BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC CÔNG NGHỆ TP.HCM

CƠ SỞ DỮ LIỆU VÀ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Biên Soạn:

Cao Tùng Anh

www.hutech.edu.vn

CƠ SỞ DỮ LIỆU VÀ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Các ý kiến đóng góp về tài liệu học tập này, xin gửi về e-mail của ban biên tập: tailieuhoctap@hutech.edu.vn

MỤC LỤC

HƯỚNG DẪN

MÔ TẢ MÔN HỌC

Nhu cầu lưu trữ và xử lý dữ liệu nảy sinh trong mọi công việc, trong mọi hoạt động của con người. Ví dụ như: Quản lý kho hàng, quản lý nhân sự, quản lý tiền lương trong các cơ quan xí nghiệp hay quản lý điểm, quản lý học sinh trong các trường…

Khi chưa có cơ sở dữ liệu (CSDL), người xử lý có thể lưu trữ và khai thác dữ liệu một cách thủ công. Với sự xuất hiện của máy tính, cho phép lưu trữ và xử lý dữ liệu một cách tự động, từ đó phát sinh ra nhu cầu xây dựng các phần mềm ứng dụng phục vụ cho các công tác quản lý bằng máy tính điện tử. Để thực hiện được vấn đề này, trước tiên người ta phải biết cách phân tích, thiết kế, tổ chức thông tin của các hệ thống quản lý. Sau đó cài đặt trên các phần mềm quản trị CSDL để từ đó có thể xử lý và khai thác một cách tốt nhất.

Các hệ cơ sở dữ liệu đầu tiên được xây dựng theo các mô hình phân cấp và mô hình mạng, đã xuất hiện vào những năm 1960, được xem là thế hệ thứ nhất của các hệ quản trị cơ sở dữ liệu (hệ QTCSDL).

Tiếp theo là thế hệ thứ hai, các hệ quản trị cơ sở dữ liệu quan hệ, được xây dựng theo mô hình dữ liệu quan hệ do E.F. Codd đề xuất vào năm 1970.

Các hệ QTCSDL có mục tiêu tổ chức dữ liệu, truy cập và cập nhật những khối lượng lớn dữ liệu một cách thuận lợi, an toàn và hiệu quả.

Hai thế hệ đầu các hệ QTCSDL đã đáp ứng được nhu cầu thu thập thông tin, xây dựng CSDL và tổ chức khai thác dữ liệu của các cơ quan, các xí nghiệp và các tổ chức một cách hiệu quả.

Mục đích của giáo trình Cơ Sở Dữ Liệu và Quản Trị Cơ Sở Dữ Liệu nhằm trình bày các khái niệm về CSDL và các thuật toán cơ bản như: Tính bao đóng của tập thuộc tính, tìm khóa của lược đồ quan hệ, xác định dạng chuẩn của lược đồ quan hệ… Để từ đó sinh viên có thể thiết kế một CSDL áp dụng được trong thực tế. Ngoài ra, giáo trình còn cung cấp các ngôn ngữ cơ bản như: đại số quan hệ, SQL (Structured Query Language) và các đối tượng của hệ QTCSDL quan hệ SQL Server để sinh viên có thể tạo và quản trị CSDL tốt trên hệ QTCSDL cơ sở dữ liệu này.

NỘI DUNG MÔN HỌC

- Bài 1. Tổng quan về cơ sở dữ liệu: Bài này cung cấp cho học viên các khái niệm cơ bản về cơ sở dữ liệu, hệ quản trị CSDL, các mô hình dữ liệu đang sử dụng trong CSDL quan hệ và cách xây dựng các mô hình dữ liệu này.
- Bài 2: Mô hình dữ liệu quan hệ. Các khái niệm về cơ sở dữ liệu quan hệ. Ngôn ngữ đại số quan hệ gồm các phép toán trên tập hợp và các phép toán trên quan hệ
- Bài 3: Ngôn ngữ xử lý vấn tin. Bài này trình bày các phương pháp xử lý, tạo ra CSDL và truy vấn dữ liệu từ CSDL bằng ngôn ngữ SQL.
- Bài 4: Ràng buộc toàn vẹn. Bài này trình bày các khái niệm về ràng buộc toàn vẹn, các yếu tố của ràng buộc toàn vẹn và phân loại các ràng buộc toàn vẹn. Các phương pháp để tìm ra ràng buộc toàn vẹn có trong một CSDL.
- Bài 5: Thiết kế cơ sở dữ liệu. Bài này trình bày các khái niệm về tập thuộc tính, phụ thuộc hàm, các tính chất của phụ thuộc hàm. Các thuật toán về tính bao đóng của tập thuộc tính, các thuật toán tìm khóa của một lược đồ quan hệ.
- Bài 6: Các dạng chuẩn cơ bản. Bài này trình bày 4 dạng chuẩn cơ bản liên quan đến phụ thuộc hàm. Và phương pháp xác định dạng chuẩn của một lược đồ quan hệ.
- Bài 7: Chỉ mục (index). Bài này giới thiệu cho sinh viên khái niệm và cách sử dụng Index trong SQL Server. Index cung cấp một phương pháp giúp người sử dụng CSDL có thể nhanh chóng tìm kiếm dữ liệu dựa trên các giá trị trong các cột đã được chỉ định sắp xếp trong index.
- Bài 8: Khung nhìn (view). Giới thiệu cho sinh viên khái niệm về view. View là một bảng tạm, có cấu trúc như một bảng, view không lưu trữ dữ liệu mà nó được tạo ra khi sử dụng, view là một đối tượng trong hệ quản trị CSDL SQL server và được sử dụng trong khai thác dữ liệu như một bảng dữ liệu để kế thừa lại các kết quả truy vấn có trước, chia sẻ nhiều người dùng, an toàn trong khai thác.
- Bài 9: Thủ tục nội tại (Stored Procedure - SP). Cung cấp cho sinh viên khải năng, tạo mới, xóa, sửa và sử dụng thủ tục trong SQL server. Thủ tục là một chương trình

con nhằm hỗ trợ lập trình có cấu trúc trong hệ quản trị SQL server. SP cũng tương tự như thủ tục trong các ngôn ngữ lập trình khác, nó có thể: nhận các thông số đầu vào và trả về giá trị cho nơi gọi.

- Bài 10: Hàm (function). Sau khi học bài này sinh viên có thể tạo mới, xóa, sửa và sử dụng hàm trong SQL server. Hàm cũng là chương trình con như thủ tục, cũng có thể truyền tham số và trả về giá trị bằng lệnh RETURN.
- Bài 11: Bẫy lỗi (trigger): Bài học này cung cấp cho sinh viên khả năng làm việc với đối tượng trigger. Trigger là một Stored procedure đặc biệt được tự động chạy mỗi khi có một hành động nào đó xảy ra có liên quan đến nó. Các hành động xảy ra giúp TRIGGER hoạt động: Insert, Delete, Update. Trigger không được thực thi một cách tường minh nên cần thận trọng khi dùng Trigger. Trigger hoạt động gần giống các hàm bắt sự kiện trong javascript.
- Bài 12: Một số kiến thực nâng cao. Bài học này cung cấp cho sinh viên một số kiến thức chuyên sâu trong hệ quản trị SQL server như: kiểu dữ liệu Cursor, mã hóa dữ liệu và quản trị CSDL trong hệ quản trị SQL server.
- Mỗi bài học sẽ có các ví dụ minh họa và bài tập đề nghị, học viên cần thực hiện trên lớp và trong giờ thực hành.

KIẾN THỨC TIỀN ĐỀ

Cơ sở dữ liệu và hệ quản trị cơ sở dữ liệu là môn học cơ sở bắt buộc cho sinh viên ngành công nghệ thông tin ở tất cả các chuyên ngành.

YÊU CẦU MÔN HỌC

Người học phải dự học đầy đủ các buổi lên lớp và làm bài tập đầy đủ ở nhà.

PHƯƠNG PHÁP ĐÁNH GIÁ MÔN HỌC

Môn học được đánh giá gồm:

- Điểm kiểm tra giữa kỳ: 50%. Hình thức sẽ áp dụng kết hợp chuyên cần trong giờ học lý thuyết và thực hiện các bài tập trong giờ học lý thuyết do Giảng viên giảng dạy môn học quyết định.

- Điểm thi cuối kỳ: 50%. Hình thức là làm bài thi trắc nghiệm trong 90 phút.

BÀI 1: TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU

Với việc dữ liệu trong các công ty, xí nghiệp, trường học, bệnh viện,… ngày càng nhiều và cơ sở dữ liệu được xây dựng ở hầu hết các đơn vị trong thực tế. Cơ sở dữ liệu (CSDL) là một môn học nghiên cứu về cách tổ chức và khai thác dữ liệu. Môn học này là một trong những môn học quan trọng nhất của ngành công nghệ thông tin.

1.1 HỆ CSDL DỮ LIỆU

1.1.1 Định nghĩa hệ CSDL

Một hệ CSDL (Database System) là một tập hợp dữ liệu được tổ chức một cách có chọn lọc, ghi trên các thiết bị trữ tin, nhằm phục vụ đồng thời cho nhiều người với nhiều mục đích khác nhau.

1.1.2 Các mức biểu diễn CSDL

Khi thiết kế hay đánh giá một CSDL người ta dựa trên 3 mức mô tả (Theo tiêu chuẩn của ANSI/X3/SPARC)

Mức lược đồ ngoài (External Schema Level):

Là mức ngoài cùng, dành cho người sử dụng. Mỗi người sử dụng hay mỗi nhóm làm việc sẽ yêu cầu xử lý dữ liệu theo một mục đích riêng.

Ví dụ: Bộ phận quản lý tiền lương cần lập bảng lương cho đơn vị gồm các thông tin: STT, họ tên, bậc lương, phụ cấp, tiền lương (trong đó, phụ cấp được tính theo hệ số dựa trên bậc lương).

Bộ phận quản lý công trình cần lập danh sách công nhân viên theo công trình với các thông tin: STT, họ tên, chuyên môn, công trình đã tham gia (trong đó, nhân viên được phân công phải có chuyên môn phù hợp với yêu cầu chuyên môn của từng công trình)

Mức quan niệm (Conceptial Schema Level):

Là mức mô tả tổng thể về CSDL, cho biết CSDL chứa được đối tượng dữ liệu nào, các mối quan hệ giữa các đối tượng, các yêu cầu ràng buộc trên các đối tượng được lưu trữ.

Ví dụ: Dựa trên các yêu cầu của hai nhóm công tác trong ví dụ trên, CSDL cần chứa các đối tượng như sau:

Nhân viên: Họ tên, bậc lương, chuyên môn

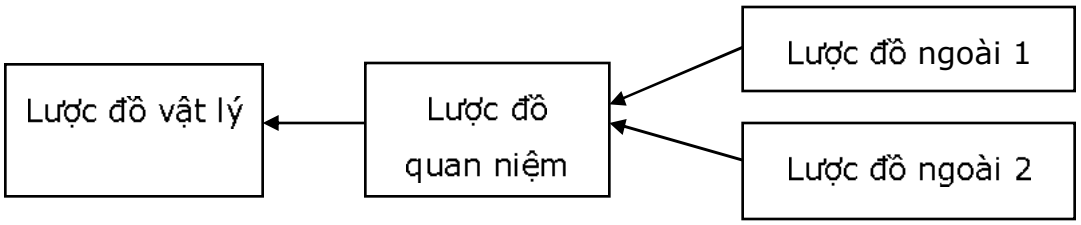
Chế độ phụ cấp: Bậc lương, hệ số phụ cấp

Công trình: Tên công trình, yêu cầu chuyên môn

Mức vật lý (Physical Schema Level):

Là mức cài đặt thật sự của CSDL, liên quan đến cách tổ chức dữ liệu, cách truy xuất các dữ liệu được lưu trữ.

Ví dụ: Từ ví dụ ở mức quan niệm ta hình thành ra mức vật lý gồm: cách tổ chức tập tin chứa dữ liệu, cấu trúc mẫu tin, kiểu dữ liệu và cách truy cập mẫu tin,…



Hình 1.1: Sơ đồ liên quan giữa các mức biểu diễn CSDL

1.1.3 Đặc tính của môi trường CSDL

a. Dữ liệu được chia sẻ cho nhiều người sử dụng khác nhau, để tiết kiệm được không gian lưu trữ, tăng hiệu quả khai thác.

b. Làm giảm tình trạng dữ liệu bị lưu trữ trùng lắp, bảo đảm được tính nhất quán trong việc truy xuất dữ liệu và tính toàn vẹn dữ liệu.

c. Có tính độc lập giữa dữ liệu và chương trình ứng dụng:

- Tính độc lập vật lý: Người dùng chỉ quan tâm đến nội dung dữ liệu, không quan tâm đến tổ chức bên trong, cũng như cách thức truy xuất dữ liệu (phụ thuộc vào chất lượng của hệ quản trị CSDL).

- Tính độc lập logic: Nếu ta sửa cấu trúc CSDL thì không cần viết lại những chương trình ứng dụng sẵn có (phụ thuộc vào chất lượng thiết kế).

Để xây dựng và quản lý CSDL chúng ta cần có các phần mềm được gọi là các Hệ Quản Trị CSDL (DBMS – Data Base Management System).

1.2 HỆ QUẢN TRỊ CSDL

1.2.1 Khái niệm Hệ Quản Trị CSDL

Hệ quản trị CSDL là một môi trường phần mềm cho phép xây dựng, quản lý và khai thác CSDL của các đề án tin học hóa. Ví dụ: Oracle, SQL Server, MS Access, My SQL, mSQL, Foxpro,…

Các hệ quản trị CSDL có nhiệm vụ hỗ trợ tích cực các nhà phân tích thiết kế CSDL cũng như những người quản trị, sử dụng, khai thác CSDL. Để thực hiện nhiệm vụ đó, hệ quản trị CSDL cần phải có các thành phần cho phép khai báo cấu trúc lưu dữ liệu và xây dựng các thao tác xử lý chúng.

1.2.2 Các thành phần của hệ quản trị CSDL

a. Ngôn ngữ định nghĩa dữ liệu (data definition language): Là phương tiện cho phép khai báo cấu trúc lưu trữ dữ liệu, khai báo các mối liên hệ giữa các loại dữ liệu, cũng như các quy tắc quản lý áp đặt trên các dữ liệu được lưu trữ.

b. Ngôn ngữ định nghĩa dữ liệu được xây dựng dựa trên 1 loại mô hình dữ liệu nào đó.Hiện nay, phần lớn các hệ quản trị CSDL đều dựa trên mô hình dữ liệu quan hệ.

c. Ngôn ngữ thao tác dữ liệu (data manipulation Language): Cho phép người sử dụng xây dựng cơ sở dữ liệu, thực hiện các thao tác cặp nhật (thêm, xóa , sửa) dữ liệu và khai thác dữ liệu với nhiều mục đích khác nhau.

d. Từ điển Từ điển dữ liệu (data dictionnary): Chứa thông tin về các thành phần cấu trúc CSDL (Các thuộc tính, các mối liên hệ...), các quan hệ, các ràng buộc dữ liệu. Thông qua từ điển dữ liệu, Các hệ quản trị CSDL cung cấp thêm các công cụ trợ giúp khai thác dữ liệu, lập hồ sơ sưu liệu (documentation) về cấu trúc của CSDL, tạo lập các CSDL mới...

Để mô tả dữ liệu ở mức quan niệm, người ta đưa ra nhiều loại mô hình

1.3 CÁC LOẠI MÔ HÌNH DỮ LIỆU

1.3.1 Định nghĩa

Mô hình dữ liệu là một tập hợp các khái niệm, các quy ước biểu diễn dữ liệu cần quản lý ở mức quan niệm. Thông qua mô hình dữ liệu, người thiết kế sẽ mô tả toàn cảnh CSDL được thiết kế bao gồm:

a. Các đối tượng, thực thể được quản lý.

b. Các mối quan hệ giữa các đối tượng.

c. Các ràng buộc dữ liệu thể hiện các quy tắc quản lý ảnh hưởng đến các đối tượng quản lý.

Có nhiều mô hình được đề nghị, ở đây chúng ta sẽ khảo sát 4 mô hình đặc trưng nhất là: Mô hình Phân cấp, Mô hình Mạng, Mô hình Quan hệ, Mô hình Thực thể kết hợp. Trong đó, mô hình Thực thể kết hợp thường được sử dụng nhiều nhất khi mô tả dữ liệu ở mức quan niệm trong qui trình thiết kế CSDL.

1.3.2 Mô hình thực thể kết hợp

Mô hình thực thể kết hợp (ERD - Entity Relation Diagram): Do Peter Chen đề xuất vào năm 1976. Đây là mô hình được sử dụng phổ biến để thiết kế CSDL ở mức quan niệm.

1.3.2.1 Các khái niệm chính trong mô hình gồm:

Thực thể (Entity): là một đối tượng cụ thể hay trừu tượng trong thế giới thực

Ví dụ: Nhân viên Nguyễn Văn A. Lớp CNTT1. Xe hơi có biển số 60E-1689

Thuộc tính (Attribute): Là các yếu tố thông tin để nhận biết được thực thể

Ví dụ: Họ tên, ngày sinh, nơi sinh ... của nhân viên

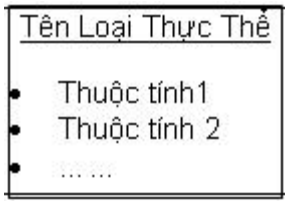
Loại thực thể (Entity Type): Tập các thực thể có chung các thuộc tính.

Ví dụ: Loại thực thể Nhân viên

Khóa của loại thực thể: Là tập thuộc tính mà giá trị của nó xác định duy nhất 1 thực thể

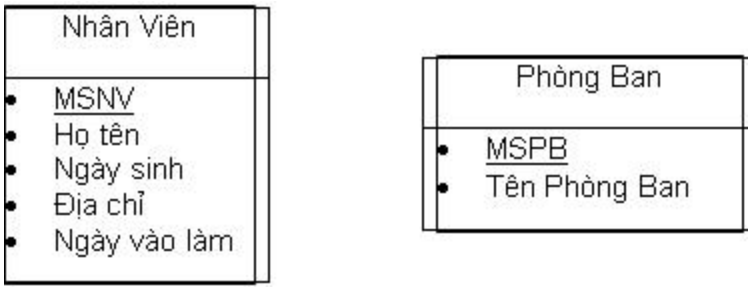
Ví dụ: Mã số nhân viên (MSNV) , Mã số phòng ban (MSPB)

Ký hiệu hình thức của loại thực thể:



Trong đó: Số thuộc tính phải lớn hơn hay bằng 1.

Ví dụ:



Loại mối kết hợp (Relationship): dùng diễn tả các liên hệ ngữ nghĩa giữa các loại thực thể.

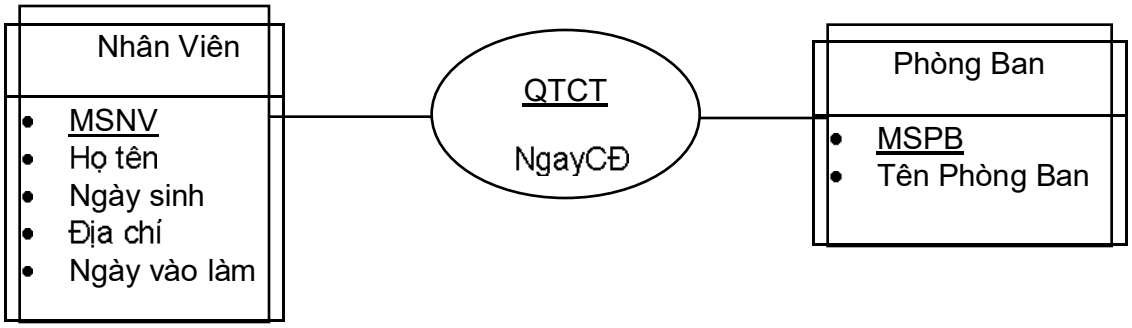
Ví dụ:

- Giữa thực thể Nhân viên và thực thể Phòng Ban có mối liên hệ ngữ nghĩa về quá trình công tác của một nhân viên ở phòng ban nào đó.
- Giữa thực thể Sinh viên và thực thể Lớp có mối liên hệ ngữ nghĩa: Mỗi sinh viên thuộc vào một Lớp học.

Ký hiệu hình thức của mối kết hợp:

Trong đó: Thuộc tính của mối kết hợp có thể không có

Ví dụ:



Thuộc tính của mối kết hợp: là thuộc tính chung của các loại thực thể tham gia mối kết hợp.

Bản số mỗi nhánh của mối kết hợp: là 1 bộ gồm 2 thành phần (min, max), đây là 1 ràng buộc toàn vẹn về số lượng tối thiểu và tối đa của 1 thực thể của nhánh đó tham gia vào các thể hiện của mối kết hợp.

Chú ý: Cận tối thiểu có thể là 0 hay 1

Nếu bài toán không qui định Min thì mặc định min =1.

Nếu bài toán không qui định Max thì đặt max = n với (n >1).

Thường xuất hiện các bản số:(0,1); (0, n); (1,1); (1,n)

1.3.2.2 Các bước thiết lập mô hình thực thể kết hợp:

B_1: Phân tích yêu cầu bài toán, chọn ra các thông tin cần quản lý, từ đó hình thành từ điển dữ liệu.

