



Giáo trình cơ sở dữ liệu

Biên tập bởi:

Ngô Trần Thanh Thảo

Giáo trình cơ sở dữ liệu

Biên tập bởi:

Ngô Trần Thanh Thảo

Các tác giả:

Ngô Trần Thanh Thảo

Phiên bản trực tuyến:

<http://voer.edu.vn/c/3eaa132c>

MỤC LỤC

1. Các khái niệm cơ bản
 2. Những khái niệm cơ bản
 3. Những cách tiếp cận một cơ sở dữ liệu
 4. Mô hình dữ liệu quan hệ của e.f.codd
 5. Ràng buộc toàn vẹn
 6. Ràng buộc toàn vẹn các loại
 7. Ngôn ngữ đại số quan hệ
 8. Ngôn ngữ đại số quan hệ (tiếp theo)
 9. Ngôn ngữ truy vấn cơ sở dữ liệu SQL
 10. Ngôn ngữ truy vấn CSDL SQL (tiếp theo)
 11. Ngôn ngữ tân từ
 12. Ngôn ngữ tân từ (tiếp theo)
 13. Tối ưu hóa câu hỏi
 14. Tối ưu hóa câu hỏi (tiếp theo)
- Tham gia đóng góp

Các khái niệm cơ bản

Dẫn nhập - Tại sao cần phải có một CSDL

Trong những năm gần đây, thuật ngữ "CƠ SỞ DỮ LIỆU" (Tiếng Anh là *DataBase*, viết tắt tiếng Việt là CSDL) đã trở nên khá quen thuộc không chỉ riêng với những người làm Tin học mà còn đối với cả những người làm trong nhiều lĩnh vực khác như Thống kê, Kinh tế, Quản lý Doanh nghiệp v.v... Các ứng dụng của Tin học vào công tác quản lý ngày càng nhiều hơn và càng đa dạng hơn. Có thể nói hầu hết các lĩnh vực kinh tế, xã hội, giáo dục, y tế v.v... đều đã ứng dụng các thành tựu mới của Tin học vào phục vụ công tác chuyên môn của mình. Chính vì lẽ đó mà ngày càng nhiều người quan tâm đến lĩnh vực thiết kế và xây dựng các CSDL.

Mục đích của chương I chỉ đơn giản là cung cấp các khái niệm cơ bản về CSDL để các học viên có một cái nhìn ban đầu về một cơ sở dữ liệu và một hệ quản trị CSDL. Trước hết chúng ta sẽ tìm hiểu lý do tại sao cần phải có một CSDL.

Hệ thống các tập tin cổ điển (File System)

Cho đến nay vẫn còn một số đơn vị kinh tế, hành chính sự nghiệp v.v... sử dụng mô hình hệ thống các tập tin cổ điển: chúng được tổ chức riêng rẽ, phục vụ cho một mục đích của một đơn vị hay một đơn vị con trực thuộc cụ thể. Chẳng hạn, hãy xét ví dụ sau:

Ví dụ 1.1:

Tại một công ty người ta trang bị máy vi tính cho tất cả các phòng, ban nghiệp vụ. Bộ phận Văn phòng sử dụng máy tính để soạn thảo các văn bản báo cáo bằng MicroSoft Word do thủ trưởng yêu cầu về tình hình hoạt động của đơn vị trong đó có chỉ tiêu về tổng số công nhân viên chức chia theo trình độ chuyên môn được đào tạo. Phòng Kế toán sử dụng máy tính để tính lương và in danh sách lương của từng bộ phận trong đơn vị dựa trên danh sách cán bộ viên chức cùng hệ số lương và các hệ số phụ cấp của họ do phòng Tổ chức cung cấp. Thông tin mà phòng Kế toán quản lý và khai thác là: Họ và Tên, Hệ số lương, Hệ số phụ cấp, Phụ cấp khác của các công nhân viên chức (CNVC) xếp theo từng phòng ban và sử dụng công cụ văn phòng là MicroSoft Excel. Phòng Tổ chức quản lý thông tin lý lịch của CNVC chi tiết hơn gồm Họ CNVC, Tên CNVC (để riêng thành một cột "Tên" để tiện sắp xếp theo vần Alphabet), Bí danh, Giới tính, Ngày sinh, Ngày tuyển dụng, Hoàn cảnh gia đình, Quá trình được đào tạo, Hệ số lương, Hệ số phụ cấp, Ngày xếp lương trên ... nhưng thiếu thông tin về Phụ cấp khác của CNVC. Phần mềm được sử dụng để quản lý là FoxPro for Windows.

Trong khi đó, tại Tổng công ty của họ, các phòng ban nghiệp vụ cũng được trang bị vi tính. Phòng Tổ chức cán bộ tại Tổng công ty sử dụng phần mềm MicroSoft Access để quản lý CNVC gồm các cán bộ chủ chốt từ trường phó phòng, quản đốc và phó quản đốc xí nghiệp trở lên của các công ty con trực thuộc. Thông tin quản lý tại đây cũng giống như thông tin quản lý tại phòng tổ chức của công ty con.

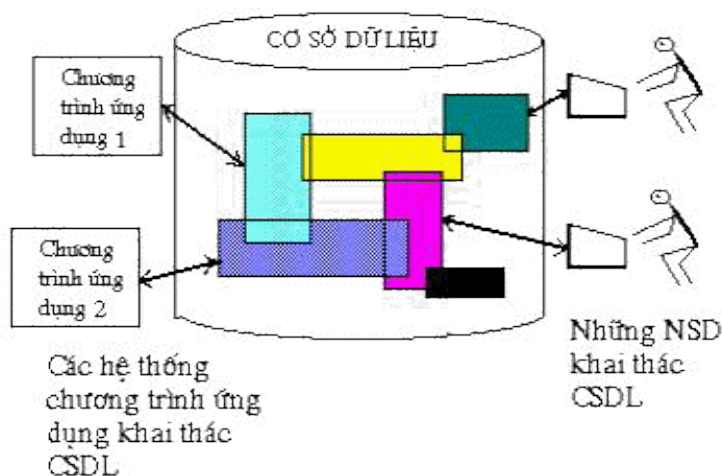
Nhận xét :

Ưu điểm:

- Việc xây dựng hệ thống các tập tin riêng tại từng đơn vị quản lý ít tốn thời gian bởi khối lượng thông tin cần quản lý và khai thác là nhỏ, không đòi hỏi đầu tư vật chất và chất xám nhiều, do đó triển khai ứng dụng nhanh.

Định nghĩa một CSDL.

Cơ sở dữ liệu là một hệ thống các thông tin có cấu trúc được lưu trữ trên các thiết bị lưu trữ thông tin thứ cấp (như băng từ, đĩa từ ...) để có thể thỏa mãn yêu cầu khai thác thông tin đồng thời của nhiều người sử dụng hay nhiều chương trình ứng dụng với nhiều mục đích khác nhau.



Hình 1.2.1 Sơ đồ tổng quát về một cơ sở dữ liệu

Trong định nghĩa này cần nhấn mạnh những khía cạnh của định nghĩa được lưu ý qua các từ gạch chân. Trước hết, CSDL phải là một tập hợp các thông tin mang tính hệ thống chứ không phải là các thông tin rời rạc, không có mối quan hệ với nhau. Các thông tin này phải có cấu trúc và tập hợp các thông tin này phải có khả năng đáp ứng các nhu cầu

khai thác của nhiều người sử dụng một cách đồng thời. Đó cũng chính là các đặc trưng của CSDL.

Rõ ràng, ưu điểm nổi bật của CSDL là:

Giảm sự trùng lặp thông tin xuống mức thấp nhất và do đó bảo đảm được tính nhất quán và toàn vẹn dữ liệu.

Đảm bảo dữ liệu có thể được truy xuất theo nhiều cách khác nhau.

Khả năng chia sẻ thông tin cho nhiều người sử dụng và nhiều ứng dụng khác nhau.

Tuy nhiên, để đạt được các ưu điểm trên, CSDL đặt ra những vấn đề cần phải giải quyết. Đó là:

Tính chủ quyền của dữ liệu. Do tính chia sẻ của CSDL nên tính chủ quyền của dữ liệu có thể bị lu mờ và làm mờ nhạt tinh thần trách nhiệm, được thể hiện trên vấn đề an toàn dữ liệu, khả năng biểu diễn các mối liên hệ ngữ nghĩa của dữ liệu, và tính chính xác của dữ liệu. Điều này có nghĩa là người khai thác CSDL phải có nghĩa vụ cập nhật các thông tin mới nhất của CSDL.

Tính bảo mật và quyền khai thác thông tin của người sử dụng. Do có nhiều người được phép khai thác CSDL một cách đồng thời nên cần phải có một cơ chế bảo mật và phân quyền hạn khai thác CSDL. Các hệ điều hành nhiều người sử dụng hay hệ điều hành mạng cục bộ (Novell Netware, Windows For WorkGroup, WinNT, ...) đều có cung cấp cơ chế này.

Tranh chấp dữ liệu. Nhiều người được phép truy nhập vào cùng một tài nguyên dữ liệu (Data Source) của CSDL với những mục đích khác nhau: Xem, thêm, xóa hoặc sửa dữ liệu. Cần phải có một cơ chế ưu tiên truy nhập dữ liệu cũng như cơ chế giải quyết tình trạng khóa chết (DeadLock) trong quá trình khai thác cạnh tranh. Cơ chế ưu tiên có thể được thực hiện bằng việc cấp quyền (hay mức độ) ưu tiên cho từng người khai thác - người nào được cấp quyền hạn ưu tiên cao hơn thì được ưu tiên truy nhập dữ liệu trước; theo biến có hoặc loại truy nhập - quyền đọc được ưu tiên trước quyền ghi dữ liệu; dựa trên thời điểm truy nhập - ai có yêu cầu truy xuất trước thì có quyền truy nhập dữ liệu trước; hoặc theo cơ chế lập lịch truy xuất hay các cơ chế khóa [7]...

Đảm bảo dữ liệu khi có sự cố. Việc quản lý dữ liệu tập trung có thể làm tăng khả năng mất mát hoặc sai lệch thông tin khi có sự cố như mất điện đột xuất, một phần đĩa lưu trữ CSDL bị hư v.v... Một số hệ điều hành mạng có cung cấp dịch vụ sao lưu ảnh đĩa cứng (cơ chế sử dụng đĩa cứng dự phòng - RAID), tự động kiểm tra và khắc phục lỗi khi có sự cố, tuy nhiên, bên cạnh dịch vụ của hệ điều hành, để đảm bảo CSDL luôn luôn ổn định, một CSDL nhất thiết phải có một cơ chế khôi phục dữ liệu khi các sự cố bất ngờ xảy ra.

Các đối tượng sử dụng CSDL:

Những người sử dụng CSDL không chuyên về lĩnh vực tin học và CSDL, do đó CSDL cần có các công cụ để cho những người sử dụng không chuyên có thể sử dụng để khai thác CSDL khi cần thiết.

Các chuyên viên tin học biết khai thác CSDL. Những người này có thể xây dựng các ứng dụng khác nhau phục vụ cho các mục đích khác nhau trên CSDL.

Những người quản trị CSDL, đó là những người hiểu biết về tin học, về các hệ quản trị CSDL và hệ thống máy tính. Họ là người tổ chức CSDL (khai báo cấu trúc CSDL, ghi nhận các yêu cầu bảo mật cho các dữ liệu cần bảo vệ ...) do đó họ phải nắm rõ các vấn đề kỹ thuật về CSDL để có thể phục hồi dữ liệu khi có sự cố. Họ là những người cấp quyền hạn khai thác CSDL, do vậy họ có thể giải quyết được các vấn đề tranh chấp dữ liệu, nếu có.

- Thông tin được khai thác chỉ phục vụ cho mục đích hẹp nên khả năng đáp ứng nhanh chóng, kịp thời.

Nhược điểm:

- Do thông tin được tổ chức ở mỗi phòng ban mỗi khác, cũng như phần mềm công cụ để triển khai mỗi nơi cũng rất khác nhau nên sự phối hợp tổ chức và khai thác ở các phòng ban là khó khăn. Thông tin ở phòng ban này không sử dụng được cho phòng ban khác, tại đơn vị con với đơn vị cấp trên. Cùng một thông tin được nhập vào máy tại nhiều nơi khác nhau gây ra lãng phí công sức nhập tin và không gian lưu trữ trên các vật mang tin. Sự trùng lặp thông tin có thể dẫn đến tình trạng không nhất quán dữ liệu. Chẳng hạn, nhân viên Nguyễn Văn Quang được ghi đầy đủ ở phòng Tổ chức, nhưng tại phòng Kế toán chỉ ghi tắt là Nguyễn v Quang.

Thông tin được tổ chức ở nhiều nơi nên việc cập nhật cũng dễ làm mất tính nhất quán dữ liệu. Một cán bộ chủ chốt của công ty có thay đổi về hoàn cảnh gia đình (mới cưới vợ / lấy chồng, sinh thêm con ...) có thể được cập nhật ngay tại đơn vị nhưng sau một thời gian mới được cập nhật tại Tổng công ty.

Do hệ thống được tổ chức thành các hệ thống file riêng lẻ nên thiếu sự chia sẻ thông tin giữa các nơi. Việc kết nối các hệ thống này hay việc nâng cấp ứng dụng sẽ là rất khó khăn.

Qua phân tích trên chúng ta nhận thấy việc tổ chức dữ liệu theo hệ thống các tập tin có nhiều nhược điểm. Việc xây dựng một hệ thống tin đảm bảo được tính chất nhất quán

dữ liệu, không trùng lặp thông tin mà vẫn đáp ứng được nhu cầu khai thác đồng thời của tất cả các phòng ban ở công ty và tổng công ty là thực sự cần thiết.

Hệ phần mềm quản trị CSDL.

Để giải quyết tốt tất cả các vấn đề đặt ra cho một CSDL như đã nêu trên: tính chủ quyền, cơ chế bảo mật hay phân quyền hạn khai thác CSDL, giải quyết tranh chấp trong quá trình truy nhập dữ liệu, và phục hồi dữ liệu khi có sự cố ... thì cần phải có một hệ thống các phần mềm chuyên dụng. Hệ thống các phần mềm đó được gọi là hệ quản trị CSDL (tiếng Anh là DataBase Management System - DBMS). Đó là các công cụ hỗ trợ tích cực cho các nhà phân tích & thiết kế CSDL và những người khai thác CSDL. Cho đến nay có khá nhiều hệ quản trị CSDL mạnh được đưa ra thị trường như: Visual FoxPro, MicroSoft Access, SQL-Server, DB2, Sybase, Paradox, Informix, Oracle... với các chất lượng khác nhau.

Mỗi hệ quản trị CSDL đều được cài đặt dựa trên một mô hình dữ liệu cụ thể. Hầu hết các hệ quản trị CSDL hiện nay đều dựa trên mô hình quan hệ (Xem chương III). Dù dựa trên mô hình dữ liệu nào, một hệ quản trị CSDL cũng phải có:

Ngôn ngữ giao tiếp giữa người sử dụng (NSD) và CSDL, bao gồm:

Ngôn ngữ mô tả dữ liệu (*Data Definition Language - DDL*) để cho phép khai báo cấu trúc của CSDL, khai báo các mối liên hệ của dữ liệu (*Data RelationShip*) và các quy tắc (*Rules, Constraint*) quản lý áp đặt lên các dữ liệu đó.

Ngôn ngữ thao tác dữ liệu (*Data Manipulation Language - DML*) cho phép người sử dụng có thể thêm (*Insert*), xóa (*Delete*), sửa (*Update*) dữ liệu trong CSDL.

Ngôn ngữ truy vấn dữ liệu, hay ngôn ngữ hỏi đáp có cấu trúc (*Structured Query Language - SQL*) cho phép những người khai thác CSDL (chuyên nghiệp hoặc không chuyên) sử dụng để truy vấn các thông tin cần thiết trong CSDL.

Ngôn ngữ quản lý dữ liệu (*Data Control Language - DCL*) cho phép những người quản trị hệ thống thay đổi cấu trúc của các bảng dữ liệu, khai báo bảo mật thông tin và cấp quyền hạn khai thác CSDL cho người sử dụng.

Từ điển dữ liệu (*Data Dictionary*) dùng để mô tả các ánh xạ liên kết, ghi nhận các thành phần cấu trúc của CSDL, các chương trình ứng dụng, mật mã, quyền hạn sử dụng v.v....

Có biện pháp bảo mật tốt khi có yêu cầu bảo mật.

Cơ chế giải quyết vấn đề tranh chấp dữ liệu. Mỗi hệ quản trị CSDL cũng có thể cài đặt một cơ chế riêng để giải quyết các vấn đề này. Một số biện pháp sau đây được sử dụng:

- Cấp quyền ưu tiên cho từng người sử dụng (người quản trị CSDL thực hiện).
- Đánh dấu yêu cầu truy xuất dữ liệu, phân chia thời gian, người nào có yêu cầu trước thì có quyền truy xuất dữ liệu trước.

Hệ quản trị CSDL cũng phải có cơ chế sao lưu (Backup) và phục hồi (Restore) dữ liệu khi có sự cố xảy ra. Điều này có thể được thực hiện bằng cách:

- Định kỳ kiểm tra CSDL, sau một thời gian nhất định hệ quản trị CSDL sẽ tự động tạo ra một bản sao CSDL. Cách này hơi tốn kém, nhất là đối với các CSDL lớn.
- Tạo nhật ký (*LOG*) thao tác CSDL. Mỗi thao tác trên CSDL đều được hệ thống ghi lại, khi có sự cố xảy ra thì tự động lần ngược lại (*RollBack*) để phục hồi CSDL.

Hệ quản trị CSDL phải cung cấp một giao diện (Interface) tốt, dễ sử dụng, dễ hiểu cho những người sử dụng không chuyên.

Ngoài ra, một hệ quản trị CSDL phải đáp ứng được một yêu cầu rất quan trọng, đó là bảo đảm tính độc lập giữa dữ liệu và chương trình: Khi có sự thay đổi dữ liệu (như sửa đổi cấu trúc trữ các bảng dữ liệu, thêm các chỉ mục (Index) ...) thì các chương trình ứng dụng (Application) đang chạy trên CSDL đó vẫn không cần phải được viết lại, hay cũng không làm ảnh hưởng đến những NSD khác.

Vài nét về quá trình phát triển các hệ quản trị CSDL:

Trải qua gần 40 năm nghiên cứu và cài đặt ứng dụng, các hệ quản trị CSDL không ngừng được phát triển. Các hệ quản trị CSDL đầu tiên ra đời vào đầu những năm 60 của thế kỷ 20 dựa trên mô hình dữ liệu phân cấp và mạng, trong số đó có hệ quản trị CSDL có tên là IMS của hãng IBM dựa trên mô hình dữ liệu phân cấp.

Năm 1976, hệ quản trị CSDL đầu tiên dựa trên mô hình dữ liệu quan hệ của hãng IBM mang tên System-R ra đời. Từ năm 1980 hãng IBM cho ra đời hệ quản trị CSDL trên các máy Main Frame mang tên DB2, tiếp theo là các hệ quản trị CSDL Dbase, Sybase, Oracle, Informix, SQL-Server ...

Từ những năm 1990 người ta bắt đầu cố gắng xây dựng các hệ quản trị CSDL hướng đối tượng (*Oriented Object DataBase Management System*) như Orion, Illustra, Itasca, ... Tuy nhiên hầu hết các hệ này đều vẫn là quan hệ - hướng đối tượng, nghĩa là, xét về

bản chất, chúng vẫn dựa trên nền tảng của mô hình quan hệ. Hệ quản trị CSDL hướng đối tượng thuần nhất có thể là hệ ODMG ra đời vào năm 1996.

Hệ phần mềm quản trị CSDL.

Để giải quyết tốt tất cả các vấn đề đặt ra cho một CSDL như đã nêu trên: tính chủ quyền, cơ chế bảo mật hay phân quyền hạn khai thác CSDL, giải quyết tranh chấp trong quá trình truy nhập dữ liệu, và phục hồi dữ liệu khi có sự cố ... thì cần phải có một hệ thống các phần mềm chuyên dụng. Hệ thống các phần mềm đó được gọi là hệ quản trị CSDL (tiếng Anh là DataBase Management System - DBMS). Đó là các công cụ hỗ trợ tích cực cho các nhà phân tích & thiết kế CSDL và những người khai thác CSDL. Cho đến nay có khá nhiều hệ quản trị CSDL mạnh được đưa ra thị trường như: Visual FoxPro, MicroSoft Access, SQL-Server, DB2, Sybase, Paradox, Informix, Oracle... với các chất lượng khác nhau.

Mỗi hệ quản trị CSDL đều được cài đặt dựa trên một mô hình dữ liệu cụ thể. Hầu hết các hệ quản trị CSDL hiện nay đều dựa trên mô hình quan hệ (Xem chương III). Dù dựa trên mô hình dữ liệu nào, một hệ quản trị CSDL cũng phải có:

Ngôn ngữ giao tiếp giữa người sử dụng (NSD) và CSDL, bao gồm:

- Ngôn ngữ mô tả dữ liệu (*Data Definition Language - DDL*) để cho phép khai báo cấu trúc của CSDL, khai báo các mối liên hệ của dữ liệu (*Data RelationShip*) và các quy tắc (*Rules, Constraint*) quản lý áp đặt lên các dữ liệu đó.
- Ngôn ngữ thao tác dữ liệu (*Data Manipulation Language - DML*) cho phép người sử dụng có thể thêm (*Insert*), xóa (*Delete*), sửa (*Update*) dữ liệu trong CSDL.
- Ngôn ngữ truy vấn dữ liệu, hay ngôn ngữ hỏi đáp có cấu trúc (*Structured Query Language - SQL*) cho phép những người khai thác CSDL (chuyên nghiệp hoặc không chuyên) sử dụng để truy vấn các thông tin cần thiết trong CSDL.
- Ngôn ngữ quản lý dữ liệu (*Data Control Language - DCL*) cho phép những người quản trị hệ thống thay đổi cấu trúc của các bảng dữ liệu, khai báo bảo mật thông tin và cấp quyền hạn khai thác CSDL cho người sử dụng.

Từ điển dữ liệu (*Data Dictionary*) dùng để mô tả các ánh xạ liên kết, ghi nhận các thành phần cấu trúc của CSDL, các chương trình ứng dụng, mật mã, quyền hạn sử dụng v.v....

Có biện pháp bảo mật tốt khi có yêu cầu bảo mật.

Cơ chế giải quyết vấn đề tranh chấp dữ liệu. Mỗi hệ quản trị CSDL cũng có thể cài đặt một cơ chế riêng để giải quyết các vấn đề này. Một số biện pháp sau đây được sử dụng:

- Cấp quyền ưu tiên cho từng người sử dụng (người quản trị CSDL thực hiện).
- Đánh dấu yêu cầu truy xuất dữ liệu, phân chia thời gian, người nào có yêu cầu trước thì có quyền truy xuất dữ liệu trước.

Hệ quản trị CSDL cũng phải có cơ chế sao lưu (Backup) và phục hồi (Restore) dữ liệu khi có sự cố xảy ra. Điều này có thể được thực hiện bằng cách:

- Định kỳ kiểm tra CSDL, sau một thời gian nhất định hệ quản trị CSDL sẽ tự động tạo ra một bản sao CSDL. Cách này hơi tốn kém, nhất là đối với các CSDL lớn.
- Tạo nhật ký (*LOG*) thao tác CSDL. Mỗi thao tác trên CSDL đều được hệ thống ghi lại, khi có sự cố xảy ra thì tự động lần ngược lại (*RollBack*) để phục hồi CSDL.

Hệ quản trị CSDL phải cung cấp một giao diện (Interface) tốt, dễ sử dụng, dễ hiểu cho những người sử dụng không chuyên.

Ngoài ra, một hệ quản trị CSDL phải đáp ứng được một yêu cầu rất quan trọng, đó là bảo đảm tính độc lập giữa dữ liệu và chương trình: Khi có sự thay đổi dữ liệu (như sửa đổi cấu trúc các bảng dữ liệu, thêm các chỉ mục (Index) ...) thì các chương trình ứng dụng (Application) đang chạy trên CSDL đó vẫn không cần phải được viết lại, hay cũng không làm ảnh hưởng đến những NSD khác.

Vài nét về quá trình phát triển các hệ quản trị CSDL:

Trải qua gần 40 năm nghiên cứu và cài đặt ứng dụng, các hệ quản trị CSDL không ngừng được phát triển. Các hệ quản trị CSDL đầu tiên ra đời vào đầu những năm 60 của thế kỷ 20 dựa trên mô hình dữ liệu phân cấp và mạng, trong số đó có hệ quản trị CSDL có tên là IMS của hãng IBM dựa trên mô hình dữ liệu phân cấp.

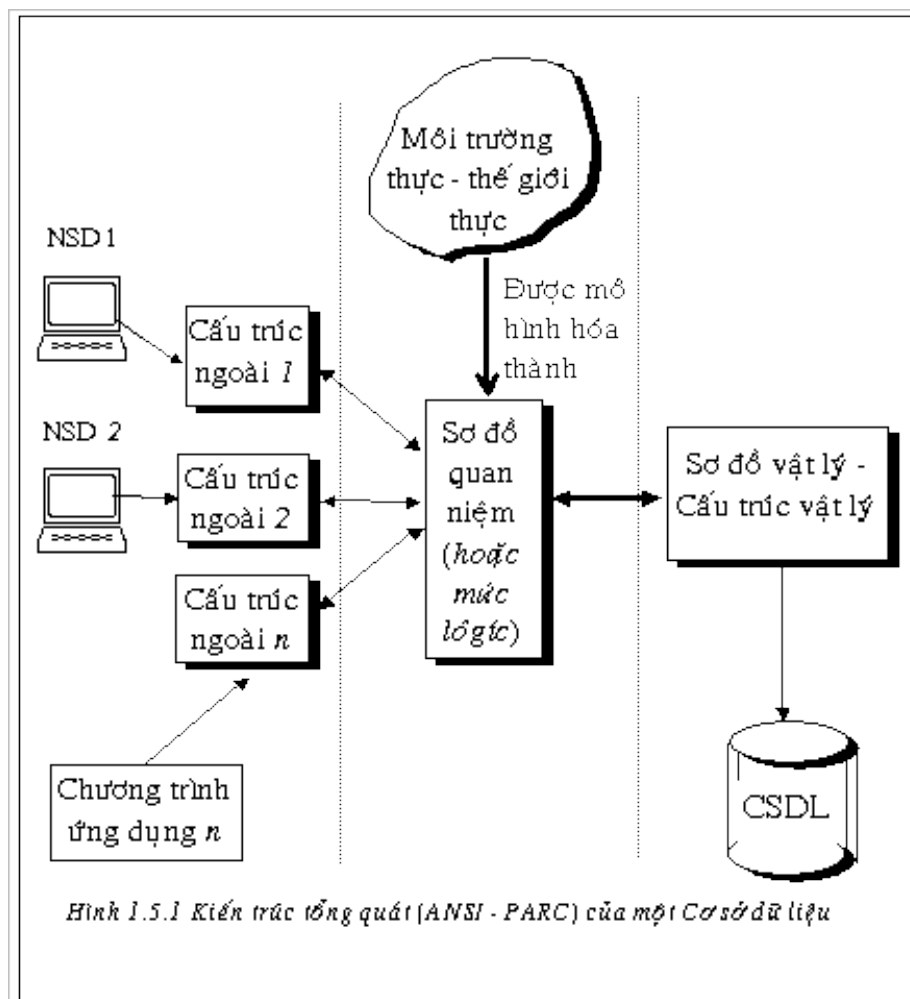
Năm 1976, hệ quản trị CSDL đầu tiên dựa trên mô hình dữ liệu quan hệ của hãng IBM mang tên System-R ra đời. Từ năm 1980 hãng IBM cho ra đời hệ quản trị CSDL trên các máy Main Frame mang tên DB2, tiếp theo là các hệ quản trị CSDL Dbase, Sybase, Oracle, Informix, SQL-Server ...

Từ những năm 1990 người ta bắt đầu cố gắng xây dựng các hệ quản trị CSDL hướng đối tượng (*Oriented Object DataBase Management System*) như Orion, Illustra, Itasca, ... Tuy nhiên hầu hết các hệ này đều vẫn là quan hệ - hướng đối tượng, nghĩa là, xét về

bản chất, chúng vẫn dựa trên nền tảng của mô hình quan hệ. Hệ quản trị CSDL hướng đối tượng thuần nhất có thể là hệ ODMG ra đời vào năm 1996.

Các mức biểu diễn một CSDL.

Theo kiến trúc ANSI-PARC, một CSDL có 3 mức biểu diễn: Mức trong (còn gọi là mức vật lý - *Physical*), mức quan niệm (*Conception* hay *Logical*) và mức ngoài.



Mức trong:

Đây là mức lưu trữ CSDL. Tại mức này, vấn đề cần giải quyết là, dữ liệu gì và được lưu trữ như thế nào? ở đâu (đĩa từ, băng từ, track, sector ... nào)? Cần các chỉ mục gì? Việc truy xuất là tuần tự (*Sequential Access*) hay ngẫu nhiên (*Random Access*) đối với từng loại dữ liệu.

Những người hiểu và làm việc với CSDL tại mức này là người quản trị CSDL (*Administrator*), những người sử dụng (NSD) chuyên môn.

Mức quan niệm:

Tại mức này sẽ giải quyết cho câu hỏi CSDL cần phải lưu giữ bao nhiêu loại dữ liệu? đó là những dữ liệu gì? Mối quan hệ giữa các loại dữ liệu này như thế nào?

Từ thế giới thực (*Real Universe*) các chuyên viên tin học qua quá trình khảo sát và phân tích, cùng với những người sẽ đảm nhận vai trò quản trị CSDL, sẽ xác định được những loại thông tin gì được cho là cần thiết phải đưa vào CSDL, đồng thời mô tả rõ mối liên hệ giữa các thông tin này. Có thể nói cách khác, CSDL mức quan niệm là một sự biểu diễn trừu tượng CSDL mức vật lý; hoặc ngược lại, CSDL vật lý là sự cài đặt cụ thể của CSDL mức quan niệm.

Ví dụ 1.2:

Người ta muốn xây dựng một hệ quản trị CSDL để quản lý các nhân viên của một công ty. Môi trường (*thế giới thực*) của công ty ở đây gồm có các phòng ban (*Department*) - mỗi phòng ban có một tên gọi khác nhau, một địa chỉ trụ sở chính (*Location*), các số điện thoại (*Telephone*) để liên lạc, có một người làm trưởng phòng ban, hàng năm được cấp một khoản kinh phí để hoạt động (*Expense Budget*), và phải đạt một doanh thu (*Revenue Budget*). Để tránh viết tên phòng ban dài dễ dẫn đến viết sai, người ta thường đặt cho mỗi phòng ban một giá trị số (gọi là số hiệu phòng ban - *Department Number*) và sử dụng số hiệu này để xác định tên và các thông tin khác của nó.

Công ty có một số công việc có thể sắp xếp cho các nhân viên trong công ty. Để thuận lợi cho việc theo dõi công việc cũng như trong công tác tuyển chọn nhân viên mới, người ta lập thành một bảng các công việc (*JOBS*) gồm các thông tin: tên tắt công việc (*Job*), tên công việc (*Job Name*), mức lương tối thiểu (*Min Salary*) và tối đa (*Max Salary*) của công việc này và cho biết công việc này cần có người lãnh đạo không. Một công việc có thể có nhiều người cùng làm.

Mỗi phòng ban có thể có từ 1 đến nhiều nhân viên (*Employee*). Mỗi nhân viên có một tên gọi, một công việc làm (*Job*), một khoản tiền lương hàng tháng (*Salary*), số hiệu phòng ban mà anh ta đang công tác. Nếu muốn, người ta có thể theo dõi thêm các thông tin khác như ngày sinh (*Birth Day*), giới tính (*Sex*) v.v... Để tránh viết tên nhân viên dài dễ dẫn đến sai sót, mỗi nhân viên có thể được gán cho một con số duy nhất, gọi là mã số nhân viên (*EmpNo*).

Nếu yêu cầu quản lý của công ty chỉ dừng ở việc theo dõi danh sách nhân viên trong từng phòng ban cùng các công việc của công ty thì cần 3 loại thông tin: Phòng ban (*DEPARTMENT*), Công việc (*JOBS*) và Nhân viên (*EMPLOYEE*) với các thông tin như

trên là đủ. Có thể công ty có thêm yêu cầu quản lý cả quá trình tuyển dụng và nâng lương thì cần có thêm một (hoặc một số) loại thông tin về quá trình: Mã số nhân viên, lần thay đổi, thời gian bắt đầu và kết thúc sự thay đổi, mức lương, .v.v...

Từ môi trường thế giới thực, xuất phát từ nhu cầu quản lý, việc xác định các loại thông tin cần lưu trữ và các mối quan hệ giữa các thông tin đó như thế nào ... đó chính là công việc ở mức quan niệm.

Mức ngoài.

Đó là mức của người sử dụng và các chương trình ứng dụng. Làm việc tại mức này có các nhà chuyên môn, các kỹ sư tin học và những người sử dụng không chuyên.

Mỗi người sử dụng hay mỗi chương trình ứng dụng có thể được "nhìn" (*View*) CSDL theo một góc độ khác nhau. Có thể "nhìn" thấy toàn bộ hay chỉ một phần hoặc chỉ là các thông tin tổng hợp từ CSDL hiện có. Người sử dụng hay chương trình ứng dụng có thể hoàn toàn không được biết về cấu trúc tổ chức lưu trữ thông tin trong CSDL, thậm chí ngay cả tên gọi của các loại dữ liệu hay tên gọi của các thuộc tính. Họ chỉ có thể làm việc trên một phần CSDL theo cách "nhìn" do người quản trị hay chương trình ứng dụng quy định, gọi là khung nhìn (*View*).

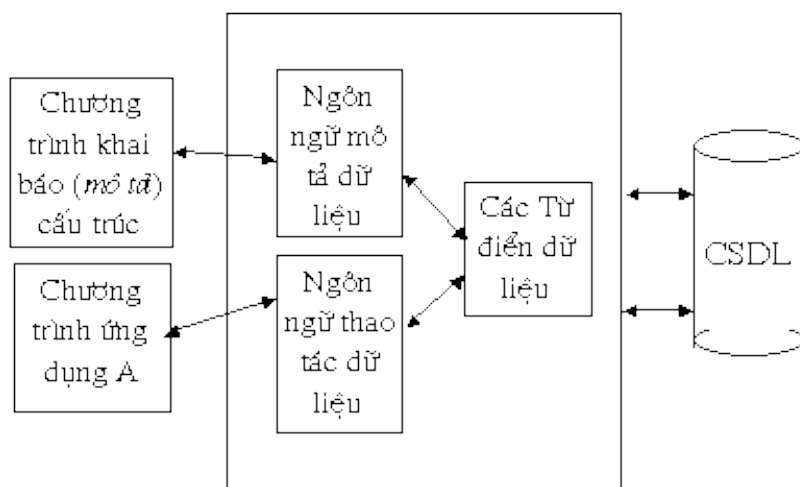
Ví dụ 1.3:

Cũng ví dụ trên, Phòng Tổ chức nhân sự giờ đây còn quản lý thêm cả các thông tin chi tiết trong lý lịch của nhân viên trong công ty: quá trình đào tạo chuyên môn kỹ thuật - kinh tế - chính trị - quản lý Nhà nước, quá trình được khen thưởng, các lần bị kỷ luật, quá trình hoạt động Cách mạng bị địch bắt - bị tù đầy, quá trình công tác, quá trình nâng lương, sơ lược tiểu sử cha mẹ - anh chị em ruột - vợ chồng - con v.v... Rõ ràng rằng, Phòng Kế toán có thể chỉ được nhìn thấy CSDL là danh sách nhân viên đang làm các công việc cụ thể trong từng Phòng ban với các mức lương thỏa thuận, mà không được thấy lý lịch của các nhân viên. Lãnh đạo công ty có thể chỉ cần "nhìn" thấy số lượng nhân viên, tổng số lương phải trả và ai là người lãnh đạo của từng Phòng ban. Trong khi đó ngay cả những người trong Phòng Tổ chức nhân sự cũng có thể có người được xem lý lịch của tất cả cán bộ, công nhân viên của công ty, nhưng cũng có thể có người chỉ được xem lý lịch của những cán bộ, công nhân viên với mức lương từ xx đồng trở xuống...

Như vậy, cấu trúc CSDL vật lý (mức trong) và mức quan niệm thì chỉ có một; nhưng tại mức ngoài, mức của các chương trình ứng dụng và người sử dụng trực tiếp CSDL, thì có thể có rất nhiều cấu trúc ngoài tương ứng.

Những khái niệm cơ bản

Sơ đồ tổng quát của một hệ quản trị cơ sở dữ liệu



Hình 1.6.1 Sơ đồ tổng quát của một hệ quản trị CSDL

Hình 1.6.1 minh họa sơ đồ tổng quát của một hệ quản trị CSDL. Chúng ta thấy có 3 mức: mức chương trình khai báo cấu trúc và chương trình ứng dụng; mức mô tả CSDL, thao tác CSDL và các từ điển dữ liệu; và mức CSDL.

Mỗi hệ quản trị CSDL có một ngôn ngữ khai báo (hay mô tả: *Data Definition Language-DDL*) cấu trúc CSDL riêng. Những người thiết kế và quản trị CSDL thực hiện các công việc khai báo cấu trúc CSDL.

Các chương trình khai báo cấu trúc CSDL được viết bằng ngôn ngữ mà hệ quản trị CSDL cho phép. Hai công việc khai báo là khai báo cấu trúc lôgic (đó là việc khai báo các loại dữ liệu và các mối liên hệ giữa các loại dữ liệu đó, cùng các ràng buộc toàn vẹn dữ liệu - RBTV) và khai báo vật lý (dữ liệu được lưu trữ theo dạng nào?, có bao nhiêu chỉ mục?).

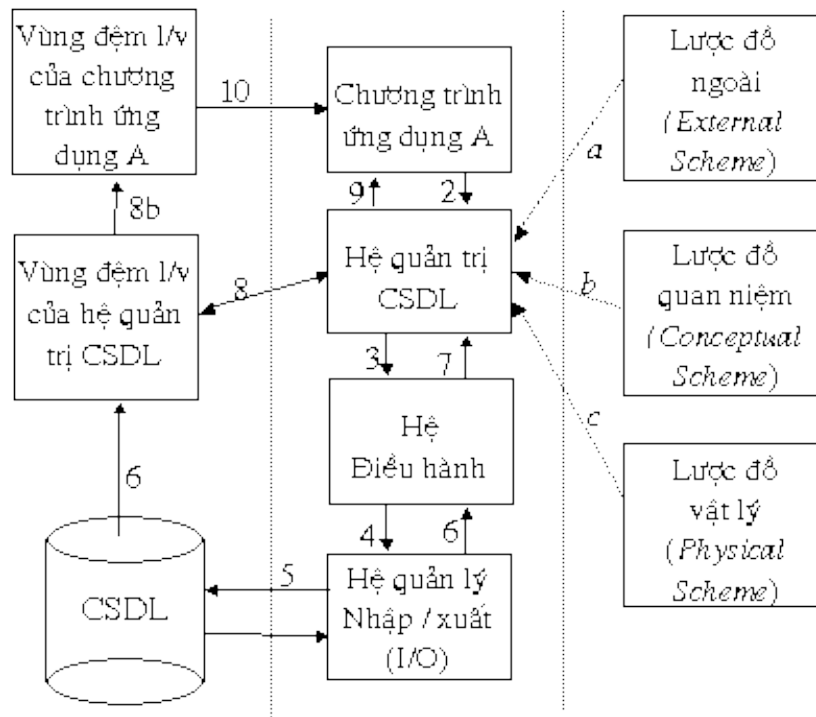
Các chương trình ứng dụng được viết bằng ngôn ngữ thao tác CSDL (*Data Manipulation Language - DML*) với mục đích:

- Truy xuất dữ liệu
- Cập nhật dữ liệu (thêm, xóa, sửa dữ liệu)

- Khai thác dữ liệu
- Ngôn ngữ thao tác CSDL còn được sử dụng cho những NSD thao tác trực tiếp với CSDL.

Từ điển dữ liệu (*Data Dictionary - DD*) là một CSDL của hệ quản trị CSDL sử dụng để lưu trữ cấu trúc CSDL, các thông tin bảo mật, bảo đảm an toàn dữ liệu và các cấu trúc ngoài. Những người đã làm quen với hệ quản trị CSDL của MicroSoft Access có thể thấy các từ điển dữ liệu này thông qua các bảng (*Table*) có tên bắt đầu bằng chữ MSys như MSysACEs, MSysColumn, MSysIMEXColumn, MSysIMEXSpecs, MSysIndexes, MSysMacros, MSysObjects, MSysQueries, MSysRelationships ... Từ điển dữ liệu còn được gọi là Siêu CSDL (*Meta-DataBase*).

(*) Quá trình hoạt động của một chương trình ứng dụng thông qua các tầng của CSDL:



Hình 1.6.2. Quá trình hoạt động của một chương trình ứng dụng thông qua các tầng của CSDL.

Hình 1.6.2 cho chúng ta một cách nhìn về quá trình hoạt động của một chương trình ứng dụng thông qua các tầng của CSDL:

Các yêu cầu của chương trình ứng dụng được chuyển tới hệ quản trị CSDL (theo con đường số 2). Tại đây hệ quản trị CSDL sẽ tham khảo các từ điển dữ liệu (*Meta DataBase*) để tìm kiếm các ánh xạ cấu trúc ngoài với cấu trúc quan niệm và cấu trúc vật lý (các ngõ a, b và c). Tại đây hệ quản trị CSDL có thể sẽ tham khảo tới vùng đệm của

nó để xác định xem câu trả lời đã có sẵn ở đó chưa, nếu có thì trả lại cho chương trình ứng dụng thông qua con đường số 9; ngược lại sẽ yêu cầu hệ điều hành truy xuất thông tin theo con đường số 3. Tới đây hệ điều hành sẽ gửi yêu cầu truy xuất thông tin trong CSDL thông qua hệ thống xuất nhập của HĐH (các con đường số 4 và 5). Nếu việc truy xuất không thành công nó sẽ trả lại yêu cầu về cho hệ quản trị CSDL (có thể thông qua các mã lỗi) qua con đường số 6; nếu thành công thì dữ liệu sẽ được chuyển vào vùng đệm của hệ quản trị CSDL. Qua xử lý, hệ quản trị CSDL sẽ chuyển dữ liệu vào vùng đệm của chương trình ứng dụng để nó xử lý (qua con đường 8a) và cho ra kết quả trả lời của chương trình ứng dụng qua con đường số 10.

Theo sơ đồ trên có thể nhận thấy các trục trặc có thể xảy ra tại các con đường (2a), (3), (4), (5), (6) và (8). Lỗi tại 2 con đường số (6) và (8) có thể là do tràn vùng làm việc.

Tính độc lập giữa dữ liệu và chương trình.

Lược đồ khái niệm là sự biểu diễn thế giới thực bằng một loại ngôn ngữ phù hợp của hệ quản trị CSDL. Qua hình 1.5.1 - Sơ đồ tổng quát của một CSDL theo kiến trúc ANSI - PARC, chúng ta có thể thấy, từ chương trình ứng dụng và người khai thác trực tiếp CSDL thông qua một khung nhìn tới CSDL (*View*) tồn tại hai mức độc lập dữ liệu. Thứ nhất, lược đồ vật lý có thể thay đổi do người quản trị CSDL mà hoàn toàn không làm thay đổi các lược đồ con. Người quản trị CSDL có thể tổ chức lại CSDL bằng cách thay đổi cách tổ chức, cấu trúc vật lý của dữ liệu trên các thiết bị nhớ thứ cấp để làm thay đổi hiệu quả tính toán của các chương trình ứng dụng, nhưng không đòi hỏi phải viết lại các chương trình ứng dụng. Điều này được gọi là tính độc lập vật lý của dữ liệu - hay tính độc lập của dữ liệu ở mức vật lý (*Physical Independence*). Tính độc lập dữ liệu mức vật lý được đảm bảo tới mức nào còn phụ thuộc vào chất lượng của hệ quản trị CSDL.

Thứ hai, giữa khung nhìn với lược đồ quan niệm cũng có thể tồn tại một loại độc lập về dữ liệu. Trong quá trình khai thác CSDL người ta có thể nhận thấy tính cần thiết phải sửa đổi lược đồ khái niệm như bổ sung thêm thông tin hoặc xóa bớt các thông tin của các thực thể đang tồn tại trong CSDL. Việc thay đổi lược đồ khái niệm không làm ảnh hưởng tới các lược đồ con, do đó không cần phải viết lại các chương trình ứng dụng. Tính chất độc lập này được gọi là tính độc lập của dữ liệu ở mức logic (*Logical Independence*).

Tính độc lập giữa dữ liệu với chương trình ứng dụng là mục tiêu chủ yếu của các hệ quản trị CSDL. C.J. Date [3] đã định nghĩa tính độc lập dữ liệu là "*tính bất biến của các*

hệ ứng dụng đối với các thay đổi bên trong cấu trúc lưu trữ và chiến lược truy nhập CSDL".

Những cách tiếp cận một CSDL

Mô hình dữ liệu là sự trừu tượng hóa môi trường thực, nó là sự biểu diễn dữ liệu ở mức quan niệm. Mỗi loại mô hình dữ liệu đặc trưng cho một cách tiếp cận dữ liệu khác nhau của những nhà phân tích - thiết kế CSDL, mỗi loại đều có các ưu điểm và mặt hạn chế của nó nhưng vẫn có những mô hình dữ liệu nổi trội và được nhiều người quan tâm nghiên cứu. Cho đến nay đang tồn tại 5 loại mô hình dữ liệu, đó là: mô hình dữ liệu mạng, mô hình dữ liệu phân cấp, mô hình dữ liệu quan hệ, mô hình dữ liệu thực thể - kết hợp và mô hình dữ liệu hướng đối tượng. Chương này sẽ lần lượt giới thiệu các loại mô hình dữ liệu nêu trên.

Cách tiếp cận theo mô hình dữ liệu mạng

Mô hình dữ liệu mạng (*Network Data Model*) - còn được gọi tắt là mô hình mạng hoặc mô hình lưới (*Network Model*) là mô hình được biểu diễn bởi một đồ thị có hướng. Trong mô hình này người ta đưa vào các khái niệm: mẫu tin hay bản ghi (*Record*), loại mẫu tin (*Record Type*) và loại liên hệ (*Set Type*):

(a) *Loại mẫu tin (Record Type)* là mẫu đặc trưng cho 1 loại đối tượng riêng biệt. Chẳng hạn như trong việc quản lý nhân sự tại một đơn vị, đối tượng cần phản ánh của thế giới thực có thể là Phòng, Nhân viên, Công việc, lý lịch ... do đó có các loại mẫu tin đặc trưng cho từng đối tượng này. Trong đồ thị biểu diễn mô hình mạng mỗi loại mẫu tin được biểu diễn bởi một hình chữ nhật, một thể hiện (Instance) của một loại mẫu tin được gọi là bản ghi. Trong ví dụ trên loại mẫu tin Phòng có các mẫu tin là các phòng, ban trong đơn vị; loại mẫu tin nhân viên có các mẫu tin là các nhân viên đang làm việc tại các phòng ban của cơ quan...

