HOÀNG BẢO HÙNG

CƠ SỞ DỮ LIỆU HƯỚNG ĐỐI TƯỢNG

NHÀ XUẤT BẢN THÔNG TIN VÀ TRUYỀN THÔNG

HOÀNG BẢO HÙNG

CƠ SỞ DỮ LIỆU HƯỚNG ĐỐI TƯỢNG

NHÀ XUẤT BẢN THÔNG TIN VÀ TRUYỀN THÔNG

BẢNG VIẾT TẮT CÁC THUẬT NGỮ

ACID (Atomicity, Consistency, Isolation, Durability)	Tính nguyên tử, tính nhất quán, tính biệt lập, tính bền vững
DCL (Data Control Language)	Ngôn ngữ điều khiển dữ liệu
DDL (Data Difinition Language)	Ngôn ngữ định nghĩa dữ liệu
DML (Data Manipulation Language)	Ngôn ngữ thao tác dữ liệu
ODL (The Object Definition Language)	Ngôn ngữ định nghĩa đối tượng
ODMG (Object Database Management Group)	Nhóm quản trị CSDL đối tượng, tổ chức đề xuất mô hình ODMG (ver 3.0) và ngôn ngữ OQL
OID (Object Identifier)	Định danh đối tượng
OODB (Object Oriented Database)	Cơ sở dữ liệu hướng đối tượng
OPG (OODB Predicate Graph)	Đồ thị tân từ hướng đối tượng
OQL (Object Query Language)	Ngôn ngữ truy vấn đối tượng
RPG (Relational Predicate Graph)	Đồ thị tân từ quan hệ
RTDB (Real-Time Database System)	Cơ sở dữ liệu thời gian thực
RTOODB (Real-Time Object Oriented Database System)	Cơ sở dữ liệu hướng đối tượng thời gian thực

RTOODDB (Real-Time Object Oriented Distributed Database System)	Cơ sở dữ liệu phân tán hướng đối tượng thời gian thực
SFW (Select...From...Where)	Khối lệnh trọng tâm của ngôn ngữ truy vấn đối tượng OQL
TID (Tuple Identifier)	Định danh bộ
UML (Unified Modeling Language)	Ngôn ngữ mô hình hợp nhất

MỤC LỤC

Lời nói đầu 11
Mở đầu 13
Chương 1. TỔNG QUAN VỀ MÔ HÌNH DỮ LIỆU 17
1.1. Giới thiệu 19
1.2. Mô hình mạng 20
1.2.1. Đặc tính nhận dạng mẫu tin 20
1.2.2. Đường liên kết 21
1.2.3. Biểu diễn các mối quan hệ 21
1.3. Mô hình phân cấp 22
1.3.1. Thuật toán chuyển đổi từ mô hình mạng đơn giản sang mô hình phân cấp 23
1.3.2. Mẫu tin cơ sở dữ liệu 24
1.4. Mô hình quan hệ 24
1.4.1. Miền 24
1.4.2. Quan hệ 25
1.4.3. Khóa 26
1.5. Mô hình thực thể - quan hệ 26
1.5.1. Thực thể và tập thực thể 26
1.5.2. Thuộc tính và khóa 27
1.5.3. Phân cấp ISA 27
1.5.4. Ràng buộc dữ liệu 28
1.5.5. Mối quan hệ 29

1.5.6. Sơ đồ thực thể - quan hệ 30
1.5.7. So sánh các khái niệm cơ bản trong mô hình dữ liệu mạng - mô hình thực thể - quan hệ 32
1.6. Mô hình đối tượng 32
1.6.1. Đối tượng 32
1.6.2. Kiểu của đối tượng 34
1.6.3. So sánh ngữ nghĩa đối tượng và mô hình thực thể - quan hệ 36
Chương 2. CÁC KHÁI NIỆM CƠ BẢN TRONG MÔ HÌNH DỮ LIỆU HƯỚNG ĐỐI TƯỢNG 39
2.1. Mô hình hạt nhân 41
2.1.1. Đối tượng và định danh đối tượng 42
2.1.2. Thuộc tính và phương thức 42
2.1.3. Sự đóng gói và chuyển thông điệp 42
2.1.4. Lớp 42
2.1.5. Phân cấp lớp và sự kế thừa 42
2.2. Mô hình dữ liệu hướng đối tượng 45
2.2.1. Hằng, giá trị và đối tượng 45
2.2.2. Kiểu - Mối quan hệ kết nhập 46
2.2.3. Phân cấp kiểu - Quan hệ kế thừa 47
2.2.4. Các tính chất của quá trình kế thừa kiểu 52
2.2.5. Phương thức của lớp 60
2.3. Mở rộng ngữ nghĩa của mô hình dữ liệu hướng đối tượng 65
2.3.1. Đối tượng phức hợp 65
2.3.2. Phiên bản của lược đồ 67

2.4. Giao diện cơ sở 68
2.4.1. Truyền thông điệp 69
2.4.2. Ngôn ngữ định nghĩa dữ liệu 69
2.4.3. Ngôn ngữ thao tác dữ liệu 70
2.4.4. Ngôn ngữ điều khiển dữ liệu 71

Chương 3. PHƯƠNG PHÁP CHUYỂN ĐỔI DỮ LIỆU GIỮA CƠ SỞ DỮ LIỆU QUAN HỆ VÀ HƯỚNG ĐỐI TƯỢNG 73

3.1. Phương pháp chuyển đổi dữ liệu từ cơ sở dữ liệu quan hệ đã tồn tại sang cơ sở dữ liệu hướng đối tượng 75
3.1.1. Tiến trình tổng quát 76
3.1.2. Điều chỉnh lược đồ quan hệ 77
3.1.3. Chuyển đổi lược đồ quan hệ sang lược đồ hướng đối tượng 85
3.1.4. Phương pháp chuyển và nạp dữ liệu từ cơ sở dữ liệu quan hệ sang cơ sở dữ liệu hướng đối tượng 91

3.2. Chuyển đổi lược đồ hướng đối tượng sang lược đồ quan hệ nhúng 93
3.2.1. Tiến trình tổng quát 93
3.2.2. Chuyển đổi lược đồ cơ sở dữ liệu hướng đối tượng sang lược đồ quan hệ nhúng 94

Chương 4. NGÔN NGỮ TRUY VẤN ĐỐI TƯỢNG OQL 109

4.1. Kiểu và lược đồ suy dẫn kiểu trong ngôn ngữ truy vấn OQL 112

4.2. Truy vấn select ... from ... where 113
4.2.1. Truy nhập các đặc trưng của đối tượng 115
4.2.2. Thuộc tính và lượng từ 115

4.2.3. Biến tham chiếu 116
4.2.4. Bộ phận của một phân cấp lớp 117
4.2.5. Phương thức tham chiếu 117
4.2.6. Kết xuất một cấu trúc 117
4.3. Đại số đối tượng 118
4.3.1. Phép toán đối tượng 118
4.3.2. Phép toán bộ 118
4.3.3. Phép toán tập hợp 119
4.3.4. Phép toán trên kiểu "túi" 120
4.3.5. Phép toán trên danh sách 120
4.3.6. Phép toán trên mảng 120
4.3.7. Các ví dụ áp dụng 120
Chương 5. TỐI ƯU HOÁ TRUY VẤN ĐỐI TƯỢNG 125
5.1. Mô hình ước lượng chi phí xử lý truy vấn 127
5.1.1. Mô hình chi phí các khối dựng sẵn 128
5.1.2. Các yếu tố chi phí cơ sở 130
5.1.3. Chi phí xử lý truy vấn trên các lớp 133
5.1.4. Thiết lập các tham số trong mô hình chi phí 137
5.2. Biên dịch các truy vấn đối tượng OQL thành các truy vấn quan hệ SQL 138
5.2.1. Biên dịch mệnh đề select trong OQL sang mệnh đề select trong SQL 139
5.2.2. Biên dịch mệnh đề from của truy vấn OQL sang mệnh đề from trong SQL 140
5.2.3. Biên dịch mệnh đề where của truy vấn OQL sang mệnh đề where của truy vấn quan hệ SQL 141

5.2.4. Tích hợp các modul chuyển đổi lược đồ và biên dịch truy vấn trong hệ thống cơ sở dữ liệu đối tượng – quan hệ 152

5.3. Tối ưu hoá truy vấn đối tượng bằng các phép biến đổi biểu thức đại số đối tượng 153

5.3.1. Các luật biến đổi đại số đối tượng 154

5.3.2. Các quy tắc tối ưu hoá truy vấn đối tượng tổng quát 156

5.3.3. Thuật toán tối ưu hoá truy vấn đối tượng dựa trên tập luật 157

5.3.4. Ví dụ minh hoạ 158

Chương 6. MỘT SỐ MÔ HÌNH CƠ SỞ DỮ LIỆU HƯỚNG ĐỐI TƯỢNG MỞ RỘNG 163

6.1. Cơ sở dữ liệu hướng đối tượng thời gian thực 165

6.1.1. Lớp đối tượng thời gian thực 165

6.1.2. Ngữ nghĩa dữ liệu thời gian thực 174

6.2. Cơ sở dữ liệu hướng đối tượng phân tán - thời gian thực 175

6.2.1. Mô hình hóa dữ liệu thời gian thực trong hệ thống 177

6.2.2. Mô hình giao dịch trong hệ thống RTOODDB 181

6.2.3. Ràng buộc thời gian của giao dịch 185

6.2.4. Các loại xung đột dữ liệu giữa các giao dịch 189

Tài liệu tham khảo 193

LỜI NÓI ĐẦU

Cơ sở dữ liệu (CSDL) là một trong những kiến thức cơ sở của các chuyên ngành về Công nghệ thông tin (CNTT), trong đó, các chuyên ngành về hệ thống thông tin, lập trình ứng dụng là những chuyên ngành mà CSDL đóng vai trò nền tảng trong việc mô hình hóa và cài đặt các phân hệ thông tin ứng dụng. Ngoài ra, nó còn cung cấp phương pháp luận về phân tích và thiết kế cho các hạ tầng kỹ thuật ngành CNTT.

Trong năm thế hệ CSDL đã và đang được sử dụng, trình bày trong nhiều giáo trình, sách chuyên khảo thì CSDL quan hệ được chọn để giảng dạy trong các trường Cao đẳng/ Đại học về ngành CNTT. Bởi vì, CSDL quan hệ được xây dựng trên mô hình toán học toàn vẹn và các phương pháp về cài đặt vật lý, truy vấn, tối ưu hóa truy vấn khá trong sáng và chặt chẽ. Tuy nhiên, với sự phát triển của các kỹ thuật phân tích thiết kế thì mô hình CSDL hướng đối tượng đã đạt được những ưu điểm nổi bật khi đặc tả các đối tượng trong thế giới thực và giảm thiểu các rủi ro về dư thừa thông tin. Vì vậy, cuốn sách "***Cơ sở dữ liệu hướng đối tượng***" là phần bổ khuyết quan trọng cho các bậc học ở Cao đẳng/ Đại học xét về cả hai khía cạnh học thuật là kỹ thuật thiết kế CSDL và công cụ đặc tả.

Cuốn sách "***Cơ sở dữ liệu hướng đối tượng***" có thể được sử dụng là giáo trình cơ sở cho các sinh viên chuyên ngành Hệ thống thông tin hay Lập trình ứng dụng ở trình độ Cao đẳng/ Đại học. Bên cạnh đó, với khuôn khổ nội dung của tài liệu, nó sẽ là tài liệu tham khảo của học viên Cao học, chuyên ngành Khoa học máy tính.

Đà Nẵng, tháng 6 năm 2015

HOÀNG BẢO HÙNG

MỞ ĐẦU

Trong hơn ba thập kỷ qua, mô hình dữ liệu của các hệ thống thông tin đã trải qua bốn thế hệ và thế hệ thứ năm đang được phát triển là mô hình dữ liệu hướng đối tượng. Suốt những năm 70, các lĩnh vực nghiên cứu về lý thuyết CSDL tập trung giải quyết trên mô hình dữ liệu quan hệ do E. F. Codd đề xuất, đây là mô hình được xây dựng trên cơ sở lý thuyết toán học về các quan hệ với cấu trúc chặt chẽ và hoàn chỉnh. Những kết quả nghiên cứu đạt được trên mô hình này không những tạo nền tảng về lý thuyết CSDL, mà còn mang tính ứng dụng cao, đó là các hệ thống quản trị CSDL thương mại phát hành vào cuối thập kỷ 70 và đầu thập kỷ 80 như Oracle, SQL/DS và DB2, .v.v... Tuy nhiên, sự đa dạng của các ứng dụng trong công nghệ phần mềm, các hệ thống tự động hoá, hệ thống đa phương tiện với sự tích hợp âm thanh, hình ảnh, tài liệu toàn văn và các ngôn ngữ lập trình, làm nảy sinh yêu cầu cần phải có một mô hình dữ liệu với các tính năng mạnh, phong phú để có thể đặc tả các đối tượng phức trong thế giới thực và đảm bảo tính mềm dẻo, khả chuyển của hệ thống khi có sự thay đổi, những đòi hỏi này tạo tiền đề cho sự hình thành và phát triển của mô hình CSDL hướng đối tượng.

Hiện nay, kỹ thuật hướng đối tượng đã được áp dụng rộng rãi trong việc phát triển phần mềm. Nhưng trên thực tế các hệ thống CSDL hướng đối tượng chỉ thống nhất trên một tập các khái niệm hướng đối tượng chung nhất – mô hình hạt nhân. Vì vậy, việc đưa ra mô hình dữ liệu chuẩn đối với mô hình dữ liệu hướng đối tượng là không nhất thiết, điều này có thể nhận thấy qua sự thể hiện các khái niệm đặc trưng hướng đối tượng trong mỗi hệ thống CSDL hướng đối tượng như GemStone, O_2, EXTRA, ORION, ODMG. Do đó, những kết quả nghiên cứu trên CSDL hướng đối tượng luôn được xem xét với một mô hình dữ liệu cụ thể, các kết quả này sẽ giải quyết cho một

lớp các bài toán với một tập con các khái niệm, tính chất đặc trưng hướng đối tượng đã được cài đặt trên mô hình. Như vậy, việc chọn một mô hình dữ liệu và ngôn ngữ truy vấn đối tượng cụ thể nhưng vẫn đảm bảo tính tổng quát khi nghiên cứu các vấn đề trong CSDL hướng đối tượng là rất quan trọng.

Trong những năm gần đây, việc phát triển các ngôn ngữ CSDL trên các hệ thống CSDL hướng đối tượng được xem là một hướng nghiên cứu quan trọng. Các ngôn ngữ CSDL được mở rộng theo hướng tích hợp với ngôn ngữ lập trình hướng đối tượng, điều này sẽ tạo điều kiện cho các hệ thống tăng khả năng tính toán phức tạp và xây dựng giao diện lập trình hướng đối tượng trong các hệ thống. Bên cạnh đó, sự tích hợp các ngôn ngữ này sẽ giải quyết được vấn đề giúp cho người thiết kế hệ thống có được công cụ hữu dụng khi thiết kế giao diện người dùng, ví dụ như O_2, Objectivity, ObjectStore, Versant đều hỗ trợ và tích hợp các ngôn ngữ lập trình C++, JAVA, Smalltalk, Active X (ObjectStore).

Ngôn ngữ CSDL bao gồm ngôn ngữ định nghĩa đối tượng (ODL), ngôn ngữ thao tác dữ liệu, truy vấn đối tượng (DML) và ngôn ngữ điều khiển dữ liệu (DCL). Trong đó, ngôn ngữ truy vấn đối tượng được mở rộng và bổ sung nhiều tính năng đặc tả các đối tượng phức trong từng phiên bản khác nhau. Ngôn ngữ truy vấn EXCESS mở rộng khái niệm tập hợp, là các đa tập cho phép có các phần tử trùng lập trong đa tập, cung cấp một tập các phép toán khá đầy đủ để xử lý cho các đối tượng là đa tập và mảng có chiều dài thay đổi. Ngôn ngữ truy vấn đối tượng OQL được phát triển kế thừa từ ngôn ngữ truy vấn quan hệ SQL, trong đó OQL đã tích hợp được khá nhiều tính năng mới trong SQL3 (2001) và tương thích hoàn toàn với SQL2 (1992).

Lớp các bài toán về tối ưu hoá truy vấn đối tượng khá phong phú và đa dạng. Những kết quả đạt được của các tác giả được xem xét trong từng hệ CSDL hướng đối tượng cụ thể. Do đó, người ta mong muốn nghiên cứu để phát triển và mở rộng các phương pháp tối ưu hoá truy vấn đối tượng được áp dụng cho từng lớp truy vấn riêng biệt

trong một mô hình dữ liệu hướng đối tượng có nhiều đặc tính đối tượng khá tổng quát.

Cuốn sách ***Cơ sở dữ liệu hướng đối tượng*** được chia làm 6 chương, trình bày theo ngữ nghĩa hình thức cho mô hình CSDL hướng đối tượng – với kiến trúc "hạt nhân", bao gồm các đặc trưng nổi bật nhất của mô hình hướng đối tượng. Nội dung của 6 chương được triển khai từ tổng quan về các hệ CSDL nhằm giới thiệu và hệ thống các hệ CSDL. Trong đó, chương 1 trình bày một số đặc trưng trong mô hình hướng đối tượng để người đọc có cách tiếp cận dễ hiểu hơn khi sang các chương tiếp theo. Chương 2, các khái niệm cơ bản của mô hình hướng đối tượng sẽ được trình bày một cách tổng quát và theo ngữ nghĩa hình thức. Đến việc chuyển đổi giữa thế hệ CSDL quan hệ và CSDL hướng đối tượng (chương 3) – là bước chuyển tiếp giữa cấu trúc, dữ liệu cho thế hệ CSDL mới – CSDL hướng đối tượng, việc chuyển đổi này sẽ làm rõ hơn về thiết kế CSDL vật lý của mô hình CSDL hướng đối tượng. Chương 4, tập trung giới thiệu về ngôn ngữ truy vấn hướng đối tượng OQL và các kỹ thuật tối ưu hóa truy vấn đối tượng trong chương 5. Chương 6 là sự mở rộng của mô hình CSDL hướng đối tượng, với hai mô hình: có yếu tố thời gian và phân tán cho các hệ thống CSDL lớn.

CHƯƠNG

1

TỔNG QUAN VỀ MÔ HÌNH DỮ LIỆU

Ngày nay việc mô tả và cài đặt một hệ thống quản trị CSDL là điều cần thiết trong việc số hóa các quá trình quản lý. Như vậy việc chọn lựa một mô hình dữ liệu để mô tả cho hệ thống CSDL là vấn đề cần đặt ra khi làm việc, nhưng thực tế cho thấy sẽ không có một mô hình dữ liệu nào tối ưu cho hệ thống CSDL mà người ta chỉ xem xét chúng tương quan trong việc sử dụng hệ thống ở đâu và vào lúc nào. Trong chương, chỉ thực hiện một bước hệ thống và trình bày một cách tổng quát sao cho người dùng có thể thấy được sự tương quan và những đặc tính kế thừa trong các mô hình CSDL, để từ đó có thể định hướng và lựa chọn một mô hình CSDL tương đối phù hợp với hệ thống CSDL của mình.

1

1.1. GIỚI THIỆU

Mô hình dữ liệu là một hệ hình thức Toán học gồm có hai phần:

- Một hệ thống ký hiệu để mô tả dữ liệu;

- Một tập hợp phép toán thao tác trên dữ liệu đó.

Để phân biệt các mô hình dữ liệu chúng ta sẽ quan tâm đến các khía cạnh sau khi xét đến một mô hình dữ liệu nào đó, thực tế cho thấy sẽ không có một mô hình dữ liệu nào tối ưu cho hệ thống CSDL mà người ta chỉ xem xét chúng tương quan trong việc sử dụng hệ thống ở đâu và vào lúc nào:

Mục đích: Các mô hình dữ liệu đều có mục đích chung là sử dụng một hệ thống ký hiệu để mô tả cho dữ liệu trong một CSDL và là hệ thống ký hiệu nền tảng cho ngôn ngữ thao tác dữ liệu (DML). Riêng, mô hình thực thể - quan hệ lại sử dụng các ký hiệu (về mặt sơ đồ) chỉ để mô tả và thiết kế lược đồ khái niệm, sau đó lược đồ đó sẽ được cài đặt trong hệ quản trị CSDL nào đó (đương nhiên rằng lúc này sẽ có thao tác chuyển đổi mô hình dữ liệu khi cài đặt). Vì vậy, nó không có hệ thống ký hiệu cho các phép toán trên dữ liệu.

Tính hướng đối tượng hoặc hướng giá trị: Thực tế hai mô hình dữ liệu: mô hình quan hệ và mô hình logic là hai mô hình hướng giá trị, chúng có tính khai báo và tác động đến việc định hình các ngôn ngữ được chúng hỗ trợ. Các mô hình mạng, mô hình phân cấp và mô hình đối tượng đều có cung cấp các đặc tính nhận dạng đối tượng (hoặc *gần như* đối tượng), nên có thể xem là "*hướng đối tượng*".

Giải quyết dư thừa: Tất cả các mô hình dữ liệu đều có cung cấp các phương pháp giúp người sử dụng tránh lưu trữ cùng một dữ kiện quá một lần (tính dư thừa - đây là điểm có thể này sinh các dị biệt khi thao tác trên các CSDL). Mô hình dữ liệu hướng đối tượng giải quyết vấn đề này tốt hơn vì chúng có thể tạo ra một bản sao duy nhất của đối tượng (đây chính là điểm khác biệt cơ bản nhất đối với các mô hình dữ liệu khác).

Biểu diễn mối quan hệ nhiều - nhiều: Mỗi mô hình dữ liệu đều có cách giải quyết vấn đề này một cách hiệu quả.

1.2. MÔ HÌNH MẠNG *(Network Model)*

Mô hình dữ liệu mạng là mô hình thực thể - quan hệ. Trong đó, các mối quan hệ bị hạn chế trong kiểu quan hệ nhị phân và nhiều - một. Hạn chế này cho phép chúng ta dùng một mô hình đồ thị có hướng cho việc biểu diễn các dữ liệu.

Ở vị trí của các tập thực thể, mô hình mạng đưa ra *kiểu mẫu tin logic*. Một kiểu mẫu tin logic là tên gán cho một tập các mẫu tin, được gọi là các mẫu tin logic. Nó được cấu tạo bởi các trường chứa các giá trị cơ bản như: số nguyên, chuỗi ký tự,... Tập các tên trường và kiểu của chúng cấu tạo nên khuôn dạng mẫu tin logic.

1.2.1. Đặc tính nhận dạng mẫu tin

Có một sự khác biệt quan trọng giữa bộ của mô hình quan hệ và mẫu tin của kiểu mẫu tin. Trong mô hình quan hệ hướng giá trị, bộ chẳng qua là giá trị của các thành phần. Hai bộ có cùng giá trị cho các thuộc tính giống nhau chỉ là một bộ. Ngược lại, mô hình mạng thuộc loại hướng đối tượng, ít nhất theo nghĩa nó hỗ trợ đặc tính nhận dạng đối tượng. Các mẫu tin của mô hình mạng có thể được xem như có một khóa là địa chỉ của mẫu tin, đó chính là "*đặc tính nhận dạng đối tượng*" của nó. Dấu hiệu nhận dạng duy nhất này làm cho các mẫu tin khác nhau, cho dù nếu chúng có các giá trị giống nhau trong các trường tương ứng.

Trong CSDL được xây dựng trên mô hình mạng, chúng là những con trỏ vật lý chỉ đến các mẫu tin khác để biểu thị các mối quan hệ mà kiểu mẫu tin của chúng có tham gia.

1.2.2. Đường liên kết *(Link)*

Thay vì gọi là mối quan hệ nhị phân nhiều - một, chúng ta sẽ sử dụng thuật ngữ *đường liên kết trong mô hình mạng*. Chúng ta dùng một đồ thị có hướng, gọi là mạng, đó là một sơ đồ thực thể - quan hệ đã được đơn giản hóa để biểu diễn các kiểu mẫu tin và đường liên kết giữa chúng. Các nút tương ứng với các kiểu mẫu tin. Nếu có một đường liên kết giữa hai mẫu tin T_1 và T_2 và đường liên kết này thuộc loại *n - 1* từ T_1 đến T_2 thì ta có một cung nối từ nút T_1 đến nút T_2. Nút và cung được đặt tên theo kiểu mẫu tin và đường liên kết.

1.2.3. Biểu diễn các mối quan hệ

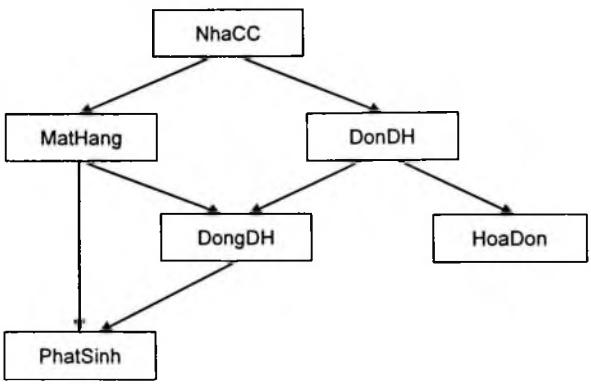
Chỉ những mối quan hệ kiểu nhị phân và *n - 1* mới có thể biểu diễn trực tiếp bằng các đường nối. Tuy nhiên, chúng ta có thể dùng một số kỹ thuật để biểu diễn các mối quan hệ bất kỳ. Giả sử chúng ta có một mối quan hệ R giữa các tập E_1, ..., E_k. Chúng ta sẽ tạo ra một kiểu mẫu tin logic T biểu diễn các *k* - bộ $(e_1, e_2, ..., e_k)$ của các thực thể trong mối quan hệ R. Khuôn dạng cho kiểu mẫu tin này có thể rỗng. Tuy nhiên, cũng dễ dàng thêm các trường mang thông tin vào trong khuôn dạng cho kiểu T. Trong mọi trường hợp, chúng ta đều có thể tạo ra được các đường liên kết L_1, L_2, ..., L_k. Đường nối L_i xuất phát từ kiểu mẫu tin T đến kiểu mẫu tin cho tập thực thể E_i. Ý nghĩa đường liên kết L_i là mẫu tin $(e_1, ..., e_k)$ có kiểu T được nối với mẫu tin e_i có kiểu E_i, do đó mỗi đường liên kết thuộc loại *n - 2*.

Trường hợp đặc biệt, nếu mối liên hệ là *n - 1* từ E_1, ..., E_{k-1} đến E_k và hơn nữa, tập thực thể E_k không nằm trong bất kỳ mối quan hệ nào khác, thì chúng ta có thể xác định kiểu mẫu tin T qua E_k bằng cách lưu các thuộc tính của E_k vào trong T.

Ví dụ 1.1. CSDL phục vụ việc quản lý đơn đặt hàng cho nhà cung cấp, với các quá trình quản lý như sau:

- Những nhà cung cấp (NhaCC) đề nghị các mặt hàng (MatHang);

- Đơn đặt hàng (DonDH) sẽ được chuyển đến NhaCC. Mỗi dòng đặt hàng (DongDH) tương ứng với một mặt hàng (MatHang) và có thể là đối tượng cho nhiều lần cung cấp liên tiếp (CC) được thể hiện qua các phát sinh (PhatSinh). Mỗi DonDH có thể tương ứng với thanh toán (ThT) gồm nhiều hóa đơn (HoaDon).



Hình 1.1. Mô hình mạng

1.3. MÔ HÌNH PHÂN CẤP *(Hierarchical Model)*

Mô hình phân cấp là mô hình dữ liệu mạng có nhiều cây, với tất cả các liên kết (cung) chỉ đi theo hướng từ con đến cha.

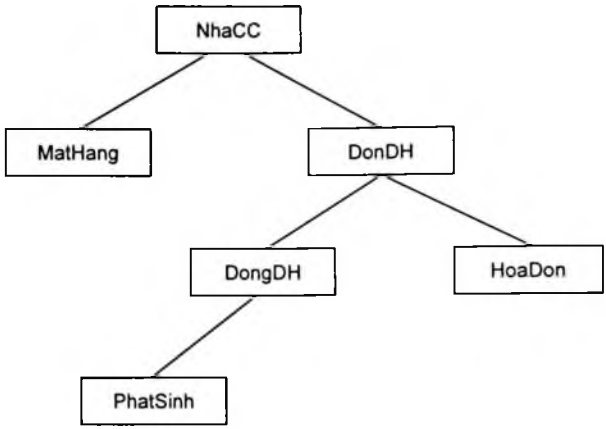
Mô hình phân cấp dùng cây như là cấu trúc cơ sở của nó. Cây chứa các nút phân cấp, với một nút đơn gọi là gốc, ở mức cao nhất. Một nút có thể có nhiều con, nhưng mỗi nút con chỉ có duy nhất một nút cha. Quan hệ *cha - con* trong cây thực chất là quan hệ *1 - n*, nhưng quan hệ *con - cha* là quan hệ *1 - 1*.

1.3.1. Thuật toán chuyển đổi từ mô hình mạng đơn giản sang mô hình phân cấp

Trước hết, chúng ta xem xét vấn đề thiết kế các phân cấp từ một mạng đã có bằng cách tách các mạng ra thành một hoặc nhiều cây. Lưu ý là trong mô hình phân cấp, tất cả các đường liên kết đều đi từ con đến cha. Vì vậy, chúng ta phải bắt đầu xây dựng từ nút có càng nhiều đường liên kết với các nút khác càng tốt, đặt nút này làm nút gốc của cây. Chúng ta gắn vào cây này tất cả các nút có thể gắn được, với yêu cầu là các đường liên kết luôn phải đi từ con đến cha. Khi không còn gắn nút nào vào thêm được nữa, ta sẽ bắt đầu với một cây khác với nút gốc là nút chưa được gắn làm gốc, quá trình lặp lại như trên. Kết quả là, mỗi nút sẽ xuất hiện một hoặc nhiều lần và khi đó chúng ta thu được một phân cấp. Cách xây dựng này được trình bày với đoạn chương trình giả mã sau:

Procedure BUILD(*n*);
  Đánh dấu chọn cho nút *n*;
  for mỗi đường liên kết từ một nút *m* nào đó đến *n* **do**
    Đặt *m* là con của *n*;
    if *m* chưa được đánh dấu chọn **then** BUILD(*m*);
  /*Chương trình chính*/
  Đánh dấu "*hủy chọn*" cho tất cả mọi nút;
  while vẫn còn nút có dấu "*hủy chọn*" **do**
    Lấy ra một nút n chưa được đánh dấu chọn;
    /* nên chọn nút *n* có nhiều đường liên kết đến và không có đường liên kết đi đến các nút có dấu "*hủy chọn*" */
    BUILD(n);

Ví dụ 1.2. Với CSDL cho ở ví dụ 1.1 và mô hình mạng trong hình 1.1, ta có thể biểu diễn mô hình phân cấp như sau:



Hình 1.2. Mô hình phân cấp

1.3.2. Mẫu tin cơ sở dữ liệu

Các phân cấp của mẫu tin logic là khái niệm ở mức lược đồ. Các thể hiện của CSDL tương ứng với một lược đồ sẽ chứa một tập các cây có nút là mẫu tin; mỗi cây được gọi là một *mẫu tin CSDL*. Một mẫu tin CSDL tương ứng với một cây trong lược đồ CSDL và mẫu tin gốc của một mẫu tin CSDL tương ứng với một thực thể của kiểu mẫu tin gốc. Nếu T là một nút của lược đồ và S là một con của nó thì mỗi mẫu tin thuộc kiểu T trong mẫu tin CSDL sẽ không có hoặc có một hay nhiều mẫu tin con thuộc kiểu S.

1.4. MÔ HÌNH QUAN HỆ *(Relational Model)*

1.4.1. Miền

Khái niệm Toán học của mô hình quan hệ là quan hệ hiểu theo nghĩa lý thuyết tập hợp: Là tập con của tích *Descarte* của các miền; ***miền*** (*domain*) là một tập các giá trị.

Ví dụ 1.3. Tập số nguyên là một miền, tập các xâu ký tự tạo thành tên người trong tiếng Anh có độ dài không quá 30 ký tự là một miền, tập hai số {0, 1} cũng là một miền,...

Gọi $D_1, D_2, \ldots, D_n$ là n miền. Tích *Descarte* của n miền là $D_1 \times \ldots \times D_n$ là tập các n - *bộ* $(v_1, v_2, \ldots, v_n)$ sao cho $v_i \in D_i$, với $i = 1...n$.

Ví dụ 1.4. Với $n = 2$, $D_1 = \{0, 1\}$, $D_2 = \{a, b, c\}$ khi đó:

$$D_1 \times D_2 = \{ (0,a), (0,b), (0, c), (1, a), (1, b), (1, c) \}$$

1.4.2. Quan hệ

Quan hệ là một tập con của tích *Descarte* của một hoặc nhiều miền. Như vậy, số phần tử của một quan hệ có thể là vô hạn, *ta giả sử quan hệ là một tập hữu hạn.*

Mỗi hàng của quan hệ gọi là *bộ*. Quan hệ là tập con của tích *Descarte*: $D_1 \times \ldots \times D_n$ gọi là quan hệ n - *ngôi*. Khi đó, mỗi bộ của quan hệ có n thành phần (n cột). Các cột của quan hệ gọi là *thuộc tính.*

Chúng ta có thể mô tả hình thức quan hệ là: Gọi $R = \{a_1, a_2, \ldots, a_n\}$ là tập hữu hạn của các thuộc tính, mỗi thuộc tính a_i với $i = 1, \ldots, n$ có miền giá trị tương ứng là $\text{dom}(a_i)$. Quan hệ trên tập thuộc tính $R = \{a_1, a_2, \ldots, a_n\}$ là tập con của tích *Descarte*:

$$r \subseteq \text{dom}(a_1) \times \ldots \times \text{dom}(a_n)$$

Ví dụ 1.5. Cho quan hệ NhanVien bao gồm các thuộc tính sau: ho_ten, nam_sinh, noi_lam_viec và luong là một quan hệ 4 - ngôi:

NhanVien (	ho_ten,	nam_sinh,	noi_lam_viec,	luong)
t_1	Lê Văn A	1960	VietHanIT	425
t_2	Hoàng Thị B	1970	Trường ĐHSP	390
t_3	Lê Văn Sơn	1945	Viện KHVN	425

t_1 = ("Lê Văn A", 1960, "VietHanIT", 425) là một bộ của quan hệ NhanVien.

1.4.3. Khóa

Khóa của một quan hệ *r* trên tập thuộc tính $R = \{a_1, a_2, ..., a_n\}$ là tập con $K \subseteq \{a_1, a_2, ..., a_n\}$ thoả mãn các tính chất sau:

Với bất kỳ 2 bộ $t_1, t_2 \in r$ *đều tồn tại* một thuộc tính $a \in K$ sao cho:

$$t_1(a) \neq t_2(a)$$

Để có thể định nghĩa khóa tốt hơn ta lưu ý rằng, nếu K' là khóa của quan hệ $r(a_1, a_2, ..., a_n)$ thì $K' \subseteq K \subseteq R$, K cũng là khóa của quan hệ *r*, nghĩa là:

$$\forall t_1, t_2 \in r, \ t_1(K') \neq t_2(K') \text{ luôn có } t_1(K) \neq t_2(K).$$

Ví dụ 1.6. Cho quan hệ HangHoa:

HangHoa (	ms_mathang	ten_hang	so_luong)
	10101	Sắt phi 6	1000
	10102	Sắt phi 8	2000
	20101	Xi măng	1000

Với quan hệ cho ở trên, trong đó thuộc tính ms_mathang (mã số mặt hàng) là khóa. Mỗi giá trị của ms_mathang đều xác định duy nhất một loại mặt hàng trong quan hệ HangHoa và nó là siêu khóa.

1.5. MÔ HÌNH THỰC THỂ - QUAN HỆ *(Entity - Relationship model)*

1.5.1. Thực thể và tập thực thể

Thuật ngữ *thực thể* không có một định nghĩa hình thức, nói rõ hơn: thực thể là một sự vật tồn tại và có thể phân biệt được. Như vậy *tính phân biệt được* rất gần với *đặc tính nhận dạng đối tượng*.

Một nhóm bao gồm tất cả các thực thể *tương tự* tạo ra một tập thực thể. Khái niệm *tập thực thể* là một khái niệm ở mức lược đồ trong hệ thống thuật ngữ về kiến trúc của hệ CSDL. Khái niệm ở mức

thể hiện tương ứng là tập con hiện hành của tất cả các phần tử của một tập thực thể cho trước đang hiện diện trong CSDL.

1.5.2. Thuộc tính và khóa

Các đặc tính của tập thực thể gọi là các thuộc tính. Chúng liên kết mỗi thực thể của tập thực thể với một giá trị từ một miền giá trị dành cho thuộc tính đó.

Khóa: Một thuộc tính hoặc một tập thuộc tính dùng để xác định một cách duy nhất mỗi thực thể trong một tập thực thể gọi là khóa đối với tập thực thể đó.

Trong mô hình CSDL thực thể - quan hệ, người ta chia ra nhiều khái niệm khóa khác nhau:

Siêu khóa: Là một khóa, thêm vào đó nó được dùng để định danh duy nhất cho một thực thể trong một tập thực thể.

Khóa dự tuyển: Là một siêu khóa, nhưng không tồn tại bất kỳ một tập con K nào đó của nó mà K là siêu khóa.

Khóa chính: Là một khóa dự tuyển, nhưng nó lại mang màu sắc của khóa phục vụ cho thao tác truy nhập các mẫu tin.

Khóa liên kết (Khóa vay mượn): Là một thuộc tính hay một tập thuộc tính của một thực thể này mà là khóa chính của thực thể khác. Do đó, khi xét trong thực thể chứa các thuộc tính này thì ta gọi chúng là khóa liên kết. Khi xuất hiện một khóa liên kết thì nó bao hàm một mối quan hệ giữa các thực thể đó.

1.5.3. Phân cấp ISA

Ta nói rằng "A **ISA** B", nếu tập thực thể B là sự tổng quát hóa của tập thực thể A, nói cách khác A là một trường hợp đặc biệt của B.

Mục đích chính của việc khai báo những mối quan hệ **ISA** giữa tập thực thể A và B là: A có thể thừa kế các thuộc tính của B nhưng có

thể có thêm những thuộc tính khác không có ý nghĩa đối với các phần tử của B không thuộc A. Nói cách khác, mỗi thực thể *a* trong A có quan hệ với đúng một phần tử *b* trong B, và như thế *a* và *b* thực sự chỉ là một thực thể. Không có phần tử nào trong B quan hệ với hai phần tử khác nhau trong A, nhưng một vài phần tử trong B có thể không có quan hệ gì với mọi phần tử trong A. Các thuộc tính khóa của tập A thực sự là các thuộc tính của tập B, và giá trị của những thuộc tính này đối với một thực thể *a* trong A được lấy từ thực thể *b* tương ứng trong B.

1.5.4. Ràng buộc dữ liệu

1.5.4.1. Phụ thuộc tồn tại

Một sự ràng buộc tồn tại hay tồn tại phụ thuộc có thể xuất hiện giữa hai tập thực thể. Nếu X và Y là các tập thực thể và mỗi thể hiện của Y phải có một thể hiện tương ứng của X, ta nói rằng Y tồn tại phụ thuộc vào X. Ta có thể gọi X là *thực thể mạnh*, hay thực thể cha, còn Y là *thực thể yếu* hay thực thể con.

1.5.4.2. Phụ thuộc nhận dạng

Phụ thuộc nhận dạng là một trường hợp đặc biệt của phụ thuộc tồn tại, xuất hiện khi một tập thực thể yếu không có khóa dự tuyển và các thể hiện của nó là không thể phân biệt được nếu không quan hệ với một thực thể khác.

1.5.4.3. Phụ thuộc tham chiếu

Phụ thuộc tham chiếu cũng là một trường hợp đặc biệt của phụ thuộc tồn tại, xuất hiện khi một thực thể yếu chứa một khóa liên kết có tư cách là khóa chính tương ứng đối với thực thể mạnh. Điều này dẫn đến một kiểu ràng buộc quan trọng gọi là ràng buộc tham chiếu, mà các giá trị khác *null* của các thuộc tính của khóa liên kết trong thể hiện thực thể yếu phải luôn bằng giá trị của khóa chính của thể hiện thực thể mạnh tương ứng.

1.5.5. Mối quan hệ

Mối quan hệ giữa các tập thực thể là một danh sách có thứ tự của các tập thực thể. Một tập thực thể đặc biệt có thể xuất hiện nhiều lần trong danh sách. Danh sách các tập thực thể này là khái niệm ở mức lược đồ của một mối quan hệ. Nếu có một mối quan hệ R giữa các tập thực thể E_1, E_2, ..., E_n thì thể hiện của R là một tập các *k*-bộ. Ta gọi một tập như thế là một tập các mối quan hệ. Mỗi *k*-bộ (e_1, ..., e_k) trong một tập quan hệ R xác nhận: các thực thể e_1, ..., e_k, trong đó, $e_1 \in E_1$, $e_2 \in E_2$, ... liên kết với nhau thành một nhóm trong mối quan hệ R.

Để mô hình hóa đầy đủ cho thế giới thực, cần phải phân loại các mối quan hệ theo số lượng các thực thể từ mỗi tập tham gia vào trong mối quan hệ.

1.5.5.1. Quan hệ 1 - 1

Dạng đơn giản và hiếm gặp nhất của mối quan hệ trên hai tập là mối quan hệ 1 - 1, tức là đối với mỗi thực thể trong một tập chỉ có nhiều nhất một phần tử được liên kết trong tập kia.

Chú ý rằng 1 - 1 của mối quan hệ này chỉ là một giả thiết về thế giới thực, vì vậy người thiết kế CSDL có thể chọn lựa tùy ý.

1.5.5.2. Quan hệ n - 1

Trong mối quan hệ *n - 1*, một thực thể trong tập E_2 có thể không liên kết với thực thể nào hoặc liên kết với một hay nhiều thực thể trong tập E_1, nhưng mỗi thực thể trong E_1 chỉ liên kết nhiều nhất với một thực thể trong E_2. Mối liên hệ này được gọi là *n - 1* từ E_1 đến E_2. Do vậy, mối quan hệ là một hàm từ E_1 đến E_2.

Khái niệm mối quan hệ *n - 1* tổng quát hóa thành các mối quan hệ giữa ba tập trở lên. Nếu có một mối quan hệ R giữa các tập E_1, E_2, ..., E_k và với các thực thể trong tất cả các tập trừ E_i chỉ có nhiều nhất một thực thể của E_i có liên hệ với chúng, thì chúng ta gọi R là mối quan hệ *n - 1* từ E_1, ..., E_{i-1}, ..., E_k đến E_i.

1.5.5.3. Quan hệ n - n

Thực tế chúng ta cũng gặp các mối quan hệ *n* - *n*, ở đây không có một hạn chế nào trên tập các *k*-bộ của các thực thể khi xuất hiện trong tập các mối quan hệ.

Nhiều mô hình dữ liệu không cho phép biểu diễn trực tiếp các mối quan hệ *n* - *n* mà yêu cầu phải phân rã chúng thành nhiều mối quan hệ *n* -*1*.

1.5.6. Sơ đồ thực thể - quan hệ

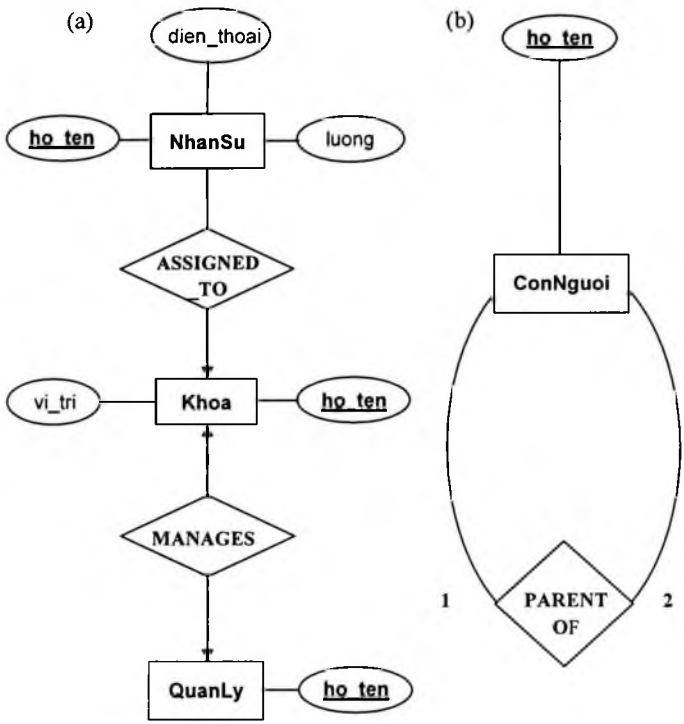
Sơ đồ thực thể - quan hệ được biểu diễn và quy ước theo những ký hiệu sau:

- Các hình chữ nhật biểu diễn các tập thực thể.

- Các vòng tròn biểu diễn thuộc tính. Chúng được liên kết với các tập thực thể bằng các cạnh (vô hướng). Các thuộc tính là thành phần của một khóa cho tập thực thể sẽ được gạch chân. Trường hợp đặc biệt khi một tập chỉ có một thuộc tính thì ta gọi tập đó bằng tên thuộc tính của tập đó. Lúc đó, tập thực thể sẽ là một vòng tròn chứ không phải là hình chữ nhật, nó gắn với các mối quan hệ mà tập được hàm chứa.

- Các hình thoi biểu diễn các mối quan hệ. Chúng được liên kết với các tập thành viên bởi các cạnh vô hướng hoặc có hướng (các cung). Thứ tự của các tập thực thể trong danh sách được chỉ ra bằng cách đánh số các cạnh, mặc dù thứ tự đó không cần thiết trừ khi một tập xuất hiện nhiều lần trong danh sách.

Ví dụ 1.7. Cho ba tập thực thể NhanSu, Khoa và QuanLy, với các tập thuộc tính: NhanSu(ho_ten, luong, dien_thoai), Khoa(ten_khoa, vi_tri) và QuanLy(ho_ten). Hai tập đầu được liên kết nhờ mối quan hệ **ASSIGNED_TO**, tập thứ hai và ba được liên kết nhờ mối quan hệ **MANAGES**.



Hình 1.3. Các ví dụ về sơ đồ thực thể - quan hệ

Hình 1.3 (a) trình bày sơ đồ thực thể - quan hệ của 3 tập NhanSu, Khoa và QuanLy. Hình 1.3 (b) có tập thực thể ConNguoi và mối quan hệ PARENT_OF giữa ConNguoi và ConNguoi. Chú ý hai cạnh từ PARENT_OF đến ConNguoi; cạnh thứ nhất biểu diễn con và cạnh thứ hai biểu diễn cha. Với giá trị hiện hành của tập quan hệ PARENT_OF là tập gồm các cặp (p_1, p_2) trong đó p_2 có quan hệ là cha của p_1

1.5.7. So sánh các khái niệm cơ bản trong mô hình dữ liệu mạng - mô hình thực thể - quan hệ

Mô hình mạng	**Mô hình thực thể - quan hệ**
Kiểu mẫu tin logic	Tập thực thể
Mẫu tin logic	Thực thể
Khuôn dạng mẫu tin logic	Tập các thuộc tính

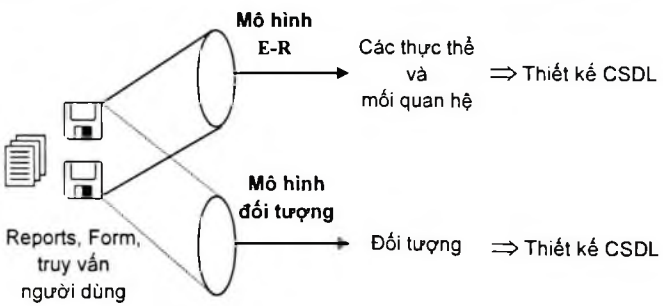
1.6. MÔ HÌNH ĐỐI TƯỢNG *(Object Model)*

1.6.1. Đối tượng

Đối tượng là tên của tập hợp các thuộc tính đủ để mô tả việc nhận dạng phân biệt một thực thể. Tương tự như thực thể, các đối tượng được nhóm vào các lớp. Một lớp đối tượng có tên xác định để phân biệt giữa các lớp. Tên của lớp phản ánh các thực thể mà nó biểu diễn.

Một đối tượng có một tập các thuộc tính. Mỗi thuộc tính của đối tượng biểu diễn một đặc tính nhận dạng của đối tượng. Hơn nữa, tập các thuộc tính này đảm bảo tính đầy đủ của việc mô tả đối tượng.