B_2: Tiến hành gom nhóm các thuộc tính theo các thực thể thật.

B_3: Xác định các mối kết hợp giữa các thực thể.

B_4: Xác định các thuộc tính của các mối kết hợp.

B_5: Xác định bản số của mối kết hợp.

Ví dụ: Xây dựng mô hình dữ liệu phục vụ cho việc quản lý điểm thi các môn học để tính điểm trung bình của từng học kỳ của sinh viên các lớp, với các quy tắc quản lý như sau: Mỗi môn học, sinh viên được phép thi 2 lần.

1.3.2.3 Một số qui định của mô hình thực thể kết hợp:

Không có các thuộc tính trùng tên giữa các thực thể.

Mọi thuộc tính của thực thể phải phụ thuộc trực tiếp vào khóa của thực thể

Thuộc tính có liên quan với các thực thể của một MKH thì đặt ở mối kết hợp

Ví dụ: Hãy lập mô hình ER cho bài toán quản lý siêu thị với các quy tắc như sau:

- Siêu thị chia thành nhiều khu vực, mỗi khu vực được đánh 1 mã số và có 1 tên, chuyên bán 1 loại mặt hàng nào đó, có 1 người quản lý và nhiều nhân viên làm việc trong khu vực đó.
- Mỗi mặt hàng trong siêu thị có thể cung cấp bởi nhiều nhà cung ứng. Mỗi nhà cung ứng cần lưu tên công ty, địa chỉ, số phone, số fax và các mặt hàng mà nhà cung ứng đó cung cấp.
- Mỗi khách hàng mua hàng thì chọn lựa hàng và sau đó đến quầy tính tiền. Ở quầy tính tiền sẽ in ra 1 hóa đơn gồm: số hóa đơn, ngày lập hóa đơn, tên khách mua, và danh sách các mặt hàng mua kèm theo đơn giá bán của các mặt hàng đó.
- Đối với khách hàng mua sỉ thì phải lưu trữ lại họ tên khách, địa chỉ, số phone, số fax.
- Các nhân viên làm việc được lưu trữ các thành phần: Msnv, họ tên, ngày sinh, địa chỉ, ngày vào làm việc, và khu vực mà nhân viên đó trực thuộc.

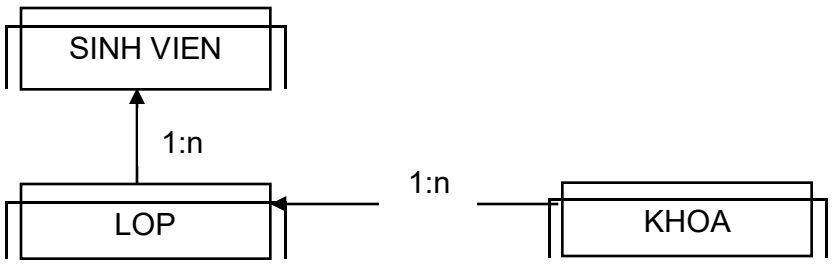
1.3.3 Mô hình dữ liệu Mạng:

Mô hình mạng (Network Data Model) sử dụng các khái niệm:

Loại mẫu tin (Record Type): thay cho khái niệm loại thực thể, chứa các mẫu tin mà mỗi mẫu tin là một thực thể.

Loại liên hệ (Set Type): là sự quan hệ ngữ nghiã giữa 1 loại mẫu tin chủ và một loại mẫu tin thành viên.

Ví dụ:



Ký hiệu 1:n: diễn tả ý nghĩa: một lớp có thể gồm nhiều sinh viên hay một khoa có thể có nhiều lớp.

Kiểu 1:1: ý nghĩa 1 mẫu tin chủ liên hệ với 1 mẫu thành viên,

Kiểu n:1: Mỗi thể hiện là 1 tập gồm nhiều mẫu tin chủ và 1 mẫu tin thành viên

Cách chuyển từ mô hình ER sang mô hình mạng:

Qui tắc 1: Mỗi thực thể hình thành 1 kiểu mẫu tin.

Qui tắc 2: Mối kết hợp thuộc loại (?,1) - (?,n) hay (?,1) - (?,1) thì hình thành 1 loại liên hệ chiều mũi tên đi từ (?,n) --> (?,1).

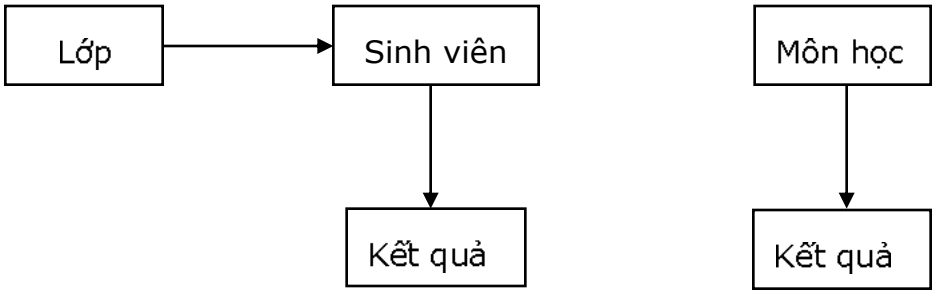
Qui tắc 3: Mối kết hợp thuộc loại (?,n) - (?,n) thì hình thành 1 kiểu mẫu tin mới, kiểu mẫu tin này chứa các thuộc tính khóa của 2 thực thể liên quan.

1.3.4 Mô hình dữ liệu phân cấp:

Giống như mô hình mạng, với các khái niệm Kiểu mẫu tin và Loại liên hệ. Nhưng 1 kiểu mẫu tin thành viên chỉ có thể phụ thuộc duy nhất 1 kiểu mẫu tin cha. Nghĩa là chỉ có những kiểu liên hệ: 1:1, 1:n. Do đó, mô hình được thể hiện như một rừng cây.

Những kiểu mẫu tin không có Cha là gốc của cây. Các kiểu mẫu tin cuối cùng không con tạo thành các lá của cây.

Ví dụ:



1.3.5 Mô hình dữ liệu Quan hệ

Mô hình dữ liệu quan hệ (Relational Data Model) Là mô hình được phát triển vào năm 1970 do Codd E.F tại công ty IBM đề xuất. Các đối tượng trong mô hình này chỉ gồm các bảng 2 chiều được gọi là các quan hệ (Relation Table) với các khái niệm:

Thuộc tính, khóa, Lược đồ Quan hệ, . Mô hình này có một cơ sở lý thuyết vững chắc nên là mô hình được phát triển rộng rãi nhất hiện nay.

Cách chuyển từ mô hình ER sang mô hình quan hệ:

Quy tắc 1: Chuyển mỗi thực thể thành 1 quan hệ

Quy tắc 2: Mối quan hệ (1,n) - (1,n) thành 1 quan hệ có khóa là khóa của các thực thể tham gia.

Quy tắc 3: Mối kết hợp (1,1) - (1,n) thì tạo thêm thuộc tính khóa của bên thực thể (1,1) vào trong thực thể phía (1,n) để làm khóa ngoại.

Quy tắc 4: Mối kết hợp (1,1) - (1,1) có thể gộp 2 thực thể thành một thực thể hoặc chọn khóa của một bên để thêm vào bên còn lại làm khóa ngoại

Ví dụ: Lớp(MSLop, TenLop)

SinhVien(MSSV, *MSLop*, HoTen, NgaySinh, Phai)

MonHoc(MSMon,TenMon)

KQHT(MSSV, *MSMon*, HocKy, DiemL1,DiemL2)

1.3.6 Mô hình dữ liệu hướng đối tượng

Mô hình dữ liệu hướng đối tượng(Object Oriented Data Model) dựa trên cách tiếp cận hướng đối tượng, với các khái niệm như: Lớp (Class) , Sự Kế thừa (Inheritence), Tính đóng gói (Encapsulation)... Nhưng hiện nay chưa được sử dụng rộng rãi một phần chưa có nhiều hệ quản trị CSDL hỗ trợ cài đặt theo mô hình này.

BÀI TẬP

Bài 1: Thiết lập mô hình quan niệm dùng quản lý việc cho mượn sách tại một thư viện (Xem tại chỗ hoặc mang về nhà) với các quy tắc quản lý như sau:

Sách gồm mã sách, tên, nguyên tác (tiếng Việt hoặc nước ngoài), tác giả. Sách được phân chia theo thể loại gồm MaTL và tên thể loại.

Đọc giả muốn mượn sách phải lập thẻ Đọc giả. Thẻ ghi nhận các thông tin gồm: MaDG, Ten DG, địa chỉ, ngày cấp, thông tin các sách đã mượn, ngày mượn, ngày trả. Hàng năm, Đọc giả phải đóng lệ phí để gia hạn thẻ mới được mượn sách, trên sổ có ghi thêm thông tin: Năm, Ngày nộp, Số tiền.

Bài 2: hãy thiết lập mô hình thực thể kết hợp và mô hình quan hệ cho bài toán quản lý siêu thị với các quy tắc quản lý như sau:

Siêu thị chia thành nhiều khu vực, mỗi khu vực được đánh một mã số và có một tên, chuyên bán một loại mặt hàng nào đó, có một người quản lý và nhiều nhân viên làm việc trong khu vực đó.

Mỗi mặt hàng trong siêu thị có thể được cung cấp bởi nhiều nhà cung cấp. Mỗi nhà cun gcấp cần lưu tên công ty, địa chỉ, số điện thoại và các mặt hàng mà nhà cung cấp đó có thể cunng cấp.

Mỗi khách hàng mua hàng thị chọn hàng sau đó đến quẩy tính tiền. Ở đây sẽ in ra hóa đơn gồm: số hóa đơn, ngày lập hóa đơn, tên khách mua, và danh sách các mặt hàng mua kèm theo đơn giá bán.

Đối với khách hàng thân thiết thì phải lưu mã số khách hàng, họ tên, địa chỉ và số điện thoại khách hàng.

Các nhân viên làm việc được lưu trữ các thành phần gồm: mã nhân viên, họ tên, ngày sinh, địa chỉ, ngày vào làm việc và khu vực mà nhân viên đó trực thuộc.

BÀI 2: CƠ SỞ DỮ LIỆU QUAN HỆ

2.1 CÁC KHÁI NIỆM CƠ BẢN

2.1.1 Thuộc tính

Là các đặc tính riêng biệt của mỗi đối tượng được quản lý.

Thuộc tính được xác định bởi:

Tên gọi: Thường được đặt một cách gợi nhớ, không nên đặt trùng tên 2 thuộc tính của 2 loại đối tượng khác nhau.

Ví dụ: Loại đối tượng SinhVien, và GiangVien đều có thuộc tính tên thì đặt TenSV, TenGV (tránh đặt thuộc tính là hoten cho cả hai loại đối tượng).

Qui ước: Trong lý thuyết, Dùng các chữ in hoa đầu tiên đại diện cho tên các thuộc tính. (VD: ABCD)

Kiểu dữ liệu: Vô hướng: Số, văn bản, Boolean; Có cấu trúc: Date/Time..

Miền giá trị (Domain): Ký hiệu Dom(A)

Ví dụ: Điểm có miền giá trị 0..10

$$Dom(Ngày\_Sinh) \subset [1,31]x[1,12]x[1900,2023]$$

Qui ước: Miền giá trị còn chứa thêm giá trị Rỗng (NULL) đặc trưng cho 1 trong 2 ngữ nghĩa: Chưa xác định tại thời điểm đang xét hay không thể xác định.

Ví dụ: TinhTrạngGiaĐình: chứa giá trị rỗng khi chưa xác định

2.1.2 Lược đồ quan hệ

Một lược đồ quan hệ (LĐQH) là một sự biểu diễn các đối tượng có chung các thuộc tính. Được biểu diễn bởi một cặp r=<U,F>. Trong đó U là tập thuộc tính, F là tập phụ thuộc hàm.

Ví dụ: Loại đối tượng tồn tại khách quan: Sinh viên, giảng viên, đơn đặt hàng...

Loại đối tượng trừu tượng: Thời khóa biểu, công nợ khách hàng...

Một LĐQH gồm có:

Một tên gọi: như quan hệ NhânVien, HọcSinh,…

Qui ước: Trong Lý thuyết, Dùng ký tự in hoa như: Q, R...

Một tập hợp hữu hạn các thuộc tính: A_1, A_2,..,A_n, ký hiệu:$\mathbf{Q^+}$ ={A_1,A_2,..,A_n}

Số thuộc tính của 1 LĐQH là gọi là **số ngôi** của LĐQH, ký hiệu: **Card(Q^+).**

LĐQH Q với tập thuộc tính A_1,...,A_n được ký hiệu: Q(A_1,…,A_n)

Một Tân từ, Ký hiệu:**||Q||**, dùng mô tả ý nghiã của LĐQH, các quy tắc, qui định giá trị và sự liên hệ của các thuộc tính.

Ví dụ: LĐQH KetQuaHT(MSSV, MSMon, HocKy, DiemL1,DiemL2)

Tân từ: Một bộ dữ liệu của kết quả học tập được xác định bởi 2 thuộc tính là (MSSV,MSMon). Mỗi môn học (MSMon) trong một học kỳ (HocKy) sinh viên được thi tối đa 2 lần.

Ví dụ:LĐQH KhachHang(MSKH,TenKH,DiaChi, SoDT)

Tân từ: Mỗi khách hàng có một mã số (MSKH) để phân biệt với các khách hàng khác, phải có tên (TenKH), địa chỉ (DiaChi), xác định và có thể có hay không có điện thoại (SoDT).

2.1.3 Bộ (Tuple / Record /Row)

Một bộ của một LĐQH Q là tập giá trị thuộc tính của một đối tượng thỏa mãn tân từ **||Q||** của lược đồ quan hệ đó**.**

q = (a_1, a_2,..,a_n) $\in$ Dom(a_1)xDom(a_2)x...xDom(a_n) và ||Q(q)|| = TRUE

Ví dụ: TRUONG_ĐH(Mã_Truong, Ten_Truong, ĐT)

q=('DHKHTN', 'Đại học Khoa Học Tự nhiên', '8124321')

2.1.4 Quan hệ (Relation)

Một quan hệ của một lược đồ quan hệ Q, ký hiệu T_Q, là một tình trạng, một thể hiện của lược đồ quan hệ Q ở một thời điểm nào đó.

Khi đó quan hệ T_Q chứa các bộ q có giá trị cụ thể thỏa mãn tân từ của lược đồ quan hệ Q:

$$T_Q = \{ q= (a_1,a_2,..,a_n) / a_i, Dom(a_i), ||Q(q)|| = TRUE \}$$

2.1.5 Lược đồ cơ sở dữ liệu

Một lược đồ cơ sở dữ liệu ***C*** là một tập hợp các lược đồ quan hệ , Ký hiệu:

$$\boldsymbol{C} = \{ Q_i \}^{t}_{i=1}$$

2.1.6 Tình trạng của lược đồ CSDL

Là tập hợp các quan hệ T_Qi của các lược đồ quan he Qi trong lược đồ cơ sở dữ liệu $C = \{ Qi \}^{t}_{i=1}$

Ký hiệu: $TC= \{ TQi\}^{t}_{i=1}$.

2.1.7 Siêu khóa và khóa của một quan hệ

Cho một lược đồ quan hệ $Q(A_1, A_2,...,A_n)$

Siêu khóa: Cho tập thuộc tính S, $S \subseteq Q^+$, S được gọi là Siêu khóa (Super Key) của lược đồ quan hệ Q nếu S có thể dùng làm cơ sở để phân biệt 2 bộ khác nhau tùy ý.

$\forall q_i, q_j \in TQ, q_i.S = q_j.S$ thì $q_i = q_j$

Qui định:

Một lược đồ quan hệ Q luôn luôn có ít nhất 1 siêu khóa và có thể có nhiều siêu khóa. Thuộc tính tham gia vào siêu khóa không được chứa giá trị rỗng

Ví dụ: KhachHang(MSKH, TenKH, DiaChi, SoDT)

có các siêu khóa: S_1={MSKH}, S_2={MSKH,TenKH}, S_3 = {MSKH,DiaChi},

S_4={MSKH,TenKH,DiaChi},…

Khóa chính (primary Key):

Là một siêu khóa tối tiểu không chứa bất kỳ một siêu khóa nào.

Ví dụ: K_1={MSKH}

Nếu có nhiều khóa chính, khi cài đặt nên chọn một khóa chính cho việc truy xuất đến các bộ. Các khóa còn lại được xem là các khóa phụ (Secondary key) hay khóa tương đương.

Thuộc tính khóa và thuộc tính không khóa: Các thuộc tính tham gia vào một khóa nào đó của LĐQH thì gọi là thuộc tính khóa, các thuộc tính không tham gia vào khóa nào gọi là các thuộc tính không khóa.

2.2 NGÔN NGỮ ĐẠI SỐ QUAN HỆ

Để truy xuất, khai thác lược đồ CSDL, Codd đã định nghĩa các toán tử chính để thao tác trên các quan hệ:

5 toán tử truyền thống trên tập hợp: Hội, Giao, Hiệu, Tích, Bù.

4 toán tử đặc thù cho các thao tác trên các quan hệ: Chiếu, chọn, kết, chia.

2.2.1 Các phép toán trên tập hợp

2.2.1.1 Phép hội (UNION)

Định nghĩa: Giả sử Q_1 và Q_2 là 2 LĐQH có cùng tập thuộc tính $\{A_1,A_2,...,A_n\}$

Hội 2 LĐQH Q_1, Q_2 hình thành 1 LĐQH mới có cùng tập thuộc tính $\{A_1,A_2,\dots,A_n\}$ và chứa tất cả các bộ của 2 quan hệ Q_1 và Q_2.

Ký hiệu: $Q_3 = Q_1 \cup Q_2$

$Q_3^+ = \{A_1,A_2,...,A_n\}$

$T_{Q3} = T_{Q1} \cup T_{Q2} = \{q / \ q \in T_{Q1}$ Hoặc $q \in T_{Q2}\}$

Ví dụ: Hội 2 Quan hệ Sinh Viên

SV_1

MaSV	Tên Sinh Viên	Phái
SV03	Nguyễn Hòa	Nam
SV04	Phạm Bằng	Nam
SV05	Nguyễn Thị Hoa	Nu

SV_2

MaSV	Tên Sinh Viên	Phái
SV07	Nguyễn Văn Minh	Nam
SV04	Phạm Bằng	Nam

$SV_1 \cup SV_2$

MaSV	Tên Sinh Viên	Phái
SV03	Nguyễn Hòa	Nam
SV04	Phạm Bằng	Nam
SV05	Nguyễn Thị Hoa	NU
SV07	Nguyễn Văn Minh	Phái

2.2.1.2 Phép giao (INTERSECTION)

Định nghĩa: Giả sử Q_1 và Q_2 là 2 LĐQH có cùng tập thuộc tính $\{A_1,A_2,...,A_n\}$

Giao của 2 lược đồ quan hệ Q_1, Q_2 hình thành 1 lược đồ quan hệ Q_3 có cùng tập thuộc tính $\{A_1,A_2,...,A_n\}$và chứa các bộ vừa thuộc Q_1 vừa thuộc Q_2.

Ký hiệu: $Q_3 = Q_1 \cap Q_2$

$Q_3^+ = \{A_1,A_2,...,A_n\}$

$T_{Q3} = T_{Q1} \cap T_{Q2} = \{q / q \in T_{Q1} \text{ và } q \in T_{Q2}\}$

Ví dụ: Giao 2 quan hệ sinh viên

$SV_1 \cap SV_2$

MaSV	Tên Sinh Viên	Phái
SV04	Phạm Bằng	Nam

2.2.1.3 Phép hiệu (MINUS)

Định nghĩa: Giả sử Q_1 và Q_2 là 2 LĐQH có cùng tập thuộc tính $\{A_1,A_2,...,A_n\}$

Hiệu của 2 lược đồ quan hệ Q_1, Q_2 hình thành 1 lược đồ quan hệ mới có cùng tập thuộc tính $\{A_1,A_2,...,A_n\}$và chứa các bộ thuộc Q_1 nhưng không thuộc Q_2

Ký hiệu: $Q_3 = Q_1 - Q_2$

$Q_3^+ = \{A_1,A_2,...,A_n\}$

$T_{Q3} = T_{Q1} - T_{Q2} = \{q / q \in T_{Q1} \text{ và } q \in T_{Q2}\}$

Ví dụ: Hiệu 2 quan hệ Sinh viên.

