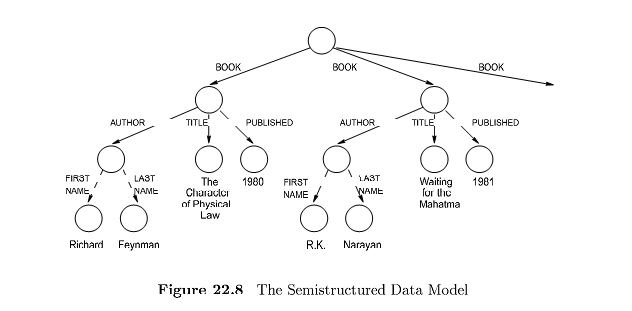
Kết quả của câu truy vấn:



## Mô hình dữ liệu bán cấu trúc

Xem xét một tập các tài liệu trên Web có chứa các hyperlink đến tài liệu khác. Những tài liệu này, mặc dù không hoàn toàn không có cấu trúc, có thể không được mô hình hóa một cách tự nhiên trong mô hình dữ liệu quan hệ vì pattern của hyperlink không thông thường đối với tài liệu. Một tập tin thư mục còn có các bậc của cấu trúc xác định dựa trên các field, như author và title, tuy nhiên lại lại đoạn text không có cấu trúc. Trong khi một số dữ liệu hoàn toàn không có cấu trúc như video stream, audio stream, và hình ảnh thì có nhiều dữ liệu không hoàn toàn có cấu trúc cũng không hoàn toàn không có cấu trúc. Khi đó chúng ta gọi các kiểu dữ liệu như vậy là dữ liệu bán cấu trúc. Tài liệu XML đại diện cho tài nguyên quan trọng và phát triển của dữ liệu bán cấu trúc, và lý thuyết của mô hình dữ liệu bán cấu trúc và truy vấn có tiềm năng phục vụ như sự hình thành của XML.

Có rất nhiều lý do tại sao dữ liệu có thể bán cấu trúc. Đầu tiên, cấu trúc của dữ liệu có thể hiểu ngầm, ẩn, không xác định hoặc người dùng có thể phớt lờ. Thứ hai, xem xét vấn đề tích hợp dữ liệu từ những nguồn dữ liệu bất đồng bộ khi đó việc chuyển đỏi dữ liệu và truyền tải là vấn đề quan trọng. Chúng ta cần một mô hình dữ liệu linh hoạt để tích hợp dữ liệu từ tất cả nguồn dữ liệu bao gồm tập tin phẳng và hệ thống thừa kế, một mô hình dữ liệu cấu trúc thường quá cứng nhắc. Thứ ba, chúng ta không thể truy vấn một cơ sở dữ liệu cấu trúc mà không biết lược đồ, nhưng đôi khi chúng ta muốn truy vấn dữ liệu mà không cần biết toàn bộ lược đồ. Ví dụ, chúng ta không thể diễn ta truy vấn “ Ở đâu trong cơ sở dữ liệu chúng ta có thể tìm thấy chuỗi Malgudi?” trong hệ quản trị cơ sở dữ liệu mà không biết lược đồ.

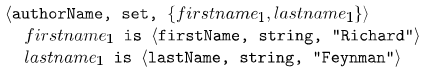


Tất cả mô hình dữ liệu đề xuất cho kiểu dữ liệu bán cấu trúc đại diện cho dữ liệu như một số loại của biểu đồ gắn nhãn. Các nút trong biểu đồ tương ứng với đối tượng phức hợp hoặc giá trị nguyên tố, và cạnh tương ứng với thuộc tính. Không có lược đồ riêng biệt và mô tả bổ sung, dữ liệu trong lược đồ tự mô tả. Ví dụ, xem xét lược đồ hình 22.8, đại diện cho một phần dữ liệu XML trong hình 22.6, nút gốc đại diện cho element ngoài cùng, BOOKLIST. Nút có 3 cạnh được đặt tên element là BOOK, vìvậy danh sách có 3 quyển sách.

Bây giờ chúng ta thảo luận về một trong những mô hình dữ liệu được đề xuất cho dữ liệu bán cấu trúc, được gọi là OEM. Mỗi đối tượng được mô tả bằng bộ 3 một label, một kiểu, và giá trị của đối tượng. Vì mỗi đối tượng có một label có thể được xem là tên một cột trong mô hình quan hệ, và mỗi đối tượng có một kiểu được xem như kiểu dữ liệu của cột trong mô hình quan hệ, mô hình trao đổi đối tượng căn bản là tự mô tả. Label trong mô hình trao đổi đối tượng nên càng nhiều thông tin càng tốt vì nó phục vụ cho hai mục đích: chúng có thể xác định một đối tượng cũng như truyền tải ý nghĩa của đối tượng. Ví dụ, chúng ta có thể biểu diễn last name của một tác giả như sau:

*(lastName, string, “Feynman”)*

Các đối tượng phức tạp hơn có thể được triển khai phân cấp vào đối tượng nhỏ hơn. Ví dụ, một tên tác giả có thể chứa first name và last name.



## Các vấn đề cài đặt dữ liệu bán cấu trúc

Hệ thống cơ sở dữ liệu hỗ trợ dữ liệu bán cấu trúc đã được tập trung nghiên cứu gần đây và cho thấy sự thành công về mặt thương mại của XML, điểm mạnh này sẽ tiếp tục được phát huy. Dữ liệu bán cấu trúc đặt ra nhiều thách thức mới vì hầu hết các chiến lược truy vấn, chỉ mục và lưu trữ truyền thống đều giả sử dữ liệu dính với lược đồ thông thường. Ví dụ, chúng ta nên lưu dữ liệu bán cấu trúc bằng cách ánh xạ nó vào mô hình quan hệ và sau đó lưu trữ dữ liệu đã ánh xạ trong hệ cơ sở dữ liệu quan hệ hay không? Thực hiện chỉ mục trên dữ liệu bán cấu trúc như thế nào?... Tất cả đều đang trong quá trình nghiên cứu

# Chỉ mục cho tìm kiếm chữ

Trong chương này, chúng ta giả sử cơ sở dữ liệu là một tập các tài liệu và chúng ta gọi cơ sở dữ liệu như vậy là cơ sở dữ liệu chữ. Để đơn giản, chúng ta giả sử cơ sở dữ liệu chỉ chứa chính xác một quan hệ và lược đồ quan hệ có chính xác một field tài liệu kiểu. Vì vậy, mỗi record trong quan hệ chứa chính xác một tài liệu. Trong thực hành, lược đồ quan hệ sẽ chứa các field khác như ngày tao tài liệu, một sự phân loại tài liệu, hoặc filed với từ khóa mô tả tài liệu. Cơ sở dữ liệu dạng chữ được sử dụng để lưu trữ các bài báo, giấy tờ hợp pháp và những loại tài liệu khác.

Một lớp quan trọng của truy vấn dựa trên từ khóa tìm kiếm cho phép chúng ta tìm kiếm tất cả tài liệu chứ từ khóa đã cho. Đây là loại truy vấn thông dụng nhất trên web ngày nay, và được hỗ trợ bởi số lượng lớn engine tìm kiếm như AltaVista và Lycos. Một vài hệ thống duy trì danh sách đồng nghĩa của các từ quan trọng và trả về tài liệu chứ từ khóa mong muốn hoặc một trong những từ đồng nghĩa của chúng: ví dụ truy vấn tìm kiếm từ “car” sẽ lấy ra tài liệu chứ “automobile”. Một câu truy vấn phức tạp hơn sẽ chứa các từ khóa AND, OR, NOT, chúng ta có thể xếp hạng các tài liệu được lấy ra bằng xác suất từ khóa truy vấn trong tài liệu.

Có 2 loại truy vấn thông dụng: truy vấn kiểu bool và truy vấn xếp hạng. Trong truy vấn kiểu bool, người dùng cung cấp các biểu thức kiểu bool như dạng sau hay còn gọi là dạng hội thông thường:

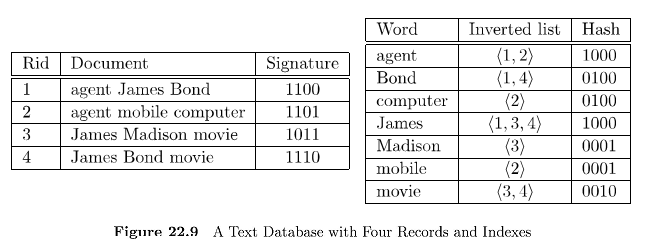


Trong đó là các thuật ngữ truy vấn hay từ khóa độc lập

Truy vấn xếp hạng có cấu trúc tương tự. Trong truy vấn xếp hạng, người dùng còn xác định danh sách các từ, nhưng kết quả của truy vấn là danh sách tài liệu được xếp hạng bằng sự tương đương với danh sách thuật ngữ của người dùng. Làm sao để định nghĩa khi nào và như thế nào một tài liệu tương ứng với một tập các thuật ngữ của người dùng là một vấn đề khó khăn. Thuật ngữ để đánh giá truy vấn thuộc về lĩnh vực truy tìm thông tin, vốn gần gũi với quản lý cơ sở dữ liệu. Hệ thống truy tìm dữ liệu, giống như hệ thống cơ sở dữ liệu, có mục tiêu cho phép người dùng truy vấn một khối lượng dữ liệu lớn, nhưng tập trung vào tập lớn các tài liệu không có cấu trúc. Cập nhật, điều khiển đồng thời, và phục hồi không được chú trọng trong hệ thống truy tìm thông tin vì dữ liệu trong ứng dụng điển hình là tĩnh

## File nghịch đảo

Một file nghịch đảo là một cấu trúc chỉ mục cho phép truy vấn nhanh tất cả tài liệu chứa thuật ngữ truy vấn. Đối với mỗi thuật ngữ, chỉ mục duy trì một danh sách sắp xếp (goi là danh sách nghịch đảo) của id tài liệu chứa thuật ngữ chỉ mục. Ví dụ, xem xét cơ sở dữ liệu dạng chữ trong hình 22.9. Thuật ngữ truy vấn ‘James’ có danh sách đảo của các id record (1,3,4).