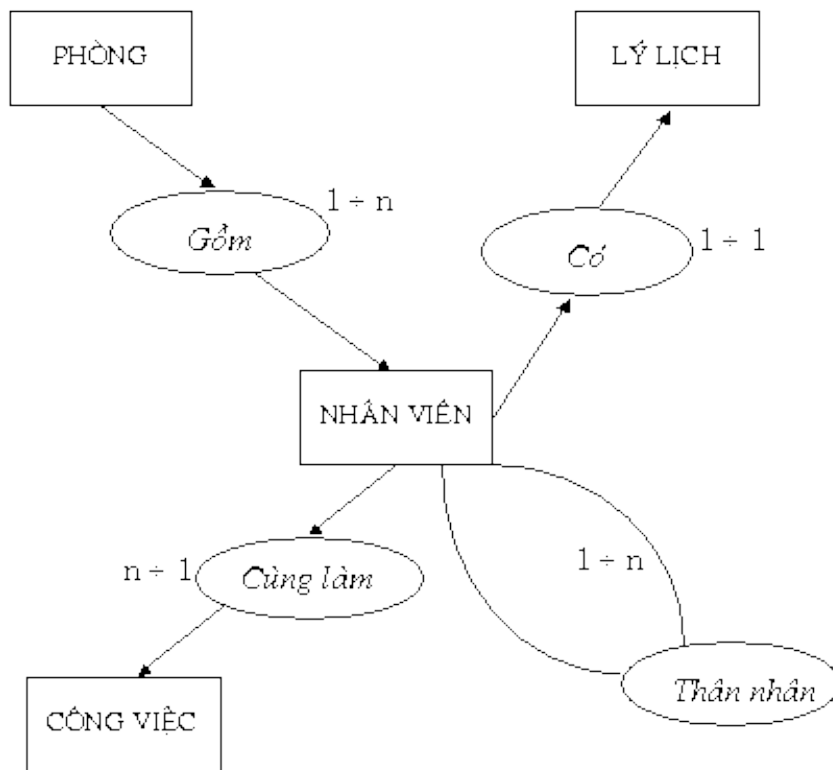
(b) *Loại liên hệ (Set Type)* là sự liên kết giữa một loại mẫu tin chủ với một loại mẫu tin thành viên. Trong đồ thị biểu diễn mô hình mạng mỗi loại liên hệ được biểu diễn bởi một hình bầu dục (*oval*) và sự liên kết giữa 2 loại mẫu tin được thể hiện bởi các cung có hướng (các mũi tên) đi từ loại mẫu tin chủ tới loại liên hệ và từ loại liên hệ tới loại mẫu tin thành viên.

Trong loại liên kết người ta còn chỉ ra số lượng các mẫu tin tham gia trong mỗi kết hợp. Có các loại liên hệ sau:

- 1:-1 (*One-to-One*): Mỗi mẫu tin của loại mẫu tin chủ kết hợp với đúng 1 mẫu tin của loại mẫu tin thành viên. Ví dụ, mỗi nhân viên có duy nhất một lý lịch cá nhân.
- 1:-n (*One-to-Many*): Mỗi mẫu tin của loại mẫu tin chủ kết hợp với 1 hay nhiều mẫu tin của loại mẫu tin thành viên. Ví dụ, mỗi phòng ban có từ 1 đến nhiều nhân viên. Mỗi 1 nhân viên chỉ thuộc một phòng ban nhất định.
- n:-1 (*Many-to-One*): Nhiều mẫu tin của loại mẫu tin chủ kết hợp với đúng 1 mẫu tin của loại mẫu tin thành viên. Ví dụ, nhiều nhân viên cùng làm một công việc.
- Đệ quy (*Recursive*): Một loại mẫu tin chủ cũng có thể đồng thời là loại mẫu tin thành viên với chính nó. Ta nói rằng loại liên hệ này là đệ quy.

Hình 2.1 biểu diễn một ví dụ về mô hình dữ liệu mạng đối với CSDL nhân sự của một đơn vị. Trong đồ thị này, chúng ta có 4 loại mẫu tin: phòng, nhân-viên, công-việc và lý-lịch; 4 loại liên hệ: phòng gồm 1 đến nhiều nhân-viên; nhân-viên có đúng 1 lý-lịch; nhiều nhân-viên cùng làm một công-việc; 1 nhân-viên có thể có 1 hay nhiều nhân-viên là thân nhân của mình.

Mô hình dữ liệu mạng tương đối đơn giản, dễ sử dụng nhưng nó không thích hợp trong việc biểu diễn các CSDL có quy mô lớn bởi trong một đồ thị có hướng khả năng diễn đạt ngữ nghĩa của dữ liệu, nhất là các dữ liệu và các mối liên hệ phức tạp của dữ liệu trong thực tế là rất hạn chế.



Hình 2.1 Mô hình dữ liệu mạng (Network Model)

Mô hình dữ liệu phân cấp.

Mô hình dữ liệu phân cấp (*Hierarchical Data Model*) - được gọi tắt là mô hình phân cấp (*Hierarchical Model*): Mô hình là một cây (*Tree*), trong đó mỗi nút của cây biểu diễn một thực thể, giữa nút con và nút cha được liên hệ với nhau theo một mối quan hệ xác định.

Mô hình dữ liệu phân cấp sử dụng các khái niệm sau:

- (a) *Loại mẫu tin*: giống khái niệm mẫu tin trong mô hình dữ liệu mạng.
- (b) *Loại mối liên hệ*: Kiểu liên hệ là phân cấp, theo cách:
 - Mẫu tin thành viên chỉ đóng vai trò thành viên của một mối liên hệ duy nhất, tức là nó thuộc một chủ duy nhất. Như vậy, mỗi liên hệ từ mẫu tin chủ tới các mẫu tin thành viên là 1,n, và từ mẫu tin (hay bản ghi - record) thành viên với mẫu tin chủ là 1,1.
 - Giữa 2 loại mẫu tin chỉ tồn tại 1 mối liên hệ duy nhất.

Ví dụ 2.2.1:

Trong cuộc Tổng điều tra số dân năm 1989, chương trình nhập phiếu điều tra được viết bằng ngôn ngữ CENS4 cho kết quả là các file dữ liệu nhập dạng văn bản được tổ chức như sau:

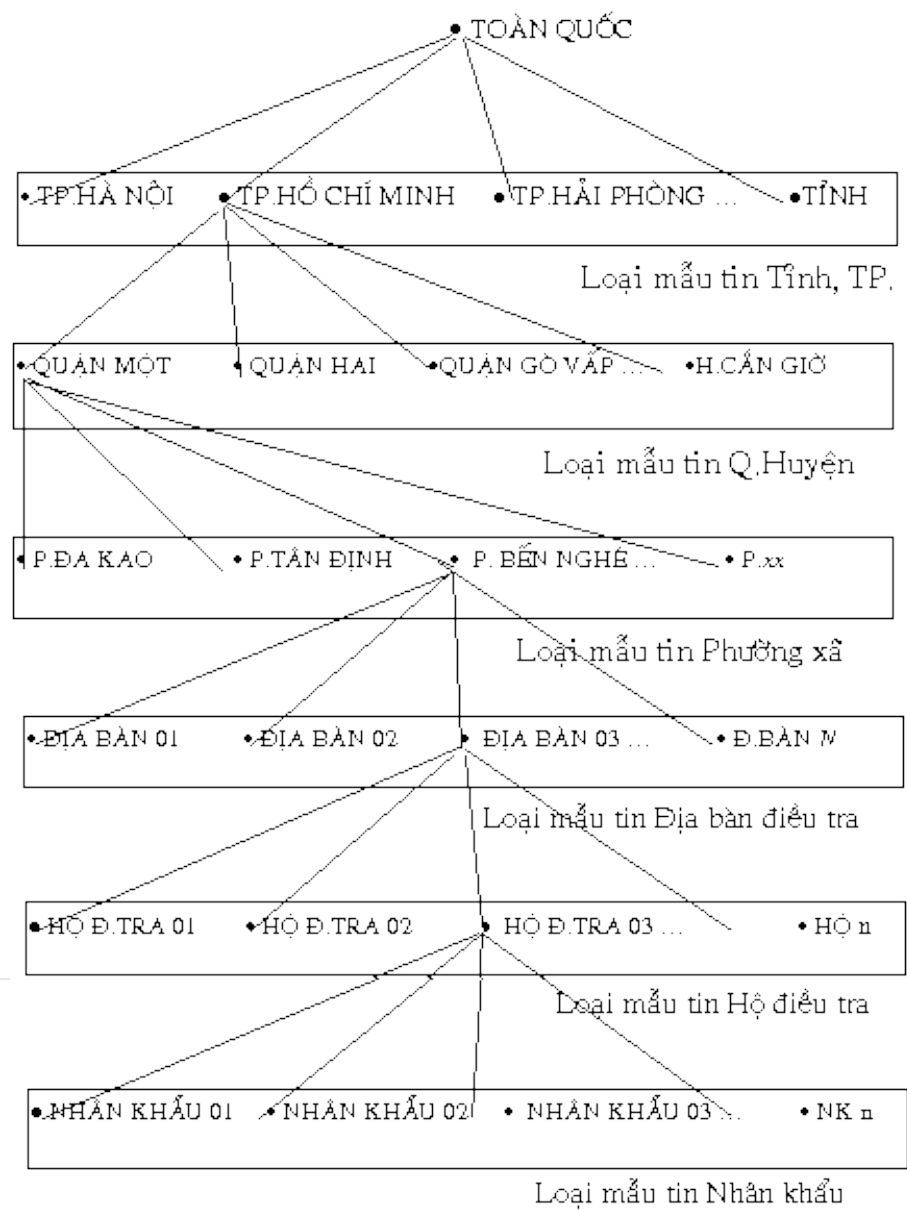
Các dòng là các mẫu tin (hay bản ghi) có độ dài thay đổi.

Có 6 loại mẫu tin:

- Mẫu tin đặc trưng cho tỉnh, thành phố gồm Mã số tỉnh thành, Tên tỉnh thành phố. '02' là Mã số Thành phố Hồ Chí Minh.
- Mẫu tin đặc trưng cho quận huyện gồm Mã số tỉnh thành+Mã số quận huyện, Tên quận huyện trong tỉnh thành phố đó. '0201' là Mã số quận Nhất của TP.Hồ Chí Minh.
- Mẫu tin đặc trưng cho phường xã gồm Mã số tỉnh thành+Mã số quận huyện+Mã số phường xã, Tên phường xã thuộc quận huyện trong tỉnh thành phố đó. '020101' là Mã số phường Bến Nghé, Quận Nhất, TP.Hồ Chí Minh.
- Mẫu tin đặc trưng cho địa bàn điều tra trong một phường xã. '02010101' là mã số địa bàn điều tra số 01 trong phường Bến Nghé.
- Mẫu tin đặc trưng cho hộ điều tra, gồm Mã số tỉnh + Mã số quận + Mã số phường + Mã số địa bàn + Số thứ tự hộ điều tra trong địa bàn, Tổng số nhân khẩu trong hộ, Trong đó: Nữ, Tổng số trẻ dưới 16 tuổi.
- Mẫu tin đặc trưng cho nhân khẩu của hộ, gồm các thông tin xác định hộ điều tra, Số thứ tự nhân khẩu trong hộ, Quan hệ với chủ hộ (1,9), Giới tính (1,2,3), Tháng sinh, Năm sinh, Trình độ văn hóa, ...

Ở đây rõ ràng là một sự phân cấp trong file CSDL. Một tỉnh thành phố (thì) có nhiều quận huyện, một quận huyện chỉ thuộc một tỉnh thành duy nhất. Một quận huyện (thì) có nhiều phường xã và một phường xã chỉ thuộc một quận huyện duy nhất. Mỗi phường xã được chia thành nhiều địa bàn điều tra, mỗi địa bàn chỉ thuộc một phường xã duy nhất v.v...

Hình 2.2 mô tả cây phân cấp của mô hình dữ liệu đối với CSDL Tổng điều tra số dân Toàn quốc 0 giờ ngày 01 tháng 01 năm 1989.



Hình 2.2 Mô hình dữ liệu phân cấp (Hierarchical Model)

Những cách tiếp cận một cơ sở dữ liệu

Mô hình dữ liệu quan hệ.

Mô hình dữ liệu quan hệ (*Relational Data Model*) - còn được gọi tắt là mô hình quan hệ (*Relational Model*) do E.F.Codd [2] đề xuất năm 1970. Nền tảng cơ bản của nó là khái niệm lý thuyết tập hợp trên các quan hệ, tức là tập của các bộ giá trị (*Value Tuples*). Trong mô hình dữ liệu này những khái niệm sẽ được sử dụng bao gồm thuộc tính (*Attribute*), quan hệ (*Relation*), lược đồ quan hệ (*Relation Schema*), bộ (*Tuple*), khóa (*Key*).

Mô hình dữ liệu quan hệ là mô hình được nghiên cứu nhiều nhất, và cho thấy rằng nó có cơ sở lý thuyết vững chắc nhất. Mô hình dữ liệu này cùng với mô hình dữ liệu thực thể kết hợp đang được sử dụng rộng rãi trong việc phân tích và thiết kế CSDL hiện nay. Chúng ta sẽ nghiên cứu chi tiết mô hình dữ liệu này ở các chương sau.

Mô hình dữ liệu thực thể - kết hợp.

Mô hình dữ liệu thực thể - kết hợp (*Entity - Relationship Model*) do P.P.Chen đề xuất vào năm 1976. Các khái niệm chủ yếu được sử dụng trong lý thuyết của mô hình này là:

Loại thực thể (*Entity Type*): Là một loại đối tượng cần quản lý trong CSDL, chẳng hạn, KHOA, LỚP-HỌC, MÔN-HỌC, GIẢNG-VIÊN, HỌC-VIÊN, tức là, cũng tương tự như khái niệm về loại mẫu tin trong mô hình mạng và mô hình phân cấp.

Thực thể (*Entity*): Là một thể hiện hoặc một đối tượng của một loại thực thể. Khái niệm này tương tự như khái niệm mẫu tin trong mô hình dữ liệu mạng và mô hình dữ liệu phân cấp.

Thuộc tính của loại thực thể (*Entity Attribute*): Là các đặc tính riêng biệt cơ bản của loại thực thể, tương tự khái niệm thuộc tính (*Attribute*) trong mô hình dữ liệu quan hệ sẽ trình bày trong Chương III. Ví dụ, loại thực thể KHOA có các thuộc tính Mã-Khoa, Tên-Khoa. Loại thực thể LỚP-HỌC có một số thuộc tính Mã-Lớp, Tên-Lớp, Niên-Khóa, Số-Học-Viên. Loại thực thể MÔN-HỌC có một số thuộc tính Mã-Môn, Tên-Môn, Số-Đv-Học-Trình. Loại thực thể HỌC-VIÊN có một số thuộc tính Mã-Học-Viên, Tên-Học-Viên, Ngày-Sinh, Quê-Quán. Loại thực thể GIẢNG-VIÊN có một số thuộc tính Mã-Giảng-Viên, Tên-Giảng-Viên, Cấp-Học-Vị, Chuyên-Ngành. v.v...

Khóa của loại thực thể (*Entity Key*): Đó là các thuộc tính nhận diện của loại thực thể. Căn cứ vào các giá trị của các thuộc tính nhận diện này người ta có thể xác định một

thực thể duy nhất của một loại thực thể. Ví dụ, khóa của loại thực thể LỚP-HỌC có thể là Mã-Lớp; khóa của loại thực thể HỌC-VIÊN có thể là Mã-Học-Viên; khóa của loại thực thể MÔN-HỌC có thể là Mã-Môn-Học ... Khái niệm này cũng tương tự như khái niệm khóa (*Key*) trong mô hình dữ liệu quan hệ sẽ trình bày trong Chương III.

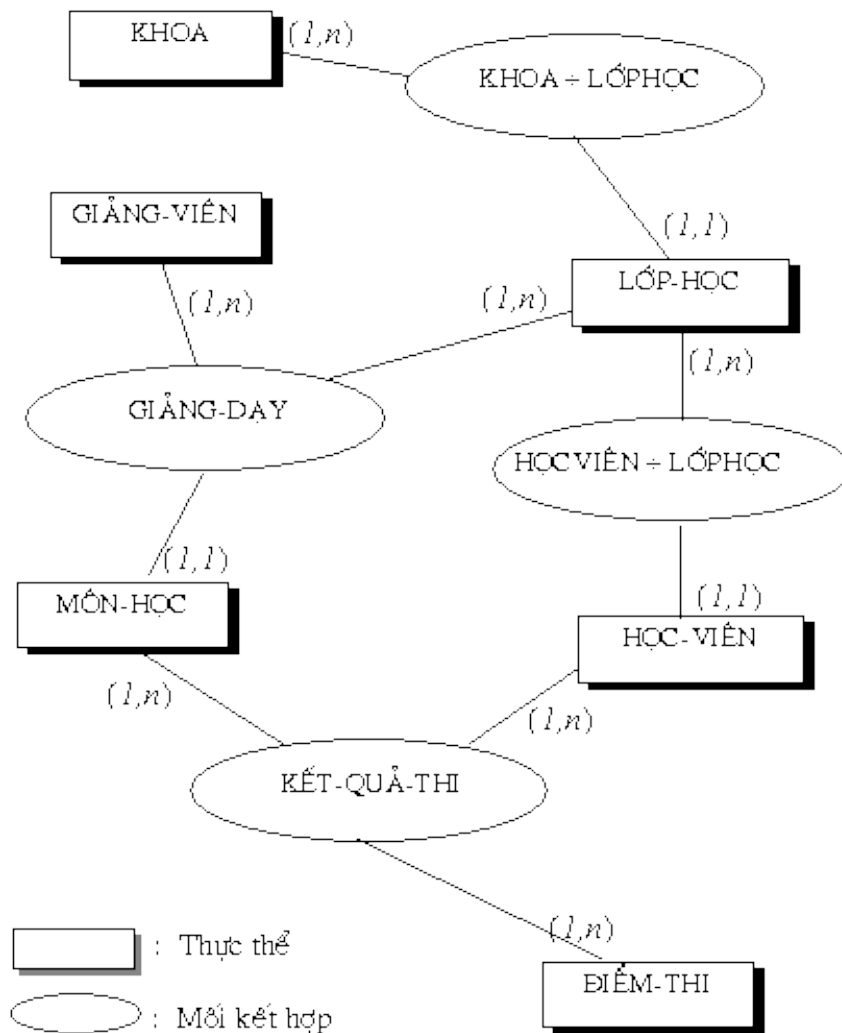
Loại mối kết hợp (*Entity Relationship*): Tương tự như loại mối liên hệ trong mô hình dữ liệu mạng. Trong đồ thị biểu diễn của mô hình này người ta cũng sử dụng hình elíp để thể hiện một mối kết hợp giữa các thực thể. Giữa 2 loại thực thể có thể tồn tại nhiều hơn một mối kết hợp.

Số ngôi của mối kết hợp (*RelationShip Degree*): Là tổng số loại thực thể tham gia vào mối kết hợp. Ví dụ, giữa loại thực thể SINH-VIÊN và KHOA tồn tại mối kết hợp "trực thuộc" - đó là mối kết hợp 2 ngôi. KẾT-QUẢ-THI (hoặc KIỂM-TRA) của sinh viên là mối kết hợp giữa 3 thực thể SINH-VIÊN, MÔN-THI và ĐIỂM-THI - đó là mối kết hợp 3 ngôi.

Thuộc tính của mối kết hợp (*RelationShip Attribute*): Mối kết hợp có thể có các thuộc tính của riêng nó. Thông thường mối kết hợp có các thuộc tính là khóa của các loại thực thể tham gia vào mối kết hợp, ngoài ra còn có thêm những thuộc tính bổ sung khác. Ví dụ, trong mối kết hợp 3 ngôi kể trên, thuộc tính của mối kết hợp này có thể bao gồm Mã-Học-Viên, Mã-Môn-Học, Điểm-Thi; và có thể có thêm các thuộc tính bổ sung khác như Lần-Thi-Thứ, Ngày-Thi, Ghi-Chú v.v...

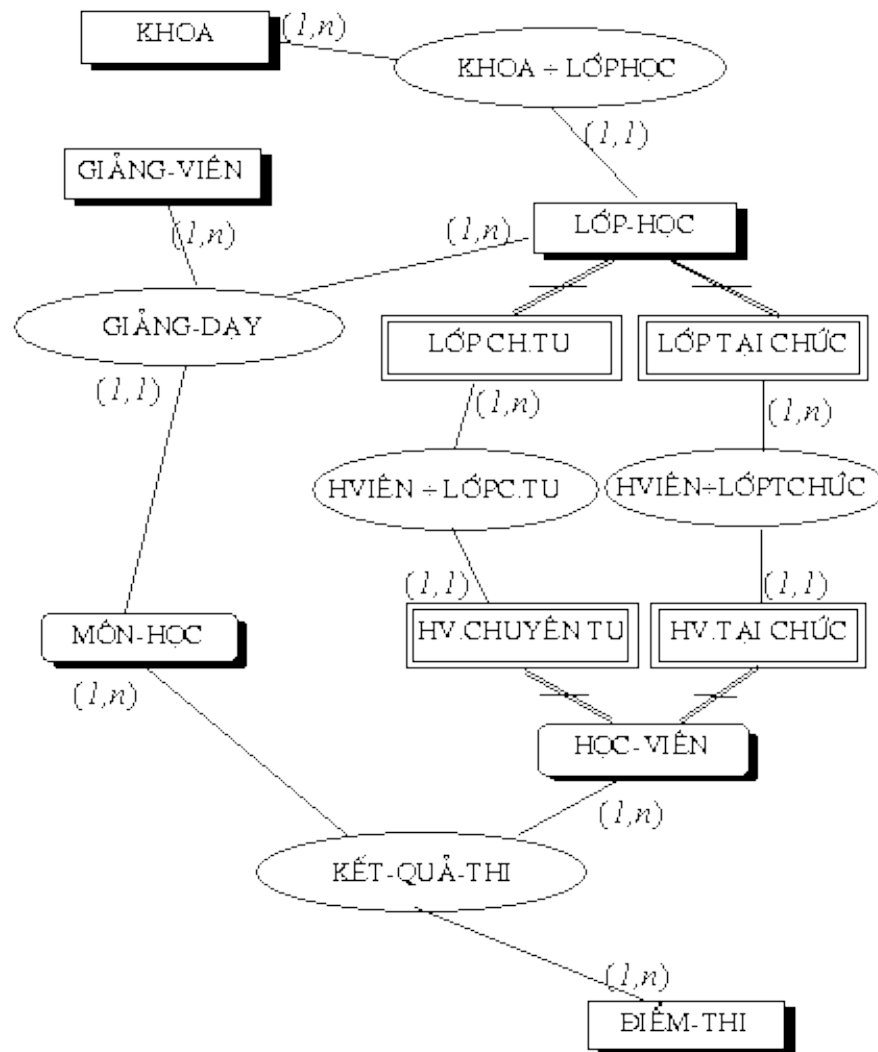
Bản số của mỗi nhánh của mối kết hợp (*RelationShip Cardinal*): Mỗi nhánh (hay mỗi chiều) của mối kết hợp là mối kết hợp nối một loại thực thể với một mối kết hợp. Trong nhánh này cần xác định số lượng tối thiểu và số lượng tối đa các thực thể của nhánh đó sẽ tham gia vào một thực thể của mối kết hợp. Hai đại lượng này - đặt trong cặp dấu ngoặc tròn - được gọi là bản số của mối kết hợp. Ví dụ, trong mối kết hợp 3 ngôi nêu trên, tại nhánh nối loại thực thể HỌC-VIÊN với mối kết hợp KẾT-QUẢ-THI là (1,n), bởi vì sẽ có ít nhất một học viên tham gia kỳ thi và nhiều nhất là tất cả số học viên học môn đó cùng dự thi.

Vào khoảng năm 1980, mô hình dữ liệu *thực thể-kết hợp* đã được mở rộng thêm một số khái niệm mới như "loại thực thể chuyên biệt hóa" (*Specialized Entity*) và "mối kết hợp đệ quy" (*Recursive RelationShip*). Mô hình này cùng với mô hình dữ liệu quan hệ và mô hình hướng đối tượng được sử dụng khá phổ biến trong việc thiết kế các CSDL hiện nay, bởi sự kết hợp này làm cho mô hình dữ liệu thể hiện được nhiều ngữ nghĩa của những loại dữ liệu trong CSDL hơn.



Hình 2.4.1 Mô hình dữ liệu thực thể - kết hợp của CSDL quản lý học viên.

Hình 2.4.1 trình bày một mô hình *thực thể - kết hợp* cho CSDL quản lý học viên gồm các loại thực thể KHOA, LỚP-HỌC, MÔN-HỌC, GIẢNG-VIÊN, HỌC-VIÊN. Mỗi kết hợp HỌCVIÊN, LỚP-HỌC giữa 2 loại thực thể HỌC-VIÊN và LỚP-HỌC. Tại nhánh HỌC-VIÊN bản số của nó là $(1,1)$, điều này nhà phân tích và thiết kế đã khẳng định tình trạng thực tế là một học viên phải theo học (hay có tên trong danh sách) ít nhất là một lớp và cũng chỉ thuộc tối đa một lớp. Bản số của nhánh LỚP-HỌC là $(1,n)$ nghĩa là một lớp (nếu đã có tên trong danh sách lớp học) thì có ít nhất 1 học viên và tối đa có thể là nhiều (n) học viên theo học.



Hình 2.4.2. Mô hình thực thể kết hợp với các thực thể chuyên biệt hóa.

Nếu yêu cầu quản lý đòi hỏi phải phân tích rạch ròi thành 2 loại lớp học chính quy và tại chức, và các lớp học tại chức có thể sẽ không phải học một số môn học hoặc số đơn vị học trình ít hơn so với các lớp chính quy, khi đó mô hình dữ liệu *thực thể - kết hợp* được thể hiện như trong sơ đồ 2.4.2. Thực thể chuyên biệt hóa được thể hiện trong mô hình bằng hình chữ nhật có khung (Frame) đôi; một đường kết hợp bằng nét đôi có dấu gạch ngang thể hiện mối quan hệ là chuyên biệt hoá thành các loại thực thể con chi tiết hơn. Các tên trong các hình hộp chữ nhật thể hiện các thuộc tính của loại thực thể đó.

Mô hình dữ liệu hướng đối tượng.

Mô hình dữ liệu hướng đối tượng (*Object Oriented Data Model*) ra đời từ cuối những năm 80 và đầu những năm 90. Đây là loại mô hình tiên tiến nhất hiện nay dựa trên cách

tiếp cận hướng đối tượng đã quen thuộc trong các phương pháp lập trình hướng đối tượng, nó sử dụng các khái niệm như lớp (*class*), sự kế thừa (*inheritance*), kế thừa bội (tức là kế thừa từ nhiều lớp cơ sở *multi-inheritance*). Đặc trưng cơ bản của cách tiếp cận này là tính đóng gói (*encapsulation*), tính đa hình (*polymorphism*) và tính tái sử dụng (*Reusability*).

Lớp là một kiểu dữ liệu có cấu trúc bao gồm các thành phần dữ liệu và các phương thức xử lý thao tác trên cấu trúc dữ liệu đó. Nó là một kiểu (hay cấu trúc) dữ liệu được trừu tượng hóa, bởi vì các tác động (còn gọi là các phương thức - *method*) là để phục vụ hoặc thao tác trên kiểu dữ liệu này. Dữ liệu và phương thức hòa quyện vào nhau thành một thể thống nhất: dữ liệu cần có những cách thức xử lý thỏa đáng, và phương thức xử lý được đưa vào trong kiểu dữ liệu đó là để phục vụ cho các đối tượng có cấu trúc như thế. Người ta gọi sự thống nhất đó là sự đóng gói.

Ví dụ, trong việc định nghĩa phép cộng (+) hai số phức c1 và c2 để cho một số phức kết quả là:

COMPLEX (c1.Real+c2.Real, c1.Image+c2.Image)

để người ta có thể sử dụng phép cộng (+) một cách tự nhiên như việc gán kết quả đó cho biến phức $c = c1 + c2$, hoàn toàn tự nhiên và trong sáng hơn rất nhiều so với việc phải viết một thủ tục `Add_COMPLEX (COMPLEX &c1, COMPLEX &c2, COMPLEX &c)` để cộng hai số phức và kết quả được gán vào tham đối thứ 3 của hàm thông qua tham chiếu địa chỉ của biến c theo cách lập trình hướng thủ tục trước đó. Và, hiển nhiên rằng, cách thức cộng như trên là chỉ được áp dụng cho các đối tượng số phức.

Tương tự, cũng có thể định nghĩa phép toán cộng (+) trong lớp ma trận (MATRIX) để cộng hai ma trận có cùng bậc. Khi đó giả sử A và B là hai ma trận $m * n$; Phép gán tổng hai ma trận A và B cho biến ma trận C có thể được viết:

$$C = A + B$$

rõ ràng là tự nhiên và dễ hiểu hơn đối với các ứng dụng lớp ma trận so với việc gọi thủ tục `Add_MATRIX (MATRIX &A, MATRIX &B, MATRIX &C, int m, int n) ...`

Phương pháp tiếp cận hướng đối tượng trong mô hình dữ liệu mặc dù còn mới mẻ nhưng hiện nay đang được nhiều người quan tâm nghiên cứu phát triển và áp dụng. Các hệ quản trị CSDL hướng đối tượng hiện nay vẫn chưa nhiều, một số còn chưa thuần nhất (nghĩa là việc lập trình là hướng đối tượng nhưng CSDL vẫn chủ yếu dựa trên mô hình quan hệ).

Mô hình dữ liệu quan hệ của e.f.codd

Các khái niệm cơ bản

Phần mở đầu của chương III sẽ trình bày kỹ hơn về các khái niệm đã được nhắc tới trong chương II về cách tiếp cận mô hình dữ liệu kiểu quan hệ và sẽ coi đó như là những cơ sở nền tảng để tiếp tục nghiên cứu các phần tiếp theo.

Thuộc tính (Attribute):

Thuộc tính là một tính chất riêng biệt của một đối tượng cần được lưu trữ trong CSDL để phục vụ cho việc khai thác dữ liệu về đối tượng.

+*Ví dụ 3.1.1:*

-Đối tượng KHOA (tương ứng với loại thực thể KHOA trong mô hình thực thể kết hợp) có các thuộc tính Mã-khoa, Tên-khoa.

-Loại thực thể LỚP-HỌC có một số thuộc tính Mã-lớp, Tên-lớp, Niên-khóa, Số-học-viên.

-Loại thực thể MÔN-HỌC có một số thuộc tính Mã-môn, Tên-môn, Số-Đv-Học-Trình.

-Loại thực thể HỌC-VIÊN có một số thuộc tính Mã-khoa, Mã-học-viên, Tên-học-viên, Ngày-sinh, Quê-quán.

-Loại thực thể GIẢNG-VIÊN có một số thuộc tính Mã-giảng-viên, Tên-giảng-viên, Cấp-học-vị, Chuyên-ngành. v.v...

Các thuộc tính được đặc trưng bởi một tên gọi, kiểu giá trị và miền giá trị của chúng. Trong giáo trình này, để trình bày một cách tổng quát và nếu không cần lưu ý đến ngữ nghĩa thì tên của các thuộc tính thường được ký hiệu bằng các chữ cái in hoa đầu tiên trong bảng chữ cái la tinh: A, B, C, D, ... Những chữ cái X, Y, W, Z, ... dùng thay cho một nhóm (hay tập hợp) gồm nhiều thuộc tính. Đôi khi còn dùng các ký hiệu chữ cái với các chỉ số A1, A2, ..., An để chỉ các thuộc tính trong trường hợp tổng quát hay muốn đề cập đến số ngôi (hay số lượng các thuộc tính) của một quan hệ.

Trong các ứng dụng thực tế, người phân tích - thiết kế thường đặt tên thuộc tính một cách gọi nhớ; nhưng để làm rõ hơn ý nghĩa của những tên gọi, người ta có thể đặt tên khá dài cho các thuộc tính với các chữ in hoa đầu từ hoặc viết cách nhau bởi dấu gạch chân (*Underscore: _*).

Trong các ví dụ của tài liệu này, các tên thuộc tính được viết bằng tiếng Việt gồm nhiều từ Việt nối với nhau bởi dấu trừ (-) có chữ cái đầu tiên được viết in hoa nhằm mục đích chuyển tải cả ngữ nghĩa của tên thuộc tính. Điều này không có gì sai, bởi vì hiện nay có một số hệ quản trị CSDL cho phép làm như vậy (*MicroSoft Access*, *SQL-Server* cho phép đặt tên dài tới 255 ký tự và có thể có chứa các khoảng trắng, các ký tự tiếng Việt có dấu và các ký tự đặc biệt khác). Những tên thuộc tính hoặc tên quan hệ như vậy, khi sử dụng trong *Micro Soft Access* hoặc *SQL-Server* phải viết chúng trong cặp dấu ngoặc vuông ([]), khi sử dụng trong *ORACLE* phải viết trong cặp dấu nháy kép (" " – *Quotes*). Trong tài liệu này chúng ta sử dụng ký pháp của *SQL-Server*.

Trong cài đặt cụ thể với một hệ quản trị CSDL cần lưu ý đến khía cạnh đặt tên cho các bảng cũng như tên của thuộc tính. Trong hầu hết các ngôn ngữ lập trình nói chung và một số ngôn ngữ quản trị CSDL nói riêng, tên đối tượng (tên biến, tên quan hệ hay tên thuộc tính v.v...) đều chỉ được phép viết bằng các chữ cái la tinh, chữ số và/hoặc dấu gạch chân (*underscore* ' _ '), bắt đầu bằng chữ cái hoặc dấu gạch chân, với độ dài tên theo quy định. Theo lý thuyết, người ta vẫn khuyên rằng không nên đặt tên thuộc tính quá dài (bởi vì nó làm cho việc viết các câu lệnh truy vấn trở nên vất vả hơn) và cũng không nên đặt tên thuộc tính quá ngắn (vì nó không cho thấy ngữ nghĩa của thuộc tính của quan hệ), đặc biệt là không đặt trùng tên hai thuộc tính mang ngữ nghĩa khác nhau thuộc hai đối tượng khác nhau. Chẳng hạn, nếu có hai đối tượng HỌC-VIÊN và GIẢNG-VIÊN đều có thuộc tính TÊN thì nên đặt tên thuộc tính rõ ràng là Tên_học-viên của loại đối tượng HỌC-VIÊN và Tên-giáo-viên cho đối tượng GIẢNG-VIÊN, bởi vì 2 thuộc tính TÊN đó mang ngữ nghĩa khác nhau trong 2 quan hệ khác nhau.

Mỗi thuộc tính đều phải thuộc một kiểu dữ liệu (*Data Type*) nhất định. Kiểu dữ liệu có thể là vô hướng (đó là các kiểu dữ liệu cơ bản như chuỗi - *String* hoặc *Text* hoặc *Charater*, số - *Number*, Luận lý - *Logical*, ...) hoặc các kiểu dữ liệu có cấu trúc được định nghĩa dựa trên các kiểu dữ liệu đã có sẵn. Một số kiểu dữ liệu vô hướng sau đây thường được sử dụng trong các hệ quản trị CSDL :

- *Text* (hoặc *Character*, *String*, hoặc *Char*) – kiểu văn bản.
- *Number* (hoặc *Numeric*, hoặc *float*) – kiểu số
- *Logical* (hoặc *Boolean*) – kiểu luận lý
- *Date/Time* – kiểu thời gian : ngày tháng năm + giờ phút
- *Memo* (hoặc *VarChar*) – kiểu văn bản có độ dài thay đổi.

Mỗi hệ quản trị CSDL có thể gọi tên các kiểu dữ liệu nói trên bằng các tên gọi khác nhau, ngoài ra còn bổ sung thêm một số kiểu dữ liệu riêng của mình. Chẳng hạn, *MicroSoft Access* có kiểu dữ liệu *OLE* để chứa các đối tượng nhúng như hình ảnh, âm

thanh, audio, video ... ORACLE có kiểu dữ liệu LONG cho phép chứa dữ liệu có kích thước lớn tới 2 tỷ bytes.

Mỗi thuộc tính có thể chỉ chọn lấy những giá trị trong một tập hợp con của kiểu dữ liệu. Tập hợp các giá trị mà một thuộc tính A có thể nhận được gọi là miền giá trị (*domain*) của thuộc tính A và được ký hiệu là MGT(A) hoặc *Dom*(A).

-Ví dụ 3.1.2:

Học viên đang theo học tại trường ĐH.KHTN thì tuổi của họ nhiều nhất là 60 và tuổi ít nhất là 18, vừa mới tốt nghiệp PTTH. Mặc dù nói rằng Năm-sinh của học viên là một số nguyên, nhưng không phải số nguyên nào cũng có thể được chọn để gán vào thuộc tính Năm-sinh. Giá trị năm sinh của học viên chỉ cần lưu 2 chữ số sau của năm sinh tức là chỉ cần một byte để ghi nhận những năm sinh của họ trong thế kỷ 20: từ năm 40 đến năm 82. Với miền giá trị chỉ chứa từ 40 đến 82, như vậy chỉ cần dùng 1 byte để lưu là đủ.

Nếu kiểu dữ liệu của thuộc tính A là có cấu trúc thì miền giá trị của A là tích Đề-các (hoặc tập con của tích Đề-các - *Cartesian*) của các miền giá trị thành phần.

-Ví dụ 3.1.3:

Ta có kiểu dữ liệu ngày tháng năm theo dương lịch với mô tả bằng ngôn ngữ Pascal như sau:

Type DATE = Record

Day : 1 .. 31;

Month : 1 .. 12;

Year : 0 .. 2500;

End;

Tích Đề-các của 3 miền giá trị của các thành phần là $[1 .. 31] \times [1 .. 12] \times [0 .. 2500]$. Nếu thuộc tính A thuộc kiểu DATE thì MGT(A) $\subseteq [1..31] \times [1..12] \times [0..2500]$, bởi vì {30,02,1999} không phải là một ngày tháng năm hợp lệ nên tổ hợp đó không thuộc MGT(A). Ngày nay, hầu hết các ngôn ngữ quản trị dữ liệu đều có định nghĩa kiểu này như một kiểu cơ bản. Tổ hợp 3 giá trị thành phần luôn luôn được kiểm tra tính đúng đắn trước khi được coi là một giá trị kiểu ngày tháng. Hai phép toán số học có thể tác động trên kiểu DATE là phép cộng (+) hoặc trừ (-) một DATE với một số nguyên để cho kết quả là một giá trị kiểu DATE; hiệu 2 giá trị kiểu DATE là số ngày trôi qua giữa 2 ngày tháng năm đó.

Trong nhiều hệ quản trị CSDL, người ta thường đưa thêm vào miền giá trị của các thuộc tính một giá trị đặc biệt gọi là giá trị rỗng (NULL). Tùy theo ngữ cảnh mà giá trị này có thể đặc trưng cho một giá trị không thể xác định được hoặc một giá trị chưa được xác định ở vào thời điểm nhập tin nhưng có thể được xác định vào một thời điểm khác.

Nếu thuộc tính có kiểu dữ liệu là vô hướng thì nó được gọi là thuộc tính đơn hoặc thuộc tính nguyên tố; nếu thuộc tính có kiểu dữ liệu có cấu trúc thì ta nói rằng nó là thuộc tính kép (hay không phải là nguyên tố).

Quan hệ (Relation):

Một quan hệ R có n ngôi được định nghĩa trên tập các thuộc tính $U = \{A_1, A_2, \dots, A_n\}$ (thứ tự của các thuộc tính là không quan trọng) và kèm theo nó là một tân từ, tức là một quy tắc để xác định mối quan hệ giữa các thuộc tính A_i và được ký hiệu là $R(A_1, A_2, \dots, A_n)$. Tập thuộc tính của quan hệ R đôi khi còn được ký hiệu là R^+ .

Với A_i là một thuộc tính có miền giá trị là $MGT(A_i)$, như vậy $R(A_1, A_2, \dots, A_n)$ là tập con của tích Đề-các: $MGT(A_1) \times MGT(A_2) \times \dots \times MGT(A_n)$.

Quan hệ còn được gọi bằng thuật ngữ khác là bảng (*Table*).

-Ví dụ 3.1.4:

KHOA (Mã-khoa, Tên-khoa), là một quan hệ 2 ngôi.

Tân từ: *"Mỗi khoa có một tên gọi và một mã số duy nhất để phân biệt với tất cả các khoa khác của trường"*.

-Ví dụ 3.1.5:

LỚP-HỌC (Mã-lớp, Tên-lớp, Niên-khoá, Số-học-viên, Mã-khoa) là quan hệ 5 ngôi với tân từ: *"Mỗi lớp học trong trường có một mã số quy ước duy nhất để phân biệt với tất cả các lớp học khác trong trường; có một tên gọi của lớp học, một số lượng học viên theo học và thuộc một khoa của trường"*.

-Ví dụ 3.1.6:

MÔN-HỌC (Mã-môn, Tên-môn, Số-đv-học-trình) là quan hệ 3 ngôi.

Tân từ: *"Mỗi môn học có một tên gọi cụ thể, được học trong một số đơn vị học trình nhất định và ứng với môn học là một mã số duy nhất để phân biệt với mọi môn học khác"*.

-Ví dụ 3.1.7:

HỌC-VIÊN (Mã-học-viên, Tên-học-viên, Ngày-sinh, Quê-quán, Mã-lớp) là quan hệ 5 ngôi.

Tên từ: "*Mỗi học viên có một họ và tên, ngày sinh, quê quán, ... và được cấp một mã số duy nhất để phân biệt với mọi học viên khác trong trường; học viên được ghi danh vào một lớp học duy nhất trong trường*".

Các thao tác cơ bản trên các quan hệ.

Trong chương này chúng ta chỉ đề cập tới những khái niệm cơ bản do đó các phép toán khác trên các quan hệ sẽ được trình bày chi tiết trong chương V. Ba thao tác cơ bản trên một quan hệ, mà nhờ đó CSDL được thay đổi, đó là Thêm (*Insert*), Xóa (*Delete*) và Sửa (*Update*) các bộ giá trị của quan hệ.

Phép thêm một bộ mới vào quan hệ.

Việc thêm một bộ giá trị mới t vào quan hệ $R (A_1, A_2, \dots A_n)$ làm cho thể hiện T_R của nó tăng thêm một phần tử mới: $T_R = T_R \cup t$. Dạng hình thức của phép thêm bộ mới là:

INSERT (R ; $A_{i1}=v_1, A_{i2}=v_2, \dots A_{im}=v_m$)

trong đó, $A_{i1}, A_{i2}, \dots A_{im}$ là các thuộc tính, và $v_1, v_2, \dots v_m$ là các giá trị thuộc $MGT(A_{i1}), MGT(A_{i2}), \dots, MGT(A_{im})$ tương ứng.

Cần lưu ý rằng các thuộc tính không có tên trong danh sách gán giá trị của bộ t trong câu lệnh INSERT sẽ có giá trị là NULL, tức là giá trị không xác định.

-Ví dụ 3.1.17:

Quan hệ:

HỌC-VIÊN (Mã-học-viên, Tên-học-viên, Ngày-sinh, Quê-quán, Mã-lớp).

Thêm bộ $q5 = (SV002, Hoàng Thị Chính, 17/05/1967, Hà nội, QTKD1)$ vào quan hệ HỌC-VIÊN bởi phép thêm như sau:

INSERT (HỌC-VIÊN; [Mã-học-viên]=Hoàng Thị Chính, [Ngày-sinh]=17/05/1967, [Quê-quán]=Hà nội, [Mã-lớp]=QTKD1).

Thể hiện $T_{HỌC-VIÊN}$ giờ đây là:

$q1 = (SV001, \text{ Nguyễn Văn Nam, } 27/03/1970, \text{ Cần Thơ, } QTKD1)$

$q5 = (SV002, \text{ Hoàng Thị Chính, } 17/05/1967, \text{ Hà nội, } QTKD1)$

$q2 = (SV005, \text{ Vũ Thị Tuyết Mai, } 26/02/1968, \text{ Đồng Nai, } KTKC1)$

$q3 = (SV014, \text{ Hồng Đăng, } 30/04/1975, \text{ Đồng Nai, } CNTK3)$

$q4 = (SV015, \text{ Lê Hoài Nhớ, } 23/03/1965, \text{ Long An, } CNTK4)$

Xin lưu ý là quan hệ HỌC-VIÊN có khóa là Mã-học-viên, do đó bản ghi mới $q5$ được đẩy lên vị trí thứ 2 theo thứ tự tăng dần giá trị của khóa. Cũng vì lý do này, phép thêm bản ghi mới còn được gọi là phép *chèn*. [7].

Nếu xem thứ tự của các thuộc tính là cố định và giá trị v_1, v_2, \dots, v_m là hoàn toàn tương ứng thì phép chèn có thể viết dưới dạng tường minh như sau:

INSERT (R; v_1, v_2, \dots, v_m).

Phép chèn có thể không thực hiện được hoặc làm mất tính nhất quán của dữ liệu trong CSDL vì các lý do:

- Giá trị khóa của bộ mới là rỗng (NULL) hoặc trùng với giá trị khóa của một bộ đã có trong CSDL. Trong trường hợp này hệ quản trị CSDL không cho bổ sung.
- Bộ mới không phù hợp với lược đồ quan hệ. Trường hợp này có thể xảy ra khi người sử dụng làm lẫn thứ tự, kiểu hoặc độ lớn của các thuộc tính. Hệ quản trị CSDL có thể không cho bổ sung nếu không tương thích kiểu giá trị, hoặc vẫn cho bổ sung bộ mới nhưng tính nhất quán dữ liệu không được đảm bảo.
- Một số giá trị của bộ mới không thuộc miền giá trị của thuộc tính tương ứng. Trong trường hợp này, nếu quan hệ đã được đảm bảo tính nhất quán bởi các RBTV về miền giá trị thì hệ quản trị CSDL sẽ không cho bổ sung, nếu không có RBTV như vậy thì tính nhất quán của dữ liệu bị vi phạm mà hệ quản trị CSDL không phát hiện được.

Phép loại bỏ bộ khỏi quan hệ.

Phép loại bỏ (hoặc xóa bỏ) một bộ t của quan hệ sẽ lấy đi (những) bộ t khỏi thể hiện của quan hệ. $TR = TR \setminus t$. Phép loại bỏ được viết một cách hình thức như sau:

DELETE (R; $A_{i1}=v_1, A_{i2}=v_2, \dots, A_{im}=v_m$).

trong đó $A_{ij}=v_j$ ($j = 1, 2, \dots, m$) được coi như những điều kiện thỏa một số thuộc tính của bộ t để loại bỏ một bộ ra khỏi quan hệ.

Ví dụ 3.1.18:

Quan hệ:

HQC-VIÊN (Mã-học-viên, Tên-học-viên, Ngày-sinh, Quê-quán, Mã-lớp)

Với phép loại bỏ như sau:

DELETE (HQC-VIÊN; [Quê-quán]=Đồng nai).

Thì các bộ:

q2 = (SV005, Vũ Thị Tuyết Mai, 26/02/1968, Đồng Nai, KTKC1)

q3 = (SV014, Hồng Đăng, 30/04/1975, Đồng Nai, CNTK3)

sẽ bị loại bỏ ra khỏi quan hệ HQC-VIÊN bởi vì cùng có chung Quê-quán là Đồng nai.
Thể hiện THQC-VIÊN lúc này là:

q1 = (SV001, Nguyễn Văn Nam, 27/03/1970, Cần Thơ, QTKD1)

q5 = (SV002, Hoàng Thị Chính, 17/05/1967, Hà nội, QTKD1)

q4 = (SV015, Lê Hoài Nhớ, 23/03/1965, Long An, CNTK4)

Phép sửa đổi giá trị của các thuộc tính của quan hệ.