$SV_1 - SV_2$

MaSV	Tên Sinh Viên	Phái
SV03	Nguyễn Hòa	Nam
SV05	Nguyễn Thị Hoa	NU

2.2.1.4 Phép tích Descartes (CARTESIAN PRODUCT)

Định nghĩa: Giả sử Q_1 có tập thuộc tính $\{A_1,A_2,...,A_n\}$, Q_2 có tập thuộc tính $\{B_1,B_2,..,B_m\}$

Tích Descartes của 2 lược đồ quan hệ Q_1, Q_2 hình thành 1 lược đồ quan hệ mới Q_3

Có tập thuộc tính $Q_3^+ = \{A_1,A_2,...,A_n,B_1,B_2,...,B_m\}$

Chứa các bộ (q_1, q_2) trong đó $q_1 \in T_{Q1}$ và $q_2 \in T_{Q2}$

Ký hiệu: $Q_3 = Q_1 \times Q_2$

$Q_3^+ = \{A_1,A_2,...,A_n,B_1,B_2,...,B_m\}$

$T_{Q3} = \{(q_1, q_2) / \ q_1 \in T_{Q1} \text{ và } q_2 \in T_{Q2}\}$

Ví dụ: Tích 2 quan hệ Sinh Viên và Lớp

SV_2

MaSV	Tên Sinh Viên	Phái
SV07	Nguyễn Văn Minh	Nam
SV04	Phạm Bằng	Nam

LOP

MaLop	Tên Lớp
V01	Lập trình viên 1
M02	Mạng máy tính 2

SV_2 x LOP

MaSV	Tên sinh viên	Phái	MaLop	Tên Lớp
SV07	Nguyễn Văn Minh	Nam	V01	Lập trình viên 1
SV07	Nguyễn Văn Minh	Nam	M02	Mạng máy tính 2
SV04	Phạm Bằng	Nam	V01	Nguyễn Văn Minh
SV04	Phạm Bằng	Nam	M02	Nguyễn Văn Minh

2.2.1.5 Phép bù (COMPLEMENT)

Cho một lược đồ quan hệ $Q(A_1,A_2,\ldots,A_n)$; T_Q là một quan hệ định nghiã trên Q.

Miền giá trị hiện hành của một thuộc tính A_i trên quan hệ T_Q là tập giá trị phân biệt trên thuộc tính A_i hiện tồn tại trong Quan hệ T_Q.

Ký hiệu: MGThh(A_i); !Q

Phép bù trên lược đồ quan hệ Q tạo thành một lược đồ quan hệ Q' sao cho:

$Q'^+ = Q^+$ và

$T_{Q'} = \{ q=(a_1, a_2, .., a_n) / a_i \in MGThh(A_i)$ và $q \notin T_Q \}$

Ví dụ:

Q

A	B
a_1	b_1
a_2	b_1
a_3	b_2
a_1	b_2

!Q

A	B
a_2	b_2
a_3	b_1

2.2.2 Các phép toán trên quan hệ

2.2.2.1 Phép chiếu (PROJECTION)

a. Phép chiếu của một bộ $q = (a_1,a_2,..a_n)$ lên tập thuộc tính $(A_{s1}, A_{s2},..,A_{sm}) \subseteq Q^+$ là phép trích ra từ bộ q một bộ con $s = (a_{s1}, a_{s2},..,a_{sn})$ sao cho $a_{si} \in Dom(A_{si})$

Ký hiệu: $s = q.A_{s1},A_{s2},..,A_{sm}$

Ví dụ: q.MSSV,MSLop = (01,A01)

b. Chiếu một quan hệ lên tập thuộc tính X:

Giả sử X là tập hợp con của tập thuộc tính của lược đồ quan hệ Q ($X \subseteq Q^+$)

Phép chiếu Q lên tập thuộc tính X sẽ tạo thành một lược đồ quan hệ Q'

Có tập thuộc tính X và

TQ' = {q' / $\exists q \in TQ$, q'.X = q.X}

Ký hiệu: $\Pi_X(Q)$ hoặc Q[X].

Ví dụ: SinhVien(MSSV, MSLop, HoTen, NgaySinh, Phai)

Câu hỏi: Liệt kê danh sách sinh viên gồm các thông tin: MSSV, HoTen

$$\Pi_{MSSV,HoTen}\ (SinhVien) \quad \text{hoặc} \quad SinhVien[MSSV,HoTen]$$

2.2.2.2 Phép chọn (SELECTION)

Định nghĩa: Cho lược đồ quan hệ $Q(A_1,A_2,...,A_n)$, E là biểu thức logic được phát biểu trên các thuộc tính của Q^+. Phép chọn trên Q theo điều kiện E sẽ hình thành 1 lược đồ quan hệ mới Q' có cùng tập thuộc tính với Q và chứa các bộ thuộc Q và thỏa điều kiện E.

TQ'= { q' / q' $\in$ Q và q' thỏa E}

Ký hiệu: $Q' = \sigma_E(Q)$ hoặc Q' = Q: E

Ví dụ: Chọn trên quan hệ SinhVien(MSSV, MSLop, HoTen, NgaySinh, Phai) những sinh viên sinh trước năm 1975

(SinhVien): (Ngaysinh< '1/1/1975') hay $\sigma_{Ngaysinh< '1/1/1975'}$(SinhVien)

Các điều kiện được sử dụng các phép toán so sánh: =, >, <, >=,<= , <> và có thể kết hợp các điều kiện đơn giản bằng các phép toán logic: and, or

Ví dụ: KetQuaHT(MSSV, MSMon, HocKy, DiemL1,DiemL2)

Chọn ra những học sinh có điểm thi lần 1 hoặc điểm thi lần 2 lớn hơn 8.

E = (DiemL1 >8) OR (DiemL2 >8)

$T_{Q'}=\sigma_{(E)}$(KetQuaHT);

$T_{Q'}$ = (KetQuaHT):((DiemL1>8)OR(DiemL2>8))

Kết quả thực hiện sẽ cho các bộ thỏa điều kiện E

2.2.2.3 Phép kết (JOIN)

Định nghĩa: Cho 2 lược đồ quan hệ $Q_1(A_1,...,A_n)$ và $Q_2(B_1,..,B_m)$ có $Dom(A_i) = Dom(B_j)$, θ là 1 phép so sánh trên miền giá trị của A_i. Phép kết Theta (θ) giữa Q_1 và Q_2 dựa trên A_i và B_j sẽ tạo thành một lược đồ quan hệ Q_3 có:

Tập thuộc tính bằng tập thuộc tính của tích Descartes Q_1xQ_2 và

Chứa các bộ thỏa điều kiện so sách $A_i \theta B_j$.

Nghĩa là: $Q_3 = (Q_1 \times Q_2)$: Ai θ Bj = { (q_1,q_2) / $q_1 \in Q_1$, $q_2 \in Q_2$, $q_1.A_i \theta q_2.B_j$ }

Ký hiệu: $Q_3 = Q_1 \overset{A_i \theta B_j}{\bowtie} Q_2$

Ví dụ: Giả sử có hai lược đồ $Q_1(ABC)$ và $Q_2(DE)$. Xác định quan hệ kết dựa trên 2 quan hệ T_{Q1}, T_{Q2} với phép so sánh là ' >= ' dựa trên 2 thuộc tính B và D (D>=B)

Q_1

A	B	C
1	2	3
4	5	6
7	8	9

Q_2

D	E
3	1
6	2

$Q_3 = Q_1 \bowtie Q_2$ điều kiện là chọn các bộ thỏa D>=B

Q_3

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

Các trường hợp đặc biệt:

Kết bằng: Nếu θ là phép so sánh với phép toán bằng (=)

Kết tự nhiên: Là phép nối kết bằng trên các thuộc tính trùng tên trong Q_1 và Q_2. Quan hệ kết quả không giữ các thuộc tính trùng.

Ký hiệu: $Q_3 = Q_1 \bowtie Q_2$

Ví dụ: Nối kết thông tin của SinhVien và KetQua dựa trên MSSV.

SinhVien(MSSV, MSLop, HoTen, NgaySinh, Phai)

KetQuaHT(MSSV, MSMon, HocKy, DiemL1,DiemL2)

BangĐiem = (SinhVien $\bowtie$ KetQua)

Có thể mở rộng phép kết dựa trên việc so sánh nhiều cặp thuộc tính với các phép toán And, Or

Kết trái: Là phép kết nối ưu tiên lấy hết các dòng dữ liệu từ table bên trái, các dòng dữ liệu tương tứng của table bên phải nếu không có thì được nhận giá trị NULL

Ký hiệu: $Q_3 = Q_1 |\!\times Q_2$

Kết phải: Là phép kết nối ưu tiên bên phải, table kết quả sẽ lấy hết các dòng của table bên phải của phép kết nối. Sau đó các dòng tương ứng của table bên trái sẽ được lấy, nếu dòng nào không có giá trị sẽ nhận giá trị NULL.

Ký hiếu: $Q_3 = Q_1 \times\!| Q_2$

2.2.2.4 Phép chia (DIVISION)

Định nghĩa: Cho 2 lược đồ quan hệ $Q_1(A_1,A_2,..,A_p,A_{p+1},..,A_n)$ và $Q_2(A_1,A_2,..,A_p)$, $Q_2^+ \subset Q_1^+$. Phép chia Q_1 cho Q_2 sẽ tạo thành 1 lược đồ quan hệ $Q_3(A_{p+1},..,A_n)$ chứa các bộ mà khi ghép nối những bộ đó với các bộ của Q_2 sẽ cho ra 1 bộ của Q_1.

$Q_3 = \{q / q_2 \in Q_2, q_1 \in Q_1, q = q_1.Q_3^+ \text{ và } q_2 = q_1.Q_2^+ \}$

Ký hiệu: $Q_3 = Q_1 \div Q_2$

Công dụng: Chọn ra những bộ con có liện hệ đồng thời với tất cả các bộ con khác

Ví dụ: Chọn ra những sinh viên cùng có điểm môn CSDL là 8 và điểm CTDL là 7

Q_3 = KetQuaHT ÷ Q_2

KetQuaHT

Mssv	MsMon	Điểm
01	Csdl	8
01	Ctdl	7
02	Csdl	8
02	Ctdl	9
03	Csdl	8
03	Ctdl	7

Q_2

MsMon	Điểm
csdl	8
ctdl	7

Q_3

Mssv
01
03

Một số ví dụ: Xét 1 lược đồ CSDL như sau:

HH(MSHH, TenHH, DVT, SoTon, ĐG)

KH(MSKH, TenKH, DC, DT, Fax)

DDH(SoDDH, NgayLap, MSKH)

CTDDH(SoDDH, MSHH, SL, Dg, GiamGia)

Câu hỏi 1: Cho biết danh sách các khách hàng gồm (TenKH, DC, DT) đã đặt mặt hàng có mã số 01

$\Pi_{TenKH, DC, DT}((DDH \bowtie KH) \bowtie \sigma_{CTDDH.MSHH = "01"}(CTDDH))$

Câu hỏi 2: Cho biết các CTDDH của DDH có số đơn đặt hàng là 111 gồm: TenHH, DVT, SL, Dg, GiamGia.

$\Pi_{TenHH, DVT, SL, Dg, GiamGia}(HH \bowtie \sigma_{CTDDH.SoDDH = "111"}(CTDDH))$

BÀI TẬP

Cho CSDL sau:

Khách_Hàng(MãKH,TênKH,ĐịaChỉ,ĐiệnThoại)

Diễn giải: Trong quan hệ khách hàng gồm có các thông tin: Mã khách hàng(MãKH) để phân biệt các khách hàng với nhau, tên khách hàng(TênKH), địa chỉ của khách hàng (ĐịaChỉ) và điện thoại của khách hàng (ĐiệnThoại).

Đơn_Đặt_Hàng(Số_ĐĐH,NgàyĐặt,MãKH)

Diễn giải: Trong quan hệ đơn đặt hàng gồm các thông tin: Số đơn đăt hàng(Số_ĐĐH) để phân biệt các đơn đặt hàng, ngày đặt (NgàyĐặt) và mã khách hàng (MãKH).

CT_ĐĐH(Số_ĐĐH,MãHg,SL,ĐG_Bán)

Diễn giải: Trong quan hệ chi tiết đơn đặt hàng (CT_ĐĐH) gồm các thông tin: Số đơn đặt hàng (Số_ĐĐH) và mã hàng (MãHg) để xác định hai bộ dữ liệu khác nhau, số lượng bán (SL) và đơn giá bán (ĐG_Bán) mặt hàng đó.

Mặt_Hàng(MãHg,TênHg,SL_Tồn,ĐG_Mua)

Diễn giải: Trong quan hệ mặt hàng gồm các thông tin: mã hàng (MãHg) để phân biệt các mặt hàng, tên hàng (TênHg), số lượng tồn (SL_Tồn) của mặt hàng và đơn giá mua mặt hàng đó (ĐG_Mua).

Câu hỏi: Hãy trả lời các câu hỏi sau bằng ngôn ngữ đại số quan hệ.

a. Cho biết tên khách hàng đã đặt mua tất cả các mặt hàng?

b. Cho biết tên các mặt hàng mà chưa có ai đặt mua?

c. Cho biết tên các khách hàng đã đặt hàng trong ngày 20/11/2011?

d. Cho biết số lượng các mặt hàng đã được đặt trong ngày 12/05/2013?

e. Cho biết tên khách hàng đã đặt mua hơn 10 mặt hàng (tại nhiều chi tiết khác nhau)?

BÀI 3: NGÔN NGỮ HỎI CÓ CẤU TRÚC

SQL (Structured Query Language) là ngôn ngữ truy vấn dữ liệu của công ty IBM phát triển vào thập niên 70, Được ANSI công nhận và phát triển thành một ngôn ngữ chuẩn quốc tế vào năm 1986.

Khởi đầu SQL được gọi là SEQUEL (Structured English QUEry Language). Bản cài đặt đầu tiên: SYSTEM R của IBM được thực hiện vào năm 1978.

Chuẩn mới (SQL-99) được cài đặt trong hầu hết các hệ quản trị CSDL cho phép người sử dụng: Truy vấn dữ liệu, cặp nhật dữ liệu, thao tác với các đối tượng dữ liệu như Table, Index, Procedure,…

3.1 TRUY VẤN DỮ LIỆU VỚI CÂU LỆNH SELECT

3.1.1 Cú pháp:

SELECT [predicate] { * | table.* | <Danh sách Field>

FROM <Danh sách Quan hệ>

[WHERE <Điều kiện>]

[GROUP BY <Danh sách Thuộc tính>]

[HAVING <Điều kiện của nhóm>]

[ORDER BY Field1 [ACS|DESC],...]

Công dụng: Tạo ra 1 quan hệ mới từ các quan hệ ghi sau từ khóa From.

predicate: [ALL | DISTINCT | DISTINCTROW | [TOP n [PERCENT]]]

<Danh sách Field>: Thể hiện phép chiếu trên tập thuộc tính. Tên 1 thuộc tính của quan hệ tham gia: [table].Field

Mở rộng: Có thể là một biểu thức: <Biểu thức> As <Alias> Trong đó, sử dụng các toán tử số học như: ^, *, /, \ (chia nguyên), MOD, +, -.

Cho lược đồ CSDL sử dụng trong các ví dụ như sau:

HH(MSHH, TenHH, DVT, SoTon, ĐG)

KH(MSKH, TenKH, DC, DT, Fax)

DDH(SoDDH, NgayLap, MSKH)

CTDDH(SoDDH, MSHH, SL, Dg, GiamGia)

Ví dụ: Hiện danh sách các mặt hàng gồm các thông tin: MSHH, TenHH, SoTon

```
SELECT MSHH, TenHH, SoTon FROM HH;
```

Ví dụ: Tính tiền bán từng mặt hàng cho từng hóa đơn trong CTDDH:

```
SELECT SoDDH,MSHH,SL,Dg,GiamGia,SL*DG-GiamGia as ThanhTien
FROM CTDDH;
```

3.1.2 Phát biểu Order By

Sắp thứ tự các bộ theo giá trị tăng hoặc giảm của các thuộc tính xuất hiện sau từ khóa Order by (Giá trị mặc định là tăng dần).

Ví dụ: Tính tiền bán từng mặt hàng trong CTDDH và sắp thứ tự giảm dần theo thành tiền bán hàng.

```
SELECT SoDDH, MSHH, SL, Dg,GiamGia, SL*DG-GiamGia as ThanhTien
FROM CTDDH
Order By ThanhTien Desc
```

3.1.3 Phát biểu Where

Dùng chọn ra những bộ thỏa một hoặc nhiều điều kiện cho trước

Điều kiện: Được thiết lập với các toán tử:

a. So sánh: =, <>, >, <, >=, <=.

Ví dụ: Hiện danh sách các mặt hàng có số lượng tồn kho < 10 gồm các thông tin: MSHH, TenHHH, SoTon:

```
SELECT MSHH, TenHHH, SoTon FROM HangHoa Where SoTon < 10;
```

b. Logic: And, Or, Not.

Ví dụ: Hãy Liệt kê những DDH được lập trong tháng 10 năm 1999:

```
SELECT * FROM DDH
Where NgayLap>='1/10/1999' and NgayLap <= '30/10/1999';
```

Ví dụ: Hiển thị danh sách các CTDH với các thông tin: SoDDH, TenHang, SL, GiáBán,ThanhTien trong đơn đặt hàng có số là 10001

```
SELECT SoDDH, TenHH, SL, DG as GiaBan, GiaBan* SL As ThanhTien
FROM CTDH, DDH
Where CTDDH.SoDDH = DDH.SoDDH and DDH.SoDDH = '10001';
```

c. Thuộc miền giá trị: <biểu thức> [Not] Between Value1 And Value2

Chức năng: So sánh giá trị biểu thức thuộc miền giá trị [Value1, Value2].

Ví dụ: Danh sách các mặt hàng có đơn giá từ 20 đến 50

```
Select * From HH Where DG Between 20 And 50;
```

d. Thuộc tập giá trị: <biểu thức> [Not] In (Value Set)

Chức năng: Kiểm tra giá trị biểu thức thuộc tập giá trị cho trước.

Ví dụ: Danh sách các khách hàng có mã số 3, 6, 9

```
Select * From KH Where MSKH in (3,6,9);
```

Ví dụ: Danh sách các khách hàng đặt hàng ngày 1/5/2000

```
Select Distinct *
From KH
Where MSKH in (Select MSKH From DDH Where NgayLap = '1/5/2000')
```

Ví dụ: Danh sách các mặt hàng không bán được

```
Select * From HH Where MSHH not in (Select MSHH From CTDH)
```

e. Not Exists (Subquery): Chọn ra các bộ có giá trị field thỏa điều kiện trong SubQuery

Ví dụ: Cho biết danh sách các mặt hàng không bán được

```
Select * From HH as A
Where Not Exists (Select * From CTDH as B Where A.MsHH = B.MsHH)
```

Biểu thức <comparison> ALL (subquery): Kiểm tra giá trị thỏa phép so sánh với tất cả các giá trị subquery.

Ví dụ: In ra SoDDH và SL mua mặt hàng số 3 của các DDH mua mặt hàng này là nhiều nhất.

```
Select A.SoDDH, A.SL
From CTDH  As A
Where A.MsHH=3 and A.SL >= ALL
                    (Select B.SL From CTHH As B Where B.MSHH = 3)
```

f. Đối sánh mẫu với toán tử Like

Chức năng: So sánh giá trị biểu thức với 'Mẫu'. Sử dụng ký tự đại diện %, _ Tróng đó % thay thế cho nhiều ký tự và _ thay thế cho một ký tự

Ví dụ: Chọn ra danh sách khách hàng họ 'Nguyễn'

```
Select * From KH Where TenKH Like N'Nguyễn%'
```

g. Biểu thức <comparison> ANY | SOME (subquery)

Kiểm tra giá trị thỏa phép so sánh với ít nhất 1 giá trị tạo bởi subquery.

Ví dụ: hiện tên khách hàng mua hàng ít nhất một lần trong năm 2014

```
Select TenKH
From KH
Where mskh any (select mskh from DDH where year(ngaylap)=2014)
```

3.1.4 Phát biểu Group By và Having

Dùng tổng hợp, thống kê dữ liệu theo từng nhóm được hình thành dựa trên giá trị của các thuộc tính xuất hiện sau từ khóa Group by. Trong đó, phát biểu Having dùng chọn ra những nhóm thỏa điều kiện nào đó.

Các hàm tổng hợp theo nhóm:

Sum(Biểu thức):Tính tổng giá trị các biểu thức số trên các bộ thuộc nhóm

Max(Biểu thức): Chọn ra giá trị lớn nhất trên các bộ thuộc nhóm

Min(Biểu thức): Chọn ra giá trị thấp nhất trên các bộ thuộc nhóm

Avg(Biểu thức): Tính trung bình cộng giá trị biểu thức trên các bộ thuộc nhóm

Count(* | Biểu thức | Distinct Biểu thức): Đếm các bộ trong từng nhóm.

Count(*): Đếm tất cả các bộ kể cả những bộ có tất cả các thuộc tính đều = NULL

Count(biểu thức): Chỉ đếm những bộ mà có giá trị biểu thức khác NULL.

Count(Distinct Biểu thức): Chỉ đếm những bộ mà có giá trị biểu thức khác nhau và khác NULL.

Ví dụ: Đếm tổng số DDH của từng khách hàng:

```
Select MSKH, Count(*) Form DDH Group By MSKH
```

Ví dụ: Liệt kê danh sách DDH và số tiền của mỗi DDH:

```
Select SoDDH, Sum(DG*SL) From CTDH Group By SoDDH
```

Ví dụ: Liệt kê danh sách DDH và số tiền của mỗi DDH nhưng chỉ hiện những DDH có tổng trị giá > 5000.

```
Select SoDDH, Sum(DG*SL) From CTDH
Group By SoDDH Having Sum(DG*SL)>5000
```

3.1.5 Thứ tự dịch của lệnh Select:

From --> Where --> Group By --> Having --> Select --> Order

Ví dụ: Hiển thị danh sách các khách hàng có tổng trị giá mua trên 10 triệu

```
Select KH.MSKH, TenKH, Sum(DG*SL) As TongTien
From KH, CTDH, DDH
Where KH.MSKH = DDH.MSKH and DDH.SoDDH=CTDH.SoDDH
Group By KH.MSKH, TenKH
```

Having Sum(DG*SL)>100000

3.2 CÁC LỆNH ĐỊNH NGHĨA DỮ LIỆU:

3.2.1 Lệnh Tạo Cấu Trúc Quan Hệ Mới:

Cú pháp: CREATE TABLE tênQHệ

(field1 type [(size)] [NOT NULL] [index1]

[, field2 type [(size)] [NOT NULL] [index2] [, ...]]