Để tìm kiếm nhanh danh sách nghịch đảo cho một thuật ngữ truy vấn, tất cả thuật ngữ truy vấn được tổ chức trong một cấu trúc chỉ mục thứ 2 như cây B+ hoặc chỉ mục hash. Để tránh bất kỳ sự hỗn loạn nào, chúng ta gọi chỉ mục thứ hai cho phép truy vấn nhanh danh sách nghịch đảo của một thuật ngữ truy vấn là chỉ mục từ vựng. Chỉ mục từ vựng chứa mỗi thuật ngữ truy vấn có thể có và một con trỏ đến danh sách đảo

Một truy vấn chứa thuật ngữ đơn được tính toán bằng cách đầu tiên cắt ngang chỉ mục từ vựng đến entry nút lá với địa chỉ của danh sách nghịch đảo cho thuật ngữ. Sau đó danh sách đảo được lấy ra, rid được ánh xạ vào địa chỉ vật lý của tài liệu, và tài liệu tương ứng được gọi lên. Một truy vấn với hội của vài thuật ngữ được tính toán bằng cách lấy ra danh sách nghịch đảo của các thuật ngữ truy vấn cùng một lúc và giao lại với nhau. Để giảm tối đa sử dụng bộ nhớ, danh sách nghịch đảo nên được lấy ra để làm tăng độ dài. Một truy vấn với sự tách rời các thuật ngữ được tính toán bằng cách trộn các danh sách nghịch đảo với nhau.

## File ký số

File ký số là một cấu trúc chỉ mục khác cho hệ thống cơ sở dữ liệu chữ hỗ trợ tính toán hiệu quả truy vấn kiểu bool. File ký số chứa một record chỉ mục cho mỗi tài liệu trong cơ sở dữ liệu. Record chỉ mục sẽ được gọi là ký số của tài liệu. Mỗi ký số có kích thước b bit cố định, b được gọi là độ rộng ký số. Làm sao chúng ta quyết định bit nào để thiết lập cho một tài liệu. Bit được thiết lập tùy thuộc vào từ hiển thị trong tài liệu. Chúng ta ánh xạ từ vào bit bằng cách áp dụng hàm hash cho mỗi từ trong tài liệu và chúng ta thiết lập bit xuất hiện trong kết quả của hàm hash. Chú ý là trừ khi chúng ta có một bit cho mỗi từ trong từ vựng, bit giống nhau có thể được thiết lập 2 lần bằng 2 từ khác nhau bởi vì hàm hash ánh xả cả 2 từ vào cùng bit. Chúng ta nói rằng ký số S1 khớp với một ký số khác S2 nếu tất cả các bit được thiết lập trong ký số S2 cũng được thiết lập trong ký số S1. Nếu ký số S1 khớp với S2, thì ký số S1 có ít nhất nhiều bit được thiết lập như ký số S2.

Đối với truy vấn chứa tập hợp của những thuật ngữ, đầu tiên chúng ta phát sinh ra ký số truy vấn bằng cách áp dụng hàm hash để đọc mỗi từ trong truy vấn. Sau đó quét các file ký số và lấy ra toàn bộ tài liệu có ký số khớp với ký số truy vấn, vì mỗi tài liệu như vậy là một kết quả tìm năng cho câu truy vấn. Vì ký số không xác định độc nhất những từ mà tài liệu chứa, chúng ta phải lấy ra mỗi lần trùng khớp và kiểm tra tài liệu đó thật sự chứa thuật ngữ truy vấn hay không. Một tài liệu có ký số phù hợp nhưng không chứa những thuật ngữ đó được gọi là một xác thực sai. Một xác thực sai là một lỗi khá đắt vì tài liệu phải được lấy ra từ đĩa, phân tích, phát sinh và kiểm tra để quyết định nó có chứa thuật ngữ truy vấn hay không.

Đối với truy vấn chứa các thuật ngữ rời rạc, ta phát sinh danh sách các ký số truy vấn, mỗi ký số một thuật ngữ. Truy vấn được tính toán bằng cách quét file ký số để tìm kiếm tài liệu có ký số trùng khớp với bất kỳ ký số nào trong danh sách ký số truy vấn.

Chú ý là mỗi truy vấn chúng ta phải quét toàn bộ file ký số, và càng nhiều record trong file ký số thì càng có nhiều tài liệu trong cơ sở dữ liệu. Để giảm lượng dữ liệu phải truy vấn cho mỗi câu truy vấn, chúng ta có thể phân vùng dọc file ký số thành các bit slice, chúng ta gọi một chỉ mụ một file ký số được cắt bit. Độ dài của mỗi bit slice vẫn bằng với số tài liệu trong cơ sở dữ liệu, nhưng với mỗi truy vấn có q bit được thiết lập trong ký số truy vấn, chúng ta chỉ cần lấy ra q bit slice

Ví dụ trong hình 22.9 với file ký số độ rộng là 4. Số bit thiết lập bởi giá trị hash của các thuật ngữ truy vấn được thể hiện trong hình. Để tính toán cho câu truy vấn ‘James’, đầu tiên ta sẽ tính toán giá trị hash của thuật ngữ là 1000. Sau đó quét file ký số và tìm record chỉ mục trùng khớp. Như trong hình 22.9, ký số của tất cả record đều có bit đầu tiên được thiết lâp, chúng ta lấy ra tất cả tài liệu và kiểm tra xác thực sai, ở đây chỉ có một xác thực sai là rid 2.

# Xếp hạng các từ khóa tìm kiếm trên web

World Wide Web chứa một lượng thông tin dị thường. Tìm kiếm một trang web tương ứng với truy vấn của người dùng còn khó hơn cả mò kim đáy bể. Sự đa dạng của các trang web trong cấu trúc, nội dung, quyền sở hữu, chất lượng, … làm chúng trở nên khó khắn nếu không không thể áp dụng các kĩ thuật truy vấn tiêu chuẩn.

Trong quá trình tìm kiếm các trang web tương ứng, chúng ta phân biệt giữa 2 loai trang: authority và hub. Một authority là một trang tương ứng với một chủ đề xác định và được nhận diện bằng các trang khác như có căn cứ trên vấn đề. Những trang được gọi là hub có số lượng lớn hyperlink đến authority, mặc dù bản thân chúng ít được biết đến và ít mang nôi dung tương đương với truy vấn. Trang hub có thể là sự biên dịch các tài nguyên về một chủ đề trên một site cho các chuyên gia, danh sách các site khuyến khích cho sở thích của một cá nhân người dùng.

## Thuật toán xếp hạng trang Web

Trong phần này chúng ta sẽ nói về HITS, một thuật toán tìm kiếm authority và hub và trả về kết quả cho truy vấn người dùng. Chúng ta sẽ xem World Wide Web như một bản đồ trực tiếp. Mỗi Web page đại diện cho một nút trong lược đò, và hyperlink từ trang A đến trang B đại diện là một cạnh giữa 2 nút tương ứng. Giả sử chúng ta được cho một truy vấn người dngf với một vài thuật ngữ. Thuật toán thực hiện trong 2 bước:

* ***Bước 1:*** bước lấy mẫu (sampling step), chúng ta thu thập tập các trang web được gọi là tập cơ bản. Tập cơ bản chứa các trang gần tương đương với truy vấn của người dùng, nhưng nó còn khá lớn.
* ***Bước 2:*** bước lặp (iteration step) chúng ta tìm các authority và hub tốt giữa các trang trong tập cơ bản.

Bước lấy mẫu sẽ lấy tập các trang Web chứa thuật ngữ truy vấn, sử dụng một vài kĩ thuật truyền thống. Ví dụ, chúng ta có thể tính toán truy vấn như một từ khóa bool , tìm kiếm và lấy ra các trang Web chứa thuật ngữ truy vấn. Chúng ta gọi tập kết quả trang là tập gốc. Tập gốc có thể không chứa các trang tương đương vì một vài authority page hoặc một vài authority page dãn đến trang trong tập gốc. Điều này thúc đẩy một khái niệm là link page (trang liên kết). Chúng ta gọi là một link pgae nếu nó có một hyperlink đến trang trong tập gốc hoặc một trang trong tập gốc có hyperlink đến nó. Để không bỏ sót các trang tương ứng, chúng ta bổ sung tập gốc bằng tất cả các link page và gọi đó là tập cơ bản. Vì vậy, tập cơ bản chứa tất cả trang gốc và trang liên kết, chúng ta sẽ gọi trang web trong tập cơ bản là trang cơ bản.