Dữ liệu của CSDL đôi khi cũng cần phải được đổi mới theo thời gian hoặc sửa lại cho đảm bảo tính chính xác hoặc nhất quán của dữ liệu. Do đó thao tác sửa dữ liệu (*Update*) là rất cần thiết. Một số hệ quản trị CSDL đưa ra nhiều câu lệnh khác nhau để sửa đổi dữ liệu: EDIT, CHANGE, BROW, UPDATE (như DBase, FoxPro v.v...). Trong ngôn ngữ hình thức, mục này đưa ra một dạng của phép sửa đổi giá trị các bộ của quan hệ:

UPDATE (R; A_{i1}=c₁, A_{i2}=c₂, ... A_{im}=c_m; A_{i1}=v₁, A_{i2}=v₂, ... A_{im}=v_m).

Trong đó R là quan hệ cần thực hiện sửa đổi; A_{ij}=c_j (j = 1, 2, ..., m) là điều kiện tìm kiếm bộ giá trị để sửa và A_{ij}=v_j (j = 1, 2, ..., m) là giá trị mới của bộ.

Ví dụ 3.1.19:

Quan hệ

HQC-VIÊN (Mã-học-viên, Tên-học-viên, Ngày-sinh, Quê-quán, Mã-lớp)

Với phép sửa đổi giá trị như sau:

UPDATE (HQC-VIÊN; [Mã-học-viên]=SV015, [Quê-quán]=Sông Bé)

thì giá trị của bộ q4 được sửa lại thành:

$q4 = (SV015, \text{Lê Hoài Nhớ}, 23/03/1965, \text{Sông Bé}, CNTK4)$

Ràng buộc toàn vẹn

Ràng buộc toàn vẹn trên một CSDL

Ràng buộc toàn vẹn (*Integrity Constraint / Rule* viết tắt là: RBTV) và kiểm tra sự vi phạm ràng buộc toàn vẹn là hai trong những vấn đề rất quan trọng trong quá trình phân tích, thiết kế và khai thác CSDL. Trong quá trình phân tích - thiết kế cơ sở dữ liệu, nếu không quan tâm đúng mức đến những vấn đề trên, thì có thể dẫn đến những hậu quả rất nghiêm trọng về tính an toàn và toàn vẹn dữ liệu, đặc biệt trong những CSDL tương đối lớn.

Chương này được chia làm 3 mục nhỏ: Mục thứ nhất trình bày các khái niệm cơ bản của ràng buộc toàn vẹn; mục thứ hai sẽ giới thiệu các cách phân loại ràng buộc toàn vẹn và phần thứ ba sẽ đề cập đến một công cụ để biểu diễn một số các ràng buộc toàn vẹn: Phụ thuộc hàm.

Ràng buộc toàn vẹn, các yếu tố của RBTV.

Định nghĩa.

Ràng buộc toàn vẹn là một điều kiện bất biến không được vi phạm trong một CSDL.

Trong một CSDL, luôn luôn tồn tại rất nhiều mối liên hệ ảnh hưởng qua lại lẫn nhau giữa các thuộc tính của một quan hệ, giữa các bộ giá trị trong một quan hệ và giữa các thuộc tính của các bộ giá trị trong các quan hệ với nhau. Các mối quan hệ phụ thuộc lẫn nhau này chính là những điều kiện bất biến mà tất cả các bộ của những quan hệ có liên quan trong cơ sở dữ liệu đều phải thỏa mãn ở bất kỳ thời điểm nào. Ràng buộc toàn vẹn còn được gọi là các quy tắc quản lý (*Rules*) được áp đặt lên trên các đối tượng của thế giới thực.

Ví dụ 4.1.1:

Trong CSDL về quản lý học viên của một trường học đã cho trong các ví dụ của mục 3.1 trong Chương III (Bài 4), chúng ta có một số ràng buộc toàn vẹn như sau:

R₁ : Mỗi lớp học phải có một mã số duy nhất để phân biệt với mọi lớp học khác trong trường.

R₂ : Mỗi lớp học phải thuộc một KHOA của trường.

R₃ : Mỗi học viên có một mã số riêng biệt, không trùng với bất cứ học viên nào khác.

R4 : Mỗi học viên phải đăng ký vào một lớp của trường.

R5 : Mỗi học viên được thi tối đa 3 lần cho mỗi môn học.

R6 : Tổng số học viên của một lớp phải lớn hơn hoặc bằng số lượng đếm được của lớp tại một thời điểm.

Khóa nội, Khóa ngoại, giá trị NOT NULL ... là những RBTV về miền giá trị của các thuộc tính. Những RBTV vừa nêu trên cũng mới chỉ là những RBTV đơn giản trong CSDL nhỏ về quản lý học viên. Trong thực tế, tất cả các RBTV của một cơ sở dữ liệu phải được người phân tích thiết kế phát hiện đầy đủ và mô tả một cách chính xác, rõ ràng trong hồ sơ phân tích, thiết kế.

Trong một CSDL, ràng buộc toàn vẹn được xem như một công cụ để diễn đạt ngữ nghĩa của cơ sở dữ liệu đó. Trong suốt quá trình khai thác cơ sở dữ liệu, các RBTV đều phải được thỏa mãn ở bất kỳ thời điểm nào nhằm đảm bảo cho CSDL luôn luôn ở trạng thái an toàn và nhất quán về dữ liệu.

Các hệ quản trị CSDL thường có các cơ chế tự động kiểm tra các RBTV về miền giá trị của Khóa nội, Khóa ngoại, giá trị NOT NULL qua khai báo cấu trúc các bảng (mô hình dữ liệu của quan hệ) hoặc thông qua những thủ tục kiểm tra và xử lý vi phạm RBTV do những người phân tích - thiết kế cài đặt. Việc kiểm tra RBTV có thể được tiến hành vào một trong các thời điểm sau:

-Kiểm tra ngay khi thực hiện một thao tác cập nhật CSDL (thêm, sửa, xóa). Thao tác cập nhật chỉ được xem là hợp lệ nếu như nó không vi phạm bất cứ một RBTV nào, nghĩa là nó không làm mất tính toàn vẹn dữ liệu của CSDL. Nếu vi phạm RBTV, thao tác cập nhật bị coi là không hợp lệ và sẽ bị hệ thống hủy bỏ (hoặc có một xử lý thích hợp nào đó).

-Kiểm tra định kỳ hay đột xuất, nghĩa là việc kiểm tra RBTV được tiến hành một cách độc lập đối với thao tác cập nhật dữ liệu. Đối với những trường hợp vi phạm RBTV, hệ thống sẽ có những xử lý ngầm định hoặc yêu cầu người sử dụng xử lý những sai sót một cách tường minh.

Các yếu tố của ràng buộc toàn vẹn:

Khi xác định một RBTV cần chỉ rõ:

Điều kiện (tức là nội dung) của RBTV, từ đó xác định cách biểu diễn.

Bối cảnh xảy ra RBTV : trên một hay nhiều quan hệ, cụ thể trên các quan hệ nào.

Tầm ảnh hưởng của RBTV. Khả năng tính toàn vẹn dữ liệu bị vi phạm, và

Hành động cần phải có khi phát hiện có RBTV bị vi phạm.

Điều kiện của RBTV:

Điều kiện của RBTV là sự mô tả, và biểu diễn hình thức nội dung của nó, có thể được biểu diễn bằng ngôn ngữ tự nhiên, thuật giải (bằng mã giả - *Pseudo Code*, ngôn ngữ tựa Pascal), ngôn ngữ đại số tập hợp, đại số quan hệ v.v hoặc bằng các phụ thuộc hàm (sẽ được trình bày chi tiết trong mục 3 của chương này).

Ví dụ 4.1.2:

Giả sử có một CSDL quản lý hóa đơn bán hàng gồm các bảng sau:

HÓAĐƠN (Số-hóa-đơn, Số-chủng-loại-mặt-hàng, Tổng-trị-giá).

DM_HÀNG (Mã-hàng, Tên-hàng, Đơn-vị-tính).

CHITIẾT_HĐ (Số-hóa-đơn, Mã-hàng, Số-lượng-đặt, Đơn-giá, Trị-giá).

Điều kiện của ràng buộc toàn vẹn có thể biểu diễn như sau:

R₁ : “Mỗi hóa đơn có một Số hóa đơn riêng biệt, không trùng với hóa đơn khác”:

" $hđ1, hđ2 \in$ HÓAĐƠN, $hđ1 \neq hđ2 \Rightarrow hđ1.Số-hóa-đơn \neq hđ2.Số-hóa-đơn$.

R₂ : “Số-chủng-loại-mặt-hàng = số bộ của CHITIẾT_HĐ có cùng Số-hóa-đơn”:

" $hđ \in$ HÓAĐƠN thì:

$hđ.Số-chủng-loại-mặt-hàng = COUNT (cthđ \in CHITIẾT_HĐ, cthđ.Số-hóa-đơn = hđ.Số-hóa-đơn)$

R₃ : “Tổng các trị giá của các mặt hàng trong CHITIẾT_HĐ có cùng Số-hóa-đơn phải bằng Tổng-trị-giá ghi trong HÓAĐƠN”:

" $hđ \in$ HÓAĐƠN thì:

$hđ.Tổng-trị-giá = SUM (cthđ.Tri-gia) \text{ đối với các } cthđ \in CHITIẾT_HĐ \text{ sao cho : } cthđ.Số-hóa-đơn = hđ.Số-hóa-đơn$.

R₄ : “Mỗi bộ của CHITIẾT_HĐ phải có mã hàng thuộc về danh mục hàng”:

CHITIẾT_HĐ [Mã-hàng] Í DM_HÀNG[Mã-hàng]

hoặc biểu diễn bằng cách khác:

" *cthd* Ỳ CHITIẾT_HĐ, \$ *hh* Ỳ DM_HÀNG

sao cho: *cthd*.Mã-hàng=*hh*.Mã-hàng.

Ghi chú: Chúng ta quy ước: chữ COUNT (...) nghĩa là đếm số bộ giá trị của một quan hệ thỏa mãn điều kiện đã cho trong dấu ngoặc tròn. Chữ SUM (...) nghĩa là lấy tổng các giá trị của các bộ trên một quan hệ thỏa mãn điều kiện đã cho trong dấu ngoặc tròn. Các ngôn ngữ quản trị CSDL đều có các thủ tục hàm này để hỗ trợ tính toán. Chương VI (Bài 9 và 10) sẽ trình bày ngôn ngữ truy vấn CSDL có cấu trúc (SQL) trong đó có đề cập tới các hàm này.

Bối cảnh của RBTV :

Bối cảnh có thể định nghĩa trên một quan hệ cơ sở hay nhiều quan hệ cơ sở. Đó là những quan hệ mà RBTV được áp dụng trên đó.

Như trong ví dụ 4.1.2, bối cảnh của ràng buộc toàn vẹn R_1 chỉ là một quan hệ HÓAĐƠN; bối cảnh của ràng buộc toàn vẹn R_2 và R_3 là hai quan hệ HÓAĐƠN và CHITIẾT_HĐ; bối cảnh của ràng buộc toàn vẹn R_4 là hai quan hệ CHITIẾT_HĐ và DM_HÀNG.

Xác định tầm ảnh hưởng của RBTV:

Một RBTV có thể liên quan đến một số quan hệ, và chỉ khi có thao tác cập nhật (Thêm, Sửa, Xóa) mới có nguy cơ dẫn đến vi phạm RBTV, do đó cần xác định rõ thao tác nào dẫn đến việc cần phải kiểm tra RBTV.

Trong quá trình phân tích, thiết kế một CSDL, người phân tích cần lập bảng xác định tầm ảnh hưởng cho mỗi ràng buộc toàn vẹn nhằm xác định khi nào thì phải tiến hành kiểm tra các ràng buộc toàn vẹn đó. Bảng này gồm 4 cột: cột 1 là cột chủ từ chứa tên các quan hệ liên quan tới RBTV; 3 cột tiếp theo là thao tác Thêm / Sửa / Xóa bộ giá trị của quan hệ. Nếu RBTV cần được kiểm tra nguy cơ dẫn tới vi phạm thì tại ô (giao điểm dòng và cột) đó người ta đánh dấu bằng dấu gạch chéo (x) hoặc dấu cộng (+), và có thể chỉ rõ thêm các thuộc tính nào nếu được cập nhật mới dẫn đến vi phạm RBTV bằng cách liệt kê chúng dưới dấu (x) hoặc dấu (+). Nếu RBTV không có nguy cơ bị vi phạm khi cập nhật CSDL thì đánh dấu trừ (-) vào ô tương ứng. Nếu không bị vi phạm vì không được phép sửa đổi thì ký hiệu là trừ với dấu sao (-(*)).

Ví dụ 4.1.3:

Bảng tầm ảnh hưởng của ràng buộc toàn vẹn R₁

Quan hệ	Thêm	Sửa	Xóa
HÓAĐƠN	+ (Số-hóa-đơn)	- (*)	+

Bảng tầm ảnh hưởng của ràng buộc toàn vẹn R₂

Quan hệ	Thêm	Sửa	Xóa
HÓAĐƠN	-	+ (Số-chủng-loại-mặt-hàng)	+
CHITIẾT_HĐ	+	-	+

Bảng tầm ảnh hưởng của ràng buộc toàn vẹn R₃

Quan hệ	Thêm	Sửa	Xóa
HÓAĐƠN	-	+ (Tổng-trị-giá)	+
CHITIẾT_HĐ	+	+ (Trị-giá)	+

Bảng tầm ảnh hưởng của ràng buộc toàn vẹn R₄

Quan hệ	Thêm	Sửa	Xóa
CHITIẾT_HĐ	+ (Mã-hàng)	- (*)	+
DM_HÀNG	-	- (*)	+

Trong thực tế, ràng buộc toàn vẹn R₁ là không cần thiết, bởi thuộc tính Số-hóa-đơn là khóa của quan hệ HÓAĐƠN, do vậy nó luôn luôn phải là duy nhất và không được phép chứa giá trị rỗng; đồng thời không được phép sửa đổi (Xem lại Chương III, mục 3.1, điểm 3.1.6 - *một số quy ước về khóa*. Bài 4).

Sau khi xây dựng các bảng tầm ảnh hưởng của từng RBTV trên các quan hệ liên quan, cần phải tổng hợp lại bằng cách xây dựng một bảng tầm ảnh hưởng tổng hợp các RBTV nhằm xác định tất cả các RBTV cần phải kiểm tra trên từng quan hệ. Bảng này gồm cột chủ từ là các RBTV, các cột còn lại là các thao tác Thêm (T), Sửa (S) và Xóa (X) của từng quan hệ nằm trong bối cảnh của các RBTV trong CSDL.

Ví dụ 4.1.4:

Lập bảng tầm ảnh hưởng tổng hợp của các RBTV trong CSDL quản lý hóa đơn bán hàng nêu trên:

Q.Hệ	HÓAĐƠN	CHITIẾT_HĐ	DM_HÀNG						
RBTV	T	S	X	T	S	X	T	S	X
R ₁	+ (Số-hđ)	- (*)	+						
R ₂	-	+ (Số-loại-MH)	+	+	- (*)	+			
R ₃	-	+ (Tổng-TG)	+	+	+ (Trị-giá)	+			
R ₄				+	-	+	-	- (*)	+

Nhìn vào bảng tổng hợp trên chúng ta có thể thấy quan hệ HÓAĐƠN khi thêm và xóa một bộ giá trị, phải kiểm tra ràng buộc toàn vẹn R₁, R₂ và R₃; khi sửa giá trị thuộc tính *Số-chủng-loại-mặt-hàng* thì phải kiểm tra ràng buộc toàn vẹn R₂ và khi sửa giá trị thuộc tính *Tổng-trị-giá* thì phải kiểm tra ràng buộc toàn vẹn R₃. Quan hệ CHITIẾT_HĐ khi được cập nhật cần kiểm tra 2 RBTV: R₂ và R₃; Quan hệ DM_HÀNG cần kiểm tra ràng buộc toàn vẹn R₄ khi xóa một bộ giá trị.

Hành động khi RBTV bị vi phạm:

Khi một RBTV bị vi phạm cần có những hành động thích hợp. Thông thường có 2 giải pháp:

Phân loại ràng buộc toàn vẹn

Trong một CSDL có thể phát hiện nhiều RBTV, tuy nhiên có thể chia chúng thành hai loại chính theo bối cảnh RBTV:

Ràng buộc toàn vẹn trong bối cảnh là một quan hệ cơ sở.

Ràng buộc toàn vẹn có bối cảnh trên nhiều quan hệ cơ sở.

Chúng ta sẽ xem xét chi tiết các loại RBTV chi tiết trong từng bối cảnh nêu trên.

Ràng buộc toàn vẹn có bối cảnh là 1 quan hệ cơ sở.

Trên một quan hệ cơ sở có thể tồn tại nhiều RBTV thuộc các loại: RBTV về miền giá trị của thuộc tính, RBTV về giá trị giữ thuộc tính này với (các) thuộc tính khác (*gọi là liên*

thuộc tính) và giữa các giá trị của các bộ giá trị khác nhau (gọi là liên bộ - liên thuộc tính)

Ràng buộc toàn vẹn về miền giá trị của thuộc tính.

Trong hầu hết các CSDL quan hệ, loại RBTV này là rất phổ biến. Như chúng ta đã biết, thuộc tính được đặc trưng không chỉ bởi kiểu giá trị mà nó còn bị giới hạn bởi miền giá trị trong kiểu dữ liệu đó. Do đó, khi thực hiện các thao tác cập nhật Thêm / Sửa bộ giá trị mới cho quan hệ đều phải kiểm tra RBTV này.

Ví dụ 4.2.1:

Trong quan hệ KQUẢ-THI mô tả trong ví dụ 3.1.12, do quy định mỗi học viên chỉ được thi một môn học tối đa là 3 lần, hiển nhiên là điểm thi của mỗi môn học trong mọi lần thi không bị âm và không vượt quá 10. Có 2 ràng buộc toàn vẹn về miền giá trị trong quan hệ này:

R₁: "kq YKQUẢ-THI thì 0 ≤ kq.Lần-thi ≤ 3

R₂: "kq YKQUẢ-THI thì 0 ≤ kq.Điểm-thi ≤ 10

Giả sử các giảng viên có “châm chước” thêm rằng điểm thi lần sau không nhỏ hơn điểm thi lần trước đó. Chúng ta có thêm ràng buộc toàn vẹn về miền giá trị:

R₃: "kq YKQUẢ-THI / kq.Điểm-thi (lần trước) ≤ kq.Điểm-thi ≤ 10.0

Ràng buộc toàn vẹn liên thuộc tính.

Đó là loại RBTV có liên quan tới nhiều thuộc tính của một quan hệ. Thông thường đó là các phụ thuộc tính toán, hoặc một suy diễn từ giá trị của một hay nhiều thuộc tính trong cùng một bộ giá trị.

Ví dụ 4.2.2:

Quan hệ CHITIẾT_HĐ trong CSDL quản lý hóa đơn bán hàng cho trong ví dụ 4.1.2 tại mục 4.1 trình bày trên, có RBTV liên thuộc tính là:

"cthđ Y CHITIẾT_HĐ / cthđ.Trị-giá = cthđ.Số-lượng-đặt * cthđ.Đơn-giá.

Ví dụ 4.2.3:

Quan hệ danh sách cán bộ - công chức Nhà nước CBCC với tập các thuộc tính: { Mã-đơn-vị, Mã-CBCC, Họ-tên, Giới-tính, Ngày-sinh, Ngày-tuyển-dụng, Ngạch-CBCC, Bậc, Hệ-số-lương, Ngày-xếp-lương }.

Với quy định nam từ 18 đến 60 và nữ từ 18 đến 55 tuổi và phải từ 18 tuổi trở lên mới được tuyển vào làm công chức Nhà nước. Chúng ta có các RBTV về miền giá trị liên thuộc tính như sau:

R1: "*cc* Ỳ CBCC / nếu *cc*.Giới-tính = Nam thì (Now() - *cc*.Ngày_sinh) / 365 trong khoảng 18 và 60. Nếu *cc*.Giới-tính = Nữ thì (Now() - *cc*.Ngày-sinh) / 365 trong khoảng 18 và 55.

R2: "*cc* Ỳ CBCC / (*cc*.Ngày-tuyen-dung - *cc*.Ngày-sinh) / 365 ³ 18 và *cc*.Ngày-tuyen-dung ≤ Now().

Ghi chú:

Now() là lấy ngày tháng năm hiện tại và một năm trung bình có 365 ngày;

Hiệu 2 giá trị ngày tháng là số ngày trôi qua giữa 2 ngày đó.

Ràng buộc toàn vẹn liên bộ, liên thuộc tính.

Đây là loại RBTV có liên quan tới nhiều bộ và có thể tới nhiều thuộc tính của (các) bộ giá trị trong một quan hệ.

Ví dụ 4.2.4:

Trong ví dụ 4.2.1 vừa nêu, chúng ta thấy điểm thi không chỉ liên quan đến thuộc tính Lần-thi mà còn liên quan tới điểm thi của lần thi trước đó nếu đã thi 1 hay 2 lần rồi. RBTV đầy đủ phải được diễn đạt bằng thuật toán như sau:

R3: "*kq* Ỳ KQUẢ-THI / Nếu *kq*.Lần-thi = 1 thì 0 ≤ *kq*.Điểm-thi ≤ 10.0

hoặc:

Nếu *kq*.Lần thi > 1 thì \$*kq*' Ỳ KQUẢ-THI

sao cho *kq*'.Lần-thi = *kq*.Lần-thi - 1 và *kq*.Điểm-thi ³ *kq*'.Điểm-thi.

Ví dụ 4.2.5:

Giả thiết thêm rằng, để xác định hệ số lương của một cán bộ - công chức Nhà nước căn cứ vào Ngạch công chức và Bậc được xếp, chúng ta có thêm bảng quy định Ngạch, bậc và hệ số lương cán bộ công chức Nhà nước theo Nghị định 25CP của Thủ tướng Chính phủ:

NGẠCH-BẬC-LƯƠNG (Mã-ngạch, Bậc, Hệ-số-lương).

Tân từ: Ứng với một Ngạch công chức và một Bạc cụ thể thì có một hệ số lương tương ứng (từ 1.0 đến 10.0).

Việc biểu diễn ràng buộc toàn vẹn Hệ-số-lương phụ thuộc vào Ngạch và Bạc của quan hệ CBCC nêu trong ví dụ 4.2.3 nêu trên bằng thuật giải có thể trở nên rắc rối. Người ta đã đưa thêm một cách biểu diễn mới để làm cho RBTV trở nên rõ ràng hơn, đó là cách biểu diễn RBTV bằng phụ thuộc hàm mà chúng ta sẽ trình bày rõ hơn trong mục 4.3 của chương này.

Ràng buộc toàn vẹn định nghĩa trên nhiều quan hệ cơ sở.

Ràng buộc toàn vẹn về phụ thuộc tồn tại.

Ràng buộc toàn vẹn về phụ thuộc tồn tại (*Existential Dependency* hay *Referential Dependency*) còn được gọi là *phụ thuộc về khóa ngoại*. Đây là loại RBTV khá phổ biến trong các CSDL bởi các quan hệ trong một CSDL luôn luôn có mối quan hệ mật thiết với nhau. Bộ giá trị của quan hệ này được thêm vào một cách hợp lệ nếu tồn tại một bản ghi tương ứng của một quan hệ khác.

Phụ thuộc tồn tại xảy ra nếu có một trong hai trường hợp sau:

- (i) Có sự hiện diện của khóa ngoại.
- (ii) Có sự lồng khóa giữa các quan hệ.

Ví dụ 4.2.6:

Trong thể hiện của quan hệ CHITIẾT_HĐ, sự tồn tại của mỗi bộ giá trị *cthđ* đều phụ thuộc vào sự tồn tại của một bộ giá trị *hđ* trong thể hiện của quan hệ HÓAĐƠN sao cho *hđ.Số-hóa-đơn* = *cthđ.Số-hóa-đơn*, và phụ thuộc cả vào sự tồn tại của một bộ giá trị *mh* trong thể hiện của quan hệ DM_HÀNG sao cho *mh.Mã-hàng* = *cthđ.Mã-hàng*.

Biểu diễn các RBTV này như sau:

RBTV₁ : “Mỗi bộ của CHITIẾT_HĐ phải có một hóa đơn với Số-hóa-đơn tương ứng”:

"cthđ Y CHITIẾT_HĐ, \$hđ Y HÓAĐƠN

sao cho cthđ.Số-hóa-đơn = hđ.Số-hóa-đơn.

hoặc biểu diễn bằng cách khác:

CHITIẾT_HĐ [Số-hóa-đơn] Í HÓAĐƠN [Số-hóa-đơn]

RBTV₂ : “Mỗi bộ của CHITIẾT_HĐ phải có mã hàng thuộc về danh mục hàng”:

"cthd Y CHITIẾT_HĐ, \$hh Y DM_HÀNG sao cho cthd.Mã-hàng=hh.Mã-hàng

hoặc biểu diễn bằng cách khác:

CHITIẾT_HĐ [Mã-hàng] Í DM_HÀNG [Mã-hàng]

Ví dụ 4.2.7:

Trong CSDL về quản lý CBCC nêu trong ví dụ 4.2.3 và 4.2.5 ở trên, RBTV về phụ thuộc tồn tại giữa 2 quan hệ CBCC và NGẠCH-BẠC-LUƠNG được xác định:

"cbcc Y CBCC, \$ng Y NGẠCH-BẠC-LUƠNG

sao cho (cbcc.Mã-ngạch = ng.Mã-ngạch) (cbcc.Bậc = ng.Bậc)

hoặc biểu diễn bằng cách khác:

CBCC [Mã-ngạch, Bậc] Í NGẠCH-BẠC-LUƠNG [Mã-ngạch, Bậc]

Ví dụ 4.2.8:

Trong CSDL về quản lý học viên đã nêu trong ví dụ 3.1.12, các RBTV về phụ thuộc tồn tại gồm:

RBTV₁ : “Mỗi LỚP-HỌC đều phải thuộc một KHOA nhất định”:

"lh Y LỚP-HỌC, \$kh Y KHOA sao cho lh.Mã-khoa = kh.Mã-khoa.

hoặc biểu diễn qua phép chiếu quan hệ:

LỚP-HỌC [Mã-khoa] Í KHOA [Mã-khoa].

RBTV₂ : “Mỗi HỌC-VIÊN đều phải thuộc một LỚP-HỌC nhất định”:

"hv Y HỌC-VIÊN, \$lh Y LỚP-HỌC sao cho hv.Mã-lớp = lh.Mã-lớp.

hoặc biểu diễn qua phép chiếu quan hệ:

$HQC-VIEN [Mã-lớp] \overset{I}{\rightarrow} LOP-HQC [Mã-lớp]$.

RBTV₃ : “Mỗi KQUẢ-THI đều phải là của một HỌC-VIÊN nhất định”:

" $kq \ Y \ KQUẢ-THI, \$_{hv} \ Y \ HỌC-VIÊN$

sao cho $kq.Mã-học-viên = hv.Mã-học-viên$.

hoặc biểu diễn qua phép chiếu quan hệ:

$KQUẢ-THI [Mã-học-viên] \overset{I}{\rightarrow} HỌC-VIÊN [Mã-học-viên]$.

RBTV₄ : “Mỗi môn thi trong KQUẢ-THI đều phải có tên trong danh sách các môn học”
:

" $kq \ Y \ KQUẢ-THI, \$_{mh} \ Y \ MÔN-HQC$ sao cho $kq.Mã-môn = mh.Mã-môn$

hoặc biểu diễn qua phép chiếu quan hệ:

$KQUẢ-THI [Mã-môn] \overset{I}{\rightarrow} MÔN-HQC [Mã-môn]$

Chúng ta có thể thấy về phải của phép toán tập con ($\overset{I}{\rightarrow}$) là phép chiếu trên thuộc tính khóa nội của một quan hệ, còn về trái là phép chiếu trên tập các thuộc tính khóa ngoại của một quan hệ khác. Chính vì lẽ đó mà người ta còn gọi RBTV loại này là RBTV về khóa ngoại. Phát biểu tổng quát về loại RBTV này là như sau:

R và S là hai quan hệ định nghĩa trên các tập thuộc tính R^+ và S^+ . $K_R \overset{I}{\rightarrow} R^+$ là tập các thuộc tính khóa nội của quan hệ R; $K_S \overset{I}{\rightarrow} S^+$ là tập các thuộc tính khóa nội của quan hệ S; và $W \overset{I}{\rightarrow} S^+$ là tập các thuộc tính khóa ngoại của S đối với R. Khi đó ta có phụ thuộc tồn tại của S vào R và được biểu diễn thông qua phép chiếu:

$S [W] \overset{I}{\rightarrow} R [W]$.

Nếu $W \overset{I}{\rightarrow} K_S$, thì ta nói rằng có sự *lồng khóa* giữa hai quan hệ R và S.

Trong bảng tầm ảnh hưởng của loại RBTV này, các thao tác *Thêm* và *Sửa* một bộ giá trị của quan hệ R (về phải của phụ thuộc tồn tại) không gây ra sự vi phạm RBTV (trừ khi có sự lồng khóa giữa R với một quan hệ khác), chỉ có thao tác *Xóa* bỏ một bộ giá trị của R mới cần có sự kiểm tra RBTV. Ngược lại, thao tác *Xóa* một bộ giá trị của S

không gây ra sự vi phạm RBTV (trừ khi có sự lỏng khóa của một quan hệ khác vào S), thao tác *Thêm* một bộ giá trị mới vào S luôn luôn phải được kiểm tra RBTV này; nếu W là các thuộc tính khóa ngoại của S thì việc *Sửa* bộ giá trị của S trên các thuộc tính khóa ngoại W vẫn phải kiểm tra RBTV; nếu có sự lỏng khóa thì việc sửa không đòi hỏi kiểm tra RBTV vì theo quy ước là không được sửa giá trị của thuộc tính khóa.

Bảng tầm ảnh hưởng có 2 dạng ứng với 2 trường hợp trên như sau:

a. Ứng với trường hợp khóa ngoại:

Quan hệ	Thêm	Sửa	Xóa
R	-	- (*)	+
S	+	+(W)	-

a. Ứng với trường hợp lỏng khóa:

Quan hệ	Thêm	Sửa	Xóa
R	-	- (*)	+
S	+	- (*)	-

Ràng buộc toàn vẹn liên bộ - liên quan hệ.

Khi có sự hiện diện của 1 thuộc tính mang tính chất tổng hợp (tức là giá trị của thuộc tính có thể được tính toán từ giá trị của các thuộc tính khác trên một hay nhiều bộ giá trị của các quan hệ trong CSDL), hay phụ thuộc tồn tại lỏng khóa thì có RBTV liên quan hệ - liên bộ.

Ví dụ 4.2.9 :

Xét CSDL về quản lý học viên nêu trong ví dụ 3.1.12 (Chương III, mục 3.1. Bài 4), RBTV liên quan hệ - liên bộ có thể được xác định: “Với mọi bộ giá trị của LỚP-HỌC, nếu Số lượng học viên lớn hơn 0 thì số lượng này phải lớn hơn hay bằng tổng số bộ giá trị đếm được của các học viên có cùng Mã lớp”. Đây chính là RBTV R₆ đã nêu trong ví dụ 4.1.1.

Biểu diễn hình thức của RBTV này như sau:

" lh Ỳ LỚP-HỌC thì:

nếu lh.Số-học-viên > 0 thì:

h.Số-học-viên = COUNT (hv Ỳ HỌC-VIÊN, hv.Mã-lớp = lh.Mã-lớp).

Ví dụ 4.2.10 :

Xét CSDL quản lý hóa đơn bán hàng đã cho trong ví dụ 4.1.2 với 3 quan hệ:

1) HÓAĐƠN (Số-hóa-đơn, Số_chủng_loại_mặt_hàng, Tổng-trị-giá).

2) CHITIẾT_HĐ (Số-hóa-đơn, Mã-hàng, Số-lượng-đặt, Đơn-giá, Trị-giá).

3) DM_HÀNG (Mã-hàng, Tên-hàng, Đơn-vị-tính).

RBTV₁ : “Số_chủng_loại_mặt_hàng = số bộ của CHITIẾT_HĐ có cùng Số-hóa-đơn” :

" hđ Ỳ HÓAĐƠN thì:

hđ.Số-chủng-loại-mặt-hàng = COUNT (cthđ Ỳ CHITIẾT_HĐ, cthđ.Số-hóa-đơn = hđ.Số-hóa-đơn)

RBTV₂ : “Tổng tất cả các Trị-giá của các mặt hàng trong CHITIẾT_HĐ có cùng Số-hóa-đơn phải bằng Tổng-trị-giá của hóa đơn đó trong HÓAĐƠN”:

" hđ Ỳ HÓAĐƠN thì hđ.Tổng-trị-giá = SUM (cthđ.Trị-giá)

đối với các cthđ Ỳ CHITIẾT_HĐ

sao cho : cthđ. Số-hóa-đơn= hđ. Số-hóa-đơn.

Chúng ta có thể nhận thấy trong CSDL này có sự dư thừa thông tin một cách cố ý. Đó là thuộc tính tính toán *Trị-giá* của các mặt hàng trong chi tiết hoá đơn bán hàng. Một trong những phương pháp kiểm định tính đúng đắn của dữ liệu được nhập vào là tổ chức nhập “*thừa*” dữ liệu tính toán được (*Computable Value*) rồi so sánh với công thức tính toán. Nếu có sự sai sót nào trong các thành phần có liên quan trong công thức thì logic biểu thức sẽ không còn phù hợp nữa.

Bây giờ để đạt được dạng chuẩn (*Normal Form*) tốt hơn cho quan hệ CHITIẾT-HĐ, chúng ta có thể loại bỏ thuộc tính Trị-giá. Khi đó RBT₂ được viết lại là:

" *hđ* Ỗ HÓAĐƠN thì

hđ.Tổng-trị-giá = SUM (*cthđ*.Số-lượng-đặt * *cthđ*.Đơn-giá)

đối với các *cthđ* Ỗ CHITIẾT_HĐ

sao cho : *cthđ*. Số-hóa-đơn = *hđ*. Số-hóa-đơn.

(1) Đưa ra thông báo và yêu cầu sửa chữa dữ liệu của các thuộc tính cho phù hợp với quy tắc đảm bảo tính nhất quán dữ liệu. Thông báo phải đầy đủ và tạo được sự thân thiện với người sử dụng. Giải pháp này là phù hợp cho việc xử lý thời gian thực.

(2) Từ chối thao tác cập nhật. Giải pháp này là phù hợp đối với việc xử lý theo lô (*Batch processing*). Việc từ chối cũng phải được lưu lại bằng những thông báo đầy đủ, rõ ràng vì sao thao tác bị từ chối và cần phải sửa lại những dữ liệu nào.

Ràng buộc toàn vẹn các loại

RBTV do sự hiện diện của chu trình

Bây giờ chúng ta biểu diễn cấu trúc CSDL dưới dạng đồ thị như sau: Mỗi nút của đồ thị biểu diễn một quan hệ hoặc một thuộc tính.

(i) Quan hệ được biểu diễn bằng nút tròn trắng (o), và

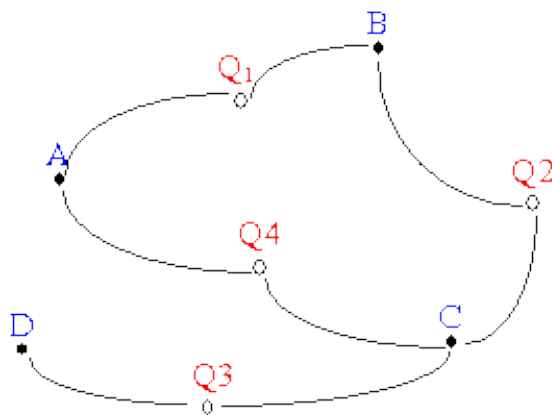
(ii) Thuộc tính được biểu diễn bởi một nút tròn đen nhỏ hơn (·).

(iii) Tất cả các nút đều được chỉ rõ bằng tên của quan hệ hoặc thuộc tính. Thuộc tính thuộc một quan hệ được biểu diễn bởi một cung nối giữa nút tròn trắng và nút tròn đen đó.

Nếu trên đồ thị chúng ta thấy xuất hiện một đường khép kín thì ta nói rằng trong lược đồ CSDL có sự hiện diện của chu trình. Sự hiện diện này làm nảy sinh một vấn đề mới: Xác định khả năng đảm bảo tính nhất quán của dữ liệu, đó là một *RBTV, do sự hiện diện của chu trình*.

Ví dụ 4.2.11:

Hãy xét lược đồ CSDL với các lược đồ quan hệ con: $Q_1(A, B)$, $Q_2(B, C)$, $Q_3(C, D)$ và $Q_4(A, C)$. Biểu diễn lược đồ CSDL bằng đồ thị như sau:



Hình 4.2.1 Biểu diễn lược đồ CSDL bằng đồ thị để phát hiện chu trình

Với a Y A thì qua quan hệ Q_1 ta có thể xác định được một giá trị b Y B. Vẫn với giá trị a Y A đó nhưng xác định qua con đường Q_4 (để có c Y C) rồi qua Q_2 để có b'. b' và b có nhất thiết phải giống nhau hay không? Một CSDL là nhất quán nếu qua các cách

khác nhau chỉ xác định được 1 giá trị duy nhất của thuộc tính liên quan. Trong trường hợp này có 3 vấn đề nảy sinh:

Hai cách xác định (hoặc 2 con đường) mang ý nghĩa hoàn toàn giống nhau.

Có một con đường phụ thuộc vào con đường kia, nghĩa là tập hợp một số thuộc tính thông qua một con đường là tập con của các thuộc tính đó thông qua con đường kia.

Hai con đường độc lập nhau thì chu trình ở đây là chu trình giả, và như vậy không có ràng buộc toàn vẹn do sự hiện diện của chu trình.

Ví dụ 4.2.12:

Giả sử các quan hệ Q1, Q2, Q3, và Q4 được định nghĩa trên các thuộc tính như sau:

Q1 (Mã-nhân-viên, Mã-phòng)

Q2 (Mã-phòng, Mã-đề-án)

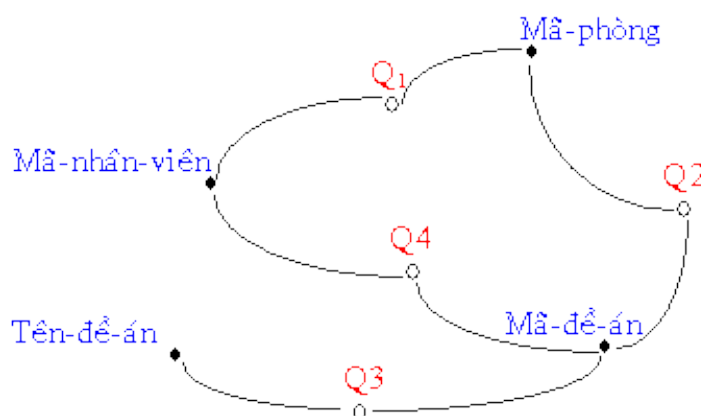
Q3 (Mã-đề-án, Tên-đề-án)

Q4 (Mã-đề-án, Mã-nhân-viên)

Đồ thị biểu diễn CSDL này như trong hình 4.2.2:

Giả thiết 1: Với tên từ sau:

Mỗi nhân viên (thể hiện qua Mã-nhân-viên) được phân công vào tất cả các đề án do phòng đó (thể hiện qua Mã-phòng) phụ trách.



Hình 4.2.2. Lược đồ CSDL được biểu diễn dưới dạng đồ thị

(i) Hai con đường mang ý nghĩa hoàn toàn giống nhau. Đường dài hơn Q1 kết nối với Q2 (ký hiệu là $Q1 \succ I Q2$) trùng với con đường ngắn hơn Q4 khi cùng xác định một đề án (*thể hiện qua Mã-đề-án*) mà một nhân viên (*thể hiện qua Mã-nhân-viên*) tham gia vào. Đây là một chu trình giả, có thể hủy bỏ (bằng cách loại bỏ quan hệ Q4).

(ii) Nếu vẫn muốn giữ lại quan hệ Q4, tức là không hủy bỏ chu trình, thì phải có một RBTV với thuật toán sau:

$$Q1 \succ I Q2 [\text{Mã-nhân-viên}, \text{Mã-đề-án}] = Q4 [\text{Mã-nhân-viên}, \text{Mã-đề-án}]$$

Giả thiết 2: Với tân từ sau:

Mỗi nhân viên (*thể hiện qua Mã-nhân-viên*) được phân công vào một số đề án do phòng đó (*thể hiện qua Mã-phòng*) phụ trách.

Con đường ngắn Q4 phụ thuộc vào con đường dài $Q1 \succ I Q2$, bởi vì một nhân viên có thể không tham gia vào tất cả các đề án do phòng mình phụ trách. Chu trình trên mang ý nghĩa thực sự. Ràng buộc toàn vẹn được thể hiện bởi thuật toán sau:

$$Q4 [\text{Mã-nhân-viên}, \text{Mã-đề-án}] \preceq I Q1 \succ I Q2 [\text{Mã-nhân-viên}, \text{Mã-đề-án}]$$

Giả thiết 3: Với tân từ sau:

Mỗi nhân viên (*thể hiện qua Mã-nhân-viên*) được phân công vào những đề án (*thể hiện qua Mã-phòng*) bất kỳ.

Hai con đường hoàn toàn độc lập nhau, không liên quan gì tới nhau, mang ý nghĩa hoàn toàn khác nhau, do đó không có RBTV.

Biểu diễn RBTV bằng phụ thuộc hàm.

Trong Chương III, mục 3.1, điểm 3.1.7, bài 4 đã trình bày khái niệm cơ bản về phụ thuộc hàm (*Functional Dependency*) trong một quan hệ. Phụ thuộc hàm có tầm quan trọng rất lớn trong việc phân tích và thiết kế mô hình dữ liệu. Mục này sẽ trình bày kỹ hơn về vấn đề này.

Khái niệm: Quan hệ R được định nghĩa trên tập thuộc tính $U = \{ A_1, A_2, \dots, A_n \}$. A, B là 2 tập con của tập thuộc tính U. Nếu tồn tại một ánh xạ $f: A \rightarrow B$ thì ta nói rằng A xác định hàm B, hay B phụ thuộc hàm vào A, và ký hiệu là $A \twoheadrightarrow B$.

Định nghĩa hình thức của phụ thuộc hàm như sau:

Quan hệ Q (A, B, C) có phụ thuộc hàm A xác định B (ký hiệu là $A \twoheadrightarrow B$) nếu:

" $q, q' \in Q$, sao cho $q.A = q'.A$ thì $q.B = q'.B$

(Nghĩa là: ứng với 1 giá trị của A thì có một giá trị duy nhất của B)

A là vế trái của phụ thuộc hàm, B là vế phải của phụ thuộc hàm.

$A \twoheadrightarrow B$ được gọi là phụ thuộc hàm hiển nhiên nếu $B \subseteq A$. Nghĩa là, một tập A lớn hơn và bao tập con B thì A xác định được giá trị của các thuộc tính trong tập B là điều hiển nhiên.

$A \twoheadrightarrow B$ được gọi là phụ thuộc hàm nguyên tố, hoặc nói cách khác, B được gọi là phụ thuộc hàm đầy đủ (*fully functional dependence*) vào A nếu "A' \subset A đều không có $A' \twoheadrightarrow B$.

Ví dụ 4.3.1 :

Trong lược đồ CSDL quản lý hóa đơn bán hàng đã cho trong ví dụ 4.1.2 và 4.2.10, quan hệ HÓAĐƠN (Số-hóa-đơn, Số_chủng_loại_mặt_hàng, Tổng-trị-giá) có các phụ thuộc hàm sau:

$f1$: Số-hóa-đơn \twoheadrightarrow Số_chủng_loại_mặt_hàng;

$f2$: Số-hóa-đơn \twoheadrightarrow Tổng-trị-giá;

bởi vì Số-hóa-đơn là khóa của lược đồ quan hệ HÓAĐƠN. Nếu biết số hóa đơn thì ta có thể xác định được tất cả các thông tin còn lại liên quan đến hóa đơn đó, trong đó có thông tin về Số_chủng_loại_mặt_hàng và Tổng-trị-giá tất cả các mặt hàng của hóa đơn. Các phụ thuộc hàm trên đều là nguyên tố.

Quan hệ CHITIẾT_HĐ (Số-hóa-đơn, Mã-hàng, Số-lượng-đặt, Đơn-giá, Trị-giá) có các phụ thuộc hàm sau:

$f1$: Số-hóa-đơn, Mã-hàng \twoheadrightarrow Số-lượng-đặt.

$f2$: Số-hóa-đơn, Mã-hàng \twoheadrightarrow Đơn-giá.

$f3$: Số-hóa-đơn, Mã-hàng \twoheadrightarrow Trị-giá.

$f4$: Số-lượng-đặt, Đơn-giá \twoheadrightarrow Trị-giá.

Thuộc tính Đơn-giá phụ thuộc hàm không đầy đủ vào khóa (Số-hóa-đơn, Mã-hàng), bởi vì nó chỉ phụ thuộc vào mặt hàng (thông qua Mã-hàng).

Qua ví dụ vừa nêu chúng ta thấy, trên một lược đồ quan hệ có thể tồn tại nhiều phụ thuộc hàm. Tập các phụ thuộc hàm thường được ký hiệu bằng chữ F .

Gọi F là tập các phụ thuộc hàm đối với lược đồ quan hệ R định nghĩa trên tập thuộc tính U và $X \bowtie Y$ là một phụ thuộc hàm; $X, Y \subseteq U$. Ta nói rằng $X \bowtie Y$ được *suy diễn logic* từ F nếu R thỏa các phụ thuộc hàm của F thì cũng thỏa $X \bowtie Y$ và ký hiệu là:

$$F \models X \bowtie Y.$$

Ví dụ 4.3.2:

Với $F = \{ X \bowtie Y, X \bowtie Z, Y \bowtie T \}$

Thì ta có các phụ thuộc hàm $X \bowtie YZ$ và $X \bowtie T$.

Gọi F^+ là bao đóng (*Closure*) của F , tức là tập các phụ thuộc hàm được *suy diễn logic* từ F . Nếu $F = F^+$ thì ta nói F là họ đầy đủ (*full family*) của các phụ thuộc hàm.

Bài toán thành viên (*MemberShip*) nêu vấn đề phụ thuộc hàm $X \bowtie Y$ có phải là được suy diễn logic từ F hay không (tức là $X \bowtie Y \in F^+ ?$) là một bài toán khó giải. Nó đòi hỏi chúng ta phải có một hệ luật dẫn để suy diễn logic các phụ thuộc hàm.

Năm 1974, Amstrong đã đưa ra hệ tiên đề (còn gọi là hệ luật dẫn Amstrong, hay các tính chất của phụ thuộc hàm) (D.Maier - 1983 [2]) như sau:

$X, Y, Z, W \subseteq U$. Phụ thuộc hàm có các tính chất sau đây:

(i) *Tính phản xạ:*

Nếu $Y \subseteq X$ thì $X \bowtie Y$.

$X \bowtie Y$ thì $XZ \bowtie YZ$.

(iii) *Tính bắc cầu:*

Nếu $X \bowtie Y$ và $Y \bowtie Z$ thì $X \bowtie Z$.

Và người ta đã chứng minh rằng hệ tiên đề Amstrong là đúng đắn và đầy đủ thông qua 3 bổ đề (ở đây không chứng minh):

Bổ đề 1: Hệ tiên đề Amstrong là đúng, nghĩa là, với F là tập phụ thuộc hàm đúng trên quan hệ R , nếu $X \twoheadrightarrow Y$ là một phụ thuộc hàm

Bổ đề 2: Từ hệ tiên đề Amstrong suy ra một số luật bổ sung sau đây:

(iv) Tính phân rã (hoặc luật tách):

Nếu $X \twoheadrightarrow YZ$ thì $X \twoheadrightarrow Y$ và $X \twoheadrightarrow Z$.