[, CONSTRAINT multifieldindex [, ...]])

Một số kiểu dữ liệu thường sử dụng:

Char(n)	Kiểu chuỗi gồm n ký tự, mặc định là 1
Integer	Số nguyên 4 byte
Smallint	Số nguyên 2 byte
Bigint	Số nguyên lớn 8 byte
Real	số thực 4 byte
Float, Double	số thực 8 byte
Date, Time	Ngay giờ
Boolean	1 Bit
Varchar(v)	Kiểu chuỗi gồm n ký tự
Nvarchar(n)	Kiểu chuỗi gồm n ký tự có sử dụng unicode

Ví dụ: Tạo quan hệ Nhanvien

Create Table Nhanvien (msnv Char(5), HoTen Char(30), NS Date, Nu Boolean, HsLg Float);

Các biểu thức ràng buộc:

Ràng buộc liên quan đến 1 thuộc tính:

CONSTRAINT name {PRIMARY KEY | UNIQUE |: Khóa chính, không lặp lại giá trị

REFERENCES foreigntable [(foreignfield1, foreignfield2)]}: Ràng buộc tồn tại

Not Null: Không thể rỗng

Check (điều kiện): Ràng buộc miền giá trị. Điều kiện có thể là một biểu thức hoặc danh sách giá trị.

Ví dụ:

Create Table NV (msnv varchar(5) Primary Key, HoTen Nvarchar(30) Not Null, NgaySinh date, Phái Boolean, Hệsốlương Float);

Ràng buộc quan hệ: Liên quan đến nhiều thuộc tính.

Primary Key(f1,..,fn):

Unique (f1,..,fn)

Foreign Key(f1,..fn) References tenQH2 [(fg1,..,fgn)]

Check (điềukiện)

Ví dụ: Create Table NV (msnv Char(5) Not Null, HoTen Char(30) Not Null, NS Date, Nu Boolean, HsLg Float, Primary Key(msnv));

Access: *CONSTRAINT name {PRIMARY KEY (primary1[, primary2 [, ...]])* |

UNIQUE (unique1[, unique2 [, ...]]) |

FOREIGN KEY (ref1[, ref2 [, ...]]) REFERENCES foreigntable [(foreignfield1 [, foreignfield2 [, ...]])]}

Ví dụ: Create Table NV (msnv Char(5) Not Null C*onstraint rb1* Primary Key(msnv), HoTen Char(30) Not Null, NS Date, Nu Boolean, HsLg Float, MsPB Char(5) Not Null Constraint References PhBan (MP));

Hay có thể viết:

Create Table NV (msnv Char(5) Not Null, HoTen Char(30) Not Null, NS Date, Nu Boolean, HsLg Float, MsPB Char(5) Not Null, Constraint RB1 Primary Key(msnv), Constraint RB2 Foreign Key (MsPB) References PhBan (MP));

3.2.2 Lệnh Xóa Một Quan Hệ:

DROP TABLE TênQH;

3.2.3 Lệnh Thay Đổi Cấu Trúc Của Quan Hệ:

Cú pháp:

a. Thêm thuộc tính:

ALTER TABLE TenQH ADD COLUMN fieldname type[(size)]

[NOT NULL] [CONSTRAINT <index>];

b. Xóa thuộc tính:

ALTER TABLE TênQH DROP COLUMN fieldname;

Chú ý: Lệnh không hợp lệ nếu thuộc tính được xóa là thuộc tính khóa.

3.2.4 Các Lệnh cập nhật dữ liệu:

a. Thêm bộ mới vào bảng:

cp1: INSERT INTO TenQH VALUES (value1[, value2[, ...])

Ví dụ: Thêm một nhân viên mới

INSERT INTO KH VALUES (5, 'Anh', #1/5/78#)

cp2: Thêm 1 bộ với các giá trị không đầy đủ (Các Field không liệt kê sẽ có giá trị NULL). INSERT INTO TenQH [(field1[, field2[, ...]])]

VALUES (value1[, value2[, ...]);

Ví dụ: Thêm 1 nhân viên mới chỉ biết Msnv, Tên, Giới tính

INSERT INTO KH (Msnv, Ten, GT) VALUES (5, 'Anh', 'Nam')

cp3: Thêm vào QH các bộ là kết quả của câu lệnh Select.

INSERT INTO TenQH [(field1[, field2[, ...]])]

SELECT [source.]field1[, field2[, ...] FROM tableexpression...;

Ví dụ: Chèn các dòng đặt hàng của DDH = 12 vào bảng CTGiaoHang

INSERT INTO CTGiaoHang (SoDDH, MSHH,SL,DG)

SELECT SoDDH, MSHH, SL, DG FROM CTDH

WHERE SoDDH = 12;

b. Xóa các bộ của một quan hệ

DELETE [tênQH.*] FROM TênQH [WHERE <điều kiện>];

Ví dụ: Xóa các DDH trước ngày 1/1/2000

DELETE FROM DDH WHERE NgayLHD <= '1/1/2000'

c. Cập nhật giá trị của các bộ

UPDATE tênQH SET <Field1>=<BT1>,<Field2>=<BT2>...

WHERE <ĐiềuKiện>;

Ví dụ: Cập nhật đơn giá của sản phẩm 2,4,6 lên 5%

UPDATE HH SET DonGia = [DonGia]*1.05 WHERE Mshh In (2,4,6);

d. Hội các bộ của các quan hệ

[TABLE] $query_1$ UNION [ALL] [TABLE] $query_2$ UNION [ALL] [TABLE]… $query_n$ [...]

Ví dụ: Tạo quan hệ chứa danh sách các DDH tháng 1/99 và tháng 4/99

Select * from DDH

Where ngaylap between '1/1/99' and '31/1/99'

union

Select * from DDH

Where ngaylap between '1/4/99' and '30/4/99'

BÀI TẬP

Bài 1: Địa Lý Việt Nam Cho CSDL địa lý có cấu trúc như sau:

Tinh_TP(T_TP, DT, DS, Ten_T)

Mien(T_TP,MN)

BG(Nuoc, T_TP)

Lang_Gieng(T_TP, T_TP_LG)

Hãy trả lời các câu hỏi sau bằng ngôn ngữ SQL

1. Cho biết dân số cùng tên tỉnh của các tỉnh thành phố có diện tích 7000 km2.
2. Cho biết dân số cùng với tên tỉnh của các tỉnh miền Bắc
3. Cho biết mã các nước biên giới của các tỉnh miền Nam
4. Cho biết diện tích trung bình của các tỉnh
5. Cho biết mật độ dân cư cùng tên tỉnh của tất cả các tỉnh
6. Cho biết tên những tỉnh có diện tích có 1 diện tích lớn hơn tất cả các tỉnh láng giềng của nó.
7. Cho biết tên những tỉnh mà ta có thể đến bằng cách đi từ TP HCM xuyên qua 3 tỉnh khác nhau và cũng khác với điểm xuất phát, nhưng láng giềng với nhau.

Giải:

Câu 1: Select Ten_T, DS From Tinh_TP Where DT>=7000

Câu 2: Select Ten_T, DS From Tinh_TP, Mien

Where Mien.T_TP= Tinh_TP.T_TP And MN="Bắc";

Câu 3: Select Nuoc From BG, Mien

Where Mien.T_TP= BG.T_TP And MN="Nam"

Câu 4: Select AVG(DT) From Tinh_TP;

Câu 5: Select Ten_T, DS / DT As MDDC From Tinh_TP;

Câu 6:

```
SELECT a.TenTinh, a.DT FROM Tinh_TP AS a
WHERE a.DT>All (Select max(b.DT) From Tinh_TP b, LG c
Where b.T_TP = c.T_TP_LG and a.T_TP=c.T_TP);
```

Bài 2: Thể Thao Đội Cho CSDL có cấu trúc như sau:

Thuộc tính:

Tên tắc	**Diễn giải**	**Miền giá trị**
MsCLB	Mã số câu lạc bộ	Nguyên(1,100)
DcCLB	Địa chỉ CLB	Văn bản
MsDoi	Mã số đội	Nguyên(1,50)
MsLT	mã số lứa tuổi	{LT1<LT2<LT3<LT4<LT5}
Phai	Phái, giới tính	Mã {"Nam", "Nữ"}
MsTD	Mã số trận đấu	Nguyên
GioTD	Giờ bắt đầu trận đấu	Time
NgayTD	Ngày diễn ra trận đấu	Date
MsPh	Mã số phòng thi đấu	Nguyên
DcPh	Địa chỉ phòng	Văn bản
SoSan	Số lượng sân của 1 phòng	Nguyên (1,5)
TGTĐ_LT	Thời gian thi đấu phù hợp cho 1 lứa tuổi Nguyên (0,20)	(đvt là phút)
MsVĐV	Mã số vận động viên	Nguyên
TenVDV	Tên vận động viên	Văn bản
DcVDV	Địa chỉ vận động viên	Văn bản

Quan hệ:

1. **CLB**(MsCLB , DcCLB)

Tân từ: Mỗi CLB có 1 mã số (MSCLB) để phân biệt với các CLB khác, và địa chỉ của CLB (DcCLB)

2. DOI(MsCLB, MsDoi, MsLT, Phai)

Tân từ: Mỗi đội có 1 mã số đội (MsDoi) để phân biệt với các đội khác của cùng một Câu lạc bộ (MsCLB); Mỗi đội thuộc về 1 lứa tuổi duy nhất (MsLT) và một phái (nam hay nữ). Tất cả các vận động viên của đội phải có cùng lứa tuổi của đội hoặc thấp hơn lứa tuổi của đội.

3. TD(MsTD, MsPh, GioTD, NgayTD)

Tân từ: Mỗi trận đấu có 1 mã số (mstd) để phân biệt với những trận đấu khác, diễn ra trong 1 phòng, vào 1 ngày (NgayTD) và bắt đầu ở một giờ (GioTD) đã quy định.

4. DOI_TD(MsTD, MsDoi, MsCLB)

Tân từ: Mỗi trận đấu là một cuộc gặp gỡ giữa 2 đội. Cả 2 đội phải thuộc cùng 1 lứa tuổi và cùng một phái.

5. PHÒNG(MsPh, DCPh, SốSân)

Tân từ: Mỗi phòng có 1 mã số (MSPh) để phân biệt với các phòng khác, có 1 địa chỉ và 1 số lượng sân (SốSân) nhất định, nơi đó có thể diễn ra các trận đấu.

6. LỨA_TUỔI(MsLT, TGTĐ_LT)

Tân từ: Mỗi lứa tuổi được phân biệt bởi mã lứa tuổi (MsLT) và có thời gian trận đấu dành cho lứa tuổi đó.

7. VĐV(MsVDV, TenVDV, DcVDV, MsCLB, Phai, MsLT)

Tân từ: Mỗi vận động viên có 1 mã số (MsVDV) dùng để phân biệt với các VDV khác, có 1 tên, 1 địa chỉ, thuộc 1 CLB, 1 phái và 1 lứa tuổi.

8. VĐV_ĐỘI(MsVDV, MsDoi)

Tân từ: Mỗi VĐV đăng ký chơi ở 1 đội. Họ có thể đăng ký vào nhiều đội khác nhau.

Yêu cầu:

1. Hãy xác định khóa của các quan hệ từ tân từ của các quan hệ

2. Thực hiện các câu hỏi sau:

a. Danh sách tên các VĐV của CLB có mả số 45

b. Tên tất cả các VĐV của đội số 3 của CLB mã số 27

c. Số lượng các trận đấu LT1 diễn ra ngày 16/06/90

d. Mã số các CLB và mã số các đội trong đó có quy tụ các VĐV thuộc 1 lứa tuổi nhỏ hơn lứa tuổi của đội.

e. Địa chỉ và mã số các CLB có 1 hay nhiều đội tham dự trận đấu diễn ra trong phòng mã số 17 ngày 06/12/90.

f. Danh sách các trận đấu bắt đầu hoặc kết thúc trong khỏng thời gian từ 13g và 16g diễn ra trên 1 sân của phòng mã số 49 ngày 05/08/90

g. Địa chỉ và tên các vận động viên đã chơi hoặc sẽ chơi đối lại đội mã số 1 của các CLB mã số 50.

BÀI 4: RÀNG BUỘC TOÀN VẸN

Ràng buộc toàn vẹn (Integrity Constraint) và kiểm tra ràng buộc toàn vẹn (RBTV) là hai trong số những vấn đề quan trọng của qúa trình phân tích, thiết kế và khai thác CSDL. Trong quá trình phân tích, thiết kế CSDL, nếu không quan tâm đúng mức đến các vấn đề của RBTV, có thể sẽ dẫn đến những hậu quả nghiêm trọng về tính an toàn và toàn vẹn dữ liệu. Đặc biệt đối với các CSDL lớn.

Vì vậy, khi xây dựng một chương trình ứng dụng, ngoài việc xây dựng các quan hệ để lưu trữ các đối tượng dữ liệu cần quản lý, người phân tích, thiết kế còn phải nắm rõ các quy tắc quản lý áp đặt trên các đối tượng dữ liệu đó, các quy tắc đó gọi là RBTV. Người phân tích, thiết kế phải phát hiện đầy đủ và mô tả một cách chính xác, rõ ràng trong hồ sơ thiết kế. Đồng thời chỉ rõ các thao tác cặp nhận dữ liệu nào cần được kiểm tra xử lý vi phạm RBTV, nhằm đảm bảo cho CSDL luôn biểu diễn chính xác và đúng ngữ nghĩa của thực tế.

4.1 CÁC YẾU TỐ CỦA RBTV

4.1.1 Khái niệm

Ràng buộc toàn vẹn là những điều kiện ràng buộc giá trị trên một thuộc tính hoặc giữa các thuộc tính, giữa các bộ trong một hay nhiều quan hệ. Các ràng buộc này là bất biến, thỏa mãn ở bất kỳ thời điểm nào khi khai thác CSDL.

Ví dụ: Cho CSDL bán hàng gồm các quan hệ

HH(MSHH, TenHH, DVT, SoTon, ĐG)

KH(MSKH, TenKH, DC, DT, Fax)

DDH(SoDDH, NgayLap, MSKH)

CTDDH(SoDDH, MSHH, SL, Dg, GiamGia)

Dựa vào tân từ ta có một số ràng buộc toàn vẹn như

RB1: Mỗi mặt hàng phải có một mã số để phân biệt với các mặt hàng khác.

RB2: Mỗi CTDDH (chi tiết đơn đặt hàng) phải có mã hàng phân biệt trong cùng một đơn hàng và mã hàng phải thuộc danh sách các mặt hàng có trong quan hệ MH.

RB3: Giá trị của các thuộc tính SoTon, Dg,SL, GiamGia phải ≥ 0.

4.1.2 Các yếu tố Của RBTV:

Một RBTV có 3 yếu tố: Bối cảnh, điều kiện và tầm ảnh hưởng của RBTV.

a. Bối cảnh của RBTV:

Là các quan hệ mà RBTV được định nghiã

b. Điều kiện:

Mô tả điều kiện của ràng buộc mà các quan hệ trong bối cảnh của ràng buộc đó phải thỏa.

Điều kiện có thể được biểu diễn bằng ngôn ngữ tự nhiên, mã giả (tựa Pascal), ngôn ngữ đại số tập hợp, ngôn ngữ vị từ ...

c. Tầm ảnh hưởng:

Chỉ định những thao tác cơ sở (thêm, xóa, sửa) trên những quan hệ nào thuộc bối cảnh RBTV có thể gây nên nguy cơ vi phạm điều kiện ràng buộc, cần phải được kiểm tra khi thực hiện thao tác đó (dấu '+' là phải kiểm tra, dấu '–' là không phải kiểm tra).

Ví dụ: RB1: Mỗi mặt hàng phải có 1 mã số phân biệt với các mặt hàng khác

HH(MSHH, TenHH, DVT, SoTon, ĐG)

Bối cảnh: HH

Điều kiện: $\forall q_i, q_j \in$ HH:

q_i.MSHH <> q_j.MSHH => $i \neq j$

Tầm ảnh hưởng:

	Thêm	Xóa	Sửa
HH	+	-	+ [MSHH]

4.2 PHÂN LOẠI CÁC RBTV

Dựa trên bối cảnh, RBTV có thể được chia làm 2 loại:

- RBTV có bối cảnh là 1 quan hệ
- RBTV có bối cảnh gồm nhiều quan hệ

4.2.1 RBTV có bối cảnh là một quan hệ

4.2.1.1 RBTV trên miền giá trị của thuộc tính

Là ràng buộc qui định giá trị hợp lệ của một thuộc tính.

Ví dụ: KetQuaHT(MSSV, MSMon, HocKy, DiemL1,DiemL2)

Tân từ: Điểm lần 1 lấy giá trị từ 0 đến 10 và có độ chính xác là 0.5

Bối cảnh: KetQuaHT

Điều kiện: $\forall q \in T_{ketQuaHT}$

(q.DiemL1= NULL) Or

(q.DiemL1 >= 0 And q.DiemL1 <=10 And (q.DiemL1 * 4) mod 2 = 0)

Tầm ảnh hưởng:

	Thêm	Xóa	Sửa
KetQuaHT	+	-	+ [DiemL1]

4.2.1.2 RBTV liên thuộc tính:

Là mối liên hệ giữa các thuộc tính trên 1 bộ trong cùng 1 lược đồ quan hệ

Ví dụ: Lang_Gieng(T_TP, T_TP_LG)

Ràng buộc(RB): Mã số thành phố và mã số thành phố láng giềng phải khác nhau.

Bối cảnh: Lang_Gieng

Điều kiện: $q \in$ Lang_Gieng: q.T_TP <> q. T_TP_LG

Tầm ảnh hưởng:

	Thêm	Xóa	Sửa
Lang_Gieng	+	-	+ [T_TP, T_TP_LG]

4.2.1.3 RBTV liên bộ:

Là sự ràng buộc giữa các bộ trong một quan hệ. Đặc biệt là các ràng buộc duy nhất về giá trị của các thuộc tính khóa.

Ví dụ: Mỗi sinh viên học được nhiều môn, điểm thi mỗi môn phải có MSMH(MÃ SỐ MÔN HỌC) để phân biệt điểm thi các môn khác của 1 sinh viên.

KetQuaHT(MSSV, MSMon, HocKy, DiemL1,DiemL2)

Bối cảnh: KetQuaHT

Điều kiện: $\forall q_i, q_j \in$ KetQuaHT ($i \neq j$):

q_i.MSSV <> q_j.MSSV Or q_i.MSMon <> q_j.MSMon

Tầm ảnh hưởng:

	Thêm	Xóa	Sửa
KetQuaHT	+	-	+ [MSSV, MSMon]

4.2.2 RBTV Có bối cảnh trên nhiều quan hệ

4.2.2.1 RBTV về phụ thuộc tồn tại (khóa ngoại)

Là điều kiện tồn tại 1 bộ trong 1 quan hệ (R) phụ thuộc vào sự tồn tại của 1 bộ nằm trong 1 quan hệ khác (Q). Điều kiện tồn tại này dựa trên khóa của quan hệ Q nằm trong quan hệ R. Còn được gọi là ràng buộc về khóa ngoại (Foreign key).

Ví dụ: Xét 2 quan hệ CTDH và DDH, Mỗi chi tiết đặt hàng phải thuộc một DDH.

Bối cảnh: DDH, CTDH

Điều kiện: CTDH[SoDDH] DDH[SoDDH]

Tầm ảnh hưởng:

	Thêm	Xóa	Sửa
CTDDH	+	-	+ [SoDDH]
DDH	-	+	+ [SoDDH]

4.2.2.2 RBTV liên bộ - liên quan hệ:

Là ràng buộc đối với từng nhóm các bộ của nhiều quan hệ khác nhau.

Ví dụ: Mỗi DDH chỉ bao gồm tối đa 5 mặt hàng khác nhau

Bối cảnh: DDH, CTDH

Điều kiện: $\forall$ dh $\in$ DDH:

$$Card(\{ct \in CTDH: ct.SoDDH = dh.SoDDH\}) \leq 5$$

Tầm ảnh hưởng:

	Thêm	Xóa	Sửa
CTDH	+	-	+ [SoDDH]
DDH	-	+	+ [SoDDH]

Ví dụ: Giả sử trên mỗi DDH có lưu trữ Tổng giá trị của DDH. Ta có RB như sau: Tổng giá trị của 1 DDH phải bằng tổng giá trị của các mặt hàng trên DDH đó.