Mục đích của bước 2 là tìm ra trang cơ bản nào là hub và authority tốt và trả về kết quả co người dùng. Để xác định số lượng chất lượng của một trang cơ bản như hub và authority, chúng ta liên kết mỗi trang cơ bản trong tập cơ bản với một hub weight và authority weight. Hub weigtht của một trang ám chỉ chất lượng của một trang theo hub, và tương tự với authority weight. Chúng ta tính toán weight của mỗi trang theo trực giác rằng nếu một trang là authority tốt nếu có nhiều hub tốt hyperlink tới nó, và một hub tốt nếu có nhiều hyperlink ra ngoài đến các authority tốt. Vì chúng ta không có bất kỳ kiến thức nào về trang nào là hub và authority tốt, chúng ta khỏi tạo tất cả weight là 1. Sau đó cập nhật weight lặp đi lặp lại như dưới đây

Hãy xem xét một trang cơ bản p với hub weight là và authority weight . Trong một vòng lặp, chúng ta cập nhật là tổng của tất cả hub weight của tất cả trang có hyperlink đến p.

Một cách tương tự, chúng ta cập nhật là tổng weight của các trang trỏ mà p trỏ tới

So sánh thuật toán với các phương pháp truy vấn chữ đã thảo luận trong chương này, chúng ta lưu ý bước lặp của thuật toán HITS – sự phân phối các weight – không xét đến các chữ trong trang cơ bản. Trong bước lặp, chúng ta chỉ quan tâ, đến quan hệ giữa các trang cơ bản được đại diện bằng hyperlink.

Thuật toán HITS thường tạo ra kết quả tốt. Ví dụ, 5 authority được xếp hạng cao nhất cho truy vấn ‘search engine’ là

<http://www.yahoo.com/>

<http://www.excite.com/>

<http://www.mckinley.com/>

<http://www.lycos.com/>

<http://www.altavista.digital.com/>

# TOPIC #8a: CÁC KHÁI NIỆM CỦA CƠ SỞ DỮ LIỆU ĐỐI TƯỢNG

Mục lục

[I.Giới thiệu 47](#_Toc309861610)

[II.Nguồn gốc của cách tiếp cận hướng đối tượng. 4](#_Toc309861611)8

[1. Nguồn gốc . 4](#_Toc309861612)8

[2. Đặc điểm của đối tượng 4](#_Toc309861613)8

[3. Các vấn đề phát sinh 49](#_Toc309861614)

[III.Định danh, cấu trúc, hàm dựng của đối tượng 49](#_Toc309861617)

[1. Định danh. 4](#_Toc309861612)9

[2. Cấu trúc 50](#_Toc309861613)

[3. Hàm dựng 5](#_Toc309861614)1

[IV.Tính đóng gói của của đối tượng, toán tử, phương thức, đối tượng lâu dài 52](#_Toc309861618)

[1. Xác định phương thức của đối tượng thông qua các lớp 52](#_Toc309861619)

[2. Xác định tính lâu dài của đối tượng thông qua việc đặt tên và tiếp cận 53](#_Toc309861620)

[V. Lớp phân cấp và kế thừa. 55](#_Toc309861621)

[1. Khái niệm 55](#_Toc309861622)

[2. Ràng buộc về mức độ tương ứng của hệ thống phân cấp. 56](#_Toc309861623)

[VI. Đối tượng phức tạp 57](#_Toc309861621)

[1. Đối tượng không có cấu trúc và khả năng mở rộng 57](#_Toc309861622)

[2. Đối tượng có cầu trúc. 58](#_Toc309861623)

[VII. Khái niệm khác của hướng đối tượng 59](#_Toc309861621)

[1. Đa hình(nạp chồng toán tử) 59](#_Toc309861622)

[2. Đa kế thừa và kế thừa có chọn lọc. 60](#_Toc309861623)

[3. Các phiên bản và cấu hình. 60](#_Toc309861623)

[VIII. Tổng kết 61](#_Toc309861621)

1. **Giới thiệu:**

* Đối với mô hình và hệ thống cơ sở dữ liệu truyền thống như là cơ sở dữ liệu quan hệ, mạng, phân cấp đã khá thành công trong việc phát triển công nghệ cơ sở dữ liệu cần thiết cho các ứng dụng cơ sở dữ liệu kinh doanh truyền thống.
* Tuy nhiên nó cũng còn thiếu sót đối với các ứng dụng cần cơ sở dữ liệu phức tạp hơn cần được thiết kế và triển khai như:
  + Hỗ trợ thiết kế và sản xuất với sự trợ giúp của máy tính (CAD/CAM);
  + Tích hợp sản xuất máy tính (CIM);
  + Hỗ trợ công nghệ phầm mềm (CASE);
  + Hệ thống thông tin địa lý (GIS);
  + Khoa học và y học;
  + Lưu trữ và khôi phục tài liệu; v.v.
* Đối với những ứng dụng mới hơn, ví dụ như cấu trúc phức tạp hơn cho các đối tượng: thời gian giao dịch, kiểu dữ liệu mới cho hình ảnh hoặc lưu trữ văn bản lớn, các tiếp cận hướng đối tượng cung cấp sự linh hoạt để xử lý một số các yêu cầu này mà không bị giới hạn về kiểu dữ liệu hay ngôn ngữ truy vấn trong mô hình dữ liệu truyền thống.
* Chìa khóa của cơ sở dữ liệu hướng đối tượng là sức mạnh mà họ cung cấp cho nhà thiết kế để xác định cả cấu trúc của đối tượng phức tạp và các xử lý áp dụng cho đối tượng đó.
* Ngoài ra cơ sở dữ liệu hướng đối tượng còn tăng tính khả dụng, thích hợp với các ngôn ngữ lập trình hướng đối tượng, là một thành phần cơ bản được tích hợp trức tiếp vào ngôn ngữ lập trình, mà cơ sở dữ liệu truyền thống khó mà đáp ứng được.(C++, Smalltalk hay Java)
* Các tính năng bổ sung của mô hình dữ liệu này đã được công nhận với các nhà cung cấp DBMS (Database Managerment System).
* Mặc dù đã có nhiều thử nghiệm cũng như là bản thương mại của cơ sở dữ liệu hướng đối tượng nhưng nó vẫn chưa được sử dụng rộng rãi vì tính phổ biến của cở sở dữ liệu quan hệ.
* Như các phiên bản thương mại hệ hệ quản trị cơ sở dữ liệu hướng đối tượng hiện hành, sự cần thiết cho một mô hình và ngôn ngữ tiêu chuẩn đã được công nhận.
* Nội dung sẽ được trình bày bao gồm:
  + Chương 20.1: Nguồn gốc của cách tiếp cận hướng đối tượng.
  + Chương 20.2: Thảo luận về: object identity, object structure, and type constructors.
  + Chương 20.3: Trình bày về tính đóng gói của các toán tử và định nghĩa các phương thức như là một phần của việc khai báo lớp, và cũng thảo luận về cơ chế lưu trữ các đối tượng trong cơ sở dữ liệu.
  + Chương 20.4: Thảo luận về tính phân cấp và kế thừa.
  + Chương 20.5: Tổng quan về các vấn đề phát sịnh khi đối tượng phưc tạp được sử dụng và lưu trữ.
  + Chương 20.6: Thảo luận về tính đa hình, nạp chồng toán tử, linh động của ràng buộc…

1. **Nguồn gốc của cách tiếp cận hướng đối tượng:**
2. **Nguồn gốc**

* Từ hướng đối tượng(OO) có nguồn gốc từ ngôn ngữ lập trình hướng đối tượng (OOPLs).
* Ngày nay khái niệm OO được áp dụng trong các lĩnh vực cơ sở dữ liệu, công nghệ phần mềm, cơ sở kiến thức, trí tuệ nhân tạo, hệ thống máy tính nói chung.
* OOPLs có nguồn gốc từ ngôn ngữ SIMULA( Tên ngôn ngữ được phát triển dành cho các ứng dụng có tính chất mô phỏng trên máy tính Simula 67 là một cột mốc quan trọng trong sự phát triển của ngôn ngữ lập trình vù nó chứa việc tổng quát hóa quan trọng cho khái niệm khối, được gọi là lớp Simula là ngôn ngữ đầu tiên mở ra một dòng ngôn ngữ mới: ngôn ngữ lập trình hướng đối tượng)
* Các ngôn ngữ lập trình Smalltalk, phát triển tại Xerox PARC trong những năm 1970, là một trong những ngôn ngữ đầu tiên một cách rõ ràng kết hợp thêm các khái niệm OO, chẳng hạn như qua tin nhắn và thừa kế. Nó được biết đến như một ngôn ngữ lập trình OO thuần, có nghĩa rằng nó đã được một cách rõ ràng được thiết kế hướng đối tượng. Điều này trái ngược với ngôn ngữ lập trình hướng đối tượng lai, kết hợp các khái niệm OO vào một ngôn ngữ đã được hiện có.Một ví dụ thứ hai là C + +, kết hợp các khái niệm OO vào ngôn ngữ lập trình C phổ biến.