(v) Tính hợp (hoặc luật hợp):

Nếu $X \twoheadrightarrow Y$ và $X \twoheadrightarrow Z$ thì $X \twoheadrightarrow YZ$.

(vi) Tính tựa bắc cầu, hoặc bắc cầu giả:

Nếu $X \twoheadrightarrow Y$ và $YZ \twoheadrightarrow W$ thì $XZ \twoheadrightarrow W$.

Định nghĩa: Bao đóng (Closure) của tập các thuộc tính X đối với tập các phụ thuộc hàm F (ký hiệu là X_F^+) là tập tất cả các thuộc tính A có thể suy dẫn từ X nhờ tập bao đóng của các phụ thuộc hàm F^+ :

$$X_F^+ = \{ A \mid X \twoheadrightarrow A \text{ Y } F^+ \}$$

Và

Bổ đề 3: $X \twoheadrightarrow Y$ được suy diễn logic từ F nhờ hệ tiên đề Amstrong khi và chỉ khi $Y \text{ Y } X_F^+$.

Định lý: Hệ tiên đề Amstrong là đúng và đầy đủ.

Đây là nền tảng để chứng minh các lý thuyết CSDL quan hệ, đặc biệt là trong “bài toán thành viên” - kiểm tra xem một phụ thuộc hàm $f : X \twoheadrightarrow Y$ có thuộc bao đóng của tập F hay không? ...

Ví dụ 4.3.3:

Cho lược đồ quan hệ $R(A, B, C, D, E, G, H)$ và tập các phụ thuộc hàm $F = \{AB \twoheadrightarrow C, B \twoheadrightarrow D, CD \twoheadrightarrow E, CE \twoheadrightarrow GH, G \twoheadrightarrow A\}$. Áp dụng hệ tiên đề Amstrong, tìm một chuỗi suy diễn $AB \twoheadrightarrow E$.

Giải:

1. $AB \twoheadrightarrow C$ (cho trước - phụ thuộc hàm f_1)

2. $AB \circ AB$ (tính chất phản xạ)
3. $AB \circ B$ (luật tách)
4. $B \circ D$ (cho trước - phụ thuộc hàm f_2)
5. $AB \circ D$ (bắc cầu 3 & 4)
6. $AB \circ CD$ (hợp 1 & 5)
7. $CD \circ E$ (cho trước - phụ thuộc hàm f_3)
8. $AB \circ E$ (bắc cầu 6 & 7). Kết thúc.

Ví dụ 4.3.4:

Cho lược đồ quan hệ $R(A, B, C, D, E, G, H, I, J)$ và tập các phụ thuộc hàm $F = \{AB \circ E, AG \circ J, BE \circ I, E \circ G, GI \circ H\}$. Tìm chuỗi suy diễn $AB \circ GH$ (Bài tập mẫu của David Maier tr. 51)

Giải:

1. $AB \circ E$ (cho trước - phụ thuộc hàm f_1)
2. $AB \circ AB$ (phản xạ)
3. $AB \circ B$ (luật tách)
4. $AB \circ BE$ (hợp của 1 & 3)
5. $BE \circ I$ (cho trước - phụ thuộc hàm f_3)
6. $AB \circ I$ (bắc cầu 4 & 5)
7. $E \circ G$ (cho trước - phụ thuộc hàm f_4)
8. $AB \circ G$ (bắc cầu 1 & 7)
9. $AB \circ GI$ (hợp 6 & 8)
10. $GI \circ H$ (cho trước - phụ thuộc hàm f_5)
11. $AB \circ H$ (bắc cầu 9 & 10)

12. AB®GH (hợp 8 & 11)

Thuật toán tìm bao đóng của X dựa trên tập phụ thuộc hàm F đối với quan hệ R được mô tả bằng ngôn ngữ tựa Pascal như sau (Xem D.Maier - 1983 tr.64-69 [4]):

Procedure Closure (X, F)

Begin

OldDep := \emptyset NewDep := X;

While NewDep \neq OldDep Do

Begin

OldDep = NewDep;

For every FD: W®Z \forall F Do

If W \subseteq NewDep Then NewDep := NewDep \cup Z;

End;

Return NewDep;

End;

Ứng dụng để giải bài toán tìm khóa của quan hệ:

Định nghĩa 3.1.6.2 trong Chương III, mục 3.1 về khóa được viết lại bằng phụ thuộc hàm như sau:

R là lược đồ quan hệ định nghĩa trên tập các thuộc tính $U = \{ A_1, A_2, \dots, A_n \}$, với tập các phụ thuộc hàm $F = \{ f_1, f_2, \dots, f_m \}$ xác định trên R. K \subseteq U là khóa của R nếu thỏa mãn hai điều kiện sau đây:

(i) K ® U.

(ii) \nexists K' \subseteq K mà K' ® U.

Bây giờ chúng ta hãy biểu diễn lược đồ quan hệ R (U) bằng đồ thị có hướng như sau:

-Mỗi nút của đồ thị là tên một thuộc tính của R.

-Cung nối 2 thuộc tính A và B thể hiện phụ thuộc hàm $A \twoheadrightarrow B$

-Thuộc tính mà chỉ có các mũi tên đi ra (tức là chỉ nằm trong vế trái của các phụ thuộc hàm) được gọi là nút gốc.

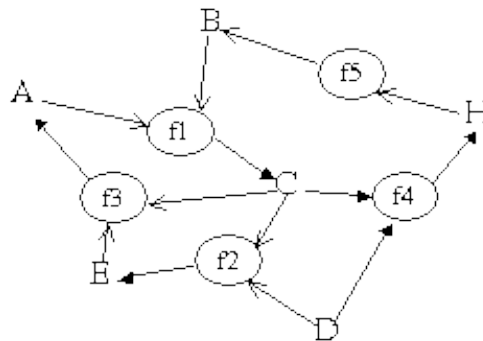
-Thuộc tính mà tới nó chỉ có các cung đi tới (tức là chỉ nằm trong vế phải của các phụ thuộc hàm) được gọi là nút lá.

Như vậy khóa của lược đồ quan hệ phải bao phủ tập các nút gốc, đồng thời không chứa bất kỳ nút lá nào của đồ thị.

Xuất phát từ tập các nút gốc (X), dựa trên tập các phụ thuộc hàm F, chúng ta tìm bao đóng X_F^+ . Nếu $X_F^+ = U$ thì X là khóa, ngược lại thì bổ sung một thuộc tính không thuộc nút lá vào X rồi tìm bao đóng. Cứ như thế cho đến khi tìm được bao đóng của X bằng U.

Ví dụ 4.3.5 :

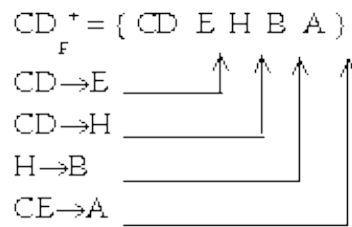
Cho R (A, B, C, D, E, H) với $F = \{ AB \twoheadrightarrow C, CD \twoheadrightarrow E, EC \twoheadrightarrow A, CD \twoheadrightarrow H, H \twoheadrightarrow B \}$. Hãy tìm khóa của R.



Trong đồ thị trên chúng ta thấy chỉ có nút D là nút gốc, các nút còn lại đều không phải nút lá. Khóa của R phải chứa thuộc tính D.

$D_F^+ = I$, bởi vì không tìm thấy phụ thuộc hàm nào có vế trái chỉ có một mình D.

Vì CD có mặt trong vế trái của 2 phụ thuộc hàm do đó ghép thêm C vào tập các nút gốc để xét khóa.



Như vậy $CD \twoheadrightarrow \{C, D, E, H, B, A\}$ do đó CD là khóa của R .

Ví dụ 4.3.6:

Cho quan hệ GIẢNG-DẠY ($MS_CBGD, MS_MH, T_CBGD, HH_CBGD, ML, TSSV$) với tập phụ thuộc hàm:

$F = \{MS_CBGD \twoheadrightarrow T_CBGD;$

$MS_MH \twoheadrightarrow HH_CBGD, ML;$

$HH_CBGD \twoheadrightarrow ML;$

$MS_CBGD \twoheadrightarrow HH_CBGD;$

$MS_CBGD, MS_MH \twoheadrightarrow TSSV$

$\}$

Ở đây MS_CBGD là mã số cán bộ giảng dạy; MS_MH là mã số môn học; T_CBGD là tên cán bộ giảng dạy; HH_CBGD là học hàm của cán bộ giảng dạy; ML là mã số lớp học; và $TSSV$ là tổng số sinh viên theo học môn MS_MH do giảng viên MS_CBGD phụ trách.

Bài toán: Xác định khóa của quan hệ GIẢNG-DẠY.

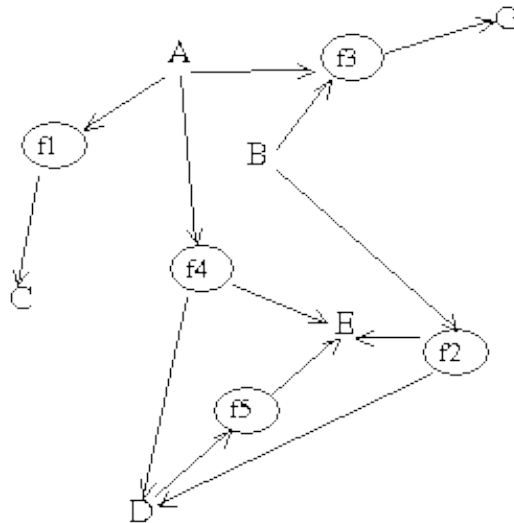
(TS. Đồng thị Bích Thủy, đề thi năm 1992).

Lời giải: Để cho đơn giản, chúng ta hãy ký hiệu tên các thuộc tính của quan hệ trên lần lượt là A, B, C, D, E, G tương ứng. Khi đó quan hệ GIẢNG-DẠY và tập phụ thuộc hàm F được viết ngắn gọn lại là:

$R(A, B, C, D, E, G)$

$F = \{ A \twoheadrightarrow C; B \twoheadrightarrow D, E; D \twoheadrightarrow E; A \twoheadrightarrow ED; AB \twoheadrightarrow G \}$

Đồ thị biểu diễn các phụ thuộc hàm như sau:



Chúng ta nhận thấy trên đồ thị, hai thuộc tính A và B là các nút gốc. E, C và G là các nút lá. Khóa của quan hệ phải chứa các thuộc tính ở các nút gốc.

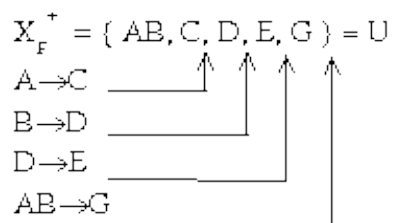
Xét $X = \{ A \}$

$X_F^+ = \{ A, C, D, E \}$ // Còn thiếu thuộc tính B và G

Xét $X = \{ B \}$

$X_F^+ = \{ D, E \}$ // Còn thiếu thuộc tính A, B, C và G

Lấy $X = \{ A, B \}$



Vậy AB là khóa của R. Tức là, khóa của quan hệ giảng-dạy là $\{ \underline{MS_CBGD}, \underline{MS_MH} \}$

Phụ thuộc hàm cùng với các loại phụ thuộc đa trị (*Multivalued Dependency*) và phụ thuộc kết (*Join Dependency*) là những khái niệm rất quan trọng trong lý thuyết, phân tích và thiết kế CSDL.

Ngôn ngữ đại số quan hệ

Dẫn nhập.

Ngôn ngữ đại số quan hệ là ngôn ngữ biểu diễn câu hỏi về các quan hệ.

Do các cách biểu diễn khác nhau nên trong tài liệu này, ngôn ngữ đại số quan hệ được chia làm 3 nhóm: Các phép toán tập hợp - các phép toán này được biểu diễn như các phép toán trên tập hợp, các phép toán quan hệ và các phép toán khác liên quan chủ yếu đến phép kết nối.

Các phép toán tập hợp trên các quan hệ.

Sáu phép toán cơ bản trên tập hợp được áp dụng trên tập các bộ giá trị của các quan hệ, đó là: Hợp (*Union*), Hiệu (*Minus*), Giao (*Intersection*), Tích Đề-các (*Cartesian*), phép chia (*Division*) và phép bù (*Complement*).

Giả thiết: $U = \{ A_1, A_2, A_3, \dots, A_n \}$ là tập các thuộc tính.

R và S là 2 quan hệ được định nghĩa trên U có cùng thứ tự của các thuộc tính. Và ở đây chúng ta luôn luôn giả thiết là R và S có số lượng hữu hạn các bộ giá trị.

Phép hợp 2 quan hệ (Union).

Hợp của hai quan hệ R và S - được ký hiệu là $R \cup S$ - là một quan hệ Q xác định trên tập thuộc tính U , có cùng thứ tự thuộc tính như trong quan hệ R và S , được định nghĩa như sau:

$$Q = R \cup S = \{ t \mid t \in R \text{ hoặc } t \in S \}$$

Nói một cách nôm na, hợp của 2 quan hệ R và S là một quan hệ có cùng ngôi với quan hệ R và S với các bộ giá trị bằng gộp các bộ giá trị của cả R và S ; những bộ giá trị trùng nhau chỉ được giữ lại 1 bộ.

Ví dụ 5.2.1:

Quan hệ Đơn vị A có các bộ giá trị sau:

MãSố	Họ-tên	Phái	Chức-danh	Lương	MãĐV
100	Nguyễn Văn Nam	Nam	Giám đốc	2.500.000	10

101	Hoàng Thị Xuân	Nữ	Kế toán trưởng	1.700.000	10
103	Đặng Ngọc Chiến	Nữ	Thư ký	1.000.000	10
105	Phan Kỳ Nhân	Nam	Lái xe	700.000	10

Quan hệ Đơn vị B có các bộ giá trị sau:

Mã Số	Họ-tên	Phái	Chức-danh	Lương	Mã ĐV
210	Nguyễn Thị Cao	Nữ	Trưởng phòng	1.200.000	30
101	Hoàng Thị Xuân	Nữ	Kế toán trưởng	1.700.000	10
221	Đỗ Hữu Ngọc	Nam	Phó phòng	1.000.000	30
233	Hoàng Thao	Nam	Chuyên viên	1.000.000	30

Hợp của hai quan hệ trên cho kết quả là quan hệ NV-CTy có 7 bộ giá trị sau:

Mã Số	Họ-tên	Phái	Chức-danh	Lương	Mã ĐV
100	Nguyễn Văn Nam	Nam	Giám đốc	2.500.000	10
101	Hoàng Thị Xuân	Nữ	Kế toán trưởng	1.700.000	10
103	Đặng Ngọc Chiến	Nữ	Thư ký	1.000.000	10
105	Phan Kỳ Nhân	Nam	Lái xe	700.000	10
210	Nguyễn Thị Cao	Nữ	Trưởng phòng	1.200.000	30
221	Đỗ Hữu Ngọc	Nam	Phó phòng	1.000.000	30
233	Hoàng Thao	Nam	Chuyên viên	1.000.000	30

Bộ giá trị có mã số nhân viên là 101 xuất hiện 2 lần trong 2 quan hệ Đơn vị A và Đơn vị B, nhưng trong quan hệ NV-Cty bộ này chỉ được giữ lại 1 lần, do đó chỉ còn 7 bộ giá trị.

Phép trừ hai quan hệ (Minus).

Hiệu của hai quan hệ R và S, được ký hiệu là $R - S$, là một quan hệ Q xác định trên tập thuộc tính U, có cùng thứ tự thuộc tính như trong quan hệ R và S, được định nghĩa như sau:

$$Q = R - S = \{ t / t \in R \text{ và } t \notin S \}$$

Nói một cách nôm na, hiệu của 2 quan hệ R và S là một quan hệ có cùng ngôi với quan hệ R và S với các bộ giá trị là các bộ giá trị của R sau khi đã loại bỏ đi các bộ có mặt trong quan hệ S.

Ví dụ 5.2.2:

Với hai quan hệ như trên, hiệu của Đơn vị A và Đơn vị B là quan hệ NV-Cty A với các bộ sau:

MãSố	Họ-tên	Phái	Chức-danh	Lương	MãĐV
100	Nguyễn Văn Nam	Nam	Giám đốc	2.500.000	10
103	Đặng Ngọc Chiến	Nữ	Thư ký	1.000.000	10
105	Phan Kỳ Nhân	Nam	Lái xe	700.000	10

Giao của 2 quan hệ (Intersection).

Giao của hai quan hệ R và S, được ký hiệu là $R \cap S$, là một quan hệ Q xác định trên tập thuộc tính U, có cùng thứ tự thuộc tính như trong quan hệ R và S, được định nghĩa như sau:

$$Q = R \cap S = \{ t / t \in R \text{ và } t \in S \}$$

Nói một cách nôm na, giao của 2 quan hệ R và S là một quan hệ có cùng ngôi với quan hệ R và S với các bộ giá trị là các bộ giống nhau của cả hai quan hệ R và S.

Ví dụ 5.2.3:

Với hai quan hệ như trên, hiệu của Đơn vị A và Đơn vị B là quan hệ NV-Cty A với các bộ sau:

MãSố	Họ-tên	Phái	Chức-danh	Lương	MãĐV
101	Hoàng Thị Xuân	Nữ	Kế toán trưởng	1700.000	10

Tích Đề-các của 2 quan hệ (Cartesian).

$R(A_1, A_2, \dots, A_n)$ và $S(B_1, B_2, \dots, B_m)$ là hai quan hệ có số bộ giá trị hữu hạn. Tích Đề-các của hai quan hệ R và S, được ký hiệu là $R \times S$, là một quan hệ Q xác định trên tập thuộc tính của R và S (với $n + m$ thuộc tính) và được định nghĩa như sau:

$Q = R \times S = \{ t / t \text{ có dạng } (a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_m) \text{ trong đó } (a_1, a_2, \dots, a_n) \in R \text{ và } (b_1, b_2, \dots, b_m) \in S \}$

Nói một cách nôm na, tích Đề-các của 2 quan hệ R và S là một quan hệ Q có số ngôi bằng tổng số ngôi của R và S, với các bộ giá trị gồm 2 phần: phần bên trái là một bộ giá trị của R và phần bên phải là một bộ giá trị của S. Như vậy, nếu R có n_1 bộ giá trị và S có n_2 bộ giá trị, thì Q sẽ có $n_1 \times n_2$ bộ giá trị.

Ví dụ 5.2.4:

$R(A, B, C) \times S(D, E, F) = Q(A, B, C, D, E, F)$

$a_1, b_1, c_1 \quad a_1, b_1, c_1, d_1, e_1, f_1$

$a_2, b_2, c_2 \quad a_1, b_1, c_1, d_2, e_2, f_2$

$a_3, b_3, c_3 \quad a_2, b_2, c_2, d_1, e_1, f_1$

và $S(D, E, F) \quad a_2, b_2, c_2, d_2, e_2, f_2$

$d_1, e_1, f_1 \quad a_3, b_3, c_3, d_1, e_1, f_1$

$d_2, e_2, f_2 \quad a_3, b_3, c_3, d_2, e_2, f_2$

Phép chia hai quan hệ (Division).

R là quan hệ n ngôi và S là quan hệ m ngôi ($n > m$ và $S \subseteq I^n$), có m thuộc tính chung (giống nhau về mặt ngữ nghĩa, hoặc các thuộc tính có thể so sánh được) giữa R và S. Phép chia 2 quan hệ R và S, ký hiệu là $R \div S$, là một quan hệ Q có $n - m$ ngôi được định nghĩa như sau:

$$Q = R \div S = \{ t / \text{sc: } t \in S, (t, u) \in R \}$$

Sử dụng định nghĩa phép tích Đề-các, có thể định nghĩa phép chia hình thức hơn như sau:

$R \div S = Q$ sao cho $Q \times S \subseteq R$ (với giả thiết thêm là thứ tự thuộc tính của R, S, Q là không quan trọng).

Ví dụ 5.2.5:

$R(A, B, C, D) \div S(C, D) = Q(A, B)$

a b c d c d a b

a b e f e f c d

b c e f

c d c d

c d e f

a b d e

Ví dụ 5.2.6: (TS. Đồng Thị Bích Thủy)

Cho quan hệ về khả năng lái các loại máy bay của các phi công:

KHẢ-NĂNG (Số-hiệu-phi-công, Số-hiệu-máy-bay)

Số-hiệu-phi-công	Số-hiệu-máy-bay
32	102
30	101
30	103
32	103
33	100
30	102
31	102
30	100
31	100

Câu hỏi: Cho biết các phi công có khả năng lái được cả 3 loại máy bay 100, 101, và 103 ?

Trả lời: Đó là thương của phép chia quan hệ **KHẢ-NĂNG** cho quan hệ **MÁY-BAY** (Số-hiệu-máy-bay):

100
101

Và kết quả là quan hệ PHI-CÔNG (Số-hiệu-phi-công) có 1 bộ giá trị (30).

Phép bù của một quan hệ (Complement).

Cho quan hệ $R (A_1, A_2, \dots, A_n)$ với các miền giá trị của thuộc tính A_i là $MGT(A_i)$. Phép bù của quan hệ R là quan hệ Q xác định trên tập thuộc tính R^+ , ký hiệu là \bar{R} , được định nghĩa như sau:

$$\bar{R} = R = \{ t (a_1, a_2, \dots, a_n) \text{ và } a_i \notin MGT(A_i) \mid i=1..n \}$$

Nghĩa là tập tất cả các bộ giá trị có thể có của tích Đề-các miền giá trị $MGT(A_i)$ nhưng chưa có mặt trong thể hiện của quan hệ R . Quan hệ bù của một quan hệ có số lượng bộ giá trị là rất lớn, vì vậy trong thực tế rất ít hệ quản trị CSDL cài đặt phép toán này.

Ví dụ 5.2.7:

Quan hệ CUNG-CẤP (Mã-NCC, Hàng-hóa) với $Mã-NCC = \{ S_1, S_2, S_3 \}$ và các Hàng-hóa cung cấp là $\{ \text{Đinh, Ốc, Vít} \}$

Mã-NCC	Hàng-hóa
S1	Đinh
S1	Vít
S2	Ốc
S2	Đinh
S3	Vít

Quan hệ bù của quan hệ CUNG-CẤP có các bộ giá trị sau:

Mã-NCC	Hàng-hóa
S1	Ốc
S2	Vít
S3	Ốc
S3	Đinh

Ngôn ngữ đại số quan hệ (tiếp theo)

Phép kết nối hai quan hệ (Join)

Giả sử có 2 quan hệ $R (A_1, A_2, \dots, A_n)$ và $S (B_1, B_2, \dots, B_m)$.

$t = (a_1, a_2, \dots, a_n)$ là một bộ giá trị của R và $u = (b_1, b_2, \dots, b_m)$ là một bộ giá trị của S .
Gọi v là bộ ghép nối u vào t (hay bộ giá trị t và u được "xếp cạnh nhau" để tạo thành bộ giá trị mới v) được định nghĩa như sau:

$$v = (t, u) = (a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_m).$$

$A \in R^+$ và $B \in S^+$ là hai thuộc tính có thể so sánh được.

Gọi q là một trong các phép toán so sánh $\{ <, <=, >, >=, =, \neq \}$.

Phép kết nối hai quan hệ (có thể nói tắt là *phép kết*) R với S trên các thuộc tính A và B với phép so sánh q , với giả thiết là giá trị cột $R[A]$ có thể so sánh được (qua phép so sánh q) với mỗi giá trị của cột $R[B]$, được định nghĩa qua:

R



$$S = \{ v = (t, u) \mid t \in R, u \in S \text{ và } t.A \ q \ u.B \}$$

Hoặc:

R



$$S = (R \times S) : (A \ q \ B).$$

Phép kết nối 2 quan hệ R và S có thể xem như được thực hiện qua 2 bước:

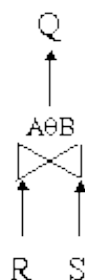
Bước 1: Thực hiện tích Đề-các hai quan hệ R và S .

Bước 2: Chọn các bộ giá trị thỏa mãn điều kiện $A \ q \ B$.

Ngữ nghĩa: Định nghĩa trên cho ta kết quả của phép kết nối hai quan hệ R và S với phép so sánh q trên 2 thuộc tính A và B là một quan hệ mới, Đó là kết quả cuối cùng của phép toán quan hệ (phép *Chọn*) trên quan hệ kết quả của phép toán tập hợp (tích *Đề-các*).

Nếu q là phép toán so sánh bằng nhau (=) thì ta gọi đó là phép *kết nối bằng* (*Equi Join*). Nếu các thuộc tính so sánh là giống tên nhau thì trong kết quả của phép kết nối sẽ loại bỏ đi một trong 2 thuộc tính đó, khi đó phép kết nối được gọi là *phép kết nối tự nhiên* (*Natural Join*) và sử dụng ký hiệu cho phép toán là " * " hoặc chỉ ký hiệu I><I (không có A q B) ở phía trên của phép toán. Trong các trường hợp còn lại, phép toán được gọi chung là *phép kết nối theta* (*q -Join*).


Phép kết nối được biểu diễn bằng sơ đồ như sau:



Hình 5.3.3. Sơ đồ biểu diễn phép kết nối

Ví dụ 5.3.3:

Cho 2 quan hệ R và S với các bộ giá trị cho trong bảng ở cột lớn thứ nhất và thứ hai bên trái. Kết quả *phép kết nối theta* (q -Join) của 2 quan hệ trên được cho trong bảng ở cột lớn thứ ba dưới đây:

R	(A	B	C)										
	a ₁	1	1										
	a ₂	2	1										
	a ₃	2	2										
S	(C	D	E)	R	$R.B \neq S.C$ 	S	=Q	(A	B	C	C	D	E)

	1	d ₁	e ₁			a ₁	1	1	1	d ₁	e ₁
	2	d ₂	e ₂			a ₂	2	1	1	d ₁	e ₁
	3	d ₃	e ₃			a ₂	2	1	2	d ₂	e ₂
						a ₃	2	2	1	d ₁	e ₁
						a ₃	2	2	2	d ₂	e ₂

Kết quả *phép kết nối tự nhiên* của 2 quan hệ R và S là quan hệ Q' với các bộ giá trị như sau:

R I >< I S	= Q	(A	B	C	D	E)
		a ₁	1	1	d ₁	e ₁
		a ₂	2	1	d ₁	e ₁
		a ₃	2	2	d ₂	e ₂

Ví dụ 5.3.4:

Cho CSDL về cán bộ - viên chức Nhà nước (CBVC) với các quan hệ sau đây:

-Quan hệ ĐƠN-VỊ:

Mã-ĐV	Tên-đơn-vị	Loại
10002	Trường Kỹ thuật nghiệp vụ máy tính	3
10003	Phòng quản lý hành chính	2
10022	Văn phòng đại diện Thanh niên	2
10070	Ban quản lý vốn sinh viên	5
10071	Lực lượng quản lý thị trường	5

-Quan hệ loại hình tổ chức của đơn vị LOAI-ĐV:

Loại	Tên-loại-hình
2	Hành chính
3	Sự nghiệp hoàn toàn
5	Hạch toán độc lập

-Quan hệ Ngạch-CBVC:

Ngạch	Tên ngạch
01002	Chuyên viên chính
01003	Chuyên viên
01004	Cán sự
01005	Kỹ thuật viên đánh máy
01008	Nhân viên văn thư
01010	Lái xe cơ quan

-Quan hệ Ngạch-Bậc-lương:

Ngạch	Bậc	Hệ-số-lương
01002	01	3.35
01002	02	3.63
01002	03	3.91
01003	05	2.82
01003	07	3.31
01003	08	3.56
01004	07	2.18
01004	08	2.30
01004	10	2.55
01004	11	2.68
01005	04	2.06
01005	05	2.18
01005	06	2.30
01008	11	2.12
01010	08	2.47
01010	11	2.80

01010	13	3.02
-------	----	------

-Quan hệ danh sách CBVC với các bộ giá trị sau:

Mã-ĐV	Mã-CC	Họ-lót	Tên	Giới	Ngày-sinh	Ngạch	Bậc	Ngày-xếp
10002	1000028	Trần Tứ	Hải	Nam	05/09/40	01003	08	01/12/96
10002	1000040	Trần Ngọc	Son	Nam	04/08/57	01003	05	01/12/97
10002	1000042	Nguyễn Văn	Sang	Nam	20/04/61	01004	10	01/01/97
10002	1000043	Nguyễn Văn	Thành	Nam	04/10/44	01004	10	01/01/97
10002	1000065	Huỳnh Thị	Hoa	Nữ	06/04/61	01004	07	01/01/97
10003	1000156	Huỳnh Ngọc	Thúy	Nữ	28/10/54	01005	06	01/09/97
10003	1000134	Nguyễn Văn	Bạc	Nam	08/09/42	01010	13	01/12/97
10003	1000159	Lê Văn	Sang	Nam	15/06/50	01008	11	01/03/96
10003	1000160	Trịnh Ngọc	Tâm	Nam	18/11/66	01010	08	01/02/96
10022	1000218	Nguyễn Cửu	Châu	Nam	19/11/47	01010	11	01/10/96
10022	1000219	Nguyễn Văn	Hùng	Nam	15/05/55	01003	05	01/12/95
10022	1000220	Nguyễn Kim	Lưu	Nữ	22/07/55	01004	12	01/12/95

(Các quan hệ 4 và 5 đã được nêu trong các ví dụ 4.2.3 và 4.2.5 trong Chương IV, mục 4.2, bài 5).

Phép kết nối tự nhiên 2 quan hệ ĐƠN-VI và LOẠI-ĐV là một quan hệ gồm 4 thuộc tính: Mã-ĐV, Tên-đơn-vị, Loại và Tên-loại với các bộ giá trị sau:

Mã-ĐV	Tên-đơn-vị	Loại	Tên-loại-hình
10002	Trường Kỹ thuật nghiệp vụ máy tính	3	Sự nghiệp hoàn toàn
10003	Phòng quản lý hành chánh	2	Hành chánh
10022	Văn phòng đại diện Thanh niên	2	Hành chánh
10070	Ban quản lý vốn sinh viên	5	Hạch toán độc lập
10071	Lực lượng quản lý thị trường	5	Hạch toán độc lập

*Ghi chú ***: Học viên cần ghi nhận lại các quan hệ trong ví dụ này. Chúng sẽ được dùng lại trong các chương tới.

Các phép toán khác.

Mục này trình bày 3 phép toán kết nối mở rộng khác đặc biệt quan trọng, mà bản chất của chúng vẫn là kết nối. Chúng đã được cài đặt trong một số hệ quan trị CSDL như MicroSoft Access, SQL-Server, Oracle. Các phép kết nối đó là: Kết nối nội (*Inner Join*), Kết nối trái (*Left Join*) và Kết nối phải (*Right Join*).

Phép kết nối nội (Inner Join).

Thực chất là phép *kết nối bằng* đã trình bày trên. Tuy nhiên, ngay cả trong trường hợp hai thuộc tính so sánh có cùng tên thì kết quả phép kết nối vẫn giữ lại 2 tên thuộc tính đó.

Ví dụ 5.4.1.1:

Cho 2 quan hệ R (A, B, C) và S (A, D, E, F) với các bộ giá trị như dưới đây. Kết quả của phép *kết nối nội* được cho trong bảng phía bên phải.

R	(A	B	C)	S	(A	D	E	F)	R	$R.A = S.A$	S	=Q	(A	B	C	A	D	E	F)
	a ₁	b ₁	c ₁		a ₁	d ₁	e ₁	f ₁					a ₁	b ₁	c ₁	a ₁	d ₁	e ₁	f ₁
	a ₂	b ₂	c ₂		a ₂	d ₂	e ₂	f ₂					a ₂	b ₂	c ₂	a ₂	d ₂	e ₂	f ₂
	a ₃	b ₃	c ₃		a ₄	d ₄	e ₄	f ₄					a ₇	b ₇	c ₇	a ₇	d ₇	e ₇	f ₇
	a ₅	b ₅	c ₅		a ₆	d ₆	e ₆	f ₆											
	a ₇	b ₇	c ₇		a ₇	d ₇	e ₇	f ₇											

Ví dụ 5.4.1.2 :

Phép kết nối nội 2 quan hệ ĐƠN-VỊ và LOẠI-ĐV cho kết quả là một bảng sau:

Mã-ĐV	Tên-đơn-vị	Loại	Loại	Tên-loại-hình
10002	Trường K.thuật nghiệp vụ máy tính	3	3	SN hoàn toàn
10003	Phòng quản lý hành chính	2	2	Hành chính

10022	Văn phòng đại diện Thanh niên	2	2	Hành chánh
10070	Ban quản lý vốn sinh viên	5	5	Hạch toán đ.lập
10071	Lực lượng quản lý thị trường	5	5	Hạch toán đ.lập

Phép kết nối trái (Left Join)

Giả sử có 2 quan hệ $R(A_1, A_2, \dots, A_n)$ và $S(B_1, B_2, \dots, B_m)$.

$t = (a_1, a_2, \dots, a_n)$ và $u = (b_1, b_2, \dots, b_m)$ là hai bộ giá trị của R và S . Gọi v là bộ ghép nối u vào t (hay bộ giá trị t và u được "xếp cạnh nhau") và ký hiệu là:

$$v = (t, u) = (a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_m).$$

Bộ $t_{\text{NULL}} = (\text{NULL}, \text{NULL}, \dots, \text{NULL})$ là một bộ đặc biệt của R gồm n giá trị của các thuộc tính A_1, A_2, \dots, A_n đều là không xác định và $u_{\text{NULL}} = (\text{NULL}, \text{NULL}, \dots, \text{NULL})$ là một bộ đặc biệt của S gồm m giá trị của các thuộc tính B_1, B_2, \dots, B_m đều là không xác định.

$A \preceq R^+$ và $B \preceq S^+$ là hai thuộc tính có thể so sánh được.

Phép *kết nối trái* hai quan hệ R với S trên các thuộc tính A và B với phép so sánh bằng ($=$), với giả thiết là giá trị cột $R[A]$ có thể so sánh tương đương được với mỗi giá trị của cột $S[B]$, được định nghĩa là:

R

$A=B$

$$S = \{ v = (t, u) \mid (t \preceq R, u \preceq S \text{ và } t.A = u.B) \text{ hoặc } (t \preceq R, u = u_{\text{NULL}} \text{ với } t.A \preceq S[B]) \}$$

nghĩa là, tất cả các bộ v có được nhờ cách đặt bộ giá trị của R và S xếp cạnh nhau, nếu có giá trị giống nhau trên 2 thuộc tính kết nối; và các bộ v có được nhờ cách đặt bộ của R với các bộ NULL của S , nếu không tìm được giá trị tương ứng của thuộc tính kết nối trên quan hệ S .

Ví dụ 5.4.2.1:

Với hai quan hệ R và S cùng các bộ giá trị của chúng đã được cho trong ví dụ 5.4.1, kết quả của phép kết nối trái của R và S là:

R

$$R.A = S.A$$

$$S = Q(A, B, C, A, D, E, F)$$

$$a_1 \ b_1 \ 1 \ a_1 \ d_1 \ e_1 \ f_1$$

$$a_2 \ b_2 \ c_2 \ a_2 \ d_2 \ e_2 \ f_2$$

$$a_3 \ b_3 \ c_3 \ - \ - \ - \ -$$

$$a_5 \ b_5 \ c_5 \ - \ - \ - \ -$$

$$a_7 \ b_7 \ c_7 \ a_7 \ d_7 \ e_7 \ f_7$$

Ký hiệu dấu trừ (-) trong các thuộc tính của S được hiểu là giá trị không xác định (*giá trị Null*).

Các dòng có giá trị thuộc tính A của R là a_3 và a_5 không tìm được giá trị của thuộc tính A tương ứng trong quan hệ S, nên phần còn lại của nó được để là không xác định. Qua bảng kết quả trình bày trên, chúng ta thấy ý nghĩa của phép toán này là nhằm xác định các bộ giá trị của quan hệ bên trái nhưng không có bộ giá trị tương ứng trong quan hệ phía bên phải.

Ví dụ 5.4.2.2 :

Giả sử chúng ta thêm một bộ mới cho bảng ĐƠN-VỊ gồm có Mã-ĐV là 10090, Tên-đơn-vị là Hội khuyến nông Quận X và loại hình đơn vị là 7. Khi đó phép kết nối trái (*Left Join*) hai quan hệ ĐƠN-VỊ và LOẠI-ĐV cho kết quả là quan hệ có thể hiện như sau:

Mã-ĐV	Tên-đơn-vị	Loại	Loại	Tên-loại-hình
10002	Trường K.thuật nghiệp vụ máy tính	3	3	SN hoàn toàn
10003	Phòng quản lý hành chánh	2	2	Hành chánh
10022	Văn phòng đại diện Thanh niên	2	2	Hành chánh
10070	Ban quản lý vốn sinh viên	5	5	Hạch toán đ.lập
10071	Lực lượng quản lý thị trường	5	5	Hạch toán đ.lập
10090	Hội khuyến nông Quận X	7	Null	Null

Bởi vì trong quan hệ LOẠI-ĐV không có bộ nào có giá trị ở cột *Loại* là 7, do đó ở dòng cuối cùng của quan hệ trên, một bộ gồm toàn giá trị NULL ở cả hai cột *Loại* và *Tên-loại-hình* được đặt cạnh bộ giá trị mới được bổ sung trong thể hiện của quan hệ ĐƠN-VỊ.

Ứng dụng của phép kết nối này có thể thấy rõ trong bài toán quản lý CBVC nêu trên: Trước hết cần xác định những đơn vị có loại hình tổ chức không có trong danh mục LOẠI-ĐV. Câu trả lời rất đơn giản: chỉ việc chọn ra các dòng của bảng kết nối có giá trị NULL ở cột *Loại* trong phần đuôi của bộ giá trị là NULL.

Cũng trong bài toán quản lý CBVC nêu trên: Theo yêu cầu quản lý, mỗi CBVC có tên trong danh sách của đơn vị phải có một bản khai lý lịch, tức là một bộ giá trị về lý lịch chi tiết. Mỗi CBVC có một mã số CBVC để phân biệt và nhận dạng đồng thời khóa của quan hệ lý lịch cũng là mã số này. Có thể ứng dụng phép kết nối trái để xác định những CBVC nào có tên trong danh sách mà chưa có lý lịch trong CSDL.

Phép kết nối phải (Right Join)

Vấn với các quan hệ R, S; các thuộc tính A, B; và các bộ giá trị $v, t, u, t_{NULL}, u_{NULL}$ được xác định như trên.

Phép kết nối phải hai quan hệ R với S trên các thuộc tính A và B với phép so sánh =, với giả thiết là giá trị cột R[A] có thể so sánh tương đương được với mỗi giá trị của cột S[B], được định nghĩa là:

R

$A=B$

$S = \{ v = (t, u) \mid (t \in R, u \in S \text{ và } t.A \neq u.B) \text{ hoặc } (t = t_{NULL}, u \in S, \text{ với } t.B \neq R[A]) \}$

nghĩa là, tất cả các bộ v có được nhờ cách đặt bộ giá trị của R và S xếp cạnh nhau nếu chúng có giá trị giống nhau trên 2 thuộc tính kết nối, và các bộ NULL của R với các bộ của S, nếu không tìm được giá trị tương ứng của thuộc tính kết nối trên quan hệ R.

Ví dụ 5.4.3.1:

Giả sử với các quan hệ R và S cùng các bộ giá trị của chúng được xác định như trong ví dụ 5.4.2 nêu trên. Kết quả của phép kết nối phải R với S là quan hệ với các bộ giá trị sau:

R



S = Q (A, B, C, A, D, E, F)

a₁ b₁ 1 a₁ d₁ e₁ f₁

a₂ b₂ c₂ a₂ d₂ e₂ f₂

- - - a₄ d₄ e₄ f₄

- - - a₆ d₆ e₆ f₆

a₇ b₇ c₇ a₇ d₇ e₇ f₇

Ký hiệu dấu trừ (-) trong các thuộc tính của R được hiểu là giá trị không xác định (*giá trị Null*).

Các dòng có giá trị tại thuộc tính A của S là a₄ và a₆ không tìm được giá trị của thuộc tính A tương ứng trong quan hệ R, do đó phần đầu của nó được để là không xác định. Qua bảng kết quả trình bày trên, chúng ta thấy ý nghĩa của phép toán này là nhằm xác định các bộ giá trị của quan hệ bên phải không có bộ giá trị tương ứng trong quan hệ phía bên trái.

Ví dụ 5.4.3.2 :

Giả sử chúng ta thêm một bộ mới cho quan hệ LOẠI_ĐV gồm có mã *Loại* là 6, *Tên-loại-hình* là Cơ quan Đảng / Đoàn. Khi đó phép kết nối phải (*Right Join*) hai quan hệ ĐƠN-VỊ và LOẠI_ĐV cho kết quả là quan hệ có thể hiện như sau:

Mã-ĐV	Tên-đơn-vị	Loại	Loại	Tên-loại-hình
10002	Trường K.thuật nghiệp vụ máy tính	3	3	SN hoàn toàn
10003	Phòng quản lý hành chính	2	2	Hành chính
10022	Văn phòng đại diện Thanh niên	2	2	Hành chính
10070	Ban quản lý vốn sinh viên	5	5	Hạch toán đ.lập
10071	Lực lượng quản lý thị trường	5	5	Hạch toán đ.lập
10090	Hội khuyến nông Quận X	7	Null	Null
Null	Null	Null	6	Cơ quan Đảng

Vì không có đơn vị nào có loại hình tổ chức đơn vị là 6, nên ở dòng cuối cùng trong bảng trên, cả 3 cột thuộc phần của quan hệ ĐƠN-VỊ đã bị để trống bằng các giá trị Null không xác định.

Một ứng dụng của phép kết nối này là: xác định những CBVC nào có lý lịch trong CSDL (do nhập sai mã số CBVC) nhưng không có tên trong danh sách (*tức là các lý lịch vô chủ*). Hoặc: xác định các loại hình đơn vị (*Loại*) mà không có đơn vị nào thuộc vào.

Trong một số ngôn ngữ truy vấn CSDL (MicroSoft SQL-Server, Oracle ...) người ta gọi 2 phép toán kết nối trái (*Left Join*) và kết nối phải (*Right Join*) bằng chung một từ là "phép kết nối ngoài" (*Outer Join*). Mục đích của phép kết nối này là cho phép giữ lại tất cả các bản ghi (hoặc bộ giá trị) của 2 quan hệ có cùng giá trị của các thuộc tính kết nối và những bộ giá trị của cả 2 quan hệ không tìm được bộ giá trị giống nhau trên các thuộc tính kết nối thuộc quan hệ đối ứng.

Ngôn ngữ truy vấn cơ sở dữ liệu SQL

Ngôn ngữ SQL

Như trong Chương I mục 1.4, bài 1 đã trình bày, một hệ quản trị CSDL phải có ngôn ngữ giao tiếp giữa người sử dụng với CSDL (*hoặc cũng còn gọi là ngôn ngữ truy nhập CSDL*). Ngôn ngữ giao tiếp CSDL gồm các phạm trù:

Ngôn ngữ mô tả dữ liệu (*Data Definition Language - DDL*) để cho phép khai báo cấu trúc các bảng của CSDL, khai báo các mối liên hệ của dữ liệu (*Data RelationShip*) và các quy tắc (*Rules, Constraint*) quản lý áp đặt lên các dữ liệu đó.

Ngôn ngữ thao tác dữ liệu (*Data Manipulation Language - DML*) cho phép người sử dụng có thể thêm (*Insert*), xóa (*Delete*), sửa (*Update*) dữ liệu trong CSDL.

Ngôn ngữ truy vấn dữ liệu, hay ngôn ngữ hỏi đáp có cấu trúc (*Structured Query Language - SQL*) cho phép những người khai thác CSDL (chuyên nghiệp hoặc không chuyên) sử dụng để truy vấn các thông tin cần thiết trong CSDL.

Ngôn ngữ quản lý dữ liệu (*Data Control Language - DCL*) cho phép những người quản trị hệ thống thay đổi cấu trúc của các bảng dữ liệu, khai báo bảo mật thông tin và cấp quyền hạn khai thác CSDL cho người sử dụng.

Những năm 1975-1976, IBM lần đầu tiên đưa ra hệ quản trị CSDL kiểu quan hệ mang tên SYSTEM-R với ngôn ngữ giao tiếp CSDL là SEQUEL (*Structured English QUery Language*), đó một ngôn ngữ con để thao tác với CSDL.

Năm 1976 ngôn ngữ SEQUEL được cải tiến thành SEQUEL2. Khoảng năm 1978-1979 SEQUEL2 được cải tiến và đổi tên thành Ngôn Ngữ Truy Vấn Có Cấu Trúc (*Structured Query Language - SQL*) và cuối năm 1979 hệ quản trị CSDL được cải tiến thành SYSTEM-R*.

Năm 1986 Viện Tiêu Chuẩn Quốc Gia Mỹ (*American National Standards Institute - ANSI*) đã công nhận và chuẩn hóa ngôn ngữ SQL, và sau đó Tổ chức Tiêu chuẩn Thế giới (*International Standards Organization - ISO*) cũng đã công nhận ngôn ngữ này. Đó là chuẩn SQL-86.

Tới nay SQL đã qua 3 lần chuẩn hóa lại (1989, 1992, 1996) để mở rộng các phép toán và tăng cường khả năng bảo mật và tính toàn vẹn dữ liệu. Tài liệu này trình bày Ngôn ngữ truy vấn CSDL dựa trên chuẩn SQL-92 và có tham khảo với SQL, SQL*PLUS, PL/

SQL của Oracle Server Release 7.3 (1996) và MicroSoft SQL Server 7.1 với các phạm trù nêu trên.

Để việc trình bày cú pháp các câu lệnh SQL được gọn gàng và dễ hiểu, tài liệu này có đưa ra một số quy ước ký pháp (*Typographic Conventions*) như sau:

Các từ khóa (*KeyWords*), các hàm (*Functions*), tên bảng (quan hệ - *Table Names*) của các câu lệnh được viết bằng chữ in hoa (*UpperCase*).

Các tên thuộc tính (*Column Names*) của các bảng được viết đậm. Những tên thuộc tính có dấu tiếng Việt hay có khoảng trắng được viết trong dấu ngoặc vuông ([]) theo ký pháp của SQL-Server.

Ví dụ: SELECT **Deptno**, **Deptname** FROM DEPARTMENT;

Các biến cú pháp (*Syntax Variables*), tức là các thành phần ngôn ngữ mà người sử dụng phải điền cụ thể vào khi viết lệnh, sẽ được viết bằng chữ thường (*LowerCase*), trong cặp dấu (< >) và nghiêng.

Ví dụ: CREATE TABLE <tên bảng> (<tên cột> <kiểu>, <tên cột> <kiểu>, ...);

Các thành phần tùy chọn (*Optional*), tức là có thể có hoặc không được viết trong cặp dấu ngoặc vuông đậm nét ([]).

Ví dụ: UPDATE <tên quan hệ>

SET <tên cột> = <biểu thức>, <tên cột> = <biểu thức>, ...

[WHERE <điều kiện>];

Việc lựa chọn một trong các khả năng được thể hiện bởi dấu xỏ đứng đậm (½).

Thành phần bắt buộc phải chọn trong danh sách được viết trong cặp dấu móc đậm nét ({ }).

Giá trị mặc định (*Default Value*) được viết với dấu gạch chân (*Underline*).