Bối cảnh: DDH, CTDH

Điều kiện: $\forall$ dh $\in$ DDH: ct $\in$ CTDDH: ct.SoDDH = dh.SoDDH

dh.TongGT = (ct.SL*ct.DG(1-ct.Giamgia)*100)

Tầm ảnh hưởng:

	Thêm	Xóa	Sửa
CTDDH	+	+	+ [SL,DG,GiamGia]
DDH	+	+	+ [TongGT]

4.2.2.3 RBTV liên thuộc tính, liên quan hệ

Là sự ràng buộc giữa các thuộc tính khác nhau trên các quan hệ khác nhau.

Ví dụ: Trong CSDL bán hàng đã cho ta có ràng buộc đối với cùng mặt hàng, số lượng bán trong CTDDH phải nhỏ hơn số lượng tồn trong quan hệ HH.

Bối cảnh: CTDDH, HH

Điều kiện: $\forall q_i \in$ CTDDH , $q_j \in$ HH:

q_i.MSMH= q_j.MSMH and q_j.SoTon>=q_i.SL

Tầm ảnh hưởng:

	Thêm	Xóa	Sửa
CTDDH	+	-	+ [SL]
HH	-	-	+ [SoTon]

4.2.2.4 RBTV do đồ thị của LĐCSDL có chu trình:

Để thấy được mối liên hệ giữa các quan hệ trong một LĐCSDL, người ta thường biểu diễn LĐCSDL bằng một đồ thị vô hướng, trong đó:

- Mỗi LĐQH được ký hiệu bởi 1 nút có dạng: 0
- Mỗi thuộc tính của LĐQH được ký hiệu bởi 1 nút dạng: *
- Giữa nút LĐQH và nút thuộc tính của nó được nối nhau bởi 1 cung vô hướng.

Ví dụ: Từ LĐCSDL quản lý bán hàng

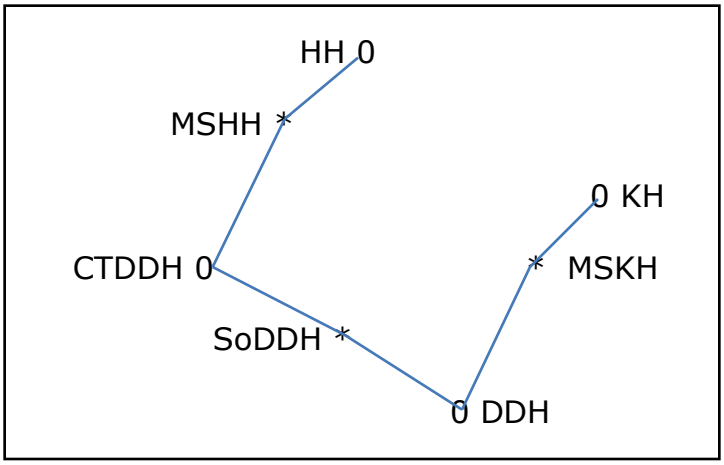
HH(MSHH, TenHH, DVT, SoTon, ĐG)

KH(MSKH, TenKH, DC, DT, Fax)

DDH(SoDDH, NgayLap, MSKH)

CTDDH(SoDDH, MSHH, SL, Dg, GiamGia)

Ta có đồ thị của lược đồ như sau:



Chu trình của đồ thị LĐCSDL là một đường cong khép kín có ít nhất 3 đỉnh quan hệ

Ví dụ: CSDL quản lý phân công thực hiện đề án như sau:

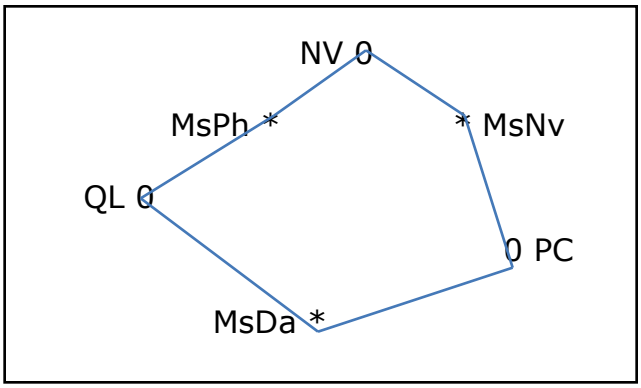
NhânViên (MsNv, MsPh)

QuanLy (MsPh, MsDa)

PhânCông (MsNv, MsDa)

Khi vẽ đồ thị ta thấy có chu trình

Như hình bên.



Ràng buộc có chu trình là thể hiện sự phụ thuộc tồn tại của một bộ trong quan hệ này với các bộ trong quan hệ khác có cùng chu trình. Trong ví dụ trên nếu chúng ta quản lý theo 3 nguyên tắc khác nhau thì có thể có ràng buộc có chu trình.

Nguyên tắc quản lý 1: Mỗi nhân viên của phòng ban được phân công vào tất cả các đề án mà phòng đó quản lý.

Nguyên tắc quản lý 2: Mỗi nhân viên của phòng ban được phân công làm các đề án trong số các đề án mà phòng đó quản lý.

Nguyên tắc quản lý 3: Mỗi nhân viên được phân vào đề án bất kỳ tùy theo năng lực chuyên môn.

Trường hợp theo nguyên tắc quản lý thứ 2 sẽ phát sinh ràng buộc có chu trình, vì lúc này đề án phân cho nhân viên của phòng ban nào phải nằm trong số đề án mà phòng ban đó quản lý.

Bối cảnh: PhânCông, NhânViên, QuảnLý

Điều kiện: (PC $\bowtie$ NV)[MaNv,MsDa] $\subseteq$ (NV $\bowtie$ QL)[MaNv, maDa]

Tầm ảnh hưởng:

	Thêm	Xóa	Sửa
PhânCông	+	-	+ [MaNv,MaDa]
NhânViên	-	+	+ [MaNv,MaPh]
QuảnLý	-	+	+ [MaDa,MaPh]

Bảng tầm ảnh hưởng tổng hợp: Là bảng tầm ảnh hưởng của tất cả các RBTV có trong một CSDL. Người ta làm bảng tầm ảnh hưởng tổng hợp để người lập trình tiện theo dõi khi cài đặt ứng dụng với CSDL đã được thiết kế.

RB	Q_1			Q_2			...	Q_n		
	Thêm	Xóa	Sửa	Thêm	Xóa	Sửa		Thêm	Xóa	Sửa
RB_1	+	-	-	-	+	+				
RB_2	-	+	+					+	+	-
...										
RB_m				+	-	+				

Bàng tầm ảnh hưởng tổng hợp cùa một CSDL

BÀI TẬP

Bài 1: Cho CSDL sau

SinhViên(MãSV, TênSV, NS, Phái, ĐịaChỉ)

Diễn giải: Mỗi sinh viên có một mã số để phân biệt (MãSV), một tên sinh viên (TênSV), một ngày sinh (NS), thuộc một phái (Phái) và có địa chỉ để liên lạc

MônHọc(MãMH, TênMH, SốTiếtLT, SốTiếtTH)

Diễn giải: mỗi môn học có một mã số để phân biệt (MãMH), có tên môn học (TênMH) và có số tiết lý thuyết (SốTiếtLT) luôn lớn hơn hay bằng số tiết thực hành (SốTiếtTH)

KếtQuả(MãSV, MãMH, Điểm)

Diễn giải: mỗi kết quả của sinh viên cho từng môn học (Điểm) được xác định bởi MãSV và MãMH.

Ghi chú: Khóa của các quan hệ được gạch dưới

Hãy phát biểu các ràng buộc toàn vẹn có trong cơ sở dữ liệu một cách chặt chẽ

Bài 2: Cho CSDL như sau

LOAI_XE(MãLX, TênLX)

Diễn giải: Mỗi loại xe (xe gắn máy 2 bánh, xe du lịch 4-7 chỗ, xe tải...) có một mã để phân biệt(MãLX) và một tên gọi(TênLX)

CHỦ_XE(MãChủXe, TênChủXe, ĐịaChỉ)

Diễn giải: Mỗi chủ xe có một mã số để phân biệt, một tên, một địa chỉ.

GPLX(Số_GPLX, NgàyCấp_GPLX, MãChủXe)

Diễn giải: Mỗi giấy phép lái xe có một số để phân biệt, một ngày cấp và một người chủ xe được cấp giấy phép lái xe

CT_GPLX(Số_GPLX, STT, NgàyCấp_CT, MãLX)

Diễn giải: Mỗi một chi tiết của một giấy phép lái xe có một số thứ tự để phân biệt với những chi tiết khác của cùng một giấy phép, có ngày cấp cho chi tiết này và một mã loại xe.

Ghi chú:

- Khóa của các quan hệ được gạch dưới.
- Ngày cấp của một chi tiết giấy phép lái xe có thể khác ngày cấp giấy phép lái xe, đó là trường hợp chủ xe thi đậu một loại xe khác và được bổ sung giấy phép của loại xe mới vào giấy phép đang có. Đối với mỗi giấy phép, các loại xe được cấp phép không được trùng nhau.

Câu hỏi: Hãy phát biểu các ràng buộc toàn vẹn có trong cơ sở dữ liệu trên một cách chặt chẽ.

Bài 3: Cho CSDL như sau

TOUR (#T,Tên_T,Gía_T_Người)

Diễn giải: Mỗi Tour du lịch có một mã số duy nhất (#T) để phân biệt, có một tên duy nhất (Tên_T), một giá duy nhất cho một người đi tour (Gía_T_Người)

KHÁCH_CÁ_NHÂN (#KCN, Tên_KCN, #T, Ngày_Đi_Tour_KCN, #KĐ, Bảng_Số_Xe, Ngày_BĐ_ĐI)

Diễn giải: Mỗi khách cá nhân có một mã số duy nhất (#KCN) để phân biệt với những khác khác, một tên duy nhất (Tên_KCN), đăng ký đi một tour duy nhất – đó là tour đăng ký sau cùng – (#T) vào một ngày (Ngày_Đi_Tour_KCN) và được phân vào một xe duy nhất để đi tour (Bảng_Số_Xe, Ngày_BĐ_ĐI). Nếu khách cá nhân là thành viên của một đoàn thì có một mã khách đoàn (#KĐ), nếu không đó là khách lẻ.

KHÁCH_ĐOÀN (#KĐ , Tên_Đ, TS_TV, #T, Ngày_Đi_Tour_KĐ)

Diễn giải: Mỗi đòan khách có một mã số duy nhất (#KĐ) để phân biệt với đòan khách khác, một tên duy nhất (Tên_Đ), có một tổng số thành viên duy nhất (TS_TV), đăng ký đi một tour duy nhất – đó là tour đăng ký sau cùng (#T) vào một ngày (Ngày_Đi_ Tour_KĐ).

PC_XE (Bảng_Số_Xe, Ngày_BĐ_Đi, #T, #KĐ)

Diễn giải: Mỗi phân công xe liên quan đến một bảng số xe và một ngày bắt đầu đi (Bảng_Số_Xe, Ngày_BĐ_Đi), liên quan đến một tour duy nhất (#T) và mã khách đoàn duy nhất (#KĐ).

Ghi chú:

Khóa của các quan hệ được gạch dưới.

Các thuộc tính: #T, Ngay_Đi_Tour_KCN, #KĐ, trong quan hệ KHÁCH_CÁ_NHÂN có thể có giá trị 'trống' (NULL). Trong trường hợp #KĐ có trị trống, Khách cá nhân được xem là khách lẻ.

Thuộc tính #KĐ trong quan hệ PC_XE có thể có giá trị 'trống' (NULL).

Câu hỏi: Hãy phát biểu các ràng buộc toàn vẹn có trong CSDL trên một cách chặt chẽ bằng ngôn ngữ đại số quan hệ hoặc SQL.

BÀI 5: THIẾT KẾ CƠ SỞ DỮ LIỆU

5.1 PHỤ THUỘC HÀM

Phụ thuộc hàm (Functional Dependency) là công cụ dùng để biểu diễn, một cách hình thức, một số RBTV. Thông qua cách biểu diễn phụ thuộc hàm (PTH), ta có thể dễ dàng xác định khóa của quan hệ, cũng như nhóm thuộc tính dùng để nhận dạng, tìm kiếm duy nhất một đối tượng hoặc một sự kiện của thế giới thực được mô hình hóa trong CSDL.

Phương pháp biểu diễn này có vai trò quan trọng trong các phương pháp thiết kế một lược đồ quan niệm của CSDL, nhằm tạo ra những quan hệ độc lập nhau, giảm thiểu sự trùng lắp, dư thừa dữ liệu lưu trữ. Do đó giảm bớt các sai xót trong thao tác cập nhật dữ liệu của người sử dụng. Ngoài ra, còn dùng để đánh giá chất lượng thiết kế một CSDL.

5.1.1 Định nghiã:

Cho một lược đồ quan hệ Q(X,Y,Z) với X,Y,Z là các tập thuộc tính con của Q^+ và với X,Y khác rỗng. Một Phụ thuộc hàm $X \rightarrow Y$ được xác định trên Q nếu và chỉ nếu

$$\forall\ q_i, q_j \in T_Q: q_i.X = q_j.X \Rightarrow q_i.Y = q_j.Y$$

Khi đó ta nói: X xác định hàm Y hay Y phụ thuộc hàm vào X

Quy ước: Nếu Y không phụ thuộc hàm vào X ta ký hiệu: X -/-> Y

Ví dụ: trong quan hệ **TD**(MsTD, MsPh, GioTD, NgayTD)

Có quy ước: mỗi trận đấu diễn ra trong một phòng, vào 1 ngày và 1 giờ bắt đầu đã qui định

Ta có phụ thuộc hàm: MsTD $\rightarrow$ MsPh, GioTD, NgayTD

Ví dụ: **DOI_TD**(MsTD, MsDoi, MsCLB)

Mỗi trận đấu là 1 cuộc gặp gỡ giữa 2 đội: MsTD -/-> MsDoi, MsCLB

5.1.2 Các Phụ thuộc hàm đặc biệt

5.1.2.1 Phụ thuộc hàm hiển nhiên

$X \rightarrow X$ hay $AB \rightarrow B$

5.1.2.2 Phụ thuộc hàm nguyên tố

$X \rightarrow Y$ là PTH nguyên tố (hay phụ thuộc hàm đầy đủ) khi và chỉ khi: $\exists X' \subseteq X$ và $X' <> X$ mà $X' \rightarrow Y$

Ví dụ: **DOI**(MsCLB, MsDoi, MsLT, Phai)

MsCLB, MsDoi $\rightarrow$ MsLT là phụ thuộc hàm nguyên tố do:

MsCLB -/-> MsLT và MsDoi -/-> MsLT

5.1.2.3 Tập phụ thuộc hàm của một quan hệ

Tập hợp các phụ thuộc hàm của Q được ký hiệu là F_Q

F_Q = { fi: $X \rightarrow Y$ xác định trên Q}

Qui ước: Chỉ mô tả các phụ thuộc hàm không hiển nhiên trong tập F.

5.1.2.4 Biểu diễn tập PTH bằng đồ thị có hướng:

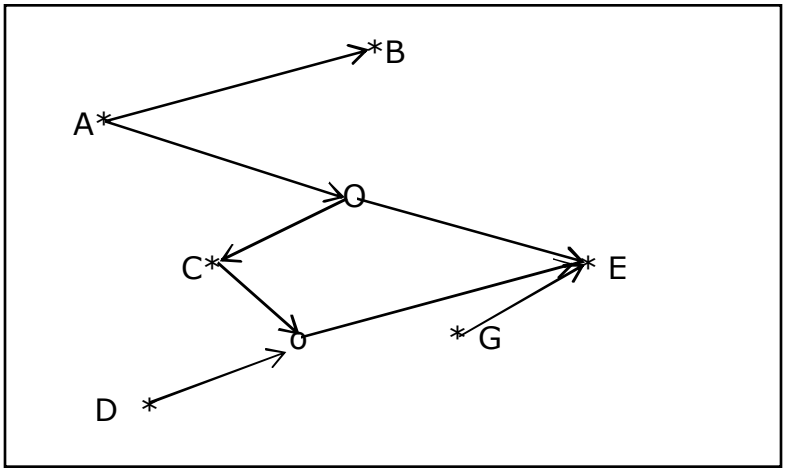
Đồ thị tập phụ thuộc hàm gồm:

- Hai loại nút: Nút thuộc tính và nút PTH
- Hai loại cung: Cung từ nút thuộc tính đến nút PTH và cung từ nút PTH đến nút thuộc tính.

Ví dụ: Cho 1 lược đồ quan hệ u=(Q,F) với Q=(ABCDEG) và tập PTH F:

F = { f_1: $A \rightarrow B$, f_2: $CD \rightarrow E$, f_3: $E \rightarrow G$, f_4: $A \rightarrow CE$}

Ta có đồ thị của tập PTH F như sau:



5.1.3 Hệ tiên đề Amstrong và ứng dụng

5.1.3.1 Hệ tiên đề Amstrong

Cho lược đồ quan hệ Q và X, Y, W, Z $\subseteq Q^+$.

LD1: Tính phản xạ: $Y \subseteq X \Rightarrow X \rightarrow Y$

LD2: Luật thêm vào: Nếu $X \rightarrow Y$ và $Z \subseteq W$ thì $X,W \rightarrow Y,Z$

LD3: Luật bắc cầu: Nếu $X \rightarrow Y$ và $Y \rightarrow Z$ thì $X \rightarrow Z$

Một số luật dẫn suy từ hệ tiên đề Amstrong:

LD4: Tính phân rã: Nếu $X \rightarrow Y,Z$ thì $X \rightarrow Y$ và $X \rightarrow Z$

LD5: Tính hội: Nếu $X \rightarrow Y$ và $X \rightarrow Z$ thì $X \rightarrow Y,Z$

LD6: Tính bắc cầu giả: Nếu $X \rightarrow Y$ và $Y,Z \rightarrow W$ thì $X,Z \rightarrow W$

Chứng minh:

LD4: $Y \subseteq Y\ Z \Rightarrow Y\ Z \rightarrow Y$ {tính phản xạ}

$Z \subseteq Y\ Z \Rightarrow Y\ Z \rightarrow Z$

Ap dụng tính bắc cầu (LD3) ta có kết quả: $X \rightarrow Y$ và $X \rightarrow Z$

LD5: $X \rightarrow Y$ và $X \subseteq X \Rightarrow X \rightarrow X,Y$ (1) {luật thêm vào}

$X \rightarrow Z$ và $Y \subseteq Y \Rightarrow X,Y \rightarrow Z$ (2)

từ (1) và (2): $X \rightarrow Y,Z$ {tính bắc cầu}

LD6: $X \rightarrow Y$ và $Z \subseteq Z$ ==> $X,Z \rightarrow Y,Z \rightarrow W$

5.1.3.2 Ứng dụng

Với tập PTH F cho trước xác định trên một quan hệ Q ta có thể suy diễn thêm nhiều phụ thuộc hàm khác dựa trên các luật dẫn của hệ tiên đề Amstrong hay nói cách khác là kiểm tra một PTH f có thể suy ra từ F hay không ?

Ví dụ: Cho Q(ABCDEH) và

F = {f_1: AB→C, f_2:AE→D, f_3:BC→D, f_4:C→E, f_5:ED→H}

Kiểm tra xem từ F có thể suy ra AB→EH hay không?

Giải:

Từ f_1 và f_4 áp dụng luật bắc cầu: => $AB \rightarrow E$ (1)

Từ f_2 và f_5 áp dụng luật bắc cầu giả: => $AE \rightarrow H$ (2)

Từ (1) và (2) áp dụng luật bắc cầu giả => $AB \rightarrow H$ (3)

Từ (1) và (3) áp dụng luật hội: => $AB \rightarrow EH$

Vậy từ F dựa vào các luật trong hệ luật dẫn Amstrong chúng ta suy ra được $AB \rightarrow EH$. Kết luận là từ F có thể suy ra $AB \rightarrow EH$.

5.1.4 Bao đóng của tập phụ thuộc hàm

5.1.4.1 Định nghĩa

Cho một quan hệ Q có tập các phụ thuộc hàm F = { X-->Y xác định trên Q}. Bao đóng của F, ký hiệu F^+, là tập hợp tất cả các PTH có thể suy diễn từ F dựa vào hệ luật dẫn Amstrong.

Ký hiệu: $F_Q^+ = \{ X \rightarrow Y / F \models X \rightarrow Y\}$

Ví dụ: Q(A,B,C) và F = {f1: $A \rightarrow B$, f2: $B \rightarrow C$}

F_Q^+ = { A→B; A→C; A→A,C; A→ABC;...}

Lưu ý: Trong thực tế, việc tìm F^+ là rất khó khăn. Vì vậy, từ một tập PTH F cho trước, người ta cố tìm cách giải quyết bài toán: Tìm một tập phụ thuộc hàm nhỏ nhất

mà từ đó có thể suy diễn ra các phụ thuộc hàm khác. Tập đó gọi là phủ tối tiểu của tập phụ thuộc hàm F ban đầu. Ngoài ra bài toán thành viên (xác định một phụ thuộc hàm có thuộc F^+ không?) cũng được đưa ra.

Ví dụ: Cho LĐQH u=(Q,F) với Q=(ABCDEG) và

$F=\{AE \rightarrow C, CG \rightarrow A, DB \rightarrow G, GA \rightarrow E\}$

Kiểm tra phụ thuộc hàm $BDC \rightarrow Q$ có thuộc F^+ không?

