**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 thứu, 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 lacation 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

1. **Hàm dựng:**