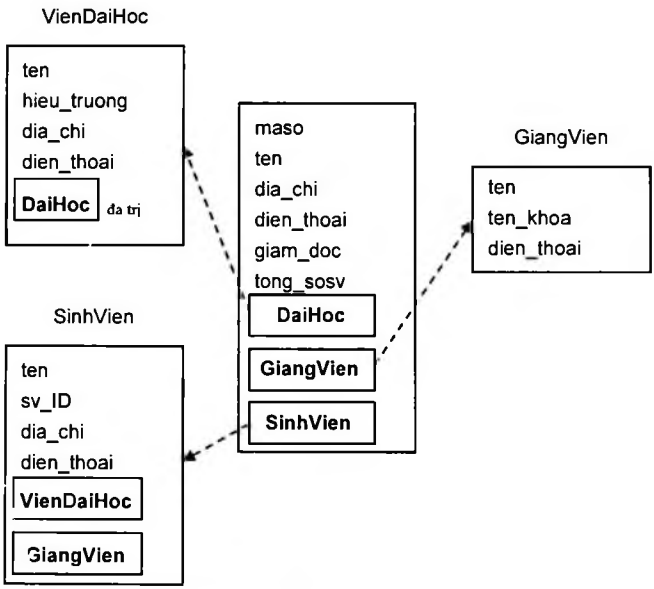
Sự khác biệt quan trọng giữa đối tượng và thực thể là các đối tượng tồn tại một cách độc lập. Trong mô hình đối tượng không tồn tại đối tượng tương tự như một thực thể yếu trong mô hình thực thể - quan hệ. Tức là, không tồn tại đối tượng tồn tại phụ thuộc vào đối tượng khác.



Hình 1.4. Hai mô hình dữ liệu cho sự phát triển thiết kế CSDL

Các thuộc tính của đối tượng xác định đặc trưng của nó. Các thuộc tính có thể là các mục dữ liệu đơn hoặc là các dữ liệu hợp thành (nhiều thành phần) và giá trị của nó có thể đơn trị hoặc đa trị. Đặc biệt, các thuộc tính cũng có thể là một đối tượng khác.

Xét đối tượng ở hình 1.5, thì 6 thuộc tính đầu tiên được gọi là các thuộc tính không phải đối tượng, giá trị của chúng là các mục dữ liệu vô hướng, 3 thuộc tính cuối cùng là các thuộc tính đối tượng. Giá trị của chúng là các đối tượng: DaiHoc, GiaoSu và SinhVien là các đối tượng chứa trong VienDaiHoc.



Hình 1.5. Lược đồ đối tượng Viện Đại học

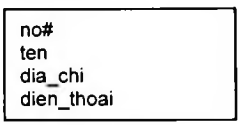
Miền của một thuộc tính là tập giá trị mà đối tượng có thể nhận được. Mô tả của miền thuộc tính không phải đối tượng, khác hẳn với miền thuộc tính đối tượng. Miền của thuộc tính không phải đối tượng

chứa trong nó cả hai mô tả về vật lý và ngữ nghĩa. Mô tả vật lý xác định kiểu dữ liệu, độ dài của thuộc tính và các giới hạn hay ràng buộc đối với dạng dữ liệu. Mô tả về ngữ nghĩa đặc tả các chức năng hoặc ý nghĩa của thuộc tính. Miền của thuộc tính đối tượng là tập các thể hiện của đối tượng.

1.6.2. Kiểu của đối tượng

1.6.2.1. Đối tượng đơn giản

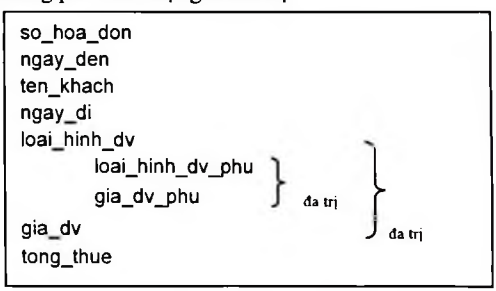
Đối tượng đơn giản (Simple object) chỉ chứa các thuộc tính không phải đối tượng và đơn trị.



Hình 1.6. Đối tượng đơn giản

1.6.2.2. Đối tượng đa hợp

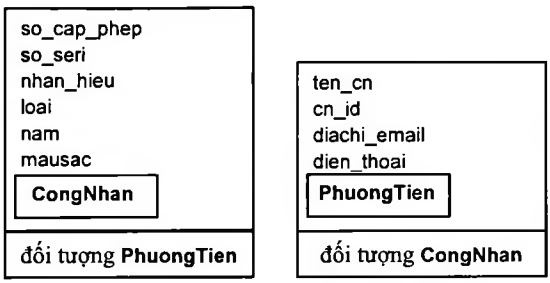
Đối tượng đa hợp (Composite object) chứa một hoặc nhiều các thuộc tính không phải đối tượng và đa trị.



Hình 1.7. Đối tượng đa hợp (Hóa đơn khách sạn)

1.6.2.3. Đối tượng phức hợp

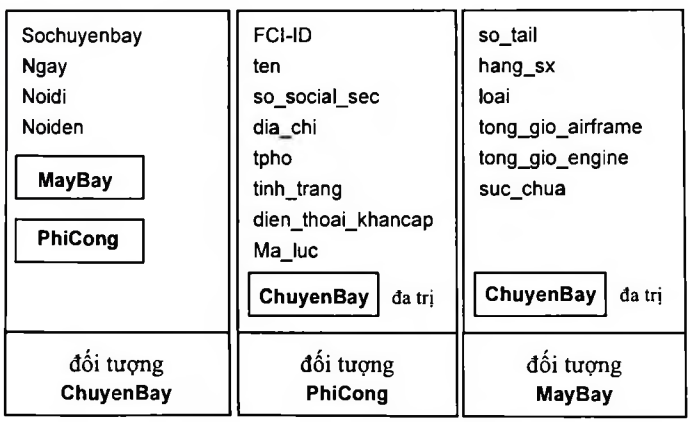
Đối tượng phức hợp (Compound object) chứa ít nhất một thuộc tính đối tượng.



Hình 1.8. Đối tượng phức hợp

1.6.2.4. Đối tượng kết hợp

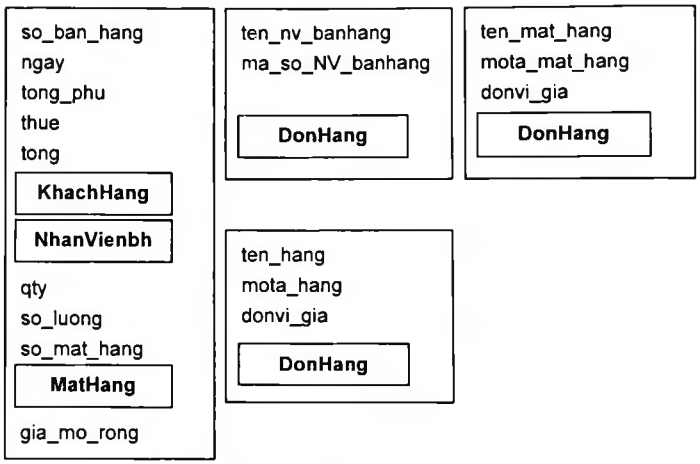
Đối tượng kết hợp (Association object) là một đối tượng có quan hệ với ít nhất 2 đối tượng và mối quan hệ của chúng được lưu trữ riêng biệt. Các đối tượng kết hợp là đối tượng hợp thành, và chúng ta có thể nói đối tượng kết hợp là một trường hợp đặc biệt của đối tượng hợp thành.



Hình 1.9. Đối tượng kết hợp

1.6.2.5. Đối tượng lai

Đối tượng lai (Hybrid object) bao hàm sự kết nối của các đối tượng phức hợp và các đối tượng hợp thành trong thuộc tính đối tượng xuất hiện ở một nhóm hợp thành.



Hình 1.10. Đối tượng lai

1.6.3. So sánh ngữ nghĩa đối tượng và mô hình thực thể - quan hệ

Mô hình thực thể - quan hệ và mô hình đối tượng có những điểm tương tự và khác nhau. Điểm tương tự đầu tiên là cả hai mô hình đều là những công cụ để phân tích và xây dựng các cấu trúc dữ liệu của người dùng. Cả hai mô hình đều cố gắng biểu đạt những mối quan hệ và các ý tưởng của người dùng trong thế giới thực.

Sự khác biệt chính yếu giữa hai mô hình là chỉ một trong chúng có tính định hướng (đặc tính nhận dạng đối tượng). Mô hình thực thể - quan hệ lấy khái niệm thực thể là khái niệm cơ sở, còn mô hình đối tượng lấy ngữ nghĩa đối tượng làm cơ sở. Các đối tượng là nguyên tử trong thế giới thực và là đơn vị có thể phân biệt nhỏ nhất mà người sử dụng muốn truy xuất.

Thực thể theo định nghĩa trong mô hình dữ liệu thực thể - quan hệ là không tồn tại. Chúng chỉ có thể là một phần của các thực thể thật. Như vậy, các thực thể có nghĩa với người dùng chỉ có thể là ngữ nghĩa đối tượng.

KẾT LUẬN

Với việc giới thiệu các đặc trưng của 5 mô hình dữ liệu, chúng ta sẽ xem xét các yếu tố cơ bản trong từng mô hình dữ liệu và sự phát triển của các mô hình thông qua việc đặc tả các đối tượng trong thế giới thực. Từ đó, người đọc sẽ có thể phân tích những thay đổi, so sánh sự phù hợp giữa các mô hình khi thiết kế CSDL. Việc trình bày các mô hình dữ liệu dưới góc độ "lõi" sẽ cho chúng ta cách nhìn gần hơn, theo hướng ngữ nghĩa đặc tả của các mô hình dữ liệu.

Chương 1 sẽ là phần kiến thức cơ bản được nhắc lại, hệ thống hóa để người đọc có sự chuẩn bị một cách hệ thống các mô hình dữ liệu và sự chuyển hóa giữa chúng. Đối với mô hình hướng đối tượng, các khái niệm, ví dụ trình bày trong chương chỉ nhằm cung cấp cho người đọc cách tiếp cận "dễ hiểu" cho việc nghiên cứu về ngữ nghĩa hình thức ở các chương sau.

CÂU HỎI ÔN TẬP CHƯƠNG 1

Câu 1.1. Phân tích và làm rõ sự phát triển có tính kế thừa giữa các mô hình dữ liệu?

Câu 1.2. Để đặc tả cho hệ thống thông tin quản lý thư viện trường, Anh(chị) hãy đề xuất một lược đồ hướng đối tượng gồm các đối tượng cơ bản của hệ thống.

Câu 1.3. Hãy chuyển các đối tượng trong hình 1.6 đến hình 1.10 thành các bảng trong mô hình quan hệ.

Câu 1.4. Trên cơ sở phân tích, so sánh giữa hai mô hình dữ liệu. Hãy nêu một phương pháp chuyển đổi "*đối tượng*" trong mô hình hướng đối tượng sang "*thực thể*" trên mô hình thực thể - quan hệ.

Câu 1.5. Trong ví dụ 1.1, hãy biểu diễn CSDL trong hình 1.1 bằng các mô hình dữ liệu đã được trình bày trong chương 1.

CHƯƠNG

2

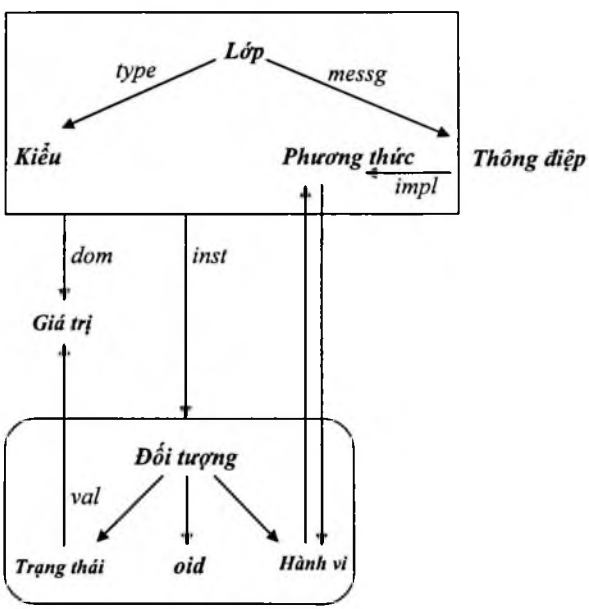
CÁC KHÁI NIỆM CƠ BẢN TRONG MÔ HÌNH DỮ LIỆU HƯỚNG ĐỐI TƯỢNG

Chương 2, giới thiệu những kiến thức cơ sở của mô hình dữ liệu hướng đối tượng như: định danh đối tượng, kiểu/ lớp, lược đồ, mối quan hệ kế thừa, phương thức và sự đóng gói. Sau đó, xem xét sự mở rộng ngữ nghĩa của mô hình dữ liệu hướng đối tượng qua các khái niệm về đối tượng phức hợp và phiên bản lược đồ. Cuối chương, là phần trình bày giao diện cơ sở gồm ngôn ngữ con định nghĩa dữ liệu, thao tác dữ liệu và điều khiển dữ liệu.

2

2.1. MÔ HÌNH HẠT NHÂN

Mô hình hạt nhân hỗ trợ mô hình dữ liệu với các điều kiện cơ bản của nhiều kiểu ứng dụng và giúp chúng ta nhìn rõ hơn sự ảnh hưởng mạnh mẽ của mô hình dữ liệu hướng đối tượng trên kiến trúc của hệ CSDL.



Hình 2.1. Sơ đồ bên ngoài của mô hình hạt nhân

2.1.1. Đối tượng và định danh đối tượng

Một thực thể bất kỳ trong thế giới thực là một đối tượng, được liên kết trong hệ thống bằng một định danh duy nhất.

2.1.2. Thuộc tính và phương thức

Đối tượng có một hoặc nhiều thuộc tính và một hoặc nhiều phương thức tác động trên giá trị của đối tượng.

Giá trị của thuộc tính của một đối tượng cũng là một đối tượng. Một thuộc tính có thể nhận giá trị đơn hoặc một tập các giá trị.

2.1.3. Sự đóng gói và chuyển thông điệp

Thông điệp được gửi đến một đối tượng để truy nhập vào giá trị của thuộc tính và các phương thức được đóng gói trong đối tượng. Không thể truy nhập đối tượng nếu không thông qua giao diện chung đã đặc tả đối với đối tượng.

2.1.4. Lớp

Tất cả các đối tượng có cùng chung một tập các thuộc tính và phương thức có thể được nhóm thành một lớp.

Một đối tượng chỉ thuộc vào một lớp có vai trò như là một thể hiện của lớp đó.

Lớp cũng là một đối tượng; đặc biệt, lớp là một thể hiện của *metaclass*.

2.1.5. Phân cấp lớp và sự kế thừa

Các lớp trong hệ thống phân cấp hoặc đồ thị định hướng không chu trình, được gọi là phân cấp lớp, sao cho đối với một lớp c và tập các lớp ở mức thấp hơn $\{s_i\}$ liên kết với c, một lớp trong tập $\{s_i\}$ là một đặc tả của lớp c, và ngược lại c là sự tổng quát hóa của các lớp trong $\{s_i\}$. Các lớp trong $\{s_i\}$ được gọi là các lớp con của lớp c; và lớp c là lớp cha (siêu lớp) của các lớp trong $\{s_i\}$.

Một lớp bất kỳ trong $\{s_i\}$ kế thừa tất cả các thuộc tính và phương thức của lớp *c* và có thể được bổ sung thêm các thuộc tính và phương thức khác cho mình.

Tất cả các thuộc tính và phương thức đối với lớp *c* được kế thừa vào trong tất cả các lớp của nó một cách đệ quy.

Một thể hiện của lớp *s* cũng là một sự thể hiện logic của tất cả các siêu lớp của *s*.

Miền của một thuộc tính là một lớp chứa các giá trị của thuộc tính và là kiểu cơ bản, bao gồm các lớp (kiểu) nguyên thủy (integer, string,...). Một thuộc tính có thể mang giá trị đơn hoặc là một tập giá trị từ miền giá trị của nó.

Phân cấp các lớp xuất hiện từ quan hệ tập hợp giữa một lớp và các thuộc tính của nó và từ miền giá trị của các thuộc tính (là các lớp). Sự phân cấp này được gọi là *phân cấp lớp đa hợp*.

Trong hệ thống hướng đối tượng, một lớp có thể có một số lớp con. Lớp kế thừa các thuộc tính và phương thức từ chỉ một siêu lớp được gọi là *kế thừa đơn*; một lớp kế thừa các thuộc tính và phương thức từ nhiều hơn một lớp được gọi là *kế thừa bội*. Khái niệm của phân cấp lớp và tính kế thừa của các thuộc tính và phương thức thuộc phân cấp lớp là sự khác biệt giữa nguyên lý lập trình hướng đối tượng với các nguyên lý lập trình khác trên các kiểu dữ liệu trừu tượng.

Lớp kế thừa các phương thức từ siêu lớp. Các phương thức được kế thừa có thể hiệu chỉnh lại với tên vẫn không đổi. Khi một thông điệp được gửi đến một đối tượng, lớp chứa đối tượng sẽ tìm kiếm để xác định một phương thức tương ứng. Nếu phương thức này không được liên kết với lớp, thì việc tìm kiếm sẽ lại diễn ra tại các siêu lớp của lớp này để xác định phương thức: Nếu tồn tại một phương thức, thì chương trình kết hợp với nó sẽ được thực thi. Việc sử dụng chỉ một định danh cho nhiều chương trình gọi là *sự tái nạp* (overloading).

Trong quá trình kế thừa có nhiều vấn đề nảy sinh cần được xem xét, đó là sự xung đột về tên, về phạm vi của sự kế thừa và vi phạm tính đóng gói:

+ Có hai kiểu xung đột về tên: Giữa một lớp và siêu lớp của nó; giữa các siêu lớp của một lớp. Trong hệ thống hỗ trợ kế thừa đơn, chỉ có kiểu xung đột thứ nhất xuất hiện. Nếu tên của một thuộc tính hay một phương thức xung đột giữa một lớp và siêu lớp của nó, thì tên dùng trong lớp được quyền ưu tiên trước: Tức là, thuộc tính và phương thức của siêu lớp không được kế thừa. Trường hợp xung đột về tên các thuộc tính và phương thức giữa các siêu lớp, hướng giải quyết thông thường là chọn một siêu lớp trong các siêu lớp để xác định các thuộc tính và phương thức kế thừa trên cơ sở thứ tự về quyền ưu tiên trước. Phần lớn các hệ thống yêu cầu một thứ tự tiền ưu tiên giữa các siêu lớp để mô tả trong phân cấp lớp; các hệ thống sẽ xác định thứ tự tiền ưu tiên khi thực thi.

+ Vấn đề thứ hai của sự kế thừa là phạm vi tác động của kế thừa. Cách giải quyết thứ nhất là xác định phạm vi tác động của sự kế thừa hoặc là kế thừa đầy đủ hoặc cho phép kế thừa có lựa chọn. Kế thừa đầy đủ có nghĩa là, một lớp kế thừa tất cả các thuộc tính và phương thức từ mọi siêu lớp của nó, ngoại trừ những thuộc tính và phương thức gây nên xung đột về tên. Kế thừa có chọn lựa cho phép một lớp con không kế thừa một số thuộc tính và phương thức từ các siêu lớp. Mặc dù kế thừa có chọn lựa có một số thuận lợi hơn, nhưng hầu hết các hệ thống đều chấp nhận kế thừa đầy đủ do kế thừa có chọn lựa tạo nên sự khó khăn đối với người dùng và hệ thống vì phải lưu vết kế thừa. Hướng thứ hai là cho phép kế thừa các thuộc tính cùng tên nhưng khác đặc tả.

+ Vấn đề thứ ba của sự kế thừa là vi phạm nguyên lý đóng gói. Nếu người dùng có thể truy nhập trực tiếp các thuộc tính của một lớp từ một lớp con, như các phép toán đổi tên hoặc xóa một thuộc tính thì nó có thể làm mất hiệu lực của các phương thức xác định trong lớp mà chúng tham chiếu đến các thuộc tính. Một hướng khác là giới hạn truy nhập đối với các thuộc tính của một lớp thông qua các phương thức xác định chúng.

2.2. MÔ HÌNH DỮ LIỆU HƯỚNG ĐỐI TƯỢNG

Các khái niệm được giới thiệu trong phần này không tiếp cận đến bất kỳ một mô hình hướng đối tượng cụ thể nào, chúng là các khái niệm chung nhất được sử dụng trong hầu hết các mô hình đối tượng. Trước hết chúng ta sử dụng một số ký hiệu sau:

T - tập hữu hạn các kiểu;

L - tập các kiểu nguyên thủy như: Integer, Char, String, Boolean, Double;

Att - tập tất cả thuộc tính của các đối tượng trong hệ thống;

O - tập tất cả đối tượng và các đại lượng trong một CSDL.

2.2.1. Hằng, giá trị và đối tượng

Tập **Dom** các giá trị nguyên thủy là hợp của tất cả các miền của kiểu nguyên thủy. Các phần tử của **Dom** được gọi là *hằng*. Hằng đặc biệt *nil* biểu diễn giá trị không xác định.

Tập **Obj** = $\{o_1, o_2, ...\}$ là tập vô hạn các *định danh đối tượng* và tập O_{id} là tập con các định danh đối tượng, $O_{id} \subset$ **Obj**. Họ các *giá trị* trên O_{id}, ký hiệu **Val**(O_{id}), được xác định như sau:

(i) Hằng $nil \in$ **Val**(O_{id})

(ii) Nếu $v \in$ **Dom** thì $v \in$ **Val**(O_{id})

(iii) Nếu $o \in O_{id}$ thì $o \in$ **Val**(O_{id})

(iv) Tập $\{v_i \mid v_i \in$ **Val**(O_{id}), $(i = 1, ..., n)\} \in$ **Val**(O_{id})

(v) Cho tập các giá trị $\{v_i \mid v_i \in$ **Val**(O_{id}), $(i = 1, ..., n)\}$ và tập các tên thuộc tính rời nhau $\{A_i \mid A_i, A_j \in Att, (i \neq j \Rightarrow A_i \neq A_j), (i, j = 1, ..., n)\}$ thì bộ $[A_1 : v_1, ..., A_n : v_n] \in$ **Val**(O_{id}).

Đối tượng là cặp (o, v), trong đó o là định danh đối tượng và v là bộ giá trị.

Trong mô hình dữ liệu quan hệ, so sánh bằng của các bộ trong quan hệ luôn dựa trên các giá trị của bộ, hai bộ bằng nhau nếu tất cả các thuộc tính khóa của hai bộ có cùng giá trị. Tuy nhiên, trong mô

hình dữ liệu hướng đối tượng quan hệ "*bằng nhau*" của các đối tượng được phát triển và định nghĩa như sau:

Định nghĩa 2.1. *Cho hai đối tượng $S_1(o_1, v_1)$ và $S_2(o_2, v_2)$.*

(i) *Đối tượng đồng nhất: Hai đối tượng S_1 và S_2 được gọi là đối tượng đồng nhất, ký hiệu $S_1 =_{oid} S_2$ nếu chúng có cùng định danh đối tượng duy nhất, tức là $S_1.o_1 = S_2.o_2$*

(ii) *Đối tượng bằng nhau về giá trị: Hai đối tượng S_1 và S_2 được gọi là bằng nhau về giá trị, ký hiệu $S_1 =_v S_2$:*

- *Nếu kiểu của đối tượng là nguyên thủy thì chúng phải có cùng giá trị;*
- *Nếu kiểu của đối tượng là kiểu không nguyên thủy thì chúng phải có cùng số các tính chất và với mọi tính chất p_i của S_1 đều tồn tại tính chất p_j của S_2 bằng nhau về giá trị.*

2.2.2. Kiểu - Mối quan hệ kết nhập

Các đối tượng được nhóm lại để tạo thành các lớp. **Class** là tập tên lớp. Tất cả đối tượng trong một lớp đều có cùng kiểu. Mỗi lớp c có một kiểu $\sigma(c)$, đây là kiểu của các đối tượng trong lớp. Tức là, mỗi đối tượng (o, v) trong lớp c, v phải có cấu trúc được mô tả là $\sigma(c)$.

(i) Các kiểu nguyên thủy Integer, Char, String, Boolean và Double $\in$ **Types**(C);

(ii) Nếu $c \in C$ thì $c \in$ **Types**(C);

(iii) Nếu $\tau \in$ **Types**(C), thì tập $\{\tau\} \in$ **Types**(C);

(iv) Cho một tập các kiểu $\{\tau_i \mid \tau_i \in$ **Types**(C), $(i = 1, ..., n)\}$ và tập các tên thuộc tính rời nhau $\{A_i \mid A_i, A_j \in Att, (i \neq j \Rightarrow A_i \neq A_j), (i, j = 1, ..., n)\}$ thì bộ $[A_1 : \tau_1, ..., A_n : \tau_n] \in$ **Types**(C) và mỗi lớp c có một kiểu với dạng bộ tương ứng, tức là $\sigma(c) = [A_1 : \tau_1, ..., A_n : \tau_n]$.

Nếu lớp c_1 có thuộc tính A_i với kiểu τ_i và $\tau_i = c_2$, mà c_2 là một lớp, thì giữa lớp c_1 và c_2 có một *mối quan hệ kết nhập* - các đối tượng của lớp c_1 được hợp thành từ các đối tượng của lớp c_2.

2.2.3. Phân cấp kiểu - Quan hệ kế thừa

Một trong những đặc tính quan trọng nhất của nguyên lý hướng đối tượng là khả năng kế thừa và sử dụng lại. Một kiểu có thể kế thừa những đặc trưng của một hay nhiều kiểu cơ sở (siêu lớp) và có thể được bổ sung một số đặc trưng mới theo yêu cầu. Nguyên nhân chính dẫn đến việc ứng dụng khái niệm kế thừa trong công nghệ phần mềm là khả năng dễ mở rộng và phát triển, giống như luật kế thừa gia sản trong xã hội loài người là tạo khả năng gộp, nhóm một số đặc trưng của hai hay nhiều kiểu đã được xây dựng tốt để tạo lập những kiểu cho những đối tượng mới đáp ứng được tính sử dụng lại trong quá trình phát triển phần mềm. Kiểu cơ sở để kế thừa thường được gọi là siêu kiểu, còn kiểu kế thừa được gọi là kiểu dẫn xuất hay là kiểu con. Quan hệ kế thừa sẽ tạo thành quan hệ thứ tự bộ phận, trong đó đối tượng của một kiểu sẽ là thể hiện của chính kiểu đó nhưng đồng thời cũng có thể xem nó là thể hiện của các kiểu cha.

Định nghĩa 2.2. *Quan hệ Sub kế thừa kiểu.*

Cho trước các kiểu $\tau_i \in T$, $i = 1,..., n$ và $\delta_l \in T \cup L$, $l = 1, 2,..., k$. Kiểu τ kế thừa (trực tiếp) từ những kiểu τ_i và được bổ sung một số thuộc tính trong các kiểu δ_l sẽ được định nghĩa là:

Type $\tau = \tau_1, \tau_2, ..., \tau_n \{p_1 : \delta_1 ; p_2 : \delta_2 ; ...; p_k : \delta_k\}$, *trong đó,* $p_l \in Att$ *với mọi* $l = 1,..., k$. *Ta nói rằng kiểu* τ *kế thừa từ các kiểu* τ_i, $i = 1,..., n$ *hay ngược lại* τ_i *là cha (trực tiếp) của* τ *và được ký hiệu* τ ***Sub*** τ_i.

Trong hệ thống, có những kiểu nguyên thủy hoặc các kiểu do người sử dụng tạo lập, không theo quan hệ kế thừa *Sub*, những kiểu đó người ta gọi là kiểu chuẩn. Kiểu chuẩn là kiểu nguyên thủy hoặc kiểu được tạo lập như sau:

Type $\tau = \{p_1 : \delta_1 ; p_2 : \delta_2 ; ...; p_k : \delta_k\}$, trong đó $\delta_l \in T \cup L$, $p_l \in Att$, $l = 1, 2,..., k$.

Gọi N là tập tất cả các kiểu chuẩn, hiển nhiên là $L \subseteq N$. Gọi I là tập tất cả các kiểu được tạo ra theo quan hệ kế thừa.

Dễ dàng nhận thấy quan hệ *Sub* có tính phản xạ: τ ***Sub*** τ, với mọi $\tau \in T$. Từ đó người ta xây dựng bao đóng bắc cầu của quan hệ cha/ con (*Sub*) và ký hiệu là $\leq_1$, gọi là *quan hệ kế thừa kiểu*.

Định nghĩa 2.3. *Quan hệ kế thừa kiểu* $\leq_1$ *được định nghĩa:*

(i) $\tau \leq_1 \tau$ *với mọi* $\tau \in T$,

(ii) $\tau_1, \tau_2, \tau_3 \in T$, τ_1 *Sub* τ_2 *và* $\tau_2 \leq_1 \tau_3$ *thì* $\tau_1 \leq_1 \tau_3$

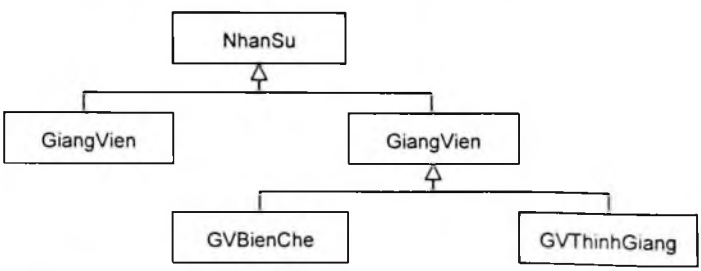
(iii) $\tau_1 \leq_1 \tau_2$ *thì* $\{a_1 : \tau_1\} \leq_1 \{a_1 : \tau_2\}$.

Từ định nghĩa 1.3, dễ nhận thấy τ_1 Sub τ_2 thì $\tau_1 \leq_1 \tau_2$. Khi đó $(T, \leq_1)$ sẽ tạo thành cấu trúc phân cấp theo quan hệ kế thừa. Dựa trên cấu trúc phân cấp này người ta có thể xác định tập các kiểu cơ sở (cha, ông, ...) của một kiểu dẫn xuất (kế thừa) thông qua hàm Sup: $T \rightarrow T \cup L$ được xác định như sau:

$$\text{Sup}(\tau) = \begin{cases} \bigcup_{i=1}^{n} \{\{\tau_i\} \cup \{\text{Sup}(\tau_i)\}\} & \text{nếu } \tau \in \text{I} \\ \varnothing & \text{nếu } \tau \in \text{N} \end{cases}$$

Như vậy, Sup(τ) xác định tất cả các kiểu cơ sở của τ trong quan hệ kế thừa.

Ví dụ 2.1. Xét hệ thống các lớp (kiểu) của hệ thống quản lý nhân sự trong trường đại học được mô tả trong hình 2.2.



Hình 2.2. Cấu trúc phân cấp theo quan hệ kế thừa giữa các lớp

Các cạnh theo mũi tên → xác định quan hệ kế thừa trực tiếp giữa các kiểu, những cạnh trên đường đi của cấu trúc phân cấp ở hình 2.2 đều có quan hệ $\leq_1$ với nhau. Ngoài ra, các kiểu cơ sở của tất cả các kiểu còn được xác định như: Sup(NhanSu) = ∅, Sup(GVBienChe) = {NhanSu, GiangVien}.

Trên cấu trúc phân cấp các kiểu người ta cũng xác định được tập các thể hiện (đối tượng) của một kiểu nào đó dựa trên hai hàm sau:

$o\_type : O \rightarrow T \cup L$ - cho biết kiểu của một thực thể trong hệ thống;

$Ext : T \rightarrow O$ - cho biết các thể hiện của một kiểu sao cho:

$$Ext(\tau) = \{o \in O \mid o\_type(o) \leq_1 \tau\}$$

Cấu trúc phân cấp các kiểu đóng vai trò rất quan trọng trong quá trình thiết kế, xây dựng những hệ thống hướng đối tượng và thiết lập các cấu trúc dữ liệu cho CSDL hướng đối tượng. Một số tính chất mô tả ngữ nghĩa tự nhiên của quan hệ $\leq_1$ được phát biểu như sau [1]:

Tính chất 1: $\forall \tau_1, \tau_2, \tau_3 \in T$

(i) $o\_type(o) = \tau \Rightarrow o \in Ext(\tau)$;

(ii) $o\_type(o) \leq_1 \tau \Leftrightarrow o \in Ext(\tau)$;

(iii) $Ext(\tau_1) \subseteq Ext(\tau_2) \Leftrightarrow \tau_1 \leq_1 \tau_2$.

Tính chất 2: $\forall \tau_1, \tau_2 \in T, \tau_1 \leq \tau_2 \Rightarrow \tau_1 \leq_1 \tau_2$.

Tính chất 3: $\forall \tau_1, \tau_2 \in T, \tau_1 \leq \tau_2 \Leftrightarrow Ext(\tau_1) \subseteq Ext(\tau_2)$.

Định nghĩa 2.4. *Lược đồ CSDL hướng đối tượng (gọi tắt là lược đồ đối tượng) là một bộ ba* $\sum = \langle T, L, Att \rangle$, *trong đó T là tập các kiểu, L là tập các kiểu nguyên thủy và Att là tập thuộc tính của tất cả các đối tượng thỏa mãn các tính chất sau:*

(i) *Tất cả những kiểu* $\tau \in T$ *phải được định nghĩa duy nhất, nghĩa là xuất hiện đúng một lần ở vế trái trong các định nghĩa kiểu của T;*

(ii) *Với mọi kiểu* $\tau \in T$, $\tau \notin Sup(\tau)$, *nghĩa là quan hệ kế thừa không tạo thành chu trình và những thuộc tính bổ sung không lặp lại các thuộc tính ở kiểu cơ sở.*

(iii) *Không có các bộ lồng nhau trong định nghĩa kiểu, mọi kiểu kế thừa và khai báo thuộc tính phải thông qua tên gọi kiểu.*

Định nghĩa 2.5. *Lược đồ đối tượng được gọi là lược đồ đối tượng dạng chuẩn nếu tất cả các kiểu* $\tau \in T$ *đều là kiểu chuẩn* ($\tau \in N$).

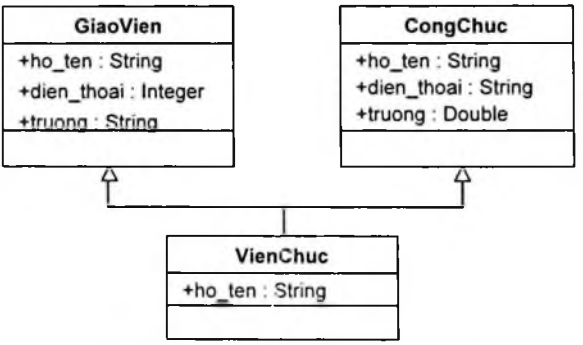
Hiển nhiên lược đồ đối tượng dạng chuẩn có $T = N$.

Ví dụ 2.2. Xét lược đồ gồm các lớp như sau:

Type GiaoVien = {ho_ten : String; dien_thoai : Integer; truong : String};

Type CongChuc = {ho_ten : String; dien_thoai : String; luong : Double};

Type VienChuc = GiaoVien, CongChuc {dia_chi : String}.



Hình 2.3. Biểu diễn bằng ký pháp UML của lược đồ trong ví dụ 2.2

Kiểu GiaoVien, CongChuc là ở dạng chuẩn còn VienChuc được thiết lập dựa trên quan hệ kế thừa *Sub* và không ở dạng chuẩn.

Quá trình xác định các thành phần của những kiểu kế thừa từ các siêu kiểu có thể dẫn đến xung đột hay không nhất quán về kiểu và phương thức xử lý. Chẳng hạn, kiểu VienChuc được kế thừa từ GiaoVien và CongChuc. Vấn đề này sinh khi kiểu dẫn xuất kế thừa cùng tên thuộc tính nhưng kiểu tương ứng ở các siêu kiểu lại không tương thích, như dien_thoai ở GiaoVien có kiểu Integer còn ở CongChuc lại khai báo kiểu String. Tất nhiên kiểu Integer và String không tương thích với nhau, cho

nên lược đồ ở ví dụ 2.2 là không nhất quán. Những thuộc tính có kiểu không nhất quán dẫn ra từ quan hệ kế thừa là không xác định và ký hiệu là ⊥. Khi đó, kiểu VienChuc được viết lại:

Type VienChuc = {ho_ten : String; dien_thoai : ⊥;
luong : Double; dia_chi : String}

Lược đồ VienChuc sẽ trở thành nhất quán nếu khai báo thuộc tính dien_thoai của GiaoVien và CongChuc cùng một kiểu, chẳng hạn là kiểu Integer. Từ đó, dễ nhận thấy là những lược đồ nhất quán có thể chuyển đổi thành lược đồ dạng chuẩn nếu mọi kiểu $\tau \in T$ không ở dạng chuẩn đều được chuyển về dạng chuẩn sau hữu hạn lần áp dụng quan hệ kế thừa để xác định tường minh các thành phần của τ.

Quá trình xác định tường minh các thành phần của các kiểu trong quan hệ kế thừa dẫn đến việc xác định kiểu con chung lớn nhất của các siêu kiểu của chúng. Một số thuộc tính của kiểu này có thể tham chiếu tới các kiểu khác và quá trình dẫn xuất để tìm kiểu con chung lớn nhất với các tham chiếu qua lại mà không kết thúc được.

Ví dụ 2.3. Xét lược đồ sau:

Type Nguoi = {ho_ten : String; tuoi : Integer; ban_huu : CongNhan}

Type CongNhan = {ho_ten: String; co_quan : String; ban_huu : Nguoi}

Type CanBo = {ho_ten : String; luong : Double; ban_huu : CanBo}

Type NhanVien = CongNhan, CanBo {dia_chi : String}.

Việc xác định các thành phần của NhanVien phải dựa vào các kiểu CongNhan và CanBo. Ở kiểu CongNhan thuộc tính ban_huu tham chiếu tới Nguoi, tiếp theo ở Nguoi thuộc tính ban_huu lại tham chiếu về CongNhan. Quá trình trên có thể lặp lại nhiều lần và rất khó khẳng định tính nhất quán của lược đồ đối tượng.

Để xây dựng được những hệ thống CSDL hướng đối tượng tin cậy, nhất quán, người ta phải nghiên cứu phương pháp kiểm tra tính đúng đắn, nhất quán dữ liệu lược đồ tương ứng. Vì vậy, trước hết người ta cần xác định được mối quan hệ giữa các lớp đối tượng trong hệ thống.

2.2.4. Các tính chất của quá trình kế thừa kiểu

2.2.4.1. Quan hệ thứ tự kiểu con

Người ta xây dựng quan hệ thứ tự kiểu con $\leq$ giữa các kiểu có quan hệ kế thừa để nghiên cứu các tính chất của tất cả thành phần đối với mọi kiểu trong hệ thống, các kết quả sau đây đã được chứng minh trong [1].

Định nghĩa 2.6. *Cho trước lược đồ* $\sum = \langle T, L, Att \rangle$, *trên* T *xác định một quan hệ* $\leq$ *:*

(i) $\forall \tau \in T \cup L,\ \tau \leq \tau$;

(ii) *Nếu* $\tau_1 = \{a_i : \eta_i; c_k : \gamma_k\}$ *và* $\tau_2 = \{a_i : \mu_i\}$ *thì* $\tau_1 \leq \tau_2 \Leftrightarrow \eta_i \leq \mu_i$
Nếu $\tau_1 \leq \tau_2$ *thì ta nói rằng* τ_1 *là kiểu con của* τ_2.

Ví dụ 2.4.

```
Type NhanSu = {ho_ten : String; dien_thoai : Integer};
Type GiaoVien = {ho_ten : String; dien_thoai : Integer; truong : String};
```

Rõ ràng là GiaoVien $\leq$ NhanSu.

Định lý 2.2. *Quan hệ kiểu con* $\leq$ *là quan hệ thứ tự bộ phận.*

Định nghĩa 2.7. *Phân cấp kiểu là một bộ ba* $\Gamma = (C, \sigma, \leq)$, *với* C *là tập các tên lớp,* σ *là ánh xạ từ* C *đến* $\mathbf{Types}(C) \subset T \cup L$ *và* $\leq$ *là một quan hệ thứ tự bộ phận - quan hệ lớp con.*

Từ cấu trúc phân cấp dựa trên quan hệ kế thừa kiểu $\leq_1$ quan hệ thứ tự bộ phận $\leq$ được xây dựng để nghiên cứu những tính chất của hệ thống kiểu trong lược đồ đối tượng của hệ thống hướng đối tượng. Sử dụng những quan hệ này, người ta tiếp tục nghiên cứu để biến đổi tất cả những kiểu không ở dạng chuẩn về dạng chuẩn đối với những lược đồ phi mâu thuẫn dựa trên việc xây dựng quá trình xác định kiểu con chung nhất của các kiểu cơ sở ở từng mức và nghiên cứu được tính đúng đắn, nhất quán dữ liệu của hệ thống dựa trên đồ thị quan hệ kế thừa hoặc cấu trúc dàn của hệ thống kiểu.

2.2.4.2. Phép tụ tập kiểu và quá trình kế thừa

Định nghĩa 2.8. *Phép tụ tập kiểu* $\downarrow$ *xác định quá trình kế thừa, được định nghĩa như sau:*

(i) $\tau \downarrow \tau = \tau,\ \tau \in T \cup L;$

(ii) *Nếu* $\tau_1 \neq \tau_2$ *và* τ_1 *hoặc* τ_2 *nguyên thủy thì* $\tau_1 \downarrow \tau_2 = \perp;$

(iii) *Nếu* $\tau_1 = \{a_i : \eta_i; b_j : \mu_j\}$, $\tau_2 = \{a_i : \eta'_i; c_k : \gamma_k\}$, $i = 1, 2, ..., m$, $j = 1, 2, ..., n$, $k = 1, 2, ..., l$ *và* $\forall j, k : b_j \neq c_k$ *thì khi đó:*

$$\tau_1 \downarrow \tau_2 = \perp \Leftrightarrow \exists\, i : \eta_i \downarrow \eta'_i = \perp;$$

$$\tau_1 \downarrow \tau_2 = \{a_i : \eta_i \downarrow \eta'_i; b_j : \mu_j; c_k : \gamma_k\} \Leftrightarrow \forall i : \eta_i \downarrow \eta'_i \neq \perp.$$

Trong đó, $\tau_1 \downarrow \tau_2 = \perp$ *nghĩa là phép tụ tập* $\downarrow$ *của* τ_1 *với* τ_2 *là không xác định (kiểu rỗng). Hay nói cách khác, trong quan hệ kế thừa từ* τ_1 *và* τ_2 *sẽ sử dụng chung những thuộc tính liên quan tới* τ_1 *và* τ_2 *mà bản thân chúng lại không tương thích nên dẫn tới mâu thuẫn.*

Ví dụ 2.5. Cho trước hệ thống kiểu:

Type Nguoi = {tuoi : Integer; con : SinhVien};

Type SinhVien = {ho_ten : String; truong : String; ban : Nguoi};

Type GiaoVien = {ho_ten : String; luong : Double; ban : GiaoVien};

Type NhanVien = SinhVien, GiaoVien { }.

Để xác định được các thuộc tính một cách tường minh của NhanVien phải thay thế SinhVien và GiaoVien bằng những thuộc tính của chúng sao cho không xảy ra mâu thuẫn. Hai kiểu SinhVien, GiaoVien có hai thuộc tính chung là ho_ten cùng kiểu String còn ban có kiểu Nguoi ở SinhVien, kiểu GiaoVien ở GiaoVien. Ta có, NhanVien = SinhVien $\downarrow$ GiaoVien = {ho_ten : String; ban : Nguoi $\downarrow$ GiaoVien; truong : String; luong : Double} với Nguoi $\downarrow$ GiaoVien = {ho_ten : String; tuoi : Integer; con : SinhVien; ban : GiaoVien}.

Như vậy nếu đặt Nguoi $\downarrow$ GiaoVien là kiểu mới để đảm bảo không có các bộ lồng nhau, chẳng hạn VienChuc thì kiểu NhanVien trong hệ thống trên không ở dạng chuẩn chuyển được về dạng chuẩn như sau:

Type NhanVien = {ho_ten : String; ban : VienChuc; truong : String; luong : Double};

Type VienChuc = {ho_ten : String; tuoi : Integer; con : SinhVien; ban : GiaoVien}.

Hai kiểu được gọi là *không tương thích* nếu kết quả của phép tụ tập kiểu $\downarrow$ của chúng là không xác định.

Ví dụ 2.6. Xét lược đồ đối tượng

Type Nguoi = {tuoi : Integer; con : SinhVien};

Type SinhVien = {ho_ten : String; truong : String; ban : Nguoi};

Type GiaoVien = {ho_ten : Hovaten; luong : Double ; ban : GiaoVien};

Type Hovaten = {ho : String; ten : String};

Type NhanVien = SinhVien, GiaoVien {dia_chi : String}.

Bởi vì String và Hovaten là không tương thích nên theo định nghĩa thì String $\downarrow$ Hovaten = $\perp$. Do vậy SinhVien $\downarrow$ GiaoVien = $\perp$, nghĩa là kiểu NhanVien được định nghĩa kế thừa từ SinhVien, GiaoVien sẽ dẫn đến mâu thuẫn. Nói cách khác, kiểu NhanVien định nghĩa trong ví dụ 2.6 là không xác định, vì vậy hệ thống này có mâu thuẫn.

Định nghĩa 2.9. *Quá trình kế thừa của $\tau \in I$ là kết thúc nếu số lần thực hiện phép tụ tập kiểu $\downarrow$ kết thúc sau hữu hạn lần áp dụng điều kiện* (iii) *của định nghĩa 2.8.*

Phương pháp thực hiện biến đổi những kiểu không phải là kiểu chuẩn $\tau \in I$ thành kiểu chuẩn $\tau \in N$ như sau:

(i) Đối với kế thừa đơn: $\tau_1 = \{a_i : \eta_i\} \in N$, $\tau \in I$, $\tau = \tau_1 \{b_j : \mu_j\}$, $a_i \neq b_j$, sau khi thế các bộ phận của τ_1 vào τ, ta có: $\tau = \{a_i : \eta_i ; b_j : \mu_j\}$;

(ii) Đối với kế thừa bội: $\tau = \tau_1, \tau_2, \ldots, \tau_n \{b_j : \mu_j\} \in I$, $n > 1$.

Nếu quá trình tính $\tau_1 \downarrow \tau_2 \downarrow \ldots \downarrow \tau_n$ kết thúc và $\tau_1 \downarrow \tau_2 \downarrow \ldots \downarrow \tau_n = \{a_i : \eta_i\}$ (xác định) thì $\tau = \{a_i : \eta_i ; b_j : \mu_j\}$. Ngược lại, nếu $\tau_1 \downarrow \tau_2 \downarrow \ldots \downarrow \tau_n = \perp$ thì $\tau = \perp$.

Vấn đề đặt ra ở giai đoạn này là kiểm tra tính kết thúc của quá trình kế thừa. Dễ nhận thấy nếu trong quan hệ kế thừa có xuất hiện đệ quy thì quá trình phân giải kế thừa có thể bị lặp lại vô hạn [2].

Ví dụ 2.7. Xét lược đồ sau:

Type Nguoi = {ho_ten : String; tuoi : Integer; ban : CongNhan};

Type CongNhan = {ho_ten : String; co_quan : String; ban : Nguoi};

Type CanBo = {ho_ten : String; luong : Double ; ban : CanBo};

Type NhanVien = CongNhan, CanBo {dia_chi : String}.

Để chuyển NhanVien về dạng chuẩn ta tính CongNhan ↓ CanBo. Vấn đề là xác định được kiểu cho thuộc tính ban hay không. Kiểu CongNhan có thuộc tính ban tham chiếu tới Nguoi và tiếp theo kiểu Nguoi có thuộc tính ban lại tham chiếu đệ quy về CongNhan. Quá trình trên có thể lặp lại nhiều lần và rất khó khẳng định tính nhất quán của các kiểu đó khi các phép tham chiếu lặp lại và không kết thúc. Để giải quyết vấn đề này, người ta dựa vào tính chất của đồ thị đặc trưng cho quan hệ kế thừa giữa các kiểu của lược đồ (gọi là *s-đồ thị*) để kiểm tra tính dừng của quá trình kế thừa và tính phi mâu thuẫn của hệ thống kiểu.

2.1.4.3. s-ĐỒ THỊ

Định nghĩa 2.10. *Cho trước lược đồ* $\sum = \langle T, L, Att \rangle$, *s-đồ thị của lược đồ* $\sum$ *là một đồ thị định hướng có gắn nhãn* $G = (V, E)$, *trong đó:*

+ *Tập các đỉnh* $V = T \cup L$;

+ *Tập cung* $E = \{(\tau_1, l, \tau_2) \subseteq T \times Att \cup \{h\} \times T \cup L\}$; $(\tau_1, l, \tau_2) \in E \Leftrightarrow \tau_1$ *kế thừa từ* τ_2 (τ_1 *Sub* τ_2) *và* $l = h$ *hoặc* $\tau_1 = \{..., a : \tau_2, ...\}$ *và* $l = a$, *nhãn* $l = h$ *hoặc* $l \in Att$.

s-đồ thị mô tả mối quan hệ giữa các kiểu trong hệ thống. Những cung có nhãn là *h* trong đồ thị mô tả quan hệ kế thừa cha/ con của hai đỉnh còn những cung có nhãn khác là những thuộc tính của một kiểu (đỉnh xuất phát) có nhận các giá trị thuộc kiểu mô tả ở đỉnh đến theo đường chỉ của mũi tên. Trong *s-đồ thị*, những kiểu không ở dạng

chuẩn sẽ tương ứng với những đỉnh có các cung đi tới các đỉnh ứng với kiểu được kế thừa và có nhãn là *h*.