Ví dụ: SELECT { * ½ <biểu thức 1>, <biểu thức 2>, ... }

FROM <các bảng>

[ORDER BY <tên cột> ½ <biểu thức> [ASC ½ DESC], ...]

Lệnh SQL có thể được viết trên nhiều dòng và kết thúc lệnh bởi dấu chấm phẩy (;), tuy nhiên từ khóa, tên hàm, tên thuộc tính, tên bảng, tên đối tượng (*Objects*) thì không được phép viết tách xuống hàng. *Trong vận dụng thực tế, từ khóa, tên thuộc tính, tên bảng, tên đối tượng được viết in hoa hoặc chữ thường là như nhau.*

Cho đến bây giờ chúng ta đã có các CSDL với đầy đủ dữ liệu về quản lý học viên - được trình bày trong Chương III, bài 4 (gồm các quan hệ: KHOA, GIẢNG-VIÊN, LỚP-HỌC, MÔN-HỌC, HỌC-VIÊN, KQUẢ-THI), quản lý nhân sự của một công ty EMPLOYMENT – được trình bày trong Chương V, mục 5.3, bài 7 (gồm các quan hệ: DEPARTMENT, EMPLOYEE, JOBS, EMPLHIST) và CSDL quản lý cán bộ - công chức CCVC – được trình bày trong Chương V, mục 5.4, bài 8 (gồm các bảng: ĐƠN-VỊ, LOẠI-ĐƠN-VỊ, NGÁCH-CBVC, NGÁCH-BẬC-LƯƠNG và CBVC). Các CSDL này sẽ được sử dụng làm các mẫu cho việc trình bày các câu lệnh SQL trong toàn bộ chương này.

Các lệnh hỏi - tìm kiếm dữ liệu: (Data Retrieval SQL)

Câu lệnh SELECT - SQL tìm kiếm dữ liệu là một trong số các câu lệnh SQL cài đặt đầy đủ các phép toán quan hệ dựa trên các từ khóa cơ bản SELECT, FROM, WHERE, GROUP BY, ORDER BY, HAVING. Đây là câu lệnh được sử dụng phổ biến nhất với mục đích tìm kiếm thông tin trong CSDL quan hệ. Cú pháp tổng quát của câu lệnh như sau:

SELECT [DISTINCT]<biểu thức 1>, <biểu thức 2>, ...

FROM <tên bảng 1>, <tên bảng 2>, ...

[WHERE <điều kiện chọn>]

[GROUP BY <tên cột 1>, <tên cột 2>, ...]

[ORDER BY <tên cột 1> | <biểu thức số 1>[ASC || DESC], ...]

[HAVING <điều kiện in kết quả>];

Chúng ta sẽ lần lượt làm rõ từng phần của cú pháp ngôn ngữ. Cơ sở dữ liệu được sử dụng để minh họa các ví dụ trong chương này là hệ quản lý nhân sự của một công ty EMPLOYMENT với các bảng – quan hệ: DEPARTMENT, EMPLOYEE, JOBS, EMPLHIST đã nói trên.

Tìm thông tin từ các cột của bảng.


```
SELECT [DISTINCT]{ * | <biểu thức 1>[AS <Tên mới 1>],
<biểu thức 2>[AS <Tên mới 2>], ... }

FROM <tên bảng>;
```

Câu hỏi 6.1.1: Cho danh sách các phòng ban (bao gồm tất cả các thông tin về Mã số (

Câu hỏi 6.1.1: Cho danh sách các phòng ban (bao gồm tất cả các thông tin về Mã số (**DeptNo**), Tên (**DeptName**), Địa điểm (**Loc**), Mã số người lãnh đạo (**Mgr**), Kinh phí hoạt động (**Exp_Budg**) và Doanh thu (**Rev_Budg**) của các phòng ban) trong Công ty:

```
SELECT Deptno, Deptname, Loc, Mgr, Exp_budg, Rev_budg

FROM DEPARTMENT;
```

Khi cần lấy thông tin về tất cả các cột của bảng chúng ta có thể sử dụng dấu sao (*) thay cho việc liệt kê các tên cột của bảng. Câu lệnh trên tương đương với câu lệnh:

```
SELECT * FROM DEPARTMENT;
```

Kết quả của câu lệnh là một bảng (nằm trong bộ nhớ trong):

DepTno	DeptName	Loc	Mgr	Exp_Budg	Rev_Budg
10	Accounting	Dallas	200	10.000	
30	Research	San Fransisco	105	125.000	
40	Sales	Boston	109	280.000	800.000
50	Manufacturing	Houston	210	130.000	
60	Shipping	Houston	215	90.000	

Câu hỏi 6.1.2: Cho Mã số, Tên, Địa điểm, Kinh phí hoạt động của từng phòng ban trong Công ty:

```
SELECT DeptNo, DeptName, Loc, Exp_Budg

FROM DEPARTMENT;
```

Câu lệnh này là cài đặt của phép chiếu trên 4 thuộc tính **DeptNo**, **DeptName**, **Loc** và **Exp_Budg** của bảng DEPARTMENT. Kết quả của câu lệnh là một bảng (nằm trong bộ nhớ trong):

DeptNo	DeptName	Loc	Exp_Budg
10	Accounting	Dallas	10.000
30	Research	San Fransisco	125.000
40	Sales	Boston	280.000
50	Manufacturing	Houston	130.000
60	Shipping	Houston	90.000

Chúng ta giả thiết rằng cần đặt tên khác (*gọi là bí danh - Alias*) cho các cột của bảng kết quả bằng tiếng Việt thay vì dùng tên của thuộc tính của bảng dữ liệu nguồn. Việc này được thực hiện bằng cách thêm từ khóa AS theo sau là một tên mới. Nếu tên có chứa các ký tự đặc biệt và/hoặc khoảng trắng thì viết tên đó trong cặp dấu ngoặc vuông ([]). Ví dụ trên được sửa thành:

SELECT DeptNo AS [Mã số], DeptName AS [Tên phòng], Loc AS [Địa điểm], Exp_Budg as [Kinh phí]

FROM DEPARTMENT;

Kết quả của câu lệnh là như sau:

Mã số	Tên phòng	Địa điểm	Kinh phí
10	Accounting	Dallas	10.000
30	Research	San Fransisco	125.000
40	Sales	Boston	280.000
50	Manufacturing	Houston	130.000
60	Shipping	Houston	90.000

Câu lệnh SELECT không chỉ thực hiện việc trích thông tin từ các cột đơn lẻ của bảng mà có thể thực hiện tính toán theo công thức hay biểu thức bất kỳ dựa trên giá trị của các cột trên từng bản ghi của bảng. Trong đó:

Biểu thức (*expression*) là một dãy các toán hạng (*Operand*) nối với nhau bởi các phép toán (*Operator*). Ở đây:

Toán hạng có thể là:

-Trực hằng (*Literals*): bao gồm hằng số (*Number* - Ví dụ. 1234.56 1234.56), hằng văn bản (*Text*) trong cặp dấu nháy đơn (*Ví dụ. 'Nguyễn Hồng Anh'*), hằng ngày tháng (*Date/Time*) đặt trong cặp dấu hàng rào (# - Ví dụ. #19/05/1890#), và hằng logic (*True* hay *False*) hoặc tên gọi của trực hằng.

-Tên thuộc tính (có thể kèm theo tên bảng và dấu chấm đứng trước). Ví dụ. DEPARTMENT.DeptNo.

- Tên hàm (*function*). Ví dụ. SUM (...), COUNT(...), SIN (...), COS(...)

- Tên biến (*Variable*).

Các phép toán có thể là:

- Các phép toán số học: ^ (*lũy thừa*); * (*nhân*), / (*chia*), % (*chia nguyên*), Mod (*phần dư*); + (*cộng*), - (*trừ*). Thứ tự ưu tiên cao nhất theo 3 cụm từ trái qua phải. Các phép toán số học thường cho kết quả là một số.

- Các phép toán so sánh: <, <=, >, >=, =, <>. Kết quả phép so sánh là giá trị logic (*True* hoặc *False*).

- Các phép toán phạm vi: IN (<*danh sách giá trị*>), BETWEEN <*Min*> AND <*Max*>, LIKE <*Mẫu v.bản*>.

- Các phép toán logic: NOT (*phủ định*), AND (*nối liền - conjunction*), OR (*nối rời - disjunction*). Kết quả các phép toán logic là một giá trị logic.

Câu hỏi 6.1.3: Cho biết Mã số, Tên và lương cả năm của các nhân viên trong công ty:

```
SELECT EmpNo AS [Mã số], Name AS [Tên], Salary * 12 AS [Lương năm]
```

```
FROM EMPLOYEE;
```

Kết quả là bảng:

Mã số	Tên	Lương năm
100	Wilson	20.400
101	Smith	30.000
103	Reed	42.000
105	Watson	54.000
109	Allen	45.600

110	Turner	21.600
200	Chen	34.800
210	Ramirez	43.200
213	McDonnell	19.500
214	Simpson	9.900
215	Di Salvo	32.400
220	Schwartz	50.400

Khi thực hiện phép chiếu tên một quan hệ, các bộ giá trị giống nhau có thể được chọn. Từ khóa DISTINCT được sử dụng nếu muốn chỉ giữ lại 1 bộ trong các bộ giá trị giống nhau tìm được.

Câu hỏi 6.1.4: Cho biết các nhân viên của công ty đang đảm nhận các công việc gì? Đây chính là phép chiếu trên thuộc tính Job của quan hệ EMPLOYEE.

SELECT DISTINCT **Job** FROM EMPLOYEE;

Kết quả là bảng với 6 dòng và 1 cột như sau:

Job
Clrk
Anlt
Mngr
Drvr
Spvr
Slsm

Chọn các dòng của bảng - Mệnh đề WHERE

Trong nhiều trường hợp chúng ta chỉ cần chọn ra những bộ giá trị của bảng thỏa mãn điều kiện nào đó. Mệnh đề WHERE (*WHERE Clause*) với cú pháp) với cú pháp WHERE <điều kiện> cho phép thực hiện điều đó. Ở đây <điều kiện> là một biểu thức mà kết quả là một giá trị logic hoặc đúng (*True*) hoặc sai (*False*). Đây là sự cài đặt của phép chọn (*Selection*) trong đại số quan hệ.

Câu hỏi 6.1.5: Cho danh sách nhân viên của phòng số 40?

```
SELECT * FROM EMPLOYEE WHERE Deptno = 40;
```

Kết quả là bảng có 3 dòng (trên tổng số 12 dòng của bảng nguồn):

EmpNo	Name	Job	Salary	Comm	DeptNo	Sex
101	Smith	Slsm	2.500	1.300	40	F
109	Allen	Mngr	3.800	8.000	40	F
220	Schwartz	Slsm	4.200	5.300	40	F

Câu hỏi 6.1.6: Cho danh sách nhân viên của phòng số 10, 30 và 50.

```
SELECT * FROM EMPLOYEE
```

```
WHERE (DeptNo = 10) OR (DeptNo = 30) OR (DeptNo = 50);
```

Hoặc viết cách khác:

```
SELECT * FROM EMPLOYEE WHERE DeptNo IN (10, 30, 50);
```

Kết quả là bảng:

EmpNo	Name	Job	Salary	Comm	DeptNo	Sex
100	Wilson	Clrk	1.700		10	M
103	Reed	Anlt	3.500		30	M
105	Watson	Mngr	4.500	0	30	M
110	Turner	Clrk	1.800		50	M
200	Chen	Mngr	2.900		10	F
210	Ramirez	Mngr	3.600		50	M

Câu hỏi 6.1.7: Cho danh sách các nhân viên có lương tháng từ 3500 đến 4500 USD:

```
SELECT * FROM EMPLOYEE
```

```
WHERE (Salary >= 3500) AND (Salary <= 4500);
```

Hoặc viết cách khác:

SELECT * FROM EMPLOYEE
WHERE **Salary** BETWEEN 3500 AND 4500;

EmpNo	Name	Job	Salary	Comm	DeptNo	Sex
103	Reed	Anlt	3.500		30	M
105	Watson	Mngr	4.500	0	30	M
109	Allen	Mngr	3.800	8.000	40	F
210	Ramirez	Mngr	3.600		50	M
220	Schwartz	Slsm	4.200	5.300	40	F

Mẫu so sánh trong phép toán LIKE là một giá trị kiểu *Text*, đó là một dãy ký tự bất kỳ trong đó có 2 ký tự có ý nghĩa đặc biệt sau đây:

_ : Đại diện cho một ký tự bất kỳ tại vị trí dấu ?

% : Đại diện cho một nhóm ký tự bất kỳ tại vị trí đó.

Ví dụ: Với tiếng Việt 1 byte (VNI, VietWare, ABC, ...) mẫu '*Nguy_n*' đại diện cho một dãy ký tự có 6 ký tự, trong đó có 4 ký tự đầu là '*Nguy*', ký tự thứ 5 là gì cũng được và ký tự thứ 6 là chữ '*n*'.

Mẫu *%tổ chức%* đại diện cho bất cứ giá trị văn bản nào có chứa hai từ "*tổ chức*".

Câu hỏi 6.1.8: Cho danh sách nhân viên có tên bắt đầu bằng chữ W:

SELECT * FROM EMPLOYEE WHERE **Name** LIKE 'W%';

EmpNo	Name	Job	Salary	Comm	DeptNo	Sex
100	Wilson	Clrk	1.700		10	M
105	Watson	Mngr	4.500	0	30	M

F Lưu ý: Trong MS Access, hằng văn bản được đặt trong cặp dấu nháy kép – ví dụ "

F Lưu ý: Trong MS Access, hằng văn bản được đặt trong cặp dấu nháy kép – ví dụ "Nguyễn Hồng An" - và ký tự đại diện trong mẫu so sánh với phép LIKE là:

? : Đại diện cho một ký tự bất kỳ tại vị trí dấu ?

* : Đại diện cho một nhóm ký tự bất kỳ tại vị trí đó.

Thứ tự hiển thị các bản ghi - Mệnh đề ORDER BY

Trong ví dụ 6.1.6 và 6.1.7 chúng ta thấy các nhân viên không được sắp xếp theo phòng ban hay không theo thứ tự tăng hay giảm dần của mức lương tháng. Để thực hiện được các điều trên, SQL hỗ trợ bởi mệnh đề ORDER BY để sắp xếp kết quả tìm được. Cú pháp mệnh đề này là:

ORDER BY <ten cột>|<biểu thức>[ASC| DESC], <ten cột> , <ten cột>|<biểu thức>[ASC| DESC], ...

Biểu thức phải có giá trị số; nó thể hiện số thứ tự của cột trong bảng kết quả được chỉ định phải sắp xếp thứ tự thay vì phải chỉ rõ tên cột, hơn nữa nếu cột kết quả là cột tính toán thì nó chưa có tên nên các sử dụng biểu thức là một biện pháp thay thế hữu dụng. Có thể sắp xếp theo thứ tự tăng dần (với từ khóa ASC - Viết tắt của ASCending - mặc định là ASC) hoặc giảm dần (DESCending) theo giá trị cột. Trước hết các bản ghi được xếp theo thứ tự của cột thứ nhất; các bản ghi có cùng giá trị ở cột 1 sẽ được sắp xếp theo thứ tự cột thứ 2, Các bản ghi có cùng giá trị ở cả 2 cột 1 và 2 sẽ được xếp theo cột thứ 3 và v.v...

Câu hỏi 6.1.9: Cho danh sách các nhân viên của phòng 10, 30 và 50. Kết quả in ra theo thứ tự tăng dần của mã phòng và giảm dần theo mức lương.

, ...

Biểu thức phải có giá trị số; nó thể hiện số thứ tự của cột trong bảng kết quả được chỉ định phải sắp xếp thứ tự thay vì phải chỉ rõ tên cột, hơn nữa nếu cột kết quả là cột tính toán thì nó chưa có tên nên các sử dụng biểu thức là một biện pháp thay thế hữu dụng. Có thể sắp xếp theo thứ tự tăng dần (với từ khóa ASC - Viết tắt của ASCending - mặc định là ASC) hoặc giảm dần (DESCending) theo giá trị cột. Trước hết các bản ghi được xếp theo thứ tự của cột thứ nhất; các bản ghi có cùng giá trị ở cột 1 sẽ được sắp xếp theo thứ tự cột thứ 2, Các bản ghi có cùng giá trị ở cả 2 cột 1 và 2 sẽ được xếp theo cột thứ 3 và v.v...

Câu hỏi 6.1.9: Cho danh sách các nhân viên của phòng 10, 30 và 50. Kết quả in ra theo thứ tự tăng dần của mã phòng và giảm dần theo mức lương.

SELECT * FROM EMPLOYEE WHERE Deptno IN (10, 30, 50)

ORDER BY **Deptno**, **Salary** DESC;

Hoặc cách viết khác:

SELECT * FROM EMPLOYEE WHERE **DeptNo** IN (10, 30, 50)

ORDER BY 6 ASC, 4 DESC;

Kết quả là bảng:

EmpNo	Name	Job	Salary	Comm	DeptNo	Sex
200	Chen	Mngr	2.900		10	F
100	Wilson	Clrk	1.700		10	M
105	Watson	Mngr	4.500	0	30	M
103	Reed	Anlt	3.500		30	M
210	Ramirez	Mngr	3.600		50	M
110	Turner	Clrk	1.800		50	M

Điều kiện hiển thị các bản ghi - Mệnh đề HAVING

Mệnh đề WHERE cho phép chọn các bản ghi của bảng thỏa mãn điều kiện tìm kiếm. Trong một số trường hợp sau khi tìm được các bản ghi thỏa điều kiện tìm, chúng ta chỉ muốn hiển thị chỉ những bản ghi thỏa một điều kiện khác nữa. SQL hỗ trợ yêu cầu này bởi mệnh đề HAVING <điều kiện>. Thông thường mệnh đề này được áp dụng trong những câu lệnh tìm các bộ giá trị thông qua các quá trình tính toán trên nhóm.

Câu hỏi 6.1.10 đưa ra sau đây để làm ví dụ không được "*đắt giá*" cho lắm nhưng cũng đủ để minh họa điều vừa nêu. "Hãy cho danh sách các nhân viên phòng 10, 30 và 50. Chỉ in những người là lãnh đạo phòng ban".

SELECT * FROM EMPLOYEE WHERE **Deptno** IN (10, 30, 50)

ORDER BY 6 ASC, 4 DESC

HAVING **Job** = "Mngr";

Kết quả là bảng:

EmpNo	Name	Job	Salary	Comm	DeptNo	Sex
-------	------	-----	--------	------	--------	-----

200	Chen	Mngr	2.900		10	F
105	Watson	Mngr	4.500	0	30	M
210	Ramirez	Mngr	3.600		50	M

Câu hỏi 6.1.11: Cho Mã phòng mà người có mức lương cao nhất của phòng lớn hơn 4000 \$US.

Rõ ràng ở đây phải thực hiện 3 công việc:

Phân tổ các nhân viên theo các phòng ban.

Xác định mức lương cao nhất của từng phòng ban.

Chọn phòng nào có mức lương cao nhất trên 4000 \$US. Việc này chỉ có thể tiến hành được sau khi đã tìm ra tất cả các mức lương cao nhất của từng phòng ban. Câu lệnh sau đây là một gợi mở của mục 6.2 sắp được trình bày dưới đây.

```
SELECT Deptno, MAX(Salary)FROM EMPLOYEE
```

```
GROUP BY DeptNo
```

```
HAVING MAX(Salary) > 4000;
```

Deptno	Max (Salary)
30	4.500
40	4.200

Truy vấn thông tin từ nhiều bảng dữ liệu.

Việc thực hiện các câu truy vấn trên nhiều bảng, về bản chất là giống như trên một bảng, tức là cần chỉ ra thông tin gì cần tìm và lấy từ các nguồn dữ liệu nào. Các bảng dữ liệu nguồn này cần chỉ ra trong mệnh đề FROM trong câu lệnh SELECT.

Nếu các bảng dữ liệu nguồn có các tên thuộc tính giống nhau thì tên thuộc tính này phải được viết tường minh trong biểu thức tìm kiếm với tên bảng đi kèm phía trước. Nói chung trong một CSDL quan hệ, các bảng thường có các mối liên hệ với nhau. Các bảng được liên hệ với nhau thông qua phép kết nối và thường là kết nối bằng (*Equi-Join*). Mỗi liên hệ phải được thể hiện trong phép kết nối của mệnh đề FROM hoặc thông qua

điều kiện của mệnh đề WHERE của câu lệnh SELECT. Nếu không thể hiện mối liên hệ này, kết quả sẽ là bảng tích Đề-các của 2 bảng.

Câu hỏi 6.1.12: Cho Mã phòng, Tên phòng và tên người lãnh đạo phòng tương ứng.

Trong câu hỏi này, Tên phòng được lấy từ bảng DEPARTMENT, Mã phòng có thể lấy từ DEPARTMENT hoặc từ bảng EMPLOYEE, còn tên nhân viên làm lãnh đạo phòng được lấy từ bảng EMPLOYEE. Hai bảng này được kết nối với nhau thông qua giá trị của thuộc tính **Mgr** của DEPARTMENT và **EmpNo** của EMPLOYEE.

Một điểm nữa cần lưu ý là thuộc tính **DeptNo** có trong cả 2 bảng DEPARTMENT và EMPLOYEE, do đó khi viết lệnh phải chỉ rõ **DeptNo** của bảng nào (mặc dù chúng là như nhau)

```
SELECT DEPARTMENT.DeptNo, DeptName FROM
```

```
DEPARTMENT, EMPLOYEE
```

```
WHERE DEPARTMENT.Mgr = EMPLOYEE.EmpNo;
```

```
;
```

Hoặc có thể viết cách khác nhờ sử dụng phép kết nối bằng INNER JOIN (đã trình bày trong chương V, mục 5.1, điểm 5.1.1) như sau:

```
SELECT DEPARTMENT.DeptNo, DeptName FROM
```

```
DEPARTMENT INNER JOIN EMPLOYEE
```

```
ON (DEPARTMENT.Mgr = EMPLOYEE.EmpNo);
```

```
);
```

Để giảm nhẹ công việc phải viết tên bảng nhiều lần trong lệnh, SQL hỗ trợ tên bí danh cho bảng bằng cách đặt bí danh ngay sau tên bảng nguồn. Bí danh này có thể được dùng trước khi nó được đặt. Dạng đầu tiên của ví dụ này được viết lại tương đương như sau:

```
SELECT D.DeptNo, D.DeptName, E.Name
```

```
FROM DEPARTMENT D, EMPLOYEE E
```

```
WHERE D.Mgr = E.Empno;
```

;

Kết quả của cả 3 cách thể hiện câu hỏi trên là:

D.Deptno	D.Deptname	E.Name
10	Accounting	Chen
30	Research	Watson
40	Sales	Allen
50	Manufacturing	Ramirez
60	Shipping	Di Salvo

Các câu truy vấn lồng nhau (Query with SubQuery).

Trong nhiều trường hợp chúng ta cần phải tìm kiếm thông tin qua nhiều bước: kết quả của bước trước được sử dụng trong biểu thức của câu truy vấn tiếp theo, rồi kết quả của câu truy vấn này lại được dùng trong biểu thức của câu truy vấn tiếp theo nữa v.v... Bằng ngôn ngữ thủ tục, qua mỗi bước chúng ta phải ghi nhớ lại các kết quả trung gian này. Nếu như vậy thì công việc truy vấn dữ liệu sẽ rất vất vả cho những người thao tác trực tiếp với CSDL. SQL - SELECT cho phép lấy ngay kết quả của một câu truy vấn để xây dựng biểu thức điều kiện cho một câu hỏi khác. Câu hỏi trung gian đó được gọi là câu hỏi con (*SubQuery*). Câu hỏi con phải được bao trong cặp dấu ngoặc tròn trong biểu thức của câu hỏi chính. Sự cho phép câu hỏi con là một trong những ưu điểm nổi bật của ngôn ngữ quản trị CSDL quan hệ.

Câu hỏi 6.1.13: Cho danh sách những người làm việc cùng phòng với ông Watson:

Phân tích câu hỏi này ta có 2 bước:

Bước 1: Tìm số hiệu phòng (mà) ông Watson là nhân viên (giả sử phòng tìm được có mã là *pp*).

Bước 2: Tìm những người có số hiệu phòng làm việc bằng *pp*

Câu hỏi ở bước 1 là câu hỏi con cho câu hỏi ở bước 2. Câu lệnh SQL như sau:

SELECT * FROM EMPLOYEE

WHERE **DeptNo** = ANY (SELECT **DeptNo**

FROM EMPLOYEE WHERE **Name** = 'Watson');

Kết quả là bảng:

EmpNo	Name	Job	Salary	Comm	DeptNo	Sex
103	Reed	Anlt	3.500		30	M
105	Watson	Mngr	4.500	0	30	M

Câu hỏi 6.1.14: Cho biết Mã số, Tên và Mức lương của người lãnh đạo của ông Smith.

Câu hỏi này phải được thực hiện qua 3 bước:

Bước 1: Tìm Mã số phòng (mà) ông Smith làm nhân viên (phòng *pp*).

Bước 2: Tìm Mã số người lãnh đạo phòng *pp* (nhân viên *xx*).

Bước 3: Tìm Mã số, Tên, Mức lương của nhân viên có mã số là *xx*.

Như vậy chúng ta phải viết 2 câu hỏi con lồng nhau trong một câu hỏi chính:

SELECT EmpNo, Name, Salary FROM EMPLOYEE

WHERE EmpNo = ANY

(SELECT Mgr FROM DEPARTMENT

WHERE DeptNo = SOME

(SELECT DeptNo FROM EMPLOYEE

WHERE Name = ‘Smith’

)

);

Câu trả lời là:

Empno	Name	Salary
109	Allen	3.800

Kết quả của câu hỏi con được sử dụng trong phép so sánh với một giá trị khác trong biểu thức điều kiện của câu hỏi bao nó. Các phép so sánh có dạng:

<phép so sánh> [<lượng từ>] (SELECT - câu hỏi con)

Ở đây :

<Phép so sánh> có thể là các phép so sánh số học (>, >=, <, <=, <>, =) hoặc phép toán tập hợp IN, LIKE hoặc NOT LIKE.

<Lượng từ> có thể là ALL, ANY (hoặc SOME). Phép so sánh = ANY có thể được thay tương đương bằng phép toán IN; phép so sánh <> ALL có thể thay tương đương bằng phép toán NOT IN.

Câu hỏi 6.1.15: Cho danh sách nhân viên có mức lương lớn hơn hay bằng mức lương cao nhất của phòng ông McDonnel:

Bước 1: Tìm số hiệu phòng (mà) ông McDonnel là nhân viên (Mã phòng *pp*).

Bước 2: Tạo nhóm nhân viên có mã phòng làm việc là *pp* rồi tính lương trung bình của những người này (Lương cao nhất *zz*).

Bước 3: Tìm những người có lương lớn hơn hay bằng *zz*.

```
SELECT * FROM EMPLOYEE
```

```
WHERE Salary >= ALL
```

```
( SELECT MAX(Salary) FROM EMPLOYEE
```

```
WHERE DeptNo = ANY
```

```
( SELECT DeptNo FROM EMPLOYEE
```

```
WHERE Name = 'McDonnel'
```

```
)
```

```
);
```

Lương cao nhất phòng ông McDonnel là 2700 \$US. Bảng kết quả là:

EmpNo	Name	Job	Salary	Comm	DeptNo	Sex
103	Reed	Anlt	3.500		30	M
105	Watson	Mngr	4.500	0	30	M

109	Allen	Mngr	3.800	8.000	40	F
200	Chen	Mngr	2.900		10	F
210	Ramirez	Mngr	3.600		50	M
215	Di Salvo	Spvr	2.700		60	M
220	Schwartz	Slsm	4.200	5.300	40	F

Nhóm thực hiện tính toán.

Các hàm tính toán trên nhóm các bản ghi (Aggregate Functions):

Qua ví dụ trên, chúng ta đã nhận thấy sự cần thiết của những tính toán trong câu lệnh SELECT. SQL cung cấp một số hàm xây dựng sẵn (*Built-in*) làm việc trên nhóm theo kỹ thuật tính toán nhanh tiên tiến *RushMore*. Đó là các hàm: COUNT (* |<tên cột>) - đếm số bản ghi có giá trị xác định tại cột được cho bởi <tên cột>, SUM (<biểu thức>) - tính tổng giá trị các *biểu thức*, MIN (<biểu thức>) - tìm giá trị nhỏ nhất, MAX (<biểu thức>) - tìm giá trị lớn nhất và AVG (<biểu thức>) - tính giá trị trung bình của *biểu thức* dựa trên các bản ghi của các nhóm. Các hàm này thường phải được đi kèm với mệnh đề GROUP BY để thực hiện phân nhóm các bản ghi theo giá trị các cột nào đó trước khi tính toán. Nếu không có mệnh đề GROUP BY thì câu lệnh sẽ coi toàn bộ các bản ghi của bảng là một nhóm.

Câu hỏi 6.2.1: Cho biết khoản tiền huê hồng (**Commission**) cao nhất và trung bình của các nhân viên (có khoản này).

```
SELECT MAX ( Comm ), AVG ( Comm ) FROM EMPLOYEE;
```

Bảng kết quả:

MAX(Comm)	AVG(Comm)
8000	3650

Chúng ta thấy rằng các hàm trên chỉ tính toán trên các giá trị xác định. Do đó, mặc dù không có sự phân nhóm nhưng hàm MAX (.), SUM (.), COUNT (.) và AVG (.) chỉ tính trên các giá trị xác định. Chỉ có 4 người có tiền huê hồng, nên giá trị trung bình là $(1300 + 0 + 8000 + 5300) / 4 = 3650$.

Câu hỏi 6.2.2: Cho biết Mã số, Tên, Tổng số nhân viên, mức lương cao nhất, thấp nhất, và trung bình của các phòng ban:

```
SELECT A.DeptNo AS [Mã số], A.DeptName AS [Tên phòng],
```

COUNT (*) AS [Tg.số n/v], MAX (**Salary**) AS [Lương Max],

MIN (**Salary**) AS [Lương Min],

AVG (**Salary**) AS [Lương TBình]

FROM DEPARTMENT A, EMPLOYEE B

WHERE A.Deptno = B.Deptno

GROUP BY A.Deptno, A.Deptname;

Bảng kết quả:

Mã số	Tên phòng	TgSố n/v	Lương Max	Lương Min	Lương TBình
10	Accounting	2	2.900	1.700	2.300
30	Research	2	4.500	3.500	4.000
40	Sales	3	4.200	2.500	3.500
50	Manufacturing	2	3.600	1.800	2.700
60	Shipping	3	2.700	825	1.716

Trên đây là danh sách các phòng ban với các mức lương cao nhất, thấp nhất và trung bình của các nhân viên trong từng phòng. Có thể biết được những ai trong phòng có mức lương cao nhất hay thấp nhất trong các phòng ban không ?

Câu hỏi 6.2.3: Cho biết các nhân viên có mức lương cao nhất (tương tự: lương thấp nhất, và lương trung bình) của các phòng:

Ở đây cần phải tiến hành theo hai bước:

Bước 1: Xác định các mức lương cao nhất của từng phòng.

Bước 2: Tìm những nhân viên có mức lương bằng với mức lương cao nhất của phòng đó.

Cần lưu ý rằng, kết quả của câu hỏi con được dùng làm toán hạng trong câu hỏi khác, do đó giá trị các phần tử (hay bộ giá trị) của bảng kết quả phải là vô hướng, tức là, câu lệnh SELECT con chỉ được chọn một biểu thức trong câu lệnh. Chúng ta không thể chỉ chọn một cột mức lương cao nhất trong bước 1, để rồi trong bước 2 chọn ra những người có mức lương trùng với một trong các mức lương tìm được. Bởi vì, người có mức lương cao nhất của một phòng có thể bằng với mức lương của một người bình thường

trong phòng khác. Phải làm sao gán được mức lương cao nhất đi kèm với phòng cụ thể. Chúng ta thực hiện việc biến đổi nhỏ: Đổi Mã phòng và mức lương cao nhất của phòng thành chuỗi rồi ghép lại với nhau thành một chuỗi đặc trưng cho từng phòng. Trong câu hỏi chính, chúng ta cũng ghép hai chuỗi đổi được từ Mã phòng và mức lương của từng nhân viên, rồi so khớp với kết quả ở bước trên. Câu lệnh được viết trong MicroSoft SQL-Server như sau:

```
SELECT *
FROM EMPLOYEE
WHERE STR ( Deptno, 2 ) + STR ( Salary, 5 ) IN
( SELECT STR ( Deptno, 2 ) + STR ( MAX (Salary), 5)
FROM EMPLOYEE GROUP BY DeptNo
ORDER BY DeptNo;
```

Bảng kết quả:

EmpNo	Name	Job	Salary	Comm	DeptNo	Sex
200	Chen	Mngr	2.900		10	F
105	Watson	Mngr	4.500	0	30	M
220	Schwartz	Slsm	4.200	5.300	40	F
210	Ramirez	Mngr	3.600		50	M
215	Di Salvo	Spvr	2.700		60	M

F Ghi chú:

Hàm STR (<ExpN> [, m[, n]]) thực hiện việc đổi một giá trị của biểu thức số <ExpN> ra dãy ký tự chữ số gồm m ký tự, trong đó có n số lẻ sau dấu chấm thập phân. Nếu trong hàm không có n thì n có giá trị mặc nhiên là 0 – không có số lẻ; nếu không có m thì giá trị mặc nhiên của m là 10.

Trong MS Access có thể sử dụng phép toán dấu "và" (&) để ghép nối hai giá trị có kiểu bất kỳ thành một giá trị có kiểu biến thể (*Variant*).

Ngoài các hàm COUNT, SUM, MIN, MAX, AVG ngôn ngữ quản trị CSDL còn cài đặt một số hàm thống kê trong đó có hàm tính phương sai (VAR = *Variance*) và tính

độ lệch chuẩn STDEV (*Standard Deviation*) của một dãy biểu thức dựa trên các cột số của các bản ghi của (các) bảng. Đó là các hàm tính toán trên nhóm các bản ghi, đòi hỏi có mệnh đề GROUP BY trong các câu lệnh truy vấn SQL.

Các hàm tính toán trên bản ghi:

Hầu hết các hệ quản trị CSDL đều cài đặt thư viện các hàm xây dựng sẵn trong ngôn ngữ truy vấn dữ liệu nhằm hỗ trợ việc xây dựng các biểu thức tính toán cho từng bộ giá trị (hay *bản ghi*) của các quan hệ. Vì mỗi hãng cung cấp hệ quản trị CSDL đặt tên các hàm có thể khác nhau hoặc bổ sung một hàm riêng biệt của hãng hay loại bỏ bớt một số hàm, do đó, mục này của bài giảng chỉ nêu ra các hàm thuộc một số lĩnh vực mang tính gợi mở. Khi cài đặt CSDL trên một ngôn ngữ quản trị CSDL cụ thể, các học viên còn phải học hỏi nhiều thông qua các bài giảng trực tiếp (*Tutorials*) và tài liệu hướng dẫn của hãng cung cấp ngôn ngữ quản trị CSDL.

Các hàm toán học:

ABS(x): Trị tuyệt đối của x

SQRT(x): Căn bậc hai của x (Access và SQL-Server là: SQR(x))

LOG(x): Logarit tự nhiên của x

EXP(x): Hàm mũ cơ số e của x : e^x .

SIGN(x): Lấy dấu của số x (-1: $x < 0$, 0: $x = 0$, +1: $x > 0$)

ROUND(x, n): Làm tròn tới n số lẻ (Access và SQL-Server: RND(x))

... và các hàm lượng giác: SIN, COS, TAN, ASIN, ACOS, ATAN ...

Các hàm xử lý chuỗi ký tự:

LEN (str): Cho chiều dài dãy ký tự str .

LEFT(str, n): Lấy n ký tự phía trái của dãy str .

RIGHT(str, n): Lấy n ký tự phía phải của dãy str .

MID(str, p, n): Lấy n ký tự của dãy str kể từ vị trí p trong dãy.

Các hàm xử lý ngày tháng và thời gian:

DATE(): Cho ngày tháng năm hiện tại (Oracle: SYSDATE)

DAY(*dd*): Cho số thứ tự ngày trong tháng của biểu thức ngày *dd*.

MONTH(*dd*): Cho số thứ tự tháng trong năm của biểu thức ngày *dd*.

YEAR(*dd*): Cho năm của biểu thức ngày *dd*.

HOURL(*tt*): Cho giờ trong ngày (0 , 23)

MINUTE(*tt*): Cho số phút của thời gian *tt*.

SECONDS(*tt*): Cho số giây của biểu thức giờ *tt*.

Các hàm chuyển đổi kiểu giá trị:

FORMAT(*bthức*, *mẫu*): Đổi biểu thức có kiểu bất kỳ thành chuỗi theo mẫu đã cho trong tham số thứ 2. Có thể sử dụng hàm STR để thay thế.

... và họ các hàm chuyển đổi biểu thức có kiểu bất kỳ thành một giá trị thuộc kiểu xác định: CSTR, CINT, CLNG, CSIN, CDBL, v.v...

Cú pháp và ngữ nghĩa cụ thể của các hàm này có thể được tìm thấy trong các tài liệu hướng dẫn cụ thể của các hãng cung cấp phần mềm. Tài liệu này không có tham vọng trình bày chi tiết các hàm của ngôn ngữ hệ quản trị CSDL cụ thể.

Ngôn ngữ truy vấn CSDL SQL (tiếp theo)

Nhóm lệnh cập nhật dữ liệu

Nhóm lệnh thao tác dữ liệu (*Data Manipulation Language*) bao gồm các lệnh *Thêm* bộ giá trị mới, *Sửa* giá trị của một bộ của quan hệ và *Hủy bỏ* (các) bộ giá trị của (các) quan hệ (hoặc các bảng). Những lệnh này được gọi chung là lệnh cập nhật CSDL.

Bổ sung (các) bộ giá trị mới.

Có 2 cách bổ sung bộ giá trị mới cho bảng. Cách 1, bổ sung trực tiếp một bộ bởi một lệnh SQL và cách 2, bổ sung nhiều bộ giá trị lấy từ (các) bộ giá trị của các bảng của CSDL.

Bổ sung trực tiếp một bộ giá trị.

- Cú pháp:

```
INSERT INTO <tên bảng> [ (<tên cột 1>, <tên cột 2> ...)]
```

```
VALUES (<biểu thức 1>, <biểu thức 2>, ...);
```

- Ngữ nghĩa: Thêm một bộ giá trị (bản ghi) mới vào bảng có tên được chỉ ra sau từ khóa INTO với giá trị của <biểu thức 1> được gán cho <tên cột 1>, <biểu thức 2> được gán cho <tên cột 2> v.v...

-Lưu ý: Số lượng biểu thức và kiểu giá trị của các biểu thức phải tương ứng với số lượng và kiểu giá trị của các tên cột trong danh sách tên cột của bảng. Ngoài ra, các giá trị còn phải phù hợp với các ràng buộc toàn vẹn định nghĩa trên quan hệ, trong đó có RBTV về khóa chính (*Primary key*), khóa ngoại (*Foreign Key*) và miền giá trị. Tên thuộc các tính khóa chính và khóa ngoại phải có mặt trong danh sách tên cột của lệnh. Nếu các giá trị của các biểu thức sau từ khóa VALUES vi phạm RBTV thì hệ quản trị CSDL sẽ thông báo lỗi và bộ giá trị mới sẽ không được bổ sung vào bảng.

-*Ví dụ 6.3.1.*

Thêm một phòng mới có tên là Marketing, Mã số là 20, đặt tại địa điểm San Diego, kinh phí hoạt động 240.000 \$ / năm. Phòng không có doanh thu và chưa có người phụ trách. Câu lệnh SQL như sau:

```
INSERT INTO DEPARTMENT
```

(DeptNo, DeptName, Loc, Mgr, Exp_Budg, Rev_Budg)

VALUES (20, 'Marketing', 'San Diego', NULL, 240000, NULL);

Kết quả ta có bảng DEPARTMENT với thể hiện mới như sau:

Deptno	Deptname	Loc	Mgr	Exp_budg	Rev_budg
10	Accounting	Dallas	200	10.000	
20	Marketing	San Diego		240.000	
30	Research	San Fransisco	105	125.000	
40	Sales	Boston	109	280.000	800.000
50	Manufacturing	Houston	210	130.000	
60	Shipping	Houston	215	90.000	

Nếu giá trị của các biểu thức trong ngoặc tròn sau từ khoá VALUES hoàn toàn phù hợp về số lượng, miền giá trị và thứ tự của các cột trong bảng thì danh sách tên các cột của bảng sau từ khóa INTO có thể được bỏ qua. Ví dụ trên có thể viết tương đương (và cho kết quả như bảng trên):

INSERT INTO DEPARTMENT

VALUES (20, 'Marketing', 'San Diego', NULL, 240000, NULL);

Thêm một hay nhiều bộ giá trị từ bảng CSDL.

Cú pháp:

INSERT INTO <tên bảng> [(<tên cột 1>, <tên cột 2> ...)]

SELECT <biểu thức 1>, <biểu thức 2> ,

FROM <danh sách các bảng nguồn>

[WHERE <điều kiện>]

[GROUP BY <danh sách cột phân nhóm>]

[ORDER BY <cột 1> [ASC | DESC], <cột 1> [, <cột 1> [ASC] DESC],...]

[...]

[HAVING <điều kiện>];

-Ngữ nghĩa: Cũng như trên, số lượng biểu thức và kiểu giá trị của các biểu thức sau SELECT phải phù hợp với số lượng và kiểu của các cột có tên trong danh sách đi sau tên bảng, đồng thời phải phù hợp với các RBTV được định nghĩa trên quan hệ đó. Nếu bộ giá trị SELECT được vi phạm RBTV định nghĩa trên quan hệ được bổ sung thì sẽ có các thông báo lỗi thích hợp và bộ đó không được bổ sung vào bảng.

Nếu giá trị của các biểu thức sau từ khoá SELECT hoàn toàn phù hợp về số lượng, miền giá trị và thứ tự của các cột trong bảng thì danh sách tên các cột của bảng sau từ khóa INTO có thể được bỏ qua.

-*Ví dụ 6.3.2*:

Bổ sung các bản ghi cho bảng EMPLHIST đối với những nhân viên chưa có quá trình công tác nào trong bảng, với giả thiết thêm rằng họ được tuyển dụng vào làm tại Công ty kể từ ngày 01/01/1980. Câu lệnh SQL được viết như sau:

```
INSERT INTO EMPLHIST
```

```
(EmpNo, Seq, Date_Beg, Salary, FrJob, ToJob, Promo, ToDept)
```

```
SELECT EmpNo, 1, #01/01/1980#, Salary, Job, Job, No, Dept
```

```
FROM EMPLOYEE
```

```
WHERE
```

```
FROM EMPLOYEE
```

```
WHERE EmpNo NOT IN
```

```
(SELECT DISTINCT EmpNo FROM EMPLOYEE);
```

Hoặc viết tương đương (thêm 2 giá trị không xác định NULL cho 2 cột **Date_End** và **FrJob** để cho chúng hoàn toàn phù hợp về số lượng, thứ tự và kiểu giá trị của danh sách thuộc tính của bảng, đồng thời bỏ qua danh sách thuộc tính sau tên bảng):

```
INSERT INTO EMPLHIST
```

```
SELECT EmpNo, 1, #01/01/1980#, NULL, Salary, Job, Job, No, NULL, DeptNo
```

```
FROM EMPLOYEE
```

WHERE

FROM EMPLOYEE

WHERE **Empno** NOT IN

(SELECT DISTINCT **Empno** FROM EMPLHIST

WHERE EMPLHIST.Empno = EMPLOYEE.Empno

);

Kết quả chúng ta có thêm bộ giá trị mới sau đây sẽ được bổ sung vào bảng EMPLHIST:

EmpNo	Seq	Date_Beg	Date_End	Salary	FrJob	ToJob	Promo	FrDept	ToDept
100	1	01/01/80		1700	Clrk	Clrk	N		10
103	1	01/01/81		3500	Anlt	Anlt	N		30
105	1	01/01/81		4500	Mngr	Mngr	N		30
110	1	01/01/81		1800	Clrk	Clrk	N		50
200	1	01/01/81		2900	Mngr	Mngr	N		10
210	1	01/01/81		3600	Mngr	Mngr	N		50
213	1	01/01/81		1625	Clrk	Clrk	N		60
214	1	01/01/81		825	Drvr	Drvr	N		60
215	1	01/01/81		2700	Spvr	Spvr	N		60

Tạo mới một bảng với các bộ giá trị lấy từ CSDL.

Các câu truy vấn dữ liệu để tìm kiếm thông tin tạo ra một bảng trung gian với những mối liên hệ sao cho có thể xem và, nếu được phép, có thể sửa chữa dữ liệu hoặc xóa bỏ chúng. Các QUERY đó đã tạo ra những khung nhìn (VIEW). Trong nhiều trường hợp chúng ta muốn các bảng này trở thành những bảng vật lý, được lưu trữ lâu dài trong CSDL. Điều này có thể được thực hiện nhờ 1 lệnh truy vấn hành động (*Action Query*) gọi là truy vấn tạo bảng mới (*Make Table Query*).

-Cú pháp:

```

SELECT <biểu thức 1>, <biểu thức 2> , ....
FROM <danh sách các bảng nguồn>
INTO TABLE <tên bảng>
[WHERE <điều kiện>]
[GROUP BY <danh sách cột phân nhóm>]
[ORDER BY <cột 1> [ASC | DESC], <cột 1> [, <cột 1> [ASC] DESC],... ]
[... ]
[HAVING <điều kiện>];

```

F Ghi chú: Trong Oracle, câu lệnh này là CREATE TABLE có cú pháp như sau:

```

CREATE TABLE <tên bảng> AS
SELECT <biểu thức 1>, <biểu thức 2> , ....
FROM <danh sách các bảng nguồn> INTO TABLE <tên bảng>
[WHERE <điều kiện>]
[GROUP BY <danh sách cột phân nhóm>]
[ORDER BY <cột 1> [ASC | DESC], <cột 1> [, <cột 1> [ASC] DESC],... ]
[... ]
[HAVING <điều kiện>];

```

-Ví dụ 6.3.3.

Tạo bảng mới tên là MANAGER bao gồm chỉ những nhân viên phụ trách các phòng ban.

```

SELECT * FROM EMPLOYEE INTO TABLE MANAGER
WHERE Job = 'Mgr' ORDER BY DeptNo;

```

Kết quả ta có một bảng mới MANAGER với các bản ghi sau:

EmpNo	Name	Job	Salary	Comm	DeptNo	Sex
200	Chen	Mngr	2.900		10	F
105	Watson	Mngr	4.500	0	30	M
109	Allen	Mngr	3.800	8.000	40	F
210	Ramirez	Mngr	3.600		50	M

Sửa nội dung các bản ghi:

Thông thường có thể sửa nội dung của (các) bản ghi bằng cách cho hiện nội dung của bảng (gọi là *Browse* hoặc *Open* bảng), di chuyển con trỏ (*Cursor*) đến bản ghi cần sửa và thực hiện việc sửa đổi giá trị của các field của bản ghi thể theo yêu cầu. Tuy nhiên cách làm này là phù hợp với các bảng của CSDL nhỏ, và số lượng bản ghi cần sửa cũng rất ít. Đối với các bảng lớn nhiều ngàn - thậm chí hàng chục ngàn như bảng CCVC trong CSDL quản lý công chức viên chức thành phố, hay trăm ngàn như bảng các đối tượng chính sách trong CSDL quản lý các đối tượng chính sách, có công Cách mạng; hoặc nhiều triệu bản ghi như bảng thông tin chi tiết về một cư dân trong CSDL quản lý dân số TP.Hồ Chí Minh v.v... thì việc sửa (cập nhật) thông tin biến động chỉ 1 bản ghi thôi cũng đã là khó khăn.