(dựa vào ví dụ của phần 5.1.3.2 sinh viên tự giải ví dụ này)

5.1.4.2 Ứng dụng:

Dựa trên bao đóng F^+ của F ta có thể xác định được tập tất cả các thuộc tính phụ thuộc vào một tập thuộc tính X cho trước và có thể kiểm tra một PTH nào đó có thuộc vào bao đóng F^+ hay không.

Tuy nhiên, Việc xây dựng bao đóng F^+ tốn rất nhiều thời gian. Để giải quyết các bài toán trên người ta dựa vào 1 khái niệm mới, *Bao đóng của một tập thuộc tính.*

5.1.5 Bao đóng của tập thuộc tính

5.1.5.1 Định nghĩa

Cho 1 lược đồ quan hệ u=(Q,F) với Q^+ là tập thuộc tính và có tập các phụ thuộc hàm F={f1, f2,..,fn} và $X \subseteq Q^+$. Bao đóng của tập thuộc tính X dựa trên F, ký hiệu X^+_F, là tập các thuộc tính phụ thuộc hàm vào X dựa trên F.

$X^+_F = \{ Y \in Q^+ / X \rightarrow Y \in F^+\}$

Nhận xét:

1. $X \in X^+_F$
2. $Y \in X^+_F <=> f: X \rightarrow Y \in F^+$.

Dựa vào nhận xét 2 ta có thể giải quyết bài toán thành viên bằng cách xác định bao đóng của tập thuộc tính.

Thuật toán xác định X^+_F

Dữ liệu vào: X, F

Dữ liệu ra: X^+_F

begin

$X^+_F = X$;

Repeat

$X' = X^+_F$

For i:=1 to m do { với m = card(F)}

if VT(fi) $\subseteq X^+_F$ then $X^+_F := X^+_F \cup$ VP(fi) { VT(fi): vế trái của fi, VP: vế phải }

Until ($X^+_F = X'$);

end;

Ví dụ: cho Q(ABCDEGH) và tập phụ thuộc hàm

F ={f1:B→A; f2:DA→CE; f3:D→H; f4:GH→C; f5:AC→D}

a. Tìm bao đóng của tập thuộc tính {BD}

$X^+_F = BD$

Do f1: $X^+_F = BDA$

Do f3: $X^+_F = BDAH$

Do f2: $X^+_F = BDAHCE$

Do f3: $X^+_F = BDAHCE$

Vậy $X^+_F = BDAHCE$.

b. Tìm bao đóng của tập thuộc tính {BCG}

$X^+_F = BCG$

Do f1: $X^+_F = BCGA$

Do f5: $X^+_F = BCGAD$

Do f2: $X^+_F = BCGADE$

Do f3: $X^+_F = BCGADEH$

Vậy $X^+_F = Q^+$.

Ví dụ: Cho Q(ABCDEH) và

F = {f1: AB→C, f2:AE→D, f3:BC→D, f4:C→E, f5:ED→H}

Kiểm tra AB→EH có thuộc vào F^+ hay không?

Cách giải: Kiểm tra EH $\subseteq \{AB\}_{F^+}$? Nếu EH nằm trong $\{AB\}_{F^+}$ thì kết luật AB → EH thuộc F^+. Ngược lai kết luận là không thuộc F^+.

5.2 KHÓA VÀ CÁCH XÁC ĐỊNH

5.2.1 Định nghĩa

Cho lược đồ quan hệ u=(Q,F) với Q là tập thuộc tính và F = { f_1,f_2,..f_n}

a. S $\subseteq Q^+$, S là siêu khóa của Q nếu S →$Q^+ \in F^+$

b. K $\subseteq Q^+$, K là khóa chính nếu K là siêu khóa và PTH K→Q^+ là PTH nguyên tố.

Nhận xét: Nếu đồ thị biểu diễn của tập pth F không chứa chu trình thì Q chỉ có duy nhất một khóa.

5.2.2 Cách xác định khóa của một quan hệ

5.2.2.1 Thuật toán xác định một khóa của lược đồ quan hệ

Ý tưởng: Xuất phát từ một siêu khóa K=Q. Duyệt lần lượt các thuộc tính $A_i \subseteq$ K. Nếu bất biến $(K-A_i)^+ = Q$ được bảo toàn thỉ loại A_i khỏi K. kết thúc quá trình lặp ta được K là một khóa chính của LĐQH.

Dữ liệu vào: Q,F

Dữ liệu ra: K là một khóa của (Q,F)

```
Begin
  K=Q;
  For each attribute Ai in Q+ do
     if ((K-Ai)+ = Q) then
        K=K-Ai ;
```

Write (K);

End.

Ví dụ: áp dụng thuật toán trên tìm một khóa của LĐQH u=(Q,F) với Q=ABCDE và

F= {B $\rightarrow$ C, C $\rightarrow$ BD, BE $\rightarrow$ A, A $\rightarrow$ C}

Bước 1: K=ABCDE

Vòng lặp thực hiện 5 lần do |Q|=5

Với i=1. Xét thuộc tính A

Tính $BCDE^+$ = Q => K=BCDE

Với i=2. Xét thuộc tính B

Tính CDE^+ = Q => K=CDE

Với i=3. Xét thuộc tính C

Tính $DE^+ \neq Q$ => K=CDE

Với i=4. Xét thuộc tính D

Tính CE^+ = Q => K=CE

Với i=5. Xét thuộc tính E

Tính $C^+ \neq Q$ => K=CE

Xuất ra khóa K=CE.

5.2.2.2 Thuật toán xác khóa dựa vào đồ thị của LĐQH

Ý tưởng:

Gọi N là tập thuộc tính nguồn, chỉ chứa các nút thuộc tính không có cung đi tới

Gọi M là tập thuộc tính trung gian, chứa các nút thuộc tính vừa có cung đi tới, vừa có cung đi ra.

Nếu từ N ta có thể lần theo các cung để đến được tất cả các nút còn lại thì N chính là khóa và là khóa duy nhất.

Ngược lại, ta lần lượt hội N với 1 tập con của M để kiểm tra có là khóa hay không.

Ví dụ: Cho lược đồ quan hệ Giảng dạy:

GD(MsGV, Hoten, MsMH, TenMH, Phòng, Giờ)

F={f1:MsGV $\rightarrow$ Hoten; f2: MsMH $\rightarrow$TenMH; f3: Phong,Gio $\rightarrow$ MSMH;

f4: MsGV,Giờ $\rightarrow$ Phòng}

Tìm khóa của quan hệ GD.

Vẽ đồ thị lược đồ quan hệ. Ta nhận thấy: N = {MsGV, Gio} và M = {MsMH, Phong}. Vậy Khóa K= {MsGV, Gio} là khóa duy nhất.

5.2.2.3 Thuật toán tìm tất cả các khóa của LĐQH

Dữ liệu vào: <Q,F>

Dữ liệu ra: K {Tập các khóa của quan hệ Q}

Begin

b1: Tìm tập N, với N là tập các thuộc tính không xuất hiện ở vế phải của các phụ thuộc hàm. Tính N^+. Nếu N^+ = Q thì K=N là khóa duy nhất (Thoát). Ngược lại sang bước 2

b2: Tìm tập M với M là tập các thuộc tính xuất hiện ở cả 2 vế trong tập PTH F.

b3: Xây dựng 2^m-1 tập con của tập M với m = Card(M)

b4: Xây dựng tập K chứa các khóa

Begin

K ={} ;

For i:=1 to (2^m-1) do

begin

K_i:= N $\cup$ M_i ;

Nếu K_i không chứa các khóa đã xác định trước đó và $X_i{}^+{}_F$ = Q^+ thì

K_i là 1 khóa của Q: K = K $\cup$ K_i.

end;

End;

Ví dụ: Cho lược đồ quan hệ u=(Q,F) với Q=(ABCDEG) và F = { f_1: AD $\rightarrow$ B;

f_2: EG $\rightarrow$ A; f_3: BC $\rightarrow$ G}. Xác định các khóa của lược đồ quan hệ u.

Giải:

N = {CDE}

M = {ABG}

Tập các khóa tìm được K={K_1=CDEA, K_2=CDEB, K_3=CDEG}.

Ví dụ: Cho lược đồ quan hệ u=(Q,F) với Q = (ABCDE) và F = {A $\rightarrow$ BE, B $\rightarrow$ E,

C $\rightarrow$ D, EC $\rightarrow$ AB }. Xác định các khóa của quan hệ Q.

Giải:

N = {C};

M = {ABE}

Các tập con của M gồm {A,B,E,AB,AE,BE,ABE}.

Lần lượt hội với tập nguồn {C} ta có 3 khóa: K_1= AC, K_2= BC, K_3= EC. Các tập khác dù bao đóng bằng Q nhưng vẫn bị loại do chứa 3 khóa đã tìm thấy và chúng được coi là siêu khóa.

BÀI TẬP

Bài 1: Cho lược đồ quan hệ r(Q,F) với Q=(A,B,C,D,E,G) ,

$$F=\{A\rightarrow B,\ CD\rightarrow A,\ BC\rightarrow D,\ AE\rightarrow BG\}$$

a. Tìm một khóa của r?

b. Các tập ABCE và ABD có phải là khóa của r không? tại sao?

c. Tìm tập các thuộc tính không tham gia vào khóa nào của r?

Bài 2: Cho lược đồ quan hệ u(Q,F) với Q=(I,J,K,L,M,N,O),

$$F=\{L\rightarrow J,\ LM\rightarrow O,\ N\rightarrow K,\ KO\rightarrow N,\ M\rightarrow I,\ LO\rightarrow MI\}$$

a. Lược đồ quan hệ u có duy nhất một khóa không? Tại sao?

b. Tìm tất cả Khóa của u?

c. Có thể thêm một phụ thuộc hàm vào F để u có khóa duy nhất khóa không? Tại sao?

Bài 3: Cho lược đồ quan hệ r(Q,F) với Q=(A,B,C,D,E,G,H),

$$F=\{B\rightarrow AC,\ HD\rightarrow AE,\ AC\rightarrow BE,\ E\rightarrow H,\ A\rightarrow D,\ G\rightarrow E\}$$

a. Tìm một khóa của r?

b. Các tập BCG và ACB có phải là khóa của r không? tại sao?

c. Tìm tập các thuộc tính không tham gia vào khóa nào của r?

BÀI 6: DẠNG CHUẨN

Trong thực tế, một ứng dụng có thể được phân tích và thiết kế thành nhiều lược đồ CSDL khác nhau. Để đánh giá chất lượng giữa các lược đồ này người ta dựa trên một số tiêu chuẩn trong đó có tiêu chuẩn về dạng chuẩn của lược đồ CSDL:

- Sự trùng lắp thông tin: Vì nó sẽ làm tăng không gian lưu trữ, cũng như chi phí kiểm tra RBTV một cách vô ích.
- Chi phí kiểm tra ràng buộc toàn vẹn.
- Bảo toàn thông tin.
- Bảo toàn qui tắc quản lý tức là bảo toàn các phụ thuộc hàm.

Dạng chuẩn được xây dựng nhằm mục đích giảm sự trùng lắp thông tin trong CSDL. Từ đó tránh các sai sót trong các thao tác cặp nhật trên CSDL.

Ví dụ: Xét một thể hiện của quan hệ quản lý học tập(QLHT) của sinh viên

QLHT(MsSV, Ten, NS, Phai, ĐC, MsLop, TenLop, MsMH, TenMH, Diem)

F = { f_1:MsSV $\rightarrow$ Ten, NS, Phai, ĐC, MsLop; f_2: MsLop $\rightarrow$ TenLop; f_3: MsMH $\rightarrow$ TenMH; f_4: MsSV, MsMH $\rightarrow$ Diem }

Quan hệ trên có sự trùng lắp thông tin. Sự trùng lắp thông tin này có thể gây nên một số vấn đề khi thực hiện các thao tác trên quan hệ:

- Sửa đổi: Giả sử có 1 sinh viên (SV) thay đổi địa chỉ, thì hệ thống cần phải duyệt trên toàn bộ quan hệ để tìm và sửa địa chỉ ở các bộ liên quan đến SV này. Nếu để sót thì sẽ dẫn đến tình trạng thông tin không nhất quán
- Xóa: Giả sử SV có mã số 1108 hiện nay chỉ đăng ký học môn CSDL. Nếu muốn xóa kết quả điểm môn này thì dẫn đến mất luôn thông tin của SV
- Thêm: Vì khóa của quan hệ là {MSSV, MSMH} do đó không thể thêm 1 SV vào quan hệ nếu SV đó chưa đăng ký học môn nào.

Qua ví dụ trên chúng ta nhận thấy sự trùng lắp thông tin là nguyên nhân làm cho CSDL có chất lượng kém. Để hạn chế tình trạng trùng lắp dữ liệu, người ta đưa ra các yêu cầu thiết kế cần thiết cho một quan hệ dựa trên khái niệm phụ thuộc hàm, được gọi là các dạng chuẩn của một quan hệ.

6.1 CÁC DẠNG CHUẨN

6.1.1 Dạng chuẩn 1

Khái niệm thuộc tính đơn: Một thuộc tính được gọi là thuộc tính đơn nếu giá trị của nó không phải là sự kết hợp bởi nhiều thông tin có ý nghĩa khác nhau và hệ thống luôn truy xuất trên toàn bộ giá trị của nó. Ngược lại, là thuộc tính kép.

Ví dụ: Xét quan hệ VatTu(MaVT, TenVT, DVT)

Nếu TenVT bao gồm tên của vật tư và cả qui cách của nó. Như vậy TenVT là thuộc tính kép.

Ví dụ: ChuyenMon(MaGV, MonGD)

Nếu MonGD là một chuỗi các môn học mà giáo viên có thể phụ trách như: CSDL, CTDL, KTLT, PTTKHĐT, thì MônGD không phải là thuộc tính đơn.

Định nghĩa (dạng chuẩn 1):

Một lược đồ quan hệ Q đạt dạng chuẩn 1 (1 Normal Form - 1NF) nếu mọi thuộc tính của Q đều là thuộc tính đơn.

Chú ý: Đối với thuộc tính lưu trữ ngày dương lịch có thể xem là thuộc tính đơn.

Nhận xét:

- Một quan hệ có 1NF được xem là quan hệ có **cấu trúc phẳng.**
- Quan hệ đạt dạng chuẩn 1 cũng có thể còn trùng lắp thông tin

Ví dụ: TKBCoiThi(GT, Ngay, Giờ, Phong, Mon, GV)

F = { f_1: GT $\rightarrow$ Ngay,Gio,Phong,Mon,GV;

f_2:Mon $\rightarrow$ GV; f_3:Ngay,gio,phong $\rightarrow$ Mon}

Với qui định Mon → GV (Mỗi môn học chỉ có một giảng viên phụ trách) sẽ dẫn đến sự trùng lắp thông tin.

Nếu thay đổi giờ thi của 2 môn thi sẽ sai giảng viên

6.1.2 Dạng chuẩn 2

6.1.2.1 Khái niệm phụ thuộc đầy đủ:

Cho phụ thuộc hàm X → A. Thuộc tính A được gọi là phụ thuộc đầy đủ vào tập thuộc tính X nếu:

Không tồn tại X' ⊆ X và X' → A

Ví dụ: phụ thuộc hàm: MsSV, MsMH → TenSV là phụ thuộc hàm không đầy đủ vì chỉ cần một thuộc tính MsSV là đã xác định được Ten: MsSV → TenSV

6.1.2.2 Định nghĩa (dạng chuẩn 2)

Một lược đồ quan hệ Q đạt dạng chuẩn 2 nếu:

a. Q ở dạng chuẩn 1,

b. Mọi thuộc tính không khóa đều phụ thuộc đầy đủ vào các khóa của Q.

Ví dụ: Xét quan hệ quản lý học tập(QLHT) sau:

QLHT(MsSV, Ten, NS, Phai, ĐC, MsLop, tenLop, MsMh, TenMh, Diem)

F= { MsSV → Ten, NS, Phai, ĐC, MsLop ;

MsLop → tenLop;

MsMh →TenMh;

MsMh, MsSV → Diem }

Quan hệ QLHT không đạt dạng chuẩn 2 do MsSV → Ten với ten là thuộc tính không khóa, không phụ thuộc đầy đủ vào khóa {MsMh, MsSV}.

Có thể thay quan hệ QLHT bằng 3 quan hệ sau để đạt dạng chuẩn 2:

QLHT(MsSV, MsMH, Diem)

F_{QLHT} ={ f_4: MsSV, MsMH → Diem}

SV(MsSV, Ten, NS, Phai, ĐC, MsLop, TenLop)

F_{SV} = {f_1:MsSV $\rightarrow$ Ten, NS, Phai, ĐC, MsLop; f_2: MsLop $\rightarrow$ TenLop}

MH(MsMH, TenMH)

F_{MH} = { f_3: MsMH $\rightarrow$ TenMH}

Ví dụ: LopHoc(Lop, Mon, NgayKG, HocPhi)

F = { f_1: Lop,Mon $\rightarrow$ NgayKG; f_2: Mon $\rightarrow$ HocPhi}

Xác định khóa và kiểm tra có đạt dạng chuẩn 2 hay không?

Giải: Dựa vào F ta có Khóa là {Lop, Mon}

Quan hệ LopHoc không ở dạng chuẩn 2 vì thuộc tính không khóa HocPhi không phụ thuộc đầy đủ vào khóa.

Tách 2 quan hệ LopHoc(Lop, Mon, NgayKG)

F_{LopHoc} = { f_1: Lop,Mon $\rightarrow$ NgayKG}

và MonHoc(Mon,HocPhi)

F_{MonHoc} = { f_2: Mon $\rightarrow$ HocPhi}

thì Q ở dạng chuẩn 2.

Ví dụ: xét quan hệ Quản lý đăng ký điện thoại khách hàng:

ĐTKH(MsĐT, TenKH, MSCáp, KinhPhí),

F = { MsĐT $\rightarrow$ TenKH; MsĐT, MsCáp $\rightarrow$ Kinhphí}

Quan hệ ĐTKH có đạt dạng chuẩn 2 hay không?

Nhận xét:

- Nếu mỗi khóa của quan hệ Q chỉ có 1 thuộc tính thì Q đạt dạng chuẩn 2.
- Quan hệ SV ở dạng chuẩn 2 nhưng vẫn trùng lắp thông tin.

6.1.3 Dạng chuẩn 3

6.1.3.1 Khái niệm phụ thuộc bắc cầu:

Thuộc tính $A \subseteq Q^+$ được gọi là phụ thuộc bắc cầu vào một tập thuộc tính X nếu tồn tại nhóm thuộc tính $Y \subseteq Q^+$ thỏa mản 4 điều kiện sau:

i. $X \rightarrow Y \in F^+$

ii. $Y \rightarrow A \in F^+$

iii.Y -/-> X

iv.$A \notin \{X \cup Y\}$

Ví dụ: Xét quan hệ SV(MsSV, Ten, Ngsinh, Phai, MsLop, TenLop)

TenLop phụ thuộc bắc cầu vào MsSV vì: MsSV $\rightarrow$ MsLop và MsLop $\rightarrow$ TenLop.

6.1.3.2 Định nghĩa (dạng chuẩn 3)

Một lược đồ quan hệ Q đạt dạng chuẩn 3 nếu:

a. Q ở dạng chuẩn 1

b. Mọi thuộc tính không khóa của Q đều không phụ thuộc bắc cầu vào một khóa nào của Q.

Ví dụ: Quan hệ SV không đạt dạng chuẩn 3. Ta có thể tách thành 2 quan hệ:

SV(MsSV, Ten, Ngsinh, Phai, MsLop)

Lop(MsLop, TenLop)

Ví dụ: Xét LĐCSDL gồm 3 LĐQH sau:

PXN (SP, Ngay, MsKH, TenKH, TrịGiá)

F ={SP $\rightarrow$ Ngay, MsKH, TrịGia; MsKH $\rightarrow$ TenKH}

CTP (SP, MsHH, TenHH, SL,ĐG)

F = {SP, MsHH $\rightarrow$ SL,ĐG; MsHH $\rightarrow$ TenHH}

TK (MSHH, MSKho, TenKho, SLT)

F={ MSHH, MSKho → SLT; MSKho → TenKho; TenKho → MsKho}

- Xét quan hệ phiếu xuất nhập PXN: có 1 khóa {SP}

Đạt dạng chuẩn 2 nhưng không đạt dạng chuẩn 3 vì phụ thuộc bắc cầu

SP → MsKH; MSKH -/-> SP; MSKH → TenKH

- Quan hệ chi tiết phiếu CTP: có 1 khóa là {SP,MsHH}

Không đạt dạng chuẩn 2 vì TenHH không phụ thuộc đầy đủ vào khóa

- Quan hệ tồn kho TK: có 2 khóa là {MSHH,MSKho} và {MsHH, TenKho}

Đạt dạng chuẩn 3 vì chỉ có 1 thuộc tính không khóa là SLT và thuộc tính này không phụ thuộc bắc cầu vào các khóa.

Nhận xét:

Quan hệ tồn kho đạt dạng chuẩn 3 nhưng vẫn còn sự trùng lắp thông tin trên các cột MsKho và TenKho.