1. **Đặc điểm hướng đối tượng**

* Hai thành phần điển hình của hướng đối tượng là: thuộc tính và phương thức, tương tự như trong ngôn ngữ lập trình, ngoại trừ cơ sở dữ liệu hướng đối tượng sẽ có một cấu trúc dữ liệu phức tạp cũng như các hoạt động cụ thể được xác định bởi các lập trình viên. Đối tượng trong một OOPL chỉ tồn tại trong quá trình thực hiện chương trình và đó gọi là một đối tượng tạm thời. Khác ở điểm này cơ sở dữ liệu hướng đối tượng có thể kéo dài sự tồn tại đến khi nào chúng bị xóa (hay còn gọi là lưu trữ vĩnh viễn), do đó các đối tượng tồn tại khi chấm dứt chương trình và có thể lấy ra sau đó, được chia sẽ bởi các chương trình khác. Nói cách khác, cơ sở dữ liệu hướng đối tượng tiến hành lưu trữ có hệ thống. Điều này làm cho nó dễ dàng hơn để phát triển dữ liệu của mọt hệ thống từng bước một, và sử dụng lại các định nghĩa khác khi tạo ra đối tượng mới.

1. **Các vấn đề phát sinh**

* Mối quan hệ giữa các đối tượng, tính đóng gói trong OO lập luận rằng mối quan hệ không nên được đại diện một cách rõ ràng, nhưng thay vào đó nên được mô tả bằng cách định nghĩa các phương thức phù hợp để xác định vị trí của đối tượng liên quan. Tuy nhiên cách tiếp cận này hoạt động không tốt đối với cơ sở dữ liệu phức tạp nhiều mối quan hệ thông qua thông qua cặp thuộc tính thảo khảo ngược – cài đặt bằng cách đặt các OIDs của các đối tượng lên quan trong phạm vi các đối tượng và duy trì tham chiếu để đối phó với nhiều phiên bản của cùng một đối tượng – một tính năng là điều cần thiết trong thiết kế và ứng dụng kỹ thuật. Ví dụ, một phiên bản cũ của một đối tượng đại diện cho một thiết kế kiểm tra và xác minh nên được giữ lại cho đến khi chỉ có một vài phiên bản mới của các đối tượng thành phần của nó, trong khi các thành phần khác vẫn không thay đổi.Ngoài ra phiên bản cho phép, các cơ sở dữ liệu OO cũng nên cho phép cho sự tiến hóa lược đồ, xảy ra khi các khai báo được thay đổi hoặc khi các kiểu mới hoặc các mối quan hệ được tạo ra. Hai tính năng này không cụ thể OODBs và lý tưởng nên được bao gồm trong tất cả các loại DBMSs.
* Nạp chồng toán tử, đề cập đến khả năng một phương thức được áp dụng cho các loại đối tượng khác nhau. Tính năng này còn được hiểu là toán tử có thể khác nhau trong cách thực hiện, tùy thuộc vào loại đối tượng ví dụ (tính chu vi của hình tam giác, hình tròn, hình chữ nhật).

1. **Định danh, cấu trúc, hàm dựng của đối tượng:**
2. **Định danh:**

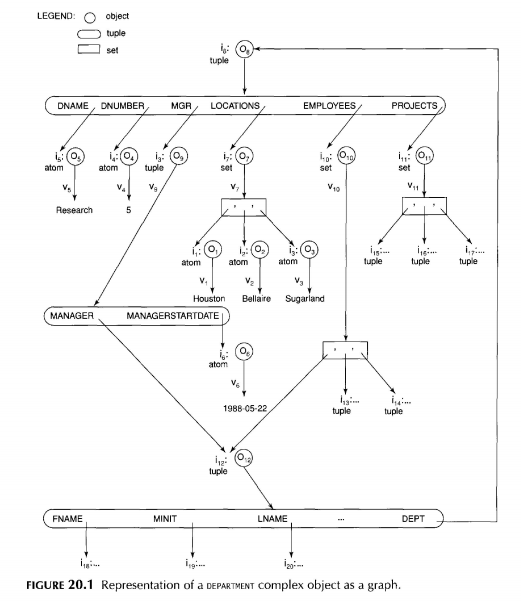
* Một hệ thống cơ sở dữ liệu hướng đối tượng cung cấp một nhận dạng duy nhất cho từng đối tượng độc lập được lưu trữ trong cơ sở dữ liệu. Nhận dạng duy nhất này thường được thực hiện thông qua một định danh duy nhất, đối tượng hệ thống tạo ra, hoặc OID. Giá trị của một OID là không hiển thị cho người dùng bên ngoài, nhưng nó được sử dụng trong nội bộ của hệ thống để xác định từng đối tượng duy nhất và để tạo và quản lý các tham chiếu đối tượng liên.OID có thể được gán cho các biến chương trình của các loại thích hợp khi cần thiết.
* Tính chất quan trọng và cần thiết của một OID đó là không thể thay đổi, đồng thời là  giá trị OID của một đối tượng cụ thể không nên thay đổi. Để giữ định danh  của đối tượng ở thế giới thực,  được đại diện. Do đó, một hệ thống cơ sở dữ liệu hướng đối tượng phải có một số cơ chế để tạo ra OIDs và lưu trữ bất biến. Đó cũng là mục đích rằng mỗi OID được sử dụng chỉ một lần, có nghĩa là, ngay cả khi đối tượng được lấy ra từ cơ sở dữ liệu, OID của nó không nên được gán cho một đối tượng khác. Hai tính chất này ngụ ý rằng OID không nên phụ thuộc vào bất kỳ giá trị thuộc tính của đối tượng, kể từ những  giá trị của một thuộc tính có thể được thay đổi hoặc sửa chữa. Nó cũng thường được xem là không phù hợp với cơ sở OID trên địa chỉ vật lý của các đối tượng trong lưu trữ, từ những địa chỉ vật lý có thể thay đổi sau khi tổ chức lại mức vật lý của cơ sở dữ liệu.Tuy nhiên, một số hệ thống sử dụng địa chỉ vật lý như OID để tăng hiệu quả thu hồi đối tượng. Nếu địa chỉ vật lý của những thay đổi đối tượng, một con trỏ gián tiếp có thể được đặt tại địa chỉ cũ, trong đó cung cấp cho location vật lý của đối tượng.Nó phổ biến hơn để sử dụng số nguyên dài như OIDs và sau đó sử dụng một số hình thức của bảng băm bản đồ giá trị OID địa chỉ vật lý hiện tại của các đối tượng trong lưu trữ.
* Một số mô hình  dữ liệu OO yêu cầu rằng tất cả mọi thứ - từ một giá trị đơn giản đến một đối tượng phức tạp - được thể hiện như một đối tượng, do đó, tất cả các giá trị cơ bản, chẳng hạn như một chuỗi, số nguyên, hoặc giá trị Boolean, có một OID. Điều này cho phép hai giá trị cơ bản để có OIDs khác nhau, có thể hữu ích trong một số trường hợp. Ví dụ giá trị 50 thỉnh thoảng là cân nặng của một người, đôi khi lại là tuổi của một người.