Ví dụ 2.8. Xét lược đồ sau:

Type CongNhan = {ho_ten : String; luong : Double; phan_xuong : String};

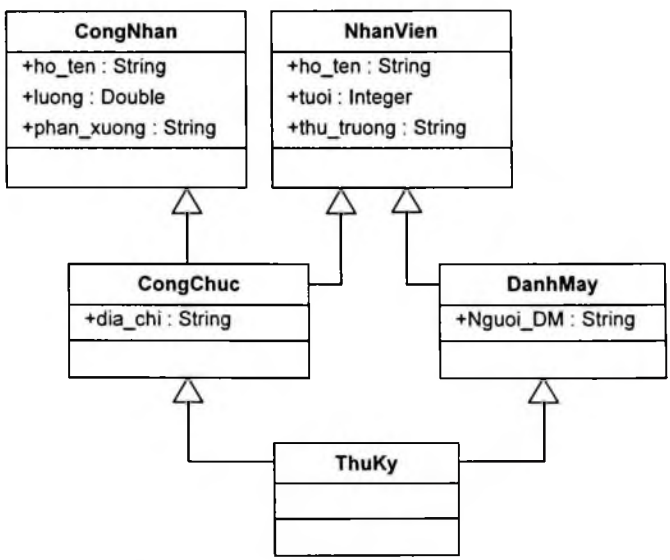
Type NhanVien = {ho_ten : String; tuoi : Integer; thu_truong : String};

Type DanhMay = NhanVien {nguoi_DM : String};

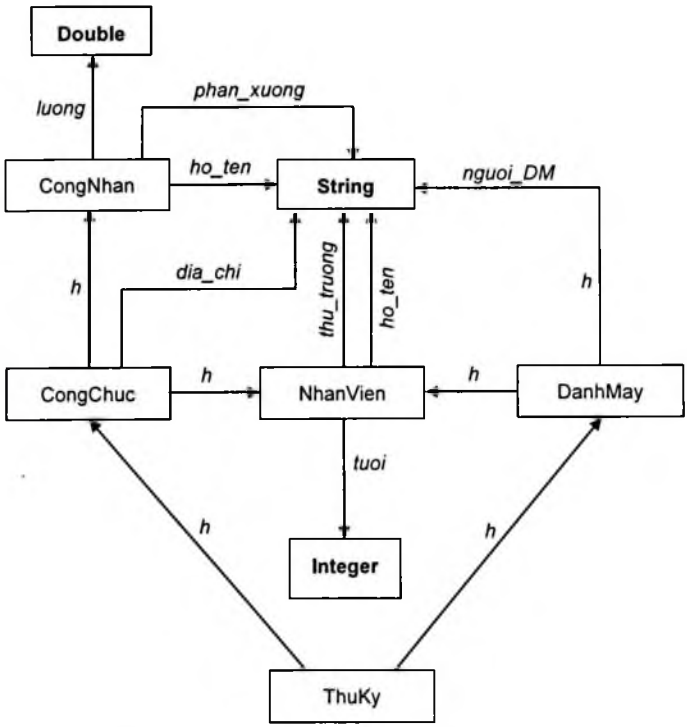
Type CongChuc = CongNhan, NhanVien {dia_chi : String};

Type ThuKy = CongChuc, DanhMay { }.

T = {CongNhan, NhanVien, DanhMay, CongChuc, ThuKy}, L = {Integer, Double, String} và Att = {ho_ten, tuoi, luong, thu_truong, phan_xuong, nguoi_DM, dia_chi}.



Hình 2.4. Biểu diễn bằng ký pháp UML của lược đồ ví dụ 2.8



Hình 2.5. *s-đồ thị* của lược đồ trong ví dụ 2.8

Nói chung, *s-đồ thị* không phải là đồ thị đơn, có thể tồn tại những cung song song và có vòng khuyên (cung tự trỏ). Trực tiếp từ định nghĩa, suy ra *s-đồ thị* của lược đồ dạng chuẩn là đồ thị không có cung có nhãn là *h*.

Với cơ chế kế thừa bội (một kiểu được kế thừa từ nhiều hơn một kiểu cơ sở) thường dẫn tới sự xung đột hay mâu thuẫn trong hệ thống. Do đó, để kiểm tra xem một lược đồ có xung đột hay không thì chỉ

cần kiểm tra trên *s-đồ thị* của nó có những đường đi bắt đầu từ những đỉnh mà kiểu tương ứng không ở dạng chuẩn, có các dãy nhãn (thuộc tính) bằng nhau sau khi loại bỏ các cung *h* trên các đường đi, có dẫn tới những đỉnh tương ứng với các kiểu không tương thích?

Để hình thức hóa cho quá trình mô tả trên, người ta sử dụng một số khái niệm liên quan đến các đường đi trong s-đồ thị.

Định nghĩa 2.11. *Cho trước s-đồ thị* $G = (V, E)$, $p = (s, d, \langle a_1.a_2...a_n\rangle)$ *là đường đi của G từ đỉnh s đến d nếu* $\exists (\tau_i, a_i, \tau_{i+1}) \in E$, $i = 1, 2, ..., n-1$ *và* $\tau_1 = s$, $\tau_n = d$, $n \geq 1$.

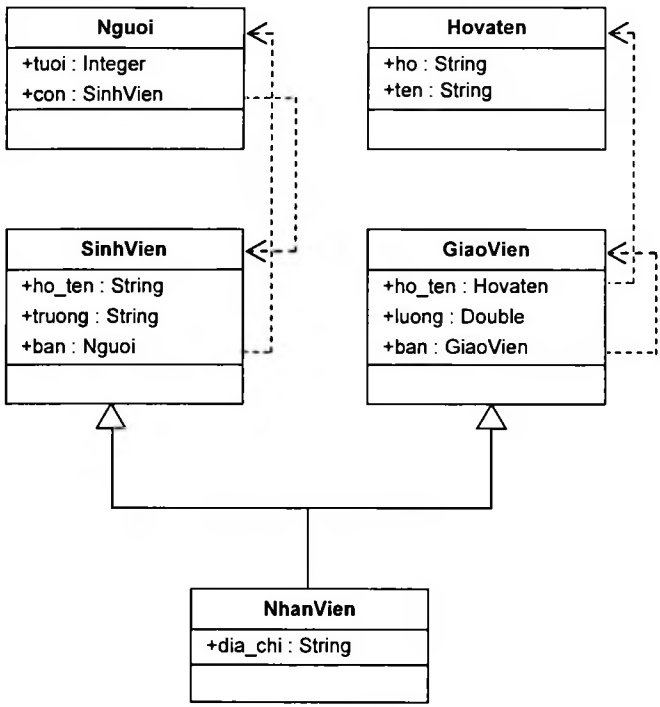
Định nghĩa 2.12. *Hai đường đi trong s-đồ thị* $G = (V, E)$, $p_1 = (s_1, d_1, \langle a_1.a_2...a_n\rangle)$ *và* $p_2 = (s_2, d_2, \langle b_1.b_2...b_n\rangle)$ *được gọi là tựa tương đẳng, ký hiệu là* $p_1 \cong p_2$ *nếu và chỉ nếu hai dãy nhãn* $\langle a_1.a_2...a_n\rangle$, $\langle b_1.b_2...b_n\rangle$ *cho cùng một kết quả sau khi loại bỏ đi nhãn có tên h và* $d_1 \neq d_2$.

Dễ nhận ra rằng, dãy các nhãn của một đường đi sau khi loại bỏ nhãn *h* chính là dãy các thuộc tính của kiểu tương ứng từ đỉnh xuất phát và thu được từ quá trình phân giải kế thừa thông qua phép tụ tập kiểu ↓.

Định nghĩa 2.13. *s-đồ thị có mâu thuẫn khi và chỉ khi tồn tại ít nhất hai đường đi tựa tương đẳng cùng bắt đầu từ một đỉnh (kiểu con không ở dạng chuẩn và kế thừa bội) và đi đến hai đỉnh đích là hai kiểu không tương thích (phép tụ tập kiểu ↓ của chúng không xác định). Lược đồ tương ứng với s-đồ thị không có mâu thuẫn được gọi là lược đồ phi mâu thuẫn.*

Ví dụ 2.9. Xét lược đồ sau:

```
Type Nguoi = {tuoi : Integer; con : SinhVien};
Type SinhVien = {ho_ten : String; truong : String; ban : Nguoi};
Type GiaoVien = {ho_ten : Hovaten; luong : Double ; ban : GiaoVien};
Type Hovaten = {ho : String; ten : String};
Type NhanVien = SinhVien, GiaoVien {dia_chi : String}.
```



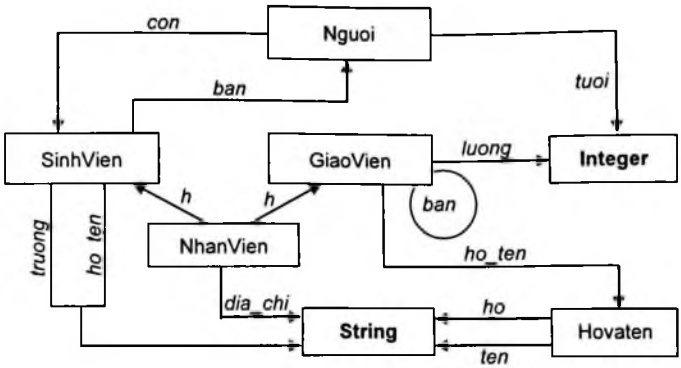
Hình 2.6. Biểu diễn bằng ký pháp UML của lược đồ ví dụ 2.9

Xét hai đường đi tựa tương đẳng trong đồ thị:

(NhanVien, Hovaten, ⟨*h.ho_ten*⟩) ≅ (*NhanVien*, *String*, ⟨*h.ho_ten*⟩)

Hai đường đi này dẫn đến hai kiểu không tương thích: String (kiểu nguyên thủy) và kiểu Hovaten (được định nghĩa), String ↓ Hovaten = ⊥.

Vậy một lần nữa dựa vào *s-đồ thị* ta khẳng định lược đồ cho trước trong ví dụ 2.9 có mâu thuẫn.



Hình 2.6. s-đồ thị của lược đồ trong ví dụ 2.9

Định nghĩa 2.14. *Lược đồ* $\sum = \langle T, L, Att \rangle$ *được gọi là đúng đắn khi và chỉ khi nó là phi mâu thuẫn và với mọi kiểu* $\tau \in T$ *(không dạng chuẩn* $\tau \in I$*) quá trình kế thừa là kết thúc.*

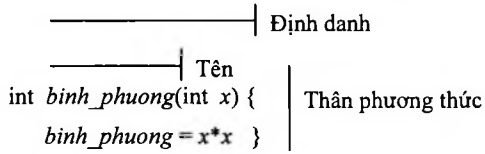
2.2.5. Phương thức của lớp

Phương thức là thành phần mô tả hành vi ứng xử của các đối tượng trong lớp. Để nghiên cứu các tính chất của phương thức, người ta sẽ hình thức hóa các khái niệm về phương thức. Một trong những đặc tính quan trọng của mô hình hướng đối tượng là cho phép các phương thức thành phần của các lớp được *nạp chồng* (overloading), *viết đè* (overriding) và *liên kết muộn* (late binding), nghĩa là các phương thức có cùng một tên gọi nhưng có thể thực thi với nhiều nội dung thực hiện khác nhau. Do vậy, một phương thức của lớp phải được xác định qua định danh của nó. Định nghĩa về phương thức của lớp đối tượng được nêu trong định nghĩa 2.15.

Định nghĩa 2.15. *Phương thức có ba thành phần: tên, định danh và thân của phương thức. Tập* ***Meth*** *là tập vô hạn các tên của phương thức. Cho* $\Gamma = (C, \sigma, \leq)$ *là một phân cấp kiểu. Đối với phương*

thức tên là m, định danh của m là biểu thức có dạng: $m : c \times \tau_1 \times ... \tau_{n-1} \times \tau_n$, trong đó c là tên lớp trong C và mỗi τ_i là một kiểu trên C. Định danh này liên kết với lớp c; phương thức m áp dụng cho các đối tượng của lớp c.

Ví dụ 2.10. Trong lớp Integer, xét phương thức:



Các phương thức có tên giống nhau có các định danh khác nhau và liên kết với các lớp khác nhau, nhưng nếu $m : c_1 \times \tau_1 \times ... \tau_{n-1} \rightarrow \tau_n$ và $m : c_2 \times \tau'_1 \times ... \tau'_{p-1} \rightarrow \tau'_p$ là hai định nghĩa của *m* và $c_1 \leq c_2$, thì $n = p$ và $\tau_i \leq_1 \tau'_i$ ($i = 1, ..., n$). Luật này được gọi là *đối biến thể* của định nghĩa phương thức *m* trong c_1 và c_2.

Ví dụ 2.11. Xét các phương thức nạp chồng xác định giá trị nhỏ nhất của hai số trong lớp *Math* như sau:

double *min*(double *a*, double *b*)

int *min*(int *a*, int *b*)

long *min*(long *a*, long *b*)

Trong ví dụ 2.11, ta có các phương thức tìm giá trị nhỏ nhất của hai phần tử tương ứng với kiểu của các đối tượng. Vấn đề đặt ra là tính giá trị nhỏ nhất cho một tập mà kiểu của các phần tử thuộc tập không biết trước vào thời điểm thực thi. Trong hệ thống quy ước, người ta sẽ xây dựng ba phép toán *min_double*, *min_int*, *min_long*. Người lập trình phải kiểm tra kiểu của mỗi đối tượng trong tập và sử dụng một trong các phép toán tương ứng trên, đoạn chương trình giả mã như sau:

```
x = x1
for xi in X do
  begin
    case of type(xi)
      double: min_double(x, xi);
    int: min_int(x, xi);
       long: min_long(x, xi);
  end
end
```

Trong mô hình hướng đối tượng, người ta định nghĩa phép toán theo cấp của kiểu đối tượng (siêu kiểu trong hệ thống). Vì vậy, phương thức *min* chỉ có một tên duy nhất và được sử dụng cho ba kiểu đối tượng là double, int và long. Tuy nhiên, phải định nghĩa lại sự cài đặt phương thức đối với mỗi kiểu tương ứng với kiểu của đối tượng, sự định nghĩa lại này gọi là *viết đè*. Để tính giá trị nhỏ nhất của một tập các phần tử, chỉ cần gọi phương thức *min* với các phần tử đầu vào và hệ thống sẽ chọn cài đặt tương ứng cho phương thức khi thực thi.

```
x = x1
for xi in X do x = min(x, xi)
```

Mặt khác, hệ thống không thể liên kết tên của phương thức với chương trình vào thời điểm biên dịch. Vì vậy, tên của các phương thức phải được biên dịch vào trong các địa chỉ chương trình vào thời điểm chạy chương trình. Sự biên dịch "muộn" này được gọi là *liên kết muộn.*

Nếu phương thức *m* được định nghĩa trong lớp c_1 và c_2 là lớp con trực tiếp của c_1, thì định nghĩa của *m* với c_2 được *kế thừa* từ c_1. Định danh của *m* trong c_2 có dạng $m : c_2 \times \tau_1 \times ... \tau_{n-1} \rightarrow \tau_n$; định danh của *m* trên c_2 là đồng nhất với định danh của *m* trên c_1, ngoại trừ thành phần thứ nhất c_1 được thay bằng c_2. Tổng quát, lớp c_2 kế thừa phương thức *m* từ lớp c_1 mà *m* đã được định nghĩa. Nếu $c_2 \leq_1 c_1$, *m* không định

nghĩa trong c_2 và m không định nghĩa trong bất kỳ lớp c_i nào, thì $c_2 \leq_1 c_i$, $c_i \leq_1 c_1$. Nếu lớp c_2 kế thừa phương thức m từ lớp c_1, thì thân của m trong c_2 là đồng nhất với thân của m trong c_1.

Tập các định danh phương thức được ký hiệu là **MId** liên kết với phân cấp kiểu $(C, \sigma, \leq)$. Mỗi lớp c của C liên kết với một tập các định danh phương thức, ký hiệu $\mu(c)$ là tập các phương thức định nghĩa trong c hoặc được lớp c kế thừa từ các siêu lớp của nó, $\mathbf{MId} = \bigcup\{\mu(c) | c \in C\}$.

Các đối tượng trong lớp c chỉ được truy nhập tập các phương thức $\mu(c)$. Các phương thức được định nghĩa trong một lớp sử dụng các thuộc tính của $\sigma(c)$ và phụ thuộc trên tập các thuộc tính đã sử dụng trong định nghĩa của chúng. Phương thức cho phép nhận kết quả hoặc hiệu chỉnh giá trị của các thuộc tính của các đối tượng trong lớp c. Tập các phương thức áp dụng cho đối tượng được gọi là *giao diện* của đối tượng. Các đối tượng chỉ được truy nhập qua giao diện của chúng, nguyên lý này gọi là *tính đóng gói*.

Quy tắc đảm bảo tính nhất quán trong sự kế thừa của các phương thức như sau: Nếu c_1 là lớp con của c_2, c_3 và có một định nghĩa của phương thức m đối với c_2 và c_3 thì tồn tại một định nghĩa của m đối với lớp con của c_2 và c_3, đó cũng chính là định nghĩa của m đối với c_1 hoặc siêu lớp của c_1.

Định nghĩa 2.16. *Cho $\Gamma = (C, \sigma, \leq)$ là một phân cấp kiểu. Lớp được định nghĩa là bộ ba: $c = (cn, SC, M)$, trong đó cn là tên lớp trong C, SC là tập có thứ tự các siêu lớp của c và $M = M_s \cup M_c$, M_s là tập các phương thức được kế thừa từ các siêu lớp của c, M_c là tập các phương thức định nghĩa trong lớp c.*

Xem xét quan hệ "bằng nhau" đối với hai kiểu/ lớp qua ví dụ 2.12 trong ngôn ngữ lập trình C# (ví dụ 2.12).

Ví dụ 2.12. C# cung cấp bốn hàm xác định các đối tượng (lớp hoặc struct) là "bằng nhau", gồm có:

```
public static bool ReferenceEquals(object left, object right);
public static bool Equals(object left, object right);
public virtual bool Equals(object right);
public static bool operator==(MyClass left, MyClass right);
```

Quan hệ bằng nhau giữa hai lớp được xét trên các kiểu giá trị và các kiểu tham chiếu. Hai đối tượng của kiểu tham chiếu là bằng nhau nếu chúng tham chiếu đến cùng một đối tượng - đối tượng đồng nhất. Hai đối tượng của một kiểu giá trị là bằng nhau nếu chúng có cùng kiểu và chứa cùng các nội dung.

Xét phương thức static Object.ReferenceEquals, phương thức này so sánh các tham chiếu của hai đối tượng, nghĩa là, nó kiểm tra các đối tượng đồng nhất. Phương thức ReferenceEquals trả về giá trị true nếu và chỉ nếu, hai đối tượng tham chiếu đến cùng đối tượng. Do đó, nó sẽ luôn trả về giá trị false nếu so sánh hai kiểu giá trị. Tiếp theo, xét phương thức static Object.Equals. Phương thức này kiểm tra hai đối tượng bằng nhau khi không biết các kiểu thực thi của chúng và phương thức operator==() so sánh bằng trên kiểu giá trị.

Định nghĩa 2.17. *CSDL hướng đối tượng là cặp: odb = (Γ, O), trong đó, Γ = (C, σ, ≤) là phân cấp lớp và O là tập các đối tượng thỏa mãn các điều kiện sau:*

(i) $\forall$ *c* $\in$ *C thì tên của lớp c.cn là duy nhất;*

(ii) $\forall$ *o* $\in$ *O, o.cn là tên của lớp trong C;*

(iii) $c_{root} \in \Gamma$;

(iv) $\forall c \in \Gamma, c \leq c_{root}$ - *c là lớp con của* c_{root}.

Trong định nghĩa 2.17, điều kiện (i), (ii) ràng buộc mỗi đối tượng trong CSDL chỉ thuộc về duy nhất một lớp. Các điều kiện (iii), (iv) bảo đảm sự đệ quy của quan hệ kế thừa lớp là kết thúc đối với mọi lớp trong CSDL.

2.3. MỞ RỘNG NGỮ NGHĨA CỦA MÔ HÌNH DỮ LIỆU HƯỚNG ĐỐI TƯỢNG

Nhằm cung cấp một mô hình dữ liệu đầy đủ có thể cài đặt một cách trực tiếp trong các hệ CSDL, người ta quan tâm đến sự mở rộng ngữ nghĩa của mô hình dữ liệu hướng đối tượng qua các khái niệm đối tượng phức hợp và phiên bản của lược đồ. Nếu những khái niệm này không có trong mô hình hạt nhân thì chúng sẽ được hỗ trợ trực tiếp bởi hệ CSDL tương ứng.

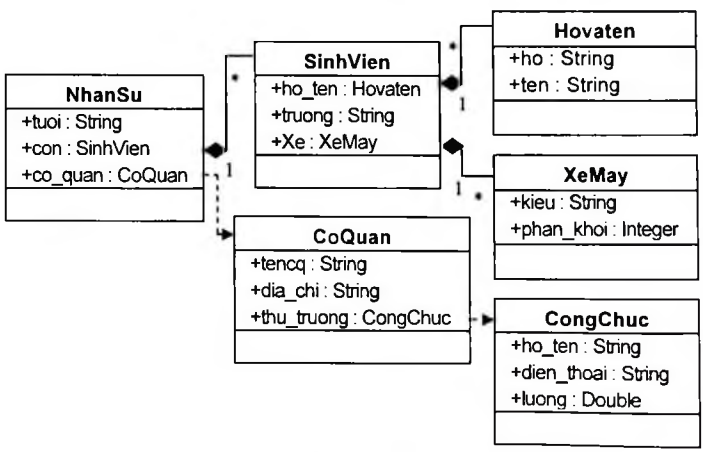
2.3.1. Đối tượng phức hợp

Trong hệ thống hướng đối tượng, đối tượng có một tập thuộc tính và giá trị của thuộc tính có thể là một đối tượng khác, đó là thể hiện miền của thuộc tính. Nếu miền của thuộc tính là lớp nguyên thủy thì giá trị của thuộc tính là một thể hiện hay là một tập các thể hiện của miền. Nếu miền của thuộc tính không phải là lớp nguyên thủy thì giá trị của thuộc tính là định danh đối tượng với thể hiện của miền. Trong trường hợp này, ta nói rằng đối tượng tham chiếu đến các đối tượng khác, nghĩa là, các thể hiện của những thuộc tính của đối tượng có miền không nguyên thủy. Định danh đối tượng trên được gọi là tham chiếu đến những thể hiện đó.

Sự tham chiếu đối tượng thường bao hàm quan hệ "*bộ phận*" (*part-of*) giữa một đối tượng và đối tượng khác tham chiếu đến nó. Có hai loại tham chiếu: *tham chiếu yếu* và *tham chiếu phức hợp*. Tham chiếu yếu là tham chiếu đối tượng nhưng không bao hàm ngữ nghĩa "*bộ phận*". Tham chiếu phức hợp là tham chiếu yếu và được bổ sung thêm quan hệ "*bộ phận*". Ngữ nghĩa của tham chiếu phức hợp được xét một cách tinh tế hơn trên cơ sở của đối tượng là một bộ phận của một hay nhiều đối tượng. Từ đây người ta sẽ xét hai loại tham chiếu phức hợp: riêng biệt và chia sẻ. Tham chiếu phức hợp riêng biệt từ đối tượng X đến đối tượng Y, nghĩa là Y là một phần của chỉ đối tượng X; trong khi tham chiếu phức hợp chia sẻ từ X đến Y, nghĩa là Y là một bộ phận của X và có thể của các đối tượng khác.

Nếu xét thêm ngữ nghĩa của tham chiếu trên cơ sở sự tồn tại của một đối tượng phụ thuộc vào sự tồn tại của đối tượng khác, tức là tham chiếu phức hợp có thể phụ thuộc hoặc độc lập. Tham chiếu phức hợp phụ thuộc từ *X* đến *Y*, tức là sự tồn tại của *Y* phụ thuộc vào sự tồn tại của *X*, trong khi tham chiếu phức hợp độc lập không mang ngữ nghĩa này. Như vậy, có bốn loại tham chiếu phức hợp: Tham chiếu phức hợp phụ thuộc riêng biệt, tham chiếu phức hợp độc lập riêng biệt, tham chiếu phức hợp phụ thuộc chia sẻ và tham chiếu phức hợp độc lập chia sẻ.

Tổng quát, đối tượng phức hợp là tập hợp không thuần nhất các đối tượng, trong đó các đối tượng thành phần được tập hợp từ một số lớp khác nhau. Các lớp chứa các đối tượng thành phần của một đối tượng phức hợp được sắp xếp trong một phân cấp gọi là *phân cấp thuộc tính phức hợp*, mỗi lớp trong phân cấp được gọi là *lớp thành phần*, thể hiện mối quan hệ "*bộ phận - tổng thể*". Đây là một dạng của quan hệ kết hợp, còn được gọi là quan hệ kết tập. Phân cấp thuộc tính phức hợp là một tập con của phân cấp lớp phức hợp. Nếu thuộc tính có tham chiếu phức hợp, thì nó được gọi là *thuộc tính phức hợp*.



Hình 2.8. Ví dụ về phân cấp lớp phức hợp (quan hệ kết tập)

Trong đó, đối tượng NhanSu có thuộc tính con là thuộc tính phức hợp, tham chiếu phức hợp độc lập chia sẻ từ NhanSu đến SinhVien. Thuộc tính co_quan là thuộc tính tham chiếu yếu đến đối tượng CoQuan.

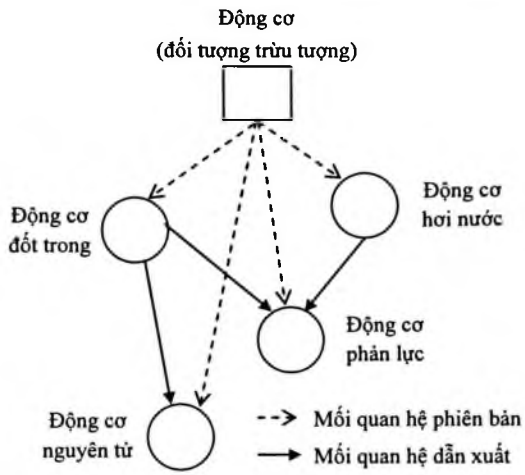
2.3.2 Phiên bản của lược đồ

Trong một hệ CSDL chỉ có một lược đồ đơn, lược đồ có thể bị phá vỡ nếu người sử dụng được phép thực hiện các thay đổi bất kỳ đối với lược đồ. Đối với hệ thống hướng đối tượng, các tác động của sự thay đổi lược đồ sẽ phá vỡ một cách cục bộ lược đồ, vì sự thay đổi các thuộc tính hay phương thức xác định của một lớp phải được lan truyền một cách đệ quy đến tất cả các lớp con của lớp.

Để giải quyết vấn đề này trong hệ thống hướng đối tượng, người ta sử dụng phiên bản lược đồ - là tính chất đặc trưng của sự tiến hóa lược đồ. Nếu người sử dụng muốn cập nhật lược đồ, hệ thống sẽ tạo một phiên bản mới từ một phiên bản đã có của lược đồ và thực hiện các hiệu chỉnh trên phiên bản mới của lược đồ. Người dùng cũng có thể tạo một CSDL "khung nhìn" từ các phiên bản khác của lược đồ.

Như vậy, phiên bản là một trong những yêu cầu thiết yếu của mô hình dữ liệu trong các ứng dụng CSDL hướng đối tượng. Sau khi khởi tạo một đối tượng, các phiên bản mới sẽ được tạo lập từ đối tượng gốc và các phiên bản mới lần lượt là nguồn gốc cho các phiên bản mới khác. Như vậy, ta sẽ có hai mối quan hệ giữa các phiên bản. Thứ nhất, *mối quan hệ dẫn xuất* giữa phiên bản mới và phiên bản cũ của đối tượng (phiên bản mới được suy dẫn từ phiên bản cũ). Thứ hai, là *quan hệ phiên bản* giữa mỗi phiên bản của đối tượng và một đối tượng trừu tượng khác mô tả đối tượng đó.

Tổng quát, phiên bản của một đối tượng có dạng là đồ thị định hướng. Số các phiên bản mới sẽ được dẫn xuất từ một phiên bản bất kỳ vào thời điểm bất kỳ và một phiên bản có thể được dẫn xuất từ nhiều hơn một phiên bản cũ. Tuy nhiên, phần lớn các mô hình đã đề xuất đều giới hạn đồ thị định hướng trong một sự phân cấp, gọi là *phân cấp phiên bản*. Đồ thị phiên bản trong hình 2.8 biểu diễn sự tiến hóa của một phiên bản đối tượng.



Hình 2.9. Các mối quan hệ phiên bản

Các mô hình dữ liệu đều sử dụng một hệ thống ký hiệu để mô tả cho dữ liệu trong một CSDL và là hệ thống ký hiệu nền tảng cho ngôn ngữ thao tác dữ liệu. Trong CSDL hướng đối tượng, giao diện lập trình bao gồm ngôn ngữ con định nghĩa dữ liệu, ngôn ngữ con thao tác dữ liệu và ngôn ngữ con điều khiển dữ liệu. Các ngôn ngữ con này được thiết kế để có thể phản ánh đầy đủ sự mềm dẻo và các ràng buộc vốn có trong mô hình dữ liệu hướng đối tượng. Hơn nữa, để tích hợp thống nhất ngôn ngữ lập trình hướng đối tượng và ngôn ngữ CSDL cần thiết phải thiết kế các ngôn ngữ con phù hợp với các dạng thức truy nhập lớp và các phương thức thông qua các thông điệp của hệ thống hướng đối tượng.

2.4. GIAO DIỆN CƠ SỞ

Khi thiết kế hệ thống CSDL hướng đối tượng, người ta sẽ chấp nhận một trong hai hướng tiếp cận để thiết kế các giao diện người dùng cho hệ thống. Đó là hướng tiếp cận CSDL truyền thống, định

nghĩa một ngôn ngữ CSDL được nhúng vào ngôn ngữ lập trình chủ. Hướng tiếp cận thứ hai là mở rộng các ngôn ngữ lập trình hướng đối tượng có các kiến trúc liên quan đến CSDL. Thuận lợi của hướng tiếp cận thứ hai là người lập trình chỉ cần học các kiến trúc mới của cùng một ngôn ngữ, hơn là phải học một ngôn ngữ hoàn toàn mới.

Phần này giới thiệu một tập cơ sở của các ngôn ngữ con được mô tả dưới dạng các thông điệp truy nhập qua giao diện của đối tượng.

2.4.1. Truyền thông điệp

Thông điệp được gửi đến một đối tượng để truy nhập vào giá trị của thuộc tính và các phương thức được đóng gói trong đối tượng. Việc truy nhập đối tượng phải thông qua giao diện chung đã đặc tả cho đối tượng. Cú pháp của thông điệp gửi đến đối tượng như sau:

(**selector** <đối_tượng> [Ts1 Ts2 Ts3 ...])

Trong đó, **selector** là tên của thông điệp, <đối_tượng> là đối tượng nhận thông điệp. Các tham số tùy chọn Ts1, Ts2,... là các đối tượng và có thể là một thông điệp. Thông điệp trả về một đối tượng. Do đó, <đối_tượng> của một thông điệp cũng có thể là kết quả của một số các thông điệp khác.

2.4.2. Ngôn ngữ định nghĩa dữ liệu

DDL mô tả lược đồ CSDL hướng đối tượng, ngữ nghĩa mà DDL hỗ trợ là các khái niệm hướng đối tượng hạt nhân đã được giới thiệu trong mục 2.1.

Thông điệp **make-class** định nghĩa một lớp mới:

(**make-class** <Tên_lớp> [**:superclasses** *<Danh sách các siêu lớp>*]
[**:attributes** *<Danh sách các thuộc tính>*]
[**:methods** *<Danh sách các đặc tả phương thức>*])

Với <Tên_lớp> là tên của lớp mới. Mỗi từ khóa sau <Tên_lớp> là tùy chọn. <Danh sách các siêu lớp> kết hợp với từ khóa **:superclasses** là danh sách các siêu lớp của lớp mới. <Danh sách các thuộc tính> kết

hợp với từ khóa **:attributes** là danh sách các thuộc tính đặc tả mới bổ sung trong lớp. Sự đặc tả thuộc tính là danh sách chứa tên thuộc tính và từ khóa cùng với tập giá trị, được viết như sau:

(<Tên_thuộc_tính> **[:domain** <Đặc tả miền>]
[:inherit-from <Siêu lớp>])

Với <Đặc tả miền> mô tả kiểu của một thuộc tính. Từ khóa **:inherit-from** được sử dụng để chỉ rõ sự kế thừa. Nếu không có từ khóa này, thì thuộc tính được mô tả là thuộc tính mới của lớp. <Siêu lớp> là tên của siêu lớp mà thuộc tính được kế thừa; nếu không có tham số <Siêu lớp>, thì thuộc tính được kế thừa từ siêu lớp đầu tiên trong <Danh sách các siêu lớp>. <Danh sách các đặc tả phương thức> cùng với từ khóa **:methods** là một danh sách các cặp (<Tên_phương_thức> <Siêu lớp>). <Tên_phương_thức> là tên của một phương thức được kế thừa từ <Siêu lớp>. Nếu không mô tả <Siêu lớp> thì phương thức mới được định nghĩa đối với lớp chứa nó.

Ví dụ sau biểu diễn việc định nghĩa lớp:

```
(make-class     SinhVien
                :superclasses   ( Nguoi )
                :attributes     (( dtb: domain   double)
                                (sv_khoa: inherit-from  Khoa ))
                :methods        (Tinh_hoc_bong   double))
```

2.4.3. Ngôn ngữ thao tác dữ liệu

DML cho phép khởi tạo, cập nhật và xóa các thể hiện riêng lẽ của một lớp bằng cách gửi một thông điệp thích hợp đến lớp hoặc các thể hiện.

Cú pháp của phép thao tác đối tượng, tức là phép tạo lập, truy vấn, xóa và cập nhật tương tự với các thao tác đối với CSDL quan hệ, dù các biểu thức truy vấn có sự khác biệt đáng kể so với CSDL quan hệ.

Thông điệp **make** tạo lập thể hiện của lớp:

(**make** <Tên_lớp> :<Thuộc tính 1> giá trị 1

...

:<Thuộc tính N> giá trị N)

Thông điệp **select** chọn các thể hiện của một lớp thỏa mãn biểu thức điều kiện cho trước. Kết quả của thông điệp là một tập đối tượng.

(**select** <Tên_lớp> <Biểu thức logic>)

Thông điệp **delete** xóa các thể hiện của một lớp thỏa mãn một biểu thức điều kiện:

(**delete** <Tên_lớp> <Biểu thức logic>)

Thông điệp **delete-object** xóa một đối tượng:

(**delete-object** Object), với Object là định danh đối tượng.

Thông điệp **change** thay đổi giá trị một thuộc tính cho các thể hiện của một lớp thỏa mãn biểu thức điều kiện.

(**change** <Tên_lớp> [<Biểu thức logic>] <Tên_thuộc_tính> <Giá trị mới>)

2.4.4. Ngôn ngữ điều khiển dữ liệu

DCL phải cho phép đặc tả các giao dịch, điều khiển tính toàn vẹn ngữ nghĩa, cấp quyền và quản lý truy nhập các phương thức.

KẾT LUẬN

Những khái niệm về đối tượng, lớp, kiểu, sự kế thừa, phương thức và sự đóng gói, v.v... đã được trình bày trong phần đầu của chương, là những kiến thức cơ sở cần thiết cho việc mở rộng nghiên cứu trong các chương tiếp theo. Qua đó, chúng ta có thể nhận thấy được sự phát triển có tính kế thừa từ các mô hình CSDL thế hệ trước đến mô hình CSDL hướng đối tượng và xác định hướng tiếp cận các vấn đề nghiên cứu dựa trên những đặc trưng mới xuất phát từ sự mở rộng và kế thừa giữa các mô hình dữ liệu. Những khái niệm cơ sở trong chương 2 được thể hiện và cài đặt khá đầy đủ trong mô hình dữ liệu hướng đối tượng ODMG sẽ giới thiệu trong chương 3.

CÂU HỎI ÔN TẬP CHƯƠNG 2

Câu 2.1. Với đặc trưng "đóng gói" của CSDL hướng đối tượng. Anh (chị) hãy nêu ý nghĩa của đặc trưng này khi thiết kế CSDL so với các kỹ thuật thiết kế của các hệ CSDL trước đó.

Câu 2.2. Sử dụng ký pháp của *s-đồ thị*, hãy biểu diễn các đối tượng trong các hình 1.6 đến hình 1.10 và kiểm tra tính mâu thuẫn của các *s-đồ thị*.

Câu 2.3. Để mô hình hóa cho các đối tượng trong CSDL quản lý biển số xe ôtô. Hãy xây dựng lược đồ đối tượng và tập các phương thức tương ứng để quản lý và cấp phát biển số xe ôtô.

Câu 2.4. Sử dụng các lệnh trong ngôn ngữ định nghĩa dữ liệu để tạo các đối tượng trong CSDL quản lý biển số xe ôtô.

Câu 2.5. Cho CSDL quản lý biển số xe ôtô. Sử dụng các lệnh trong ngôn ngữ thao tác dữ liệu, thực hiện:

a) Tạo lập các thể hiện lớp của các đối tượng;

b) Tạo lập các thể hiện lớp của các xe có đăng ký tại thành phố Đà Nẵng và dung tích xylanh từ 2.0 trở lên;

c) Xóa các thể hiện lớp của các xe có năm sản xuất là năm 1970;

d) Thay đổi giá trị tính thuế trước bạ của xe thành "<giá xe>*5%>".

CHƯƠNG

3

PHƯƠNG PHÁP CHUYỂN ĐỔI DỮ LIỆU GIỮA CƠ SỞ DỮ LIỆU QUAN HỆ VÀ HƯỚNG ĐỐI TƯỢNG

Khi mô hình hướng đối tượng trở thành xu hướng của kỹ thuật xây dựng CSDL, thì việc chuyển đổi các hệ thống CSDL quan hệ sang CSDL hướng đối tượng là điều cần thiết để cải tiến tính hiệu quả và mềm dẻo của hệ thống. Sự chuyển đổi hệ thống này bao gồm quá trình biên dịch lược đồ, chuyển đổi dữ liệu và chương trình.

3

Trong nội dung của chương, chúng ta sẽ xem xét một phương pháp tích hợp việc biên dịch lược đồ và chuyển đổi dữ liệu của hệ thống. Biên dịch lược đồ bao hàm ngữ nghĩa cấu trúc lại và ánh xạ lược đồ quan hệ vào lược đồ hướng đối tượng. Chuyển đổi dữ liệu bao hàm việc "tháo" và chuyển các bộ trong quan hệ vào các file có cấu trúc tuần tự và tái nạp chúng vào các file chứa các lớp hướng đối tượng. Phương pháp này bảo đảm các ràng buộc của CSDL quan hệ bằng ánh xạ các phụ thuộc dữ liệu tương đương.

3.1. PHƯƠNG PHÁP CHUYỂN ĐỔI DỮ LIỆU TỪ CƠ SỞ DỮ LIỆU QUAN HỆ ĐÃ TỒN TẠI SANG CƠ SỞ DỮ LIỆU HƯỚNG ĐỐI TƯỢNG

Để chuyển đổi dữ liệu từ mô hình quan hệ sang mô hình hướng đối tượng, điều kiện tiên quyết là biên dịch lược đồ quan hệ sang lược đồ hướng đối tượng. Về mặt phương pháp luận, nó cung cấp sự tiếp cận có tính hệ thống để giải quyết các vấn đề khác nhau trong lược đồ và chuyển đổi dữ liệu, tính lý thuyết đầy đủ và hoàn chỉnh trong sự đảm bảo ngữ nghĩa và các ràng buộc của CSDL quan.

Chúng ta sẽ giới thiệu một hướng tiếp cận đối với việc trích xuất các cấu trúc của lược đồ hướng đối tượng từ lược đồ quan hệ đã tồn tại. Phần đầu là quá trình biên dịch lược đồ quan hệ sang lược đồ hướng đối tượng. Phần cuối cùng trình bày một phương pháp chuyển đổi dữ liệu của CSDL quan hệ sang CSDL hướng đối tượng. Các quan hệ được sử dụng là các quan hệ ít nhất ở dạng chuẩn 2 (không ở dạng chuẩn 3).

3.1.1. Tiến trình tổng quát

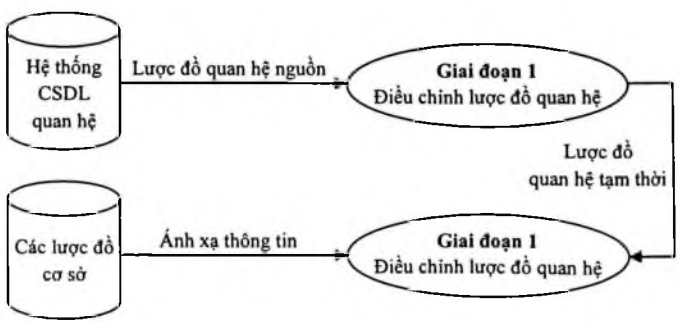
Một ánh xạ lược đồ của một lược đồ nguồn S_1^{M1} xác định trên mô hình dữ liệu M_1 đến một lược đồ đích S_2^{M2} xác định trên mô hình M_2 có thể được xác định như là một phép biến đổi T sao cho $T(S_1^{M1}) = S_2^{M2}$. Phép biến đổi này thường gồm một tập các luật cho phép chuyển đổi từ mô hình dữ liệu nguồn sang mô hình dữ liệu đích.

Nhiệm vụ trước tiên của ánh xạ từ một lược đồ quan hệ sang lược đồ hướng đối tượng là nhận dạng các cấu trúc đối tượng trích xuất từ lược đồ quan hệ. Các cấu trúc được quan tâm trong mô hình hướng đối tượng là:

- *Các lớp đối tượng*: Các quan hệ tương ứng với các lớp đối tượng phải được nhận dạng.

- *Các mối quan hệ*: Có ba kiểu quan hệ có thể được biểu diễn lại trong mô hình đối tượng. Chúng gồm tính kết hợp, tính tổng quát hóa đặc trưng và tính tập hợp.

Tiến trình ánh xạ lược đồ tĩnh gồm một tiến trình chứa hai giai đoạn: Trong giai đoạn đầu tiên, lược đồ quan hệ được điều chỉnh và biến đổi vào lược đồ quan hệ tạm thời với các thuộc tính đặc trưng nào đó. Trong giai đoạn hai, các cấu trúc hướng đối tượng sẽ được trích xuất từ các lược đồ quan hệ tạm thời này.



Hình 3.1. Hai giai đoạn ánh xạ lược đồ tĩnh

3.1.2. Điều chỉnh lược đồ quan hệ

Có bốn bước thực hiện điều chỉnh lược đồ trong giai đoạn này như sau:

- *Bước 1*: Loại bỏ các quan hệ 2NF và thay chúng bằng các quan hệ tạm thời 3NF;
- *Bước 2*: Khởi tạo các quan hệ lớp con tạm thời đối với các quan hệ siêu lớp không có quan hệ lớp con;
- *Bước 3*: Khởi tạo các quan hệ siêu lớp đối với các quan hệ lớp con đơn (các quan hệ lớp con chưa có liên kết với quan hệ siêu lớp);
- *Bước 4*: Loại bỏ các thuộc tính đa trị và thay chúng bằng các quan hệ 3NF.

Chú ý rằng, trong quá trình thực hiện điều chỉnh lược đồ quan hệ, lược đồ gốc không được thay đổi, bởi vì sự thay đổi lược đồ nguồn sẽ dẫn đến việc thay đổi dữ liệu đã tồn tại và các ứng dụng không thích hợp với người sử dụng. Vì vậy, trước khi hiệu chỉnh, phải tạo lập một lược đồ tạm thời với các thuộc tính có ý nghĩa như lược đồ nguồn.

Thuật toán tại bước 1:

Thuật toán 3.1. Loại bỏ các quan hệ 2NF.

Vào/ Ra: Lược đồ quan hệ tạm thời $\Re$ (được tạo lập từ lược đồ nguồn), trên tập các thuộc tính U.

Phương pháp:

(1) **for** mỗi quan hệ R trong lược đồ quan hệ $\Re$ **do**

(2) **for** mỗi cặp phụ thuộc hàm $X \rightarrow Y$, và $Y \rightarrow Z$, mà Y không phải là khóa dự tuyển của R **do**

(3) - Tạo một quan hệ mới tạm thời R_1 có các thuộc tính $U_{R1} = U_R - Z$, $PK_{R1} = PK_R$.

(4) - Tạo một quan hệ mới tạm thời R_2 có các thuộc tính $U_{R2} = Y \cup Z$, $PK_{R2} = Y$.

(5) - Lấy $R_1.Y$ là khóa liên kết tham chiếu đến R_2

(6) - Lấy R_2.Y là khóa chính của của R_2.

(7) - Thay thế quan hệ R trong $\Re$ bằng hai quan hệ tạm thời R_1 và R_2.

Ví dụ 3.1: Xét quan hệ ở 2NF:

$R(\underline{A}, B, C, D, E)$; $A \rightarrow B$, $B \rightarrow CD$.

Quan hệ R có thể được tách thành hai lớp - quan hệ tạm thời

R_1 ($\underline{A}$, B^*, E) và R_2 ($\underline{B}$, C, D)

Trong đó, các thuộc tính gạch chân là khóa chính và các thuộc tính có dấu "*" là các thuộc tính liên kết.

Tất cả các quan hệ ở dạng chuẩn 3 (3NF) có đủ tiêu chuẩn để là lớp - quan hệ và vì vậy ta ánh xạ mỗi quan hệ sang một lớp đối tượng. (Ta định nghĩa *lớp - quan hệ* là một quan hệ mà ánh xạ trực tiếp đến lớp đối tượng).

Ở bước 2 thực hiện việc loại bỏ các *quan hệ "góa"* (widow relation), tương tự như các "*dòng văn bản góa*" (widow lines - dòng văn bản cuối cùng trong đoạn được in vào đầu trang tiếp theo, được sử dụng trong kỹ thuật dàn trang văn bản); quan hệ "góa" là quan hệ có các thuộc tính tương ứng với lớp con được nhúng trong một quan hệ đơn. Kết quả của bước 2 để trích xuất những cấu trúc nhúng từ các quan hệ và biểu hiện chúng một cách rõ ràng trong dạng thức của các quan hệ tạm thời và những mối quan hệ khóa liên kết. Các kiểu kiến trúc này biểu diễn một cách đặc trưng các mối quan hệ kế thừa mà những thuộc tính tương ứng đối với mỗi lớp con được nhúng trong một quan hệ đơn. Các thuộc tính chứa các giá trị *Null* không áp dụng được cho một lớp con cụ thể nào. Chương trình ứng dụng sử dụng một trong các thuộc tính của quan hệ như là một thuộc tính phân biệt theo quy ước để xác định các thuộc tính của một bộ cho trước phải được xử lý. Những người thiết kế CSDL quan hệ dùng kiểu này để tránh các phép toán kết nối khi những lớp con được biểu diễn bởi nhiều quan hệ tách rời nhau.

Ví dụ, chúng ta xét bảng sau:

DuAn(maduan#, tenduan, loaiduan, nhacungcap#, ngonngu, diadiem)

Trong bảng DuAn, thuộc tính nhacungcap# chỉ có liên quan với các dự án phần cứng (tức là: thuộc tính loaiduan = 'H') và các thuộc tính ngonngu, diadiem chỉ liên quan với các dự án phần mềm (tức là: thuộc tính loaiduan = 'S'). Bảng chứa các giá trị *Null* đối với thuộc tính nhacungcap# đối với tất cả dự án phần mềm và giá trị *Null* đối với thuộc tính ngonngu và diadiem cho các dự án phần cứng. Đây là một ví dụ của thể hiện hai lớp con được lưu trữ trong quan hệ đơn.

Trên quan hệ DuAn, ta có thể nhận dạng các luật như sau:

loaiduan = 'S' $\Rightarrow$ nhacungcap# = *Null*

loaiduan = 'H' $\Rightarrow$ (ngonngu = *Null* $\wedge$ diadiem = *Null*)

Như vậy, các luật được gọi một cách đặc trưng là "*luật mạnh*", vì chúng chứa tất cả các thể hiện của quan hệ. Thực ra, các giải thuật khai phá tri thức có thể nhận dạng thuộc tính phân biệt bằng cách tìm ra các luật mà vế phải có dạng A = *Null*, với A là tên một thuộc tính nào đó.