6.1.4 Dạng chuẩn BC(Boyce - Codd)

Định nghĩa: Một lược đồ quan hệ Q ở dạng chuẩn BC nếu mọi phụ thuộc hàm không hiển nhiên đều có vế trái chứa khóa (hay còn gọi vế trái là một siêu khóa).

$X \to A \in F^+$: $A \notin X$ và X phải chứa khóa của Q (X là một siêu khóa)

Nhận xét: Nếu Q đạt dạng chuẩn BC thì mọi vế trái của tập PTH đều là siêu khóa.

Ví dụ: Quan hệ TK không đạt dạng chuẩn BC vì: MsKho → TenKho

Ta tách thành 2 quan hệ: TK (MSHH, MSKho, SLT) và Kho(MSKho, TenKho) thì lúc này cả 2 quan hệ này đều đạt BCNF.

6.1.5 Dạng chuẩn của một LĐCSDL

Là dạng chuẩn thấp nhất trong các lược đồ quan hệ của LĐCSDL.

Ví dụ: Cho lược đồ CSDL gồm hai lược đồ quan hệ sau:

$Q_1(ABC)$, $F_1=\{A \to BC, C \to B\}$

$Q_2(BC)$, $F_2=\{C \to B\}$

Cho biết LĐCSDL trên ở dạng chuẩn cao nhất nào ?

$<Q_1,F_1>$ ở 2NF do có phụ thuộc bắc cầu vào khóa. $<Q_2,F_2>$ ở BCNF.

Vậy LĐCSDL ở 2NF.

6.2 CHUẨN HOA MỘT LDQH BẰNG PHAN RÃ:

6.2.1 Khái Niệm Chuẩn Hóa

Xuất phát từ tập thuộc tính của lược đồ quan hệ Q đã phân tích được, cùng với tập phụ thuộc hàm(PTH) F_Q xác định trên Q thể hiện các qui tắc quản lý. Q chưa ở dạng chuẩn 3 và gọi là quan hệ phổ quát. Ta phân rã thành các quan hệ con $\{Q_i\}$ dựa trên tập PTH F_Q sao cho các quan hệ con đạt dạng chuẩn cao nhất, tối thiểu là DC3, được gọi là quá trình chuẩn hoá.

Trong quá trình chuẩn hóa cần chú trọng đến 2 tính chất sau:

- Bảo toàn thông tin: Phân rã nhận được phải bảo đảm tính kết không mất thông tin, nghiã là:

  $Q_1^+ \cup Q_2^+ = Q^+$.

  $\forall T_Q$: $T_Q = T_Q[Q_1^+] \bowtie T_Q[Q_2^+]$

- Bảo toàn PTH:

  $\cup Q_i^+ = Q^+$

  $\cup F_{Qi} = F_Q$

6.2.2 Định lý Delobel(1973)

Cho lược đồ quan hệ u=(Q,F). Nếu $f:X \to A \in F^+$ sao cho $X \cup A \subset Q^+$ (là con thực sự của Q^+) thì phép phân rã Q thành 2 lược đồ quan hệ con:

$<Q1(X, A), F1=\{f\in F: VT(f) \cup VP(f) \subset Q_1^+\} >$

$<Q2(Q^+\setminus A), F2=\{f\in F: VT(f) \cup VP(f) \subset Q_2^+\} >$

là bảo toàn thông tin

6.2.3 Thuật toán phân rã

Ý tưởng: Dựa vào định lý Delobel, ta phân rã quan hệ Q thành 2 quan hệ Q_1 và Q_2 bằng 1 PTH f thỏa điều kiện của định lý. Lặp lại phân rã trên Q_1 và Q_2 cho đến khi các Q_i thu được đều đạt BCNF.

Procedure PhanRa(Q, F)

Dữ liệu vào: <Q, F>

Dữ liệu ra: = { Qi }$^{n}_{i=1}$ {tập các lược đồ quan hệ được phân rã}

Begin

b1. Loại bỏ các PTH có VT $\cup$ VP = Q^+ khỏi F

F* = F \ { f $\in$ F: VT(f) $\cup$ VP(f) = Q^+ }

b2. Nếu F* = $\varnothing$ thì C = C $\cup$ {Q} và kết thúc {Điểm dừng}

ngược lại, thực hiện phân rã

b2.1. Chọn f:X $\rightarrow$ Y $\in$ F

b2.2. Phân rã thành 2 lược đồ con:

<Q_1^+ = {X, A}, F1={f $\in$ F: VT(f) $\cup$ VP(f) Q_1^+} >

<Q_2^+ = Q^+ \ A, F2={f $\in$ F: VT(f) $\cup$ VP(f) Q_2^+} >

b2.3. Phân rã đệ qui Q_1 và Q_2:

PhanRa(Q_1, F_1);

PhanRa(Q_2, F_2);

end;

Ví dụ: Xét LƯỢC ĐỒ QUAN HỆ Q(MsKH, TP, CTyVC, MsHH, SL)

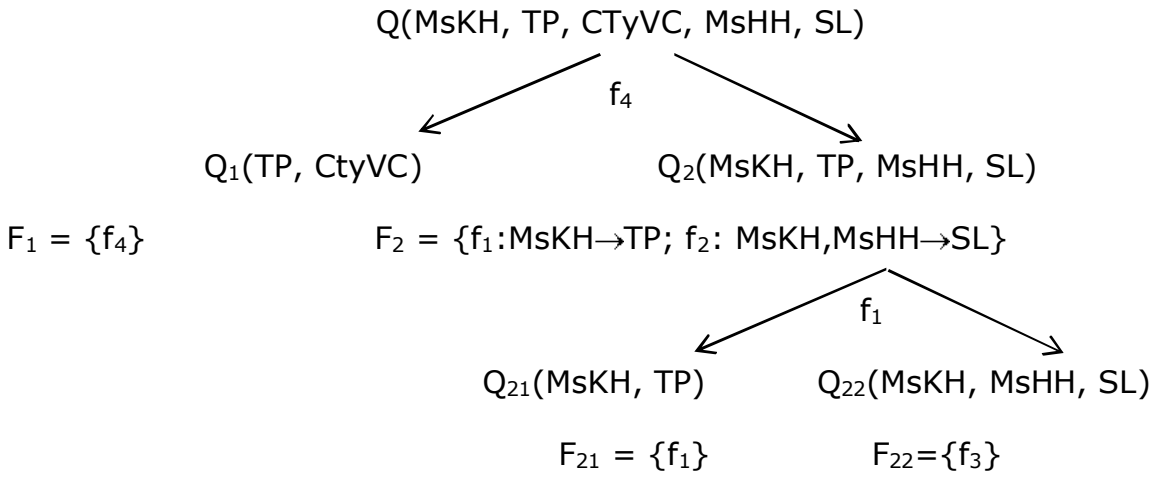
MsKH: Mã số Khách hàng. TP: Thành phố của nhà cung cấp. CtyVC: công ty vận chuyển hàng. MsHH: mã hàng hóa. SL: số lượng.

F = {f_1: MsKH $\rightarrow$ TP; f_2: MsKH $\rightarrow$ CTyVC; f_3: MsKH, MsHH $\rightarrow$ SL; f_4:TP $\rightarrow$ CtyVC}

Khóa là: {MsKH, MsHH}

Đạt dạng chuẩn 1 không đạt dạng chuẩn 2 vì: CtyVC không phụ thuộc đầy đủ vào khóa.

Sử dụng PP phân rã để nâng cấp lược đồ quan hệ Q:



Kết quả: $\mathbb{C}$ = { <Q_1(TP, CtyVC), $F_1 = \{f_4\}$>;

<Q_{21}(MsKH, TP), $F_{21} = \{f_1\}$>;

<Q_{22}(MsKH, MsHH, SL), $F_{22} = \{f_3\}$> }

LĐCSDL trên đạt dạng chuẩn BC, bảo toàn thông tin, bảo toàn PTH;

Nhận xét:

Thuật toán phân rã như trên là bảo toàn thông tin theo đinh lý delobel.

Tất cả các quan hệ kết quả đều đạt dạng chuẩn BC.

Tùy theo thứ tự các PTH được xét trong quá trình phân rã, mà kết quả và số lượng quan hệ con có thể khác nhau.

Hậu quả:

Thuật toán có thể dẫn đến 1 lược đồ CSDL không bảo toàn phụ thuộc hàm.

Kết quả thu được có thể chứa một quan hệ con mà ngữ nghĩa của nó không có ích cho ứng dụng thực tế.

Thông thường PTH được chọn ưu tiên là PTH không chứa khóa, và là PTH gây chất lượng xấu của lược đồ quan hệ (như PTH không đầy đủ hoặc các PTH bắc cầu vào khóa).

Ví dụ: Nếu thứ tự chọn PTH là: f_3 , f_4 , f_1 thì sẽ cho lđcsdl như sau:

Kết quả: **C** = {<Q_1(MsKH, MsHH, SL), F_1 = {f_3 }>; <Q_2(TP, CtyVC), F_2 = {f_4}>;

<Q_3(MsKH, TP), F_3 = {f_1}>; <Q_4(MsKH, MsHH), F_4 ={} }

BÀI TẬP

Bài 1: Cho LĐCSDL có các phụ thuộc hàm F = { f_1: A→BC; f_2: C→B} và 2 quan hệ sau:Q_1(A B C), Q_2(B C).

a. Xác định tập phụ thuộc hàm trên từng quan hệ.

b. Xác định dạng chuẩn cao nhất của LĐCSDL.

Giải:

a. F_{Q1} ={ f_1: A → BC; f_2: C → B}

F_{Q2} ={ f_2: C →B}

b. Q_1(A B C) đạt dạng chuẩn 2 không đạt dạng chuẩn 3 vì:

B phụ thuộc bắc cầu vào khóa A: A → C; C → B; C-/->A

Q_2(B C) đạt dạng chuẩn BC.

Bài 2: Xét lược đồ quan hệ Q(MsKH, TP, CTyVC, MsHH, SL)

MsKH: Mã số Khách hàng. TP: Thành phố của nhà cung cấp. CtyVC: công ty vận chuyển hàng. MsHH: mã hàng hóa. SL: số lượng.

F = { f_1: MsKH → TP, CTyVC; f2: MsKH, MsHH → SL; f3: TP → CtyVC}

a. Xác định khóa của quan hệ?

b. Xác định dạng chuẩn của quan hệ?

c. Có thể tách quan hệ trên thành các quan hệ nào để lđcsdl đạt dạng chuẩn cao nhất.

Giải:

a. Khóa: {MsKH, MsHH}

b. Đạt dạng chuẩn 1 không đạt dạng chuẩn 2 vì: CtyVC không ptđđ vào khóa.

c. Thay Q thành 2 quan hệ: KH(MsKH, TP, CtyVC); CTVC(MsKH, MsHH, SL)

lđ trên đạt dạng chuẩn 2 nhưng không đạt dạng chuẩn 3 vì quan hệ KH không đạt dạng chuẩn 3: CtyVC phụ thuộc bắc cầu vào khóa.

Nên Tách thành các quan hệ:

KH(MsKH, TP); PC(TP, CtyVC); CTVC(MsKH, MsHH, SL)

Các quan hệ trên đạt dạng chuẩn BC.

Bài 3: Xét lược đồ quan hệ TKBCoiThi(GT, Ngay, Giờ, Phong, Mon, GV)

F = { f_1: GT $\rightarrow$ Ngay,Gio,Phong,Mon; f_2:Mon $\rightarrow$ GV; f_3:Ngay,gio,phong $\rightarrow$Mon}

a. Xác định khóa của quan hệ?

b. Xác định dạng chuẩn của quan hệ?

c. Có thể tách quan hệ trên thành các quan hệ nào để lđcsdl đạt dạng chuẩn cao nhất.

BÀI 7: CHỈ MỤC (INDEX)

SQL server là một hệ quản trị CSDL quan hệ mạnh hiện nay. Chỉ mục là một đối tượng tồn tại trong hệ QTCSDL SQL Server. Để sinh viên có thể sử dụng tốt hệ quản trị CSDL SQL Server, bắt đầu từ bài 7 chúng ta sẽ làm quen với các đối tượng cơ bản có trong hệ QTCSDL SQL Server. Sau khi nghiên cứu bài này sinh viên có thể:

Nắm được khái niệm về chỉ mục, đây là một đối tượng quan trọng đối với một CSDL, đặc biệt là CSDL lớn. Index được thiết lập từ một hoặc nhiều cột dữ liệu của bảng dữ liệu. Các giá trị của khóa Index sẽ được sắp xếp và lưu trữ theo một danh sách (bảng khác). Mỗi giá trị trong khóa Index là duy nhất trong danh sách, mỗi giá trị khóa Index sẽ liên kết đến giá trị trong bảng dữ liệu (liên kết dạng con trỏ). Việc lưu trữ dữ liệu của bảng có khóa Index được thực hiện theo cấu trúc cấy B-Tree nhằm tăng tốc độ tìm kiếm, truy xuất dữ liệu trong quá trình thực hiện truy vấn.

7.1 KHÁI NIỆM

Index (chỉ mục) thực chất là một dạng tương tự như phần mục lục của một cuốn sách hay của một cuốn từ điển, các giá trị trong bảng được sắp theo cột, các giá trị trong cột đó sẽ được sắp xếp theo một trật tự dựa vào dữ liệu của bảng đó, giúp cho việc truy xuất trở nên nhanh hơn. Một chỉ mục trên một thuộc tính A của bảng T sẽ giúp tăng tốc quá trình tìm kiếm các giá trị cố định trên thuộc tính A.

LỢI ÍCH CỦA INDEX

- Là 1 trong những cách tốt nhất để giảm hoạt động đọc/ghi của đĩa cứng.
- Cho phép SQL Server tìm thấy dữ liệu mà không cần phải duyệt qua toàn bộ table.

NHƯỢC ĐIỂM CỦA INDEX

Việc tạo Index đôi khi rất tốn thời gian. Đối với dự án có CSDL nhỏ, vừa thì không đáng kể nhưng với CSDL lớn thì việc này rất tốn thời gian.

Do index là tạo 1 cấu trúc bảng, nên dẫn đến tốn tài nguyên. Nếu có sự thay đổi về dữ liệu thì index cũng được update theo.

Trong thực tế, hầu hết các trường hợp, việc sử dụng index vẫn là lựa chọn hàng đầu.

7.2 PHÂN LOẠI

Có 2 loại Index chính thường được dùng là Clustered Index (chỉ mục kết cụm) và Nonclustered Index (chỉ mục không kết cụm).

Clustered Index: Mặc định khi tạo khóa chính cho 1 table nào đó tức là ta đã tạo 1 Clustered Index. Một bảng mà không có Clustered Index được gọi là 1 Heap table. Trong clustered index, các dữ liệu ở cấu trúc bảng trong được sắp xếp một cách vật lý, tức là dữ liệu bảng trong được sắp xếp đúng theo thư mục cây dựa vào bảng chữ cái.

Non-Clustered Index: không sắp xếp dữ liệu theo một trật tự vật lý như clustered mà là "loạn xà ngầu" trong bảng thông tin, miễn sao nó nằm trong một logic do index qui định. Có thể có các biến thể như Covering Index (chỉ mục bao phủ), Filtered Index (chỉ mục lọc)… Ngoài ra còn có các loại Index khác như Full Text Index (chỉ mục toàn văn), Spatial Index (chỉ mục không gian), XML Index (chỉ mục XML).

7.3 TẠO INDEX

```
CREATE INDEX <Tên_index>
ON <Tên_bảng> (Cột1 [ASC|DESC], ..., n)
```

Ví dụ:

```
CREATE INDEX ID_PHG ON NHANVIEN (PHG)
--
CREATE INDEX ID_PHANCONG
ON PHANCONG (SODA, MA_NVIEN DESC)
```

7.4 XÓA INDEX

```
DROP INDEX <Tên_index> ON <Tên_bảng>
```

Ví dụ:

```
DROP INDEX ID_PHG ON NHANVIEN

DROP INDEX ID_PHANCONG ON PHANCONG
```

TÓM TẮT

Trong bài này, sinh viên làm quen với các khái niệm về Index. Sử dụng Index là một cách rất hữu hiệu để tăng hiệu năng thực hiện các câu lệnh SQL. Index trong SQL cũng giống như mục lục của những quyển sách, giúp SQL Server xác định chính xác nơi dữ liệu được lưu trữ. Index là 1 trong những cách tốt nhất để giảm hoạt động của đĩa cứng (disk I/O) và logical reads, cho phép SQL Server tìm thấy dữ liệu mà không cần phải quét toàn bộ Table.

Việc tạo Index đôi khi rất tốn thời gian. Đối với dự án có CSDL nhỏ, vừa thì không đáng kể nhưng với CSDL lớn thì việc này là rất mất thời gian. Do index là tạo 1 cấu trúc bảng nên dẫn đến tốn tài nguyên. Nếu có sự thay đổi về dữ liệu thì index cũng được update theo. Tuy nhiên, trong thực tế, trong hầu hết các trường hợp, việc sử dụng index vẫn là lựa chọn hàng đầu.

Tóm lại, chúng ta đã tìm hiểu các vấn đề cơ bản của Index trong SQL Server và các vấn đề cần lưu ý khi triển khai Index. Điều đó không có nghĩa là chúng ta đã nắm được đầy đủ về Index trong Server. Có một số loại Index mà chúng ta cần tìm hiểu thêm, như index cho cột có kiểu là XML, Filtered Index, Spatial Index được hỗ trợ từ phiên bản SQL Server 2008.

CÂU HỎI ÔN TẬP

Câu 1: Khái niệm Index? Khi nào dùng Index?

Câu 2: Nêu ưu và nhược điểm của Index?

BÀI 8: KHUNG NHÌN (VIEW)

Sau khi học xong bài này, sinh viên có thể:

- *Nắm được khái niệm về View (khung nhìn), view là một bảng tạm thời, có cấu trúc như một bảng, view không lưu trữ dữ liệu mà nó được tạo ra khi sử dụng, view là đối tượng thuộc CSDL. View được tạo ra từ câu lệnh truy vấn dữ liệu (lệnh Select), truy vấn từ một hoặc nhiều bảng dữ liệu.*
- *View được sử dụng khai thác dữ liệu như một bảng dữ liệu, chia sẻ nhiều người dùng, an toàn trong khai thác, không ảnh hưởng dữ liệu gốc. Có thể thực hiện truy vấn dữ liệu trên cấu trúc của View.*

8.1 KHÁI NIỆM

View có thể được coi như là một bảng ảo (virtual table) hay một truy vấn được lưu trữ. Các dữ liệu truy cập thông qua một view không được lưu trữ trong CSDL như là một đối tượng riêng biệt. Dữ liệu được lưu trữ trong CSDL thông qua câu lệnh SELECT. Tập kết quả của câu lệnh SELECT lưu trữ trong bảng ảo thông qua view. Người dùng có thể sử dụng bảng ảo này bằng cách tham chiếu đến tên view thông qua câu lệnh Transact-SQL, cách thực hiện giống như cách tham chiếu đến một table.

View được dùng trong các trường hợp sau:

Hạn chế người dùng truy cập đến các dòng trong một bảng. Ví dụ, chỉ cho phép một nhân viên chỉ thấy các dòng thông tin liên quan đến công việc của mình trong một bảng theo dõi công việc.

Hạn chế người dùng truy cập đến các cột nào đó trong một bảng. Ví dụ, cho phép các nhân viên xem tên, văn phòng, điện thoại nơi làm việc, bộ phận trong một bảng nhân viên, nhưng lại không cho phép họ nhìn thấy bất kỳ cột thông tin về tiền lương hoặc thông tin riêng của cá nhân nào.

Liên kết nhiều cột dữ liệu từ các bảng để tạo ra bảng mới, trông giống như một bảng duy nhất.

Tạo ra thông tin tổng hợp thay vì cung cấp chi tiết. Ví dụ, hiển thị giá trị tổng, giá trị lớn nhất, nhỏ nhất của một cột...

View được tạo ra bằng cách dùng câu lệnh SELECT để lấy các dữ liệu. Các bảng dữ liệu tham chiếu bởi các câu lệnh SELECT được gọi là các bảng cơ sở.

8.2 TẠO VIEW

```
CREATE VIEW [< owner > . ] view_name [ ( column [
,...n ] ) ]
[ WITH < view_attribute > [ ,...n ] ]
AS
```

Giải thích:

Owner: Là tên của người dùng đã sở hữu view..

view_name: Tên của view.

column: Là tên cột của view, nếu cột không được chỉ định, các cột sẽ có cùng tên với các cột trong câu lệnh SELECT.

n: có thể chỉ ra nhiều cột.

AS: bắt đầu phần thân của view, nơi đây sẽ là câu lệnh select. Trong đó, có thể tham chiếu đến nhiều table và view khác. Trong thân của view không được dùng các từ khóa sau:

- COMPUTE hoặc COMPUTE BY
- Chỉ dùng ORDER BY nếu có phát biểu TOP phía trên câu lệnh SELECT
- INTO: Tham chiếu một bảng tạm thời hoặc một biến bảng

Một view có thể được tạo ra chỉ trong CSDL hiện tại, có thể tham chiếu tối đa là 1.024 cột.