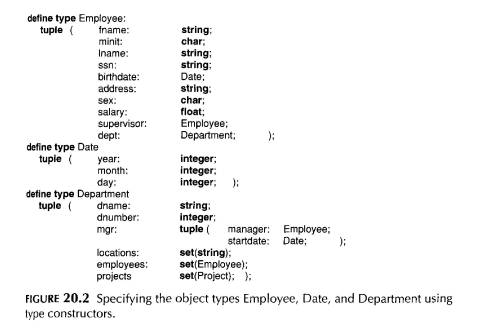
1. **Cấu trúc:**

* Trong cơ sở dữ liệu OO, state(giá trị hiện tại) của một đối tượng phức tạp có thể được xây dựng từ các đối tượng khác (hoặc các giá trị khác) bằng cách sử dụng một số kiểu hàm dựng. Một cách chính thức đại diện cho các đối tượng này để xem từng đối tượng như là một bộ ba(i, c, v), nơi mà i là một định danh đối tượng duy nhất (OID), c là một loại hàm tạo (có nghĩa là dấu hiệu của  đối tượng tạo như thế nào), và v là trạng thái đối tượng(hoặc giá trị hiện hành).Các mô hình dữ liệu thông thường sẽ bao gồm một số kiểu hàm dựng. Ba hàm dựng cơ bản nhất là atom, tuple, và set. Các hàm dựng khác thường được sử dụng bao gồm list, bag, và array. Các hàm dựng atom được sử dụng để đại diện cho tất cả các giá trị nguyên tử cơ bản, chẳng hạn như số nguyên, số thực, chuỗi ký tự, Booleans,và bất kỳ dữ liệu cơ bản khác mà hệ thống hỗ trợ trực tiếp.
* Một thể hiện v của một đối tượng (i, c, v) được giải thích dựa trên loại hàm dựng c.
  + Nếu c = atom thì thể hiện(value) v là một giá trị nguyên tử từ miền giá trị mà hệ thống hỗ trợ.
  + Nếu c= set, thì thể hiện (value) v là tập đối tượng xác định {i1, i2, …, in} với các OIDs từ <a1:i1, a2:i2, …, an:in>, mỗi aj là một tên giá trị với mỗi i­­I là một OID.
  + Nếu c = tuple thì thể hiện dữ liệu v của một bộ từ <a1:i1, a2:i2, …, an:in> với mỗi ai là một giá trị tên và mỗi ii là một OID.
  + Nếu c = list, thì giá trị của v là một ordered list [i1, i2, …, in] thuộc OIDs của một đối tượng có cùng loại. Một list thì giống như set gồm các ngoại trừ các OID.
  + Cho c = array, thì thể hiện dữ liệu là một list đơn một chiều, sự khác nhau giữa array và list là một list thì số phần tử có thể tùy ý, nhưng mảng thì thường có kích thước cố định. Đồng thời sự khác biệt giữa set và bag là các phần tử trong một set thì phải khác nhau, nhưng bag thì không(có thể trùng lắp)

1. **Hàm dựng:**

* Một ngôn ngữ định nghĩa đối tượng (ODL) kết hợp các hàm dựng có trước có thể được sử dựng để xác định ODL tiêu chuẩn của ODMG, đầu tiên chúng tôi sẽ giới thiệu dần trong phần này bằng cách sử dụng các ký hiệu đơn giản.Các loại hàm dựng sẽ được sử dụng để xác định cấu trúc dữ liệu cho một lược đồ cơ sở dữ liệu hướng đối tượng. Trong phần 20.3 sẽ xem xét làm thế nào để kết hợp cac định nghĩa của phương thức cho một sơ đồ OO. Hình 20.2 được định nghĩa là một bộ chứ không phải là một giá trị nguyên tử(atom) như trong hình 20.1. Chúng tôi sử dụng các khóa tuple, set, và list cho các hàm dựng và loại dữ liệu chuẩn (int, string, float) cho các loại atom.





1. **Tính đóng gói của toán tử, phương thức, lâu dài**

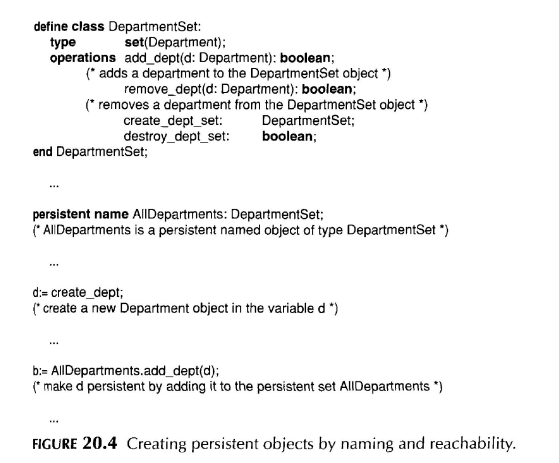
* Tính đóng gói là một trong những đặc điểm chính của ngôn ngữ và hệ thống OO. Nó cũng liên quan đến các khái niệm về kiểu dữ liệu trừu tượng và ẩn dữ liệu bằng các ngôn ngữ lập trình. Trong các mô hình dữ liệu truyền thống khái niệm này không được sử dụng, vì nó là cách để làm cho cấu trúc của cơ sở dữ liệu của các đối tượng hiển thị ho người dùng và chương trình bên ngoài. Trong mô hình truyền thống, một số phương thức được áp dụng với tất cả các loại đối tượng. Ví dụ như chèn, xóa, sửa các bộ dữ liệu trong mô hình quan hệ. Mối quan hệ và các thuộc tính của nó được hiển thị cho người dùng và các chương trình bên ngoài truy cập vào các mối quan hệ sử dựng các phương thức này.

1. **Xác định phương thức của đối tượng thông qua các lớp:**

* Các khái niệm ẩn thông tin và đóng gói có thể được áp dụng cho các đối tượng cơ sở dữ liệu. Ý tưởng chính là để xác định hành vi của một loại đối tượng dựa trên các hoạt động bên ngoài có thể được áp dụng cho các đối tượng của loại hình đó. Cấu trúc bên trong của đối tượng được ẩn, và các đối tượng chỉ có thể truy cập thông qua một số phương thức được xác định trước. Một số phương thức có thể được sử dụng để tạo ra (insert) hoặc hủy đối tượng, các hoạt động khác có thể cập nhật trạng thái đối tượng, và những người khác có thể được sử dụng để lấy các bộ phận của đối tượng hoặc áp dụng một số tính toán.Các hoạt động khác vẫn có thể thực hiện một sự kết hợp thu hồi, tính cập nhật. Nói chung, việc thực hiện của một hoạt động có thể được xác định trong một ngôn ngữ lập trình mục đích chung cung cấp sự linh hoạt và sức mạnh trong việc xác định các hoạt động.
* Người sử dụng chỉ được thực hiện nhận thấy của giao diện của các loại đối tượng, qua đó xác định tên và đối số (tham số) của từng phương thức. Sự thực thi được ẩn đối với người dùng, nó bao gồm các định nghĩa của cấu trúc dữ liệu bên trong. Trong thuật ngữ OO \, phần giao diện của mỗi hoạt động được gọi là chữ ký, và thực hiện hoạt động được gọi là một phương thức. Thông thường, một phương thức được gọi bằng cách gửi một thông điệp tới đối tượng để thực hiện phương thức thông báo tương ứng, như là một phần của thực thi một phương thức, một thông báo sau khi một đối tượng khác có thể được gửi, và cơ chế này có thể được sử dụng để trả lại giá trị từ các đối tượng bên ngoài hoặc cho các đối tượng khác.
* Trong hầu hết các trường hợp, các hoạt động cập nhật trạng thái của một đối tượng được đóng gói. Đây là một cách xác định các ngữ nghĩa cập nhật của lược đồ. Mỗi loại đối tượng có tính toàn vẹn của nó ràng buộc lập trình vào các phương pháp tạo, xóa, và cập nhật các đối tượng dứt khoát viết mã để kiểm tra đối với các hành vi vi phạm ràng buộc và xử lý các trường hợp ngoại lệ.Trong trường hợp này, tất cả các hoạt động cập nhật được thực hiện bởi các hoạt động đóng gói. Gần đây, ODL cho các tiêu chuẩn ODMG cho phép các đặc điểm kỹ thuật của một số khó khăn chung như các khóa và các mối quan hệ nghịch đảo (Làm rõ tham chiếu) để hệ thống tự động có thể thực thi những hạn chế.

1. **Xác định tính lâu dài của đối tượng thông qua đặt tên và tiếp cận:**

* Một OODBMS thường gắn với một OOPL. OOPL được sử dụng để xác định triển khai phương pháp cũng như các mã ứng dụng khác. Một đối tượng thường được tạo ra bởi một số chương trình ứng dụng thực hiện, bằng cách gọi hàm xây dựng đối tượng. Không phải tất cả các đối tượng có nghĩa là để được lưu giữ vĩnh viễn trong cơ sở dữ liệu. Đối tượng thoáng qua tồn tại trong chương trình thực hiện và biến mất một khi chương trình chấm dứt. Đối tượng liên tục được lưu trữ trong cơ sở dữ liệu và vẫn tồn tại sau khi kết thúc chương trình. Các cơ chế điển hình để làm cho một liên tục đối tượng được đặt tên và tiếp cận.
* Cơ chế đặt tên liên quan đến việc đưa ra một đối tượng một tên duy nhất lâu dàithông qua đó nó có thể được lấy ra bằng cách này và chương trình khác.Tên đối tượng liên tục có thể được thông qua một tuyên bố hoặc hoạt động cụ thể trong chương trình, như minh họa trong hình 20,4.Tất cả các tên cho các đối tượng phải là duy nhất trong một cơ sở dữ liệu cụ thể. Do đó, đối tượng tên là liên tục được sử dụng như các điểm đưa vào các cơ sở dữ liệu thông qua đó người sử dụng và các ứng dụng có thể bắt đầu truy cập cơ sở dữ liệu của họ. Hiển nhiên, nó là không thực tế để cung cấp cho được thực hiện liên tục bằng cách sử dụng cơ chế thứ hai, gọi là tiếp cận. Cơ chế tiếp cận hoạt động bằng cách làm cho đối tượng có thể truy cập từ một số đối tượng liên tục. Một đối tượng B được cho là có thể truy cập từ một đối tượng nếu một chuỗi các tham chiếu trong đồ thị đối tượng đầu từ đối tượng đến đối tượng B. Ví dụ, tất cả các đối tượng trong hình 20,1 truy cập từ đối tượng 08 do đó, nếu 08 được thực hiện lâu dài, tất cả các đối tượng khác trong hình 20,1 cũng trở nên lâu dài.
* Nếu chúng ta đầu tiên tạo ra một lâu dài đặt tên bằng cách thêm chúng vào các set hoặc list, và một class C, chúng ta có thể làm cho các đối tượng của C lâu dài bằng cách thêm chúng vào các set hoặc list, và do đó làm cho chúng tiếp cận từ N. Do đó, N xác định một bộ lâu dài của các đối tượng của lớp C.