$Rule_i$: $(D = V_j) \Rightarrow X_i$ = *Null*, D là thuộc tính phân biệt và X_i là một thuộc tính hay tổ hợp các thuộc tính.

Giải thuật chính ở bước 2 chứa một vòng lặp duyệt kiểm tra mỗi thuộc tính, với chương trình con *MakeSubclasses* được sử dụng để tính toán các luật (kiểm tra thuộc tính phân biệt) và khởi tạo các quan hệ mới tạm thời tương ứng với các lớp con.

Để giảm thiểu các truy nhập CSDL quá nhiều khi kiểm tra các thuộc tính phân biệt, chúng ta sử dụng ba điều kiện để "làm ngắn" các tính toán trên CSDL tương ứng. Thứ nhất, nếu kiểu dữ liệu của thuộc tính là BLOB (**B**inary **L**arge **O**bject, đối với các thuộc tính đa trị), thì thuộc tính này không thể là một thuộc tính phân biệt. Tương tự, nếu thuộc tính có giá trị *Null* trong CSDL, nó cũng không thể là một thuộc tính phân biệt. Ngoại trừ, trong trường hợp xấu nhất (khi bộ cuối cùng

trong quan hệ chứa một thuộc tính có giá trị *Null*), hai điều kiện này thường được kiểm tra mà không truy nhập tất cả các bộ của quan hệ. Điều kiện thứ ba dùng biến tài nguyên gọi là *Unique_Threshold*, nó là biến chỉ báo việc chọn thuộc tính. Các thuộc tính phân biệt đặc trưng không chứa nhiều giá trị phân biệt (tương ứng với tổng số các bộ). Biến *Unique_Threshold* mô tả số các giá trị phân biệt lớn nhất cho phép đối với một thuộc tính phân biệt. Giá trị đặc trưng đối với *Unique_Threshold* là 5.

Nếu một thuộc tính thỏa mãn cả ba điều kiện, tức là thuộc tính đó không có kiểu dữ liệu BLOB, không chứa giá trị *Null* và không có số giá trị phân biệt vượt ngưỡng *Unique_Threshold*, nó là thuộc tính phân biệt, dựa vào giá trị của thuộc tính này chúng ta sẽ tính một ma trận Boolean với các phần tử như sau:

$$A = (m_{ij}),\ m_{ij} = \begin{cases} \text{True} & \text{nếu } \prod_{Aj}(\sigma_{A=unique\_values[i]}(R)) = Null \\ \text{False} & \text{các trường hợp còn lại.} \end{cases}$$

Với *unique_values*[] là mảng chứa các giá trị phân biệt của thuộc tính A (thuộc tính phân biệt) trong quan hệ R. Ví dụ, chúng ta xét quan hệ DuAn sau:

maduan#	tenduan	loaiduan	nhacungcap#	ngonngu	diadiem
0001	Dự án 1	"H"	2102	*Null*	*Null*
0002	Dự án 2	"H"	2102	*Null*	*Null*
0003	Dự án 3	"H"	2107	*Null*	*Null*
0004	Dự án 4	"H"	3102	*Null*	*Null*
0005	Dự án 5	"H"	2102	*Null*	*Null*
0006	Dự án 6	"S"	*Null*	C++	"A"
0007	Dự án 7	"S"	*Null*	Java	"B"
0008	Dự án 8	"S"	*Null*	Java	*Null*
0009	Dự án 9	"S"	*Null*	Smalltalk	"C"
0010	Dự án 10	"S"	*Null*	Lisp	"D"

Với quan hệ trên, ma trận tương ứng với thuộc tính loaiduan như sau:

	maduan	tenduan	loaiduan	nhacungcap#	ngonngu	diadiem
loaiduan = "H"	F	F	F	F	T	T
loaiduan = "S"	F	F	F	T	F	F

Thuật toán tại bước 2:

Thuật toán 3.2. Loại bỏ các quan hệ "góa".

Vào: *Unique_Threshold.*

Ra: Lược đồ quan hệ tạm thời $\Re$, tập thuộc tính U.

Phương pháp:

(1) **for** mỗi quan hệ R trong $\Re$ **do**

(2) **for** mỗi thuộc tính $A_i \in U$ **do**

(3) **call** *MakeSubclasses*(R, A_i, $\Re$, *Unique_Threshold*)

Thủ tục *MakeSubclasses*:

Vào: Quan hệ R, thuộc tính A, *Unique_Threshold*

Ra: Lược đồ quan hệ tạm thời $\Re$, tập thuộc tính U.

Phương pháp:

(1) **if** data_type(A) = BLOB **or** $|\sigma_{A=Null}(R)| > 0$ **or** |unique_values(A)| > *Unique_Threshold*

(2) **then return**

(3) Tính unique_values[], ma trận các giá trị phân biệt của A.

(4) Tính ma trận Boolean $M = (m_{ij})$, với $i = 1,2,...,|unique\_values|$, $j = 1,2,...,|U|$

(5) **if** ma trận M không chứa ít nhất một cột có giá trị False **then return**

(6) Lấy $U_{super} = \cup_p A_p$, với chỉ số *p* được chọn sao cho (m_{ip} = False, $\forall i$) **or** (m_{ip} = True, $\forall i$)

(7) Khởi tạo một quan hệ tạm thời S tương ứng với siêu lớp - quan hệ có các thuộc tính U_{super}

(8) Thay thế quan hệ R trong $\Re$ bằng S

(9) **for** mỗi giá trị phân biệt v_i của thuộc tính A **do**

(10 Lấy $U_{new} = \cup_p A_p$, với p được chọn sao cho (m_{ip} = FALSE $\wedge$ $A_p \notin U_{super}$)

(11) Tạo quan hệ tạm thời V có các thuộc tính $U_{new} \cup PK_S$ (PK_S là khóa chính của quan hệ S)

(12) Lấy khóa chính của V là một khóa liên kết tham chiếu đến S

(13) Bổ sung V vào $\Re$.

Bước 3 liên quan với việc loại bỏ các quan hệ "mồ côi" (*orphan relation*, thuật ngữ này tương tự như "dòng mồ côi" trong kỹ thuật soạn thảo, đó là dòng đầu tiên của đoạn ở cuối trang phía trên), tức là các quan hệ lưu trữ các bộ một cách độc lập (không liên kết với các quan hệ khác). Kết quả của bước 3 là khởi tạo các quan hệ tạm thời (quan hệ "bố mẹ") tương ứng với siêu lớp chứa các thuộc tính chung giữa nhóm các quan hệ. Ví dụ, quan hệ DuAn có thể được cài đặt thành hai quan hệ DuAn_Phancung và DuAn_Phanmem như sau:

DuAn_Phancung

maduan#	tenduan	nhacungcap#
0001	Dự án 1	2102
0002	Dự án 2	2102
0003	Dự án 3	2107
0004	Dự án 4	3102
0005	Dự án 5	2102

DuAn_Phanmem

maduan#	tenduan	ngonngu	diadiem
0006	Dự án 6	C++	"A"
0007	Dự án 7	Java	"B"
0008	Dự án 8	Java	*Null*
0009	Dự án 9	Smalltalk	"C"
0010	Dự án 10	Lisp	"D"

Kết quả của bước 3 đối với hai quan hệ trên là một quan hệ mới có các thuộc tính: Quanhe_Tamthoi(maduan#, tenduan). Hai quan hệ được xem là có cùng siêu lớp nếu chúng có cùng khóa chính và chúng phải có thêm một thuộc tính chung khác khóa chính. Tiến trình của giải thuật sẽ thực hiện lặp đi lặp lại đối với các cặp quan hệ. Tập G sẽ chứa các cặp lớp con - siêu lớp thu được khi thực hiện giải thuật, sau đó loại bỏ các phụ thuộc bắc cầu trong G. Đối với mỗi cặp trong G, khóa liên kết tham chiếu đến quan hệ tương ứng với siêu lớp được bổ sung vào quan hệ tương ứng với lớp con và các thuộc tính chung giữa hai quan hệ được loại bỏ đối với quan hệ tương ứng với lớp con.

Ví dụ minh họa giải thuật bước 3, xét ba quan hệ giả thiết:

R_1 (A, B, C, D, E)

R_2 (A, B, C, F, G)

R_3 (A, B, H, I)

Kết quả áp dụng bước tạo lập các quan hệ "cha mẹ" tạm thời là:

T_1(A, B, C)

T_2(A, B)

G = { <R_1, T_1>, <R_2, T_1>, <R_1, T_2>,
<R_2, T_2>, <R_3, T_2>, <T_1, T_2>}

Cuối cùng, loại bỏ các cặp phụ thuộc bắc cầu bằng cách xóa <R_1,T_2>, <R_2,T_2> từ G. Vì vậy, phân cấp tổng quát cuối cùng sau khi áp dụng tiến trình này ta có:

G = { <R_1, T_1>, <R_2, T_1>, <R_3, T_2>, <T_1, T_2>}

Thuật toán tại bước 3:

Thuật toán 3.3. Loại bỏ các quan hệ "mồ côi".

Vào/ Ra: Lược đồ quan hệ tạm thời ℜ.

Phương pháp:

(1) Lấy G là tập chứa các cặp có thứ tự <T_1, T_2>, T_1 là lớp con của lớp T_2, và khóa chính của T_1 là khóa liên kết tham chiếu đến T_2.

(2) **for** với mỗi quan hệ R_i trong $\Re$ **do**

(3) **for** với mỗi quan hệ R_j trong $\Re$, $i \neq j$ **do**

(4) **if** $PK_{Ri} = PK_{Rj}$ **and** $PK_{Ri} \subset U_{Ri} \cap U_{Rj}$ **then**

(5) Đặt $U_{new} = U_{Ri} \cap U_{Rj}$

(6) Tạo quan hệ tạm thời V có các thuộc tính trong U_{new}

(7) Xóa U_{new} - PK_{Ri} trong U_{Ri}

(8) Xóa U_{new} - PK_{Rj} trong U_{Rj}

(9) Bổ sung V vào $\Re$

(10) Bổ sung các cặp $<R_i, V>$ và $<R_j, V>$ vào G.

(11) **if** $\exists$ T sao cho $<T_1, T> \in G \wedge <T, T_2> \in G$ **then**

(12) Xóa $<T_1, T_2>$ trong G

(13) **for** với mỗi cặp $<T_1, T_2> \in G$ **do**

(14) Tạo khóa chính của T_1 là khóa tham chiếu đến T_2

Đối với bước 4 chúng ta thực hiện thao tác thay thế các thuộc tính đa trị bởi các quan hệ 3NF.

Thuật toán tại bước 4:

Thuật toán 3.4. Loại bỏ các thuộc tính có kiểu BLOB.

Vào/ Ra: Lược đồ quan hệ tạm thời $\Re$.

Phương pháp:

(1) **for** mỗi quan hệ R trong $\Re$ **do**

(2) $U_{aggr} = U_R$

(3) **for** với $A_i \in U_R$ **do**

(4) **if** data_type(A_i) = BLOB **and** A_i chứa dữ liệu phức **then**

(5) U_{new} = các thuộc tính của dữ liệu phức tương ứng với A_i

(6) Tạo một quan hệ tạm thời V có các thuộc tính $U_{new} \cup PK_R$

(7) Bổ sung V một khóa liên kết tham chiếu đến R

(8) Bổ sung V vào $\Re$.

(9) Xóa thuộc tính $R.A_i$ khỏi U_{aggr}

(10) Tạo quan hệ S có các thuộc tính trong U_{aggr}

(11) Thay quan hệ R trong $\Re$ bằng quan hệ S

3.1.3. Chuyển đổi lược đồ quan hệ sang lược đồ hướng đối tượng

Các tiến trình trong chuyển đổi lược đồ bao gồm:

- *Nhận dạng các lớp đối tượng*: Các quan hệ tương ứng với lớp đối tượng được nhận dạng tại bước này.

- *Nhận dạng các mối quan hệ*: Có ba kiểu quan hệ có thể được biểu diễn trong mô hình đối tượng. Gồm sự liên kết, sự tổng quát hóa/đặc tả hóa và sự kết nhập:

+ Nhận dạng sự liên kết: Mô hình đối tượng cho phép xác định sự liên kết như là các lớp, ta phải kiến lập một sự liên kết đơn giản giữa hai lớp đối tượng.

+ Nhận dạng sự kế thừa: Các cấu trúc kế thừa thu được mối quan hệ tổng quát hóa và đặc tả hóa giữa các lớp đối tượng đã được nhận dạng.

+ Nhận dạng sự kết nhập: Mô hình mối quan hệ kết nhập giữa một đối tượng và đối tượng khác. Như vậy, các đối tượng phức phải được nhận dạng. Sự khác biệt giữa tính kết nhập và tính liên kết là sự tồn tại phụ thuộc hoàn toàn của một đối tượng con vào một đối tượng nào đó. Chẳng hạn, đối tượng mô tả động cơ xe (DongCo) có quan hệ "*bộ phận*" với đối tượng mô tả xe hơi (XeHoi), do đó không thể tồn tại đối tượng DongCo nếu XeHoi không tồn tại. Mặt khác, ví dụ, sự ghi danh học của một sinh viên trong một khóa học là một sự liên kết đúng hơn là sự kết nhập bởi vì các đối tượng SinhVien và KhoaHoc có thể tồn tại một cách độc lập với nhau.

3.1.3.1. Nhận dạng lớp đối tượng

Tất cả các quan hệ ở 3NF đủ tiêu chuẩn để trở thành các lớp - quan hệ và vì vậy mỗi quan hệ đó được chuyển đổi thành một lớp đối

tượng. Các lớp tương ứng với các quan hệ có khóa chính bao hàm những khóa liên kết biểu diễn các liên kết giữa chính các lớp đó. Các lớp như vậy gọi là các lớp - liên kết (*association - classes*).

Thuật toán 3.5. Nhận dạng các lớp.

Vào: Lược đồ quan hệ tạm thời $\Re$

Ra: Các lớp đối tượng **C** và lớp liên kết **S**.

Phương pháp:

(1) **for** với mỗi quan hệ $R \in \Re$ **do**

(2) Tạo một lớp C có cùng các thuộc tính như R

(3) Bổ sung C vào **C**

(4) **if** $|PK_R|>1$ **and** $PK_R \subseteq FK_R$ **then**

(5) Bổ sung C vào **S**

Trong đó, PK_R và FK_R là khóa chính và tập khóa liên kết của R.

3.1.3.2. Nhận dạng các liên kết

Liên kết giữa các lớp đối tượng có thể được nhận dạng dựa vào hai trường hợp. Thứ nhất, mỗi quan hệ có khóa chính bao gồm các khóa liên kết là lớp mô tả một liên kết. Mối quan hệ liên kết này là giữa tất cả các lớp tương ứng với lớp - quan hệ mà các khóa liên kết tham chiếu đến. Thứ hai, đối với mỗi lớp - quan hệ không có liên kết lớp, thì thiết lập một liên kết quan hệ tương ứng giữa lớp và lớp ứng với mỗi khóa liên kết không thuộc khóa.

Thuật toán 3.6. Nhận dạng các liên kết.

Vào: Lược đồ quan hệ tạm thời $\Re$

Ra: Các lớp đối tượng **C** và lớp liên kết **S**.

Phương pháp:

(1) **for** mỗi quan hệ $R \in \Re$ **do**

(2) Lấy C là lớp tương ứng với quan hệ R

(3) **for** với mỗi khóa liên kết $FK \in FK_R$ **do**

(4) Đặt T là lớp tương ứng với quan hệ mà FK tham chiếu đến

(5) Tạo liên kết nhiều - một giữa C và T

(6) **for** với mỗi lớp liên kết C ∈ **C do**

(7) Lấy quan hệ R trong $\Re$ tương ứng với lớp C

(8) Đặt *F* là tập các khóa liên kết mà khóa chính của R tham chiếu đến

(9) **for** mỗi thuộc tính $F_i \in F$ **do**

(10) **for** mỗi $F_j \in F$, $i \neq j$ **do**

(11) Lấy C_1 là lớp tương ứng với quan hệ mà F_i tham chiếu đến

(12) Lấy C_2 là lớp tương ứng với quan hệ mà F_j tham chiếu đến

(13) Khởi tạo liên kết nhiều - nhiều giữa C_1 và C_2 với C đóng vai trò là lớp liên kết.

3.1.3.3. Nhận dạng sự kế thừa

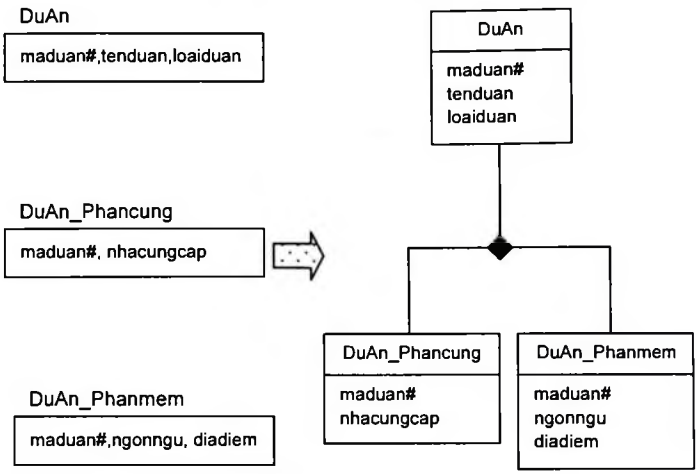
Ba dạng cấu trúc quan hệ có thể được biểu thị bằng mối quan hệ kế thừa. Các trường hợp sau này sinh trong quá trình điều chỉnh lược đồ quan hệ và ví dụ minh họa đã trình bày ở trên.

a) Trường hợp 1

Lược đồ quan hệ bao hàm một quan hệ đối với siêu lớp và một quan hệ đối với tất cả các lớp con. Mỗi cặp lớp - quan hệ (CR_1, CR_2) có cùng khóa chính có thể được chứa trong một mối quan hệ kế thừa. Mối quan hệ kế thừa CR_1 ISA CR_2 tồn tại nếu khóa chính của CR_1 cũng là khóa liên kết và tham chiếu đến lớp - quan hệ CR_2.

Đây là trường hợp đơn giản nhất trong ba trường hợp và được minh họa bằng ví dụ trong hình 3.2. Trong ví dụ này, quan hệ DuAn và quan hệ DuAn_Phancung có maduan# là khóa và DuAn_Phancung.maduan# tham chiếu đến DuAn.maduan#.

Vì vậy, DuAn_Phancung là một lớp con của DuAn. Tương tự, DuAn_Phanmem là một lớp con của DuAn.



Hình 3.2. Ví dụ cho trường hợp 1

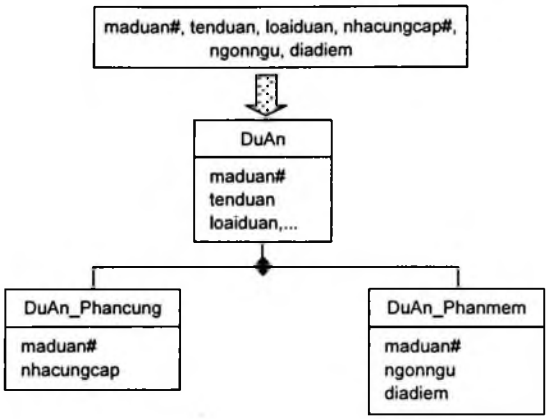
b) Trường hợp 2

Ở một số trường hợp, mối quan hệ kế thừa có thể được nhúng trong một lớp quan hệ đơn. Chẳng hạn, chúng ta xét bảng sau:

DuAn(maduan#, tenduan, loaiduan, nhacungcap#*, ngonngu, diadiem)

Trong bảng DuAn, thuộc tính nhacungcap# chỉ có liên quan với các dự án phần cứng (tức là: thuộc tính loaiduan = 'H') và các thuộc tính ngonngu, diadiem chỉ liên quan với các dự án phần mềm (tức là: thuộc tính loaiduan = 'S'). Bảng chứa các giá trị *null* đối với thuộc tính nhacungcap# đối với tất cả các dự án phần mềm và giá trị *null* đối với các thuộc tính ngonngu, diadiem cho các dự án phần cứng. Đây là một ví dụ của thể hiện hai lớp con được lưu trữ trong quan hệ đơn.

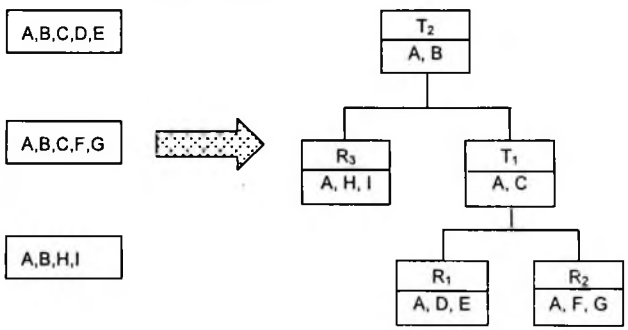
Quan hệ DuAn có thể chuyển đổi thành hai lớp - quan hệ tạm thời và hai mối quan hệ tổng quát như trong hình 3.3.



Hình 3.3. Ví dụ cho trường hợp 2

c) Trường hợp 3

Trong trường hợp thứ ba, mỗi lớp con được mô tả bằng một quan hệ đơn không có quan hệ tương ứng với siêu lớp. Các thuộc tính của siêu lớp được lập lại trong lược đồ của mỗi quan hệ tương ứng với các lớp con. Ví dụ minh họa cho ở hình 3.4.



Hình 3.4. Ví dụ của trường hợp 3

Thuật toán 3.7. Nhận dạng sự kế thừa.

Vào: Lược đồ quan hệ tạm thời $\Re$.

Ra: Các lớp đối tượng **C**.

Phương pháp:

(1) **for** mỗi quan hệ $R \in \Re$ **do**

(2) **if** $PK_R \in FK_R$ **then**

(3) Lấy $FK \in FK_R$ là khóa liên kết tương ứng với khóa chính PK_R

(4) Lấy C_1 là lớp tương ứng với quan hệ R.

(5) Lấy C_2 là lớp tương ứng với quan hệ mà FK tham chiếu đến.

(6) Tạo lập lớp C_1 là lớp con của lớp C_2.

3.1.3.4. Nhận dạng sự kết nhập

Mỗi quan hệ kết nhập biểu thị sự phụ thuộc tồn tại. Tức là, nếu A là "is-part-of" B, thì sự tồn tại của A phụ thuộc vào sự tồn tại của B. Xét một quan hệ - lớp CR_1 có khóa chính với nhiều hơn một thuộc tính và ít nhất có một thuộc tính trong chúng không phải là khóa liên kết. Lấy CR_2 là quan hệ - lớp tương ứng với tập con lớn nhất của các khóa liên kết trong khóa chính của CR_1. Nếu như tồn tại một quan hệ - lớp, thì CR_1 "is-part-of" CR_2.

Thuật toán 3.8. Nhận dạng sự kết nhập.

Vào: Lược đồ quan hệ tạm thời $\Re$.

Ra: Các lớp đối tượng **C**.

Phương pháp:

(1) **for** mỗi quan hệ $R \in \Re$ **do**

(2) **if** $|PK_R|>1 \wedge PK_R \subset FK_R$ **then**

(3) Lấy C_1 là lớp tương ứng với quan hệ R

(4) **if** $\exists FK \in FK_R$ sao cho $(PK_R \cap FK_R) = FK$ **then**

(5) Lấy C_2 lớp tương ứng với quan hệ mà FK tham chiếu đến

(6) Khởi tạo C_1 "is-part-of" C_2

3.1.4. Phương pháp chuyển và nạp dữ liệu từ cơ sở dữ liệu quan hệ sang cơ sở dữ liệu hướng đối tượng

Trong quá trình chuyển đổi dữ liệu từ CSDL quan hệ sang hướng đối tượng, quá trình của chúng ta có ba bước bao gồm biên dịch lược đồ, chuyển và nạp lại dữ liệu từ CSDL quan hệ sang CSDL hướng đối tượng. Như vậy, trong phần này chúng ta trình bày việc chuyển và tái nạp dữ liệu từ CSDL quan hệ sang CSDL hướng đối tượng.

3.1.4.1. Chuyển các bộ của quan hệ vào file có cấu trúc tuần tự

Tương ứng với việc biên dịch lược đồ hướng đối tượng, các bộ của mỗi quan hệ sẽ được chuyển vào một file tuần tự. Tiến trình chuyển sẽ được chia làm 3 bước sau:

- *Bước thứ nhất*, để chuyển mỗi bộ của quan hệ vào file với các lệnh chèn (Insert) (chú ý: các lệnh này sẽ được sử dụng để tái nạp dữ liệu vào OODB sao cho mỗi lớp sẽ được khởi tạo nạp dữ liệu từ các bộ của quan hệ tương ứng).

- *Bước thứ hai*, đối với mỗi khóa liên kết, nó được dựa vào bộ của quan hệ cha sẽ được tái nạp và file khác với câu lệnh cập nhật (Update). Ý tưởng thực hiện là dùng câu lệnh chèn (Insert) để lưu trữ OID khi tái nạp dữ liệu ở bước thứ nhất. Câu lệnh cập nhật (Update) là để thay thế thuộc tính liên kết khi chúng được tái nạp đến CSDL hướng đối tượng đích.

- *Bước thứ ba*, đối với mỗi quan hệ lớp con, nó dựa vào siêu lớp các bộ của quan hệ sẽ được tái nạp vào file thứ ba với câu lệnh cập nhật (Update). (Chú ý: các lệnh cập nhật được dùng để cài đặt các thuộc tính kế thừa khi chúng được nạp lại đến một CSDL hướng đối tượng đích).

Đoạn chương trình giả mã được mô tả như sau:

BEGIN

Lấy tất cả các quan hệ R_1, ..., R_n trong lược đồ quan hệ;

/*Bước 1: Nạp mỗi lớp với các bộ dữ quan hệ tương ứng*/

for i = 1 **to** n **do**

while (R_i đang còn các bộ) **do**

Kết xuất giá trị thuộc tính khóa không liên kết đến một file tuần tự F_i với câu lệnh Insert;

/* Bước 2: cập nhật mỗi lớp đã nạp với giá trị thuộc tính liên kết của nó */

for j = 1 **to** n **do**

while (R_j đang còn giá trị khóa liên kết khác null) **do**

- Lấy dựa vào các bộ của quan hệ cha từ R_p, mà R_p là một quan hệ cha đối với R_j;
- Kết xuất dựa vào các bộ của quan hệ cha vào file tuần tự F_j với lệnh cập nhật;

/*Bước 3: cập nhật mỗi lớp con để kế thừa giá trị thuộc tính siêu lớp của nó */

for k = 1 **to** n **do**

while (Lớp con tương ứng quan hệ R_k chưa rỗng) **do**

- Lấy dựa vào bộ của siêu lớp quan hệ từ R_S mà R_S là một siêu lớp quan hệ với R_k;
- Xuất dữ liệu của siêu lớp quan hệ đến file tuần tự F_k với câu lệnh cập nhật (Update);

END.

3.1.4.2. Tái nạp các file tuần tự vào CSDL hướng đối tượng

Như một điều kiện tiên quyết của việc chuyển đổi dữ liệu, biên dịch một lược đồ từ mô hình quan hệ sang mô hình hướng đối tượng phải được thực hiện trước. Biên dịch lược đồ hướng đối tượng sau đó sẽ được tạo lập trong CSDL hướng đối tượng. Đầu tiên file tuần tự F_i sẽ được tái nạp vào CSDL hướng đối tượng để làm đầy trong lớp giá

trị của các thuộc tính. Sau đó, file F_j sẽ được tái nạp vào CSDL hướng đối tượng để làm đầy mỗi lớp giá trị các thuộc tính liên kết. Cuối cùng, file tuần tự F_k sẽ được tái nạp để làm đầy trong mỗi lớp con của các giá trị các thuộc tính kế thừa.

3.2. CHUYỂN ĐỔI LƯỢC ĐỒ HƯỚNG ĐỐI TƯỢNG SANG LƯỢC ĐỒ QUAN HỆ NHÚNG

Quá trình chuyển đổi lược đồ từ CSDL quan hệ sang CSDL hướng đối tượng đã được Fong J. [14], [16], Meng W. [25] và Ramanathan C. [27] nghiên cứu đề xuất một cách đầy đủ với các ánh xạ từ lược đồ, phương thức, dữ liệu và truy vấn giữa hai mô hình dữ liệu. Việc chuyển đổi theo chiều ngược lại, từ CSDL hướng đối tượng sang CSDL quan hệ được Fong J. [15] đề xuất sử dụng một mô hình khung làm trung gian giữa cấu trúc lưu trữ là các bảng và giao diện người dùng là hướng đối tượng, các phương thức được chuyển đổi thành các thủ tục trong hệ quản trị CSDL FoxPro, truy vấn OSQL được biên dịch thành các truy vấn SQL. Với hướng tiếp cận này, các đối tượng phức, các mối quan hệ kế thừa giữa các lớp trong lược đồ đối tượng sẽ được phân tích để đưa vào các bảng quan hệ, điều này làm nảy sinh quá nhiều các liên kết giữa các bảng trong lược đồ quan hệ và các truy vấn sẽ chứa các kết nối lớn.

Để làm giảm các liên kết giữa các bảng trong lược đồ, chúng ta tập trung xử lý cho các đối tượng phức, các thuộc tính đa trị và sự kế thừa trong các lớp bằng việc sử dụng mô hình lưu trữ quan hệ nhúng - là cấu trúc dữ liệu phân cấp mà giá trị của các thuộc tính có thể là nguyên thủy hoặc có thể là các quan hệ (quan hệ con) trong quá trình chuyển đổi lược đồ đối tượng sang lược đồ quan hệ. Mở rộng và bổ sung hoàn chỉnh cho phương pháp biên dịch truy vấn OQL thành truy vấn quan hệ SQL.

3.2.1. Tiến trình tổng quát

Quy trình thực hiện tối ưu hóa truy vấn đối tượng OQL bằng phương pháp chuyển đổi lược đồ và biên dịch truy vấn được đề xuất như sau:

Bước 1: Chuyển đổi lược đồ hướng đối tượng sang lược đồ quan hệ nhúng.

Bước 2: Biên dịch các câu truy vấn đối tượng OQL trên CSDL hướng đối tượng sang truy vấn quan hệ SQL.

Bước 3: Tối ưu hóa truy vấn quan hệ SQL (ở bước 2) bằng các phương pháp truyền thống.

3.2.2. Chuyển đổi lược đồ cơ sở dữ liệu hướng đối tượng sang lược đồ quan hệ nhúng

Trước hết, chúng ta xem xét một số định nghĩa liên quan về quan hệ nhúng được sử dụng trong nội dung của phương pháp.

Định nghĩa 3.1. *Quan hệ nhúng được biểu diễn hình thức như sau:*

$$<\text{TênQuanHệ}>(a_1, \ldots, a_n, R_1, \ldots, R_m)$$

Trong đó, a_i ($i = 1,..., n$) là các thuộc tính có giá trị nguyên thủy, R_j ($j = 1,..., m$) là các quan hệ có dạng: $R_j = <\text{TênQuanHệ}>(b_1, \ldots, b_k, R_l)$.

Ví dụ 3.2. Xét các lớp NhanSu, SinhVien:

```
class NhanSu
type tuple (maso: int, hoten: string, pho: string, tpho : string,
            matinh: int, ngaysinh:  tuple (ngay: int, thang: int, nam:int))
class  SinhVien  inherits NhanSu
type   tuple (gvhd: string, dtb: float, hocbong: float, tenkhoa: Khoa)
```

Được biểu diễn bằng quan hệ nhúng như sau:

```
SinhVien(gvhd, dtb, hocbong, tenkhoa, NhanSu(maso, hoten, pho,
                                    tpho, matinh, NgaySinh(ngay, thang, nam)))
```

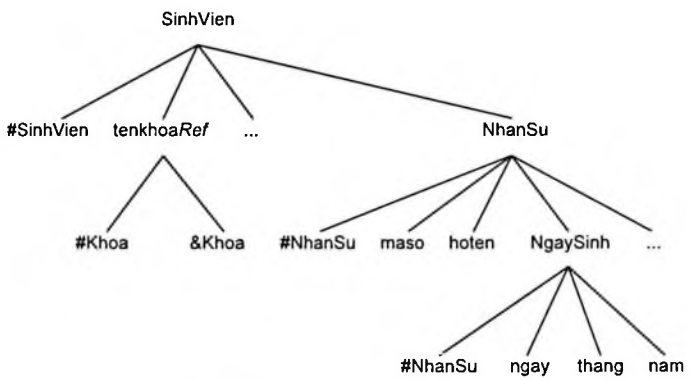
SinhVien là một quan hệ nhúng đa cấp, có thuộc tính tenkhoa là thuộc tính tham chiếu đối tượng, trường hợp này sẽ được xem xét trong phần tiếp theo, siêu lớp NhanSu được chuyển đổi thành quan hệ con, trong đó thuộc tính NgaySinh là thuộc tính đa trị cũng được chuyển thành một quan hệ con trong quan hệ SinhVien.

Định nghĩa 3.2. *Cho trước một quan hệ R. Thuộc tính định danh bộ (TID) là địa chỉ tham chiếu của R-bộ, được ký hiệu là &R. Định danh đối tượng duy nhất (OID) là giá trị được dùng để xác định duy nhất đối tượng, do hệ thống tạo lập, được ký hiệu là #R.*

Thuộc tính &R là một thuộc tính của quan hệ nhúng *R* hoặc là thuộc tính (lưu trữ) trong một quan hệ khác (với vai trò là trường liên kết để mô tả việc lưu trữ các bộ chứa một tham chiếu đến một *R*-bộ).

Ví dụ 3.3. Cho các lớp SinhVien và NhanSu như trong ví dụ 3.2.

Lúc đó, quan hệ SinhVien được khởi tạo như sau:



Hình 3.4. Biểu diễn quan hệ SinhVien dưới dạng cây

3.2.2.1. Đặc tả các tham chiếu đối tượng và chỉ mục kết nối

Giả sử hai quan hệ *R* và *S* quan hệ với nhau qua tân từ P, các lược đồ quan hệ nhúng được mô tả như sau:

Lớp không có quan hệ kế thừa: Các quan hệ *R* và *S* được lưu trữ một cách độc lập:

R (#R, ... các thuộc tính ...)

S (#S, ... các thuộc tính ....)

Tham chiếu nhúng: Giả sử mỗi *R-bộ* có quan hệ với nhiều nhất là một *S*-bộ, ta sẽ lưu trữ địa chỉ của *S*-bộ với mỗi *R*-bộ tương ứng. Tức là, trong phép chèn vào *R*-bộ, tân từ P được định giá trên quan hệ *S* đối với *R*-bộ đó, tìm *S*-bộ tương ứng và lưu trữ các địa chỉ vật lý của nó (&S), hoặc định danh đối tượng duy nhất của nó (#S), hoặc cả hai trong *R*-bộ.

S(S#, ... các thuộc tính ...)

và R(#R, #S, ... các thuộc tính ...)

hoặc R(#R, &S, ... các thuộc tính ...)

hoặc R(#R, #S, &S, ... các thuộc tính ...)

Trong trường hợp tân từ P là một hàm có kiểu đối tượng, phải tổ chức lưu trữ ít nhất là định danh đối tượng duy nhất của đối tượng tham chiếu.

Tham chiếu nhúng kiểu tập: Tương tự, nếu một *R-bộ* có quan hệ nhiều hơn một *S-bộ*, ta có thể lưu trữ một tập các tham chiếu (*OID* và/hoặc *TID*) đối với mỗi *R-bộ* trong một quan hệ con nhúng *SRef* (đối với "*tham chiếu đến S*").

S(S#, ... các thuộc tính ...)

và R(#R, SRef(#S), ... các thuộc tính ...)

hoặc R(#R, SRef(&S), ... các thuộc tính ...)

hoặc R(#R, SRef(#S, &S), ... các thuộc tính ...)

Chỉ mục kết nối: Các con trỏ liên kết các đối tượng liên quan được lưu trữ thành các quan hệ riêng biệt đối với các *đối tượng - bộ*, hai quan hệ JI1 và JI2 là các quan hệ lưu trữ các chỉ mục kết nối giữa quan hệ *R* và *S*:

R(#R, ... một vài thuộc tính ....)

S(#S, ... một vài thuộc tính ....)

JI1(&R, SRef(&S))

và JI2(&S, SRef(&R))

Như vậy, một tập các địa chỉ được nhóm trong mỗi cặp chỉ mục kết nối.

3.2.2.2. Chuyển đổi các thành phần trong CSDL hướng đối tượng

Quá trình ánh xạ các lược đồ CSDL hướng đối tượng từ mức khái niệm lên các quan hệ nhúng sẽ được tiến hành theo từng bước qua các khái niệm cơ sở của mô hình đối tượng và đưa ra một số phương án lựa chọn cho tiến trình chuyển đổi. Từ việc lựa chọn sự biểu diễn cho mỗi khái niệm, sẽ có một không gian quyết định khá lớn đặt cơ sở cho việc thiết kế và xây dựng lớp bằng công cụ thiết kế CSDL vật lý.

a) Đối tượng

Một đối tượng được đặc tả bằng một định danh duy nhất và được khởi tạo bởi hệ thống. Mọi dữ liệu liên quan đến đối tượng sẽ tham chiếu đến định danh này. Giả sử kiểu đối tượng là T thì các quan hệ nhúng chứa định danh đối tượng duy nhất của các đối tượng có kiểu T là $\#T$.

b) Phương thức của lớp

Về mặt nguyên lý, có thể biểu diễn một phương thức như là một quan hệ nhị nguyên, với một thuộc tính cho tham số định danh đối tượng duy nhất và một thuộc tính khác cho giá trị kết quả. Trường hợp các phương thức có giá trị - tập thì thuộc tính thứ hai sẽ là một quan hệ con thực sự của các bộ con đơn nguyên. Vì vậy, hàm đơn trị $f_s : T_1 \rightarrow T_2$ và hàm đa trị $f_m : T_1 \rightarrow \text{set}(T_3)$ được biểu diễn bằng hai quan hệ nhị nguyên tương ứng như sau:

$$f_s(\#T_1, \#T_2) \text{ và } f_m(\#T_1, T_3Ref(\#T_3))$$

Tham chiếu logic và tham chiếu vật lý: Hàm cho kết quả là một đối tượng (tập đối tượng), được cài đặt bằng cách lưu trữ định danh đối tượng duy nhất (tham chiếu logic) của các đối tượng kết quả hoặc chứa định danh bộ (TID). Các hàm đơn trị trở thành các quan

hệ 3 - thuộc tính và các hàm đa trị trở thành các quan hệ có chứa các quan hệ con. Quan hệ có chứa các tham chiếu vật lý đối với hai hàm f_s và f_m, sẽ cho kết quả như sau: T_1 (#T_1 , f_s#, f_s&, f_m#Set(#T_3 , &T_3)).

c) Kiểu, lớp và sự kế thừa

Từ sự phân cấp kế thừa kiểu, sẽ có hai phương án lựa chọn để chuyển đổi: Thứ nhất, đối với các hàm gói, sự kế thừa các hàm trong kiểu bảng tương ứng được cài đặt với các kiểu con. Thứ hai, các đối tượng sẽ được biểu diễn tương ứng với một bộ - đối tượng trong bảng của kiểu con đặc trưng nhất hoặc trong nhiều bộ - đối tượng, mỗi bộ chứa trong mỗi siêu kiểu.

Tiếp theo, xét phân cấp giữa các kiểu (cấu trúc, sự kế thừa phương thức) và phân cấp giữa các lớp (mối quan hệ bao hàm).

Kiểu: Mỗi kiểu đối tượng ***T*** được ánh xạ đến một kiểu bảng *T'* chứa ít nhất một thuộc tính *#T*. Các thuộc tính bổ sung trong bảng *T'* do các hàm (phương thức) đóng gói trong kiểu ***T*** và các hàm trả về ***T*** - đối tượng.

Lớp: Mỗi lớp ***c*** được chuyển đổi thành một khung nhìn trên cơ sở kiểu bảng. Nếu lớp được định nghĩa bởi một lượng từ "*forall*", lượng từ này được dùng như điều kiện lựa chọn trên lớp. Nếu lớp có một số đối tượng thành viên được xác định bởi "*forsome*" thì kiểu bảng cơ sở được mở rộng bằng một thuộc tính logic *B* có giá trị *true* nếu và chỉ nếu đối tượng là một thành viên của lớp ***c***.

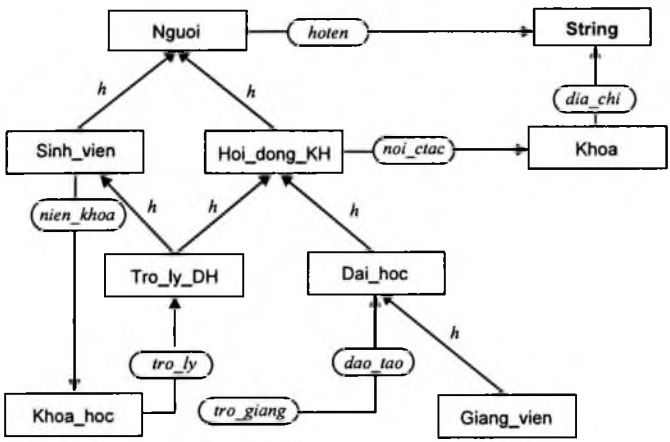
Sự kế thừa: Lớp con được chuyển đổi tương ứng với một kiểu bảng. Xem xét hai trường hợp sau:

Siêu lớp gốc (lớp đặc trưng nhất) được lưu trữ trong một bảng (bảng cơ sở). Các hàm trong siêu lớp gốc (bảng cơ sở) cũng thuộc về các bảng của các lớp con.

Lớp con có kế thừa từ các siêu lớp, mỗi siêu lớp được tạo thành một quan hệ con, có chứa thuộc tính định danh đối tượng của lớp.

3.2.2.3. Chỉ mục

Sử dụng chỉ mục lồng hoặc chỉ mục đường dẫn.



Hình 3.5. Đồ thị lược đồ

Định nghĩa 3.3. *Đường dẫn kết nhập.*

Một đường dẫn kết nhập P được định nghĩa là $t_1.p_1.p_2...p_n$ *với* $n \geq 1$. *Miền của một đường dẫn kết nhập, ký hiệu p_domain(P) tương ứng với miền của* p_n, $p\_domain(p_n)$. *Một đường dẫn kết nhập được gọi là đúng cú pháp nếu:*

(i) τ_1 *là một kiểu trong T;*

(ii) p_1 *là một thuộc tính của kiểu* τ_1, *tức là,* $\tau_1 \leq p\text{-}type(p_1)$;

(iii) p_i *là một thuộc tính của miền* p_{i-1}, *mà* $p\_domain(t_1.p_1.p_2...p_{i-1}) \leq p\_type(p_i)$ *với* $1 < i \leq n$.

Các điều kiện trên có nghĩa là các miền p_1 , ..., p_{n-1} là các kiểu đối tượng và miền của p_n là một kiểu đối tượng hoặc là một kiểu. Hơn nữa, $length(P) = n$ và $types(P) = \tau_1 \cup (\cup_{1 < i \leq n} p\_domain(p_i))$ là độ dài của đường dẫn P và tập các kiểu tham chiếu bởi đường dẫn P.

Ví dụ 3.4. Xét các đường dẫn kết nhập sau:

P1	Nguoi.hoten
P2	SinhVien.nien_khoa.tro_ly.noi_ctac.dia_chi
P3	GiangVien.dao_tao.tro_giang
P4	SinhVien.KhoaHoc.tro_giang.hoten

- $length(P_1) = 1$,

  $types(P_1) = \{\tau_1 = \text{Nguoi}\}$

  $p\_domain(P_1) = \text{string}$

- $length(P_2) = 4$

  $types(P_2) = \{\tau_1 = \text{SinhVien}, \tau_2 = \text{KhoaHoc}, \tau_3 = \text{DaiHoc}, \tau_4 = \text{Khoa}\}$

  $p\_domain(P_2) = \text{string}$

- $length(P_3) = 2$

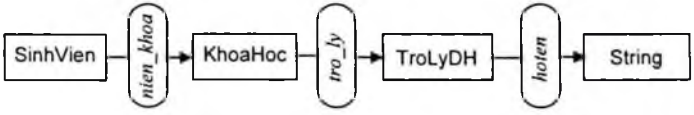
  $types(P_3) = \{\tau_1 = \text{GiangVien}, \tau_2 = \text{KhoaHoc}\}$

  $p\_domain(P_3) = \text{HoiDongKH}$

- $length(P_4) = 3$

  $types(P_4) = \{\tau_1 = \text{SinhVien}, \tau_2 = \text{KhoaHoc}, \tau_3 = \text{TroLyDH}\}$

  $p\_domain(P_4) = \text{string}$



Hình 3.6. Đường dẫn của P_4

Định nghĩa 3.4. *Cho đường dẫn kết nhập P là $t_1.p_1.p_2...p_n$, $o_1.p_1...p_n$ là biểu thức đường dẫn của P nếu và chỉ nếu $o\text{-}type(o_1) \leq \tau_1$. Ngữ nghĩa của biểu thức đường dẫn $o_1.p_1...p_n$ của đường dẫn P cho bởi hàm val: $E(P) \rightarrow P(Ext(p\_domain(P)))$ với $E(P)=\{ o_1.p_1...pn \mid o_1 \in Ext(\tau_1) \}$. val là các ánh xạ từ tập các biểu thức đường dẫn của P đến tập tất cả các tập giá trị kết quả từ p_domain(P). Ký hiệu [o.p] là tập*

các giá trị lưu trữ trong o đối với thuộc tính p. Cho V là một sự rút gọn của val($o_1.p_1...p_{n-1}$), val được định nghĩa như sau:

$$val(o_1.p_1...p_n) = \begin{cases} [o_1.p_1] & \text{nếu } n = 1 \\ \cup_{o \in V}[o.p_n] & \text{nếu } n > 1 \end{cases}$$

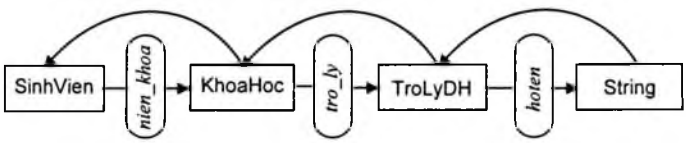
Định nghĩa 3.5. *Đa chỉ mục.*

Đa chỉ mục MX(P) đối với đường dẫn $P = t_1.p_1...p_n$ được xác định là $MX(P) = \{IX(\tau_i.p_i)$ với $1 \le i \le n\}$. Chỉ mục thành phần $LX(P_i)$ đối với đường dẫn con $P_i = \tau_i.p_i$ được xác định:

$$LX(P_i) = \{ (o, S) \mid S = \{ o' \mid o \in val(o'.p_i) \} \}$$

Ví dụ 3.5. Đa chỉ mục đối với đường dẫn P_4 = Sinh_vien.nien_khoa.tro_ly.hoten gồm ba thành phần như sau:

MX(Sinh_vien.nien_khoa.tro_ly.hoten) = {*IX*(Sinh_vien.nien_khoa), *IX*(Khoa_hoc.tro_ly), *IX*(Tro_ly_DH.hoten) }

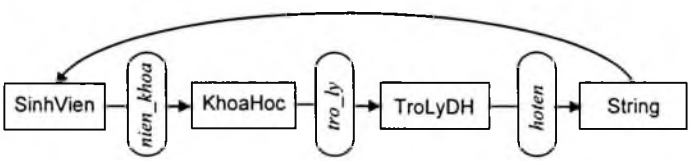


Hình 3.7. Đường dẫn của P_4

a) Chỉ mục lồng

Định nghĩa 3.6. *Chỉ mục lồng của đường dẫn $P = t_1.p_1.p_2...p_n$ được xác định: $NX(P) = \{ (o, S) \mid S = \{ o' \mid o \in val(o'.p_1... p_n) \} \}$*

Ví dụ 3.6. Dữ liệu được lưu trữ trong chỉ mục lồng của P_4:

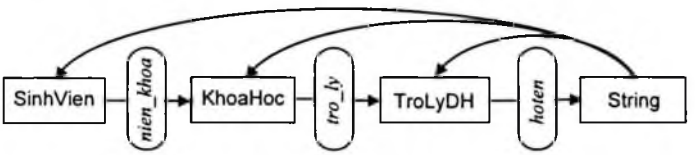


Hình 3.8. Đường dẫn của P_4

b) Chỉ mục đường dẫn

Định nghĩa 3.7. *Chỉ mục đường dẫn đối với đường dẫn* $P = t_1.p_1.p_2...p_n$ *được xác định:* $PX(P) = \{ (o, S) \mid S = (o_j.o_{j+1}... o_n) \}$ *với* $1 \le j \le n$ *mà* $o_j.o_{j+1}... o_n.o$ *là trường hợp cụ thể của P kết thúc ở o và không cụ thể* $o_k...o_j.o_{j+1}... o_n.o$ *của P với* $1 \le k < j \le n$.