Ngôn ngữ quản trị CSDL cung cấp một lệnh cho phép sửa đổi nội dung các bản ghi trong CSDL một cách dễ dàng, chính xác và nhanh chóng.

-Cú pháp lệnh như sau:

UPDATE <tên bảng>

SET <tên cột 1> = <biểu thức 1>,

<tên cột 2> = <biểu thức 2>, ...

<tên cột n> = <biểu thức n>

[WHERE <điều kiện>];

-Ngữ nghĩa: Giá trị của các field có tên trong danh sách <tên cột 1>, <tên cột 2> ... của những bản ghi thoả mãn điều kiện sau WHERE sẽ được sửa đổi thành giá trị của các

<biểu thức 1>, <biểu thức 2> ... tương ứng. Nếu không có mệnh đề điều kiện WHERE, thì tất cả các bản ghi của bảng sẽ được sửa đổi.

F Ghi chú: SQL/Plus của Oracle mở rộng cú pháp, cho phép liệt kê danh sách các tên cột của bảng cần sửa đổi giá trị trong cặp dấu ngoặc tròn sau từ khóa SET và danh sách các giá trị mới của chúng trong cặp dấu ngoặc tròn theo sau dấu bằng (=), đồng thời chấp nhận nhiều danh sách như thế trong câu lệnh UPDATE. Cú pháp như sau:

UPDATE <tên bảng>

SET <d/sách tên cột 1> = <d/sách biểu thức 1> ,

< d/sách tên cột 2> = < d/sách biểu thức 2> , ...

< d/sách tên cột n> = < d/sách biểu thức n>

[WHERE <điều kiện>];

-Ví dụ 6.3.4:

Tăng lương thêm 10% cho các nhân viên phụ trách các phòng ban.

UPDATE EMPLOYEE

SET **Salary** = **Salary** * 1.1

WHERE **Job** = 'Mngr';

Kết quả là, lương mới của các nhân viên phụ trách các phòng ban được thể hiện như trong bảng dưới đây:

EmpNo	Name	Job	Salary	Comm	Dept No	Sex
200	Chen	Mngr	3.190		10	F
105	Watson	Mngr	4.950	0	30	M
109	Allen	Mngr	4.180	8.000	40	F
210	Ramirez	Mngr	3.960		50	M

Xóa bỏ các bản ghi khỏi một bảng:

Việc loại bỏ một bản ghi khỏi một bảng trong CSDL là một trong những thao tác cập nhật dữ liệu được tiến hành một cách thường xuyên nhằm đảm bảo phản ánh tình trạng mới nhất của CSDL.

-Cú pháp:

DELETE FROM <ten bảng>

[WHERE <điều kiện>];

-Ngữ nghĩa: Các bản ghi thỏa mãn điều kiện sau WHERE sẽ bị xóa khỏi bảng. Nếu không có mệnh đề WHERE thì tất cả các bản ghi của bảng sẽ bị xóa khỏi bảng.

F Lưu ý: Nếu thao tác trên View, tức là một bảng trung gian của một câu lệnh truy vấn SQL, và người sử dụng được quyền cập nhật xóa dữ liệu, thì khi thực hiện lệnh này các bản ghi trong các bảng vật lý của CSDL cũng sẽ bị xóa.

-Ví dụ 6.3.5:

Xóa tất cả các nhân viên phụ trách các phòng ban khỏi bảng nhân viên EMPLOYEE. Câu lệnh SQL như sau:

DELETE FROM EMPLOYEE

WHERE **Job** = 'Mgr';

Kết quả là, bảng EMPLOYEE chỉ còn các bản ghi sau:

EmpNo	Name	Job	Salary	Comm	Dept No	Sex
100	Wilson	Clrk	1.700		10	M
101	Smith	Slsm	2.500	1.300	40	F
103	Reed	Anlt	3.500		30	M
110	Turner	Clrk	1.800		50	M
213	McDonnel	Clrk	1.625		60	M
214	Simpson	Drvr	825		60	M

215	Di Salvo	Spvr	2.700		60	M
220	Schwartz	Slsn	4.200	5.300		

Các lệnh khai báo cấu trúc CSDL

Mục này sẽ trình bày các khía cạnh logic về bảng và cột của bảng, và các lệnh cần thiết để tạo các bảng cùng các ràng buộc toàn vẹn định nghĩa trên các bảng. Đây là phần đầu tiên trong số các lệnh SQL trong ngôn ngữ định nghĩa dữ liệu (*Data Definition Language - DDL*).

Cách đặt tên đối tượng:

SQL chuẩn hóa (86, 89, 92, 96) đều quy định cách đặt tên các đối tượng như tên bảng, tên cột của bảng, tên View, tên RBTV v.v... (gọi chung là định danh - *Identifier*) như sau:

Tên gọi gồm tối đa 32 ký tự chữ cái Latinh, chữ số Ả-rập và dấu gạch chân (*Underscore*) và phải bắt đầu bằng một chữ cái Latinh hoặc dấu gạch chân. Tuyệt đối không chứa khoảng trắng hay ký tự chữ cái không phải Latinh như tiếng Việt chẳng hạn. Chữ in hoa hay chữ thường đều được xem là như nhau. Tên bảng phải là duy nhất trong CSDL và tên bảng trung gian, và không trùng với bất cứ từ khóa nào trong ngôn ngữ quản trị CSDL.

Tên cột của một bảng phải là khác nhau, nhưng chúng có thể giống nhau nếu chúng nằm trong các bảng khác nhau. Người ta đề nghị rằng những tên mang ngữ nghĩa giống nhau và mô tả cùng một thực thể thì nên đặt tên giống nhau. Chẳng hạn, tên cột số hiệu phòng ban nên đặt là DEPTNO trong cả hai bảng EMPLOYEE và DEPARTMENT, hay tên cột mã ngạch công chức viên chức (CCVC) nên đặt là MA_NGACH trong cả 3 bảng CCVC, danh mục ngạch CCVC: NGACH_LG và bảng ngạch - bậc - hệ số lương: NGACH_BAC_HESO.

-Ví dụ 6.4.1:

EMP_85 : là tên hợp lệ

85EMPL : không hợp lệ, vì tự đầu tiên là chữ số.

BẢNG_NV : không hợp lệ vì có ký tự tiếng Việt

BANG-N/V : không hợp lệ vì có dấu trừ (-) và dấu chia (/)

DANH SACH : không hợp lệ vì có khoảng trắng

UPDATE : không hợp lệ vì là từ dành riêng của ngôn ngữ .

SQL Tạo bảng CSDL:

Một trong những dạng đơn giản nhất của câu lệnh tạo bảng CSDL quan hệ có cú pháp như sau:

```
CREATE TABLE <tên bảng> (  
  
<tên cột 1> <kiểu dữ liệu 1>(<kích thước 1>),  
  
<tên cột 2> <kiểu dữ liệu 2>(<kích thước 2>),  
  
...  
  
<tên cột n> <kiểu dữ liệu n>(<kích thước n>)  
  
);
```

Kiểu dữ liệu có thể là:

-CHAR (*w*) : Kiểu ký tự với kích thước cố định. Chiều dài của giá trị dữ liệu luôn luôn là *w* ký tự. Nếu cột được gán giá trị là một chuỗi ký tự có chiều dài nhỏ hơn *w* thì các ký tự khoảng cách (*space*) sẽ được điền vào bên phải chuỗi (*Right Padded*) để cho đủ *w* ký tự. Kích thước tối thiểu là 1 và tối đa là 255 ký tự.

-VARCHAR(*w*) : Kiểu ký tự với kích thước thay đổi từ 0 đến *w* ký tự. Giá trị lớn nhất của *w* là 2000.

-NUMBER (*w, s*): Kiểu dữ liệu số có kích thước tối đa *w* ký tự (kể chứa dấu chấm thập phân), trong đó có *s* chữ số sau dấu chấm thể hiện phần số lẻ.

-DATE : Kiểu dữ liệu ngày tháng năm

-LOGICAL : Kiểu dữ liệu logic 1 byte có giá trị hoặc đúng (*True*), hoặc sai (*False*).

-*Ví dụ 6.4.2:*

Lệnh SQL để tạo bảng DEPARTMENT như đã nêu trong Chương V, mục 5.3, bài 7 như sau:

```
CREATE TABLE DEPARTMENT (
```

Deptno NUMBER (2),
Deptname CHAR (15),
Loc CHAR (15),
Mgr NUMBER (3),
Exp_budg NUMBER (7),
Rev_budg NUMBER (7)
);

Trong bảng này chúng ta đã xác định **DeptNo** là thuộc tính khóa; một phòng ban phải có một tên gọi, địa điểm, và kinh phí hoạt động xác định, nghĩa là **DeptName**, **Loc**, **Exp_Budg** phải khác NULL; đồng thời chúng ta có thể muốn địa điểm mặc định của các phòng ban là 'Houston'. Cách khai báo các RBTV kiểu này được hỗ trợ bởi các từ khóa PRIMARY KEY, DEFAULT, UNIQUE và NOT NULL như sau:

```
CREATE TABLE DEPARTMENT (  
  
  DeptNo NUMBER (2) PRIMARY KEY,  
  
  DeptName CHAR (15) NOT NULL,  
  
  Loc CHAR (15) NOT NULL DEFAULT 'Houston',  
  
  Mgr NUMBER (3),  
  
  Exp_Budg NUMBER (7) NOT NULL,  
  
  Rev_Budg NUMBER (7)  
);
```

-Ví dụ 6.4.3:

Tạo cấu trúc bảng các chức danh (công việc) JOBS có thể có của công ty:

```
CREATE TABLE JOBS (  
  
  Job CHAR (5) CONSTRAINT PK PRIMARY KEY,
```

Jobname CHAR(15) NOT NULL UNIQUE,

Minsalary NUMBER(4),

Maxsalary NUMBER(4),

Mgrflag CHAR(1)

);

Nhưng chỉ với cách khai báo như trên thì chúng ta vẫn còn quá nhiều việc phải làm trong việc định nghĩa các RBTV trên bảng và các bảng: RBTV về miền giá trị của một thuộc tính, RBTV định nghĩa trên nhiều thuộc tính của một bảng, RBTV phụ thuộc tồn tại, RBTV liên thuộc tính, liên quan hệ v.v... Ngôn ngữ quản trị CSDL cho phép khai báo các RBTV này trong khi tạo cấu trúc các bảng bởi mệnh đề CONSTRAINT mà chúng ta sẽ được làm quen trong mục 6.5 dưới đây.

Khai báo các RBTV

Như đã trình bày trong Chương V, mục 5.3, các ví dụ từ 5.3.1 đến 5.3.12 và phần bài tập cuối Chương V, cơ sở dữ liệu về quản lý nhân viên với các bảng DEPARTMENT, EMPLOYEE, JOBS và EMPLHIST có các RBTV định nghĩa trên chúng như:

-Mã số phòng ban **DeptNo** trong bảng DEPARTMENT, EMPLOYEE chỉ có thể là 10, 20, 30, 40, 50, 60, 70, 80, 90; đồng thời mã phòng ban trong bảng EMPLOYEE là khóa ngoại.

-Mã số nhân viên Empno là một số gồm 3 chữ số có giá trị từ 100 đến 999.

-Mã công việc trong bảng EMPLOYEE là khóa ngoại tham chiếu tới bảng các công việc JOBS.

-Trong bảng công việc JOBS, mức lương tối thiểu (**MinSalary**) của mỗi công việc phải nhỏ hơn mức lương tối đa của nó.

-Khóa của quan hệ (bảng) EMPLHIST gồm 2 thuộc tính: **EmpNo** và **Seq**.

-Các thuộc tính **FrJob**, **ToJob** của bảng EMPLHIST là các thuộc tính khóa ngoại tham chiếu tới bảng JOBS; **FrDept**, **ToDept** của bảng EMPLHIST là các thuộc tính khóa ngoại tham chiếu tới bảng DEPARTMENT.

Cách khai báo các RBTV trên như thế nào, chúng ta sẽ lần lượt được làm quen từng bước qua các mệnh đề CONSTRAINT dưới đây:

RBTV về miền giá trị:

-Cú pháp:

[CONSTRAINT <tên RBTV>] CHECK (<điều kiện>)

Ràng buộc toàn vẹn về miền giá trị này có thể định nghĩa trên một cột của bảng, có thể được viết ngay sau tên thuộc tính mà không cần phải viết tên RBTV trong cụm CONSTRAINT <tên RBTV>. Nếu định nghĩa trên nhiều cột của bảng thì nên sử dụng cụm này và được viết sau khi đã khai báo xong các thuộc tính của bảng.

<Điều kiện> là một biểu thức logic bất kỳ như đã trình bày trong phần câu lệnh truy vấn thông tin, tuy nhiên trong biểu thức không được chứa các câu hỏi con (*SubQueries*).

-*Ví dụ 6.5.4:*

Viết lại câu lệnh khai báo cấu trúc bảng DEPARTMENT với RBTV về miền giá trị cho cột **DeptNo**:

```
CREATE TABLE DEPARTMENT (  
  
  DeptNo NUMBER (2) PRIMARY KEY CHECK (DeptNo MOD 10 = 0),  
  
  DeptName CHAR (15) NOT NULL,  
  
  Loc CHAR (15) NOT NULL DEFAULT "Houston",  
  
  Mgr NUMBER (3),  
  
  Exp_Budg NUMBER (7) NOT NULL,  
  
  Rev_Budg NUMBER (7)  
  
);
```

Phép toán MOD dùng để lấy phần dư của một phép chia hai số nguyên. Mệnh đề CHECK ở trên cho phép chỉ nhận những bản ghi có giá trị ở cột **DeptNo** là bội của 10: 10, 20, 30, ... và 90. Câu lệnh trên có thể được viết cách khác tương đương như sau:

```
CREATE TABLE DEPARTMENT (
```

DeptNo NUMBER (2) PRIMARY KEY,
DeptName CHAR (15) NOT NULL,
Loc CHAR (15) NOT NULL DEFAULT 'Houston',
Mgr NUMBER (3),
Exp_Budg NUMBER (7) NOT NULL,
Rev_Budg NUMBER (7)
 CONSTRAINT PK_VAL CHECK (**DeptNo** MOD 10 = 0)
);

RBTV về khóa ngoại hay phụ thuộc tồn tại:

-Cú pháp: Định nghĩa cho bảng trên nhiều cột của bảng
 ,[CONSTRAINT <tên RBTV>]

FOREIGN KEY (<các thuộc tính khóa ngoại>)

REFERENCES <tên bảng> (<các cột khóa chính>)

Và định nghĩa cho một cột khóa ngoại của bảng:

[CONSTRAINT <tên RBTV>]

REFERENCES <tên bảng> (<các cột khóa chính>)

F *Lưu ý*: trong cú pháp định nghĩa trên một cột của bảng không có các từ khóa FOREIGN KEY và không có dấu phẩy ở trước CONSTRAINT. Các RBTV định nghĩa trên cột được viết ngay trong dòng mô tả cột.

-*Ví dụ 6.5.5*:

Định nghĩa cấu trúc các bảng EMPLOYEE và EMPLHIST:

CREATE TABLE EMPLOYEE (

EmpNo

EmpNo NUMBER(3) PRIMARY KEY,
Name CHAR (10) NOT NULL,
Job CHAR (5) REFERENCES JOBS (**Job**),
Salary NUMBER (5) NOT NULL,
Comm NUMBER (5),
DeptNo NUMBER (2) REFERENCES DEPARTMENT (**DeptNo**),
Sex CHAR (1) CHECK (**Sex** = 'F' OR **Sex** = 'M')
);

Và:

CREATE TABLE EMPLHIST (
EmpNo
EmpNo NUMBER(3),
Seq NUMBER(3) NOT NULL,
Date_Beg DATE NOT NULL,
Date_End DATE,
Salary NUMBER (5) NOT NULL,
FrJob CHAR (5) REFERENCES JOBS (**Job**),
Tojob CHAR (5),
Promo LOGICAL NOT NULL,
FrDept NUMBER (2),
ToDept NUMBER (2) REFERENCES DEPARTMENT (**DeptNo**),
CONSTRAINT P_KEY PRIMARY KEY (**EmpNo**, **Seq**)

);

Với RBTV về khóa ngoại còn có thể sử dụng tùy chọn (*Option*) ON DELETE CASCADE để thực hiện việc xóa một bộ của bảng chứa khóa chính thì sẽ xóa tất cả các bản ghi của bảng chứa khóa ngoại. Chẳng hạn, khi xóa đi một phòng ban khỏi bảng DEPARTMENT, thì tất cả các nhân viên thuộc phòng đó cũng sẽ bị xóa bỏ khỏi bảng EMPLOYEE. Mệnh đề xóa ON DELETE CASCADE mang tính "*lan tỏa*" này được đặt sau mệnh đề REFERENCES.

-Ví dụ 6.5.6:

Định nghĩa cấu trúc các bảng EMPLOYEE và EMPLHIST được viết lại với RBTV về khóa ngoại có sự xóa "*lan tỏa*" như sau:

```
CREATE TABLE EMPLOYEE (  
  
  EmpNo  
  
  EmpNo NUMBER(3) PRIMARY KEY,  
  
  Name CHAR (10) NOT NULL,  
  
  Job CHAR (5) REFERENCES JOBS (Job),  
  
  Salary NUMBER (5) NOT NULL,  
  
  Comm NUMBER (5),  
  
  DeptNo NUMBER (2),  
  
  Sex CHAR (1) CHECK (Sex = 'F' OR Sex = 'M'),  
  
  CONSTRAINT DEPT_CASC REFERENCES DEPARTMENT (DeptNo)  
  
  ON DELETE CASCADE  
  
);
```

Và:

```
CREATE TABLE EMPLHIST (  
  
  EmpNo
```

EmpNo NUMBER(3),
Seq NUMBER(3) NOT NULL,
Date_Beg DATE NOT NULL,
Date_End DATE,
Salary NUMBER (5) NOT NULL,
FrJob CHAR (5) REFERENCES JOBS (**Job**),
ToJob CHAR (5),
Promo LOGICAL NOT NULL,
FrDept NUMBER (2),
ToDept NUMBER (2) REFERENCES DEPARTMENT (**DeptNo**),
 CONSTRAINT PKEY PRIMARY KEY (**EmpNo**, **Seq**),
 CONSTRAINT FKEY FOREIGN KEY (**EmpNo**)
 REFERENCES EMPLOYEE (**EmpNo**, **Seq**)
 ON DELETE CASCADE
);

Một số tùy chọn khác:

-DISABLE : Nếu bổ sung từ khóa này vào sau RBTv, thì hệ quản trị CSDL vẫn ghi nhận RBTv nhưng không bắt buộc kiểm tra sự vi phạm RBTv khi cập nhật dữ liệu cho bảng. Người chủ sở hữu hoặc người quản trị CSDL có thể bắt đầu bắt buộc kiểm tra RBTv bằng cách kích hoạt với từ khóa ENABLE khi thực hiện lệnh sửa đổi cấu trúc bảng ALTER TABLE (trình bày trong mục 6.6 của chương này).

-EXCEPTIONS INTO <tên bảng> : Những bản ghi vi phạm RBTv khi cập nhật dữ liệu (trong quá trình cập nhật theo lô - *Batch Updating*) hoặc khi kích hoạt kiểm tra RBTv sẽ không được đưa vào bảng chính mà sẽ được lưu lại trong bảng các ngoại lệ chỉ ra sau các từ khóa EXCEPTIONS INTO.

Tóm lại, cú pháp tổng quát của lệnh tạo cấu trúc bảng trong CSDL như sau:

```
CREATE TABLE <tên bảng> (  
  <Column 1> <Type 1> (<Size 1>) [ CONSTRAINT clause 1],  
  <Column 2> <Type 2> (<Size 2>) [ CONSTRAINT clause 2],  
  ...  
  <Column n> <Type n> (<Size n>) [ CONSTRAINT clause n]  
  , [ CONSTRAINT clause 1],  
  , [ CONSTRAINT clause 2],  
  1/4  
  , [ CONSTRAINT clause m],  
);
```

Trong đó cú pháp của các khai báo RBTV là như sau:

```
[ CONSTRAINT <tên RBTV> ] NULL 1/2 NOT NULL 1/2  
UNIQUE [ (<tên cột>, <tên cột>... ) ] 1/2  
1/2  
PRIMARY KEY [ (<tên cột>, <tên cột>... ) ] 1/2  
[ 1/2  
[ FOREIGN KEY [ (<tên cột>, <tên cột>... ) ] ]  
REFERENCES <tên bảng> (<tên cột>, <tên cột>... ) 1/2  
CHECK (<điều kiện>)
```

Các lệnh khai báo cấu trúc CSDL

Mục này sẽ trình bày các khía cạnh logic về bảng và cột của bảng, và các lệnh cần thiết để tạo các bảng cùng các ràng buộc toàn vẹn định nghĩa trên các bảng. Đây là phần

đầu tiên trong số các lệnh SQL trong ngôn ngữ định nghĩa dữ liệu (*Data Definition Language - DDL*).

Cách đặt tên đối tượng:

SQL chuẩn hóa (86, 89, 92, 96) đều quy định cách đặt tên các đối tượng như tên bảng, tên cột của bảng, tên View, tên RBTV v.v... (gọi chung là định danh - *Identifier*) như sau:

Tên gọi gồm tối đa 32 ký tự chữ cái Latinh, chữ số Ả-rập và dấu gạch chân (*Underscore*) và phải bắt đầu bằng một chữ cái Latinh hoặc dấu gạch chân. Tuyệt đối không chứa khoảng trắng hay ký tự chữ cái không phải Latinh như tiếng Việt chẳng hạn. Chữ in hoa hay chữ thường đều được xem là như nhau. Tên bảng phải là duy nhất trong CSDL và tên bảng trung gian, và không trùng với bất cứ từ khóa nào trong ngôn ngữ quản trị CSDL.

Tên cột của một bảng phải là khác nhau, nhưng chúng có thể giống nhau nếu chúng nằm trong các bảng khác nhau. Người ta đề nghị rằng những tên mang ngữ nghĩa giống nhau và mô tả cùng một thực thể thì nên đặt tên giống nhau. Chẳng hạn, tên cột số hiệu phòng ban nên đặt là DEPTNO trong cả hai bảng EMPLOYEE và DEPARTMENT, hay tên cột mã ngạch công chức viên chức (CCVC) nên đặt là MA_NGACH trong cả 3 bảng CCVC, danh mục ngạch CCVC: NGACH_LG và bảng ngạch - bậc - hệ số lương: NGACH_BAC_HESO.

-Ví dụ 6.4.1:

EMP_85 : là tên hợp lệ

85EMPL : không hợp lệ, vì tự đầu tiên là chữ số.

BẢNG_NV : không hợp lệ vì có ký tự tiếng Việt

BANG-N/V : không hợp lệ vì có dấu trừ (-) và dấu chia (/)

DANH SACH : không hợp lệ vì có khoảng trắng

UPDATE : không hợp lệ vì là từ dành riêng của ngôn ngữ .

SQL Tạo bảng CSDL:

Một trong những dạng đơn giản nhất của câu lệnh tạo bảng CSDL quan hệ có cú pháp như sau:

```
CREATE TABLE <tên bảng> (
<tên cột 1> <kiểu dữ liệu 1>(<kích thước 1>),
<tên cột 2> <kiểu dữ liệu 2>(<kích thước 2>),
...
<tên cột n> <kiểu dữ liệu n>(<kích thước n>)
);
```

Kiểu dữ liệu có thể là:

-CHAR (*w*) : Kiểu ký tự với kích thước cố định. Chiều dài của giá trị dữ liệu luôn luôn là *w* ký tự. Nếu cột được gán giá trị là một chuỗi ký tự có chiều dài nhỏ hơn *w* thì các ký tự khoảng cách (*space*) sẽ được điền vào bên phải chuỗi (*Right Padded*) để cho đủ *w* ký tự. Kích thước tối thiểu là 1 và tối đa là 255 ký tự.

-VARCHAR(*w*) : Kiểu ký tự với kích thước thay đổi từ 0 đến *w* ký tự. Giá trị lớn nhất của *w* là 2000.

-NUMBER (*w*, *s*): Kiểu dữ liệu số có kích thước tối đa *w* ký tự (kể chứa dấu chấm thập phân), trong đó có *s* chữ số sau dấu chấm thể hiện phần số lẻ.

-DATE : Kiểu dữ liệu ngày tháng năm

-LOGICAL : Kiểu dữ liệu logic 1 byte có giá trị hoặc đúng (*True*), hoặc sai (*False*).

-*Ví dụ 6.4.2:*

Lệnh SQL để tạo bảng DEPARTMENT như đã nêu trong Chương V, mục 5.3, bài 7 như sau:

```
CREATE TABLE DEPARTMENT (
Deptno NUMBER (2),
Deptname CHAR (15),
Loc CHAR (15),
Mgr NUMBER (3),
```

Exp_budg NUMBER (7),

Rev_budg NUMBER (7)

);

Trong bảng này chúng ta đã xác định **DeptNo** là thuộc tính khóa; một phòng ban phải có một tên gọi, địa điểm, và kinh phí hoạt động xác định, nghĩa là **DeptName**, **Loc**, **Exp_Budg** phải khác NULL; đồng thời chúng ta có thể muốn địa điểm mặc định của các phòng ban là 'Houston'. Cách khai báo các RBTV kiểu này được hỗ trợ bởi các từ khóa PRIMARY KEY, DEFAULT, UNIQUE và NOT NULL như sau:

CREATE TABLE DEPARTMENT (

DeptNo NUMBER (2) PRIMARY KEY,

DeptName CHAR (15) NOT NULL,

Loc CHAR (15) NOT NULL DEFAULT 'Houston',

Mgr NUMBER (3),

Exp_Budg NUMBER (7) NOT NULL,

Rev_Budg NUMBER (7)

);

-Ví dụ 6.4.3:

Tạo cấu trúc bảng các chức danh (công việc) JOBS có thể có của công ty:

CREATE TABLE JOBS (

Job CHAR (5) CONSTRAINT PK PRIMARY KEY,

Jobname CHAR(15) NOT NULL UNIQUE,

Minsalary NUMBER(4),

Maxsalary NUMBER(4),

Mgrflag CHAR(1)

);

Nhưng chỉ với cách khai báo như trên thì chúng ta vẫn còn quá nhiều việc phải làm trong việc định nghĩa các RBTV trên bảng và các bảng: RBTV về miền giá trị của một thuộc tính, RBTV định nghĩa trên nhiều thuộc tính của một bảng, RBTV phụ thuộc tồn tại, RBTV liên thuộc tính, liên quan hệ v.v... Ngôn ngữ quản trị CSDL cho phép khai báo các RBTV này trong khi tạo cấu trúc các bảng bởi mệnh đề CONSTRAINT mà chúng ta sẽ được làm quen trong mục 6.5 dưới đây.

Khai báo các RBTV

Như đã trình bày trong Chương V, mục 5.3, các ví dụ từ 5.3.1 đến 5.3.12 và phần bài tập cuối Chương V, cơ sở dữ liệu về quản lý nhân viên với các bảng DEPARTMENT, EMPLOYEE, JOBS và EMPLHIST có các RBTV định nghĩa trên chúng như:

- Mã số phòng ban **DeptNo** trong bảng DEPARTMENT, EMPLOYEE chỉ có thể là 10, 20, 30, 40, 50, 60, 70, 80, 90; đồng thời mã phòng ban trong bảng EMPLOYEE là khóa ngoại.

- Mã số nhân viên Empno là một số gồm 3 chữ số có giá trị từ 100 đến 999.

- Mã công việc trong bảng EMPLOYEE là khóa ngoại tham chiếu tới bảng các công việc JOBS.

- Trong bảng công việc JOBS, mức lương tối thiểu (**MinSalary**) của mỗi công việc phải nhỏ hơn mức lương tối đa của nó.

- Khóa của quan hệ (bảng) EMPLHIST gồm 2 thuộc tính: **EmpNo** và **Seq**.

- Các thuộc tính **FrJob**, **ToJob** của bảng EMPLHIST là các thuộc tính khóa ngoại tham chiếu tới bảng JOBS; **FrDept**, **ToDept** của bảng EMPLHIST là các thuộc tính khóa ngoại tham chiếu tới bảng DEPARTMENT.

Cách khai báo các RBTV trên như thế nào, chúng ta sẽ lần lượt được làm quen từng bước qua các mệnh đề CONSTRAINT dưới đây:

RBTV về miền giá trị:

- Cú pháp:

[CONSTRAINT <tên RBTV>] CHECK (<điều kiện>)

Ràng buộc toàn vẹn về miền giá trị này có thể định nghĩa trên một cột của bảng, có thể được viết ngay sau tên thuộc tính mà không cần phải viết tên RBTV trong cụm CONSTRAINT <tên RBTV>. Nếu định nghĩa trên nhiều cột của bảng thì nên sử dụng cụm này và được viết sau khi đã khai báo xong các thuộc tính của bảng.

<Điều kiện> là một biểu thức logic bất kỳ như đã trình bày trong phần câu lệnh truy vấn thông tin, tuy nhiên trong biểu thức không được chứa các câu hỏi con (*SubQueries*).

-Ví dụ 6.5.4:

Viết lại câu lệnh khai báo cấu trúc bảng DEPARTMENT với RBTV về miền giá trị cho cột **DeptNo**:

```
CREATE TABLE DEPARTMENT (  
  
  DeptNo NUMBER (2) PRIMARY KEY CHECK (DeptNo MOD 10 = 0),  
  
  DeptName CHAR (15) NOT NULL,  
  
  Loc CHAR (15) NOT NULL DEFAULT "Houston",  
  
  Mgr NUMBER (3),  
  
  Exp_Budg NUMBER (7) NOT NULL,  
  
  Rev_Budg NUMBER (7)  
  
);
```

Phép toán MOD dùng để lấy phần dư của một phép chia hai số nguyên. Mệnh đề CHECK ở trên cho phép chỉ nhận những bản ghi có giá trị ở cột **DeptNo** là bội của 10: 10, 20, 30, ... và 90. Câu lệnh trên có thể được viết cách khác tương đương như sau:

```
CREATE TABLE DEPARTMENT (  
  
  DeptNo NUMBER (2) PRIMARY KEY,  
  
  DeptName CHAR (15) NOT NULL,  
  
  Loc CHAR (15) NOT NULL DEFAULT 'Houston',  
  
  Mgr NUMBER (3),  
  
  Exp_Budg NUMBER (7) NOT NULL,
```

Rev_Budg NUMBER (7)

CONSTRAINT PK_VAL CHECK (**DeptNo** MOD 10 = 0)

);

RBTV về khóa ngoại hay phụ thuộc tồn tại:

-Cú pháp: Định nghĩa cho bảng trên nhiều cột của bảng

,[CONSTRAINT <tên RBTV>]

FOREIGN KEY (<các thuộc tính khóa ngoại>)

REFERENCES <tên bảng> (<các cột khóa chính>)

Và định nghĩa cho một cột khóa ngoại của bảng:

[CONSTRAINT <tên RBTV>]

REFERENCES <tên bảng> (<các cột khóa chính>)

F *Lưu ý*: trong cú pháp định nghĩa trên một cột của bảng không có các từ khóa FOREIGN KEY và không có dấu phẩy ở trước CONSTRAINT. Các RBTV định nghĩa trên cột được viết ngay trong dòng mô tả cột.

-*Ví dụ 6.5.5*:

Định nghĩa cấu trúc các bảng EMPLOYEE và EMPLHIST:

CREATE TABLE EMPLOYEE (

EmpNo

EmpNo NUMBER(3) PRIMARY KEY,

Name CHAR (10) NOT NULL,

Job CHAR (5) REFERENCES JOBS (**Job**),

Salary NUMBER (5) NOT NULL,

Comm NUMBER (5),

DeptNo NUMBER (2) REFERENCES DEPARTMENT (**DeptNo**),

Sex CHAR (1) CHECK (**Sex** = 'F' OR **Sex** = 'M')

);

Và:

CREATE TABLE EMPLHIST (

EmpNo

EmpNo NUMBER(3),

Seq NUMBER(3) NOT NULL,

Date_Beg DATE NOT NULL,

Date_End DATE,

Salary NUMBER (5) NOT NULL,

FrJob CHAR (5) REFERENCES JOBS (**Job**),

Tojob CHAR (5),

Promo LOGICAL NOT NULL,

FrDept NUMBER (2),

ToDept NUMBER (2) REFERENCES DEPARTMENT (**DeptNo**),

CONSTRAINT P_KEY PRIMARY KEY (**EmpNo**, **Seq**)

);

Với RBTV về khóa ngoại còn có thể sử dụng tùy chọn (*Option*) ON DELETE CASCADE để thực hiện việc xóa một bộ của bảng chứa khóa chính thì sẽ xóa tất cả các bản ghi của bảng chứa khóa ngoại. Chẳng hạn, khi xóa đi một phòng ban khỏi bảng DEPARTMENT, thì tất cả các nhân viên thuộc phòng đó cũng sẽ bị xóa bỏ khỏi bảng EMPLOYEE. Mệnh đề xóa ON DELETE CASCADE mang tính "*lan tỏa*" này được đặt sau mệnh đề REFERENCES.

-Ví dụ 6.5.6:

Định nghĩa cấu trúc các bảng EMPLOYEE và EMPLHIST được viết lại với RBTV về khóa ngoại có sự xóa "*lan tỏa*" như sau:

```
CREATE TABLE EMPLOYEE (  
  
  EmpNo  
  
  EmpNo NUMBER(3) PRIMARY KEY,  
  
  Name CHAR (10) NOT NULL,  
  
  Job CHAR (5) REFERENCES JOBS (Job),  
  
  Salary NUMBER (5) NOT NULL,  
  
  Comm NUMBER (5),  
  
  DeptNo NUMBER (2),  
  
  Sex CHAR (1) CHECK (Sex = 'F' OR Sex = 'M'),  
  
  CONSTRAINT DEPT_CASC REFERENCES DEPARTMENT (DeptNo)  
  
  ON DELETE CASCADE  
  
);
```

Và:

```
CREATE TABLE EMPLHIST (  
  
  EmpNo  
  
  EmpNo NUMBER(3),  
  
  Seq NUMBER(3) NOT NULL,  
  
  Date_Beg DATE NOT NULL,  
  
  Date_End DATE,  
  
  Salary NUMBER (5) NOT NULL,  
  
  FrJob CHAR (5) REFERENCES JOBS (Job),
```

ToJob CHAR (5),
Promo LOGICAL NOT NULL,
FrDept NUMBER (2),
ToDept NUMBER (2) REFERENCES DEPARTMENT (**DeptNo**),
 CONSTRAINT PKEY PRIMARY KEY (**EmpNo**, **Seq**),
 CONSTRAINT FKEY FOREIGN KEY (**EmpNo**)
 REFERENCES EMPLOYEE (**EmpNo**, **Seq**)
 ON DELETE CASCADE
);

Một số tùy chọn khác:

DISABLE : Nếu bổ sung từ khóa này vào sau RBTV, thì hệ quản trị CSDL vẫn ghi nhận RBTV nhưng không bắt buộc kiểm tra sự vi phạm RBTV khi cập nhật dữ liệu cho bảng. Người chủ sở hữu hoặc người quản trị CSDL có thể bắt đầu bắt buộc kiểm tra RBTV bằng cách kích hoạt với từ khóa ENABLE khi thực hiện lệnh sửa đổi cấu trúc bảng ALTER TABLE (trình bày trong mục 6.6 của chương này).

-EXCEPTIONS INTO <tên bảng> : Những bản ghi vi phạm RBTV khi cập nhật dữ liệu (trong quá trình cập nhật theo lô - *Batch Updating*) hoặc khi kích hoạt kiểm tra RBTV sẽ không được đưa vào bảng chính mà sẽ được lưu lại trong bảng các ngoại lệ chỉ ra sau các từ khóa EXCEPTIONS INTO.

Tóm lại, cú pháp tổng quát của lệnh tạo cấu trúc bảng trong CSDL như sau:

```

CREATE TABLE <tên bảng> (
  <Column 1> <Type 1> (<Size 1>) [ CONSTRAINT clause 1],
  <Column 2> <Type 2> (<Size 2>) [ CONSTRAINT clause 2],
  ...

```

<Column n > <Type n > (<Size n >) [CONSTRAINT *clause n*]
 , [CONSTRAINT *clause 1*],
 , [CONSTRAINT *clause 2*],
 $\frac{1}{4}$
 , [CONSTRAINT *clause m*],
);

Trong đó cú pháp của các khai báo RBTV là như sau:

[CONSTRAINT <tên RBTV>] NULL $\frac{1}{2}$ NOT NULL $\frac{1}{2}$

UNIQUE [(<tên cột>, <tên cột>...)] $\frac{1}{2}$

$\frac{1}{2}$

PRIMARY KEY [(<tên cột>, <tên cột>...)] $\frac{1}{2}$

[$\frac{1}{2}$

[FOREIGN KEY [(<tên cột>, <tên cột>...)]]

REFERENCES <tên bảng> (<tên cột>, <tên cột>...) $\frac{1}{2}$

CHECK (<điều kiện>)

Ngôn ngữ tân từ

Lôgic toán và ứng dụng của nó vào CSDL

Định nghĩa 7.1.1:

Biểu thức lôgic là một phát biểu (*Statment*) mà giá trị của nó có thể hoặc là đúng hoặc là sai. Biểu thức lôgic có giá trị luôn luôn đúng (hoặc luôn luôn sai) được gọi là hằng đúng (hoặc hằng sai - tương ứng).

Ví dụ 7.1:

Hôm nay trời sẽ có mưa.

$2 > 1$ là biểu thức hằng đúng.

$1 > 5$ là biểu thức hằng sai.

Lôgic tân từ là một mệnh đề dựa trên tân từ của phát biểu đó. Nó gồm có 2 phần: Phần cú pháp và phần diễn giải.

Phần cú pháp:

Trước hết chúng ta cần phải thống nhất về các ký hiệu:

() - Biểu thức trong ngoặc.

Biến: Tên gọi của một đại lượng biến thiên trong một miền giá trị xác định. Thường dùng các chữ cái nhỏ ở cuối bảng mẫu tự để đặt tên cho biến: x, y, z, ...

Hằng: Là các đại lượng không đổi. Thường sử dụng các chữ cái ở đầu bảng mẫu tự như a, b, c, ... làm tên hằng.

Hàm: Là một ánh xạ từ một miền giá trị vào tập hợp gồm 2 giá trị hoặc đúng hoặc sai. Thường dùng các chữ nhỏ ở giữa bảng mẫu tự như f, g, h. ...

Tân từ: Là một biểu thức được xây dựng dựa trên biểu thức lôgic. Thường dùng các chữ in hoa ở giữa như P, Q, R, ...

Các phép toán lôgic: phủ định (\neg), kéo theo (\supset), nối liền (\wedge - *conjunction*) và nối rời (\vee - *disjunction*).

Các lượng từ: với mọi (") và tồn tại (\$)

Định nghĩa 7.1.2:

Tân từ một ngôi được định nghĩa trên 1 tập X và một biến x có giá trị chạy trên các phần tử của X .

Với mỗi giá trị của x , tân từ $P(x)$ là một mệnh đề logic, tức là nó có giá trị hoặc là đúng (Đ), hoặc là sai (S).

Ví dụ 7.2:

$P(x)$, x là biến chạy trên X , là một tân từ.

$P(gt)$, $gt \in X$ là một mệnh đề.

Ví dụ 7.3: X là tập hợp những người có tên như sau:

$X = \{ \text{Nguyễn Văn Nam, Đặng Thị Thúy, Hồ thiệu Hùng ...} \}$

Với tân từ $NỮ(x)$ được xác định như: " x là người nữ". Khi đó mệnh đề:

$NỮ(\text{Nguyễn Văn Nam})$ - cho kết quả là sai.

$NỮ(\text{Đặng Thị Thúy})$ - Cho kết quả là đúng.

Định nghĩa 7.1.3:

Tân từ n ngôi được định nghĩa trên các tập X_1, X_2, \dots, X_n và n biến x_1, x_2, \dots, x_n lấy giá trị trên các tập X_i tương ứng. Với mỗi $a_i \in X_i$ $x_i = a_i$. Tân từ n ngôi là một mệnh đề.

Ký hiệu: $P(x_1, x_2, \dots, x_n)$

Ví dụ 7.4. $CHA(x_1, x_2)$: " x_1 là CHA của x_2 ".

Chú ý:

Các X_i không nhất thiết phải là rời nhau.

Với $x_i = a_i$, $P(x_1, x_2, \dots, a_i, \dots, x_n)$ là tân từ $n-1$ ngôi.

Định nghĩa 7.1.4:

Từ (Word) được định nghĩa một cách truy hồi như sau:

(i) Từ là một hằng hay một biến.

(ii) $f(t_1, t_2, \dots, t_n)$ là một hàm n ngôi thì f là một từ.

Định nghĩa 7.1.5:

Công thức (Formula):

(i) Công thức nguyên tố là một tân từ n ngôi $P(t_1, t_2, \dots, t_n)$;

Ở đây t_1, t_2, \dots, t_n là các từ.

Công thức nguyên tố là một công thức.

(ii) Nếu F_1, F_2, \dots, F_n là các công thức thì các biểu thức sau cũng là các công thức:

$(\cdot) F_1 F_2$

$(\cdot) F_1 F_2$

$(\cdot) F_1 \text{ P } F_2$

$(\cdot) F_1$

(iii) Nếu F_1 là một công thức thì $\neg F_1, \exists x: F_1$ cũng là các công thức.

(iv) Nếu F_1 là một công thức thì (F_1) cũng là một công thức.

**** Lưu ý:** Chúng ta đã biết các phép biến đổi tương đương sau:

$(\cdot) F_1 \text{ P } F_2 \circ F_1 F_2$

$(\cdot) (F_1 \text{ P } F_2) \circ F_1 F_2$

$(\cdot) (\neg x F_1) \circ \neg x F_1$

Định nghĩa 7.1.6:

(i) Một công thức được gọi là "đóng" nếu mọi biến của nó đều có kèm với lượng từ.

(i) Một công thức được gọi là "mở" nếu tồn tại một biến không có kèm với lượng từ. Biến này còn được gọi là biến tự do.

Định nghĩa 7.1.7: Dạng chuẩn Prenexe:

F là một công thức có dạng: $Q_1 x_1, Q_2 x_2, \dots, Q_n x_n M$

Với: Q_i là các lượng từ ($i = 1, 2, \dots, n$).

x_i là các biến.

M là một ma trận công thức.

Nếu M không chứa lượng từ thì ta nói rằng F có dạng chuẩn Prenexe.

Ví dụ 7.5:

" $x \neq t$ " $y \neq z$ ($P(x, y, a) \vee Q(y, z, t) \wedge R(x, t)$)

Diễn giải và mô hình

Diễn giải của 1 công thức:

Một diễn giải của một công thức gồm 4 phần:

Miền giá trị của các biến của công thức (ký hiệu là tập M)

Việc sử dụng công thức:

Hằng là một giá trị cụ thể thuộc M .

Hàm n ngôi của M_n lên M là một ánh xạ $f : M_n \rightarrow M$

Tên từ trên M_n là một quan hệ n ngôi trên tập M_n .

Ý nghĩa của công thức và

Xác định 1 quan hệ n ngôi trên tập M_n

Ví dụ 7.6 :

Cho $M = \{ \text{Trí, Minh, Hưng, Long, Đoàn, Tuấn} \}$ và một công thức C có dạng như sau:

$C: "x \neq y" \wedge (\neq z (P(x,y) \wedge P(y,z) \vee Q(x, z))$

Tập diễn giải của công thức có thể là:

M: là miền giá trị của các biến x, y, z .

Các tân từ:

P: CHA

Q: ÔNG

Ý nghĩa:

CHA (x, y): x có cha là y (hoặc: cha của x là y).

ÔNG (x, y): x có ông là y (hoặc ông của x là y)

Các quan hệ 2 ngôi trên M^2 :

CHA = {(Trí, Minh), (Long, Đoàn), (Minh, Huy), (Đoàn, Tuấn)}

ÔNG = { (Trí, Huy), (Long, Tuấn) }

Mỗi khi xác định được diễn giải phải đánh giá

(i) Công thức là đúng hay là sai. Ở đây C là đúng.

(ii) Với công thức mở có m biến tự do thì nó xác định 1 quan hệ m ngôi trên M^m .

Với một bộ của quan hệ này, nếu ta thay thế m giá trị vào cho m biến thì công thức trở thành "đóng" và nó có giá trị hoặc đúng hoặc sai.

Nếu tập các bộ m được cho bởi công thức "mở" mà rỗng (không có bộ nào để công thức là đúng, thì ta nói rằng "*công thức này là sai*", ngược lại thì công thức là đúng.

Các quy tắc lượng giá công thức.

(i). Lượng giá Tân từ:

Tân từ bậc n $P(x_1, x_2, \dots, x_n)$ được liên kết với 1 quan hệ n ngôi trên R .

Lượng giá P khi x_i lấy giá trị $a_i, i = 1, 2, \dots, n$, và $(a_1, a_2, \dots, a_n) \in R$:

$P(x_1, x_2, \dots, x_n) : D(a_1, a_2, \dots, a_n) \in R$

$P(x_1, x_2, \dots, x_n) : S(a_1, a_2, \dots, a_n) \in R$.

(ii) Với các phép toán \wedge , \vee , \neg và \rightarrow thì sử dụng bảng chân trị (hoặc còn gọi là bảng sự thật) sau:

F_1	F_2	$F_1 \wedge F_2$	$F_1 \vee F_2$	$F_1 \rightarrow F_2$
Đ	Đ	Đ	Đ	Đ
Đ	S	S	Đ	S
S	Đ	S	Đ	Đ
S	S	S	S	Đ

(iii) x là biến của công thức F : Công thức $\forall x F(x)$ là đúng với mọi trị a \in M mà x lấy giá trị $F(a)$ là đúng.

Nếu ngược lại thì $\forall x F(x)$ là sai. Do đó $\forall x F(x)$ đồng nhất với công thức: $F(a_i)$ trên tất cả các $a_i \in M$.

(iv) Công thức $\exists x F(x)$ là đúng khi có ít nhất một $a_i \in M$ sao cho $F(a_i)$ là đúng. Do đó nó đồng nhất với $F(a_i)$ trên tất cả các $a_i \in M$.

Định nghĩa 7.1.8

Một diễn giải của một tập các công thức F được gọi là một mô hình nếu và chỉ nếu $\models F$ $\forall F \in F$, F là đúng trên diễn giải này.

Định nghĩa 7.1.9

Một công thức E là hệ quả logic của F , ký hiệu là $F \models E$, nếu và chỉ nếu E là đúng trên mọi mô hình của F .

Ứng dụng logic toán trong CSDL.

Dẫn nhập

CSDL: mô hình hóa thông tin gồm các các sự kiện được liên kết hay biểu diễn một tình trạng của thế giới thực.

Ví dụ 7.7: Cho tập cơ sở về người:

$M = \{ \text{Trí, Minh, Huy, Long, Đoàn, Tuấn} \}$

Ta có các sự kiện:

Cha của Trí là Minh.

Cha của Long là Đoàn.

Các sự kiện này được tập hợp lại thành CSDL.

Trong ngôn ngữ tân từ nó là công thức đóng, không có biến, chỉ có hằng (tức là các mệnh đề).

Khái niệm cha được liên kết với toán tử CHA:

CHA (Trí, Minh): thay cho phát biểu "*Cha của Trí là Minh*".

