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

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:**

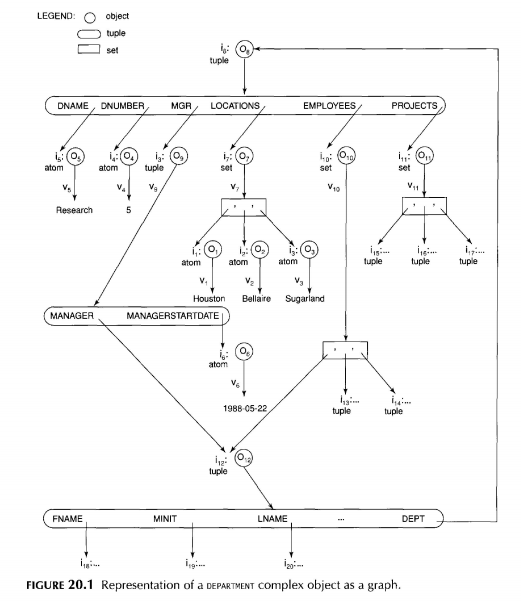
* Một hệ thống cơ sở dữ liệu hướng đối tượng cung cấp một nhận dạng duy nhất cho từng đối tượng độc lập được lưu trữ trong cơ sở dữ liệu. Nhận dạng duy nhất này thường được thực hiện thông qua một định danh duy nhất, đối tượng hệ thống tạo ra, hoặc OID. Giá trị của một OID là không hiển thị cho người 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.

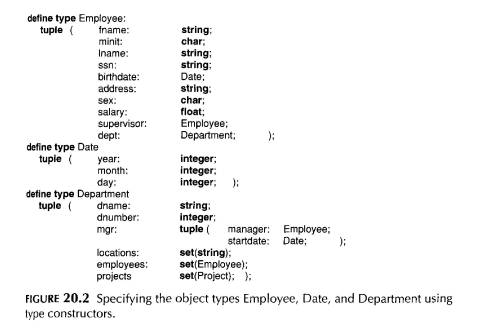
1. **Cấu trúc:**

* 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**

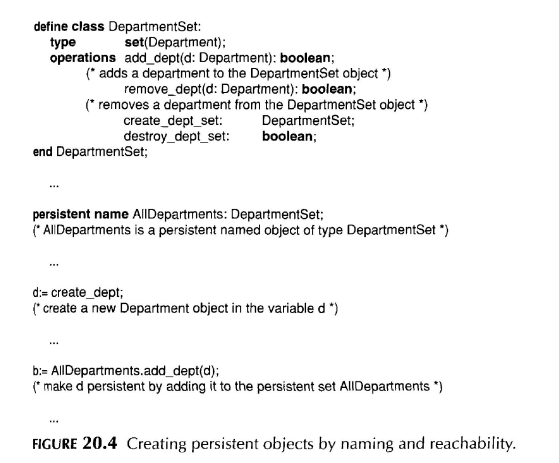
* Tính đóng gói là một trong những đặc điểm chính của ngôn ngữ và hệ thống OO. Nó cũng liên quan đến các khái niệm về kiểu dữ liệu trừu tượng và ẩn dữ liệu bằng các ngôn ngữ lập trình. Trong các mô hình dữ liệu truyền thống khái niệm này không được sử dụng, vì nó là cách để làm cho cấu trúc của cơ sở 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**

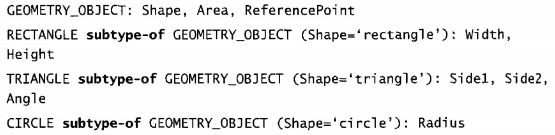
* Một đối tượng cấu trúc phức tạp khác nhau từ một đối tượng không có cấu trúc phức tạp mà cấu trúc của đối tượng được xác định bởi ứng dụng lặp đi lặp lại của – hàm dựng các loại cung cấp bởi OODBMS. Do đó, cấu trúc đối tượng được định nghĩa và được biết đến với OODBMS. Ví dụ, hãy xem xét - đối tượng 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