* Ví dụ, chúng ta có thể định nghĩa một lớp DepartmentSet (xem hình 20,4) có đối tượng được thiết lập loại(Department).Giả sử rằng một đối tượng của loại DeparmentSet là tạo ra, và giả sử rằng nó được đặt tên là AllDepartment và do đó được thực hiện liên tục, như được minh họa trong hình 20,4.Bất kỳ đối tượng cục được thêm vào các thiết lập củaAllDepartment bằng cách sử dụng các hoạt động add\_dept trở thành lâu dài tục nhờcó thể truy cập được từ ALlDepartment.Các đối tượng Tất cả các Bộ thường được gọi là mức độ của lớp Department1, vì nó sẽ giữ tất cả các đối tượng liên tục củakiểu Department1.Như chúng ta sẽ thấy trong Chương 21, ODMG ODL tiêu chuẩncung cấp cho các nhà thiết kế giản đồ tùy chọn đặt tên một mức độ như là một phầnđịnh nghĩa lớp.
* Chú ý sự khác biệt giữa các mô hình cơ sở dữ liệu truyền thống và cơ sở dữ liệu hướng đối tượng trong lĩnh vực này.Trong các mô hình cơ sở dữ liệu truyền thống, chẳng hạn như mô hình quan hệ hoặc mô hình EER, tất cả các đối tượng được cho là lâu dài. Do đó, khi một loại thực thể, lớp, chẳng hạn như NHÂN VIÊN được định nghĩa trong mô hình EER, nó đại diện cho cả khai báo kiểu cho nhân viên và một tậplâu dài của tất cả các đối tượng NHÂN VIÊN. Trong cách tiếp cận OO, một khai báo lớp của nhân viên và thiết lập một liên tục của tất cả các đối tượng EMPLOYEE. Trong cách tiếp cận OO, một khai báo lớp của người lao động quy địnhcụ thể chỉ có một loại và hoạt động cho một lớp học của các đối tượng. Người sử dụng phải xác định riêng một đối tượng liên tục thiết lập loại (NHÂN VIÊN) hoặc danh sách (nhân viên) có giá trị là tập hợp các tài liệu tham khảo cho tất cả các đối tượngNHÂN VIÊN liên tục, nếu điều này là mong muốn, như minh họa trong hình 20,4. Điều này cho phép các đối tượng thoáng qua và liên tục để theo cùng loại và tờ khai lớp học của ODL và OOPL. Nói chung, nó có thể để xác định các tập liên tục cho cùng một định nghĩa lớp, nếu muốn.

1. **Lớp phân cấp và lớp kế thừa**

* Một đặc điểm chính của hệ thống cơ sở dữ liệu OO là họ cho phép loại và phân cấpthừa kế. Hệ thống phân cấp loại trong cơ sở dữ liệu thường bao hàm sự hạn chế về mức độ tương ứng với các kiểu trong hệ thống phân cấp.Chúng tôi thảo luận về các loại  hệ thống phân cấp  (trong phần 1), và sau đó các hạn chế về mức độ (tại mục 2).

1. **Khái niệm**

* Trong hầu về về ứng dụng cơ sở dữ liệu, có nhiều đối tượng về cùng loại hay lớp. Do đó, cơ sở dữ liệu hướng đối tượng phải cung cấp một khả năng phân loại đối tượng dựa trên loại có, cũng như hệ về cơ sở dữ liệu khác. Tuy nhiên, trong cơ sở dữ liệu hướng đối tượng, như yêu cầu thêm rằng hệ thống cho phépđịnh về của các loại mới dựa trên các loại khác được xác định trước, dẫn đến một loại (hoặc lớp) hệ thống phân cấp.
* Thông thường, một kiểu được định nghĩa bằng cách gán cho nó một tên loại và sau đó xác định một số các thuộc tính (các biến chẳng hạn) và các hoạt động(phương thức) đối với loại.Trong một số trường hợp, các thuộc tính và hoạt độngvới nhau được gọi là chức năng, từ khi các thuộc tính giống như chức năng với các đối số không.Một tên chức năng có thể được sử dụng để tham khảo giá trịcủa một thuộc tính hoặc tham khảo giá trị kết quả của một hoạt động (phương thức).Trong phần này, chúng tôi sử dụng các chức năng hạn để tham khảo cho cả hai thuộc tính và hoạt động của một loại đối tượng, kể từ khi họ được đối xử tương tự trong một giới thiệu cơ bản để thừa kế.
* Một loại ở dạng đơn giản có thể được xác định bằng cách đặt tên và liệt kế các chức năng(public). Khi xác định loại trong thành phần này, bài này sử dụng mẫu sau đây, không xác định đối số của các chức năng, để đơn giản hóa các ví dụ:
  + TYPE\_NAME: function, function, …, function
  + Ex:
    - PERSON: Name, Address, Birthday, Age, SSN.
  + Trong loại PERSON, tên, địa chỉ, và các chức năng sinh có thể được thực hiện như là thuộc tính được lưu trữ, trong khi chức năng Age có thể được thực hiện như một phương pháp tính toán thời từ giá trị của thuộc tính Ngày sinh và ngàyhiện tại.
  + Ở đây khái niệm kiểu phụ (subtype) rất hữu ích khi thiết kế hoặc người sử dụng phải tạo ra một loại mới tương tự nhưng không giống với loại đã xác định. Kiểu phụ này sau đó sẽ thừa hưởng tất cả chức năng của các loại trước mà ta sẽ gọi là subtype. Ví dụ như định nghĩa 2 loại mới là STUDENT và EMPLOYEE:
    - EMPLOYEE: Name, Address, Birthdate, Age, SSN, Salary, HireDate, Seniority.
    - STUDENT: Name, Address, Birthdate, Age, SSN, Major, GPA.
  + Thay vì như thế ta sẽ cho 2 loại này kế thừa từ PERSON:
    - EMPLOYEE **subtype of** PERSON: Salary, HireDate, Seniority.
    - STUDENT **subtype of** STUDEN: Major, GPA.
  + Nói chung, một loại con (subtype) bao gồm tất cả các chức năng được định nghĩa cho loại cha của nó công với một số chức năng bổ sung cụ thể duy nhất để phân loại. Do đó, nó có thể tạo ra một hệ thống phân cấp loại để hiển thị mối quan hệ (loại cha, loại con) trong số tất cả các khai báo trong hệ thống.
  + Thông báo định nghĩa kiểu tat mô tả đối tượng nhưng không tạo ra các đối tượng riêng của họ chỉ là tờ khai của một số loại, và như là một phần của rằng việc khai báo, việc triển khai của một số loại, và như là một phần của việc khai báo, việc thực hiện các chức năng của từng loạiđược quy định.Trong một ứng dụng cơ sở dữ liệu, có nhiều đối tượng của từng loại. Khi một đối tượng được tạo ra, nó thường thuộc về một hoặc nhiều các loại đã được khai báo.