Ngoài ra, chúng ta còn quan tâm đến các tính chất tổng quát hơn:

(T1) Nếu cha của x là y thì y có con là x .

(T2) Mọi người x đều có cha là y .

(T3) Nếu x có cha là y và y có cha là z , thì ông của x là z .

Các tính chất trên đảm bảo tính chất nhất quán của thông tin cơ sở.

Chúng ta có thể công thức hóa các tính chất trên như sau:

(T1) " x " y (CHA(x, y) \vdash CON(y, x))

(T2) " x ($\$ x$ (CHA(x, y))

(T3) " y ($\$ z$ (CHA (x, z) CHA(z, y) \vdash ÔNG (x, y))

Tham khảo CSDL:

(i) Câu hỏi đóng tương ứng với công thức đóng. Câu trả lời là có hiệu lực, là đúng hoặc sai.

Ví dụ 7.8:

Trí có cha là Minh ? CHA (Trí, Minh) ?

Con của Minh là ai ? $\$ x \text{ CON (Minh, x) ?}$

Ông của Long là Tuấn ? $\text{ÔNG (Long, Tuấn) ?}$

Câu trả lời cho (1) và (3) là đúng hoặc sai được dựa trên thông tin cơ sở. Đối với câu (2) việc tìm câu trả lời phải dựa vào suy luận:

Áp dụng (T1) thay thế biến y bằng Minh, ta có:

" $x \text{ CHA (x, Minh) } \vdash \text{CON (Minh, x)}$

Sử dụng công thức này, dựa trên thông tin cơ sở ta có:

$\text{CHA (Trí, Minh) } \vdash \text{CON (Minh, Trí)}$

Vậy:

CON (Minh, Trí) : là câu trả lời *đúng*.

(ii) Câu hỏi mở tương ứng với một công thức mở.

Ví dụ 7.9:

(c1) Cha của Trí là ai? CHA (Trí, x) ?

(c2) Xác định ông của Trí ? ÔNG (Trí, x) ?

Một phần tử là một câu trả lời của công thức mở nếu khi thay vào biến x thì công thức đóng thu được trị là đúng.

Lôgic *monotypéc*: Các biến lấy trị từ 1 tập giá trị duy nhất.

Lôgic *multitypéc*: Mỗi biến lấy giá trị từ một tập giá trị riêng.

Ví dụ 7.10:

CƯỚI (NAM, NỮ)

CƯỚI (x, y) x lấy trị từ miền giá trị NAM

y là trị từ miền giá trị NỮ

Ngôn ngữ tân từ có biến là bộ.

Một câu hỏi trong ngôn ngữ tân từ có biến là bộ- n thỏa các quy tắc sau:

Biến là một bộ của quan hệ.

Từ: là hằng, biến hay biểu thức có dạng $s[c]$, c là biến; c là tập các thuộc tính của quan hệ, còn được gọi là từ chiếu.

Các biểu thức:

$R_s : R$: là một quan hệ;

s : là biến bộ – n .

được gọi là từ. Miền giá trị sẽ định nghĩa miền biến thiên của s .

$t_1 q a, t_1 q t_2$.

Ở đây t_1 và t_2 là các từ chiếu, còn a là một hằng. q là toán tử so sánh $=, ^, <, <=, >, >=$ được gọi là công thức nguyên tố.

Một công thức nguyên tố là một công thức.

Nếu F_1 và F_2 là các công thức thì $F_1 \vee F_2, F_1 \wedge F_2, F_1 \rightarrow F_2$ và $F_1 \leftrightarrow F_2$, cũng là các công thức.

Nếu F là một công thức và s là một biến tự do thì $\neg s F$ và $\exists s F$ cũng là các công thức.

Nếu F là công thức thì (F) cũng là công thức (công thức đặt trong ngoặc tròn cũng là công thức).

Định nghĩa 7.2.1:

Một câu hỏi trong ngôn ngữ tân từ có biến là bộ- n được biểu diễn như sau:

$\{ s \mid F \}$

Ở đây s là biến bộ – n , F là một công thức chỉ có một biến tự do là s .

Ví dụ 7.11:

Quan hệ BIÊN GIỚI (Nước, Tỉnh_TP): Danh sách các nước có biên giới giáp với Việt nam.

Phép toán quan hệ BIÊN GIỚI [Nước] được chuyển thành câu hỏi trong ngôn ngữ tân từ có biến là bộ:

$$\{ s \text{ [Nước] BIÊN GIỚI } s \}$$

Công thức an toàn:

Miền giá trị của công thức:

Định nghĩa 7.2.2:

Cho công thức F có sự hiện diện của các quan hệ Q_1, Q_2, \dots, Q_n và các hằng a_1, a_2, \dots, a_p . Miền giá trị của công thức F , ký hiệu là $DOM(F)$ hay $MGT(F)$, được định nghĩa là:

$DOM(F) = \{ (a_1, a_2, \dots, a_p) : \text{là tập giá trị của các thành phần của các bộ thuộc } Q_1, Q_2, \dots, Q_n \}$.

Ví dụ 7.12: Cho 2 quan hệ R và S như sau:

R	A	B	S	A	B
	1	2		3	4
	3	4		2	2
	2	5			

Công thức $F(s)$: $R_s S_s$.

Miền giá trị của công thức F là $DOM(F) = \{ 1, 2, 3, 4, 5 \}$

Nhận xét: $DOM(F) = DOM(F)$. Thật vậy:

$F = R_s R_s$. S vẫn biến thiên trên R và S .

Công thức an toàn:

Định nghĩa 7.2.3 :

Một công thức F được gọi là an toàn nếu nó thỏa mãn 3 điều kiện sau đây:

Nếu s là bộ- n thỏa: $F(s)$ là đúng thì mọi thành phần của s là phần tử của $DOM(F)$.

$(F(s) : \text{Đúng } P \text{ s } i \text{ } \forall \text{ } DOM(F); " i, s = (s_1, s_2, \dots, s_n)$

Với mỗi công thức con F' của F có dạng $s F'(s)$, nếu s thỏa $F'(s)$: Đúng, thì $s \forall \text{ } DOM(F')$.

Với mỗi công thức con F' của F có dạng $s F'(s)$, nếu có một thành phần nào đó của s không nằm trong $DOM(F')$ thì $F'(s)$ là đúng. Nghĩa là $s \in \text{DOM}(F) \Rightarrow F'(s)$: Đúng, hay $F'(s)$: Sai $\Rightarrow s \forall \text{ } DOM(F')$.

Ví dụ 7.13: Như ví dụ trong điểm (a): 2 quan hệ R và S như sau:

R	A	B	S	A	B
	1	2		3	4
	3	4		2	2
	2	5			

Các câu hỏi:

$c_1: \{ s \mid \frac{1}{2} R s S s \}$

$c_2: \{ s \mid \frac{1}{2} R s S s \}$

$F_1: R s S s$ không an toàn vì cho $s = \{ 3, 7 \}$, $s \in \text{DOM}(F)$ mà không thỏa $S \Rightarrow s$ thỏa F_1 .

$(s : F_1(s))$ là Đúng và $s \in \text{DOM}(F_1)$

$F_2 = R s S s$: là an toàn, nếu s thỏa $F_2 \Rightarrow R s$: là Đúng $\Rightarrow s \forall \text{ } R$ do đó mọi thành phần của $s \forall \text{ } \text{DOM}(F_2)$.

Chúng ta có:

() $F = R s F'(s)$ là an toàn, với F' là 1 công thức (không có công thức con có dạng $s u F(u)$ và $s u F(u)$).

() $s (F(s) G(s))$ với F là an toàn.

() $s (F(s) G(s))$ với G là một công thức.

$\Rightarrow (2)$.

: $\$ u F'(u)$, nếu u thỏa $F'(u)$: Đúng $\vdash u \in \text{DOM}(F')$, nếu u thỏa $F'(u)$: Đúng $\vdash u \in \text{DOM}(F')$.

$\$ u F'(u) \vdash \$ u F' \vdash \$ u F'(u)$,

" $u F'$),

" $u F'(u)$.

Theo (1):

" $u (F'(u) : \text{Đúng} \vdash u \in \text{Dom}(F'))$.

Theo (1):

" $u (F'(u) : \text{Đúng} \vdash u \in \text{Dom}(F'))$.

" $u(u \in \text{Dom}(F') \vdash F'(u) : \text{Đúng})$,

mà

$\text{DOM}(F') = \text{DOM}(F')$,

nên ta có:

" $u(u \in \text{DOM}(F') \vdash F'(u) : \text{Đúng})$

Ngôn ngữ tân từ (tiếp theo)

Ngôn ngữ tân từ có biến là miền giá trị

x là biến lấy giá trị trên miền thuộc tính của các quan hệ.

Quy tắc định nghĩa công thức:

Từ: là hằng hoặc biến

Công thức nguyên tố:

(i) $Q(t_1, t_2, \dots, t_n)$ t_i là các từ, Q là một quan hệ.

(ii) $t_1 \text{ q } a_1, t_2 \text{ q } a_2, \dots$ ở đây t_i là các từ và q là phép toán.

Công thức nguyên tố là một công thức.

F_1, F_2 là các công thức thì $F_1 \wedge F_2$ và $F_1 \vee F_2$ cũng là các công thức.

F là một công thức và t là một biến tự do thì $\neg t \wedge F$ và $\exists t F$ cũng là các công thức.

F là công thức thì (F) cũng là một công thức.

Trong một CSDL, câu hỏi bằng ngôn ngữ tân từ có dạng: $\{ (x_1, x_2, \dots, x_k) \mid F(x_1, x_2, \dots, x_k) \}$

Ở đây x_i ($i = 1, 2, \dots, k$) là các biến tự do của F , và F không có biến tự do nào khác.

Câu trả lời là tập các bộ giá trị (x_1, x_2, \dots, x_k) mà khi thay vào F thì F là đúng.

Công thức F là an toàn nếu nó thỏa các điều kiện sau:

$F(x_1, x_2, \dots, x_n)$ là Đúng $\forall x_i \in \text{DOM}(F)$ với $i = 1, 2, \dots, n$.

Công thức con $\exists u F'(u)$ nếu u thỏa $F'(u)$ là Sai

thì $\forall u \in \text{DOM}(F')$.

Ví dụ 7.3.1:

Cho một CSDL gồm các quan hệ sau:

VẬN_CHUYỂN (Khách_Hàng, Giá_Trị, Ga_Đến, #Toa, Mặt_Hàng, Khối_Lượng, N_Dỡ_Hàng)

TÀU (#Tàu, #Toa)

TOA (#Toa, Loại_Toa, TL_Rỗng, Khả_Năng_Chứa, Tình_Trạng, Ga)

HÀNG (Ga_Gốc, Ga_Đích, GA_TT, #T_Đường)

TUYẾN_ĐƯỜNG (#T_Đường, Hàng, Ga)

LỊCH_TRÌNH (#Tàu, #T_Đường, Ngày)

Câu hỏi 1: Cho danh sách các loại toa "Frigo" đang sẵn sàng ở ga "Tour" và có khả năng chứa trên 10 toa ?

Câu hỏi trên có thể được biểu diễn bằng ngôn ngữ đại số quan hệ như sau:

(TOA : (Loại_Toa = "Frigo" Khả_Năng_Chứa > 10 Ga = "Tour" Tình_Trạng = "Td") [#Toa],

Diễn tả bằng ngôn ngữ tân từ có biến là bộ như sau:

{ r [#Toa] | TOA r r[Loại_Toa] = "Frigo" r[Khả_Năng_Chứa] > 10 r[Ga] = "Tour" r[Tình_Trạng] = "Td" }

Và bằng ngôn ngữ tân từ có biến là miền giá trị như sau:

{ n | \$ x \$ c (TOA (n, "Frigo", x, c, "Td", "Tour") c > 10) }

Câu hỏi 2: Cho danh sách các loại toa của tàu số 4002 ?

Câu hỏi được biểu diễn bằng ngôn ngữ đại số quan hệ:

(((TÀU : #Tàu = 4002) [#Toa]) * TOA) [Loại_Toa]

Bằng ngôn ngữ tân từ có biến là bộ giá trị :

{ r [Loại_Toa] | \$ p (TOA r TÀU p r [#Toa] = p[#Toa] p[#Tàu] = 4002 }

Bằng ngôn ngữ tân từ có biến là miền giá trị chúng ta có công thức sau:

{ t | \$ u \$ p \$ c \$ e \$ g (TÀU (w, 4002) TOA (w, t, p, c, e, g)) }

Câu hỏi 3: Cho danh sách các con đường đi qua ga "Tour" ?

Câu hỏi được biểu diễn bằng ngôn ngữ đại số quan hệ:

$R1 \neg \text{TUYẾN_ĐƯỜNG} * (\# \text{Tuyến_đường1} = \# \text{Tuyến_đường2})$

Tuyến_đường2:

$R2 \neg (R1 : (\text{Ga} = \text{"Tour"} \text{ Hạng1} < \text{Hạng2}) [\# \text{Tuyến_đường1}]$

Bằng ngôn ngữ tân từ có biến là bộ giá trị :

$\{ r \mid [\# \text{Tuyến_Đường}] \mid \text{TUYẾN_ĐƯỜNG } r \ \$ \ p \ (\text{TUYẾN_ĐƯỜNG } p \ r \mid [\# \text{Tuyến_Đường}] = p[\# \text{Tuyến_Đường}] r[\text{Hạng}] < p[\text{Hạng}] r[\text{Ga}] = \text{"Tour"} \}$

Bằng ngôn ngữ tân từ có biến là miền giá trị chúng ta có công thức sau:

$\{ l \mid \$ r_1 \$ r_2 \$ g \ (\text{TUYẾN_ĐƯỜNG } (l, r_1, \text{"Tour"}) \text{ TUYẾN_ĐƯỜNG } (l, r_2, g) (r_2 > r_1)) \}$

Câu hỏi 4: Cho danh sách các ga mà toa tàu đi từ "Anger" đến "Bezier" ?

Chúng ta cần một lược đồ – công thức sau:

$x := \{ g \mid \$ \text{IHẠNG} (\text{"Anger"}, \text{"Bezier"}, g, l) \}$

Output (x)

While x "Bezier"

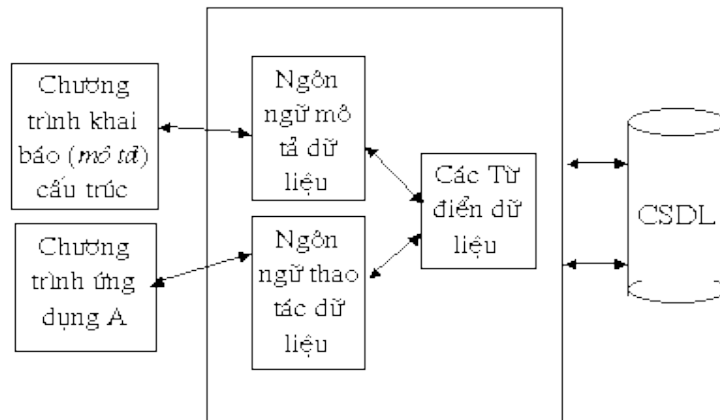
$X := \{ g \mid \$ \text{IHẠNG} (\text{"Anger"}, \text{"Bezier"}, g, l) \}$

Output (x)

End.

Sự tương đương của các ngôn ngữ thao tác CSDL:

Tính tương đương: Một câu hỏi được biểu diễn bằng một trong 3 ngôn ngữ thao tác CSDL: ngôn ngữ đại số quan hệ, ngôn ngữ tân từ có biến là bộ, ngôn ngữ tân từ có biến là miền giá trị là tương đương nhau:



Quy trình chứng minh như trên.

Một số định lý:

Định lý 7.1:

Nếu B là một biểu thức trong ngôn ngữ đại số quan hệ thì tồn tại 1 biểu thức an toàn trong ngôn ngữ tân từ có biến là n-bộ tương đương với biểu thức B.

Chứng minh: (Bằng phương pháp quy nạp theo số phép toán hiện diện trong B).

Trường hợp B không có phép toán:

B là một bảng các hằng $\{ t_1, t_2, \dots, t_n \}$ hoặc B là một quan hệ.

Nếu B là một quan hệ Q, thì chúng ta có biểu thức an toàn có dạng $\{ t \mid Q \}$.

Nếu B là một bảng hằng $\{ t_1, t_2, \dots, t_n \}$, thì chúng ta có công thức : $B' = \{ t \mid t = t_1 \ t = t_2 \ t = t_3 \ \dots \ t = t_n \}$.

Trong đó t_i là dạng viết tắt của $t^1 = t_i^1, t^2 = t_i^2, \dots, t^k = t_i^k$; ở đây t^j là thành phần thứ j của bộ t.

Vì t_1, t_2, \dots, t_n là các bộ hoặc các hằng nên miền giá trị $MGT(B) = \{ t_1, t_2, \dots, t_n \}$ và $DOM(B') = \{ t_1, t_2, \dots, t_n \}$; hay công thức : $\{ t \mid t = t_1 \ t = t_2 \ t = t_3 \ \dots \ t = t_n \}$ là tương đương với biểu thức B đã cho.

Trường hợp B có chứa phép toán.

Giả sử B có dạng $B = B_1 \ q \ B_2$.

B_1, B_2 là các biểu thức quan hệ và q là phép toán quan hệ có số phép toán ít hơn B . Chúng ta cần tìm C , biểu thức an toàn tương đương với B .

Các phép toán quan hệ được xét là: Hợp (- *Union*), Hiệu (- *Minus*), Chiều ([] - *Project*), Tích Đề các (\times - *Cartesian*) và Chọn (:() - *Selection*).

Phân tích phép toán:

Trường hợp 1: Phép Hợp (- *Union*). Nếu $B = B_1 \cup B_2$ thì ta có:

$$B_1 \cup \{ t \mid C_1(t) \}$$

$$B_2 \cup \{ t \mid C_2(t) \}$$

$C_1(t)$ và $C_2(t)$ là các công thức an toàn.

Xét công thức $C(t) = C_1(t) \cup C_2(t)$. Chúng ta cần chứng minh $C(t)$ là công thức an toàn. Xét công thức trong ngôn ngữ tân từ $B' = \{ t \mid C(t) \}$ với $C(t)$ là công thức an toàn, tức là với mọi t là n -bộ $t \in \text{DOM}(B')$ để $C(t)$ là đúng. Khi đó B tương đương với B' , ký hiệu: $B \equiv B'$.

Nhận xét: $\text{DOM}(C_1) \cup \text{DOM}(C_2) \subseteq \text{DOM}(C)$.

Xét điều kiện 1 của định nghĩa công thức an toàn:

Nếu t_0 thỏa $C(t) = C_1(t) \cup C_2(t)$ thì:

Hoặc t_0 thỏa $C_1(t) \in t_0 \in \text{DOM}(C_1)$, bởi vì C_1 là công thức an toàn;

Hoặc t_0 thỏa $C_2(t) \in t_0 \in \text{DOM}(C_2)$, bởi vì C_2 là công thức an toàn.

\Rightarrow Như vậy điều kiện 1 là thỏa.

Xét điều kiện 2 & 3:

Nếu có công thức $\exists u F(u)$ hoặc $\forall u F(u)$ là các công thức con của C thì chúng nằm gọn trong C_1 hoặc C_2 .

Do C_1 và C_2 là các công thức an toàn nên các công thức con cũng thỏa các điều kiện an toàn.

Do u_0 thỏa $\exists u F(u) \in u_0 \in \text{DOM}(F)$.

Do u_0 thỏa $\forall u F(u) \in u_0 \in \text{DOM}(F)$.

Bởi vì $C(t) = C_1(t) \cdot C_2(t)$, do đó $B \circ B'$ theo định nghĩa của phép hợp (\circ).

Trường hợp 2: Phép Hiệu (-Minus). Nếu $B = B_1 - B_2$ thì ta có:

$$B' \circ \{ t \mid C_1(t) \cdot C_2(t) \}$$

$$C(t) = C_1(t) \cdot C_2(t).$$

Xét điều kiện 1 của định nghĩa công thức an toàn:

(** Lưu ý: $\text{DOM}(C) = \text{DOM}(C_1) \cup \text{DOM}(C_2)$).

Nếu t_0 thỏa $C(t) \neq t_0 \nexists \text{DOM}(C_1) \cap \text{DOM}(C)$

Nghĩa là với t là một n -bộ thỏa $C(t)$ thì t thuộc vào miền giá trị của công thức $C(t)$.
Điều kiện 1 thỏa.

Điều kiện 2:

Nếu có công thức $\$ u F(u)$ là công thức con của C thì :

$\$ u F(u)$ là công thức con C_1 . C_1 là an toàn nên $u \neq t_0$ thỏa $F(u)$ là đúng $\neq u \neq t_0 \nexists \text{DOM}(F)$.
Và

$\$ u F(u)$ là công thức con C_2 . C_2 là an toàn nên $u \neq t_0$ thỏa $F(u)$ là sai $\neq u \neq t_0 \nexists \text{DOM}(F)$.
Hay $u \neq t_0$ thỏa $F(u)$ là đúng $\neq u \neq t_0 \nexists \text{DOM}(F)$.

Vậy điều kiện 2 thỏa mãn.

Điều kiện 3: (hoàn toàn tương tự như trên)

Nếu có công thức $" u F(u)$ là công thức con của C thì :

$" u F(u)$ là công thức con C_1 . C_1 là an toàn nên $u \neq t_0$ thỏa $F(u)$ là đúng $\neq u \neq t_0 \nexists \text{DOM}(F)$.
Và

$" u F(u)$ là công thức con C_2 . C_2 là an toàn nên $u \neq t_0$ thỏa $F(u)$ là sai $\neq u \neq t_0 \nexists \text{DOM}(F)$.
Hay $u \neq t_0$ thỏa $F(u)$ là đúng $\neq u \neq t_0 \nexists \text{DOM}(F)$.

Vậy điều kiện 3 thỏa mãn.

Trường hợp 3: Phép chiếu ([] - Project)

Biểu thức quan hệ có dạng: $B = B_1[X]$. Ở đây, X là tập các thuộc tính.

B_1 có biểu thức tương đương là $B' = \{ t \mid C_1(t) \}$ và $C_1(t)$ là một công thức an toàn.

Xét $B' = \{ u \mid \exists t (C(t) \wedge u = t[X]) \}$

A là tập các hằng và các giá trị của thuộc tính X của các quan hệ có mặt trong biểu thức đại số quan hệ B_1 . Khi đó:

$\text{DOM}(C_1) = \text{MGT}(X) \cap A$.

Đặt $E = \text{MGT}(X) - A$.

$\text{DOM}(C) = \text{DOM}(C_1) - E$.

Về điều kiện 1:

u_0 thỏa C , $\exists t_0 (C_1(t_0) \wedge u = t_0[X])$ là đúng. $\exists C_1(t_0)$: là đúng.

C_1 là an toàn $\exists t_0 \forall \text{DOM}(C_1)$.

$t_0^j = u_0^j$, thành phần thứ j của t_0 và u_0 .

$u_0^j \in \text{DOM}(C_1) - E$, hay $u_0 \in \text{DOM}(C_1)$

Như vậy C là công thức an toàn.

Điều kiện 2 & 3:

$\exists u F(u)$ hoặc $\forall u F(u)$ là các công thức con của C . Suy ra chúng là các công thức con của C_1 .

Như vậy $C(t)$ là công thức an toàn và chúng ta có: $B \circ B'$.

Trường hợp 4: Phép tích Đề các (x - Cartesian).

Biểu thức B có dạng $B = B_1 \times B_2$.

B_1 có công thức an toàn C_1 tương đương và

B_2 có công thức an toàn C_2 tương đương.

Chúng ta xây dựng biểu thức trong ngôn ngữ tân từ có biến *bộ-n* như sau:

$B' = \{ u \mid \exists t \exists v (C_1(t) \wedge C_2(v) \wedge u[X] = tv[Y]) \}$

Ở đây X và Y là tập các thuộc tính của B_1 và B_2 . Chúng ta cần chứng minh công thức là an toàn

$\models B \circ B'$.

Xét điều kiện 1:

u_0 thỏa C . \models

$u_0[X] = t_0$ thỏa C_1 , C_1 là công thức an toàn $\models t_0 \forall Y \text{ DOM}(C_1)$.

$u_0[Y] = v_0$ thỏa C_2 , C_2 là công thức an toàn $\models v_0 \forall Y \text{ DOM}(C_2)$.

$\models u_0 \forall Y \text{ DOM}(C_1) \text{ DOM}(C_2) \text{ Í } \text{DOM}(C) \models \text{Điều kiện 1 thỏa.}$

Điều kiện 2& 3: (không có)

Trường hợp 5: Đối với phép chọn (:() - Selection)

Biểu thức trong ngôn ngữ đại số quan hệ có dạng:

$B = B_1 : (E)$

(*phép chọn theo điều kiện E*). B_1 có biểu thức an toàn là C_1 . Biểu thức trong ngôn ngữ tân từ có biến *bộ-n* là:

$B' = \{ t \mid C_1(t) E' \}$

Ở đây E' được suy từ E bằng cách lấy các toán hạng biểu diễn bởi thuộc tính X là $t[X]$.

$C(t) = C_1(t) E'$. Cần chứng minh C là một công thức an toàn.

$\text{DOM}(C) = \text{DOM}(C_1) \cup \text{Tập hằng}(E')$.

t_0 thỏa C_1 là đúng $\models t_0 \forall Y \text{ DOM}(C_1)$.

\models điều kiện 1 là thỏa.

Điều kiện 2 & 3 (không có) Như vậy $C(t) = C_1(t) E'$ là một công thức an toàn. Ta có $B \circ B'$.

Kết luận: Một câu hỏi được biểu diễn bằng ngôn ngữ đại số quan hệ luôn luôn tìm được biểu thức tương đương với nó biểu diễn bằng ngôn ngữ tân từ có biến là bộ-n.

Ví dụ 7.3.1:

Cho 2 quan hệ R(A,B) và S(C,D).

Biểu thức trong ngôn ngữ đại số quan hệ là:

$((R \times S) : (B = C)) [A, B, D].$

Biểu diễn trong ngôn ngữ tân từ có biến là *bộ-n*:

Với $R \times S$:

$\{ t \mid \exists u \exists v (R \cup S \vee (t[A] = u[A]) (t[B] = u[B]) t[C] = v[C]) t[D] = v[D]) \}$

Sâu hơn nữa tới phép chọn $(: (B=C))$, và rồi phép chiếu :

$\{ w \mid \exists t (\exists u \exists v (R \cup S \vee (t[A] = u[A]) (t[B] = u[B]) t[C] = v[C]) t[D] = v[D]) (t[B] = t[C]) (w[A] = t[A]) (w[C] = t[C]) (w[D] = t[D])) \}$

Đơn giản biểu thức chúng ta thu được :

$\{ w \mid \exists u \exists v (R \cup S \vee (w[A] = u[A]) (w[C] = v[C]) w[D] = v[D]) (u[B] = v[C]) \}$

đó là biểu thức tương đương được biểu diễn bằng ngôn ngữ tân từ có biến là *bộ-n* của biểu thức trong ngôn ngữ đại số quan hệ nêu trên.

Định lý 7.2:

Ứng với mỗi biểu thức an toàn trong ngôn ngữ tân từ có biến là bộ-n ta có một biểu thức an toàn tương đương trong ngôn ngữ tân từ có biến là miền giá trị.

Quá trình biến đổi câu hỏi tron ngôn ngữ tân từ có biến là *bộ-n* sang câu hỏi tron ngôn ngữ tân từ có biến là miền giá trị:

Cho câu hỏi dạng $\{ t \mid C(t) \}$, Ở đây t là một *bộ-n* có k bậc tự do. t_0 phát sinh ra k biến miền giá trị tự do là $x_1, x_2, x_3, \dots, x_k$.

Trong công thức $C(t)$, mỗi biểu thức $Q \ t$ mà Q là một quan hệ, thì được đổi thành $Q(x_1, x_2, x_3, \dots, x_k)$.

vMỗi biểu thức có sự hiện diện của $t[x_i]$ thì được đổi thành x_i .

Trong các biểu thức con có dạng $\$ u G(u)$ hay $" u G(u)$ được biến đổi như sau:

Biến u được đổi thành y_1, y_2, \dots, y_m , Ở đây m là bậc của u .

$R(u)$ được đổi thành $R(y_1, y_2, \dots, y_m)$.

$u[y_i]$ được đổi thành y_i .

$\$ u$ được đổi thành $\$ y_1, \$ y_2, \$ y_3, \dots, \$ y_m$,

$" u$ được đổi thành $" y_1, " y_2, " y_3, \dots, " y_m$,

Tối ưu hóa câu hỏi

Đặt vấn đề

Các ngôn ngữ bậc cao nói chung và ngôn ngữ con dữ liệu nói riêng khi thực hiện trong máy đều mất rất nhiều thời gian. Do đó, trước khi thực hiện các câu lệnh thuộc các ngôn ngữ đó cần thiết phải biến đổi hợp lý về dạng tương đương, tức là dạng cho cùng một kết quả, để giảm thời gian tính toán. Việc làm đó được gọi là "tối ưu hóa" (*Optimiztation*). Việc tối ưu hóa không nhất thiết phải được tối ưu trên mọi khả năng có thể có của các cách cài đặt các câu hỏi. Có thể thỏa hiệp giữa thuật giải phức tạp, hoặc / và tốn kém không gian lưu trữ với việc tiết kiệm thời gian xử lý. Chúng ta biết rằng, CSDL có thể là rất lớn, hàng chục ngàn (như hệ quản lý cán bộ - công chức) hay hàng trăm ngàn (như việc quản lý các đối tượng chính sách - có công Cách mạng), hoặc hàng triệu bản ghi (như CSDL quản lý dân cư thành phố Hồ Chí Minh) v.v... Những bài toán này, nếu mỗi thao tác trên một bản ghi chúng ta tiết kiệm được q thời gian thì một câu truy vấn trên 1 bảng đã có thể tiết kiệm được $10^n * q$ thời gian, ở đây n là số bản ghi của bảng. Chúng ta hãy thử tưởng tượng, với bảng lưu trữ các bản ghi có độ dài cố định về nhân khẩu, trong vòng 1 giây máy có thể xử lý cực nhanh 1000 bản ghi, thì để xử lý 4.850.000 nhân khẩu của bảng, máy phải tốn 4.850 giây! Quả là một thời gian khó có thể chấp nhận cho việc chờ đợi nhận kết quả của một câu lệnh truy vấn !

Nói chung, trong việc tối ưu hóa xử lý thông tin, người ta ưu tiên việc tối ưu hóa về thời gian hơn so với việc tối ưu hóa lưu trữ dữ liệu. Trong một số trường hợp, người ta còn phải "hy sinh" cả dạng chuẩn (*Normal Form*) để đạt tốc độ xử lý nhanh hơn. Trong ví dụ về hệ CSDL quản lý CBVC nêu tại mục 5.3 trong chương V, (ví dụ 5.3.4), mỗi khi cần xem xét đến hệ số lương của một CBVC người ta đều phải tham khảo qua bảng NGẠCH-BẬC-LƯƠNG thông qua phép kết tự nhiên với bảng CBVC trên các thuộc tính **Ngạch** và **Bậc**. Nếu trong danh sách CBVC chúng ta bổ sung cột **Hệ-số-lương** nữa thì dẫn tới việc trùng lặp (và dư thừa) thông tin - nguy cơ dẫn đến việc mất tính nhất quán dữ liệu - đồng thời không đảm bảo dạng chuẩn tốt cho bảng này và buộc phải bổ sung RBTV :

CBVC [**Ngạch,Bậc, Hệ-số-lương**] Í NGẠCH-BẬC-LƯƠNG [**Ngạch,Bậc, Hệ-số-lương**].

Bộ nhớ thứ cấp (băng từ, đĩa từ ...) của máy tính giờ đây không còn là vấn đề làm mọi người phải bận tâm khi xây dựng các hệ CSDL lớn. Dung lượng một đĩa cứng đã đạt tới hàng chục Giga bytes. Tuy nhiên vẫn có thể tồn tại nguy cơ thiếu không gian tính toán cho máy nếu không lưu tâm đến việc tối ưu hóa các câu hỏi trước khi đưa vào máy và yêu cầu máy thi hành. Một nguyên nhân dẫn tới điều này là việc thực hiện các phép kết nối và phép tích Đề-các của các quan hệ. Chúng ta đã biết, tích Đề-các (đã trình bày

tại điểm 5.2.4, mục 5.2,) và các phép kết (*q-Join, Natural Join, Equi Join, Outer Join* - đã trình bày tại điểm 5.3.3 mục 5.3 và mục 5.4 của Chương V) đòi hỏi khá nhiều không gian lưu trữ và thời gian xử lý.

Cho $R(A_1, A_2, \dots, A_n)$ và $S(B_1, B_2, \dots, B_m)$ là hai quan hệ định nghĩa trên 2 tập thuộc tính $\{A_1, A_2, \dots, A_n\}$ và $\{B_1, B_2, \dots, B_m\}$. Nếu R có N_1 bộ giá trị và S có N_2 bộ giá trị thì tích Đề-các sẽ cho kết quả là một bảng mới gồm $m + n$ thuộc tính với $N_1 * N_2$ bộ giá trị. Chỉ cần mỗi quan hệ có 10 thuộc tính chứa số lượng bộ giá trị khá khiêm tốn, 1000 bộ, thì tích Đề-các đã tạo ra bảng trung gian có số thuộc tính là 20 và số lượng bộ giá trị là 1.000.000. Nếu mỗi field chỉ rộng 5 bytes thôi (mà trên thực tế thì rất ít có những quan hệ như vậy) thì quan hệ trung gian này đã "ngốn" mất 1/10 Giga bytes ! Đương nhiên là, thời gian truy nhập CSDL và truy xuất đĩa từ để tạo ra ngần ấy bản ghi cũng không phải là ít. Sự bùng nổ số lượng và "bành trướng" kích thước (chiều dài) bản ghi này đặt chúng ta vào việc phải nghiên cứu để tối ưu câu hỏi trước khi đưa vào máy tính toán.

Chương này chủ yếu trình bày một vài phương pháp tối ưu hóa các biểu thức quan hệ, đặc biệt là xử lý biểu thức có liên quan tới phép kết nối và tích Đề-các. Sau đó sẽ trình bày chi tiết một phương pháp tối ưu hóa cho một lớp phổ cập các biểu thức quan hệ. Nguồn tài liệu tham khảo chủ yếu dựa trên các lý thuyết và ví dụ minh họa trong chương VI của PGS.PTS.Lê Tiến Vương [8].

Các nguyên tắc tổng quát để tối ưu hóa một câu hỏi.

Chúng ta hãy xét một ví dụ đơn giản sau đây :

Cho hai quan hệ $R(A,B)$ với n bản ghi và $S(C,D)$ với m bản ghi. Tích Đề-các của R và S là một quan hệ $Q(A,B,C,D)$ có $n * m$ bản ghi (*Xem 5.2.4. Phép tích Đề-các của 2 quan hệ, mục 5,2, Chương V*). Chúng ta có câu hỏi "Lấy giá trị của thuộc tính A sao cho $B=C$ và $D=50$ ". Câu hỏi được viết lại dưới dạng ngôn ngữ đại số quan hệ như sau:

$$((R(A,B) \times S(C,D)) : (B=C \ D=50)) [A]$$

Nếu đưa phép chọn $D=50$ vào bên trong phép tích Đề-các sẽ được:

$$((R(A,B) \times (S(C,D) : (D=50))) : (B=C)) [A]$$

và sau đó chuyển phép chọn $B=C$ của tích Đề-các thành phép "kết nối bằng" chúng ta thu được:

Rõ ràng, phép tính cuối cùng sẽ đỡ tốn kém thời gian hơn rất nhiều.

-*Cụ thể là:* Chỉ chọn trên quan hệ S những bộ có giá trị $D=50$ thì số bộ lấy ra sẽ ít hơn toàn bộ số bộ của cả quan hệ trên S . Số bộ được chọn ra xong từ S mới đem kết nối với quan hệ R . Phép kết nối này chỉ chọn ra bộ nào thuộc R mà có giá trị tại B là bằng bộ có giá trị tại C thuộc S mới được lấy ra để kết lại với nhau. Điều đó hoàn toàn nhanh hơn là lấy tích Đề-các của $R \times S$ rồi mới chọn trong kết quả những bộ có giá trị tại B bằng giá trị tại C .

Việc biến đổi câu hỏi thành câu hỏi tương đương như ví dụ nêu trên là một minh họa cho việc giảm bớt thời gian trả lời câu hỏi bằng cách giảm bớt số lần cần truy nhập tới bộ nhớ thứ cấp dựa trên nguyên tắc thực hiện phép chọn càng sớm càng tốt. Trình tự thực hiện các phép tính sẽ đóng một vai trò quan trọng quá trình tổ chức câu hỏi.

J. D. Ullman [5] trong các kết quả nghiên cứu công bố lần đầu tiên của mình đã trình bày 6 chiến lược tổng quan cho việc tối ưu hóa câu hỏi như sau:

-. *Thực hiện phép chọn càng sớm càng tốt.*

Biến đổi câu hỏi để đưa phép chọn vào thực hiện trước nhằm làm giảm bớt kích cỡ của kết quả trung gian và do vậy chi phí phải trả cho việc truy nhập bộ nhớ thứ cấp cũng như lưu trữ của bộ nhớ chính sẽ nhỏ đi.

-*Tổ hợp những phép chọn xác định với phép tích Đề-các thành phép kết nối.*

Như đã biết, phép kết nối, đặc biệt là phép kết nối bằng (*Equi Join*) có thể được thực hiện ít tốn kém hơn nhiều so với phép tích Đề-các trên cùng các quan hệ. Nếu kết quả của tích Đề-các $R \times S$ là đối số của phép chọn và phép chọn liên quan tới các phép so sánh giữa các thuộc tính của R và S thì rõ ràng phép tích Đề-các là phép kết nối.

-*Tổ hợp dãy các phép toán quan hệ một ngôi như các phép chọn và phép chiếu.*

Một dãy các phép một ngôi như phép chọn hoặc phép chiếu mà kết quả của chúng phụ thuộc vào các bộ của một quan hệ độc lập thì có thể nhóm các phép đó lại.

-*Tìm các biểu thức con chung trong một biểu thức.*

Nếu kết quả của một biểu thức con chung (tức là biểu thức xuất hiện nhiều hơn một lần) là một quan hệ không lớn và nó có thể được đọc từ bộ nhớ thứ cấp với ít thời gian thì nên tính toán trước biểu thức đó chỉ một lần. Nếu biểu thức con chung có liên quan tới một phép kết nối thì trong trường hợp tổng quát không thể thay đổi được nó bằng cách "đẩy" phép chọn vào trong.

Điều đáng quan tâm là, các biểu thức con chung có tần số xuất hiện lớn thường được biểu diễn trong các VIEW (khung nhìn) của người sử dụng, bởi vì, để thực hiện các câu hỏi đó cần phải thay thế nó bằng một biểu thức cố định cho VIEW.

- *Tiền xử lý các quan hệ / bảng (Table Preprocessing).*

Có hai vấn đề quan trọng cần xử lý trước cho các quan hệ là sắp xếp trước các bộ giá trị theo thứ tự vật lý và sắp xếp logic - tức là thiết lập các bảng chỉ mục (*Index*) cho các bản ghi. Khi đó việc thực hiện các phép toán có liên quan tới hai quan hệ (các phép toán hai ngôi) sẽ nhanh hơn rất nhiều.

- *Đánh giá trước khi thực hiện tính toán.*

Mỗi khi cần chọn trình tự thực hiện các phép toán trong biểu thức, hoặc chọn một trong hai đối số của một phép hai ngôi, thì cần tính toán xem chi phí (*Cost*) thực hiện các phép tính đó (thường tính theo số phép toán, thời gian, hoặc/và dung lượng bộ nhớ cần thiết so với kích thước của các quan hệ, từ đó xác định được chi phí tổng thể phải trả cho các cách khác nhau khi thực hiện các câu hỏi).

Dựa vào các nguyên tắc nêu trên, chúng ta sẽ biến đổi câu truy vấn thành câu hỏi tương đương tối ưu hơn, để việc thực hiện có chi phí xử lý ít hơn. Nhưng trước khi có thể "tối ưu hóa" các biểu thức, cần làm rõ khái niệm khi nào thì hai biểu thức được gọi là tương đương. Trong phần sau sẽ cho biết một cách hình thức khái niệm tương đương.

Biểu thức tương đương.

Trước hết, xem xét lại khái niệm về định nghĩa quan hệ đã được sử dụng trong các chương 2 và 3 để thấy sự tương đương của các quan hệ dựa theo định nghĩa. Với các định nghĩa khác nhau chúng cho các tính chất toán học cũng khác nhau. Nếu quan niệm quan hệ là một tập hợp các bộ (k - bộ) với k cố định, khi đó hai quan hệ là tương đương khi và chỉ khi chúng có cùng một tập các bộ. Với quan niệm quan hệ là tập các ánh xạ từ tập tên thuộc tính vào tập giá trị, thì khi đó hai quan hệ là bằng nhau nếu chúng có cùng một tập ánh xạ. Định nghĩa thứ nhất có thể biến đổi sang định nghĩa thứ hai bằng cách thay đổi tên thuộc tính thành tên các cột trong bảng và, ngược lại đổi từ định nghĩa thứ hai sang định nghĩa thứ nhất bằng cách cố định thứ tự cho các thuộc tính. Sau đây sẽ sử dụng định nghĩa thứ hai, nghĩa là, quan hệ là một tập ánh xạ từ tập thuộc tính vào tập các giá trị. Một ngôn ngữ truy vấn hiện hữu luôn luôn đòi hỏi phải biết tên của các cột trong một quan hệ. Do đó, tên các thuộc tính trong một quan hệ phải là một biến trong một biểu thức và cũng thể hiện ở kết quả của biểu thức.

+ Biểu thức trong ngôn ngữ đại số quan hệ có các hạng thức là *biến quan hệ (relation variables)* R_1, \dots, R_n ; các *quan hệ hằng (constant relation)*, được xác định như là một ánh xạ từ các k -bộ của các quan hệ (r_1, \dots, r_k) trong đó r_i là quan hệ trên lược đồ r_i và thay thế r_i vào R_i khi đánh giá biểu thức. Hai biểu thức E_1 và E_2 được gọi là *tương đương (Equivalent)*, viết tắt là $E_1 \equiv E_2$, nếu chúng biểu diễn cùng một ánh xạ, nghĩa là, nếu chúng ta thay thế cùng các quan hệ cho tên các lược đồ tương ứng ở hai biểu thức E_1 và E_2 , thì chúng sẽ cho ra cùng một kết quả.

Với định nghĩa tương đương này chúng ta có một phép chuyển dịch đại số thông thường sau đây:

Các quy tắc liên quan tới phép kết và tích Đề-các.

⊕ Quy tắc giao hoán của phép kết nối và tích Đề-các

Nếu E_1 và E_2 là hai biểu thức quan hệ và F là điều kiện trên các thuộc tính của E_1 và E_2 thì:

$$E_1 \bowtie_{E_2 \circ E_2} E_1$$

// Tính giao hoán của kết

$E_1 * E_2 \circ E_1 * E_2$ // Tính giao hoán của kết bằng

$E_1 \times E_2 \circ E_1 \times E_2$ // Tính giao hoán của tích Đề-các.

Chúng ta chứng minh quy tắc giao hoán của phép kết nối. Giả sử E_1 có các thuộc tính A_1, \dots, A_n ; E_2 có các thuộc tính B_1, \dots, B_m và các thuộc tính A, B không nhất thiết phải là phân biệt. Gọi r_1 và r_2 là hai quan hệ tương ứng của E_1 và E_2 . Giá trị của $E_1 \bowtie E_2$ là tập các ánh xạ a từ $A_1 \dots A_n, B_1 \dots B_m$ vào tập giá trị sao cho các ánh xạ m_1 và m_2 thỏa mãn điều kiện:

$$a[A_i] = m_1[A_i], i = 1, 2, \dots, n$$

$$a[B_i] = m_2[B_i], i = 1, 2, \dots, m$$

và điều kiện F là đúng khi thay $a[C]$ cho mỗi thuộc tính C trong F .

Nếu biểu diễn $E_2 \bowtie E_1$ cũng như trên, chúng ta dễ dàng nhận thấy hai phép kết trên cho chính xác cùng một tập kết quả. Do vậy hai biểu thức là tương đương.

Chú ý: Nếu quan niệm quan hệ là tập các bộ (có thứ tự thuộc tính cố định) thì phép q-kết, kết tự nhiên và tích Đề-các không thể giao hoán được vì thứ tự các thuộc tính trong quan hệ kết quả bị thay đổi.

- Quy tắc kết hợp của phép kết nối và tích Đề-các.

Nếu E_1, E_2 và E_3 là các biểu thức quan hệ: F_1, F_2 là điều kiện thì:

$$1. (E_1 \bowtie_{E_2} E_3) \bowtie_{E_1 E_2} (E_2 \bowtie_{E_3} E_1)$$

$$(E_1 * E_2) * E_3 \quad E_1 * (E_2 * E_3)$$

$$(E_1 * E_2) \times E_3 \quad E_1 \times (E_2 \times E_3)$$

Việc kiểm tra tính tương đương của các quy tắc trên khá dễ dàng.

Các quy tắc liên quan tới phép chọn và phép chiếu

Dãy các phép chiếu có thể tổ hợp lại thành một phép chiếu, biểu diễn theo các trường hợp sau:

Ž *Dãy các phép chiếu*

$$(E [B_1 B_2 \dots B_m]) [A_1 A_2 \dots A_n] \quad E [A_1 \dots A_n]$$

Ở đây, các thuộc tính A_1, \dots, A_n phải nằm trong tập các thuộc tính B_1, \dots, B_m . Ngữ nghĩa của việc biến đổi tương đương này là: Nếu thực hiện một phép chiếu biểu thức quan hệ E trên tập các thuộc tính B , rồi sau đó thực hiện tiếp phép chiếu trên tập con các thuộc tính $A \subseteq B$ của quan hệ vừa tìm được, thì kết quả của dãy phép chiếu này hoàn toàn tương đương với một phép chiếu biểu thức quan hệ E trên tập thuộc tính A .

Tương tự, dãy các phép chọn có thể tổ hợp thành một phép chọn để kiểm tra tất cả các điều kiện cùng một lúc và được biểu diễn như sau:

- *Dãy các phép chọn:*

$$(((E : (f_1)) : f_2) : \dots) : f_n \quad E : (f_1 \ f_2 \ \dots \ f_n)$$

Ngữ nghĩa: Việc lần lượt thực hiện các phép chọn trên quan hệ kết quả của một phép chọn trước đó đối với biểu thức quan hệ E là tương đương với việc chọn trên E các bộ giá trị thỏa mãn đồng thời tất cả các điều kiện chọn $f_1, f_2 \dots f_n$.