Ví dụ 3.7. Chỉ mục đường dẫn của P_4:



Hình 3.9. Đường dẫn của P_4

3.2.2.4. Thuật toán chuyển đổi lược đồ đối tượng sang lược đồ quan hệ nhúng

Dựa trên những phân tích và cài đặt ở trên, thuật toán chuyển đổi lược đồ đối tượng sang lược đồ quan hệ nhúng được thể hiện như sau:

Giả sử lược đồ đối tượng được biểu diễn thành dãy $S = (s_1, s_2, ..., s_n)$, s_i là các lớp trong lược đồ.

Thuật toán 3.9: *Chuyển đổi lược đồ đối tượng sang lược đồ quan hệ nhúng*

Vào: Lược đồ đối tượng $S(s_1, s_2, ..., s_n)$

Ra: Lược đồ quan hệ nhúng *NR*.

Phương pháp:

(1) Khởi tạo $NR := \varnothing$

(2) **for** mỗi $s_i \in S$ **do**

(3) **if** s_i là siêu lớp gốc **then**

(4) Khởi tạo quan hệ *r* chứa các thuộc tính có giá trị nguyên thủy trong s_i

(5) Bổ sung thuộc tính $\#s_i$ vào *r*

(6) $NR := NR \cup r$

(7) **else if** s_i là lớp có kế thừa từ các siêu lớp s_j **then**

(8) Khởi tạo quan hệ nhúng $r_i := \varnothing$

(9) **for** mỗi siêu lớp s_j **do**

(10) Tạo quan hệ con r_j chứa các thuộc tính của lớp s_j

(11) Bổ sung thuộc tính $\#s_j$ vào r_j

(12) Bổ sung r_j vào r_i

(13) **for** mỗi thuộc tính $s_i.a_k \in s_i$ **do**

(14) **if** a_k là định danh đối tượng **then**

(15) Bổ sung thuộc tính $\#s_i$ vào r_i

(16) **if** (a_k là thuộc tính đơn trị, có giá trị nguyên thủy) **then**

(17) Bổ sung thuộc tính a_k tương ứng trong quan hệ r_i

(18) **else if** a_k là thuộc tính đa trị **then**

(19) Tạo quan hệ con r_i' tương ứng với a_k

(20) Bổ sung thuộc tính $\#a_k$ trong r_i'

(21) Bổ sung r_i' vào r_i

(22) **else if** a_k là các thuộc tính tham chiếu **then**

(23) **if** (a_k là tham chiếu nhúng kiểu tập đến lớp s_l) **then**

(24) Tạo quan hệ con $a_kRef(\#s_l, \&s_l)$

(25) Bổ sung quan hệ a_kRef vào quan hệ r_i

(26) **else** Bổ sung hai thuộc tính $\{\#s_l, \&s_l\}$ vào r_i

(27) $NR := NR \cup r_i$

Thuật toán 3.9 tập trung xử lý trên các lớp chứa các thuộc tính đa trị, các tham chiếu (không tạo thành chu trình) và mối quan hệ kế thừa giữa các lớp. Thuật toán cho kết quả là lược đồ quan hệ nhúng của lược đồ đối tượng.

Đối với các phương thức của lớp được chuyển đổi tương ứng thành các quan hệ nhị nguyên, lưu trữ độc lập và có liên kết đến lớp qua các tham chiếu logic (tham chiếu vật lý).

Thuật toán 3.10: *Chuyển đổi phương thức của lớp*

Vào: Lược đồ đối tượng $S(s_1, s_2, \ldots, s_n)$ và Lược đồ quan hệ nhúng $NR(r_1, r_2, \ldots, r_j)$

Ra: Lược đồ quan hệ nhúng *NR*.

Phương pháp:

(1) **for** mỗi lớp $s_i \in S$ **do**

(2) **for** mỗi phương thức f_k của lớp s_i **do**

(3) **if** f_k là hàm đơn trị **then** // Hàm đơn trị $f_k: \boldsymbol{T_1} \rightarrow \boldsymbol{T_2}$

(4) Tạo quan hệ f_k (#T_1, #T_2)

(5) Bổ sung các thuộc tính {f_k#, f_k&} vào quan hệ r_k

(6) **else** Tạo quan hệ f_k (#T_1, T_3*Ref*(#T_3))

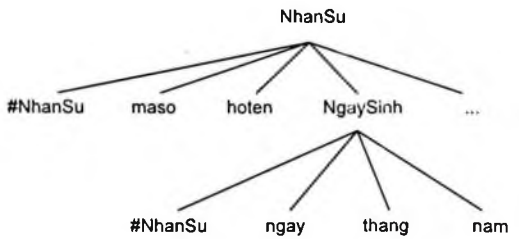
//Hàm đa trị $f_k: \boldsymbol{T_1} \rightarrow$ set($\boldsymbol{T_3}$)

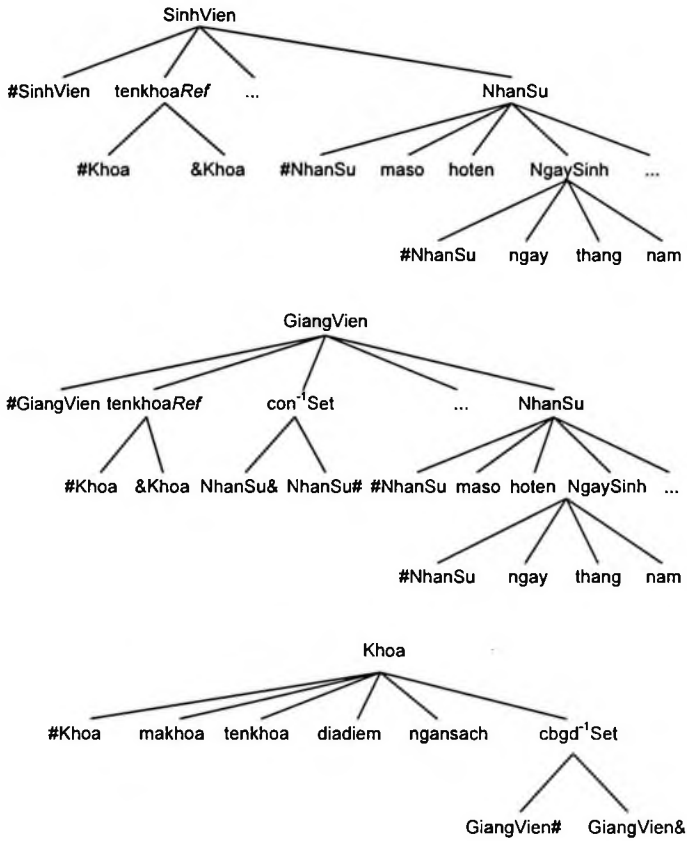
(7) Bổ sung quan hệ f_k#Set(#T_3 , &T_3) vào r_k

(8) Bổ sung f_k vào *NR*

3.2.2.5. Ví dụ tổng quát

Sử dụng lược đồ đối tượng trong ví dụ 3.2, lược đồ CSDL quan hệ nhúng được chuyển đổi tương ứng như sau:





Hình 3.10. Biểu diễn lược đồ quan hệ nhúng dưới dạng cây

Tiến trình xác định các phương án thực thi truy vấn trong quá trình tối ưu hóa truy vấn đối tượng là khá phức tạp, điều này xuất phát từ ngữ nghĩa của mô hình CSDL hướng đối tượng, là sự biểu diễn phong phú của các đối tượng phức, sự kế thừa và tính đóng gói. Bên

cạnh đó, các kỹ thuật tối ưu hóa truy vấn quan hệ đã được nghiên cứu hoàn chỉnh và rất hiệu quả. Do đó, việc nghiên cứu để biên dịch các truy vấn đối tượng thành các truy vấn quan hệ và áp dụng các kỹ thuật tối ưu hóa truy vấn quan hệ sẽ mang lại các giải pháp có tính ứng dụng cao trong thực tế khi xử lý các truy vấn đối tượng trên các hệ thống CSDL hướng đối tượng.

KẾT LUẬN

Mô hình dữ liệu quan hệ được E. F. Codd đưa ra vào năm 1970 và nó là mô hình dữ liệu được chọn để cài đặt cho các hệ thống CSDL, mặc dù mô hình CSDL quan hệ không phải là mô hình dữ liệu đầu tiên trong các hệ quản trị CSDL, nhưng với sự hỗ trợ các ngôn ngữ khai báo, khá đơn giản nhưng hiệu quả và tập các phép tính khá đầy đủ và hoàn thiện mà mô hình CSDL quan hệ trở nên phổ biến. Phần lớn các hệ thống quản trị CSDL trước đây đang sử dụng mô hình dữ liệu quan hệ là mô hình chính trong hệ thống, do đó vấn đề đặt ra là việc tận dụng và chuyển đổi dữ liệu hiện đang lưu trữ trong các CSDL quan hệ sang CSDL hướng đối tượng là vấn đề cần quan tâm.

Chúng ta đã xem xét một phương pháp chuyển đổi dữ liệu từ CSDL quan hệ sang CSDL hướng đối tượng, quá trình chuyển đổi này gồm hai giai đoạn, giai đoạn thứ nhất gồm các bước tinh chỉnh lược đồ quan hệ sao cho phù hợp với việc chuyển đổi các khái niệm trong mô hình hướng đối tượng và quá trình ánh xạ lược đồ từ quan hệ sang hướng đối tượng. Giai đoạn thứ hai, gồm các thao tác đối với dữ liệu vật lý mà CSDL quan hệ đã lưu trữ, đó là quá trình tái nạp dữ liệu các bộ từ CSDL quan hệ lên CSDL hướng đối tượng tương ứng được kiến lập từ các lược đồ ở giai đoạn thứ nhất.

CÂU HỎI ÔN TẬP CHƯƠNG 3

Câu 3.1. Sử dụng hệ quản trị CSDL quan hệ mà Anh (chị) biết để cài đặt CSDL quan hệ nhúng được mô tả ở hình 3.10.

Câu 3.2. Hãy phân tích và đánh giá sự toàn vẹn dữ liệu khi cài đặt các đối tượng bằng các quan hệ nhúng.

Câu 3.3. Với các đối tượng được mô tả trong hình 1.8, 1.9, 1.10. Hãy biểu diễn các đối tượng bằng các quan hệ nhúng tương đương.

Câu 3.4. Cài đặt các thuật toán trong mục 3.1.

Câu 3.5. Trên cơ sở các quan hệ nhúng, hãy đề xuất phương pháp biểu diễn dữ liệu lai ghép giữa bảng và các danh sách để đặc tả cho các đối tượng trong CSDL hướng đối tượng.

Câu 3.6. Hãy đề xuất một phương pháp cài đặt lưu trữ lớp và phương thức khi chuyển đổi sang các quan hệ nhúng.

CHƯƠNG

4

NGÔN NGỮ TRUY VẤN ĐỐI TƯỢNG OQL

Ngôn ngữ truy vấn đối tượng OQL (Object Query Language) là ngôn ngữ truy vấn có tính thân thiện trong biểu diễn truy vấn, khả năng đặc tả các đối tượng phức trong CSDL hướng đối tượng - là cơ sở để xem xét lựa chọn ngôn ngữ truy vấn thích hợp sử dụng trong các phương pháp tối ưu hoá truy vấn đối tượng. Bên cạnh đó, các phép toán đại số đối tượng là công cụ được sử dụng để "viết lại" các truy vấn dưới dạng các biểu thức đại số đối tượng, các biểu thức này sẽ là đầu vào của các thuật toán tối ưu hóa truy vấn đối tượng dựa trên tập luật.

4

OQL là ngôn ngữ truy vấn CSDL hướng đối tượng đã đề xuất trong ODMG-93. Phiên bản cuối cùng của OQL trong ODMG-93 (Release 1.2) là siêu tập của SQL92, được mô tả như là một phần của ODMG chuẩn. OQL là ngôn ngữ truy vấn không đưa ra môi trường lập trình đầy đủ, chẳng hạn như, OQL không thể biểu diễn cho tất cả các tính toán phức tạp. Do đó, nó không phải là hệ tính toán đầy đủ. Vì vậy, để tăng khả năng tính toán cho OQL cần phải liên kết với các ngôn ngữ lập trình khác:

(i) OQL được dùng để mô tả truy vấn và các thao tác của đối tượng;

(ii) Ngôn ngữ định nghĩa đối tượng (ODL): Sử dụng mã OQL để định nghĩa giao diện với các kiểu đối tượng;

(iii) Các ngôn ngữ lập trình khác như: C++, Smalltalk và Java được dùng để cài đặt các phương thức và các chương trình ứng dụng. Đồng thời có thể nhúng các câu lệnh OQL trong các thao tác trên CSDL đối tượng.

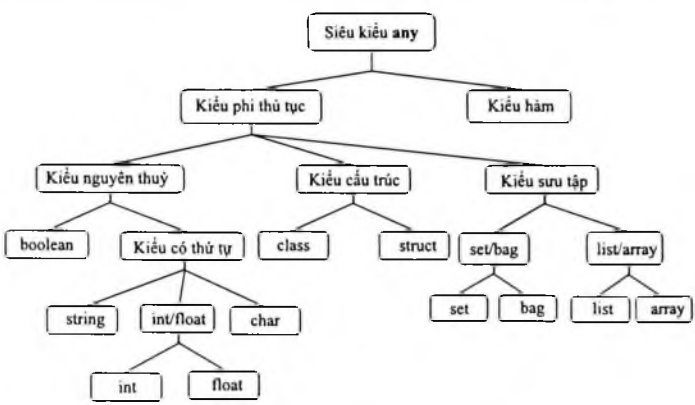
Trong (iii) OQL được dùng như ngôn ngữ truy vấn nhúng, nó cũng có thể được sử dụng như một ngôn ngữ truy vấn độc lập, nghĩa là, người dùng có thể biểu diễn các thao tác CSDL đối tượng hoàn toàn bằng OQL mà không cần một chương trình nhúng nào được viết bằng ngôn ngữ lập trình khác.

OQL được thiết kế theo hướng phát triển kế thừa, đó là thiết kế một ngôn ngữ mới đầy đủ dựa trên SQL, ngôn ngữ chuẩn của CSDL quan hệ. Thiết kế của OQL ở dạng hàm, kết quả của truy vấn có kiểu, điều này cho phép kết quả của truy vấn này là đầu vào của một truy vấn khác, vì vậy các truy vấn phức tạp có thể được xây dựng bằng

OQL. Bằng cách tích hợp các tính năng của SQL, OQL xây dựng trên nền tảng của CSDL quan hệ. Mục đích của mô hình ODMG là duy trì một mức độ tương thích giữa kỹ thuật CSDL hướng đối tượng và kỹ thuật CSDL đối tượng - quan hệ bằng cách tích hợp các tính năng của phiên bản tương lai của SQL trong phiên bản tương lai của OQL.

4.1. KIỂU VÀ LƯỢC ĐỒ SUY DẪN KIỂU TRONG NGÔN NGỮ TRUY VẤN OQL

Hệ thống các kiểu dữ liệu nguyên thuỷ và các kiểu dữ liệu suy dẫn gắn với phân cấp kiểu được giới thiệu trong hình 4.1. Trên cây phân cấp, ta nhận thấy rằng các *kiểu tổng quát* là các nút nằm ở nút nhánh trên cây, các kiểu này sẽ không được sử dụng trong các lược đồ CSDL (khai báo hình thức). Kiểu được gọi là *kiểu đặc trưng*, nếu nó được suy dẫn từ các kiểu cơ sở. Tất cả các nút lá trong hình 4.1 là các kiểu đặc trưng. Ở vị trí của các tập thực thể, mô hình mạng đưa ra *kiểu mẫu tin logic*. Một kiểu mẫu tin logic là tên gán cho một tập các mẫu tin, được gọi là các mẫu tin logic. Nó được cấu tạo bởi các trường chứa các giá trị cơ bản như: số nguyên, chuỗi ký tự,... Tập các tên trường và kiểu của chúng cấu tạo nên khuôn dạng mẫu tin lc gic.



Hình 4.1. Sự phân cấp các kiểu trong OQL

Các kiểu tổng quát của hệ thống các kiểu kết quả được thể hiện như sau:

Kiểu: T ::= int | float | bool | char | string | void

| T × ... × T → T

| bag(T) | set(T) | list(T) | array(T) | struct(ℓ: T, ..., ℓ:T)

| C

| any | nonfunctional

| atomic | orderable | int/ float

| constructor(ℓ: T, ..., ℓ:T)

| collection(T) | set/ bag(T) | list/ array(T)

4.2. TRUY VẤN SELECT ... FROM ... WHERE

Cú pháp của khối lệnh trọng tâm **select** trong OQL là:

select [distinct]	<Danh sách kết quả>
from	<Danh sách lớp/ sưu tập lớp>
where	<Biểu thức điều kiện>
[group by	<Các thuộc tính phân nhóm>**]**
[having	<Điều kiện lọc>**]**
[order by	(<Khóa sắp xếp> **asc|desc**, ...)]

Trong đó, từ khóa **distinct** nếu sử dụng thì kết quả của truy vấn có kiểu tập đối tượng. Mệnh đề **group by** phân nhóm trên lớp sưu tập, các đối tượng được nhóm theo giá trị của các thuộc tính phân nhóm. Mỗi cấu trúc của nhóm chứa một nhóm các thuộc tính và nhóm các đối tượng có cùng giá trị. Mệnh đề **having** đặt điều kiện để lựa chọn (lọc). Mệnh đề **order by** sắp thứ tự các đối tượng trong lớp sưu tập kết quả.

Định nghĩa kiểu: d ::= define x as q

| define x(x, ..., x) as q

Ký hiệu *b, f, i, c, s* tương ứng là các kiểu dữ liệu boolean, float, integer, character và string, x là tập đếm được các định danh, ℓ là tập đếm được các nhãn và C là tập đếm được các tên lớp, *unop* và *binop* là tập các phép toán 1-ngôi, 2-ngôi.

Kiểu: T ::= int | float | bool | char | string | void

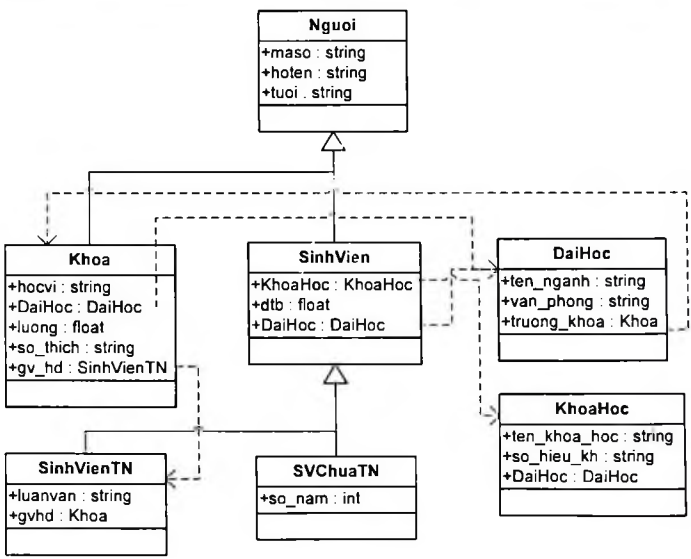
| $T \times \ldots \times T \rightarrow T$

| bag(T) | set(T) | list(T) | array(T)

| struct(ℓ: T, ..., ℓ:T)

| C

Sử dụng lược đồ CSDL VienDH được biểu diễn trong hình 4.2 minh hoạ cho các ví dụ về các dạng thức biểu diễn của truy vấn OQL:



Hình 4.2. Biểu diễn lược đồ VienDH bằng ký pháp UML

4.2.1. Truy nhập các đặc trưng của đối tượng

Người ta sử dụng tên của đối tượng kết hợp với tên các đặc trưng của đối tượng (thuộc tính, mối quan hệ và các phép toán) để truy nhập vào các thành phần của một đối tượng hoặc các đặc tính liên quan đến nó.

Đối tượng và tên các đặc trưng được đặt trong *đường dẫn*. Các bước trong đường dẫn được ngăn cách bằng dấu chấm '.' hoặc dấu mũi tên '→'. Dấu chấm và dấu mũi tên có thể được dùng thay cho nhau, nhưng người ta quy ước dùng dấu chấm là ký hiệu đường dẫn từ một đối tượng đến các đặc trưng của nó, còn dấu mũi tên là ký hiệu đường dẫn từ một đối tượng này có mối quan hệ với một đối tượng khác.

Ví dụ 4.1. Cho biết tên của các sinh viên chưa tốt nghiệp có tham gia các khóa học thuộc khoa có Trưởng khoa tên là Thanh.

```
select  s.hoten
from  SVChuaTN  as  s
where  s.DaiHoc.truong_khoa.hoten = 'Thanh'
```

Cấu trúc như s.DaiHoc.truong_khoa.hoten gọi là biểu thức đường dẫn.

4.2.2. Thuộc tính và lượng từ

Biểu thức logic trong OQL thực hiện việc kiểm tra trên các thành viên của một lớp sưu tập. Các phép toán kiểm tra được sử dụng trong biểu thức lôgic là:

(i) forall - kiểm tra với một điều kiện đúng (*true*) cho tất cả các đối tượng chứa trong lớp sưu tập.

(ii) exists - kiểm tra sự tồn tại của một đối tượng trong lớp sưu tập thỏa mãn một điều kiện cho trước.

(iii) unique - kiểm tra tính duy nhất của các đối tượng trong lớp sưu tập.

(iv) in - kiểm tra một đối tượng có thuộc về một lớp sưu tập hay không.

(v) some, any và all - đây là những lượng từ được dùng để so sánh sự ít nhất (some hay any) hay với mọi (all) cho các đối tượng thuộc (in) trong một lớp sưu tập.

(vi) set inclusion - Các phép toán so sánh bao hàm {<, <= , >, >=} áp dụng với các lớp sưu tập để kiểm tra sự bao hàm của một lớp sưu tập này và lớp kia.

Cú pháp của truy vấn có lượng từ forall, exists và in là:

<Truy vấn> ::= **forall** <Truy vấn> **in** <Truy vấn> : <Truy vấn>
| **exists** <Tên> **in** <Truy vấn> : <Truy vấn>
| **exists** | **unique** (<Truy vấn>)
| <Truy vấn> <Phép toán so sánh> [**some** | **any** | **all**] <Truy vấn>

Ví dụ 4.2. Truy vấn sử dụng toán tử in và lượng từ exists.

```
select  f.hoten
from  Khoa as f
where exists  'Am nhac'  in  f.so_thich
```

Trong biểu thức đường dẫn có thể dùng nhiều hơn một lượng từ với sự kết hợp giữa exists và forall.

Ví dụ 4.3.

```
select  s.hoten
from  Khoa as f
where exists  f.gv_hd  in
        (forall KhoaHoc.DaiHoc.ten_nganh = 'Cong nghe thong tin')
```

4.2.3. Biến tham chiếu

Biến vùng mô tả cho một biểu thức đường dẫn có thuộc tính cuối cùng là một thuộc tính phức được gọi là *biến tham chiếu*. Người ta sử dụng các biến vùng biểu diễn cho các đối tượng của lớp trong một biểu thức đường dẫn.

Ví dụ 4.4.

```
select s.hoten
from SinhVienTN as s, s.gvhd as f
where s.tuoi > f.tuoi
```

Trong đó, *f* là một biến tham chiếu đến các đối tượng thuộc lớp SinhVienTN.

4.2.4. Bộ phận của một phân cấp lớp

Cho *c* là một siêu lớp của c_1, c_2, ..., c_n. Khi thực hiện một truy vấn trên *c*, việc tìm kiếm sẽ được thực hiện với tất cả các đối tượng trong phân cấp lớp có gốc tại *c*, để truy vấn trên một vài lớp xác định chứ không phải tất cả các lớp, người ta sử dụng kết hợp các phép toán tập hợp như union, difference và except trong truy vấn.

Các phép toán trên tập hợp được cài đặt tương đương với các công thức trong lý thuyết tập hợp với cú pháp:

<Truy vấn> :: = <Truy vấn> **union / intersect / except** <Truy vấn>

Ví dụ 4.5.

```
select s.hoten
from (SinhVien except (SinhVienTN union SVChuaTN)) as s
```

4.2.5. Phương thức tham chiếu

Phương thức trong truy vấn OQL được thể hiện qua hai dạng. Dạng thứ nhất là *phương thức thuộc tính - suy dẫn*, sử dụng để tính giá trị cho mỗi đối tượng trong lớp mà phương thức tác động. Phương thức thuộc tính - suy dẫn có thể được sử dụng giống như một thuộc tính trong truy vấn. Dạng thứ hai là *Phương thức tân từ*, trả về một giá trị kiểu boolean đối với mỗi đối tượng trong lớp. Ví dụ, ta có phương thức "VienDaiHoc( ) : Boolean" trên lớp DaiHoc, phương thức này trả về giá trị *true* nếu Đại học có trên 5 trường thành viên hoặc trên 30.000 sinh viên và *false* nếu ngược lại.

4.2.6. Kết xuất một cấu trúc

Mô hình dữ liệu hướng đối tượng cho phép sử dụng các đối tượng có cấu trúc tập, danh sách và bộ để thiết lập các đối tượng phức từ các đối tượng đơn. Ngôn ngữ truy vấn OQL cho phép thiết lập các

giá trị phức trong kết quả trả về của một truy vấn bằng việc dùng các phép toán thiết lập kiểu đối tượng phức ở mệnh đề select với cú pháp:

<Truy vấn> ::= **struct** ([<Định danh> : <Truy vấn> [, <Định danh> : <Truy vấn>]*])
|set | bag | list | array ([<Truy vấn> [,<Truy vấn>]*])

Ví dụ 4.6.

```
select   tuple(masv : s.masv, hoten : s.hoten,
         Cnghe_Shoc : select   c
                      from   s.KhoaHoc  as  c
                      where  c.DaiHoc.ten_nganh = 'Sinh hoc')
from   SVChuaTN   as   s
where  s.dtb  > 3
```

4.3. ĐẠI SỐ ĐỐI TƯỢNG

Các phép toán đại số đối tượng được chia làm sáu loại: phép toán đối tượng, phép toán bộ, phép toán tập hợp, phép toán "túi", phép toán danh sách và phép toán mảng.

4.3.1. Phép toán đối tượng

Mỗi đối tượng được biểu diễn như một bộ ba (*oid*, *TênLớp*, *GiáTrị*), *oid* là định danh của đối tượng và *TênLớp* là kiểu của đối tượng. Giá trị của đối tượng do người sử dụng định nghĩa thường là kiểu bộ. Có ba phép toán đối tượng:

- Chiếu định danh (π_o): Phép toán π_o nhận vào một đối tượng và trả về định danh của đối tượng.

- Chiếu giá trị (π_V): Phép toán π_V nhận vào một đối tượng và trả về giá trị của đối tượng.

- Chiếu đối tượng (π_D): Phép toán π_D nhận vào một định danh đối tượng và trả về đối tượng có định danh tương ứng.

4.3.2. Phép toán bộ

- Thiết lập bộ: **tuple**($a_1 : v_1$, ... , $a_n : v_n$). Phép toán **tuple** nhận vào các thuộc tính và các cặp giá trị ($a_i : v_i$) và trả về một bộ (a_1, ... , a_n).

- Chiếu bộ ($\pi_{(Attrs)}$): Phép toán $\pi_{(Attrs)}$ nhận một bộ và trả về một bộ con với tên các thuộc tính được mô tả trong tập *Attrs*.

- Trích xuất giá trị thuộc tính (π_{Attr}): Nhận vào một bộ và trả về giá trị của thuộc tính mô tả *Attr*.

- Ghép bộ (**tuple_cat**): Nhận vào hai bộ và nối chúng vào một bộ mới. Phép toán này dùng thay thế tích Đề các.

4.3.3. Phép toán tập hợp

- Thiết lập tập hợp (**set**( ) hoặc {}): Khởi tạo một tập từ một số phần tử.

- Hợp của hai tập hợp: **set_union**.

- Hiệu của hai tập hợp: **set_diff**.

- Chọn trên tập hợp ($\sigma^s_{\lambda s.f}$): Phép toán $\sigma^s_{\lambda s.f}$ nhận vào một tập (thường là một tập đối tượng) và trả về một tập (đối tượng) sao cho mỗi phần tử ở tập kết quả đều thoả mãn điều kiện của biểu thức *f*.

Ví dụ, cho A = {2, 6, 4, 9}, $\sigma^s_{\lambda s.s > 5}$ (A) = {6, 9}. Ký hiệu "λ" trong "λs" được dùng như một biến đếm các phần tử trong tập đã cho. Với tập đã cho là một lớp, *s* là một biến thể hiện lớp.

- Làm phẳng tập (**set_flat**): Phép toán **set_flat** nhận vào một tập các tập và trả về một tập chứa phần hợp của các phần tử của các tập lồng nhau.

Ví dụ, *set_flat*({1, 2, 3}, {2, 3, 4}, {4, 5}) = {1, 2, 3, 4, 5}.

- Áp dụng hàm trên tập (**set_apply**$_{\lambda s.e}$): Phép toán **set_apply**$_{\lambda s.e}$ nhận vào một tập và áp dụng biểu thức đại số *e* cho mỗi phần tử trong tập.

Ví dụ, A = {{1, 2, 3}, {2, 3, 4}, {4, 5}} là một tập các tập, thì set_apply$_{\lambda s.A\ set_diff\ \{3\}}$(A) = {{1, 2}, {2, 4}, {4, 5}}. Phép toán *set_apply* và chiếu bộ (hoặc trích xuất giá trị thuộc tính) thường được

sử dụng như phép chiếu trên một tập các bộ. Ví dụ, xét truy vấn "Cho biết tên và tuổi của các sinh viên", ta có:

$$set\_apply_{\lambda s.\pi_{(hoten,tuoi)}(\pi_V(s))}(\text{SinhVien})$$

4.3.4. Phép toán trên kiểu "túi"

Các phép toán trên kiểu dữ liệu "túi" gồm có: thiết lập "túi", hợp, hiệu, chọn, làm phẳng, áp dụng hàm trên "túi": **bag**, **bag_union**, **bag_diff**, $\sigma^b_{\lambda s.f}$, **bag_flat**, **bag_apply** và chuyển đổi một "túi" về tập: **bagtoset** (chuyển đổi một "túi" về tập hợp bằng cách loại bỏ các trùng lặp trong "túi").

4.3.5. Phép toán trên danh sách

Thiết lập danh sách, lấy phần tử đầu tiên, lấy phần tử cuối cùng, ghép danh sách, chọn, làm phẳng, áp dụng hàm là: **list**, **first**, **last**, **list_cat**, $\sigma^l_{\lambda s.f}$, **list_flat**, **list_apply**$_{\lambda s.e}$.

4.3.6. Phép toán trên mảng

- Thiết lập mảng, ghép mảng, áp dụng: **array**, **array_cat**, **array_apply**$_{\lambda s.e}$.

- Trích xuất phần tử (π_i): Phép toán này trả về phần tử thứ *i* trong mảng đã cho.

- Chiếu mảng ($\pi_{i,j}$), $j > i$: Phép toán trả về một mảng con chứa các phần tử có chỉ số từ *i* đến *j* của mảng đã cho.

4.3.7. Các ví dụ áp dụng

(i) Cho biết tên của các sinh viên chưa tốt nghiệp có điểm trung bình lớn hơn 4.0.

$$set\_apply_{\lambda t.\pi_{hoten}(\pi_V(t))}(\sigma_{\lambda s.\pi_{dtb}(\pi_V(s))>4}(\text{SVChuaTN}))$$

(ii) Tìm tên và luận văn của sinh viên đã tốt nghiệp, mà có tuổi lớn hơn Giáo sư hướng dẫn.

$$set\_apply_{\lambda s.\pi_{hoten}(\pi_V(s))}$$

$$(\sigma_{\lambda t.\pi_{hoten}(\pi_V(\pi_D(\pi_{truong\_khoa}(\pi_V(\pi_D(\pi_{Dailloc}(\pi_V(t))))))))='Thanh'}(\text{SVChuaTN}))$$

```
select  tuple(s.hoten, s.luanvan)
from  SinhVienTN as s
where  s.tuoi > s.gvhd.tuoi
```

Biểu thức đại số tương đương với truy vấn trên được viết như sau:

$$set\_apply_{\lambda t.\pi_{(hoten,luanvan)}(\pi_V(t))}$$

$$(\sigma_{\lambda s.\pi_{tuoi}(\pi_V(s))>\pi_{tuoi}(\pi_V(\pi_D(\pi_{gvhd}(\pi_V(s))))))}(\text{SinhVienTN}))$$

(iii) Tìm tên các sinh viên chưa tốt nghiệp.

$$apply_{\lambda t.\pi_{hoten}(\pi_V(t))}(apply_{\lambda u.\pi_D(u)}(apply_{\lambda s.\pi_{@}(s)}(\text{SinhVien})$$

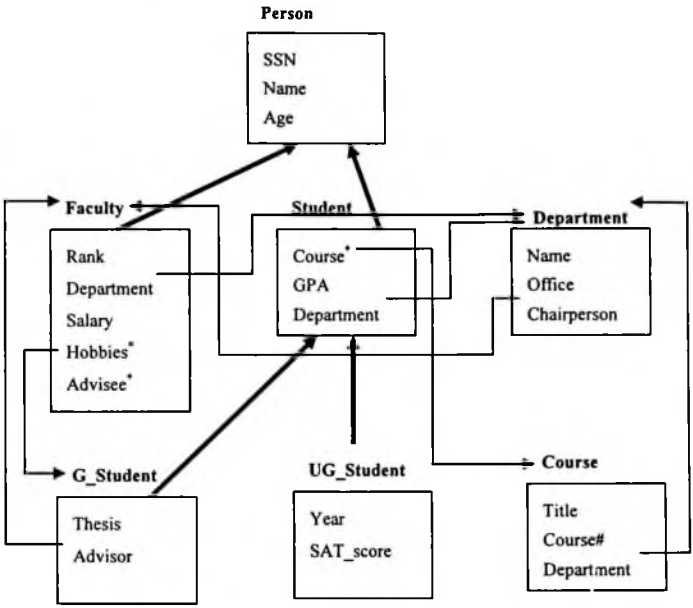
$$set\_diff \quad apply_{\lambda g.\pi_o(g)}(\text{SinhVienTN})))$$

KẾT LUẬN

Trong một số mô hình dữ liệu hướng đối tượng, tập các phép toán đại số đối tượng không đưa ra các ánh xạ một cách đầy đủ từ các truy vấn đối tượng đến các biểu thức đại số hoặc thiếu hỗ trợ cho tiến trình tối ưu hóa truy vấn. Chẳng hạn, đại số EXCESS không đưa ra các cấu trúc và phép toán trên kiểu sưu tập và chỉ cung cấp những phép toán về mảng, nhưng đa số các phép toán trên mảng khó có thể thực hiện việc tối ưu trong xử lý truy vấn, vì chúng không thể sắp xếp lại các phần tử của mảng với các toán tử tập truyền thống như phép kết nối, chiếu và chọn. Những hạn chế này làm cho một số đại số đối tượng không thích hợp cho tiến trình tối ưu hóa các truy vấn đối tượng bằng các phép biến đổi đại số.

CÂU HỎI ÔN TẬP CHƯƠNG 4

Câu 4.1. Cho lược đồ CSDL hướng đối tượng như sau:



1) Biểu diễn các truy vấn sau trong OQL:

a) Tìm tên của các giảng viên (thuộc lớp Faculty) có tuổi đời nhỏ hơn 35 và làm việc tại trường (lớp Department) Đại học CNTT.

b) Tìm tên các sinh viên đã tốt nghiệp có điểm trung bình (GPA) lớn hơn 3.5 và chỉ đăng ký học các khóa đào tạo (Course) tại trường Đại học CNTT.

c) Tìm tên của các sinh viên chưa tốt nghiệp đã đăng ký học các học phần "Hệ thống CSDL" và "Phục hồi thông tin".

2) Sử dụng đại số đối tượng, biểu diễn các truy vấn sau:

a) Tìm tên các giảng viên có tuổi đời nhỏ hơn 35.

b) Tìm tên các khóa học của khoa có Trưởng khoa tên là "Thanh".

c) Tìm tên các sinh viên chưa tốt nghiệp có điểm trung bình lớn hơn 4.

d) Tìm mã số và danh sách sinh viên hướng dẫn của các giảng viên trường Đại học CNTT.

Câu 4.2. Chứng minh 2 luật biến đổi sau:

1) Tính giao hoán của phép *apply* và phép chọn: Nếu biểu thức điều kiện chỉ chứa các thuộc tính được kết xuất bởi phép toán *apply* thì:

$$\text{apply}_{\lambda s.e}(\sigma_{\lambda t.f}(S)) = \sigma_{\lambda t.f}(\text{apply}_{\lambda s.e}(S))$$

2) Giao hoán giữa phép *flatten* và *apply*: Giả sử S là thể hiện của một lớp và CSA là thuộc tính phức kiểu tập thì:

$$flat(apply_{\lambda s.(apply_{\lambda t.e}(\pi_{CSA}(\pi_V(S))))}(S)) = apply_{\lambda t.e}(flat(apply_{\lambda s.\pi_{CSA}(\pi_V(S))}(S)))$$

Câu 4.3. Cho lược đồ CSDL VienDH (Viện Đại học) trong Hình 4.2.

1) Bổ sung một thuộc tính *so_sv* (số lượng sinh viên) vào lớp Khoa với miền là int. Sử dụng ngôn ngữ truy vấn OQL và đại số đối tượng, biểu diễn các truy vấn sau:

a) Trường đại học nào có khoa "Công nghệ thông tin".

b) Tổng số sinh viên chuyên ngành "Công nghệ thông tin" học ở tất cả các trường đại học.

c) Tìm tên khoa có số sinh viên lớn nhất.

2) Cho biểu thức đại số đối tượng sau:

$$set\_apply_{\lambda t.\pi_{(hoten,luanvan)}(\pi_V(t))}(\sigma_{\lambda s.\pi_{tuoi}(\pi_V(s))>\pi_{tuoi}(\pi_V(\pi_D(\pi_{gvhd}(\pi_V(s)))))}(\text{SinhVienTN}))$$

Hãy biểu diễn biểu thức đại số đối tượng trên cây phân tích cú pháp.

3) Sử dụng lược đồ trên, biểu diễn các truy vấn sau trong OQL:

a) Tìm tên của các giảng viên (thuộc lớp Khoa) có lương lớn hơn 5 và làm việc tại trường (lớp DaiHoc) Đại học CNTT.

b) Tìm tên các sinh viên đã tốt nghiệp có điểm trung bình (*dtb*) lớn hơn 3.5 và chỉ đăng ký học các khóa đào tạo (KhoaHoc) tại trường Đại học CNTT.

c) Tìm mã số, tên và danh sách sinh viên được hướng dẫn bởi các giảng viên thuộc Đại học CNTT.

Câu 4.4. Sử dụng đại số đối tượng, biểu diễn các truy vấn sau:

a) Tìm tên các giảng viên có tuổi đời nhỏ hơn 35.

b) Tìm tên các sinh viên chưa tốt nghiệp có điểm trung bình lớn hơn 4.

c) Tìm mã số, tên và danh sách sinh viên hướng dẫn của các giảng viên trường Đại học CNTT.

Câu 4.5. Bổ sung một thuộc tính *Tro_giang* (giảng viên trợ giảng) vào lớp KhoaHoc với miền là Khoa. Hãy phân tích cú pháp của truy vấn sau trên cây phân tích cú pháp.

```
Select c.ten_khoa_hoc
    From KhoaHoc c
    Where c.DaiHoc.truong_khoa = "Thanh"
        and c.DaiHoc.ten_nganh = "Khoa hoc may tinh"
        and c.tro_giang.DaiHoc.ten_nganh
                                = "Cong nghe thong tin"
```

CHƯƠNG

5

TỐI ƯU HÓA TRUY VẤN ĐỐI TƯỢNG

Khi truy vấn được thực thi, có nhiều phương án để hệ thống CSDL cho phép xử lý nhằm có câu trả lời chính xác. Các phương án đều có kết quả cuối cùng là tương đương nhưng khác nhau trong chi phí thực hiện, tức là tổng thời gian cần để thực hiện một truy vấn. Lựa chọn phương án nào để có tổng thời gian thực hiện truy vấn là nhỏ nhất. Như vậy, vấn đề phải quan tâm là làm sao cực tiểu tần suất sử dụng của CPU, bộ nhớ, chi phí vào/ ra và các nguồn tài nguyên về lĩnh vực truyền thông. Với kỹ thuật phần cứng hiện nay (khả năng của các chip nhớ) việc tối ưu thực thi một truy vấn chỉ còn là vấn đề làm cực tiểu thời gian trả lời của truy vấn, trong khi đó các hệ thống lại chịu sự chi phối chính ở thời gian trao đổi vào/ ra. Do đó, các kỹ thuật tối ưu hóa truy vấn chủ yếu tập trung giải quyết vấn đề cực tiểu chi phí xử lý vào/ ra khi một truy vấn được thực thi.

5

Phương pháp tối ưu hóa truy vấn đối tượng được nghiên cứu và giải quyết dưới góc độ xem xét trên những đặc trưng của mô hình hướng đối tượng. Trong chương 5, chúng ta giới thiệu phương pháp tối ưu hóa truy vấn bằng biến đổi biểu thức đại số đối tượng dựa vào tập luật. Phương pháp tối ưu hóa truy vấn đối tượng được trình bày trên cơ sở của mô hình dữ liệu ODMG và ngôn ngữ truy vấn đối tượng OQL, nhưng không mất tính tổng quát đối với các mô hình dữ liệu hướng đối tượng có hỗ trợ các đặc trưng này.

5.1. MÔ HÌNH ƯỚC LƯỢNG CHI PHÍ XỬ LÝ TRUY VẤN

Để thống nhất trong trình bày các ví dụ trong chương 5, lược đồ đối tượng NhanSu định nghĩa trong OQL được sử dụng minh họa như sau:

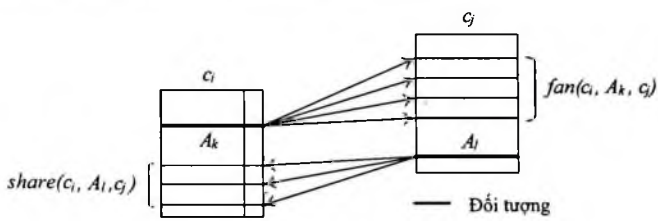
Ví dụ 5.1. Cho lược đồ đối tượng VienDaihoc:

```
class NhanSu
    type tuple (maso: int, hoten: string, pho: string, tpho: string,
                matinh: int, ngaysinh: tuple (ngay: int, thang: int, nam:int))
class SinhVien inherits NhanSu
    type tuple (gvhd: string, dtb: float, hocbong: float, tenkhoa: Khoa)
class GiangVien inherits NhanSu
    type tuple (bomon: string, mabomon: int, chucvu: string,
                tenkhoa: Khoa, luong: int, con: set(NhanSu))
class Khoa
    type tuple (makhoa: int, tenkh: string, diadiem: string,
                ngansach: float, cbgd: set(GiangVien))
```

Chúng ta phân tích mô hình chi phí xử lý truy vấn tổng quát trên các lớp. Mô hình này được sử dụng để ước lượng chi phí xử lý của truy vấn đối tượng trong tiến trình tối ưu hóa truy vấn [18], [42].

Định nghĩa 5.1. *Biểu thức đường dẫn có dạng $c_1.A_2.A_3...A_n.v$, với c_1 là lớp gốc, $A_i \in c_{i-1}$ là đối tượng dựa trên biến thể hiện được xác định trong lớp c_{i-1} (với miền của lớp c_i, $2 \le i \le n$). Biểu thức điều kiện gồm các kết nối ẩn có dạng $c_1.A_2.A_3...A_n.v$ rel_op const, với rel_op là phép toán quan hệ: "=", "<>", ">", "<", ">=" và "<=", const là giá trị hằng trong miền của biến thể hiện dựa trên giá trị v của lớp c_n.*

Định nghĩa 5.2. *Cho hai lớp c_i và c_j trong một đường dẫn P (c_j là miền giá trị của thuộc tính A_k của c_i). Hệ số phân đầu ra, ký hiệu **fan**(c_i, A_k, c_j) là giá trị trung bình của số các đối tượng c_j tham chiếu đến một đối tượng của c_i thông qua thuộc tính A_k. Tương tự, mức chia sẻ, ký hiệu **share**(c_i, A_l, c_k) là giá trị trung bình của số các đối tượng c_i tham chiếu đến cùng đối tượng của c_j thông qua thuộc tính A_l. **FAN**(c_i, c_j) và **SHARE**(c_i, c_j) là giá trị trung bình của fan(c_i, A_k, c_j) và share(c_i, A_l, c_j) trên tất cả các đối tượng của lớp c_j và lớp c_i.*



Hình 5.1. ***fan***(c_i, A_k, c_j) **và** ***share***(c_i, A_k, c_k)

5.1.1. Mô hình chi phí các khối dựng sẵn

Tổng chi phí của tiến trình xử lý truy vấn được tính với công thức:

$$Total\_cost = IO\_cost + CPU\_cost$$

Trong đó, *IO_cost* là chi phí vào/ ra của đĩa và *CPU_cost* là chi phí tính toán trong tiến trình xử lý truy vấn. Người ta tập trung nghiên cứu trên chi phí *IO_cost* và bỏ qua chi phí *CPU_cost*. Điều này sở dĩ như vậy là vì mỗi ứng dụng CSDL lớn với số lượng khổng lồ các truy xuất dữ liệu thì sự tham gia của chi phí *CPU_cost* vào tổng chi phí *Total_cost* sẽ không có ý nghĩa.

Bảng 5.1. Bảng các tham số dùng trong mô hình chi phí

Phạm vi	Tham số	Ý nghĩa
CSDL	$\|\|c_{i,k}\|\|$	Lực lượng của lớp $c_{i,k}$ (tức là lớp con thứ k của lớp thứ i theo phân cấp lớp hợp thành).
	$\|c_{i,k}\|$	Số các trang sử dụng cho lớp $c_{i,k}$.
	$SC_{i,k}$	Kích thước của đối tượng (tính bằng byte) của lớp $c_{i,k}$.
	q_i	Số các lớp con trong phân cấp lớp con có gốc là lớp c_i.
	$fan_{i-1,j,i,k}$	Hệ số phân đầu ra (*fan-out*) đối với phân cấp lớp hợp thành từ lớp con thứ j của lớp c_{i-1} với lớp con thứ k của lớp c_i.
	l_p	Chiều dài đường dẫn của biểu thức đường dẫn, nghĩa là, số các lớp thuộc biểu thức đường dẫn trong phân cấp lớp hợp thành.
	$NP_{i,k}$	Số các đối tượng (của lớp con thứ k của lớp c_i) trên mỗi trang. Nếu $SC_{i,k} < PS$ thì $NP_{i,k} = \left\lfloor \frac{PS}{SC_{i,k}} \right\rfloor$, ngoài ra nó có giá trị là 1.
	b	Hệ số phân đầu ra trung bình *fan* của chỉ mục B^+-cây.
	PS	Kích thước trang của hệ thống file (đơn vị là byte).
Truy vấn	$ref_{i,k}$	Số các tham chiếu đối tượng đối với lớp con thứ k trong lớp c_i trong việc định giá biểu thức đường dẫn theo phân cấp lớp hợp thành.
	SEL_i	Số các đối tượng được chọn của truy vấn theo tân từ trên lớp c_i.
	$SCout_{j,i,k}$	Kích thước kết quả ra trong phạm vi phân mảnh thứ j của lớp con thứ k của lớp c_i.

Mô hình chi phí được xây dựng với các tham số được chia thành hai nhóm trong bảng 5.1, gồm các tham số hệ thống CSDL và tham số truy vấn.

Chi phí xử lý truy vấn được ước lượng theo kích thước của các đối tượng trong các biến thể hiện lớp, mỗi đối tượng có thể được lưu trữ trên một hoặc nhiều hơn một trang bộ nhớ. Người ta phân tích mô hình chi phí dựa trên sự khác biệt về dung lượng của bộ nhớ chính trong tiến trình xử lý truy vấn, dung lượng của bộ nhớ chính sẽ ảnh hưởng đến số lần truy xuất đĩa:

(1) Giả thiết bộ nhớ đủ lớn: Kích cỡ bộ nhớ chính là khá lớn có đủ các vùng nhớ đệm để nạp các đối tượng (các đối tượng được nạp vào từ đĩa và chỉ nạp một lần).

(2) Giả thiết bộ nhớ nhỏ: Kích cỡ bộ nhớ chính nhỏ, ta chỉ có thể cấp một trang bộ nhớ đệm cho mỗi lớp hoặc phân mảnh lớp (các đối tượng hoặc các phân mảnh đối tượng sẽ được nạp vào bộ nhớ chính trong nhiều lần và điều này làm tăng số lần truy xuất đĩa).