1. **Ràng buộc về mức độ tương ứng của hệ thống phân cấp**

* Trong hầu hết các cơ sở dữ liệu hướng đối tượng, tập các đối tượng trong một mức độ có cùng loại hoặc lớp. Tuy nhiên, đây không phải là điều kiện cần thiết. Ví dụ, Smalltalk, cái gọi là ngôn ngữ OO typeless, cho phép một bộ sưu tập của các đối tượng có chứa các đối tượng của các loại khác nhau. Điều này cũng có thể là trường hợp khi không hướng đối tượng ngôn ngữ khác typeless, chẳng hạn như LISP, được mở rộng với các khái niệm OO. Tuy nhiên, vì phần lớn các loại OO cơ sở dữ liệu hỗ trợ, trong bài này giả định rằng mức độ là tập các đối tượng của loại OO cơ sở dữ liệu hỗ trợ, và mức tập của các đối tượng cùng loại để còn lại của phần này.
* Nó được phổ biến trong các ứng dụng cơ sở dữ liệu từng loại hoặc kiểu phụ sẽ có một mức độ liên kết với nó, nắm giữ các bộ sưu tập của tất cả các đối tượng lâu dài của loại hoặc kiểu phụ.Trong trường hợp này, hạn chế là tất cả các đối tượng trong một mức độ tương ứng với một kiểu phụ cũng phải là một thành viên của các mức tương ứng với supertype của nó. Một số hệ thống cơ sở dữ liệu OO có một loại hệ thống được xác định trước (gọi là lớp tế ROOT hoặc lớp OBJECT). Có mức độ chứa tất cả các đối tượng trong hệ thống. Phân loại sau đó tiến hành bằng cách chỉ định các đối tượng vào phân nhóm bổ sung có ý nghĩa cho các ứng dụng, tạo ra một loại hệ thống phân cấp hoặc hệ thống cấp bậc lớp cho hệ thống.Tất cả các mức độ cho hệ thống và các lớp người dùng định nghĩa là các tập con của mức độ tương ứng với đối tượng lớp, trực tiếp hoặc gián tiếp.Trong mô hình ODMG (xem Chương 21), người sử dụng có thể hoặc không thể xác định một mức độ cho từng loại (type), tùy thuộc vào ứng dụng.
* Trong hầu hết các hệ thống hướng đối tượng, sự phân biệt được tạo ra giữa các đối tượng lâu dài và thoáng qua và tập. Một tập lâu dài nắm giữ một tập của các đối tượng được lưu trữ vĩnh viễn trong cơ sở dữ liệu và do đó có thể được truy cập và chia sẻ bởi nhiều chương trình.Một tập tạm thời tồn tại tạm thời trongquá trình thực hiện chương trình một chút không được giữ lại khi chương trìnhchấm dứt.Ví dụ, một tập thoáng qua.Các bộ sưu tập tạm thời nắm giữ cùng loạicủa các đối tượng là bộ sưu tập liên tục.Chương trình sau đó có thể thao tác các đối tượng trong bộ sưu tập thoáng qua, và một khi chương trình chấm dứt, bộ sưu tập tạm thời chấm dứt tồn tại.Nói chung, cáctập rất nhiều – tạm thời hay lâu dài - có thể chứa các đối tượng cùng loại.

1. **Đối tượng phức tạp**

* Một động lực chính dẫn đến sự phát triển của hệ thống OO là mong muốn để đại diện cho đối tượng phức tạp. Có hai loại chính của các đối tượng phức tạp: cấu trúc và phi cấu trúc. Một đối tượng có cấu trúc phức tạp được tạo thành từ các thành phần và được xác định bằng cách áp dụng xây dựng các loại đệ quy ở các cấp độ khác nhau. Một đối tượng không có cấu trúc phức tạp thường là một kiểu dữ liệu yêu cầu một lượng lớn dung lượng lưu trữ, chẳng hạn như một kiểu dữ liệu đại diện cho một hình ảnh hoặc một đối tượng văn bản lớn.

1. **Đối tượng không có cấu trúc và khả năng mở rộng**
2. Top of Form

* Một đối tượng cơ sở không có cấu trúc phức tạp được cung cấp bởi một DBMS cho phép lưu trữ và phục hồi của các đối tượng lớn là cần thiết bởi các ứng dụng cơ sở dữ liệu.Ví dụ điển hình của các đối tượng đó là những hình ảnh bitmap và các chuỗi văn bản dài (chẳng hạn như file văn bảng), họ còn được gọi là đối tượng nhị phân lớn, hoặc BLOB. Chuỗi ký tự còn được gọi là các DBMS không biết cấu trúc của chúng là gì - chỉ có các ứng dụng sử dụng chúng có thể giải thích ý nghĩa của chúng.Ví dụ, các ứng dụng có thể có chức năng hiển thị một hình ảnh hoặc tìm kiếmcác từ khóa nhất định trong một chuỗi văn bản dài.Các đối tượng được coi là phức tạp bởi vì họ yêu cầu một khu vực rộng lớn của lưu trữ và không phải là một phần củacác loại dữ liệu tiêu chuẩn được cung cấp bởi truyền thống DBMS.Bởi vì kích thước đối tượng khá lớn, một DBMS có thể lấy một phần của đối tượng và cung cấp nó chocác chương trình ứng dụng trước khi toàn bộ đối tượng được lấy. DBMS cũng có thể sử dụng các kỹ thuật đệm và bộ nhớ đệm prefetch phần của đối tượng trước khi các chương trình ứng dụng cần để truy cập chúng.

1. Bottom of Form

* Các phần mềm DBMS không có khả năng trực tiếp xử lý lựa chọn và các hoạt động khác dựa trên các giá trị của các đối tượng này, trừ khi ứng dụng cung cấp mã để làm các hoạt động so sánh cần thiết để lựa chọn.Trong một OODBMS, điều này có thể được thực hiện bằng cách xác định kiểu dữ liệu trừu tượng mới cho các đối tượng không giải thích và bằng cách cung cấp các phương pháp để lựa chọn, so sánh, và hiển thị các đối tượng như vậy.Ví dụ, hãy xem xét các đối tượng là hai hình ảnh bitmap chiều. Giả sử rằng ứng dụng cần để lựa chọn từ một tập hợp các đối tượng này chỉ bao gồm một khuôn mẫu nhất định.Trong trường hợp này, người sử dụng phải cung cấp các chương trình nhận dạng mẫu như một phương thức trên đối tượng của loại hình bitmap. OODBMS Các lấy một đối tượng từ cơ sở dữ liệu và chạy các phương pháp nhận dạng mẫu vào nó để xác định xem đối tượng bao gồm các mô hình cần thiết.
* Bởi vì một OODBMS cho phép người dùng tạo ra các loại mới, và bởi vì loại bao gồm cả cấu trúc và hoạt động, chúng ta có thể xem một OODBMS như có một kiểu hệ thống mở rộng.Chúng ta có thể tạo ra các thư viện của các loại mới bằng cách xác định cấu trúc và hoạt động của họ, bao gồm các loại phức tạp.Các ứng dụng sau đó có thể sử dụng hoặc sửa đổi các loại, trong trường hợp này bằng cách tạo ra cácphân nhóm của các loại được cung cấp trong các thư viện.Tuy nhiên, - bên trong DBMS phải cung cấp cho lưu trữ cơ bản và khả năng thu hồi cho các đối tượng yêu cầu số lượng lớn dung lượng lưu trữ để các hoạt động có thể được áp dụng mộtcách hiệu quả.Nhiều OODBMSs cung cấp cho việc lưu trữ và phục hồi của các đối tượng lớn không có cấu trúc như chuỗi ký tự hoặc các chuỗi bit, có thể được thông qua "như là" các chương trình ứng dụng để giải thích.Gần đây, DBMSs quan hệ vàmở rộng quan hệ cũng có thể cung cấp khả năng như vậy.Kỹ thuật lập chỉ mục đặc biệt cũng đang được phát triển.

1. **Đối tượng có cấu trúc**

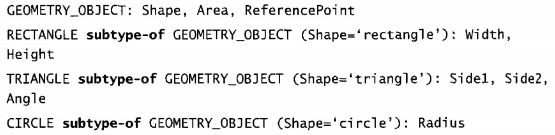
* Một đối tượng cấu trúc phức tạp khác nhau từ một đối tượng không có cấu trúc phức tạp mà cấu trúc của đối tượng được xác định bởi ứng dụng lặp đi lặp lại của – hàm dựng các loại cung cấp bởi OODBMS. Do đó, cấu trúc đối tượng được định nghĩa và được biết đến với OODBMS. Ví dụ, hãy xem xét - đối tượng DEPARTMENT thể hiện trong hình 20.1. Ở cấp độ đầu tiên, đối tượng đã tuple một cấu trúc với sáu thuộc tính cụ thể là, DNAME và DNUMBER - có giá trị cơ bản, các cấu trúc phức tạp và do đó xây dựng mức độ thứ hai của - NHÂN VIÊN đối tượng phức tạp, dự án) đã thiết lập cấu trúc.Ở cấp độ thứ ba, cho một giá trị tuple Mgr, chúngtôi có một thuộc tính cơ bản (MANAGERSTARDATE) và thuộc tính (MANAGER)đề cập đến một đối tượng nhân viên, trong đó có một cấu trúc tuple.Đối với một số vị trí, chúng tôi có một tập hợp các giá trị cơ bản, nhưng đối với cả nhân viênvà các bộ dự án, chúng tôi có bộ tuple-cấu trúc đối tượng.
* Là một phần của mối quan hệ ngữ nghĩa quyền sở hữu đối tượng phức tạp để xây dựng tài sản mà các đối tượng thành phần được đóng gói trong các đối tượng phức tạp và được coi là một phần của nhà nước nội bộ đối tượng. Họ không cần phải cónhận dạng đối tượng và chỉ có thể được truy cập bằng phương pháp của đối tượng đó. Họ sẽ bị xóa nếu itselt đối tượng bị xóa. Mặt khác, thành phần tham chiếu được coi là các đối tượng indepentdent có thể có bản sắc riêng của họ và phương pháp. Khi một đối tượng phức tạp cần để truy cập các thành phần tham chiếu của nó,nó phải làm như vậy bằng cách gọi các phương pháp thích hợp của các thành phần, vì chúng là không gói gọn trong đối tượng phức tạp. Do đó, ngữ nghĩa tham khảo đạidiện cho mối quan hệ giữa các đối tượng độc lập. Ngoài ra, một đối tượng tham chiếu thành phần có thể được tham chiếu nhiều hơn so với đối tượng phức tạp và do đó không phải là tự động bị xóa khi các đối tượng phức tạp sẽ bị xóa.
* Một hệ quản trị CSDL hướng đối tượng nên cung cấp cách lưu trữ các thành phần phân nhóm các đối tượng thành phần của một đối tượng phức tạp với nhau về lưu trữ thứ cấp để nâng cao hiệu quả của các hoạt động truy cập vào các đối tượng phức tạp. Trong nhiều trường hợp, cấu trúc đối tượng được lưu trữ trên các trang đĩa khác nhau. Khi một trang đĩa bao gồm một đối tượng được lấy vào bộ nhớ, OORDBMS có thể xây dựng các đối tượng có cấu trúc phức tạp từ những thông tin trên các đĩa, có thể tham khảo các trang đĩa bổ sung phải được lấy ra. Điều này được gọi là lắp ráp đối tượng.