- *Giao hoán phép chọn và phép chiếu:*

$$(E [A_1 \dots A_n] : (f))(E : (f)) [A_1 \dots A_n]$$

Một cách tổng quát hơn, nếu điều kiện chọn f liên quan tới các thuộc tính B_1, \dots, B_m mà không nằm trong tập thuộc tính A_1, \dots, A_n thì:

$$(E [A_1 \dots A_n]) : (f) \quad (E [A_1 \dots A_n \ B_1 \dots B_m]) : (f) \quad [A_1 \dots A_n]$$

- *Giao hoán phép chọn và tích Đề-các:*

Nếu tất cả các thuộc tính của F là thuộc tính của E_1 thì:

$$(E_1 \times E_2) : (f) (E_1 : (f)) \times E_2$$

Từ đó dễ dàng suy ra rằng, nếu F có dạng $f = f_1 \text{ L } f_2$ trong đó f_1 chỉ liên quan tới các thuộc tính của E_1 ; f_2 chỉ liên quan tới các thuộc tính của E_2 , thì có thể sử dụng các luật CE và ‘ để có:

$$(E_1 \times E_2) : (f) (E_1 : (f_1)) \times (E_2 : (f_2))$$

Hơn nữa nếu f_1 chỉ liên quan tới các thuộc tính của E_1 , nhưng f_2 liên quan tới các thuộc tính của cả E_1 và E_2 thì:

$$(E_1 \times E_2) : (f) ((E_1 : (f_1)) \times E_2) : (f_2)$$

-Giao hoán phép chọn và một phép hợp:

Nếu có biểu thức $E = E_1 E_2$; có thể giả thiết thêm rằng, các thuộc tính của E_1 và E_2 có cùng tên như của E hoặc ít nhất mỗi thuộc tính của E là phù hợp với một thuộc tính duy nhất của E_1 và một thuộc tính duy nhất của E_2 . Khi đó:

$$(E_1 E_2) : (f) (E_1 : (f))(E_2 : (f))$$

Nếu tên các thuộc tính của E_1 và hoặc E_2 khác với tên thuộc tính của E thì trong f ở vế phải của công thức trên cần thay đổi để sử dụng tên cho phù hợp.

-Giao hoán phép chọn và một phép hiệu tập hợp

$$(E_1 - E_2) : (f) (E_1 : (f)) - (E_2 : (f))$$

Như đã nêu trong luật ‘, nếu tên các thuộc tính của E_1 và E_2 là khác nhau thì cần thay thế các thuộc tính trong f ở vế phải biểu thức tương đương tương ứng với E_1 . Chú ý rằng, phép chọn $(E_2 : (f))$ có thể là không cần thiết. Trong nhiều trường hợp, việc thực hiện phép chọn $(E_2 : (f))$ trước sẽ có hiệu quả hơn là tính toán trực tiếp với E_2 vì kích cỡ quan hệ lúc đó sẽ bé đi rất nhiều.

Các quy tắc nêu trên nói chung là đẩy phép chọn xuống trước phép kết nối vì phép kết nối thường thực hiện lâu như phép tích Đề Các. Quy tắc đẩy phép chọn xuống trước phép kết nối suy ra từ quy tắc và ‘. Quy tắc đẩy phép chiếu xuống trước phép tích Đề Các hoặc phép hợp cũng tương tự như quy tắc ‘ và ‘. Chú ý là không có phương pháp tổng quát cho việc đẩy phép chiếu xuống trước phép hiệu các tập hợp.

-Giao hoán một phép chiếu với một phép tích Đề-các:

Gọi E_1, E_2 là hai biểu thức quan hệ, $A_1 \dots A_n$ là tập các thuộc tính trong đó $B_1, \dots B_m$ là các thuộc tính của E_1 , các thuộc tính còn lại C_1, \dots, C_k thuộc E_2 . Khi đó:

$$(E_1 \times E_2) [A_1 \dots A_n] E_1 [B_1 \dots B_m] \times E_2 [C_1 \dots C_k]$$

-*Giao hoán một phép chiếu với một phép hợp:*

$$(E_1 E_2) [A_1 \dots A_n] E_1[A_1 \dots A_n] E_2[A_1 \dots A_n]$$

Như đã nêu trong luật ' , nếu tên các thuộc tính của E_1 và / hoặc E_2 là khác với các thuộc tính trong $E_1 E_2$ thì cần thay $A_1 \dots A_n$ bên vế phải bằng các tên phù hợp.

Ví dụ về một thuật toán tối ưu hóa biểu thức quan hệ.

Tới đây có thể áp dụng các quy tắc nêu trong mục 8.2 để có thể tối ưu hóa các biểu thức quan hệ. Biểu thức "tối ưu" kết quả phải tuân theo các nguyên tắc đã nêu ở phần 8.1 mặc dù rằng các nguyên tắc đó không có nghĩa là bảo đảm để tối ưu cho mọi trường hợp tương đương.

Lưu ý rằng, luôn luôn đẩy phép chọn và phép chiếu xuống mức càng sâu càng tốt trong cây biểu diễn biểu thức quan hệ nhằm tạo nên một dãy các phép chọn cũng như phép chiếu để từ đó có thể tổ chức thành một phép chọn theo sau một phép chiếu. Nhóm các phép chọn và phép chiếu lại trong một nhóm để thực hiện trước các phép tính hai ngôi như phép hợp, tích Đề-các, hiệu tập hợp v.v...

Có một số trường hợp đặc biệt xảy ra khi một phép tính hai ngôi có các hạng thức chứa phép chọn và / hoặc phép chiếu được áp dụng đối với lá của cây biểu diễn biểu thức. Khi đó cần xem xét cẩn thận tác động của phép tính hai ngôi vì một số trường hợp cần phải liên kết phép chọn hoặc phép chiếu với phép hai ngôi đó.

Kết quả đầu ra (*Output*) của thuật toán là một chương trình bao gồm các bước như sau:

-Áp dụng của một phép chọn hoặc một phép chiếu đơn giản.

-Áp dụng của một phép chọn và một phép chiếu hoặc

- Áp dụng của một tích Đề-các, phép hợp hoặc phép hiệu tập hợp cho hai hạng thức mà trước đó các phép chọn hoặc các phép chiếu đã được áp dụng cho một hoặc cả hai hạng thức.

Hãy xét một CSDL quản lý thư viện bao gồm các quan hệ sau đây:

- **SÁCH (Tên-sách, Tác-giả, Nhà_XB, Mã-sách)**: là quan hệ về các loại sách trong thư viện.

- **NHÀ-XUẤT-BẢN (Nhà_XB, Địa-chỉ, Thành-phố)**: quan hệ về nhà xuất bản.

- **ĐỘC-GIẢ (Tên-ĐG, Địa-chỉ-ĐG, Tphố-ĐG, Số-thẻ)** : quan hệ về độc giả

- **MƯỢN-SÁCH (Số-thẻ, Mã-sách, Ngày-mượn)**: quan hệ sổ theo dõi mượn.

Để lưu trữ thông tin về sách có thể giả thiết thêm rằng có một khung nhìn (VIEW) theo dõi các sách được mượn, TD-MƯỢN, bao gồm một số thông tin bổ sung về sách được mượn, là kết quả của kết nối tự nhiên của quan hệ SÁCH, ĐỘC-GIẢ và MƯỢN-SÁCH, chẳng hạn được xác định qua biểu thức quan hệ:

$((\text{SÁCH} \times \text{ĐỘC-GIẢ} \times \text{MƯỢN-SÁCH}) : (f)) [\{S\}]$

Ở đây:

$f ::= (\text{ĐỘC-GIẢ.Số-thẻ} = \text{MƯỢN-SÁCH.Số-thẻ}) (\text{SÁCH.Mã-sách} = \text{MƯỢN-SÁCH.Mã-sách}).$

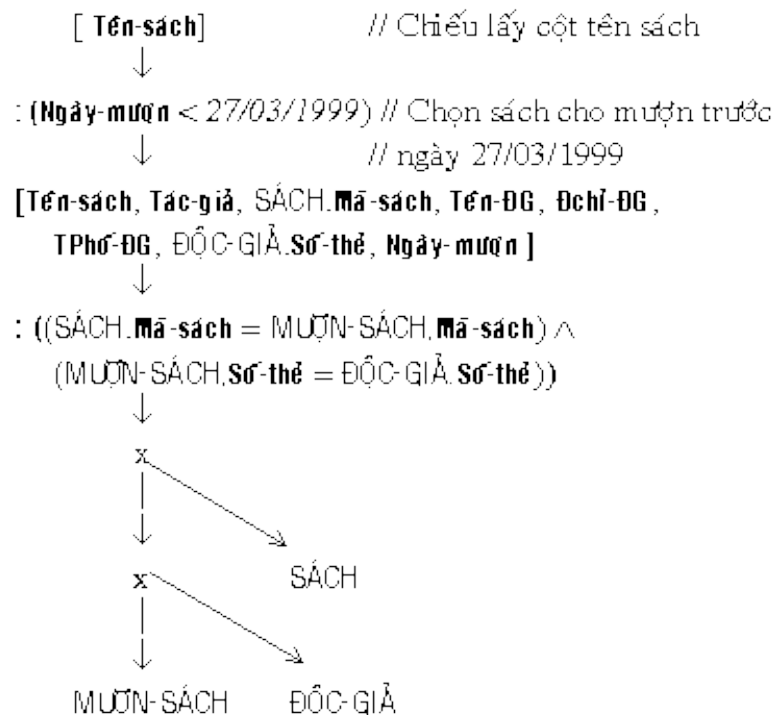
và S là tập các thuộc tính:

$S = \{ \text{Tên-sách, Tác-giả, Nhà_XB, SÁCH.Mã-sách, Tên_ĐG, Địa-chỉ-ĐG, Tphố-ĐG, ĐỘC-GIẢ.Số-thẻ, Ngày-mượn} \}.$

-Câu hỏi: Cho danh sách những cuốn sách đã cho mượn trước ngày 27/03/1999. Biểu thức quan hệ được viết như sau:

$(\text{TD-MƯỢN} : (\text{Ngày-mượn} < 27/03/1999)) [\text{Tên-sách}]$

Hình cây của biểu thức trên được biểu diễn bằng hình 8.1. Xin lưu ý là, các phép toán nằm ở phía dưới là các phép toán được thực hiện trước các phép toán ở phía trên của cây.



Hình 8.1: Biểu diễn cây của biểu thức hỏi

Thay thế các giá trị f và S vào biểu thức hỏi có được cây biểu diễn của biểu thức quan hệ như trong hình 8.1

-*Bước thứ nhất* của tối ưu là tách phép chọn f thành hai phép chọn với điều kiện:

$$\text{SÁCH.Mã-sách} = \text{MUỖN-SÁCH.Mã-sách}$$

và

$$\text{MUỖN-SÁCH.Số-thẻ} = \text{ĐỘC-GIẢ.Số-thẻ}$$

Bây giờ chúng ta có 3 phép chọn. Cần "đẩy" chúng xuống mức thấp hơn chừng nào còn có thể được.

Phép chọn với điều kiện **Ngày-mượn** < 27/03/1999 được đẩy xuống dưới phép chiếu và hai phép chọn kia bằng cách áp dụng các quy tắc (hoặc luật) và . Phép chọn đầu được áp dụng cho tích Đề-các ((MUỖN-SÁCH x ĐỘC-GIẢ) x SÁCH). Vì thuộc tính **Ngày-mượn** trong phép chọn chỉ có ở quan hệ MUỖN-SÁCH nên có thể thay thế:

$$((\text{MUỖN-SÁCH} \times \text{ĐỘC-GIẢ}) \times \text{SÁCH}) : (\text{Ngày-mượn} < 27/03/1999)$$

bằng biểu thức:

$((\text{MUỖN-SÁCH} \times \text{ĐỘC-GIẢ}) : (\text{Ngày-muỗn} < 27/03/1999)) \times \text{SÁCH}$

và tiếp tục đẩy xuống nữa, cuối cùng ta được biểu thức:

$((((\text{MUỖN-SÁCH} : (\text{Ngày-muỗn} < 27/03/1999)) \times \text{ĐỘC-GIẢ}) \times \text{SÁCH})$

Như vậy đã đây được phép chọn theo ngày mượn sách này xuống sâu như có thể.

Bây giờ tiếp tục đây phép chọn với điều kiện

Như vậy đã đây được phép chọn theo ngày mượn sách này xuống sâu như có thể.

Bây giờ tiếp tục đây phép chọn với điều kiện $\text{SÁCH.Mã-sách} = \text{MUỖN-SÁCH.Mã-sách}$. xuống mức thấp nhất nếu có thể. Không thể đây phép chọn này xuống dưới tích Đề-các vì nó liên quan tới một thuộc tính của quan hệ SÁCH và một thuộc tính thuộc quan hệ MUỖN-SÁCH. Do vậy phép chọn:

$: \text{MUỖN-SÁCH.Số-thẻ} = \text{ĐỘC-GIẢ.Số-thẻ}$

Số-thẻ

có thể đây xuống để áp dụng cho tích Đề-các:

$(\text{MUỖN-SÁCH} \times \text{ĐỘC-GIẢ}) : (: (\text{Ngày-muỗn} < 27/03/1999))$

Chú ý rằng MUỖN-SÁCH.Số-thẻ là tên một thuộc tính của phép chọn:

$\text{MUỖN-SÁCH} : (\text{Ngày-muỗn} < 27/03/1999).$

-Bước tiếp theo: Tổ hợp hai phép chiếu thành một phép chiếu là $[\text{Tên-sách}]$ nhờ luật Ỗ. Kết quả được cho như trong hình 8.2. Sau đó áp dụng quy tắc mở rộng thay thế:

$: (\text{MUỖN-SÁCH.Số-thẻ} = \text{ĐỘC-GIẢ.Số-thẻSố-thẻ})$ và chiếu $[\text{Tên_sách}]$

nhờ đây phép toán:

$[\text{Tên-sách}, \text{SÁCH.Mã_sách}, \text{MUỖN-SÁCH.mã-sách}]$ (1)

$: (\text{SÁCH.Mã_sách} = \text{MUỖN-SÁCH.Mã-sách})$ (2)

rồi chiếu để lấy tên sách:

$[\text{Tên-sách}]$ (3)

Áp dụng quy tắc ” để thay thế biểu thức đầu tiên, biểu thức số (1), của phép chiếu nhờ phép chiếu:

[Tên-sáchTên-sách, SÁCH.Mã_sách]

để áp dụng cho quan hệ SÁCH và [MUỖN-SÁCH.mã-sách] áp dụng cho hạng thức phía trái của tích Đề-các trong hình 8.2.

Phép chiếu cuối và phép chọn có thể áp dụng quy tắc mở rộng ? để có dãy:

[áp dụng cho hạng thức phía trái của tích Đề-các trong hình 8.2.

Phép chiếu cuối và phép chọn có thể áp dụng quy tắc mở rộng ? để có dãy:

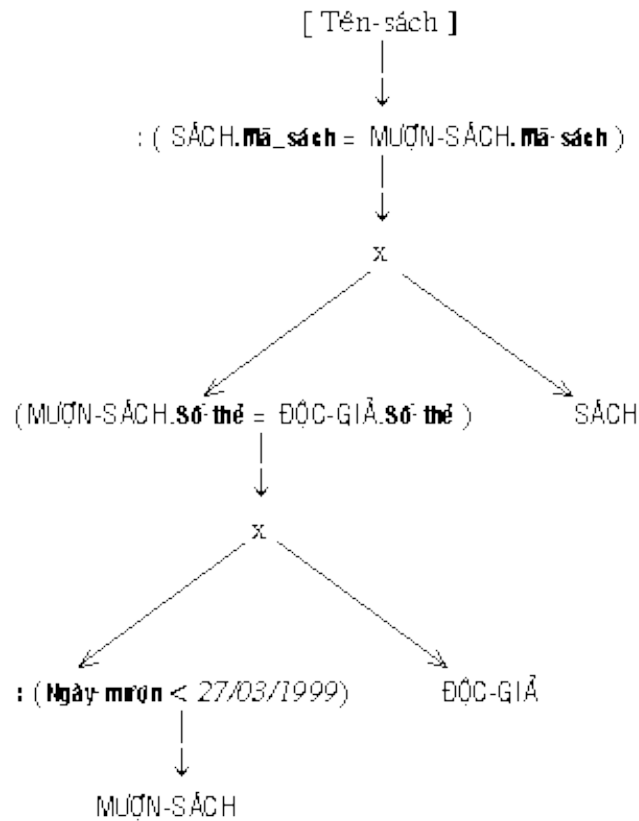
[MUỖN-SÁCH.Mã_sáchMã_sách, MUỖN-SÁCH.Số-thẻSố-thẻ, ĐỘC-GIẢ.Số-thẻ]
(5)

: **(MUỖN-SÁCH.Số-thẻ = ĐỘC-GIẢ.Số-thẻ)** (6)

[MUỖN-SÁCH.Mã_sách]Mã_sách] (7)

Phép chiếu đầu, số (5), được phân tách chuyển xuống tích Đề-các nhờ quy tắc ” .

Một phần phép chiếu ĐỘC-GIẢ.Số-thẻ xuống hạng thức ĐỘC-GIẢ vì là thuộc tính của quan hệ này.



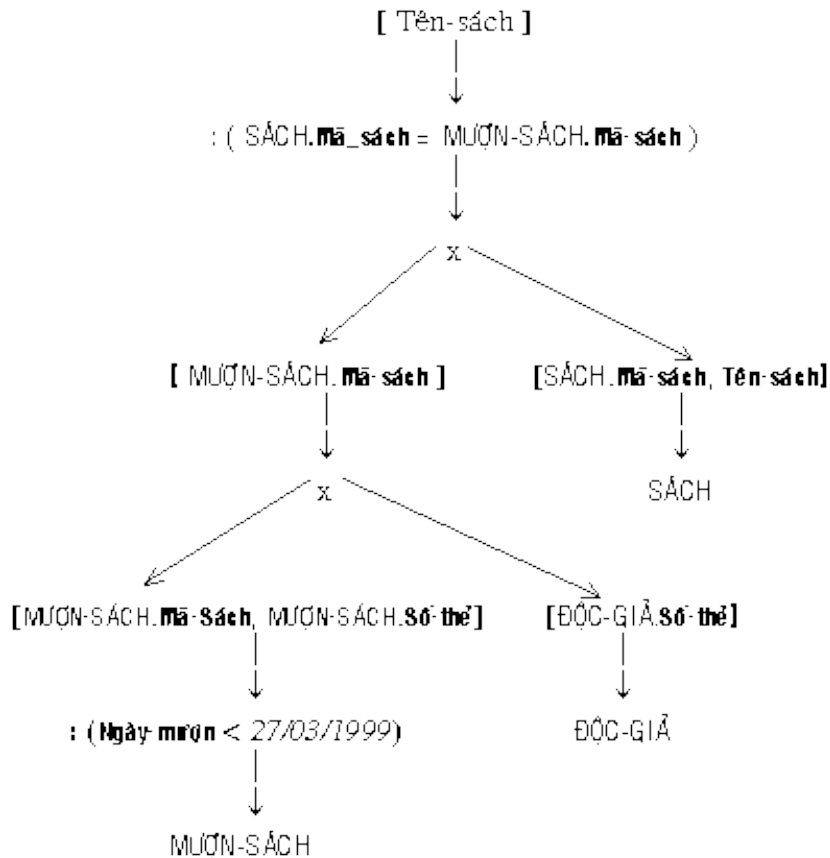
Hình 8.2 Cây với tổ hợp phép chọn và phép chiếu

Phần còn lại là phép chiếu để lấy 3 thuộc tính:

[MUỠN-SÁCH.Mã_sách, MUỠN-SÁCH.Số_thẻSố_thẻ, Ngày-mượn]

được đẩy xuống hạng thức thứ MUỠN-SÁCH. Các thuộc tính không phù hợp sẽ bị loại bỏ. Biểu diễn cây cuối cùng của biểu thức như trong hình 8.3.

Nhóm các phép tính bằng đường mũi tên gián đoạn. Mỗi tích Đề-các được tổ hợp với phép chọn để tạo thành một phép kết nối bằng nhau (*Equi Join*) rất có hiệu quả. Đặc biệt phép chọn trên quan hệ MUỠN-SÁCH và phép chiếu của ĐỘC-GIẢ để lấy thuộc tính **Số_thẻ** ở phía dưới là đủ để tổ hợp với tích Đề-các. Thứ tự thực hiện của cây biểu thức trong các hình 8.3, 8.2 và 8.3 là từ dưới lên: Nhóm các phép toán nằm ở phía dưới được thực hiện trước các phép toán ở phía trên.



Hình 8.3 Cây kết quả biểu diễn việc phân nhóm các biểu thức.

Một thuật toán để tối ưu hóa câu hỏi trong ngôn ngữ ĐSQH

Ví dụ trên cho ta một minh họa về việc chuyển đổi một câu hỏi bằng ngôn ngữ Đại số quan hệ về dạng tương đương tốt hơn (hay tối ưu hơn). Phương pháp trên tập trung chủ yếu vào các phép chiếu, phép chọn và tích Đề-các, với mục đích làm sao "đẩy" được phép chọn và phép chiếu xuống mức thấp nhất, tức là thi hành các phép toán này càng sớm càng tốt, nếu có thể. Tiếp theo, kết hợp các phép chọn với tích Đề-các thành phép kết tự nhiên để làm giảm các kết quả trung gian. Cốt lõi của vấn đề tối ưu hóa chính là việc làm giảm thiểu lưu trữ trung gian và từ đó làm tăng nhanh tốc độ xử lý câu hỏi.

Tuy nhiên, để thực hiện được các quá trình tối ưu hóa như trên, chúng ta cần lưu ý tới thứ tự thực hiện các phép toán để có thể "đẩy" các phép toán xuống các mức hợp lý cần thiết. Bảng dưới đây cho phép chúng ta cách thực hiện các phép biến đổi tương đương đối với các phép Hội (*Union*), Trừ (*Minus*), Giao (*Intersect*), Tích Đề-các (*Cartesian*), Chia (*Division*), Chiếu (*Projection*) và Chọn (*Selection*).

-(B1). Kết tự nhiên tương đương với dãy phép tích Đề-các, phép chọn và phép chiếu:

$$Q_1(A,B) * Q_2(B,C) (Q_1 \times Q_2 : (Q_1[B]=Q_2[B])) [A,B,C]$$

-(B2). Phép *theta* kết tương đương với dãy phép tích Đề-các và phép chọn với điều kiện *theta*:

$$Q_1(A,B) \text{ } Q_2(C,D) \text{ } (Q_1 \times Q_2) : (B \text{ } D)$$

-(B3). Phép giao (*Intersect*) tương đương với phần bù (*Complement*) của hội hai phần bù của 2 quan hệ:

$$Q_1 \text{ } Q_2((Q_1) \text{ } (Q_2)) \text{ và } (B4)$$

-(B4). Phép bù của một quan hệ tương đương với tích Đề-các của các phép chiếu trên từng thuộc tính của quan hệ trừ đi các bộ giá trị đã có trong thể hiện của quan hệ:

$$Q(X_1, X_2, \dots, X_n) (Q[X_1] \times Q[X_2] \times \dots \times Q[X_n]) - Q(X_1, \dots, X_n)$$

-(B5). Thương của 2 quan hệ tương đương với hiệu của các quan hệ trung gian sau:

$$Q_1(A,B) \text{ } Q_2(A) = Q_1[B] - ((Q_1[B] \times Q_2[A] - Q_1(A,B)) [B])$$

Áp dụng các cách biến đổi tương đương trên, kết hợp với các quy tắc "đẩy" và kết hợp như đã trình bày trong mục 8.2, chúng ta đưa ra thuật toán tổng quát để tối ưu hóa các câu hỏi trong ngôn ngữ đại số quan hệ.

-*Thuật toán*:

Đầu vào (*Input*): Sơ đồ cú pháp câu hỏi bằng ngôn ngữ đại số quan hệ.

Đầu ra (*Output*): Sơ đồ cú pháp tối ưu.

-*Bước 1*. Áp dụng các phép biến đổi tương đương nêu trong bảng (B1) đến (B5) trên.

-*Bước 2*. Áp dụng luật biến đổi dãy các phép chọn tương đương: tách phép chọn thành các phép chọn con.

-*Bước 3*. Đối với mỗi phép chọn, áp dụng các luật ‘, ’ và “ nhằm đẩy các phép toán chọn đó xuống càng sâu càng tốt.

-*Bước 4*. Đối với mỗi phép chiếu, áp dụng các quy tắc \checkmark , ” và nhằm đẩy các phép toán chiếu đó xuống càng sâu càng tốt.

-*Bước 5*.

-Tập trung các phép chọn nhằm áp dụng luật

-Áp dụng luật Σ để loại bỏ bớt các phép chiếu vô ích.

-Tập trung các phép chọn với tích Đề-các, nếu được, để chuyển thành phép kết tự nhiên hay *theta* kết bằng cách áp dụng các luật Σ và σ .

+**Nhận xét:**

- Thuật giải nêu trên chủ yếu nhằm giảm khối lượng dữ liệu trung gian chứ không chỉ ra thứ tự thực hiện các phép kết. Ví dụ:

$$(Q_1(A,B) * Q_2(B,C)) * Q_3(A,C)$$

$$(Q_1(A,B) * Q_3(A,C)) * Q_2(B,C)$$

-Thuật giải này không cho chúng ta một kết quả tối ưu mà nó chỉ đưa ra một giải pháp tốt.

- Các phép biến đổi chỉ dựa trên các phép toán cơ bản là Hội (*Union*), Trừ (*Minus*), Giao (*Intersect*), Tích Đề-các (*Cartesian*), Chia (*Division*), Chiếu (*Projection*) và Chọn (*Selection*) mà chúng ta còn có thể thực hiện các phép biến đổi dựa trên các phép toán khác nữa.

Tối ưu hóa câu hỏi (tiếp theo)

Một số kỹ thuật tối ưu cấp thấp.

Trong mục 8.2 đã trình bày các tư tưởng cơ bản để thực hiện tối ưu một biểu thức quan hệ. Trong phần này sẽ trình bày một lớp các câu truy vấn rất phổ dụng và đã được nhiều hệ thống cài đặt có hiệu quả. Vấn đề đặt ra là khi câu hỏi đã được biến đổi và tối ưu có tương đương với câu hỏi ban đầu hay không? Lớp câu hỏi sẽ quan sát bao gồm phép chiếu (*Projection*), phép chọn (*Selection*) và phép kết tự nhiên (*Natural Join*) có nghĩa là các biểu thức truy vấn chỉ bao gồm ba phép tính đó. Kết quả cài đặt của nhiều hệ thống (như trong System R) là hiệu quả nhưng không phải là tổng quát cho tất cả.

Như phần trên đã trình bày, trong các phép toán quan hệ thì phép tích Đề các, phép kết (*Join*) làm mất rất nhiều thời gian xử lý và tốn nhiều bộ nhớ. Vì vậy, nguyên tắc cơ bản của phần này là làm thế nào để biểu thức tối ưu cuối cùng (tương đương với biểu thức ban đầu) có số phép kết là tối thiểu. Do vậy, trong quá trình biến đổi tương đương, điều quan trọng là loại bỏ được các phép kết không cần thiết.

Để tiện lợi cho quá trình biến đổi, cần làm quen với khái niệm "quan hệ vũ trụ" (*universal relation*), có thể hiểu một cách đơn giản đó là phép kết tự nhiên của tất cả các quan hệ trong CSDL thành một quan hệ chung.

Câu hỏi hội và bảng

Một câu hỏi hội (*conjunctive query*) là câu hỏi có dạng:

$$\{a_1 \dots a_n \mid (\$ b_1) \dots (\$ b_m) (p_1 \dots p_k) \}$$

trong đó $p_i, i = 1, \dots, k$ là một trong các dạng:

$R(c_1, c_2, \dots, c_r)$, nghĩa là các bộ c_1, c_2, \dots, c_r thuộc quan hệ R ; c_j hoặc là các giá trị a , hoặc các giá trị b hoặc là một trực hằng (*literal*);

$c \text{ q } d$ trong đó c và d là các giá trị a hoặc b hoặc là trực hằng; q là một trong các phép so sánh $<, \leq, >, \geq$. Chú ý rằng các phép $=$ và \neq là không được tính trong danh sách.

Thông thường, câu hỏi được biểu diễn dưới dạng bảng (*tableau*). Để mô tả khối lượng bảng, trước hết cần giả thiết rằng tập các thuộc tính của các quan hệ đang quan sát là như nhau. Các thuộc tính có cùng tên ở các quan hệ khác nhau đều có cùng một ý nghĩa và các thuộc tính có ý nghĩa khác nhau thì phải mang tên khác nhau.

Định nghĩa bảng

Bảng có thể được xem như một mảng hai chiều và thêm một số điều kiện ràng buộc. Hàng thứ nhất của bảng là danh sách tất cả các thuộc tính. Mỗi thuộc tính tương ứng với một cột của bảng. Hàng thứ hai gọi là hàng đích (*summary*) bao gồm các giá trị rỗng (*blank*), các ký hiệu phân biệt (*distinguished symbols*) và hằng. Ký hiệu phân biệt thường được ký hiệu bằng chữ cái a (thường) biểu diễn trong câu hỏi hội. Các hàng còn lại nhận các ký hiệu a hoặc trống hoặc b , với b là biến không phân biệt (*nondistinguished variable*) hoặc ký hiệu không phân biệt (*nondistinguished symbols*). Các hàng này biểu diễn các hạng thức của câu hỏi hội dưới dạng $R(c_1, c_2, \dots, c_k)$ tức là xác định các bộ trong từng quan hệ. Tại vị trí c_i trong cột là tương ứng với thuộc tính thứ i của quan hệ, và sẽ là giá trị trống trong cột của bảng nếu không tương ứng với thuộc tính nào của lược đồ quan hệ R . Các tần từ c q d của câu hỏi hội được xem là các ràng buộc của bảng.

Chú ý bảng đòi hỏi rằng cùng một biến không được phép xuất hiện đồng thời trên hai cột khác nhau của bảng, còn biến phân biệt không xuất hiện ở hàng đích của cột đó.

Như vậy bảng có thể được diễn giải như một câu hỏi hội, tức là một ánh xạ từ các giá trị của các biến quan hệ quan hệ. Quan hệ kết quả là quan hệ trên tập thuộc tính với các ký hiệu phân biệt và hằng trên hàng đích.

Gọi T là một bảng và S là tập tất cả các ký hiệu xuất hiện trong T (tức là các biến và hằng). Một đánh giá r cho T liên quan với mỗi ký hiệu của S là một hằng, sao cho nếu một hằng $c \in S$ thì $r(c) = c$ đối với hàng đích và các hàng khác của T có thể mở rộng như sau:

Gọi w_0 là hàng đích của T , $w_1 \dots w_n$ là các hàng còn lại thì $r(w_i)$ là bộ nhận được nhờ thay thế $r(v)$ cho mỗi biến v xuất hiện trong w_i . Bảng được định nghĩa là ánh xạ từ các trạng thái vào các quan hệ trên một tập các thuộc tính. Vậy, nếu T là một bảng, I là một trạng thái thì $T(I)$ là một quan hệ trên tập các thuộc tính mà các thuộc tính đó là những cột có giá trị khác trống trên hàng đích, sao cho

$$T(I) = \{r(w_0) \text{ với đánh giá } r(w_i) \in I, 1 \leq i \leq n\}.$$

Chú ý: Quy định rằng I là bảng rỗng. Bảng này biểu diễn ánh xạ mỗi trạng thái vào quan hệ rỗng.

Tính tương đương của bảng

Hai bảng T_1 và T_2 là tương đương, ký hiệu $T_1 \equiv T_2$ nếu với mọi tình trạng I , $T_1(I) = T_2(I)$. Nói rằng T_1 chứa trong T_2 , viết tắt là $T_1 \subseteq T_2$ nếu với mọi I , $T_1(I) \subseteq T_2(I)$. Lưu

ý rằng điều kiện cần nhưng không đủ để cho $T_1 \circ T_2$ và $T_1 \upharpoonright T_2$ là các quan hệ được xác định bởi T_1 và T_2 phải có cùng lược đồ đích (lược đồ đích là tập hợp các thuộc tính biểu diễn hàng đầu tiên của bảng).

Biểu diễn một biểu thức nhờ bảng

Trong phần này trình bày cách thức cấu trúc một bảng để biểu diễn một biểu thức quan hệ bằng các phép Chiếu, Chọn và Kết tự nhiên. Việc cấu trúc bảng trước hết tiến hành cho từng hạng thức của biểu thức, sau đó tổ hợp các bảng lại thành một bảng chung. Các quy tắc thiết lập bảng T cho biểu thức E được thể hiện đệ quy như sau:

(1) Nếu E là lược đồ quan hệ R thì bảng T đối với E chỉ có một hàng và hàng đích, trong đó:

(i) Nếu $A \in R$ thì tại cột cho thuộc tính A của T sẽ có cùng giá trị biến phân biệt cho cả hàng đích và hàng đó:

(ii) Nếu $A \notin R$ thì giá trị của cột tại hàng đích là trống và tại hàng là biến không phân biệt.

(2a) Giả sử biểu thức chọn E có dạng $E_1 : (A = C)$ và đã cấu trúc bảng T_1 cho biểu thức E_1 .

(i) Nếu hàng đích của T_1 có ký hiệu trống tại cột A thì $T = I$

(ii) Nếu có một hằng c và c' tại A của hàng đích, thì $T = I$ nếu $c = c'$ thì $T = T_1$.

(iii) Nếu T_1 có biến phân biệt a ở cột A của hàng đích thì bảng T đối với E được cấu trúc nhờ việc thay a bằng c tại mọi chỗ mà a xuất hiện trong T_1 (tức là các giá trị a xuất hiện trong cột a của bảng T_1).

(2b) Giả sử E là phép chiếu có dạng $E_1 [X]$ và T_1 là bảng của E_1 . Cấu trúc bảng T cho E bằng cách xóa tất cả các ký hiệu xuất hiện tại hàng đích của tất cả các cột không xuất hiện trong T . Các biến phân biệt trong những cột này được thay bằng biến không phân biệt.

(2c) Giả sử E là biểu thức đại số quan hệ chứa phép kết tự nhiên có dạng $E_1 * E_2$ và T_1, T_2 là các bảng tương ứng của E_1 và E_2 . Gọi S_1 và S_2 là các ký hiệu của T_1 và T_2 . S_1 và S_2 có các tập hợp của biến không phân biệt rời nhau nhưng các biến phân biệt là trùng nhau trên các cột tương ứng của phép kết nối.

(i) Nếu T_1 và T_2 có một số cột trùng nhau nhưng giá trị của chúng tại hàng đích là những hằng số khác nhau $T = I$

(ii) Nếu các vị trí tương ứng trong các hàng đích không có các hằng phân biệt thì các hàng của bảng T cho E được bao gồm từ hợp của các hàng thuộc bảng T_1 và T_2 .

Hàng đích của T có các cột của hai bảng, các cột chung nhận các giá trị:

(a) là hằng c nếu một trong hai bảng T_1, T_2 có hằng c tại cột tương ứng của hàng đích. Trong trường hợp này chúng ta sẽ thay thế mọi giá trị của các biến phân biệt trong cột bằng giá trị hằng c .

(b) là biến phân biệt a nếu không áp dụng được quy tắc (a) nhưng một trong hai giá trị của T_1 và T_2 tại cột tương ứng có giá trị là a ở hàng đích.

(c) là ký hiệu trống trong những trường hợp còn lại.

Chú ý: Trong trường hợp biểu thức E chứa phép chọn có dạng $E_1 : (A = B)$ với A và B tên hai thuộc tính, khi đó đồng nhất các giá trị biến phân biệt trong hàng đích của T cho cả hai cột A và B . Trong tài liệu này chỉ trình bày trường hợp phép chọn $E_1 : (A = c)$ với c là một hằng số.

Xét ví dụ: Cho A, B và C là các thuộc tính và giả sử rằng có biểu thức quan hệ:

$((R(AB) * S(BC)) : (B = 0)) [AC]$

Theo quy tắc (1) thiết bị lập bảng cho lược đồ $R(AB)$ và lược đồ $S(BC)$ như sau:

R	A	B		S	B	C
	a1	a2	Và		a2	a3
	a1	a2			a2	a3

Theo quy tắc (2c) bảng cho phép kết tự nhiên $R(AB) * S(BC)$ từ hai bảng trên:

A	B	C
a1	a2	a3
a1	a2	b1
b2	a2	a3

Và cuối cùng, theo quy tắc (2b), bảng cho phép chọn rồi chiếu $((R(AB) * S(BC)) : (B=0)) [AC]$ là:

A	B	C
a1		a3
a1	0	b1
b2	0	a3

Kiểm tra tính tương đương của bảng

Vấn đề đặt ra làm thế nào để tối thiểu hóa số hàng của một bảng mà vẫn tương đương với bảng xuất phát. Như trên đã nêu, cần thiết phải ánh xạ các hàng của bảng xuất phát lên một bảng khác có số hàng ít hơn (ví dụ $T_1 \rightarrow T_2$) sao cho $T_2 \models T_1$ với điều kiện:

Các bảng phải có cùng lược đồ đích (xác định trên cùng một tập thuộc tính) và có cùng biến phân biệt trên cùng một vị trí của bảng.

Với mỗi phép gán quan hệ cho biến quan hệ tại các hàng của bảng, quan hệ sinh ra nhờ T_1 phải là tập con của quan hệ sinh ra nhờ T_2 .

Để kiểm tra các bảng trong quá trình biến đổi liệu có tương đương với nhau không, trước hết cần thêm khái niệm ánh xạ hàng - hàng (*Row - Row Mapping*) được gọi là ánh xạ cuốn (*Containment Mapping*).

Gọi T_1 và T_2 là hai bảng với các tập ký hiệu tương ứng là S_1 và S_2 . Một đồng cấu (*Homomorphism*) là một ánh xạ $y : S_1 \rightarrow S_2$ sao cho:

- (i) Nếu c là một hàng số thì $y(c) = c$.
- (ii) Nếu a là một biến phân biệt thì $y(a)$ hoặc là biến phân biệt hoặc là một hằng số xuất hiện trong cột tương ứng của hàng đích của T_2 .
- (iii) Nếu w là một hàng của T_1 thì $y(w)$ là một hàng của T_2 .

Như vậy có thể ánh xạ các hàng của T_2 lên các phần tử của một trạng thái I và vì thế đồng cấu y sẽ cho một ánh xạ từ các hàng của T_1 lên I . Do đó $T_2(I) \models T_1(I)$ với mọi I , từ đó có $T_2 \models T_1$.

Ngược lại cũng hoàn toàn đúng, nếu $T_2 \models T_1$ suy ra được:

$T_2(I) \models T_1(I)$.

Có thể hình thức hóa dưới định lý sau đây:

Định lý 8.1:

Gọi T_1 và T_2 là hai bảng với tập các ký hiệu tương ứng là S_1 và S_2 . $T_2 \leq T_1$ khi và chỉ khi chúng có cùng lược đồ đích và có một đồng cấu $y: S_1 \rightarrow S_2$.

Ánh xạ cuốn

Ánh xạ cuốn là ánh xạ từ các hàng của bảng này lên các hàng của bảng khác mà bảo toàn biến phân biệt và các hằng số nhưng không ánh xạ một ký hiệu lên hai ký hiệu phân biệt. Một cách hình thức có thể nói rằng, nếu T_1, T_2 là hai bảng, q là ánh xạ từ các hàng của T_1 lên các hàng của T_2 , q được gọi là ánh xạ cuốn nếu:

- (a) Với mỗi hàng i của T_1 , nếu hàng i có biến phân biệt ở cột A thì hàng $q(i)$ của T_2 cũng có biến phân biệt hoặc hằng số ở cột A .
- (b) Nếu hàng i của T_1 có hằng số c ở cột A thì hàng $q(i)$ có c ở cột A .
- (c) Nếu hàng i và j của T_1 có cùng một biến không phân biệt ở cột A thì hàng $q(i)$ và $q(j)$ có cùng một ký hiệu ở cột đó. Ký hiệu có thể là hằng số, biến phân biệt hoặc biến không phân biệt. (Chú ý rằng, vẫn có thể $q(i) = q(j)$). Từ đó có định lý sau:

Định lý 8.2 :

$T_2 \leq T_1$ nếu và chỉ nếu chúng xác định trên cùng một lược đồ đích và tồn tại một ánh xạ cuốn q từ T_1 lên T_2 .

Chứng minh: (Nếu) Gọi $y: S_1 \rightarrow S_2$ là ánh xạ ký hiệu - ký hiệu sao cho nếu ký hiệu d xuất hiện ở cột A của hàng r của T_1 và ký hiệu d' xuất hiện ở cột A của hàng $q(r)$ của hàng T_2 thì $y(d) = d'$. Ánh xạ y là phù hợp nhờ điều kiện (c). Điều kiện (i) - (iii) cho y dễ dàng suy trực tiếp. Thật vậy, (a) kéo theo (ii), (b) kéo theo (i) và (iii) được suy ra từ định nghĩa của y và q . Do vậy y là một đồng cấu. Theo định lý 8.1 thì $T_2 \leq T_1$.

(Chỉ nếu): Từ định lý 8.1, có một đồng cấu $y: S_1 \rightarrow S_2$ thỏa (i) - (iii). Tồn tại ánh xạ từ các hàng của T_2 thỏa (c) suy ra từ (iii), (i) và (ii). Từ kéo theo (a) và (b).

Ví dụ: Cho biểu thức $(AB * BC)[AB]$, có bảng:

		A	B	C
		a1	a2	

T1 =	w1	a1	a2	b1
	w2	b2	a2	a3

Biểu thức AB trên các thuộc tính A, B và C có bảng:

		A	B	C
T2=		a1	a2	
	w3	a1	a2	b1

Theo chiều xuôi, ánh xạ cả hai hàng w₁ và w₂ lên w₃ là ánh xạ cuốn. Đồng cấu là:

Trong T ₁	Trong T ₂
a1	a1
a2	a2
b1	b1
b2	a1
b3	b1

Theo chiều ngược lại, ánh xạ w₃ lên w₁ và chúng ta nhận thấy rõ ràng rằng ánh xạ là đúng.

Do vậy AB và (AB*AB) [AB] là tương đương.

Ví dụ:

$E_1 = AB * BC$ và $E_2 = (ABC) * ((BC) : (C=0))$

Bảng tương ứng của E₁, E₂ là:

	A	B	C		A	B	C
	a1	a2	a3		a1	a2	0
T ₁ = w1	a1	a2	b1	T ₂ = w4	a1	a2	0
w2	a1	b2	a3	W5	b1	a2	0
w3	b3	a2	a3				

Do vậy $T_2 \leq T_1$ vì có thể tạo một ánh xạ cuộn chuyển w_1, w_2 và w_3 lên w_4 . Có thể chọn ánh xạ w_3 lên w_5 nếu muốn. Chiều ngược lại không tồn tại ánh xạ cuộn vì hàng số 0 không thể ánh xạ lên biến. Do đó $T_1 \not\leq T_2$.

Từ đó có định lý sau đây.

Định lý 8.3:

Nếu T_1 và T_2 là hai bảng tương đương và không bảng nào trong hai bảng đó là tương đương với một bảng khác có số dòng ít hơn thì tồn tại một tương ứng 1-1 của các hàng thuộc T_1 đối với các hàng thuộc T_2 . Đó là ánh xạ cuộn cả hai chiều.

Tối thiểu hóa bảng

Cho T_0 là bảng xuất phát. T_m là bảng tương đương với số hàng tối thiểu như có thể. Các hàng của bảng tương đương T_m phải là một tập con của các hàng thuộc T_0 (kể cả việc đặt tên lại các ký hiệu). Có định lý sau đây:

Định lý 8.4 :

Nếu T_0 là một bảng, luôn có thể thiết lập được một bảng khác tương đương và có số hàng là tối thiểu được biến đổi từ T_0 nhờ xóa đi không hoặc một số hàng.

Hệ quả :

Mọi bảng với số hàng tối thiểu tương đương với bảng cho trước đều là một (kể cả việc đặt tên lại các ký hiệu)

Từ định lý 8.4 gợi ý một thủ tục tổng quát để tìm bảng tối thiểu. Trước hết tìm ánh xạ có thể để ánh xạ tất cả các hàng lên một tập con của các hàng. Cần chú ý xác định các ánh xạ hàng đích lên hàng đích (tức là đồng nhất trên các biến phân biệt) và bảo toàn các ràng buộc.

Vấn đề tìm ánh xạ để có số hàng ít nhất là một bài toán NP - đầy đủ nhưng số lượng hàng một bảng trong thực tế là nhỏ cho nên không phải luôn luôn là vấn đề thật khó khăn.

Tham gia đóng góp

Tài liệu: Giáo trình cơ sở dữ liệu

Biên tập bởi: Ngô Trần Thanh Thảo

URL: <http://voer.edu.vn/c/3eaa132c>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Các khái niệm cơ bản

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/0bc507c9>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Những khái niệm cơ bản

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/17d2f805>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Những cách tiếp cận một cơ sở dữ liệu

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/aafc3b70>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Mô hình dữ liệu quan hệ của e.f.codd

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/1783ba73>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Ràng buộc toàn vẹn

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/5d5ea197>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Ràng buộc toàn vẹn các loại

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/f50fe2aa>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Ngôn ngữ đại số quan hệ

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/5b211420>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Ngôn ngữ đại số quan hệ (tiếp theo)

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/caee7f61>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Ngôn ngữ truy vấn cơ sở dữ liệu SQL

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/dd774834>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Ngôn ngữ truy vấn CSDL SQL (tiếp theo)

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/f024e7e6>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Ngôn ngữ tân từ

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/44860fdb>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Ngôn ngữ tân từ (tiếp theo)

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/557735f3>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Tối ưu hóa câu hỏi

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/102467c8>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Module: Tối ưu hóa câu hỏi (tiếp theo)

Các tác giả: Ngô Trần Thanh Thảo

URL: <http://www.voer.edu.vn/m/14063540>

Giấy phép: <http://creativecommons.org/licenses/by/3.0/>

Chương trình Thư viện Học liệu Mở Việt Nam

Chương trình Thư viện Học liệu Mở Việt Nam (Vietnam Open Educational Resources – VOER) được hỗ trợ bởi Quỹ Việt Nam. Mục tiêu của chương trình là xây dựng kho Tài nguyên giáo dục Mở miễn phí của người Việt và cho người Việt, có nội dung phong phú. Các nội dung đều tuân thủ Giấy phép Creative Commons Attribution (CC-by) 4.0 do đó các nội dung đều có thể được sử dụng, tái sử dụng và truy nhập miễn phí trước hết trong môi trường giảng dạy, học tập và nghiên cứu sau đó cho toàn xã hội.

Với sự hỗ trợ của Quỹ Việt Nam, Thư viện Học liệu Mở Việt Nam (VOER) đã trở thành một cổng thông tin chính cho các sinh viên và giảng viên trong và ngoài Việt Nam. Mỗi ngày có hàng chục nghìn lượt truy cập VOER (www.voer.edu.vn) để nghiên cứu, học tập và tải tài liệu giảng dạy về. Với hàng chục nghìn module kiến thức từ hàng nghìn tác giả khác nhau đóng góp, Thư Viện Học liệu Mở Việt Nam là một kho tàng tài liệu khổng lồ, nội dung phong phú phục vụ cho tất cả các nhu cầu học tập, nghiên cứu của độc giả.

Nguồn tài liệu mở phong phú có trên VOER có được là do sự chia sẻ tự nguyện của các tác giả trong và ngoài nước. Quá trình chia sẻ tài liệu trên VOER trở lên dễ dàng như đếm 1, 2, 3 nhờ vào sức mạnh của nền tảng Hanoi Spring.

Hanoi Spring là một nền tảng công nghệ tiên tiến được thiết kế cho phép công chúng dễ dàng chia sẻ tài liệu giảng dạy, học tập cũng như chủ động phát triển chương trình giảng dạy dựa trên khái niệm về học liệu mở (OCW) và tài nguyên giáo dục mở (OER). Khái niệm chia sẻ tri thức có tính cách mạng đã được khởi xướng và phát triển tiên phong bởi Đại học MIT và Đại học Rice Hoa Kỳ trong vòng một thập kỷ qua. Kể từ đó, phong trào Tài nguyên Giáo dục Mở đã phát triển nhanh chóng, được UNESCO hỗ trợ và được chấp nhận như một chương trình chính thức ở nhiều nước trên thế giới.