5.1.2. Các yếu tố chi phí cơ sở

5.1.2.1. Ước lượng số các trang cho một lớp sưu tập

Tổng số trang lưu trữ cho một lớp *c* với kích thước đối tượng *SC* và lực lượng ||*c*|| được tính với công thức:

$$|c| = \left\lceil \frac{\|c\| \times SC}{PS} \right\rceil \qquad (5.1)$$

Với *PS* là kích thước trang bộ nhớ dùng cho hệ thống CSDL hướng đối tượng. Người ta có thể áp dụng công thức (5.1) với phân cấp lớp con, với giả sử các đối tượng của lớp (lớp con) được lưu trữ liên tục, nhưng giữa các lớp (lớp con) khác nhau thì các đối tượng của mỗi lớp được lưu trữ tách rời nhau.

5.1.2.2. Ước lượng số trang truy xuất để chọn các bản ghi một cách ngẫu nhiên

Định lý 5.1. [44] (Định lý Yao) *Cho n bản ghi cùng kiểu được nhóm vào m trang (1 < m ≤ n), mỗi trang chứa n/ m bản ghi. Nếu k bản ghi*

($k \le n - n/m$) được chọn một cách ngẫu nhiên từ n bản ghi, thì số các trang truy xuất là:

$$Yao(n,m,k) = m \times \left[1 - \prod_{i=1}^{k} \frac{nd - i + 1}{n - i + 1}\right], \text{ với } d = 1 - \frac{1}{m} \qquad (5.2)$$

Số trang truy xuất là nhỏ hơn *k* bởi vì một số trang có thể chứa hai hoặc nhiều hơn các bản ghi kết quả. Áp dụng hàm Yao vào CSDL hướng đối tượng, ta có giá trị của các tham số *n*, *m*, *k* là: $n = \|c\|, m = |c|$ và $k = SEL \times \|c\|$ (*SEL* là số các đối tượng được chọn thoả điều kiện trên lớp hiện thời *c*), lúc đó công thức (5.2) trở thành:

$$Yao(\|c\|, |c|, SEL \times \|c\|) = |c| \times \left[1 - \prod_{i=1}^{k} \frac{\|c\| \times d - i + 1}{\|c\| - i + 1}\right], \qquad (5.3)$$

Với $d = 1 - \frac{1}{|c|}$

Ngoài ra, nếu *c* là một sưu tập gộp nhóm công thức (5.2) không thể sử dụng để ước lượng các trang truy xuất, vì sưu tập gộp nhóm *c* có nhiều hơn một phân mảnh, các đối tượng được gộp nhóm với nhau từ các sưu tập khác nhau, trong đó vẫn tồn tại một số đối tượng được lưu trữ riêng lẻ. Vì vậy, mật độ của các đối tượng trong các phân mảnh khác nhau là không bằng nhau. Trong trường hợp này người ta sử dụng công thức ước lượng số trang truy xuất trong định lý 5.2 [18].

Định lý 5.2. [18] (Định lý Yao') *Cho sưu tập c có p phân mảnh, mỗi phân mảnh có $||c_i||$ đối tượng. Nếu k đối tượng* ($k < \sum_{i=1}^{p} \|c_i\|$) *được chọn từ c một cách ngẫu nhiên thì số các trang được truy xuất là:*

$$Yao'(c,k) = \sum_{i=1}^{p} Yao\left(\|c_i\|, |c_i|, k_i\right) \qquad (5.4)$$

Nếu số các đối tượng được chọn lấy giống nhau giữa các phân mảnh thì $k_i = \frac{\|c_i\|}{\|c\|} \times k$. Ngược lại, nếu số các đối tượng chọn không giống nhau giữa các phân mảnh khác nhau thì phải xác định chính xác

giá trị của mỗi k_i trên mỗi phân mảnh dựa vào mật độ của các đối tượng trong mỗi phân mảnh. Công thức (5.2) là một trường hợp đặc biệt của (5.4) khi sưu tập c chỉ có một phân mảnh.

Mặt khác, công thức (5.2) chỉ áp dụng được khi $m \le n$, tức là khi kích thước đối tượng là nhỏ hơn hoặc bằng kích thước của trang. Đối với kích thước đối tượng lớn hơn kích thước trang, số các trang được ước lượng bằng: $\sum_{i=1}^{p} |c_i| \times \frac{k}{\sum_{i=1}^{p} \|c_i\|}$.

Vì vậy, hàm Y được định nghĩa trong trường hợp này như sau:

$$Y(c,k) = \begin{cases} \sum_{i=1}^{p} Yao\left(\|c_i\|, |c_i|, k_i\right) & \text{với kích thước đối tượng} \le \text{kích thước trang} \\ \sum_{i=1}^{p} |c_i| \times \frac{k}{\sum_{i=1}^{p} \|c_i\|} & \text{với kích thước đối tượng} > \text{kích thước trang} \end{cases} \quad (5.5)$$

Như vậy, với tính chất (5.5) và công thức (5.4), định lý 5.2 được phát biểu lại như sau:

Định lý 5.3. [36] *Cho sưu tập c có p phân mảnh, mỗi phân mảnh có* $||c_i||$ *đối tượng. Nếu k đối tượng (*$k < \sum_{i=1}^{p} \|c_i\|$*) được chọn từ c một cách ngẫu nhiên thì số các trang được truy xuất là:* $Yao'(c,k) = Y(c,k)$, k_i *là số các đối tượng được chọn trong phân mảnh* c_i.

5.1.2.3. Ước lượng số các trang truy xuất với chỉ mục tìm kiếm

Nếu tân từ trong truy vấn có chứa một biến thể hiện liên kết với một chỉ mục, người ta sử dụng chỉ mục này để nạp các đối tượng của lớp gốc. Với chỉ mục gộp nhóm B^+ - cây và giá trị trung bình hệ số phân đầu ra b (*FAN*) thì số các trang truy xuất là $\log_b\left(SEL \times \sum_{k=0}^{q_1} \frac{\|c_{1,k}\|}{NP_{1,k}}\right)$ để tìm kiếm

với chỉ mục gộp nhóm, *SEL* là số các đối tượng được chọn theo tân từ trên lớp gốc và $NP_{1,k}$ là số các đối tượng của lớp con thứ k của lớp gốc trong mỗi trang. Đối với chỉ mục không gộp nhóm, số các trang truy xuất là $\log_b\left(SEL \times \sum_{k=0}^{q_1} \left\|c_{1,k}\right\|\right)$.

5.1.2.4. Ước lượng các tham chiếu đối tượng

Trong định giá tân từ, ta cần ước lượng các tham chiếu đối tượng (cùng với phân cấp lớp hợp thành). Duyệt tuần tự, $ref_{1,k} = \left\|c_{1,k}\right\|$, $0 \le k \le q_1$ và $ref_{i,k} = \left(\sum_{j=0}^{q_{i-1}} ref_{i-1,j} \times fan_{i-1,i,j,k}\right)$, $1 < i \le n$ và $0 \le k \le q_i$.

Duyệt với chỉ mục gộp nhóm $ref_{1,k} = SEL_1 \times \left\|c_{1,k}\right\|$, $0 \le k \le q_1$, $ref_{2,k} = \left(\sum_{j=0}^{q_1} ref_{1,j} \times fan_{1,2,j,k}\right)$, $0 < k \le q_2$ và $ref_{i,k} = \left(\sum_{j=0}^{q_{i-1}} ref_{i-1,j} \times fan_{i-1,i,j,k}\right) \times SEL_{i-1}$, $2 < i \le n$. Duyệt với chỉ mục không gộp nhóm, $ref_{i,k}$ có công thức giống như duyệt với chỉ mục gộp nhóm.

5.1.3. Chi phí xử lý truy vấn trên các lớp

Mô hình chi phí truy vấn tổng quát được phân tích trên các phân cấp lớp, các phân cấp lớp con khác nhau có số các nút con khác nhau và chúng cũng có số cấp của cây là khác nhau trong mỗi phân cấp lớp. Ký hiệu lớp con thứ k của phân cấp lớp con (gốc là lớp c_i) là $c_{i,k}$, $1 < k \le q_i$ (q_i là tổng số lớp con trong lớp c_i) và lớp gốc c_i là $c_{i,0}$. Mô hình chi phí được xây dựng với ba thành phần: chi phí nạp của lớp sưu tập (*IO_Load*), chi phí ước lượng tân từ (*IO_Eval*) và chi phí kết xuất kết quả (*IO_Build*).

Tổng chi phí *vào/ ra* của việc xử lý truy vấn chứa ba thành phần:

$$Total\_IO = IO\_Load + IO\_Eval + IO\_Build \quad (5.6)$$

Với *IO_Load* là số các trang truy xuất để nạp các đối tượng thuộc lớp gốc, là lớp bắt đầu duyệt theo biểu thức đường dẫn, *IO_Eval* là số trang truy xuất để "đi qua" biểu thức đường dẫn theo phân cấp lớp hợp thành và *IO_Build* là số trang truy xuất để tạo ra kết quả. Để duyệt lớp bắt đầu theo biểu thức đường dẫn, trước hết người ta sẽ nạp các đối tượng trong lớp gốc. Có hai phương án duyệt các đối tượng: (a) duyệt tuần tự; (b) duyệt theo chỉ mục. Hai phương án này có sự khác biệt về công thức chi phí đối với thành phần *IO_Load*. Nhưng các phương án duyệt để nạp các đối tượng thuộc lớp gốc không làm ảnh hưởng đến các chi phí duyệt đường dẫn trong các lớp khác. Vì vậy, chi phí *IO_Eval* và *IO_Build* cũng không ảnh hưởng gì khi chọn một trong hai phương án trên.

5.1.3.1. Trường hợp bộ nhớ đủ lớn

a) IO_Load

(i) Duyệt tuần tự: Hầu hết các lớp gốc được duyệt tuần tự, số các trang truy xuất để nạp các đối tượng thuộc lớp gốc được duyệt tuần tự là: $\sum_{k=0}^{q_1} \left| c_{1,k} \right|$, q_1 là số các lớp con trong phân cấp lớp gốc là $c_{1,0}$.

(ii) Duyệt với chỉ mục: Nếu tân từ trong truy vấn chứa biến thể hiện liên kết với một chỉ mục, người ta sử dụng chỉ mục này để nạp các đối tượng lớp gốc.

Nếu chỉ mục gộp nhóm và b là giá trị trung bình của hệ số phân đầu ra *fan-out*:

$$IO\_Load = \log_b \left(SEL_1 \times \sum_{k=0}^{q_1} \frac{\left\| c_{1,k} \right\|}{NP_{1,k}} \right) + \sum_{k=0}^{q_1} Y\left(\left\| c_{1,k} \right\|, \left| c_{1,k} \right|, ref_{1,k} \right) \quad (5.7)$$

Với SEL_1 là kết quả chọn theo tân từ trên lớp gốc, $\log_b\left(SEL_1 \times \sum_{k=0}^{q_1} \frac{\|c_{1,k}\|}{NP_{1,k}}\right)$ là số các trang truy xuất để tìm kiếm với chỉ mục gộp nhóm và $\sum_{k=0}^{q_1} Y\left(\|c_{1,k}\|, |c_{1,k}|, ref_{1,k}\right)$ là số các trang truy xuất để nạp các đối tượng suy dẫn từ chỉ mục tìm kiếm (tổng trên q_1 lớp con).

Nếu chỉ mục không gộp nhóm:

$$IO\_Load = \log_b\left(SEL_1 \times \sum_{k=0}^{q_1} \|c_{1,k}\|\right) + SEL_1 \times \sum_{k=0}^{q_1}\left(\|c_{1,k}\| \times \left\lceil \frac{SC_{1,k}}{PS} \right\rceil\right) \quad (5.8)$$

Trong đó, $\log_b\left(SEL_1 \times \sum_{k=0}^{q_1} \|c_{1,k}\|\right)$ là số các trang sử dụng để tìm kiếm với chỉ mục không gộp nhóm và $SEL_1 * \sum_{k=0}^{q_1}\left(\|c_{1,k}\| \times \left\lceil \frac{SC_{1,k}}{PS} \right\rceil\right)$ là số các trang dùng nạp các đối tượng suy dẫn từ chỉ mục tìm kiếm. Công thức (5.8) là tổng trên tất cả các lớp con, nhưng không giống trong trường hợp chỉ mục gộp nhóm, nó không chứa hàm **Y** (5.5). Bởi vì với chỉ mục không gộp nhóm, các đối tượng dữ liệu không được sắp thứ tự trong các trang dữ liệu.

b) IO_EVAL

Sử dụng hàm **Y** (5.5) để ước lượng số các trang truy xuất đối với một tập (lớp) theo một tân từ thuộc biểu thức đường dẫn là:

$$IO\_Eval = \sum_{i=2}^{n}\left[\sum_{k=0}^{q_i} Y\left(\|c_{i,k}\|, |c_{i,k}|, ref_{i,k}\right)\right] \quad (5.9)$$

Tổng ngoài là tổng trên các lớp thuộc phân cấp lớp hợp thành, với chỉ số từ 2 đến *n*, vì khi duyệt lớp gốc chi phí này đã được tính trong thành phần *IO_Load*. Tổng trong là tổng trên các lớp con khác nhau trong phân cấp lớp con gốc là lớp $c_{i,0}$, $ref_{i,k}$ là số các tham chiếu đối tượng trong lớp $c_{i,k}$ với ước lượng tân từ.

c) IO_Build

Số các trang truy xuất để kết xuất kết quả là:

$$IO\_Build = \sum_{i=1}^{n}\left[\sum_{k=0}^{q_i}\frac{SEL_i \times \|c_{i,k}\| \times SCout_{j,i,k}}{PS}\right] \quad (5.10)$$

Tương tự công thức tính cho *IO_Eval*, tổng ngoài là tổng trên các lớp thuộc phân cấp lớp hợp thành, nhưng trong trường hợp này chỉ số bắt đầu từ 1 đến *n*. Tổng trong là tổng trên các lớp con khác nhau trong phân cấp lớp con có gốc là lớp $c_{i,0}$. Số trang yêu cầu để kết xuất kết quả trên lớp $c_{i,k}$ là $\frac{SEL_i \times \|c_{i,k}\| \times SCout_{j,i,k}}{PS}$.

5.1.3.2. Trường hợp bộ nhớ nhỏ

Trong trường hợp này, công thức của *IO_Load* với duyệt tuần tự, *IO_Load* của chỉ mục không gộp nhóm và *IO_Build* là tương tự với các trường hợp tương ứng trong trường hợp bộ nhớ lớn. Nếu chỉ có một trang trong bộ đệm đối với mỗi lớp, thì việc duyệt với chỉ mục gộp nhóm giảm so với duyệt chỉ mục không gộp nhóm:

$$IO\_Load_{\text{(chỉ mục gộp nhóm)}} = IO\_Load_{\text{(chỉ mục không gộp nhóm)}}$$

$$= IO\_Load_{\text{(chỉ mục không gộp nhóm trong trường hợp bộ nhớ đủ lớn)}}$$

$$= \log_b\left(SEL_1 \times \sum_{k=0}^{q_1}\|c_{1,k}\|\right) + SEL_1 \times \sum_{k=0}^{q_1}\left(\|c_{1,k}\| \times \left\lceil\frac{SC_{1,k}}{PS}\right\rceil\right) \quad (5.11)$$

IO_Eval trong trường hợp bộ nhớ nhỏ khác biệt so với trường hợp bộ nhớ đủ lớn:

$$IO\_Eval = \sum_{i=2}^{n}\left[\sum_{k=0}^{q_i}\left[\left(\prod_{r=2}^{i} ref_{r,k}\right) \times \left\lceil\frac{SC_{i,k}}{PS}\right\rceil\right]\right] \quad (5.12)$$

Chi phí ước lượng tân từ là tổng các trang truy xuất đối với các lớp thuộc biểu thức đường dẫn. Trong lớp thứ *i* thuộc biểu thức đường

dẫn, số trang cần truy xuất là $\sum_{k=0}^{q_i}\left[\left(\prod_{r=2}^{i} ref_{r,k}\right)\times\left\lceil\frac{SC_{i,k}}{PS}\right\rceil\right]$. Vì vậy, trong trường hợp này số các trang truy xuất để nạp vào bộ nhớ chính nhiều hơn trường hợp bộ nhớ lớn.

5.1.4. Thiết lập các tham số trong mô hình chi phí

Trong mô hình chi phí xử lý truy vấn tổng quát, các tham số cơ sở được áp dụng để ước lượng và so sánh đồng nhất chi phí xử lý truy vấn. Người ta sử dụng các giả thiết trong thiết lập tham số cho mô hình ước lượng chi phí xử lý truy vấn tổng quát như sau:

(a) Lực lượng: Đối với lớp c_i, người ta giả sử rằng tất cả các lớp con c_i có cùng lực lượng như c_i, nhưng các lớp c_j khác có lực lượng khác lớp c_i.

(b) Kích thước của đối tượng: Trong mỗi bước ước lượng, các đối tượng trong tất cả các lớp/ lớp con là có cùng kích thước.

(c) Kích thước của các phân mảnh trong lớp sưu tập (tức là, tổng kích thước các biến thể hiện trong phân mảnh): Trong mỗi bước ước lượng, người ta giả sử kích thước của các phân mảnh là giống nhau đối với tất cả các lớp/ lớp con, nghĩa là chúng phân chia các lớp vào trong các phân mảnh có kích thước bằng nhau. Trong thực tế, phần lớn các hệ thống không cài đặt như giả thiết nhưng giả thiết này sẽ làm rõ hơn sự so sánh đồng nhất và tính hiệu quả của sự khác biệt về tham số của kích thước phân mảnh trong lớp sưu tập.

(d) Số các phân mảnh: Trong mỗi bước ước lượng, số các phân mảnh là giống nhau cho các lớp/ lớp con.

(đ) Số các lớp con trong phân cấp lớp ISA: Người ta lấy số các lớp con q_i có gốc tại lớp c_i tối đa là 3, $1 \le i \le 3$, như vậy có tối đa 4 lớp trong mỗi phân cấp lớp bao gồm cả lớp gốc.

(e) Hệ số phân đầu ra: $fan_{i-1,i,j,k} = 0$ với $j \ne k$ và $2 \le i \le n$. Các hệ số phân đầu ra $fan_{i-1,i,j,k}$ là giống nhau, $2 \le i \le n$ và $0 \le k \le q_i$.

Nếu định nghĩa tổng lực lượng của phân cấp lớp ISA là $\|c_i\_ISA\| = \sum_{k=0}^{q_i} \|c_{i,k}\|$ và lấy $fan_{i-1,i,j,k} = FAN$ thì $\|c_2\_ISA\| = \|c_1\_ISA\| \times FAN$ và $\|c_3\_ISA\| = \|c_2\_ISA\| \times FAN$

(g) Danh sách kết quả trong truy vấn đối tượng: Giả thiết danh sách kết quả thu được từ phép chiếu trên các biến thể hiện chỉ thực hiện từ lớp gốc c_1.

(h) Hệ số chọn: Kết quả sẽ thu được từ lớp gốc, vì vậy người ta tập trung thiết lập hệ số lựa chọn trên lớp gốc và với các lớp khác hệ số lựa chọn được thiết lập là 1.

5.2. BIÊN DỊCH CÁC TRUY VẤN ĐỐI TƯỢNG OQL THÀNH CÁC TRUY VẤN QUAN HỆ SQL

Vấn đề chuyển đổi truy vấn quan hệ SQL sang truy vấn đối tượng OQL đã được Fong J. [16] và Meng W. [26] đề xuất trên cơ sở của quá trình chuyển đổi CSDL quan hệ sang CSDL hướng đối tượng. Các kết quả này mang lại một hệ thống chuyển đổi khá hoàn thiện cho các CSDL quan hệ đã tồn tại sang các hệ thống CSDL hướng đối tượng hoặc các hệ thống đối tượng - quan hệ. Chúng ta sẽ xem xét quá trình biên dịch các truy vấn đối tượng OQL thành truy vấn quan hệ SQL trên cơ sở các lược đồ quan hệ nhúng, chuyển đổi các mệnh đề select, from và mệnh đề where của truy vấn OQL thành truy vấn quan hệ. Trong quá trình chuyển đổi, chúng ta sẽ sử dụng khái niệm đồ thị tân từ để biểu diễn và thực hiện việc chuyển đổi trên đồ thị tân từ cho các thành phần của mệnh đề where [45].

Lược đồ đối tượng S được biểu diễn hình thức là $S = (s_1, \ldots, s_n)$, s_i là các lớp trong S và truy vấn đối tượng $QE = (s_1, \ldots, s_m, Res, p_1, \ldots, p_k)$, với $s_i \in S$ $(i = 1,\ldots, m)$ là các lớp tham gia truy vấn, Res là lớp/ kiểu kết quả của truy vấn và p_j $(j = 1, \ldots, k)$ là các biểu thức điều kiện ở mệnh đề where.

5.2.1. Biên dịch mệnh đề select trong OQL sang mệnh đề select trong SQL

Trong mệnh đề select của truy vấn OQL cho phép xây dựng các cấu trúc xuất của các đối tượng chứa các thuộc tính đa trị. Do đó, nếu các thuộc tính là đơn trị thì chúng được chuyển đổi bằng các thuộc tính tương ứng trong mệnh đề select quan hệ. Nếu các thuộc tính là đa trị thì khởi tạo một quan hệ con với các thuộc tính của nó, sau đó bổ sung quan hệ này vào mệnh đề from và một kết nối vào mệnh đề where trong truy vấn SQL.

Thuật toán 5.1: Biên dịch mệnh đề select từ OQL sang SQL.

Vào: Mệnh đề select của truy vấn đối tượng (*Res*)

Ra: Truy vấn quan hệ SQL.

Phương pháp:

//*Res* là lớp kết quả của truy vấn; *r* là quan hệ được chuyển đổi tương ứng từ *Res*

(1) **for** mỗi thuộc tính kết quả $a \in Res$ **do**

(2) **if** *a* là định danh đối tượng **then**

(3) *a* được thay bằng **#***r* // *a* được xem như khóa của quan hệ *r*

(4) **else if** *a* là thuộc tính đơn trị **then**

(5) *c.a* được thay bằng *r.a*

(6) **if** *a* là thuộc tính tập **then**

(7) - Khởi tạo quan hệ *s* với thuộc tính *a* // *#s* là khóa của *s*

(8) - Bổ sung khóa liên kết *#s* vào *r*

(9) - Bổ sung *s* vào mệnh đề from của truy vấn quan hệ (nếu nó chưa tồn tại)

(10) - Bổ sung *r.#r* = *s.#s* vào mệnh đề where của SQL

5.2.2. Biên dịch mệnh đề from của truy vấn OQL sang mệnh đề from trong SQL

Việc chuyển đổi mệnh đề from của OQL, chính là chuyển đổi các lớp tham gia truy vấn đối tượng sang các quan hệ ở mệnh đề from của truy vấn quan hệ. Giả sử các lớp có các thuộc tính phức là các thuộc tính có giá trị tập thì quá trình chuyển đổi được thực hiện trên ngữ nghĩa lược đồ CSDL. Đối với lớp có kế thừa từ các siêu lớp, tạo lớp tạm thời gồm các thuộc tính được định nghĩa trong lớp và các thuộc tính kế thừa từ các siêu lớp, sau đó, tạo lập quan hệ tương ứng với lớp tạm thời có chứa khóa liên kết đến siêu lớp.

Thuật toán 5.2: Biên dịch mệnh đề from của OQL sang SQL.

Vào: Truy vấn đối tượng $QE = (s_1, \ldots, s_m, R, p_1, \ldots, p_l)$

Ra: Tập các quan hệ $FR = \{r_1, \ldots, r_k\}$ $(k \geq m)$ tham gia ở mệnh đề from của truy vấn SQL.

Phương pháp:

(1) $FR := \varnothing$ // Chứa tập các quan hệ được chuyển đổi từ mệnh đề from của OQL

(2) $V := (s_1, \ldots, s_m)$

(3) **for** mỗi lớp $s_i \in V$ **do**

(4) **if** (s_i là siêu lớp gốc) **then**

(5) Bổ sung thuộc tính $\#s_i$ vào r_i

(6) Tạo quan hệ r_i có tập các thuộc tính tương ứng của s_i

(7) **else for** (mỗi lớp s_j là siêu lớp của s_i) **do**

(8) Tạo quan hệ con tạm thời r_j có các thuộc tính từ lớp kế thừa s_j

(9) Bổ sung thuộc tính $\#s_j$ vào r_j

(10) Tạo quan hệ r_i từ các quan hệ con r_j

(11) $FR := FR \cup r_i$

Các thuật toán 5.1 và 5.2 dừng sau một số hữu hạn bước, vì tập các thuộc tính và các lớp tham gia trong mệnh đề select, from là hữu hạn. Độ phức tạp tính toán của các thuật toán có thời gian đa thức.

5.2.3. Biên dịch mệnh đề where của truy vấn OQL sang mệnh đề where của truy vấn quan hệ SQL

Quá trình chuyển đổi mệnh đề where OQL sang mệnh đề where SQL được thực hiện qua các bước: Đầu tiên, khởi lập đồ thị tân từ đối tượng từ mệnh đề where của truy vấn đối tượng. Tiếp theo, đồ thị tân từ đối tượng được biên dịch thành đồ thị tân từ quan hệ. Cuối cùng, mệnh đề where của truy vấn quan hệ được tạo lập từ đồ thị tân từ quan hệ.

Trước hết, các khái niệm về đồ thị tân từ đối tượng và đồ thị tân từ quan hệ được trình bày như sau:

Định nghĩa 5.3. *Cho mệnh đề* where *của truy vấn đối tượng trong OQL được ký hiệu là OW. Đồ thị tân từ đối tượng của OW được định nghĩa như sau:*

$$OPG(OW) = OPG(OV, OE1, OE2)$$

Với OV là tập đỉnh, OE1 là tập các cạnh định hướng và OE2 là tập các cạnh vô hướng.

Mỗi đỉnh v trong OV tương ứng với một biến thể hiện lớp của một lớp (có thể có hai hoặc nhiều đỉnh trong đồ thị OPG tương ứng với cùng một lớp). Mỗi đỉnh v trong OV được gán nhãn bằng các biểu thức điều kiện trên các biến thể hiện lớp tương ứng với v. Hơn nữa, đỉnh v được gán nhãn là "LS" nếu nó là biến thể hiện lớp đầu tiên trong vế trái (ký hiệu là VT) của tân từ. Tương tự, đỉnh v được gán nhãn là "RS" nếu nó là biến thể hiện lớp đầu tiên trong vế phải (ký hiệu là VP) của tân từ.

Cạnh định hướng e trong OE1 từ đỉnh v_1 đến đỉnh v_2 được gán nhãn "*Attr*" biểu diễn đường đi từ thuộc tính phức *Attr* của lớp c_1 tương ứng với v_1 đến lớp miền của *Attr* tương ứng với v_2 (một cạnh định hướng biểu diễn kết nối ẩn). Nếu *Attr* tương ứng với thuộc tính

kiểu tập của lớp c_1 và được ước lượng bằng một lượng từ *forall* hoặc *exists*, lượng từ này sẽ là nhãn trên cạnh *e*.

Cạnh vô hướng trong OE2 biểu diễn kết nối hiện giữa vế trái (VT) và vế phải (VP) của một tân từ. Mỗi cạnh vô hướng được gán nhãn bằng phép kết nối hiện giữa các biến thể hiện lớp tương ứng.

Định nghĩa 5.4. *Cho mệnh đề* where *của truy vấn SQL trong mô hình quan hệ được ký hiệu là RW. Đồ thị tân từ quan hệ của RW được định nghĩa:*

$$RG(RW) = RG(RV, RE1, RE2)$$

Với RV là tập đỉnh, RE1 là tập các cạnh định hướng và RE2 là tập các cạnh vô hướng.

Đỉnh *v* trong RV là biến bộ quan hệ và được đánh số n ($n \geq 0$), n là cấp của truy vấn con mà ở đó các biến bộ được định nghĩa. Truy vấn ngoài cùng có cấp bằng không (0). Cạnh định hướng *e* từ đỉnh v_1 đến đỉnh v_2 trong RE1 biểu diễn một phép toán như "in", "not in", v.v... Các phép toán này được gán nhãn trên cạnh. Cạnh vô hướng e_2 giữa đỉnh v_1 và v_2 trong RE2 biểu diễn kết nối hiện.

5.2.3.1. Xây dựng đồ thị tân từ đối tượng từ mệnh đề where của truy vấn OQL

Từ định nghĩa 5.3, các luật tạo lập đồ thị tân từ đối tượng (OPG) từ mệnh đề where của truy vấn OQL được phát biểu như sau:

(**R1**) Biểu thức đường dẫn có dạng $c_1.A_2.A_3...A_n.v$, với c_1 là lớp gốc, $A_i \in c_{i-1}$ là đối tượng dựa trên biến thể hiện được xác định trong lớp c_{i-1} (với miền của lớp c_i, $2 \leq i \leq n$). Biểu thức đường dẫn trong đồ thị tân từ đối tượng được chuyển đổi như sau:

(a) Với mỗi $c_{i-1}.A_i$ ($i = 3, \ldots, n$) được biểu diễn là một đỉnh và được gán nhãn là điều kiện trên A_i. Đỉnh tương ứng với A_2 được gán nhãn là "LS" hoặc "RS" nếu A_2 là điểm bắt đầu của VT hoặc VP của một tân từ.

(b) Tạo lập cạnh định hướng với nhãn là tên của thuộc tính phức A_{i+1} từ đỉnh tương ứng với $c_{i-1}.A_i$ đến đỉnh tương ứng với $c_i.A_{i+1}$, nếu điều kiện trên là A_i chứa lượng từ *exists* (*forall*). Nếu A_i là thuộc tính kiểu tập thì lượng từ *exists* (*forall*) là nhãn trên cạnh định hướng.

(**R2**) Nếu có nhiều biểu thức đường dẫn trong mệnh đề where, thực hiện việc "trộn" các cạnh định hướng biểu diễn các biểu thức đường dẫn như sau: Cho P1 và P2 là các cạnh định hướng của các biểu thức đường dẫn tương ứng EXP1 và EXP2.

(a) Tìm biểu thức con chung dài nhất EXP của EXP1 và EXP2 với cùng biến thể hiện lớp đầu tiên là *v*. Tạo lập cạnh định hướng của EXP và các cạnh định hướng của các biểu thức con còn lại của EXP1, EXP2 sau khi tách biểu thức con chung dài nhất.

(b) Lần lượt trộn hai cạnh định hướng của các biểu thức con vào cạnh định hướng của biểu thức con chung nhất. Nếu có các điều kiện trên các đỉnh tương ứng trong các đường dẫn gốc thì tạo lập điều kiện mới "*<điều kiện>* AND *<điều kiện>*" gán cho các đỉnh đã trộn.

(c) Giữ nguyên các cạnh không thực hiện trộn của P1 và P2.

(**R3**) Với tân từ có cả hai vế VT và VP, kết nối hiện giữa VT và VP được thực hiện trên các thuộc tính nguyên thủy: $...Attr_1$ *<phép so sánh>* $...Attr_2$. Cho v_1 và v_2 là các đỉnh được biên dịch từ các biến thể hiện cuối của VT và VP là O_1 và O_2. Lúc đó, kết nối giữa VT và VP được chuyển đổi thành một cạnh vô hướng giữa đỉnh v_1 và v_2 với nhãn "$O_1.Attr_1$ *<phép so sánh>* $O_2.Attr_2$".

Ví dụ 5.4. Xét lược đồ đối tượng trong ví dụ 4.1, bổ sung các lớp Oto, ThanhPho, CongTy:

```
class Oto
  type tuple (maso_xe: int, kieu_xe: string, mausac: string,
              dong_co: string, noi_sx: string, ten_cty: string)
class ThanhPho
  type tuple (maso_tp: int, ten_tpho: string, thi_truong: string, danso: int)
```

class CongTy

type tuple (maso_cty: int, ten_cty: string, maso_xe: int)

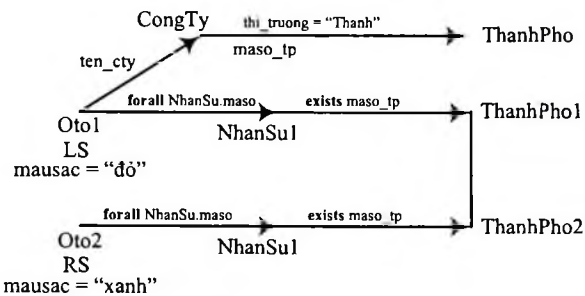
Với các biến thể hiện lớp tương ứng của các lớp là Oto1, Oto2, NhanSu1, NhanSu2, ThanhPho1, ThanhPho2.

Xét truy vấn: Tìm tất cả các xe ôtô có màu đỏ, mà chủ xe có nhà ở thành phố với dân số bằng với dân số của các thành phố mà các chủ xe có ôtô màu xanh và công ty sản xuất loại xe có màu đỏ đặt tại thành phố có Thị trường tên là "Thanh". Truy vấn đối tượng được tạo lập như sau:

select Oto1.maso_xe from Oto1, Oto2

where (Oto1: mausac = "đỏ").(forall NhanSu.maso:).(exists maso_tp: ThanhPho) = (Oto2: mausac = "xanh").(forall NhanSu.maso:).(exists maso_tp: ThanhPho) and (Oto1:).(ten_cty:).(maso_tp: thi_truong = "Thanh")

Đồ thị tân từ đối tượng của mệnh đề where là:



Hình 5.2. Đồ thị tân từ đối tượng của mệnh đề where ở ví dụ 5.4

5.2.3.2. Chuyển đổi đồ thị tân từ đối tượng sang đồ thị tân từ quan hệ

Sử dụng một ngăn xếp Vet1 để lưu vết của các đỉnh đã thăm trong đồ thị tân từ đối tượng và thứ tự chúng được thăm suốt trong quá trình chuyển đổi. Mỗi thực thể trong Vet1 có cấu trúc gồm hai trường: tên của biến bộ quan hệ và cấp của nó. Số của cấp sẽ được khởi tạo và hiệu chỉnh suốt trong quá trình chuyển đổi và dùng để xác định truy vấn

con khi nó lồng trong một truy vấn con khác. Đỉnh được gọi là đỉnh kích hoạt nếu nó là đỉnh liền kề được thăm trong quá trình chuyển đổi. Khởi đầu, không có đỉnh nào được kích hoạt và ngăn xếp Vet1 rỗng.

Thuật toán 5.3: Chuyển đổi đồ thị tân từ OPG đến RPG.

Vào: Đồ thị tân từ đối tượng OPG(OV, OE1, OE2)

Ra: Đồ thị tân từ quan hệ RPG(RV, RE1, RE2)

Phương pháp:

// Bước 1: Chuyển đổi các đỉnh trong OV

(1) **for** mỗi đỉnh *v* trong OV tương ứng với một biến thể hiện lớp *O* **do**

(2) Tạo một biến quan hệ mới $rv(O)$ với quan hệ $r(O)$

(3) Đưa đỉnh ($rv(O)$, 0) vào đồ thị tân từ quan hệ RPG

(4) Sao chép nhãn trên *v* đến đỉnh ($rv(O)$, 0) gồm nhãn "LS" hoặc "RS" nếu có;

// Bước 2: Chuyển đổi các cạnh trong OPG

(5) **repeat**

(6) *Bước 2.1*:

if có một đỉnh kích hoạt trong OPG **then goto** *Bước 2.2*

else if Vet1 $\neq \varnothing$ **then**

Chọn đỉnh đồ thị tân từ OPG là **pop**(Vet1, top)

else Chọn đỉnh bất kỳ và gán nhãn "LS";

(7) *Bước 2.2*: Với mỗi cạnh định hướng e_1 từ đỉnh kích hoạt v_1 đến đỉnh v_2, $e_1 \in$ OE1. Xét ba trường hợp, phụ thuộc vào nhãn trên e_1, ký hiệu là Nhan_canh (trường hợp 4 áp dụng cho cạnh vô hướng *e*).

Trường hợp 1: Nhan_canh = *Attr*, *Attr* là thuộc tính phức, không phải kiểu tập của $c.O_1$ với miền $c.O_2$. Lúc đó, $r(O_1)$ và $r(O_2)$ là các quan hệ tương ứng với các lớp $c.O_1$ và $c.O_2$. Các biến quan hệ $rv(O_1)$ và $rv(O_2)$ được tạo lập từ O_1 và O_2. Trong trường hợp

này, đặt cạnh đầu tiên là cạnh vô hướng giữa đỉnh ($rv(O_1)$, n) và đỉnh ($rv(O_2)$, m), cạnh có nhãn **PKey** là khóa chính của quan hệ $r(O_2)$. Đặt $n = m$. Biểu diễn cạnh như sau:

Trường hợp 2: Nhan_canh = *exists Attr*, *Attr* là thuộc tính kiểu tập của $c.O_1$ với miền $c.O_2$. Trong trường hợp này, tồn tại quan hệ $r(O_{1,2})$ liên kết với $r(O_1)$ và $r(O_2)$ trong lược đồ quan hệ gốc. Với cạnh định hướng e_1 ta thực hiện:

(i) Tạo lập biến quan hệ mới $rv(O_{1,2})$ đối với quan hệ $r(O_{1,2})$;

(ii) Tạo một cạnh vô hướng giữa đỉnh ($rv(O_1)$, n) và đỉnh ($rv(O_{1,2})$, k), cạnh có nhãn **PKey1** là khóa chính của quan hệ $r(O_1)$. Đặt $k := n$;

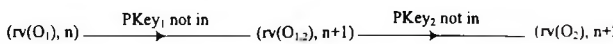
(iii) Tạo một cạnh vô hướng giữa đỉnh ($rv(O_{1,2})$, n) và đỉnh ($rv(O_2)$, m), cạnh có nhãn **PKey2** là khóa chính của quan hệ $r(O_2)$. Đặt $m := n$; Biểu diễn các cạnh như sau:

Trường hợp 3: Nhan_canh = *forall Attr*, *Attr* là thuộc tính kiểu tập của $c.O_1$ với miền $c.O_2$. Tương tự, ta có quan hệ $r(O_{1,2})$ liên kết với $r(O_1)$ và $r(O_2)$ trong lược đồ quan hệ gốc.

(i) Tạo lập biến quan hệ mới $rv(O_{1,2})$ đối với quan hệ $r(O_{1,2})$

(ii) Tạo một cạnh định hướng từ đỉnh ($rv(O_1)$, n) đến đỉnh ($rv(O_{1,2})$, k), cạnh có nhãn **PKey1** *not in*, **PKey1** khóa chính của quan hệ $r(O_1)$. Đặt $k := n + 1$;

(iii) Lấy một cạnh định hướng từ đỉnh ($rv(O_{1,2})$, $n + 1$) đến đỉnh ($rv(O_2)$, m), cạnh có nhãn **PKey2** *not in*, **PKey2** là khóa chính của quan hệ $r(O_2)$. Đặt $m := n + 2$;



Trường hợp 4: Nhan_canh = $O_1.Attr_1$ <phép so sánh> $O_2.Attr_2$, được gán cùng với cạnh vô hướng *e* giữa v_1 và v_2. Với *e* lấy một cạnh vô hướng *e'* giữa ($rv(O_1)$, *n*) và đỉnh ($rv(O_2)$, *m*), có nhãn "$rv(O_1).Attr_1$ *<phép so sánh>* $rv(O_2).Attr_2$". Đối với cạnh vô hướng khác giữa v_1 và v_2 có cùng dạng nhãn, bổ sung nhãn của nó vào nhãn của cạnh *e'*. Biểu diễn cạnh như sau:

$(rv(O_1), n)$ ——— $rv(O_1).Attr_1$ <phép so sánh> $rv(O_2).Attr_2$ ——— $(rv(O_2), m)$

(8) *Bước 2.3*:

(a) Xóa cạnh đã xử lý từ đồ thị tân từ đối tượng OPG.

if (đỉnh v_1 không phân lập) và ($rv(O_1)$ không ở đỉnh của Vet1) **then**

push(Vet1, ($rv(O_1)$, *n*), top);

if (đỉnh v_1 phân lập) và ($rv(O_1)$ ở đỉnh của Vet1) **then**

pop(Vet1, top) /* Lấy ($rv(O_1)$, *n*) ra khỏi ngăn xếp Vet1;

(b) Xóa thiết lập đỉnh kích hoạt ($rv(O_1)$, *n*);

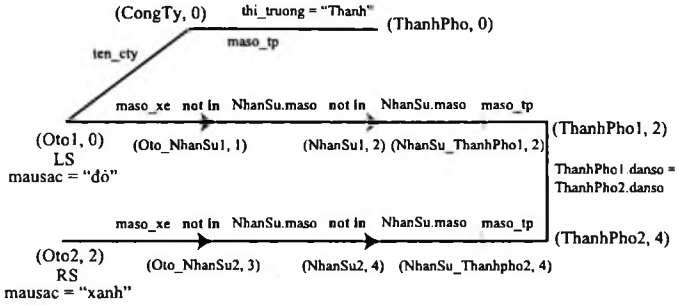
(c) Đối với các trường hợp 1, 2 và 3, nếu đỉnh v_2 không phân lập trong OPG, thiết lập đỉnh v_2 là đỉnh kích hoạt. Trong trường hợp 4, đặt đỉnh *v* có nhãn "RS" và kết nối với v_2 là đỉnh kích hoạt. Gán số cấp của *v* là số cấp của $rv(O_1)$.

(9) **until** (tất cả các cạnh trong đồ thị tân từ đối tượng bị loại bỏ).

Ví dụ 5.5. Xét đồ thị tân từ đối tượng OPG trong hình 5.2. Áp dụng thuật toán 5.5, đồ thị tân từ quan hệ được biểu diễn như sau:

Thực hiện chuyển đổi cạnh từ Oto1 đến NhanSu1. Trong trường hợp này, nhãn cạnh là Nhan_canh = **forall** NhanSu.maso, tương ứng với trường hợp 3 trong thuật toán 5.5. Quan hệ kết hợp giữa NhanSu và Oto là quan hệ Oto_NhanSu trong lược đồ quan hệ. Vì vậy, bổ sung một biến quan hệ mới Oto_NhanSu1 với cấp tương ứng trong đồ thị tân từ quan hệ. Sau đó, tạo cạnh định hướng từ đỉnh Oto1 đến đỉnh Oto_NhanSu1 và cạnh được gán nhãn "maso_xe **not in**", maso_xe là khóa chính của quan hệ Oto. Tương tự, tạo cạnh định hướng từ đỉnh

của Oto_NhanSu1 đến đỉnh NhanSu1 với nhãn "NhanSu.maso **not in**", với NhanSu.maso là khóa chính của quan hệ NhanSu.



Hình 5.3. Chuyển đổi đồ thị tân từ đối tượng thành đồ thị tân từ quan hệ

5.2.3.3. Xây dựng mệnh đề where quan hệ từ đồ thị tân từ quan hệ

Sử dụng một ngăn xếp Vet2, cấu trúc của mỗi phần tử trong ngăn xếp có hai trường: tên của biến quan hệ bộ và cấp của nó. Khởi đầu, không có đỉnh nào được kích hoạt và ngăn xếp Vet2 rỗng.

Thuật toán 5.4: Tạo lập mệnh đề where quan hệ từ đồ thị tân từ quan hệ.

Vào: Đồ thị tân từ quan hệ RG(RV, RE1, RE2)

Ra: Mệnh đề where của truy vấn quan hệ tương ứng.

Phương pháp:

(1) **repeat**

// Bước 1: Chọn một đỉnh kích hoạt

(2) **if** Vet2 $\neq \varnothing$ **then**

Chọn đỉnh kích hoạt là **pop** (Vet2, top)

(3) **else** Chọn đỉnh bất kỳ có nhãn là "LS";

// Bước 2: Xử lý đỉnh kích hoạt

(4) (a) Kiểm tra nếu biến quan hệ bộ tương ứng với đỉnh kích hoạt xuất hiện trong mệnh đề from của truy vấn con hiện thời sâu nhất *SQ* hoặc trong truy vấn con bất kỳ mà *SQ* lồng trong nó. Nếu *SQ* không xuất hiện trong cả hai trường hợp trên, thì bổ sung thành phần "*r*(*rv*) *rt*" vào trong mệnh đề from của *SQ*, *rt* là biến vùng của quan hệ *r*(*rv*).

(b) Nếu tồn tại *k* điều kiện Nhan_canh(*j*) tại đỉnh kích hoạt, *j* = *1*, …, *k* thì *k* tân từ *rt*.Nhan_canh(*j*) được kết nối bằng phép toán "and", bổ sung nó vào mệnh đề where quan hệ và loại bỏ *k* Nhan_canh(*j*);

(c) Nếu đỉnh kích hoạt nối với nhiều hơn một cạnh, thì đẩy đỉnh kích hoạt vào ngăn xếp Vet2;

(d) Nếu đỉnh kích hoạt (*rv*, *n*) có liên kết với cạnh nào đó, **chuyển đến** Bước 3 còn không, nếu nó là đỉnh phân lập ở cuối một truy vấn con, thì bổ sung *q* dấu ngoặc phải ")" vào mệnh đề where quan hệ. Nếu ngăn xếp Vet2 là rỗng thì *q* := *n*, còn không *q* := *n* – *p*, *p* là cấp của phần tử ở đỉnh của ngăn xếp Vet2. Xóa đỉnh kích hoạt (*rv*, *n*) trong đồ thị tân từ quan hệ RG và **quay lại** Bước 1.

// Bước 3: Xử lý cạnh liên kết với đỉnh kích hoạt

(5) Trường hợp 1:

// Cạnh định hướng. Nhãn cạnh có dạng: *Attr* <phép toán>

Giả sử cạnh định hướng nối từ đỉnh kích hoạt (*rv*, *n*) đến đỉnh (*rv1*, *n* + *1*). Lúc đó, bổ sung một truy vấn con vào mệnh đề where của truy vấn con hiện thời.

rv.Attr <phép toán>

(select *rv1.Attr* from *r*(*rv1*) as *rt1* where …)

Trường hợp 2: // Cạnh vô hướng

Giả sử cạnh vô hướng nối từ đỉnh kích hoạt (*rv*, *n*) đến đỉnh (*rv1*, *m*).

- *Trường hợp 2.1*: Nếu $n < m$, có nghĩa rằng biến quan hệ bộ của đỉnh liền kề (*rv1*, *m*) chưa được xác định vì số cấp của đỉnh liền kề lớn hơn cấp của đỉnh kích hoạt. Ta không chuyển đổi cạnh này cho đến khi đỉnh (*rv1*, *m*) được xác định. Lúc đó, đỉnh kích hoạt tiếp theo được chọn từ đỉnh bất kỳ có nhãn "RS" và kết nối với đỉnh (*rv*, *n*) qua đường dẫn trong đồ thị tân từ quan hệ. **Quay lại** Bước 2;

- *Trường hợp 2.2*: $n \geq m$: // có ba khả năng của trường hợp này. Nếu $n > m$, thì (*rv*, *n*) là đỉnh cuối cùng của VP của tân từ đối tượng. Nếu $n = m$ và nhãn có dạng "nhan_canh = *rv.Attr <phép so sánh> rv1.Attr1*" thì (*rv*, *n*) là đỉnh cuối cùng của một VT. Trong trường hợp này, (*rv1*, *m*) không có lượng từ "*forall*". Cuối cùng, nếu $n = m$ và "nhan_canh = *Attr*", cạnh sẽ thuộc trong VT hoặc VP của tân từ đối tượng //.

Khi $n \geq m$, cả hai biến quan hệ bộ tương ứng với đỉnh kích hoạt và đỉnh liền kề với nó được xác định. Lúc đó, nếu nhãn trên cạnh có dạng "*Attr*", bổ sung "*rv.Attr = rv1.Attr*" vào mệnh đề where của truy vấn con hiện thời; còn không, nếu nhãn có dạng "*rv.Attr <phép so sánh> rv1.Attr1*", thì bổ sung các kết nối quan hệ trong nhãn trên cạnh này trong mệnh đề where của truy vấn con hiện thời, nếu có nhiều hơn một tân từ trong truy vấn con thì sử dụng phép toán "and" liên kết giữa các biểu thức.

// Bước 4: Xóa các đỉnh đã chuyển đổi.

(6) Nếu cấp $n \leq m$: Nếu đỉnh kích hoạt là đỉnh phân lập thì xóa đỉnh kích hoạt (*rv*, *n*) trong đồ thị tân từ quan hệ. Đỉnh (*rv1*, *m*) được chọn là đỉnh kích hoạt kế tiếp. **Quay lại** Bước 2.