1. **Các khái niệm khác của hướng đối tượng**

* Trong chương này, sẽ giới thiệu tổng quan về các khái nhiệm mở rộng của hướng đối tượng, bao gồm tính đa hình (nạp chồng toán tử), đa kế thừa, lựa chọn kế thừa, tính phiên bản và cấu hình.

1. **Đa hình(nạp chồng toán tử)**

* Một đặc tính của hệ thống OO là cung cấp cho tính chất đa hình hoạt động, còn được gọi là nạp chồng toán tử. Khái niệm này cho phép các toán tử cùng tên hoặc biểu tượng bị ràng buộc vào hai hay nhiều hiện thực khác nhau của toán tử. Tùy thuộc vào đối tượng mà toán tử được áp dụng. Một ví dụ đơn giản từ các ngôn ngữ lập trình có thể minh họa cho khái niệm này. Trong số ngôn ngữ, biểu tượng toán tử “+” có thể có nghĩa khác nhau khi các toán hạng khác nhau. Trình biên dịch có thể xác định hoạt động để thực hiện dựa trên hoạt động để thực hiện dựa trên các loại toán hạng được cung cấp. Trong OO databased, một tình huống tương tự có thể xảy ra. Chúng ta có thể sử dụng các ví dụ GEOMETRY\_OBJECT thảo luận tại mục 20.4 để minh họa tính đa hình trong cơ sở dữ liệu OO. Giả sử rằng chúng ta khai báo GEOMETRY\_OBJECT và các phân nhóm:

****

* Ở đây hàm Arena(tính diện tính) được khai báo cho tất cả đối tượng GEOMETRY\_OBJECT. Tuy nhiên, việc thực hiện của phương pháp cho khu vực có thể khác nhau cho mỗi kiểu phụ thuộc của GEOMETRY\_OBJECT. Một khả năng là có cách thực hiện chung cho việc tính toán diện tích của một GEOMETRY\_OBJECT tổng quát(ví dụ, viết một thuật toán chung để tính diện tích của một đa giác) và sau đó viết loại các thuật toán hiệu quả hơn để tính cho từng hình cụ thể.
* Và giờ OODBMS phải lựa chọn phương pháp thích hợp cho các khu vực dựa vào đối tượng hình hoạc mà nó được áp dụng. Trong các hệ thống mạnh mẽ, điều này có thể được thực hiện tại thời gian biên dịch, kể từ khi các đối tượng phải được biết đến, trong các hệ thống với cách thực hiện yếu(như Smalltalk và LISP), có loại của đối tượng có chức năng ứng dụng có thể không được biết đến cho đến khi chạy runtime. Trong trường hợp này, hàm phải kiểm tra các đối tượng trong thời gian chạy và sau đó gọi các phương pháp thích hợp. Đây được gọi là ràng buộc động.

1. **Đa kế thừa và kế thừa có chọn lọc**

* Đa kế thừa trong một hệ thống phân cấp xảy ra khi một đối tượng phân nhóm nào đó là một subtype của hai (hoặc hơn) loại và do đó kế thừa chức năng(thuộc tính) của cả hai supertypes. Ví dụ, ta có thể tạo ra một ENGINEERING\_MANAGER subtype là một phân nhóm của cả hai loại MANAGER và ENGINEER. Điều này dẫn đến việc tạo ra một kiểu mạng lưới chứ không phải là loại phân cấp. Một vấn đề có thể xảy ra với kế thừa nhiều là các supertype mà từ đó kế thừa kiểu phụ có thể là chức năng riêng biệt nhưng cùng tên, tạo ra một sự mơ hồ. Ví dự cả 2 MANAGER và ENGINEER điều có tiền lương. Nếu chức năng mức lương được thực hiện bằng phương pháp khác nhau trong siêu kiểu MANAGER và ENGINEERING\_MANAGER subtype. Tuy nhiên, có thể là MANAGER và ENGINEER thừa hưởng lương từ supertype tượng tự Nguyên tắc chung là nếu một chức năng được kế thừa từ supertype phổ biến, sau đó nó được kế thừa chỉ một lần. Trong trường hợp này sự mơ hồ không có, vấn đề chỉ phát sinh nếu các chức năng riêng biệt trong hai siêu kiểu.

1. **Các phiên bản và cấu hình**

* Nhiều ứng dụng CSDL sử dụng hệ thống hướng đối tượng đòi hỏi sự tồn tại của một số phiên bản của cùng một đối tượng. Ví dụ, hãy xem xét một ứng dụng cơ sở dữ liệu cho một môi trường kỹ thuật phần mềm mà các việc lưu trữ các phần mềm khác nhau, chẳng hạn như các mô-đun thiết kế, mô-đun mã nguồn, và thông tin cấu hình để mô tả những mô-đun được liên kết với nhau để tạo thành một chương trình phức tạp và trường hợp thử nghiệm để thử nghiệm hệ thống. Thông thường, các hoạt động bảo dưỡng được áp dụng cho một hệ thống phần mềm theo yêu cầu của nó phát triển. Việc bảo trì thường liên quan đến việc thay đổi một số các mô-đun thiết kế và thực hiện. Nếu hệ thống đã hoạt động, và nếu một hoặc nhiều các mô-đun phải được thay đổi, người thiết kế nên tạo ra một phiên bản mới của các mô-đun để kiểm tra phiên bản mới của các mô-đun. Tuy nhiên, các phiên bản hiện tại không nên được loại bỏ cho đến khi các phiên bản mới đã được kiểm tra kỹ lưỡng và đã được phê duyệt, chỉ sau đó các phiên bản mới sẽ thay thế những phiên bản củ.
* Chú ý rằng có thể có được nhiều hơn hai phiên bản của một đối tượng. Ví dụ, hãy xem xét hai lập tình viên làm việc để cập nhật các mô-đun phần mềm tương tự đồng thời. Trong trường hợp này, hai phiên bản, ngoài các mô-đun ban đầu là cần thiết. Những người lập trình có thể cập nhật các phiên bản của cùng một module phần mềm đồng thời. Điều này thường được gọi là kỹ thuật đồng thời. Tuy nhiên cuối cùng trở nên cần thiết để hợp nhất hai phiên bản này lại với nhau để phien bản mới (hybrid) có thể bao gồm các thay đổi được thực hiện bởi cả hai lập trình viên. Trong quá trình sát nhập, nó cũng cần thiết để làm cho chắc chắn rằng những thay đổi của họ tương thích. Điều này đòi hỏi phải tạo ra một phiên bản của đối tượng.Một trong đó là kết quả của việc sát nhập hai phiên bản độc lập cập nhật.

1. **Tổng kết**

* Trong chương này, chủ yếu thảo luận về các khái niệm phương pháp tiếp cận đối tượng cho hệ thống cơ sở dữ liệu, đã được đề xuất để đáp ứng nhu cầu của các ứng dụng cơ sở dữ liệu phức tạp và thêm các chức năng cơ sở dữ liệu với các ngôn ngữ lập trình hương đối tượng C++. Chúng tối lần đầu tiên thảo luận về các khái niệm chính được sử dụng trong cơ sở dữ liệu hướng đối tượng, trong đó bao gồm các điều sau đây:
  + Object identity: Mỗi đối tượng có một giá trị nhận diện đối tượng duy nhất và độc lập với giá trị của nó.
  + Hàm dựng kiểu dữ liệu: Đối tượng phức tạp có hàm dựng bằng cách đệ qui áp dụng một thiết kế , list, bag, set, tuple.
  + Tính đóng gói: Cả cấu trúc đối tượng, và phương thức có thể áp dụng cho đối tượng đều được bao gồm trong một định nghĩa class.
  + Khả năng tương thích ngôn ngữ lập trình.
  + Tính ké thừa: Một đối tượng có thể sử dụng lại các thuộc tính hay phương thức của đối tượng định nghĩa trước nó thông qua tính kế thừa.
  + Mở rộng: Tất cả những đốit tượng lâu dài của một loại đặc biệt có thể được lưu trữ ở một mức độ. Mức độ tương ứng với hệ thống phân cấp của loại đã thiết lập/ tập hợp con ràng buộc trên chúng
  + Hỗ trợ các loại index phức tạp.
  + Đa hình và nạp chồng toán tử.
  + Phiên bản