Nếu $n > m$: (trường hợp này chỉ xuất hiện với kết nối hiện giữa VT và VP, và có ít nhất một lượng từ "*forall*"). Đặt một số dấu ngoặc phải ")" trong mệnh đề where (sử dụng bước 1.(d)); xóa (*rv*, *n*). Nếu (*rv1*, *m*) là đỉnh phân lập, xóa đỉnh khỏi đồ thị tân từ quan hệ. **Quay lại** Bước 1 để chọn đỉnh kích hoạt tiếp theo;

(7) **until** RG = ∅;

5.2.3.4. Ví dụ minh họa

Sử dụng đồ thị tân từ quan hệ trong hình 5.3, thực hiện chuyển đổi sang truy vấn quan hệ khi xử lý tại đỉnh (NhanSu1, 2).

```
select Oto1.maso_xe
from Oto Oto1, CongTy, ThanhPho
where Oto1.mausac = "đỏ" and Oto1.ten_cty = CongTy.ten_cty and
CongTy.ten_tpho = ThanhPho.ten_tpho and ThanhPho.thi_truong =
"Thanh" and Oto1.maso_xe not in
   (select Oto_NhanSu1.maso_xe
   from Oto_NhanSu Oto_NhanSu1
   where Oto_NhanSu1.hoten not in
      (select NhanSu1.hoten
      from NhanSu NhanSu1))
```

Mệnh đề select và from được biên dịch tương ứng với các mệnh đề trong truy vấn quan hệ. Ngăn xếp Vet2 chứa (Oto1, 0) tại đỉnh nếu chọn (Oto1, 0) là đỉnh kích hoạt đầu tiên. Truy vấn quan hệ kết quả của quá trình biên dịch như sau:

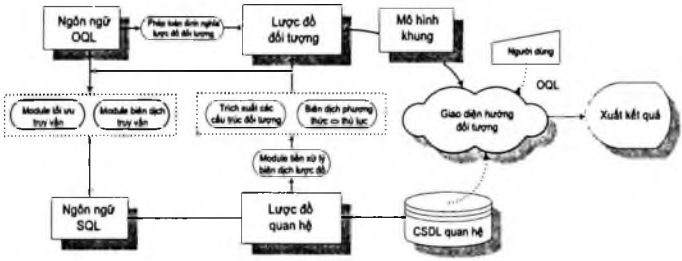
```
select Oto1.maso_xe
from Oto Oto1, CongTy, ThanhPho
where Oto1.mausac = "đỏ" and Oto1.ten_cty = CongTy.ten_cty and
CongTy.ten_tpho = ThanhPho.ten_tpho and ThanhPho.Thi_truong =
"Thanh" and Oto1.maso_xe not in
   (select Oto_NhanSu1.maso_xe
   from Oto_NhanSu Oto_NhanSu1
   where Oto_NhanSu1.hoten not in
      (select NhanSu1.hoten
      from NhanSu NhanSu1, NhanSu_ThanhPho NhanSu_ThanhPho1,
      ThanhPho ThanhPho1, Oto Oto2
      where NhanSu1.hoten = NhanSu_ThanhPho1.hoten and
      NhanSu_ThanhPho1.ten_tpho = ThanhPho1.ten_tpho and
      Oto2.mausac = "xanh" and Oto2.maso_xe not in
```

```
        (select Oto_NhanSu2.maso_xe
        from Oto_NhanSu Oto_NhanSu2
        where Oto_NhanSu2.hoten not in
            (select NhanSu2.hoten
            from NhanSu NhanSu2, NhanSu_ThanhPho
            NhanSu_ThanhPho2, ThanhPho ThanhPho2
            where NhanSu2.hoten = NhanSu_ThanhPho2.hoten and
            NhanSu_ThanhPho2.ten_tpho = ThanhPho2.ten_tpho
            and ThanhPho1.danso = ThanhPho2.danso
))))
```

5.2.4. Tích hợp các modul chuyển đổi lược đồ và biên dịch truy vấn trong hệ thống CSDL đối tượng - quan hệ

Vấn đề xung đột giữa các hệ thống CSDL quan hệ và CSDL hướng đối tượng sẽ có thể được giải quyết bằng việc tích hợp hệ CSDL đối tượng - quan hệ trên cơ sở sử dụng quan hệ nhúng làm nền cho việc cài đặt lược đồ hướng đối tượng. Với cách tiếp cận này, có thể tận dụng được các ưu điểm của cả hai mô hình dữ liệu: tính chặt chẽ, lôgic về mặt toán học và các kỹ thuật CSDL đã xây dựng hoàn chỉnh trên mô hình quan hệ; khả năng linh hoạt, mềm dẻo trong phân tích, thiết kế lược đồ CSDL và sự đặc tả phong phú các thực thể ở thế giới thực trong CSDL hướng đối tượng.

Hình 5.4 là sơ đồ mô tả cho hệ thống CSDL đối tượng - quan hệ. Trong đó, mô hình khung được sử dụng để cài đặt giao diện hướng đối tượng ở mức người dùng, giao diện này không làm thay đổi hệ thống đã tồn tại (mô hình quan hệ) nhưng có thể dễ dàng hiệu chỉnh mô hình với nhiều trường hợp phát sinh khác nhau. Mô hình khung có dạng siêu dữ liệu trong quá trình biên dịch lược đồ và có thể được kiến trúc lại như một giao diện chương trình ứng dụng. Hơn nữa, nó cho phép người thiết kế phối hợp các luật với nhau trong một lớp và kết hợp các lớp trong mô hình khung. Người sử dụng có thể áp dụng các tính năng hướng đối tượng một cách dễ dàng trong hệ thống quan hệ.



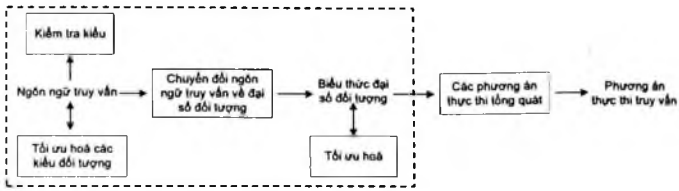
Hình 5.4. Sơ đồ tích hợp của hệ thống CSDL đối tượng - quan hệ

Các thuật toán được cài đặt thành những modul tích hợp trong hệ thống như các modul tiền xử lý lược đồ, trích xuất đặc trưng, biên dịch và tối ưu hóa truy vấn v.v... Các hệ thống này sẽ xử lý cho một lớp các bài toán quản trị CSDL quan hệ khi phát triển mô hình dữ liệu với các đặc trưng mạnh hơn trong mô hình CSDL hướng đối tượng.

Mặt khác, vấn đề tối ưu hóa truy vấn trong các hệ CSDL hướng đối tượng cũng tồn tại những điểm khác biệt đối với các phương pháp truyền thố g trên quan hệ, điều này dễ dàng nhận thấy từ những đặc trưng của mô hình hướng đối tượng. Do đó, các phương pháp tối ưu hóa truy vấn đối tượng sau đây sẽ được phát triển và mở rộng trên cơ sở xử lý các trường hợp riêng biệt cho mô hình CSDL hướng đối tượng.

5.3. TỐI ƯU HÓA TRUY VẤN ĐỐI TƯỢNG BẰNG CÁC PHÉP BIẾN ĐỔI BIỂU THỨC ĐẠI SỐ ĐỐI TƯỢNG

Tiến trình tổng quát tối ưu hóa truy vấn đối tượng dựa trên tập luật được mô tả trong hình 5.5, đầu vào của tiến trình xử lý là các truy vấn được viết bằng ngôn ngữ truy vấn đối tượng, chuyển đổi các truy vấn thành các biểu thức đại số đối tượng tương đương. Sau đó, áp dụng các luật biến đổi trên các phép toán đại số như chọn, chiếu, kết nối với các lớp sưu tập, loại bỏ trùng lặp trong các đa tập, v.v... Cuối cùng, kết quả là phương án thực thi được chọn trong tiến trình tối ưu truy vấn.



Hình 5.5. Tiến trình khung xử lý truy vấn

Các biểu thức đại số có thể được ước lượng với các chi phí xử lý khác nhau. Vì vậy, về mặt lý thuyết người ta mong muốn tìm được các biểu thức đại số tương đương với một truy vấn sao cho có thể đạt được một phương án thực thi hiệu quả hơn. Tuy nhiên, về mặt cài đặt, vì số lượng các truy vấn tương đương quá lớn, trong lúc đó chỉ cần một tập con các truy vấn này mà thôi. Do đó, để tìm ra các truy vấn tương đương khác, người ta cần một tập luật biến đổi bảo toàn tương đương, nhưng mô hình dữ liệu hướng đối tượng lại không có một đại số đối tượng chuẩn áp dụng được cho tất cả các mô hình hướng đối tượng, cho nên sự kỳ vọng để có một tập chuẩn tắc gồm các luật biến đổi bảo toàn tương đương là không tồn tại. Vì vậy, chứng tỏ rằng sự biến đổi bảo toàn tương đương trên cơ sở đại số đối tượng là đúng, với một số luật biến đổi được trình bày sau đây.

5.3.1. Các luật biến đổi đại số đối tượng

Ký hiệu S, S_1, S_2, S_3 là các tập đối tượng; e, f, g, h là các biểu thức đại số, phép toán $op \in \{union, diff\}$. Những luật này chỉ áp dụng trên các phép toán đối tượng, phép toán bộ, phép toán tập hợp và các phép toán đa tập (*bag*). Về mặt ký hiệu, các ký hiệu phép toán được sử dụng một cách hình thức, các phép toán này có thể được cài đặt với một số thay đổi trong các mô hình khác nhau.

L1. *Tính giao hoán của phép chọn*: $\sigma_{\lambda t.g}(\sigma_{\lambda s.f}(S)) = \sigma_{\lambda s.f}(\sigma_{\lambda t.g}(S))$

L2. *Tổ hợp các phép chọn*:

$$\sigma_{\lambda s.(f \wedge g \wedge \ldots \wedge h)}(S) = \sigma_{\lambda s.f}(\sigma_{\lambda t.g}(\ldots(\sigma_{\lambda u.h}(S))\ldots))$$

L3. *Dãy các phép chiếu*:

$$\pi_{(a_1 \ldots a_n)}(\pi_{(b_1 \ldots b_m)}(S)) = \pi_{(a_1 \ldots a_n)}(S), \text{ với } \{a_1, \ldots, a_n\} \subset \{b_1, \ldots, b_m\}$$

L4. *Giao hoán giữa phép chọn và phép chiếu*

$$\sigma_{\lambda s.e}(\pi_{(a_1 \ldots a_n)}(S)) = \pi_{(a_1 \ldots a_n)}(\sigma_{\lambda s.e}(S))$$

L5. *Giao hoán một phép chiếu với phép hợp, hiệu trên tập/ đa tập*

$$\pi_{(a_1 \ldots a_n)}(S_1 \ op \ S_2) \ = \ \pi_{(a_1 \ldots a_n)}(S_1) \ op \ \pi_{(a_1 \ldots a_n)}(S_2)$$

L6. *Phân phối phép chọn với phép hợp và phép hiệu trên tập/ đa tập*

$\sigma_{\lambda s.f}(S_1 \ op \ S_2) = \sigma_{\lambda s.f}(S_1) \ op \ S_2$, nếu f chỉ liên quan với S_1.

Tổng quát: $\sigma_{\lambda s.(f \wedge g \wedge h)}(S_1 \ op \ S_2) = \sigma_{\lambda u.h}(\sigma_{\lambda s.f}(S_1) \ op \ \sigma_{\lambda t.g}(S_2))$, nếu f liên quan S_1, g liên quan S_2 và h liên quan cả S_1 và S_2.

L7. *Giao hoán giữa phép apply và phép chọn:* nếu điều kiện chọn chỉ chứa các thuộc tính do phép toán *apply* trả về thì:

$$apply_{\lambda s.e}(\sigma_{\lambda t.f}(S)) = \sigma_{\lambda t.f}(apply_{\lambda s.e}(S))$$

L8. *Giao hoán giữa phép làm phẳng (flat) và phép apply trên tập/ đa tập*: Giả sử S là thể hiện của một lớp và X là một tập thuộc tính phức của lớp.

$$flat(apply_{\lambda s.(apply_{\lambda t.e}(\pi_{(X)}(\pi_Y(S))))}(S))$$
$$= apply_{\lambda t.e}(flat(apply_{\lambda s.\pi_{(X)}(\pi_Y(S))}(S)))$$

Biểu thức ở vế trái, có biểu thức e tác động trước tập các tập (thu được bởi π_X) sau đó làm phẳng thành một tập; biểu thức ở vế phải có phép toán làm phẳng được tác động trước (kết quả thu được là một tập), sau đó thực hiện phép toán *apply*.

L9. *Tính kết hợp của phép hợp*

$$(S_1 \ union \ S_2) \ union \ S_3 = S_1 \ union \ (S_2 \ union \ S_3)$$

L10. *Các luật kế thừa đối với phép chọn và phép apply:* Nếu S_2 là một lớp con của S_1, thì thể hiện của S_2 là một tập con của thể hiện của S_1:

$$\sigma_{\lambda s.f}(S_1) \textit{ union } \sigma_{\lambda s.f}(S_2) = \sigma_{\lambda s.f}(S_1)$$

$$apply_{\lambda s.e}(S_1) \textit{ union } apply_{\lambda s.e}(S_2) = apply_{\lambda s.e}(S_1)$$

5.3.2. Các quy tắc tối ưu hóa truy vấn đối tượng tổng quát

Tiếp theo, với tập luật biến đổi đại số đối tượng trong mục 5.3.1, các quy tắc cho phép chọn lựa các luật thích hợp áp dụng trên các biểu thức đại số đầu vào nhằm tạo ra các bước ước lượng trên các biểu thức đại số đối tượng có chi phí xử lý thấp hơn tương đương với biểu thức đã cho được đề xuất như sau:

(R1) *Đưa các phép chọn, phép chiếu trên đối tượng, phép bagtoset vào thực hiện trước* các phép kết nối, tích Đề các, nhóm bộ (đối tượng) trên các lớp theo thứ tự lần lượt: Nhằm làm giảm số lượng các đối tượng tham gia trong các phép toán. Phép chiếu được áp dụng cho trường hợp tập thuộc tính của các lớp quá lớn nhưng không tham gia trong kết quả của truy vấn, do đó cho phép giảm kích thước lưu trữ của mỗi lớp.

(R2) *Tổ hợp dãy các phép chọn và phép chiếu*: Dãy các phép toán chọn và chiếu có thể nhóm gộp bằng một phép chọn hoặc một phép chiếu ($\pi_{(A1)}(\pi_{(A2)}(S)) = \pi_{(A1)}$, nếu $A1 \subseteq A2$). Với phép biến đổi này sẽ làm giảm số lần truy xuất trên các lớp.

(R3) *Làm phẳng các cấu trúc phức* với các phép toán *set_flat*, *bag_flat*, *list_flag*: Chuyển các cấu trúc phức về các kiểu tập, bộ và các danh sách với các phần tử đơn trị (lồng nhau). Phép biến đổi này loại bỏ được các tham chiếu lồng, lặp (tự trỏ) trong các cấu trúc phức làm giảm độ phức tạp tính toán trong quá trình xử lý truy vấn.

(R4) *Xử lý trước các lớp sưu tập bằng các phép toán một ngôi*: Làm giảm kích thước các lớp sưu tập khi tham gia kết nối hay lập nhóm.

(**R5**) *Tính các thành phần cơ sở trong một biểu thức đại số đối tượng*: Xác định thành phần cơ sở chung nhất trên các biến vùng, nếu tồn tại một thành phần cơ sở chung nhất thì tính trước các biểu thức con chung này, chúng được xem là đầu vào cho các bước truy vấn tiếp theo.

5.3.3. Thuật toán tối ưu hóa truy vấn đối tượng dựa trên tập luật

Áp dụng các luật trong mục 5.3.1 để thực hiện ước lượng các biểu thức đại số đối tượng, việc ước lượng các biểu thức trung gian phải đảm bảo các quy tắc được đặt ra trong mục 5.3.2. Thuật toán 5.5 sẽ tập trung xử lý các phép toán chiếu, chọn, áp dụng biểu thức đại số (*set_apply*) trên các kiểu đối tượng và phép toán loại bỏ trùng lặp trên các đa tập, lớp sưu tập.

Thuật toán 5.5: Tối ưu hóa các biểu thức đại số đối tượng dựa trên tập luật.

Vào: Biểu thức đại số đối tượng.

Ra: Một dãy các bước ước lượng biểu thức đại số đối tượng.

Phương pháp:

(1) Khởi tạo cây phân tích cú pháp từ biểu thức đại số đối tượng.

(2) Sử dụng luật (L2) tách phép chọn $\sigma_{\lambda s.(f\wedge g\wedge\ldots\wedge h}(S)$ thành chuỗi các phép chọn:

$$\sigma_{\lambda s.f}(\sigma_{\lambda t.g}(\ldots(\sigma_{\lambda u.h}(S))\ldots))$$

(3) Sử dụng các luật kế thừa đối với các phép chiếu (L3), phép chọn và phép apply (L10) tổ hợp dãy các phép chiếu, chọn thành một phép chiếu và một phép chọn.

(4) Đối với mỗi phép chọn, sử dụng các luật (L4, L6, L7, L10) "đẩy" các phép chọn xuống các lớp thành phần hoặc "qua" các nút kết nối và phép tạo nhóm.

(5) Đối với mỗi phép chiếu (đối tượng, tập, bộ), sử dụng luật (L3, L4, L5) để di chuyển phép chiếu xuống càng sâu càng tốt. Nếu

tập thuộc tính được chiếu bao gồm tất cả các thuộc tính của biểu thức thì loại bỏ phép chiếu đó.

(6) Sử dụng các luật (L8, L9, L10) trên các lớp sưu tập, để loại bỏ các phần tử trùng lặp trong các lớp sưu tập; di chuyển phép làm phẳng (*flat*), phép loại bỏ trùng lặp trong các đa tập (*bagtoset*) lên trước các phép toán nhóm hoặc kết nối.

(7) Tạo ra dãy các bước biến đổi để ước lượng mỗi nhóm theo một thứ tự sao cho không có nhóm nào được ước lượng trước các nhóm con của nó.

Độ phức tạp tính toán của thuật toán 5.5 có thời gian đa thức theo kích thước (số các biến thể) của các lớp tham gia trong biểu thức.

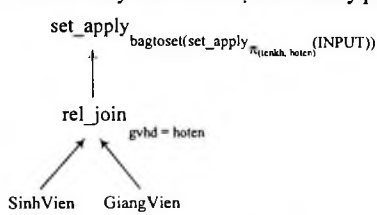
5.3.4. Ví dụ minh họa

Sử dụng cây phân tích cú pháp để thực hiện việc biến đổi các biểu thức đại số đối tượng.

Ví dụ 5.6. Ta có truy vấn:

```
define  SinhVien as s, GiangVien as  e
select distinct (s.tenkhoa.tenkh, e.hoten)
where s.gvhd = e.hoten
```

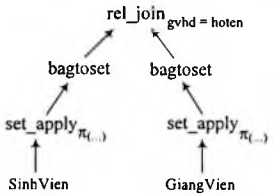
Hình 5.6 là biểu diễn truy vấn của ví dụ 5.6 trên cây phân tích cú pháp.



Hình 5.6. Khởi tạo cây phân tích cú pháp

Hình 5.7 biểu diễn cây phân tích cú pháp khi áp dụng luật "đẩy" phép toán *bagtoset* lên trước toán tử kết nối *rel_join*, luật này áp dụng khi các thành phần trùng lặp quá lớn tồn tại trong các đa tập, như vậy toán tử *bagtoset* chỉ thực hiện trên |s| + |e| biến thể (trong trường hợp

xấu nhất) tốt hơn |s|*|e| biến thể (đối với trường hợp phép kết nối được thực hiện trước). Và tiếp tục đẩy toán tử π qua nút "*join*".

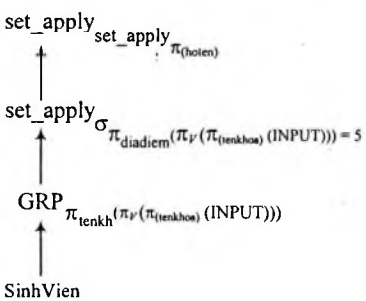


Hình 5.7. Bước chuyển đổi đầu tiên

Ví dụ 5.7. Tìm tên các sinh viên của khoa có văn phòng khoa đặt ở tầng 5 (diadiem). Tên các sinh viên được nhóm theo khoa với thuộc tính tenkh (ví dụ: Sinh học, Công nghệ thông tin, ...):

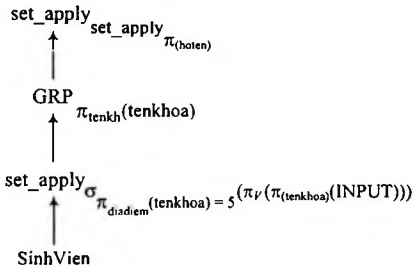
```
define  SinhVien as s
select (s.hoten)
group by s.tenkhoa.tenkh
where s.tenkhoa.diadiem = 5
```

Hình 5.8 biểu diễn cây phân tích cú pháp đại số cho ví dụ 5.7, nhóm đa tập trên thuộc tính tenkh của thuộc tính tenkhoa, sau đó loại bỏ các sinh viên của các khoa không ở tầng 5, cuối cùng chiếu lấy thuộc tính hoten.



Hình 5.8. Khởi tạo cây của ví dụ 5.7

Một phương pháp tối ưu ví dụ 5.7 được suy trực tiếp từ hình 5.9. Trước hết, ta đưa phép chọn lên trước phép tạo nhóm GRP và sử dụng các phép chiếu trên đối tượng (π_V) để loại bỏ tham chiếu tên thuộc tính tenkhoa, sau đó trích chiếu lấy giá trị của thuộc tính diadiem.



Hình 5.9. Cây kết quả sau khi áp dụng các luật chuyển đổi

Trong CSDL hướng đối tượng, các truy vấn đối tượng có cấu trúc lồng được sử dụng khá thường xuyên, các cấu trúc lồng thể hiện trong các biểu thức điều kiện của truy vấn theo hai dạng thức là các truy vấn con lồng, hoặc các biểu thức đường dẫn chứa các kết nối ẩn - các tân từ lồng ở mệnh đề where.

Để xử lý các tân từ lặp (lồng), Cho W. [19] đưa ra phương pháp ước lượng chi phí phụ thuộc tỷ số giữa số các đối tượng của lớp bắt đầu trong biểu thức đường dẫn và tổng số các đối tượng của lớp, dựa trên mối quan hệ nhiều - nhiều giữa các lớp. Tỷ số này là một trong những tham số lựa chọn trong quá trình thiết kế CSDL vật lý.

Đối với là các truy vấn con lồng, chúng ta có thể thực hiện phương pháp tối ưu theo hai bước. Đầu tiên, biến đổi các truy vấn ở mức ngôn ngữ nhằm xử lý một cách hiệu quả các biểu thức con chung và các truy vấn con độc lập. Sau đó, các truy vấn được biên dịch thành các biểu thức đại số lồng nhau và áp dụng các phương pháp biến đổi đại số. Tuy nhiên, khi phân tích sự ước lượng đối với các vòng lặp lồng nhau trong các biểu thức đại số, ta nhận thấy biểu thức kết quả có chi phí là không hiệu quả. Do đó, phương pháp được đề xuất trong phần

sau sẽ giải quyết vấn đề xử lý cho các truy vấn con lồng trong giai đoạn "làm phẳng" các truy vấn lồng nhau bằng phương pháp rút gọn siêu đồ thị kết nối đối tượng, giúp cho phép định giá hiệu quả hơn.

KẾT LUẬN

Việc sử dụng CSDL quan hệ nhúng làm nền trong lưu trữ vật lý và các thao tác dữ liệu được thực hiện qua giao diện hướng đối tượng là giải pháp ứng dụng phù hợp cho một số hệ thống CSDL đối tượng - quan hệ. Vì vậy, bài toán chuyển đổi dữ liệu, ánh xạ lược đồ, biên dịch truy vấn giữa mô hình hướng đối tượng và mô hình quan hệ nhúng được giải quyết khá trọn vẹn là cần thiết.

Bên cạnh đó, phương pháp tối ưu hóa truy vấn đối tượng dựa vào các luật biến đổi biểu thức đại số được áp dụng cho các mô hình dữ liệu hướng đối tượng có hỗ trợ tập luật. Phương pháp này xử lý cho lớp các truy vấn trên các kiểu đối tượng phức như kiểu "túi", đa tập, danh sách, bộ, v.v..., nhưng khi thực hiện trên các lớp sưu tập, các biểu thức đại số đối tượng lồng thì chi phí ước lượng và các phương án thực thi truy vấn chưa hiệu quả. Vì vậy, việc sử dụng siêu đồ thị kết nối đối tượng để giải quyết cho lớp các truy vấn đối tượng lồng với các thuật toán ước lượng trên các siêu cạnh đối tượng sẽ có chi phí xử lý truy vấn hiệu quả hơn so với phương pháp tối ưu các truy vấn đối tượng lồng bằng các phép biến đổi đại số trên các biểu thức đại số đối tượng lồng nhau.

CÂU HỎI ÔN TẬP CHƯƠNG 5

Câu 5.1. Cho lược đồ CSDL VienDH (Viện Đại học), với biểu thức đại số đối tượng sau:

$$set\_apply_{\lambda t.\pi_{(hoten,luamon)}(\pi_V(t))}$$
$$(\sigma_{\lambda s.\pi_{tuoi}(\pi_V(s))>\pi_{tuoi}(\pi_V(\pi_D(\pi_{gvhd}(\pi_V(s)))))}(\text{SinhVienTN}))$$

a) Viết lại câu truy vấn OQL từ biểu thức đại số ở trên;

b) Chuyển đổi sang truy vấn SQL;

c) Sử dụng các luật để tối ưu hóa biểu thức đại số đối tượng trên.

Câu 5.2. Bổ sung một thuộc tính *Tro_giang* (giảng viên trợ giảng) vào lớp KhoaHoc với miền là Khoa. Cho truy vấn:

```
Select c.ten_khoa_hoc
From KhoaHoc c
Where c.DaiHoc.truong_khoa = "Tuyet"
   and c.DaiHoc.ten_nganh = "Khoa hoc may tinh"
   and c.tro_giang.DaiHoc.ten_nganh = "Cong nghe thong tin"
```

a) Chuyển đổi truy vấn trên sang truy vấn SQL;

b) Biểu diễn truy vấn trên cây phân tích cú pháp. Sử dụng các luật để tối ưu hóa.

Câu 5.3. Cho CSDL hướng đối tượng gồm các lớp sau:

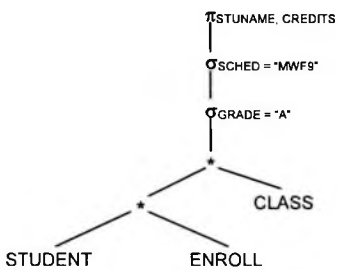
STUDENT (STUID, STUNAME, MAJOR, CREDITS): Quan hệ sinh viên

FACULTY (FACID, FACNAME, DEPT, RANK): Quan hệ về các khoa

CLASS (COURSE#, FACID, SCHED, ROOM): Quan hệ về lớp học

ENROLL (COURSE#, STUID, GRADE): Quan hệ về điểm của sinh viên

Cho cây phân tích cú pháp sau:



a) Viết lại câu truy vấn OQL từ cây phân tích cú pháp;

b) Chuyển đổi truy vấn ở câu a về SQL;

c) Tối ưu hóa truy vấn đối tượng dựa trên tập luật trên cây phân tích cú pháp.

Câu 5.4. Hãy cài đặt mô hình chi phí xử lý truy vấn được trình bày mục 5.1.

CHƯƠNG

6

MỘT SỐ MÔ HÌNH CƠ SỞ DỮ LIỆU HƯỚNG ĐỐI TƯỢNG MỞ RỘNG

Việc quản lý dữ liệu theo mô hình tập trung là chưa đủ mạnh để giải quyết cho lớp các ứng dụng như: viễn thông, hệ thống truyền thông di động, các hệ thống điều khiển giao thông, người máy học, v.v... Do đó, việc nghiên cứu xây dựng hệ thống RTOODB theo mô hình phân tán, gọi là hệ thống CSDL phân tán hướng đối tượng thời gian thực (RTOODDB) là một trong những giải pháp hữu hiệu trong lĩnh vực nghiên cứu đặc tả đối tượng trên các mô hình CSDL tiên tiến.

6

Hệ thống RTOODDB là một hệ thống RTOODB có quan hệ với nhau về mặt logic, được lưu trữ và xử lý phân tán trên một mạng máy tính. Nó có thể được xem như là hệ thống kế thừa các cơ chế của hệ thống CSDL phân tán truyền thống và hệ thống RTOODB.

6.1. CƠ SỞ DỮ LIỆU HƯỚNG ĐỐI TƯỢNG THỜI GIAN THỰC

Một cơ sở dữ liệu hướng đối tượng thời gian thực là một tập hợp các đối tượng được sử dụng để quản lý các hệ thống động với thời gian - tới hạn trong thế giới thực. Mỗi đối tượng có một số trạng thái bên trong được biểu diễn bởi các đối tượng trừu tượng. Một giao dịch truy cập một đối tượng bằng cách gọi các phương thức được định nghĩa bởi lớp của đối tượng.

6.1.1. Lớp đối tượng thời gian thực

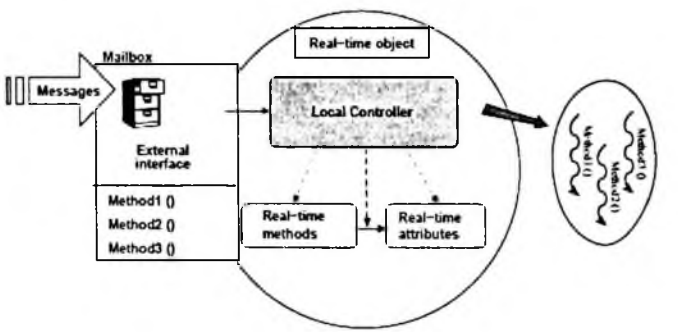
Trong mô hình RTOODB, một lớp đối tượng dữ liệu trừu tượng bao gồm (*N, A, M, CC*), trong đó, *N* là tên của lớp, *A* là tập các thuộc tính, *M* là tập giao diện các phương thức và *CC* là bộ điều khiển cục bộ liên kết với mỗi đối tượng. Một đối tượng dữ liệu *o* là một thể hiện logic của một lớp đối tượng.

Bảng 6.1. Các thành phần của lớp đối tượng thời gian thực

TÊN LỚP (*N*)	
Thuộc tính (*A*)	Các thuộc tính không thời gian thực
	Các thuộc tính cảm biến (sensor attributes)
	Các thuộc tính suy diễn (derived attributes)

Phương thức (*M*)	Các phương thức có chu kỳ (*periodic methods*)
	Các phương thức rời rạc (sporadic methods)
	Các phương thức không có chu kỳ (aperiodic methods).
Bộ điều khiển cục bộ (*CC*)	

Các đối tượng thời gian thực là các thực thể trong mô hình RTOODB. Mỗi đối tượng thời gian thực được đóng gói dữ liệu ràng buộc thời gian, phương thức ràng buộc thời gian và cơ chế điều khiển tương tranh. Như trong hình 6.1, mỗi đối tượng thời gian thực được tạo ra gồm bốn thành phần: (i) một tập hợp các thuộc tính thời gian thực, (ii) một tập hợp các phương thức thời gian thực, (iii) một hộp thư (Mailbox) và (iv) một bộ điều khiển cục bộ (Local Controller).



Hình 6.1. Sơ đồ đối tượng thời gian thực

Hình sau minh họa cho một lớp đối tượng dữ liệu thời gian thực.

Bảng 6.2. Ví dụ lớp đối tượng thời gian thực

MÁY BAY	
Định danh Đích đến	Thuộc tính không thời gian

Hướng bay Vị trí Độ cao Vận tốc	Thuộc tính cảm biến
Quỹ đạo Đường bay	Các thuộc tính suy diễn
Capnhat_Vitri() Capnhat_Docao()	Các phương thức có chu kỳ
Tinh_Quydao() Tinh_Duongbay()	Các phương thức rời rạc
Gan_dinhdanh() Lay_Duongbay() Lay_Tocdo()	Các phương thức không có chu kỳ

a) Thuộc tính

Dữ liệu thuộc tính được chia thành hai loại là dữ liệu không có tính thời gian và dữ liệu thời gian thực. Dữ liệu không có tính thời gian là dữ liệu truyền thống, trong khi dữ liệu thời gian thực chỉ có giá trị trong một khoảng thời gian hiệu lực. Dữ liệu thời gian thực thay đổi liên tục để phản ánh trạng thái thế giới thực. Mỗi dữ liệu thời gian thực có một nhãn thời gian để phản ánh sự quan sát sau cùng của trạng thái thế giới thực.

Thuộc tính không có tính thời gian: Là các thuộc tính mà giá trị của nó không có các ràng buộc thời gian. Do vậy nó không được cập nhật định kỳ theo thời gian.

Thuộc tính thời gian thực: Là các thuộc tính mà giá trị của nó có các ràng buộc thời gian thực, được phân thành hai loại: *thuộc tính cảm biến (sensor attributes)* và *thuộc tính suy diễn (derived attributes)*.

Thuộc tính cảm biến được sử dụng để lưu trữ dữ liệu cảm biến và phải được cập nhật một cách định kỳ nhằm phản ánh chặt chẽ trạng thái thế giới thực của môi trường ứng dụng. Thể hiện trong bảng 6.2, bốn thuộc tính cảm biến của đối tượng máy bay là: *hướng bay, vị trí, độ cao* và *vận tốc*. Những thuộc tính này được cập nhật định kỳ để phản ánh tình trạng của một đối tượng máy bay.

Thuộc tính suy diễn được sử dụng để lưu trữ dữ liệu được tính toán suy diễn từ *thuộc tính cảm biến*. Thể hiện trong bảng 6.2, hai thuộc tính suy diễn của đối tượng máy bay là: *quỹ đạo*, được tính từ hướng bay và các giá trị vị trí; và *đường bay*, được tính từ độ cao và các giá vị trí.

Các thuộc tính thời gian thực được sử dụng để xử lý các tính chất đặc trưng của mô hình dữ liệu thời gian thực. Nó hỗ trợ cho các ràng buộc logic (d_{value}), ràng buộc thời gian ($d_{timestamp}$, d_{avi}) và ràng buộc về lỗi dữ liệu lớn nhất được chấp nhận (d_{mde}). Như trong bảng 6.3, mỗi thuộc tính thời gian thực được đặc trưng bởi (*N, CV, TS, VD, MDE*).

Name (*N*)			
Current Value (*CV*)	Time Stamp (*TS*)	Validity Duration (*VD*)	Maximum Data Error (*MDE*)

Trong đó:

N: Là tên của thuộc tính.

CV: Giá trị hiện tại, được sử dụng để lưu trữ giá trị hiện tại của thuộc tính có được bởi lần cập nhật cuối cùng tương ứng với phương thức. Thành phần này được sử dụng bởi hệ thống để xác định các ràng buộc toàn vẹn logic của giá trị thuộc tính.

TS: Nhãn thời gian, được sử dụng để lưu trữ thời điểm tại đó giá trị của thuộc tính được cập nhật lần cuối. Truy cập tới nhãn thời gian của thuộc tính là cần thiết để xác định ràng buộc thời gian của thuộc

tính. Ví dụ, trong một đối tượng máy bay, thuộc tính *độ cao*, được cung cấp bởi việc đọc thường xuyên bởi một thiết bị cảm biến. Việc cập nhật này được báo cáo mỗi 20 giây một lần. Do đó, thuộc tính độ cao được xem là không nhất quán thời gian nếu việc cập nhật không xảy ra theo đúng với khuôn dạng thời gian này.

Trong mô hình trên, nhãn thời gian là thời điểm khi giá trị được tạo ra. Nếu giá trị được tạo ra bởi một thiết bị cảm biến, khi đó nhãn thời gian là thời điểm giá trị được đọc bởi thiết bị cảm biến. Nếu giá trị được tạo ra bởi một giao dịch, khi đó nhãn thời gian là thời điểm khi giao dịch hoàn tất. Hệ thống sử dụng thuộc tính này để xác định liệu có hoặc không các ràng buộc thời gian đã bị vi phạm.

VD: Khoảng hiệu lực, được sử dụng để lưu trữ độ dài của *khoảng hiệu lực tuyệt đối* (*absolute validity interval,* ký hiệu là *avi*) của giá trị thuộc tính, thể hiện khoảng thời gian mà giá trị thuộc tính được cho là có hiệu lực.

Khoảng hiệu lực của đối tượng, ký hiệu là $vi = [vi_b, vi_e]$ với vi_b và vi_e tương ứng là thời điểm bắt đầu và thời điểm kết thúc của khoảng hiệu lực tuyệt đối.

VD là một số, được kết hợp với *TS*, nhằm xác định tính nhất quán tuyệt đối của thuộc tính. Giá trị một thuộc tính là nhất quán tuyệt đối theo khía cạnh thời gian với điều kiện là thời hạn của giá trị dữ liệu nằm trong một khoảng xác định. Ví dụ, giá trị *độ cao* được coi là mới nếu thời điểm hiện tại là sau nhãn thời gian của độ cao nhưng trước thời điểm kết thúc của khoảng hiệu lực, nghĩa là:

$$\{Độ\_cao.TS \leq current\_time < Độ\_cao.TS + Độ\_cao.VD\}$$

MDE: Lỗi dữ liệu lớn nhất, được sử dụng để ghi nhớ lỗi dữ liệu lớn nhất tuyệt đối chấp nhận trên giá trị thuộc tính. Hiện nay, nhu cầu về các dịch vụ cơ sở dữ liệu thời gian thực đã tăng ở hầu hết các ứng dụng, với mong muốn thực hiện giao dịch trong thời hạn. Do đó, dữ

liệu sử dụng được làm mới nhằm phản ánh sự thay đổi liên tục của môi trường bên ngoài. Tuy nhiên, trong nhiều ứng dụng, các giao dịch cần đáp ứng được cả thời hạn và duy trì được tính nhất quán của CSDL. Để hỗ trợ các ứng dụng này, các hệ thống CSDL thời gian thực cho phép dữ liệu lưu trữ trong cơ sở dữ liệu có thể có một số sai lệch so với giá trị của nó trong thế giới thực. Do đó, lỗi dữ liệu, ký hiệu là *DE*, có cùng kiểu như trường *CV*, đại diện cho độ lệch giữa giá trị hiện tại và giá trị dữ liệu cập nhật. Cận trên của lỗi này được xác định bởi lỗi dữ liệu lớn nhất. Thuộc tính này cho phép hệ thống xử lý khối lượng công việc không thể đoán trước của cơ sở dữ liệu bằng cách loại bỏ các giao dịch cảm biến mà $DE > N.MDE$. Ví dụ, lỗi tối đa trên giá trị *vận tốc* là 5 km/h, hệ thống sẽ loại bỏ các giao dịch cập nhật vận tốc của đối tượng máy bay nếu độ lệch của giá trị hiện tại và giá trị sẽ cập nhật là lớn hơn 5 km/h.

Trong bảng 6.3, *Name* và *Current Value* của một thuộc tính là của người sử dụng. Các trường khác được sử dụng bởi hệ thống CSDL thời gian thực để duy trì tính nhất quán thời gian của cơ sở dữ liệu thời gian thực.

Mô hình hóa cho thuộc tính thời gian thực:

Trong mô hình RTOODB, mỗi thuộc tính thời gian thực được mô hình hóa như một lớp đối tượng. Chúng ta có hai loại lớp đối tượng mô hình hóa cho hai loại thuộc tính thời gian thực như sau.

Lớp đối tượng cho thuộc tính cảm biến: Lớp này mô hình hóa cho các thuộc tính cảm biến. Nó bao gồm các tham số của thuộc tính thời gian thực. Các tham số là (*Oid, Aid, CV, TS, VD, MDE*). Trong đó *Oid* là định danh của đối tượng mà thuộc tính thuộc về, *Aid* là định danh của thuộc tính.

Lớp đối tượng cho thuộc tính suy diễn: Lớp này biểu diễn cho các thuộc tính suy diễn. Nó bao gồm các tham số của thuộc tính thời gian thực. Các tham số là (*Oid, Aid, CV, TS, RCS, RVD*). Trong đó

Oid là định danh của đối tượng mà thuộc tính thuộc về, *Aid* là định danh của thuộc tính, *RCS* (*Relative Consistency Set*) là tập các đối tượng dữ liệu cảm biến được sử dụng để tính toán ra giá trị cho đối tượng này, *RVD* là khoảng hiệu lực của đối tượng suy diễn, độ dài của khoảng hiệu lực đối tượng suy diễn, ký hiệu là $rvi = [vi_b, vi_e]$ được định nghĩa là:

$$rvi = \cap \{vi(o), o \in RCS\}$$

b) Các phương thức thời gian thực

Một phương thức được thực hiện là một giao dịch, trong đó bao gồm một hoặc nhiều giao dịch con (một phương thức có thể gọi các phương thức khác). Các phương thức hướng đối tượng thời gian thực được phân thành ba loại: các phương thức có chu kỳ, các phương thức rời rạc và các phương thức không có chu kỳ.

Các phương thức có chu kỳ: Tính nhất quán thời gian của từng thuộc tính cảm biến được đảm bảo bởi một giao dịch cảm biến, được cập nhật theo chu kỳ các giá trị của các dữ liệu cảm biến. Do đó, mỗi thuộc tính cảm biến được kết hợp với một phương thức có chu kỳ, cập nhật một cách định kỳ các giá trị của các trường *CV* và *TS*. Giả định rằng một phương thức thực hiện có chu kỳ là một giao dịch cảm biến. Sau này được định nghĩa như là một giao dịch chỉ ghi nhằm ghi lại trạng thái của môi trường vào CSDL. Ràng buộc thời gian cho các phương thức có chu kỳ là ràng buộc thời gian tuyệt đối, tức là thời hạn và thời kỳ. Chu kỳ thực hiện phương thức được áp dụng theo khoảng thời gian hiệu lực của mỗi giá trị của thuộc tính cảm biến. Một phương thức có chu kỳ phải hoàn tất việc thực hiện của nó trước thời hạn, nếu không giá trị được ghi sẽ được coi là lỗi thời.

Ví dụ, các đối tượng máy bay được mô tả bởi một tập hợp các phương thức có chu kỳ như: *Capnhat_Docao()*: Được thực hiện định kỳ các thao tác ghi các trường *CV* và *TS* của thuộc tính *Độ cao*; *Capnhat_Vitri()*: Được định kỳ thực hiện các thao tác ghi các trường *CV* và *TS* của thuộc tính *Vị trí*.

Các phương thức rời rạc: Dữ liệu suy diễn là những dữ liệu tính toán từ dữ liệu cảm biến. Do đó, mỗi thuộc tính suy diễn được kết hợp với một phương thức rời rạc, mà không thường xuyên tính toán giá trị của nó từ các thuộc tính cảm biến.

Chế độ truy cập của phương thức rời rạc tới giá trị thuộc tính suy diễn luôn luôn là "ghi". Các ràng buộc thời gian cũng là thời hạn và thời kỳ. Thời điểm thực hiện phương thức phụ thuộc vào các cơ chế cập nhật xem xét. Chẳng hạn, các đối tượng máy bay được mô tả bằng các phương thức rời rạc sau đây: *Tinh_Duongbay()*: tính giá trị thuộc tính *Đường bay* sử dụng giá trị các thuộc tính *Vị trí* và *Độ cao*; *Tinh_Quydao()*: tính giá trị thuộc tính *Quỹ đạo* bằng cách sử dụng các giá trị của các thuộc tính *Vị trí* và *Hướng bay*.

Phương thức không có chu kỳ: Gồm các phương thức mà cho phép đọc/ ghi các thuộc tính không có tính thời gian và chỉ đọc các thuộc tính thời gian thực.

Các giao dịch người sử dụng thường không có chu kỳ. Để xem xét các giao dịch lồng nhau trong mô hình đối tượng, chúng ta giả định rằng việc thực hiện một phương thức không có chu kỳ là một giao dịch người dùng có thể gọi các thao tác nguyên tử hoặc gọi các phương thức khác trên các đối tượng khác. Các thao tác biểu diễn cho hành động của phương thức, bao gồm các câu lệnh điều kiện rẽ nhánh, vòng lặp, các thao tác I/O, đọc/ ghi các thuộc tính không có tính thời gian, và chỉ đọc các thuộc tính thời gian thực, kể cả các trường *CV, TS, VD* và *MDE*.

c) Hộp thư

Đặc trưng của mô hình đối tượng là tính modul và tính đóng gói, nó sử dụng một phương thức duy nhất cho việc trao đổi thông tin giữa hai đối tượng là truyền thông điệp. Trong ngữ cảnh này, tất cả các quá trình xử lý đều được kích hoạt bởi các thông điệp đến. *Hộp thư* được sử dụng để lưu trữ các thông điệp nhận được bởi đối tượng thời gian

thực. Một hộp thư gắn với một đối tượng và thường cho một đối tượng nhất định. Nó được sử dụng để lưu trữ các thông điệp đến đối tượng và chờ được xử lý.

Trong hầu hết các ứng dụng thời gian thực, các ràng buộc thời gian thực được xử lý thông qua các thông điệp. Mỗi thông điệp có một thời hạn phải được đáp ứng, nếu không thông điệp sẽ bị từ chối.

d) Bộ điều khiển cục bộ

Do bản chất phong phú của thế giới thực, nhiều giao dịch có thể gửi yêu cầu đồng thời tới cùng một đối tượng thời gian thực. Việc thực hiện đồng thời nhiều giao dịch cho phép một số phương thức xử lý đồng thời trên cùng một đối tượng. Vì vậy, để quản lý việc thực hiện đồng thời các phương thức, mỗi đối tượng thời gian thực được kết hợp với một cơ chế điều khiển tương tranh cục bộ, gọi là *điều khiển cục bộ*.

Bộ điều khiển cục bộ phải tính toán mỗi yêu cầu nhận được, lựa chọn các yêu cầu để được thực hiện theo các ràng buộc thời gian của các yêu cầu khác nhau của hộp thư, với hai trường hợp: (1) khi một luồng có sẵn; (2) khi phải đình chỉ yêu cầu hiện tại có quyền ưu tiên thấp để giải phóng một luồng dành cho một yêu cầu có quyền ưu tiên cao hơn. Mỗi yêu cầu tương ứng với một phương thức của đối tượng đích. Bên cạnh đó, bộ điều khiển cục bộ phải kiểm tra những ràng buộc đồng thời giữa các phương thức của các yêu cầu cần lựa chọn và các phương thức đang thực hiện của đối tượng thời gian thực. Nếu phát hiện có xung đột, cơ chế điều khiển tương tranh được thực hiện. Khi một phương thức kết thúc, các luồng tương ứng được giải phóng và ràng buộc đồng thời được hủy bỏ. Tuy nhiên, nếu giao dịch là có chu kỳ, luồng chưa được giải phóng, vì nó được sử dụng để thực hiện theo chu kỳ của các giao dịch yêu cầu. Như vậy bộ điều khiển cục bộ làm nhiệm vụ của bộ quản lý hộp thư, bộ lập lịch điều khiển ràng buộc, điều khiển ràng buộc đồng thời và quản lý luồng.

6.1.2. Ngữ nghĩa dữ liệu thời gian thực

Trong mô hình dữ liệu thời gian thực, mỗi khi dữ liệu thời gian thực được cập nhật thì giao dịch cập nhật tạo ra một phiên bản mới cho đối tượng dữ liệu đó. Do đó, các phiên bản khác nhau của đối tượng dữ liệu thời gian thực thay đổi theo thời gian được lưu trữ trong CSDL. Có hai cách tiếp cận để mô hình hóa dữ liệu đa phiên bản của dữ liệu thời gian: *phiên bản thuộc tính* (*attribute versioning*) và *phiên bản đối tượng* (*object versioning*). Trong phiên bản thuộc tính, một khoảng hiệu lực được kết hợp với mỗi thuộc tính của đối tượng. Trong phiên bản đối tượng, một khoảng hiệu lực được kết hợp với toàn bộ đối tượng. Ở đây chúng ta tập trung vào phiên bản đối tượng, nó duy trì nhiều phiên bản của mỗi đối tượng. Mỗi trạng thái của đối tượng dữ liệu thời gian thực có một khoảng hiệu lực trong suốt khoảng thời gian mà trạng thái đó được xem là có hiệu lực.

Phiên bản thứ i của đối tượng o, ký hiệu o_i (i =1, 2, ...) được định nghĩa:

$$(value(o_i), vi(o_i))$$

Với $value(o_i)$ biểu diễn trạng thái thứ i của đối tượng o và $vi(o_i)$ biểu diễn cho khoảng hiệu lực của $value(o_i)$, tức là khoảng thời gian mà $value(o_i)$ được xem là có hiệu lực. Sau thời điểm $vi_e(o_i)$, $value(o_i)$ không còn hiệu lực. Bởi vậy, các thuộc tính của đối tượng dữ liệu thời gian được định nghĩa như sau:

o_i: Phiên bản thứ i của đối tượng dữ liệu o;

$vi_b(o_i)$: Thời điểm bắt đầu khoảng hiệu lực của o_i;

$vi_e(o_i)$: Thời điểm kết thúc khoảng hiệu lực của o_i;

$vi(o_i)$: Khoảng thời gian có hiệu lực (gọi tắt là khoảng hiệu lực) của o_i;

$$vi(o_i) = [vi_b(o_i), vi_e(o_i)], \text{ với } vi_b(o_i) < vi_e(o_i);$$

Phiên bản thứ i của đối tượng dữ liệu o, là nhất quán thời gian tại thời điểm t nếu và chỉ nếu:

$$vi_b(o_i) <= t < vi_e(o_i)$$

Ví dụ: Minh họa cho đối tượng thời gian thực *Vận tốc*, giá trị của các thuộc tính *VD* và *MDE* là giống nhau cho tất cả các phiên bản của đối tượng, nhưng giá trị của thuộc tính *CV* và *TS* được thay đổi đối với mỗi phiên bản.

Bảng 6.4. Minh họa cho các phiên bản khác nhau của đối tượng

Vận tốc			
CV	TS	VD	MDE
900	10:20:02	7s	5km/h
920	10:20:09	7s	5km/h
925	10:20:16	7s	5km/h

Như vậy, *VD* của phiên bản o_i của đối tượng o là hiệu của $vi_e(o_i) - vi_b(o_i)$.

6.2. CƠ SỞ DỮ LIỆU HƯỚNG ĐỐI TƯỢNG PHÂN TÁN - THỜI GIAN THỰC

Hệ thống RTOODB tập trung là chưa đủ mạnh để xây dựng các hệ thống lớn, phức tạp. Bởi vì, nhiều hệ thống RTOODB của các tổ chức được phân tán ở nhiều vị trí địa lý khác nhau. Như vậy, một hệ thống cơ sở dữ liệu phân tán hướng đối tượng thời gian thực (RTOODDB) là một hệ thống phải thỏa mãn được các đặc điểm của mô hình cơ sở dữ liệu phân tán và mô hình RTOODB.

Đối với các hệ thống phân tán, một ứng dụng ở một vị trí đòi hỏi sự truyền thông/ giao tiếp với cùng một ứng dụng ở vị trí khác. Sự bắt buộc này cần thiết cho một phiên bản phân tán của hệ thống CSDL thời gian thực. Áp dụng các nguyên tắc của hệ thống CSDL thời gian thực và hệ thống CSDL phân tán làm vấn đề trở nên phức tạp, như sự trao đổi giữa một CSDL ở một vị trí với một CSDL ở một vị trí khác, điều khiển tương tranh, duy trì tính nhất quán thời gian, v.v... Cộng thêm các chi phí truyền thông làm cho tính chất thời gian của yêu cầu

trên các vị trí cách biệt khó đoán trước. Như vậy, một hệ thống phân tán thời gian thực kết hợp các đặc điểm của hệ thống phân tán và hệ thống thời gian thực. Điều này có nghĩa rằng trong một hệ thống như vậy, các vấn đề liên quan đến phân tán như thực hiện các thuật toán phân tán và truyền thông mạng phải được giải quyết với sự quan tâm đến các yêu cầu thời gian thực.

Trong mô hình RTOODDB, một CSDL *D* được phân tán trên *N* nút, mỗi CSDL *di* (với $0 < i \leq N$) thường trú trên một nút thứ *i* được gọi là một vị trí. Mỗi vị trí lưu trữ một tập hợp các đối tượng dữ liệu thời gian thực và các đối tượng dữ liệu không có tính thời gian, được gọi là vị trí gốc cho các đối tượng dữ liệu đó. Mỗi vị trí cũng duy trì một tập các bản sao của đối tượng dữ liệu thời gian tổ chức bởi các vị trí khác. Một bản sao $r \in R$ của đối tượng logic *o* là một khung nhìn cục bộ tại một vị trí về trạng thái của *o*. Mỗi vị trí chứa nhiều nhất một bản sao của một đối tượng logic cụ thể. Trong trường hợp lý tưởng, tất cả các bản sao của một đối tượng *o* phù hợp với trạng thái nhất quán của đối tượng *o* (chẳng hạn, trạng thái của chúng là đồng nhất). Như vậy, CSDL địa phương tại mỗi vị trí bao gồm một tập các đối tượng dữ liệu logic của chúng và một tập các bản sao của các đối tượng định vị bởi các nút khác.

Kích thước *D* của CSDL là tổng kích thước của CSDL tại tất cả các vị trí, được định nghĩa như sau:

$$D = \sum_{i=1}^{N} d_i$$

Mỗi đối tượng có một số trạng thái bên trong được bảo vệ bởi các đối tượng trừu tượng. Một giao dịch truy cập một đối tượng bằng cách gọi các phương thức được định nghĩa bởi lớp của đối tượng này.

Trong các hệ thống RTOODDB, cần duy trì được tính nhất quán giữa trạng thái thực tại của đối tượng thời gian thực trong môi trường mở rộng và ảnh hưởng của nó lên tất cả các bản sao của nó trên nhiều vị trí.

Giá trị mới của đối tượng dữ liệu thời gian được cập nhật định kỳ từ vị trí gốc và lan truyền tới các vị trí có chứa bản sao. Với một đối tượng dữ liệu cụ thể, dữ liệu sao chép tại vị trí gốc được gọi là sao chép bản gốc và sự sao chép dữ liệu bản sao được gọi là sao chép bản sao.

6.2.1. Mô hình hóa dữ liệu thời gian thực trong hệ thống

6.2.1.1. Mô hình dữ liệu

Mỗi đối tượng logic tại mỗi vị trí được định danh bởi một *OID* và các thành phần (*OID, Attributes, Methods, CCManager(), Mailbox()*). Trong đó, *Attributes* là tập các thuộc tính, *Methods* là tập các phương thức, *CCManager()* là bộ điều khiển tương tranh cục bộ của đối tượng, *Mailbox()* là hộp thư. Mỗi thuộc tính thời gian thực được biểu diễn như một đối tượng thời gian thực trong hệ thống. Do vậy, trong mô hình RTOODDB, mỗi đối tượng thời gian thực có các thành phần như sau:

a) Tập các thuộc tính

- Thuộc tính cho đối tượng cảm biến (Sensor Object).

Tập các thuộc tính cho đối tượng cảm biến gồm:

{*Id, LsiteId, CV, TS, VD, MDE, BUF*}

Trong đó:

Id: Là định danh duy nhất của đối tượng trên ví trí gốc của nó.

LsiteId: Là định danh *id* của vị trí mà đối tượng được hình thành, thuộc tính này chỉ ra một dấu hiệu cho biết đối tượng là đối tượng dữ liệu logic hoặc là một bản sao vật lý của đối tượng tại vị trí khác. Chẳng hạn, nếu *LsiteId = SiteId* (tức là định danh *id* của vị trí nơi đối tượng được hình thành trùng với định danh *id* của vị trí đang lưu trữ đối tượng) thì đối tượng là một đối tượng nguyên thủy hình thành tại vị trí này, ngược lại nó là một bản sao của một đối tượng dữ liệu ở vị trí khác.

CV, TS, VD, MDE: Tương ứng là giá trị hiện tại, nhãn thời gian, chiều dài của khoảng hiệu lực và lỗi dữ liệu lớn nhất của đối tượng.

BUF (*Basic Update Frequency*): Là tần số cập nhật cơ bản, mỗi đối tượng dữ liệu thời gian thực logic được cập nhật một cách định kỳ theo một tần số cập nhật xác định tại vị trí gốc của nó, trong khi các bản sao vật lý của nó được cập nhật theo tần số cập nhật mở rộng khác là *EUF (Extended Update Frequency)*.

- Thuộc tính cho đối tượng suy diễn (Derived Object).

Tập các thuộc tính cho đối tượng suy diễn gồm:

{Id, LsiteId, CV, TS, RCS, RVD, BUF}

Trong đó:

RCS: Là tập ràng buộc quan hệ, nó bao gồm một tập các đối tượng dữ liệu cảm biến mà được sử dụng để tính ra giá trị cho đối tượng dữ liệu suy diễn.

RVD: Kí hiệu cho các ràng buộc quan hệ của đối tượng. Chẳng hạn, xét hai đối tượng *o* và *o'* có nhãn thời gian tương ứng là TS_1 và TS_2. *o* và *o'* thỏa mãn tính nhất quán quan hệ nếu:

$|o_{TS1} - o'_{TS2}| \leq RVD$, *với RVD* độ dài khoảng hiệu lực quan hệ.

BUF: Là tần số cập nhật cơ bản cho giá trị của đối tượng suy diễn, phụ thuộc vào yêu cầu của ứng dụng cụ thể. Nếu đối tượng này là một bản sao, nó sẽ được tái tạo một cách định kỳ theo một tần số cập nhật xác định nhận được từ vị trí gốc của nó.

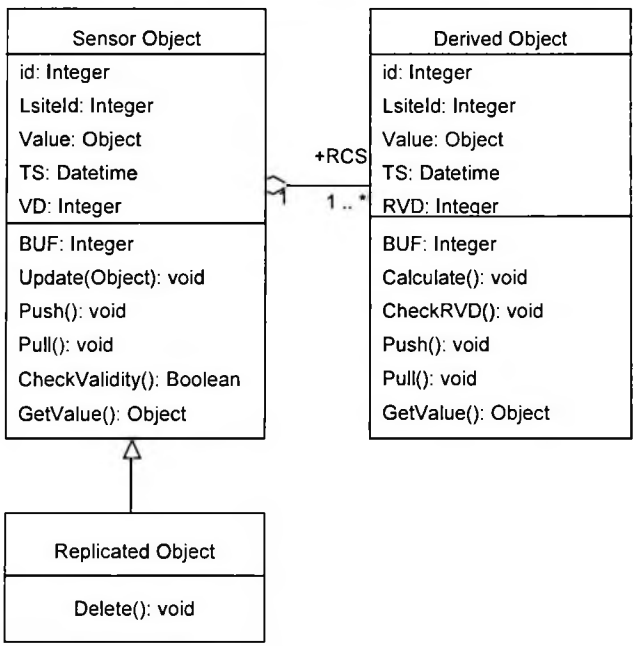
b) Tập các phương thức

Tập các phương thức được phân loại thành ba lớp: Các phương thức có chu kỳ, các phương thức rời rạc và các phương thức không có chu kỳ.

- Các phương thức có chu kỳ:

Bao gồm các phương thức nhằm duy trì tính nhất quán thời gian đối với các đối tượng dữ liệu thời gian thông qua việc cập nhật dữ liệu cảm biến hoặc cập nhật cho các đối tượng dữ liệu bản sao ở các vị trí từ

xa. Do đó, chúng ta có thể phân biệt hai hành động liên quan với chiến lược làm mới bản sao: hành động đẩy ra (*push* action) trong đó việc cập nhật một bản sao được gửi một cách định kỳ theo tần suất cập nhật xác định; và hành động kéo vào (*pull* action) khi sự cập nhật được yêu cầu từ một vị trí từ xa. Phương thức cập nhật (*Update*) sử dụng *BUF* để cập nhật đối tượng dữ liệu cảm biến, và phương thức lan truyền (*Propagate*) để cập nhật bản sao của nó trên vị trí từ xa sử dụng *EUF*.



Hình 6.2. Mô hình hóa lớp đối tượng

- Các phương thức rời rạc:

Được kết hợp với dữ liệu suy diễn. Bao gồm một phương thức *Calculate()* để tính giá trị của nó từ các mục dữ liệu cảm biến định rõ

trên *RCS*. Và một phương thức *CheckRVD()* có thể được định nghĩa để kiểm tra nhằm duy trì tính nhất quán quan hệ của đối tượng đó bằng cách kiểm tra *RVD*.

- Các phương thức không có chu kỳ:

Bao gồm các phương thức còn lại cho phép đọc các đối tượng dữ liệu thời gian thực, các giao dịch đặc thù của người sử dụng đến không theo chu kỳ. Các phương thức này không ghi dữ liệu thời gian thực nhưng có thể đọc/ ghi dữ liệu không có tính thời gian và chỉ đọc dữ liệu thời gian thực. Một phương thức *CheckValidity()* có thể được định nghĩa để kiểm tra tính hiệu lực của mỗi đối tượng dữ liệu, sử dụng nhãn thời gian và khoảng hiệu lực của nó như được mô tả ở trên.

6.2.1.2. Tính nhất quán dữ liệu

Có ba kiểu tính chất khác nhau đối với tính nhất quán:

(i) *Tính nhất quán thời gian bên ngoài (External temporal consistency)*: Giải quyết mối quan hệ giữa một đối tượng của thế giới bên ngoài với giá trị của nó được lưu trong CSDL.

(ii) *Tính nhất quán thời gian đối tượng-bên trong (Inter-object temporal consistency)*: Là mối quan hệ giữa các đối tượng khác nhau hoặc các sự kiện (trong một nút đơn), nó cũng bao gồm mối quan hệ giữa các đối tượng dữ liệu thời gian thực và các đối tượng dữ liệu không có tính thời gian mà phụ thuộc vào các đối tượng dữ liệu thời gian thực này.

(iii) *Tính nhất quán qua lại (Mutual consistency)*: Phản ánh mối quan hệ giữa các đối tượng logic và các bản sao vật lý của nó tại các vị trí khác nhau.

Bản sao trong CSDL thời gian thực, được tạo bằng cách lặp lại các đối tượng dữ liệu thời gian, thay thế cho các yêu cầu truy xuất dữ liệu từ các vị trí ở xa. Các giao dịch cần đọc các dữ liệu ở vị trí ở xa giờ đây có thể truy cập các bản sao tại vị trí của nó, điều này hỗ trợ cho các giao dịch đạt được thời hạn và các yêu cầu làm mới dữ liệu.

6.2.2. Mô hình giao dịch trong hệ thống RTOODDB

Giao dịch là một dãy các thao tác trên CSDL nhằm duy trì các tính chất ACID. Tính nguyên tử (*Atomic*) có nghĩa là một giao dịch hoặc tất cả mọi thao tác đều được thực hiện hoặc không có thao tác nào được thực hiện. Tính nhất quán (*Consistent*) có nghĩa là chuyển CSDL từ một trạng thái nhất quán này tới một trạng thái nhất quán khác. Tính biệt lập (*Isolation*) có nghĩa là việc thực hiện giao dịch là hoàn toàn độc lập với việc thực hiện của giao dịch khác. Và tính bền vững (*Durability*) có nghĩa là khi giao dịch ủy thác các kết quả của nó được ghi trong các tham số lưu trữ là không thể thay đổi trở lại.

6.2.2.1. Phân loại giao dịch

Giao dịch trong hệ thống RTOODDB cũng được phân thành hai loại:

Giao dịch cập nhật: Thực hiện các thao tác cập nhật giá trị cho dữ liệu thời gian. Bao gồm các giao dịch định kỳ cập nhật các giá trị cho đối tượng dữ liệu cảm biến, các giao dịch rời rạc cập nhật cho các giá trị đối tượng suy diễn tại vị trí gốc và các giao dịch cập nhật giá trị cho các đối tượng bản sao tại các vị trí khác.

Giao dịch người sử dụng (giao dịch ứng dụng): Là các yêu cầu của người sử dụng hoặc đến từ các ứng dụng trong hệ thống, các giao dịch này chỉ đọc các đối tượng dữ liệu thời gian thực nhưng đọc hoặc ghi các đối tượng dữ liệu không có tính thời gian thực.

Giao dịch được biểu diễn như một dãy các thao tác trên các đối tượng dữ liệu. Các thao tác trong một giao dịch được thực hiện theo cơ chế tuần tự.

Các giao dịch trong hệ thống có thể được phân lớp thành các *giao dịch cục bộ* và *giao dịch toàn cục*. Một giao dịch được gọi là giao dịch cục bộ nếu tất cả các thao tác của nó được thực hiện trong một vị trí và không cần truy cập tới bất kỳ đối tượng dữ liệu tại vị trí nào khác. Các giao dịch là toàn cục nếu có ít nhất một thao tác của nó thực

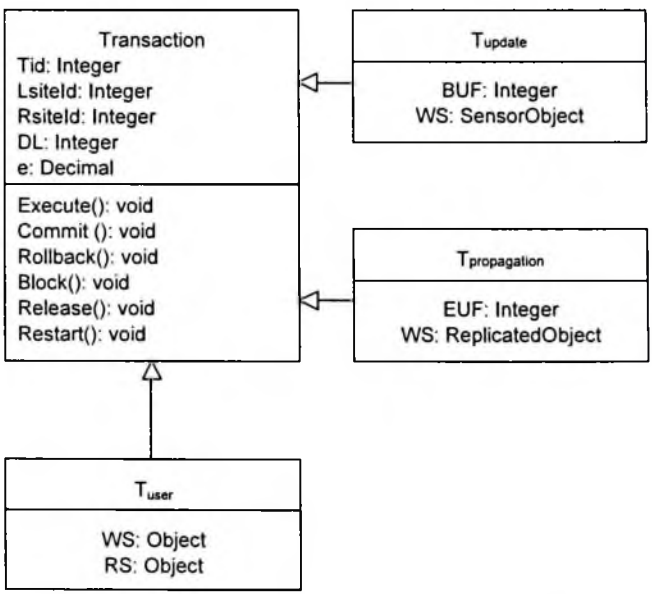
hiện trên một vị trí từ xa. Bộ quản lý giao dịch trong hệ thống có chức năng ánh xạ các giao dịch cập nhật và giao dịch người sử dụng thành giao dịch cục bộ và giao dịch toàn cục.

6.2.2.2. Mô hình giao dịch

Biểu diễn hình thức của giao dịch có các tham số như sau:

$$T_{Type}(Tid, LsiteId, RsiteId, dl, e)$$

Trong đó, *Type* là kiểu của giao dịch, giao dịch cập nhật hoặc giao dịch người sử dụng; *Tid* là định danh duy nhất của giao dịch; *LsiteId* là vị trí mà giao dịch được hình thành. *RsiteId* là vị trí mà giao dịch được gửi đến; *dl* và *e* tương ứng là thời hạn và thời gian thực hiện của giao dịch.



Hình 6.3. Mô hình hóa lớp giao dịch

a) Đối với các giao dịch cập nhật dữ liệu thời gian thực

Bao gồm các giao dịch thực hiện việc cập nhật giá trị cho dữ liệu thời gian thực tại ví trí gốc và cập nhật giá trị cho dữ liệu thời gian thực là bản sao tại ví trí từ xa.

$$T_{update} = \{Tid, LsiteId, RsiteId, WS, BUF, dl, e\}$$

Trong đó kiểu *update* là một giao dịch cập nhật dữ liệu thời gian thực tại vị trí gốc, lúc này *LsiteId= RsiteId, WS* là tập ghi của giao dịch (tập các đối tượng cảm biến cần cập nhật), *BUF* (hay period) là tần số/chu kỳ cập nhật cơ bản của giao dịch.

Khoảng thời gian thực hiện của giao dịch cập nhật luôn luôn nhỏ hơn thời gian hiệu lực của đối tượng, vì vậy đối tượng luôn được cập nhật trước khi nó không còn nhất quán về mặt thời gian, điều giả định này sẽ đem lại một khoảng thời gian cho giao dịch cập nhật bản sao ở các vị trí từ xa.

Giao dịch *T* cập nhật cho một đối tượng dữ liệu thời gian thực tại vị trí gốc trong hệ thống được thực hiện một cách có chu kỳ, việc thực hiện giao dịch có thể được mô tả theo thuật toán sau:

Thuật toán: UPDATE

Vào: Giao dịch *T*, tần số cập nhật *BUF* của *T*

Ra: Phiên bản o_i của đối tượng cần cập nhật

Phương pháp:

```
(1)  Begin
(2)     s(T) = Get_Current _Time();
(3)     next_time = s(T) + BUF;
(4)     sleep_time = BUF;
(5)     LOOP
(6)        Sleep(sleep_time);
```

// Thực hiện việc cập nhật (tạo phiên bản mới) cho đối tượng thời gian thực //

(7) Current_time = Get_Current_Time();

(8) next_time = next_time + BUF;

(9) sleep_time = next time - current time;

(10) **END LOOP**

(11) **End**;

$T_{propagation}$ = {*Tid, LsiteId, RsiteId, WS, EUF, dl, e*}

Trong đó, kiểu *propagation* là cập nhật cho các bản sao của các đối tượng dữ liệu trên các vị trí từ xa, *WS* là tập ghi của giao dịch (tập các bản sao của đối tượng cảm biến cần cập nhật), *EUF* là tần số cập nhật mở rộng xác định bởi các yêu cầu của hệ thống.

Tất cả các bản sao của một đối tượng dữ liệu cụ thể được cập nhật bằng cách sử dụng giá trị mới (giá trị đang có hiệu lực) từ bản sao chép gốc của chúng (đối tượng dữ liệu lưu trữ tại vị trí gốc). Khi một bản sao được cập nhật một cách định kỳ, nó được gọi là *bản sao hoạt động* (*active replica*). Ngược lại, nó được gọi là *bản sao không hoạt động* (*dormant replica*). Một bản sao hoạt động sẽ trở thành bản sao không hoạt động nếu khoảng hiệu lực của nó là hết hạn và chưa được cập nhật giá trị mới. Thời điểm mà nó trở thành bản sao không hoạt động được gọi là thời điểm đóng của nó, ký hiệu là *CT* (*closing time*).

Tập *WS* của một giao dịch chứa các đối tượng được định dạng như sau:

DataObject = {OId, SiteId, DataType}

Mỗi đối tượng dữ liệu chứa các thông tin về định danh của đối tượng dữ liệu (*OId*); định danh CSDL (*SiteId*); kiểu dữ liệu (*DataType*), là đối tượng dữ liệu logic (Sensor Object) hay đối tượng dữ liệu bản sao (Replicated Object).

b) Đối với giao dịch người sử dụng

$$T_{user} = (Tid, LsiteId, RsiteId, RS, WS, dl, e)$$

Bao gồm tất cả các giao dịch mà được thực hiện hoặc ở vị trí địa phương hoặc ở vị trí từ xa. *RS* là tập đọc (Sensor Object hoặc Replicated Object) và WS là tập ghi của giao dịch (Object).

6.2.3. Ràng buộc thời gian của giao dịch

6.2.3.1. Điều kiện ủy thác

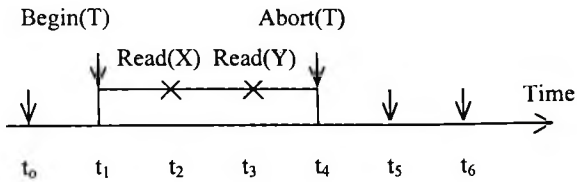
Một giao dịch *T* trong hệ thống cơ sở dữ liệu thời gian thực có thể ủy thác nếu và chỉ nếu:

(i) *T* là nhất quán logic, tức là khả tuần tự và thỏa mãn tất cả các ràng buộc toàn vẹn dữ liệu.

(ii) *T* đạt được thời hạn của nó.

(iii) *T* đọc các đối tượng dữ liệu nhất quán thời gian và dữ liệu mà *T* đọc vẫn còn hợp lệ (tức là còn mới) khi *T* ủy thác.

Dữ liệu được đọc bởi giao dịch phải có hiệu lực khi giao dịch hoàn thành, điều này dẫn tới một ràng buộc khác đối với thời điểm hoàn thành, được thêm vào thời hạn của giao dịch. Ràng buộc này được gọi là *thời hạn - dữ liệu (data-deadline)*. Trong cùng lớp giao dịch, thuật toán lập lịch quan tâm về thời hạn dữ liệu của giao dịch, đó là thời điểm cuối cùng mà giao dịch sẽ vi phạm ràng buộc thời gian. Thuật toán lập lịch sẽ tính thời hạn - dữ liệu khi lập lịch giao dịch. Mỗi khi thời hạn - dữ liệu là nhỏ hơn thời hạn của giao dịch tương ứng, ít có khả năng để giao dịch có thể được ủy thác trước thời hạn của nó. Do đó, giao dịch có thể bị hủy bỏ hoặc khởi động lại hoặc phải chờ cho tới khi phiên bán mới của đối tượng dữ liệu được tạo ra. Ví dụ sau minh họa khái niệm thời hạn - dữ liệu.



Hình 6.4. Minh họa cho ràng buộc thời hạn - dữ liệu

Trong hình 6.4, giao dịch *T* cần đọc hai đối tượng dữ liệu thời gian thực là *X* và *Y*. *T* đọc các đối tượng dữ liệu đó tương ứng tại các thời điểm t_2 và t_3. Thời hạn của giao dịch *T* là t_6. Đối tượng dữ liệu *X* có khoảng hiệu lực là $[t_0, t_5]$, đối tượng dữ liệu *Y* có khoảng hiệu lực là $[t_0, t_4]$. Giao dịch *T* bắt đầu tại thời điểm t_1, và nó không có thời hạn-dữ liệu tại thời điểm này. Tại thời điểm t_2, giao dịch *T* đọc đối tượng dữ liệu *X*, thời hạn-dữ liệu của *T* trở thành t_5 và nó sẽ vi phạm tính nhất quán thời gian sau thời điểm t_5. Để thỏa mãn tính nhất quán thời gian, giao dịch *T* phải được lập lịch để ủy thác trước thời điểm t_5, tức là trước khi giá trị *X* trở nên không còn hiệu lực. Lưu ý rằng thời hạn của giao dịch *T* là sau thời điểm t_5. Tiếp theo, giao dịch *T* xử lý và đọc đối tượng dữ liệu *Y* mà giá trị sẽ trở nên hết hiệu lực sau thời điểm t_4. Lúc này, thời hạn - dữ liệu của giao dịch *T* trở thành t_4. Tại thời điểm t_4, giao dịch *T* đã không hoàn thành. Do đó, giao dịch *T* bị hủy bỏ. Chú ý rằng, có thể xảy ra khả năng để khởi động lại giao dịch *T* và sử dụng các phiên bản dữ liệu mới của *X* và *Y* để đạt được thời hạn của giao dịch *T* là t_6.

6.2.3.2. Một số thuộc tính của giao dịch

Một số thuộc tính của giao dịch *T* sử dụng trong quá trình xử lý giao dịch là:

a(T): Thời điểm đến của giao dịch *T*.

s(T): Thời điểm bắt đầu của giao dịch *T*.

$dl(T)$: Thời hạn của giao dịch T.

$dd_t(T)$: Thời hạn - dữ liệu của giao dịch T tại thời điểm t.

$L(T)$: Số đối tượng dữ liệu mà T truy cập (bao gồm các đối tượng trong tập WS và tập RS).

$L_t^{rt}(T)$: Số đối tượng thời gian còn lại được truy cập bởi T tại thời điểm t.

$L_t^{nort}(T)$: Số đối tượng không có tính thời gian còn lại được truy cập bởi T tại thời điểm t.

$L_t(T)$: Số đối tượng còn lại truy cập bởi T tại thời điểm t,

$$L_t(T) = L_t^{rt}(T) + L_t^{nort}(T)$$

$$L_{a(T)}(T) = L(T)$$

$E_t(T)$: Thời gian thực hiện còn lại ước lượng của giao dịch T tại thời điểm t.

$R_t(T)$: Thời gian trả lời còn lại ước lượng của giao dịch T tại thời điểm t.

$C_t(T)$: Thời điểm hoàn thành ước lượng của T tại thời điểm t.

$$C_t(T) = t + E_t(T)$$

$RS_t(T)$: Tập đọc của T tại thời điểm t; tập này chứa các phiên bản của tất cả các đối tượng dữ liệu thời gian đọc bởi T.

$P_t(T)$: Quyền ưu tiên của T tại thời điểm t.

Thời hạn - dữ liệu của giao dịch T tại thời điểm t, ký hiệu là $dd_t(T)$, được định nghĩa là: $dd_t(T) = \min_{o \in RS_t^{rt}(T)} vi_e(o)$.

6.2.3.3. Tính thời hạn cho giao dịch

Thời gian tồn tại của giao dịch phân tán được chia thành hai giai đoạn: *giai đoạn thực hiện* và *giai đoạn ủy thác*. Trong giai đoạn thực hiện, các thao tác của giao dịch được xử lý tại các vị trí khác nhau của

hệ thống, trong khi trong giai đoạn ủy thác, một giao thức ủy thác phải đảm bảo tính nguyên tử của giao dịch. Các giao dịch trong giai đoạn thực hiện được gọi là *giao dịch thực hiện* và các giao dịch trong giai đoạn ủy thác được gọi là *giao dịch ủy thác*.

Thời hạn của giao dịch được kiểm soát bằng cách ước lượng thời gian thực hiện một giao dịch và hệ số gia tăng, ký hiệu là *SF* (*SlackFactor*), yếu tố này cung cấp sự điều khiển tính thu hẹp và tính gia tăng của thời hạn. Chúng ta tính toán thời hạn của các giao dịch sử dụng phương pháp đưa ra dưới đây. Thời hạn của giao dịch (toàn cục và cục bộ) được tính dựa trên thời gian thực hiện dự kiến của chúng.

Thời hạn của giao dịch *T*, ký hiệu là *dl(T)* được xác định là:

$$dl(T) = a(T) + SF * R(T)$$

Trong đó *a(T)* là thời điểm đến của giao dịch *T*; *SF* là hệ số nới lỏng; *R(T)* là thời gian trả lời tối thiểu của giao dịch. Theo mô hình sử dụng, các giao dịch thành viên được thực hiện trong môi trường song song, do đó *R(T)* có thể được tính toán:

$$R(T) = R(T_p) + R(T_c)$$

Trong đó $R(T_p)$ là tổng thời gian xử lý trong giai đoạn thực hiện và giai đoạn ủy thác; $R(T_c)$ là độ trễ truyền thông trong giai đoạn thực hiện và giai đoạn ủy thác, được cho như sau:

+ Đối với giao dịch toàn cục:

$$R(T_p) = Max(T_{process} * N_{oper\_local\_rt} + (2T_{lock} + T_{process}) * N_{oper\_local\_nort},$$
$$T_{process} * N_{oper\_remote\_rt} + (2T_{lock} + T_{process}) * N_{oper\_remote\_nort})$$

$$R(T_c) = N_{comm} * T_{com}$$

+ Đối với giao dịch cục bộ:

$$R(T_p) = T_{process} * N_{oper\_local\_rt} + (2T_{lock} + T_{process}) * N_{oper\_local\_nort}$$

$$R(T_c) = 0$$

Trong đó, T_{lock} là thời gian yêu cầu để khóa hoặc mở khóa một đối tượng dữ liệu không có tính thời gian; $T_{process}$ là thời gian để xử lý một đối tượng dữ liệu (giả sử rằng thao tác đọc và thao tác ghi mất cùng khoảng thời gian như nhau); N_{comm} là số thông điệp; T_{com} là độ trễ truyền thông, tức là thời gian ước lượng cho một thông điệp đi từ vị trí này tới vị trí khác; $N_{oper\_local\_rt}$ là số thao tác cục bộ trên các đối tượng dữ liệu thời gian thực; $N_{oper\_local\_nort}$ là số thao tác cục bộ trên các đối tượng dữ liệu không có tính thời gian; Mỗi vị trí từ xa thực hiện một tập các thao đọc trên các đối tượng dữ liệu thời gian thực ($N_{oper\_remote\_rt}$) và tập các thao tác đọc/ ghi các đối tượng dữ liệu không có tính thời gian ($N_{oper\_remote\_nort}$). Do đó, thời gian thực hiện lớn nhất của các thành viên từ xa được chọn cho việc tính thời hạn của giao dịch toàn cục, tức là $T_{process} * N_{oper\_remote\_rt} + (2T_{lock} + T_{process}) * N_{oper\_remote\_nort}$ lớn nhất trong số các thành viên ở xa.

6.2.4. Các loại xung đột dữ liệu giữa các giao dịch

Duy trì ngữ nghĩa ACID của giao dịch là vấn đề khá phức tạp, vì CSDL phân tán thời gian thực phải xử lý trên các dữ liệu phân tán, các ràng buộc thời gian của dữ liệu và ràng buộc thời gian đối với các giao dịch. Có nhiều yếu tố góp phần làm tăng sự khó khăn trong việc đáp ứng thời hạn giao dịch trong hệ thống RTOODDB. Xung đột dữ liệu là một trong những yếu tố đó. Có hai loại xung đột giữa các giao dịch phát sinh. Loại thứ nhất, xung đột xảy ra giữa các giao dịch thực hiện, được giải quyết bằng một giao thức điều khiển tương tranh để đảm bảo tính khả tuần tự trong các giao dịch phân tán; Loại thứ hai, xảy ra giữa các giao dịch ủy thác với các giao dịch thực hiện, chúng được giải quyết bằng một giao thức ủy thác để đảm bảo tính nguyên tử trong các giao dịch phân tán.

Tại cùng một thời điểm, có thể xảy ra trường hợp nhiều giao dịch có yêu cầu truy nhập đến cùng một đối tượng dữ liệu. Do đó sẽ xảy ra hiện tượng xung đột dữ liệu. Gọi T_1 là giao dịch có định danh id_1 đang

nắm giữ khóa trên đối tượng dữ liệu o và T_2 là giao dịch có định danh id_1 đang yêu cầu truy xuất đến cùng đối tượng dữ liệu o. Các tình huống xung đột giữa hai giao dịch T_2 - T_1 có thể xảy ra như sau:

(i) Xung đột ghi - đọc: T_1 đang nắm giữ khóa đọc trên đối tượng o và T_2 có yêu cầu khóa ghi trên o.

(ii) Xung đột ghi - ghi: T_1 đang nắm giữ khóa ghi trên đối tượng o và T_2 cũng có yêu cầu khóa ghi trên o.

(iii) Xung đột đọc - ghi: T_1 đang nắm giữ khóa ghi trên đối tượng o và T_2 có yêu cầu khóa đọc trên o.

(iv) Xung đột đọc - đọc: T_1 đang nắm giữ khóa đọc trên đối tượng o và T_2 cũng có yêu cầu khóa đọc trên o.

Khi xung đột dữ liệu xảy ra giữa hai giao dịch thực hiện, chúng ta sử dụng cơ chế điều khiển tương tranh nhằm giải quyết xung đột. Nếu có xung đột dữ liệu xảy ra giữa một giao dịch ủy thác và một giao dịch thực hiện, một giao thức ủy thác được sử dụng nhằm giải quyết xung đột.

KẾT LUẬN

Việc thiết kế hướng đối tượng là nền tảng để sử dụng hiệu quả các kỹ thuật CSDL và khai thác tốt cấu trúc hệ thống thông tin. Bên cạnh đó, có nhiều vấn đề liên quan đến việc mô hình hóa và thiết kế cơ sở dữ liệu thời gian thực trong nhiều công trình nghiên cứu về các vấn đề có liên quan như: cấu trúc mô hình RTOODB, kiến trúc chung của mô hình RTOODDB, các thành phần của mô hình RTOODDB, v.v...

Đối với hệ thống RTOODDB, cần duy trì tính nhất quán giữa trạng thái thực tại của đối tượng thời gian thực trong môi trường mở rộng và *ảnh* của nó - sự phản ánh của tất cả các bản sao của nó phân tán trên nhiều điểm nút. Bởi vậy, việc tạo bản sao của dữ liệu trong hệ thống RTOODDB là vấn đề được quan tâm khi thực thi hệ thống.

Ngoài ra, xử lý giao dịch cũng là yếu tố quan trọng trong hệ thống RTOODDB. Các giao dịch phải đảm bảo được yêu cầu của một hệ thống phân tán và hệ thống hướng đối tượng thời gian thực.

CÂU HỎI ÔN TẬP CHƯƠNG 6

Câu 6.1. Để quản lý các xe buýt nội đô thành phố Đà Nẵng. Anh (chị) hãy thiết kế CSDL hướng đối tượng thời gian thực để đặc tả cho các đối tượng của hệ thống.

Câu 6.2. Với CSDL ở trên, anh (chị) hãy xây dựng các phương thức:

a) Tìm một tuyến xe với điểm đầu và điểm cuối, với kết quả là số hiệu các tuyến xe buýt;

b) Xác định các xe buýt chạy trên tuyến x vào thời gian y;

c) Cho biết vận tốc tối đa trên mỗi tuyến đường;

d) Cho biết thời gian di chuyển (dự kiến) của xe buýt khi biết điểm đầu và điểm cuối.

Câu 6.3. Với kiến trúc quan hệ nhúng ở chương 3. Anh (chị) hãy chuyển đổi CSDL trên vào lược đồ quan hệ nhúng tương ứng.

Câu 6.4. Hãy liệt kê một số ứng dụng trong thực tế có thể được cài đặt bằng mô hình CSDL hướng đối tượng phân tán - thời gian thực.

Câu 6.5. So sánh hai mô hình RTOODB và RTOODDB. Từ đó, hãy đề xuất các tiêu chí để khuyến nghị việc sử dụng mô hình phù hợp cho lớp một số bài toán quản lý.

TÀI LIỆU THAM KHẢO

[1] Đoàn Văn Ban (1999), "Một số tính chất của quá trình thừa kế kiểu trong mô hình cơ sở dữ liệu hướng đối tượng", *Tạp chí Tin học và Điều khiển học*, 15(3), tr. 1-7.

[2] Đoàn Văn Ban (2000), "Tính đúng đắn của lược đồ cơ sở dữ liệu hướng đối tượng", *Tạp chí Tin học và Điều khiển học*, 16(3), tr. 7-5.

[3] Đoàn Văn Ban (2005), *Lập trình hướng đối tượng với JAVA*, Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội.

[4] Đoàn Văn Ban, Lê Mạnh Thạnh và Hoàng Bảo Hùng (2004), "Sự tương đương trong biểu diễn giữa ngôn ngữ truy vấn OQL và đại số đối tượng", *Tạp chí Tin học và Điều khiển học*, 20(3), tr. 257-269.

[5] Bierman G.M. and Trigoni A. (2000), "Towards A Formal Type System For ODMG OQL", *Technical Report 497*, University of Cambridge, Computer Laboratory.

[6] Blaha M., Premerlani W., Shen H. (1994), "Converting OO Models Into RDBMS Schema", *IEEE Software*, pp. 28-39.

[7] Braumandl R., Claussen J., Kemper A., Kossmann D. (2000), "Functional-join processing", *The VLDB Journal*, (8), pp. 156-177.

[8] Cattel R.G.G., Barry D.K. (2000), *The Object Database Standard: ODMG 3.0*, Morgan Kaufmann Publishers.

[9] Cherniack M., S. Zdonik (1998), "Changing the Rules: Transformations for Rule-Based Optimizers", *In Proc. ACM SIGMOD Int'l Conference on Management of Data*, pp. 61-72.

[10] Cheung, S. (1999), "Adding an Object-Oriented Interface to Relational Database Using Frame Model", *Proceedings of the 9th International Database Conference*, ISBN 962-937-046-8, pp. 138-154.

[11] Cho W.S., Park C.M., Whang K.Y. and So S.H. (1996), "A New Method for Estimating the Number of Objects Satisfying an Object-Oriented Query Involving Partial Participation of Classes", *Information Systems*, 21(3), pp. 253-267.

[12] Cho Wan-Sup, Han Wook-Shin, Hong Ki-Hyung and Whang Kyu-Young (2000), "Estimating Nested Selectivity in Object-Oriented Databases", *ACM*, pp. 94-101.

[13] Cluet, Sophie and Moerkotte, Guido (1995), "Nested Queries In Object Bases", *In Fifth International Workshop on Database Programming Languages*, Italy.

[14] Fong J. (1997), "Converting Relational to Object-Oriented Databases", *SIGMOD Record*, 26(1), pp. 53-64.

[15] Fong J. and Cheung S. (1999), "An Architectural Framework for Translating OODB Method to RDB Routine for ORDBMS", 2nd ACM HKPRC.

[16] Fong J. and Chitson P. (1997), "Query Translation from SQL to OQL for Database Reengineering", *International Journal of Information Technology*, 3(1), pp. 83-101.

[17] Fung C.W., Karlapalem K. and Li Q. (2002), "An Evaluation of Vertical Class Partitioning for Query Processing in Object-Oriented Databases", *IEEE Transactions on Knowledge and Data Engineering*, 14(5), pp. 1095-1118.

[18] Gardarin G., J.-R. Gruser and Z.-H. Tang (1995), "A Cost Model for Clustered Object-Oriented Databases", *Proceedings of the 21st VLDB Conference*, Switzerland, pp. 323-334.

[19] Gluche, D. and Marc H. Scholl (1993), "Physical Design in OODBMS, Department of Mathematics and Computer Science", P.O Box 5560/D188, D-78434 Konstanz, Germany.

[20] Han, Jia Liang (1998), "Optimizing Relational Queries in Connection Hypergraphs: Nested Queries, Views, and Binding Propagations", *The VLDB Journal*, 7, pp. 1-11.

[21] Hoàng Bảo Hùng (2001), *Một vài vấn đề về tối ưu hoá truy vấn trong cơ sở dữ liệu hướng đối tượng*, Luận văn tốt nghiệp Thạc sĩ chuyên ngành Công nghệ thông tin, Đại học Bách khoa, Hà Nội.

[22] Jiménez, J. Samos (1997), *Definition of external schemas and derived classes in object oriented databases*, Thesis of Doctor of Philosophy (Computer Sciences), Universitat Politécnica De Catalunya, Barcelona.

[23] Karlapalem K., Li Q. (2000), "A Framework for Class Partitioning in Object-Oriented Databases", *Distributed and Parallel Databases*, 8, pp. 333-366.

[24] Kim W. (1991), *Introduction to Object - Oriented Databases*, Massachusetts Institute of Technology, 2th Ed.

[25] Meng W., Kamada A., Chang Y (1995), "Transformation of Relational Schemas to Object - Oriented Schemas", *IEEE*, pp. 356-361.

[26] Meng W., Yu C., Kim W., Wang G., Pham T. and Dao S. (1993), "Construction of Relational Front - end for Object - Oriented Database Systems", *Proc. 9th Intl. Conf. on Data Engineering, IEEE*, pp. 476-483.

[27] Ramanathan C. (1997), *Providing Object-Oriented Access to Existing Relational Databases*, Ph.D thesis in Computer Science, in the Department of Computer Science, Mississippi State University.

[28] Riedel H. and Scholl M. H. (1997), "A Formalization of ODMG Queries", *Proc. Seventh Int'l Conf. Data Semantics* (DS-7).

[29] S. S. Khan (2009), "Prototype Architecture for Real-Time Object Oriented Distributed Database", *International Journal of Engineering and Technology*, Volume 2, Number 1.

[30] Scholl Marc H. (1992), *Physical Database Design for an Object-Oriented Database System*, University of Ulm, Department of Computer Science, Oberer Eselsberg, D-7900 ULM, Morgan Kaufman Publishers, Germany.

[31] Shekhar S., Lu C., Chawla S. and Ravada S. (2002), "Efficient Join-Index-Based Spatial-Join Processing: A Clustering Approach", *IEEE Transactions on Knowledge and Data Engineering*, 14(6), pp. 1400-1420.

[32] Lê Mạnh Thạnh, Đoàn Văn Ban và Hoàng Bảo Hùng (2004), "Sử dụng kỹ thuật lưu trữ quan hệ lồng trong thiết kế cơ sở dữ liệu vật lý cho hệ thống cơ sở dữ liệu hướng đối tượng", *Kỷ yếu Hội thảo khoa học Quốc gia, lần thứ nhất, Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR)*, Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội, tr. 305-314.

[33] Lê Mạnh Thạnh, Đoàn Văn Ban, Hoàng Bảo Hùng (2005), "Phương pháp ước lượng các truy vấn lồng trong cơ sở dữ liệu hướng đối tượng bằng siêu đồ thị kết nối", *Chuyên san Tạp chí Bưu chính Viễn thông và Công nghệ thông tin, "Các công trình nghiên cứu - Triển khai Viễn thông và Công nghệ thông tin"*, 14, tr. 43-49.

[34] Lê Mạnh Thạnh, Hoàng Bảo Hùng (2001), "Ngôn ngữ truy vấn hướng đối tượng và tối ưu hóa truy vấn trên CSDL hướng đối tượng bằng phương pháp biến đổi đại số", *Kỷ yếu Hội nghị khoa học kỷ niệm 25 năm thành lập Viện Công nghệ thông tin*, Hà Nội, tr. 175-185.

[35] Lê Mạnh Thạnh, Hoàng Bảo Hùng (2002), "Một phương pháp chuyển đổi dữ liệu từ cơ sở dữ liệu quan hệ sang cơ sở dữ liệu hướng đối tượng", *Tạp chí khoa học*, (11), Đại học Huế, tr. 35-44.

[36] Lê Mạnh Thạnh, Hoàng Bảo Hùng (2006), "Mô hình ước lượng chi phí xử lý truy vấn đối tượng trong cơ sở dữ liệu hướng đối tượng", *Kỷ yếu Hội thảo Quốc gia, lần thứ VIII, "Một số vấn đề chọn lọc về CNTT và truyền thông", chủ đề "Mã nguồn mở"*, 25/8-27/8/2005, Hải Phòng, Nhà xuất bản Khoa học và Kỹ thuật, Hà Nội, tr. 568-579.

[37] Trigoni A. (2002), "Semantic Optimization of OQL Queries", *Technical Report, Number 547*, University of Cambridge, Computer Laboratory, UCAM-CL-TR-547, ISSN 1476-2986.

[38] Trigoni A. and Bierman G.M. (2001), "Inferring the Principal Type and the Schema Requirements of an OQL Query", *In 18th British National Conference on Databases* (BNCOD), pp. 185-201.

[39] Ullman, Jeffrey D. (1999), *Nguyên lý các hệ cơ sở dữ liệu và cơ sở tri thức*, Tập 1, 2, Trần Đức Quang biên dịch, Nhà xuất bản Thống kê.

[40] Vanderberg, Scott Lee (1993), *Algebras for Object - Oriented Query Languages*, Ph.D. Dissertation, University of Winconsin-Madison.

[41] Vossen Gottfried (1994), "Formalization of OODB models", *Proceedings of 1st Workshop KRDB'94*, Germany.

[42] Wang Q. (2000), *Cost-Based Object Query Optimization*, Ph.D Dissertation, Oregon Graduate Institute of Science and Technology.

[43] Woochun Jun (2008), "A Transaction Processing Technique in Real-Time Object-Oriented Databases", *IJCSNS International Journal of Computer Science and Network 122 Security*, Vol.8 No.1.

[44] Yao S.B. (1977), "Approximating Block Accesses in Database Organizations", *Communications of the ACM*, 20(4), pp. 260-261.

[45] Yu C., Zhang Y., Meng W., Kim W., Wang G., Pham T. and Dao S. (1995), "Translation of Object-Oriented Queries to Relational Queries", *11-th IEEE International Conference on Data Engineering, Proc. of IEEE on Data Engineering*, pp. 90-97.

[46] Yu, Clement T., Meng, Weiyi (1998), *Principles of Database Query Processing for Advanced Applications*, Morgan Kaufmann Publishers, Inc. San Francisco, California.

[47] Yuan Wei Sang, H. Son, John A. Stankovic (2004), "Maintaining Data Freshness in Distributed Real-Time Databases", *In 5th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*.

CƠ SỞ DỮ LIỆU HƯỚNG ĐỐI TƯỢNG

Chịu trách nhiệm xuất bản
NGUYỄN THỊ THU HÀ
(Giám đốc - Tổng biên tập)

Biên tập : NGUYỄN TIẾN SỸ
LÊ HỒ DIỆU THẢO
TRẦN MINH QUANG
Trình bày sách : TRẦN MINH QUANG
Sửa bản in : NGUYỄN TIẾN SỸ
Thiết kế bìa : TRẦN HỒNG MINH

NHÀ XUẤT BẢN THÔNG TIN VÀ TRUYỀN THÔNG

Trụ sở chính: Số 9, ngõ 90, Nguỵ Như - Kon Tum, Q. Thanh Xuân, TP. Hà Nội
Điện thoại: 04.35772138, 04.35572139 Fax: 04.35579858
E-mail: nxb.tttt@mic.gov.vn
Website: www.nxbthongtintruyenthong.vn

Chi nhánh TP. Hồ Chí Minh: 8A đường D2, P. 25, Q. Bình Thạnh, TP. Hồ Chí Minh
Điện thoại: 08.35127750 Fax: 08.35127751
E-mail: cnsg.nxbtttt@mic.gov.vn

Chi nhánh TP. Đà Nẵng: 42 Trần Quốc Toản, Q. Hải Châu, TP. Đà Nẵng
Điện thoại: 0511.3897467 Fax: 0511.3843359
E-mail: cndn.nxbtttt@mic.gov.vn

In 500 bản, khổ 16×24 cm tại Công ty In Hải Nam
Địa chỉ nơi in: Số 18 ngách 68/53/9 Quan Hoa, Cầu Giấy, Hà Nội
Số xác nhận đăng ký xuất bản: 1549 - 2015/CXBIPH/4-370/TTTT
Số quyết định xuất bản: 162/QĐ - NXB TTTT ngày 24 tháng 6 năm 2015.
In xong nộp lưu chiểu tháng 7 năm 2015. *Mã số: HT 07 ĐM15*