Cho database QLBH gồm 4 bảng sau:

KHACHHANG(MAKH, TENKH, DCKH, DTKH)

HOADON(SOHD, NGAY, MAKH)

MATHANG(MAMH, TENMH, QUYCACH, GIAMUA)

CTHD(SOHD, MAMH, SL, GIABAN, THANHTIEN)

Trong ví dụ dưới đây, view có tên "Tonghop" trong database "QLBH" sẽ lấy dữ liệu từ 3 bảng cơ sở để thống kê tổng trị giá của từng hóa đơn:

```
CREATE VIEW Tonghop
AS
 SELECT H.SOHD, sum(GIABAN*SL) as [Tổng trị giá]
 FROM HOADON H, MATHANG M, CTHD C
 WHERE H.SOHD = C.SOHD and M.MAMH = C.MAMH
 GROUP BY H.SOHD
```

Ta có thể tham chiếu đến view "Tonghop" giống như cách tham chiếu đến một table:

```
SELECT *
FROM Tonghop
```

Một view có thể tham chiếu đến một view khác, ở DB trên, ta có thể xây dựng view thống kê tổng tiền của từng khách hàng đã mua hàng. Ta xây dựng view có tên "Thongke", câu lệnh select bên trong sẽ gọi lại view "Tonghop":

```
CREATE VIEW Thongke
AS
 SELECT K.MAKH, TENKH, sum([Tổng trị giá]) as [Tổng tiền]
 FROM KHACHHANG K, HOADON H, Tonghop T
 WHERE K.MAKH = H.MAKH and H.SOHD = T.SOHD
 GROUP BY K.MAKH, TENKH
```

8.3 HIỆU CHỈNH VIEW

```
ALTER VIEW [< owner > . ] view_name [ ( column [
,...n ] ) ]
[ WITH < view_attribute > [ ,...n ] ]
AS
   select_statement
```

8.4 XÓA VIEW

```
DROP VIEW [ schema_name. ] view_name [ ...,n ] [ ; ]
```

schema_name: Là tên của các lược đồ mà view trực thuộc.

view_name: Là tên của view cần xóa.

TÓM TẮT

View (khung nhìn/virtual table - bảng ảo) thực chất là một đối tượng mà bên trong nó chỉ lưu trữ duy nhất một câu lệnh SELECT dùng để chỉ định các cột, các dòng dữ liệu bên dưới các bảng dữ liệu mà nó chọn lựa ra để hiển thị cho người sử dụng xem hoặc cập nhật. Với nguyên tắc này, bạn có thể hiển thị ra đúng các thông tin tối thiểu mà người sử dụng cần dùng, không cần thiết phải hiển thị ra tất cả các thông tin hiện đang được lưu trữ bên trong bảng (đáp ứng được tính bảo mật thông tin). Ngoài ra View còn giúp những người sử dụng dễ dàng truy xuất đến các thông tin mà họ đang cần, khi đó đơn giản sẽ thông qua việc thực hiện các truy vấn trực tiếp đến các bảng ảo mà không cần quan tâm các thông tin này đang được lưu trữ trong những bảng dữ liệu nào (đáp ứng được tính dễ sử dụng).

Trong thực tế, bạn thường tạo ra các bảng ảo để lưu trữ các thông tin cho các loại báo cáo đơn giản hoặc dữ liệu của các màn hình nhập liệu phức tạo có liên kết dữ liệu với nhiều bảng khác hoặc các màn hình tra cứu thông tin cho các người sử dụng.

Lưu ý: bảng ảo hoàn toàn không lưu trữ dữ liệu một cách riêng lẻ. Các dữ liệu được hiển thị trong bảng ảo sẽ được lấy từ bên dưới dữ liệu của các bảng cơ sở trong CSDL hiện hành. Tuy nhiên bạn vẫn có thể cập nhật (thêm, sửa, xóa) dữ liệu trong các bảng ảo như là đang cập nhật dữ liệu trong các bảng cơ sở.

CÂU HỎI ÔN TẬP

Câu 1: Khái niệm View? Khi nào dùng View? Nêu ưu và nhược điểm của View?

Câu 2: Làm các câu hỏi phần View trong bài thực hành số 2 và 3.

BÀI 9: THỦ TỤC (PROCEDURE)

Sau khi học xong bài này, sinh viên có thể:

- *Nắm được khái niệm cơ bản về chương trình con (Sub-Program) trong ngôn ngữ lập trình, cách xây dựng và vận dụng.*
- *Hiểu rõ ý nghĩa và cách sử dụng thủ tục nội tại (Stored Procedure): cách tạo, sửa và xóa, đặc biệt là cách thức truyền tham số đầu vào và đầu ra.*
- *Phân biệt những điểm giống và khác nhau giữa View và Stored Procedure.*

9.1 KHÁI NIỆM

Trong bài học trước, chúng ta đã tìm hiểu qua View, rõ ràng ưu điểm của View rất nhiều khi quản trị một CSDL. Tuy nhiên, trong thực tế, view gặp bất lợi khi có cùng một công việc nhưng chúng ta phải tạo ra nhiều View (chỉ khác nhau rất ít). Ví dụ như, công ty muốn đếm xem có bao nhiêu người đi làm vào ngày hôm qua, hôm nay hay một ngày cụ thể trong tháng... Ta phải xây dựng rất nhiều View tương ứng với từng ngày cụ thể. Công việc đếm này chỉ khác nhau về ngày cần thống kê. Thủ tục nội tại (Stored procedure) sẽ giúp chúng ta giải quyết vấn đề này.

Thủ tục nội tại trong Microsoft SQL Server cũng tương tự như thủ tục trong các ngôn ngữ lập trình khác, nó có thể:

- Chấp nhận các thông số đầu vào và trả về nhiều giá trị trong các hình thức của các thông số đầu ra.
- Chứa các câu lệnh thực hiện các công việc trong CSDL, bao gồm cả việc gọi các thủ tục khác.
- Trả về giá trị thông báo trạng thái việc gọi một thủ tục hay khối lệnh (batch) thành công hay thất bại (và lý do của sự thất bại).

- Ta có thể sử dụng câu lệnh EXECUTE để thi hành một thủ tục lưu trữ. Thủ tục lưu trữ khác với hàm (function) ở chỗ nó không trả về giá trị thay cho tên của nó và nó không thể được sử dụng trực tiếp trong một biểu thức.

  Những lợi ích của việc sử dụng thủ tục lưu trữ:

- Đã được đăng ký tại các máy chủ.
- Có thể có các thuộc tính bảo mật (chẳng hạn như các quyền) và quyền sở hữu, các chứng thực có thể được gắn kèm với nó.
- Người dùng có thể được cấp phép để thi hành một thủ tục lưu trữ mà không cần phải có quyền truy cập trực tiếp vào các đối tượng tham chiếu trong thủ tục.
- Có thể nâng cao tính bảo mật của ứng dụng.
- Tham số của thủ tục lưu trữ có thể giúp bảo vệ ứng dụng từ các cuộc tấn công SQL Injection. (*Để biết thêm thông tin, xem SQL Injection*).
- Cho phép lập trình mô-đun.
- Ta có thể tạo ra các thủ tục một lần, và gọi nó trong chương trình. Điều này có thể cải thiện khả năng bảo trì của ứng dụng và cho phép ứng dụng truy cập CSDL một cách thống nhất.
- Có thể làm giảm lưu lượng mạng.

Thủ tục lưu trữ có thể bảo vệ người dùng không thể biết các chi tiết của các bảng trong CSDL. Nếu một tập hợp các thủ tục lưu trữ hỗ trợ tất cả các chức năng kinh doanh người dùng cần phải thực hiện, người dùng không bao giờ cần truy cập vào bảng trực tiếp, họ có thể thực hiện các thủ tục được lưu trữ để thi hành các công việc kinh doanh mà họ đã quen thuộc.

9.2 PHÂN LOẠI SP

Stored Procedure có thể được chia thành 5 nhóm như sau:

1. **System Stored Prcedure**: Là những stored procedure chứa trong Master database và thường bắt đầu bằng tiếp đầu ngữ **sp_**. Các stored procedure này thuộc loại built-in và chủ yếu dùng trong việc quản lý database (administration) và security.

Ví dụ, bạn có thể kiểm tra tất cả các processes đang được sử dụng bởi user `DomainName\Administrators`, cú pháp:

```
sp_who @loginame='DomainName\Administrators'.
```

Có hàng trăm system stored procedure trong SQL Server.

2. **Local Stored Procedure**: Đây là loại thường dùng nhất. Chúng được chứa trong user database và thường được viết để thực hiện một công việc nào đó. Thông thường người ta nói đến stored procedure là nói đến loại này. Local stored procedure thường được viết bởi DBA hoặc programmer.
3. **Temporary Stored Procedure**: Là những stored procedure tương tự như local stored procedure nhưng chỉ tồn tại cho đến khi connection đã tạo ra chúng bị đóng lại hoặc SQL Server shutdown. Các stored procedure này được tạo ra trên TempDB của SQL Server nên chúng sẽ bị delete khi connection tạo ra chúng bị cắt đứt hay khi SQL Server down. Temporary stored procedure được chia làm 3 loại: **local** (bắt đầu bằng #), **global** (bắt đầu bằng ##) và stored procedure được tạo ra trực tiếp trên TempDB. Loại local chỉ được sử dụng bởi connection đã tạo ra chúng và bị xóa khi disconnect, còn loại global có thể được sử dụng bởi bất kỳ connection nào. Permission cho loại global là dành cho mọi người (public) và không thể thay đổi. Loại stored procedure được tạo trực tiếp trên TempDB khác với 2 loại trên ở chỗ ta có thể set permission, chúng tồn tại kể cả sau khi connection tạo ra chúng bị cắt đứt và chỉ biến mất khi SQL Server shut down.
4. **Extended Stored Procedure**: Đây là một loại stored procedure sử dụng một chương trình ngoại vi (external program) vốn được biên dịch thành một DLL để mở rộng chức năng hoạt động của SQL Server. Loại này thường bắt đầu bằng tiếp đầu ngữ **xp_**. Ví dụ, *xp_sendmail* dùng đề gởi mail cho một người nào đó hay *xp_cmdshell* dùng để chạy một DOS command... Ví dụ *xp_cmdshell 'dir c:\'*. Nhiều loại extend stored procedure được xem như system stored procedure và ngược lại.
5. **Remote Stored Procedure**: Những stored procedure gọi stored procedure ở server khác.

9.3 TẠO SP

Sử dụng lệnh CREATE PROCEDURE để tạo SP, SP được lưu ở DB hiện hành.

Nếu trong SP có tạo một bảng tạm, thì bảng tạm chỉ tồn tại khi thực thi SP, bảng tạm sẽ tự động bị xóa khi thi hành xong SP (bảng tạm có tên bắt đầu bằng ký hiệu #, ví dụ: #NHANVIEN)

Trong SP không được chứa các câu lệnh: CREATE PROCEDURE, CREATE RULE, CREATE VIEW, CREATE TRIGGER.

Để thi hành lệnh CREATE PROCEDURE, người dùng phải là thành viên của một trong các role: sysadmin, db_owner, db_ddladmin hoặc được cấp quyền CREATE PROCEDURE.

Để thi hành lệnh CREATE PROCEDURE, người dùng phải là thành viên của một trong các role: sysadmin, db_owner, db_ddladmin hoặc được cấp quyền CREATE PROCEDURE.

Cú pháp:

```
CREATE PROC [EDURE] <Tên_Procedure>
[@<Tham_số> <Kiểu_dữ_liệu> [ = <Giá_trị_mặc_định> ]
[ OUTPUT ] ]
[ ,...n ]
[ WITH {RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION} ]
AS
    <Các_câu_lệnh_SQL>
```

Giải thích:

- Tên_Procedure: tên procedure, các SP tạm cục bộ có ký hiệu # trước tên của SP
- @<Tham_số>: tên tham số của procedure.
- <Kiểu_dữ_liệu>: kiểu dữ liệu của tham số.
- <Giá_trị_mặc_định>: giá trị mặc định của tham số.
- OUTPUT: cho phép tham số nhận giá trị trả về.

- RECOMPILE: nếu có thêm tùy chọn này thì mỗi lần thi hành SQL Server sẽ biên dịch lại SP và mã của SP không được lưu vào vùng đệm của thủ tục.
- ENCRYPTION: nếu có thêm tùy chọn này thì văn bản lệnh được mã hóa và lưu trong syscomments.

Ví dụ:

```
CREATE PROC DS_NHANVIEN
AS
   SELECT * FROM NHANVIEN
```

Thực thi SP:

```
EXEC[UTE] <Tên_Procedure> [Danh_sách_tham_số]
```

Ví dụ:

EXEC DS_NHANVIEN

9.4 SỬA SP

Cú pháp:

HIỆU CHỈNH SP:

```
ALTER PROC [EDURE] <Tên_Procedure>
[  @<Tham_so> <Kieu_du_lieu> [ = <Gia_tri_mac_dinh> ] [
OUTPUT ] ]
[ ,...n ]
[ WITH {RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION} ]
AS
    <Cac_cau_lenh_SQL>
```

Ví dụ:

```
ALTER PROC DS_NHANVIEN
AS
    SELECT * FROM nhanvien WHERE PHAI= 'Nam'
```

9.5 XÓA SP

Cú pháp:

```
DROP PROC [EDURE] <Tên_Procedure>
```

Ví dụ:

```
DROP PROC DS_NHANVIEN
```

MỘT VÀI VÍ DỤ:

Ví dụ 1 – Có tham số vào, default:

Xem danh sách nhân viên theo Phòng:

```
CREATE PROC DSNV_THEOPHONG

@Phong int = 1
AS
  SELECT * FROM NHANVIEN
  WHERE PHG = @Phong
```

Thực thi SP:

- Cách 1:

```
EXEC DSNV_THEOPHONG 4
```

- Cách 2:

```
DECLARE @P int
Set @P = 5
EXEC DSNV_THEOPHONG @P
```

Ví dụ 2 – Có tham số vào, ra:

Xóa thân nhân theo MANV:

```
CREATE PROC XOA_THANNHAN_THEOMANV
@MANV nvarchar(20), @SoNVXoa int OUTPUT
AS
  DELETE THANNHAN
  WHERE MA_NVIEN = @MaNV
  SET @SoNVXoa = @@rowcount
```

Thực thi SP:

```
DECLARE @SoNVXoa int
EXEC XOA_THANNHAN_NHANVIEN '123', @SoNVXoa OUTPUT
PRINT 'So mau tin bi xoa:' + str(@SoNVXoa,3)
```

Ví dụ 3 – Có recompile, encryption:

Tăng lương cho nhân viên theo phòng lên @Tyle lần.

```
CREATE PROC TANGLUONG_NHANVIEN
@Phong int, @Tyle Decimal(3,1), @So_NV_Tang int OUTPUT
WITH RECOMPILE, ENCRYPTION
AS
  UPDATE NHANVIEN
  SET LUONG = LUONG * @Tyle
  WHERE PHG = @Phong
  SET @So_NV_Tang = @@rowcount
```

Ví dụ 4 – Có chặn lỗi:

Thêm phòng ban:

```
CREATE PROC THEM_PHONGBAN @TENPHG NVARCHAR(40),
@MAPHG INT, @TRPHG NVARCHAR(20),
@NG_NHANCHUC SMALLDATETIME, @Loi int OUTPUT
AS
BEGIN TRY
  INSERT PHONGBAN VALUES
  (@TENPHG, @MAPHG, @TRPHG, @NG_NHANCHUC)
END TRY
BEGIN CATCH
```

```
    SET @Loi = @@error
    RAISERROR(N'Lỗi thêm dữ liệu !',10,1)
    RETURN
END CATCH
SET @Loi = @@error.
```

TÓM TẮT

Stored Procedure (Thủ tục nội tại) là một nhóm câu lệnh Transact-SQL đã được biên dịch và chứa trong SQL Server dưới một tên nào đó và được xử lý như một đơn vị (chứ không phải nhiều câu SQL riêng lẻ). Stored Procedure được kết cấu từ một kịch bản câu lệnh T-SQL, có những đặc điểm cơ bản sau:

- *Truyền tham số.*
- *Gọi thủ tục khác.*
- *Trả về các giá trị tham số, chuyển giá trị tham số cho các thủ tục được gọi.*
- *Trả về giá trị trạng thái thủ tục là thành công hay không thành công.*

Stored Procedure có nhiều ưu điểm so với thực hiện câu lệnh T-SQL từ các máy khách:

- *Lập trình theo module: Thủ tục được thiết lập trong từng CSDL một lần, có thể gọi thực hiện nhiều lần trong một ứng dụng, có thể gọi từ nhiều ứng dụng.*
- *Thực hiện nhanh hơn: Khi cần thực hiện một lượng lớn câu lệnh T-SQL, thủ tục lưu trữ thực hiện nhanh hơn vì khi máy chủ nhận được nhiều cầu lệnh cùng một lúcđều phải kiểm tra tính hợp lệ quyền của tài khoản từ máy khách và các tham số khác. Khi thủ tục cần gọi nhiều lần trên các máy khách thì thủ tục thực hiện một lầnđầu tiên, những lần sau máy khách sẽ chạy thủ tục đã được biên dịch.*
- *Làm giảm lưu lượng trên mạng: Thay vì máy khách phải gửi nhiều dòng lệnh từ các ứng dụng đến máy chủ, khi sử dụng thủ tục thì nó chỉ cần gửi một lệnh, từ đó dẫn đến lưu lượng thông tin lệnh truyền qua mạng giảm.*
- *An ninh bảo mật hơn: Khi không muốn cho một user trực tiếp khai thác một đối tượng hay bảng dữ liệu nào đó, mà cần cho user đó được khai thác thì thủ tục có thể giúp bạn gán quyền khai thác cho người đó. Việc gán quyền khai thác như nói trên sẽ giúp cho vấn đề an ninh bảo mật trong CSDL tốt hơn.*

CÂU HỎI ÔN TẬP

Câu 1: Stored Procedure là gì? Nêu những điểm giống nhau và khác nhau giữa View và Stored Procedure?

Câu 2: Cho biết nguyên lý hoạt động của Stored Procedure? Hãy phân loại Stored Procedure? Ta dùng từ khóa OUTPUT phía sau tham số trong trường hợp nào?

Câu 3: Thực hiện các câu hỏi trong bài thực hành số 2 và 3 liên quan đến Stored Procedure.

BÀI 10: HÀM (FUNCTION)

Sau khi học xong bài này, sinh viên có thể:

- Nắm được các khái niệm cơ bản về chương trình con (bổ trợ kiến thức từ bài học trước), cấu trúc và nguyên lý hoạt động của hàm nội tại.
- Cách thức tạo, sửa, xóa và cách thức gọi một hàm do người dùng định nghĩa.
- Phân biệt được các loại hàm người dùng và cách thức vận dụng.
- Hiểu rõ hơn về cách sử dụng, truyền tham số cho hàm, giá trị trả về của hàm.

10.1 KHÁI NIỆM

Hầu hết các ngôn ngữ lập trình, hay CSDL lớn, luôn luôn có một phần mở rộng cho phép người dùng tự định nghĩa một số quy tắc, hàm hoặc thủ tục. Hàm người dùng (User-defined function) giống như Stored Procedure của SQL server. Hàm người dùng cũng có thể truyền tham số nhưng không được mang thuộc tính OUTPUT, thay vào đó chúng ta dùng câu lệnh RETURN.

Function cho phép nhận tham số vào, gọi hàm bằng lệnh EXECUTE giống như gọi PROCEDURE (đối với hàm trả về kiểu vô hướng – Scalar-valued Functions).

Chú ý:

- Hàm do người dùng định nghĩa không dùng giá trị với kiểu dữ liệu *ntext, text, image, cursor, timestamp* làm giá trị trả về.
- Có thể cung cấp thông tin về lỗi nếu phát sinh.
- Có thể sử dụng các hàm do người dùng định nghĩa trong các câu lệnh SQL như SELECT.

10.2 PHÂN LOẠI

Có hai loại hàm do người dùng định nghĩa:

- Hàm người dùng trả về kiểu vô hướng (Scalar-valued Functions)
- Hàm người dùng trả về một bảng dữ liệu (Table-valued Functions)

10.3 TẠO FUNCTION

- Cú pháp hàm người dùng trả về giá trị vô hướng (Scalar-valued Functions):

```
CREATE FUNCTION <Tên_hàm> ([@<Tham_số> <Kiểu_dữ_liệu>
[ = <Giá_trị_mặc_định>]]
[ ,...n ])
RETURNS <Kiểu_dữ_liệu_trả_về>
[ AS ]
BEGIN
    <Thân_hàm>
    RETURN <giá_trị_trả_về>
END
```

Ví dụ 1: Tính tổng 2 số nguyên.

```
CREATE FUNCTION TONG(@a INT, @b INT) RETURNS INT
AS
    RETURN @a+@b
```

Gọi hàm:

- **Cách 1:**

```
PRINT 'TONG: ' + STR(DBO.TONG(6,3))
```

- **Cách 2:**

```
DECLARE @TONG INT
SET @TONG = DBO.TONG(6,3)
SELECT @TONG
```

Ví dụ 2: Hàm tự viết: Chuyển ngày về dạng DDMMYYYY.

```
Create Function dbo.dngayDDMMYYYY(@Date smalldatetime)
returns varchar(10)
As
Begin
    return convert(varchar(10),@Date,103)
End
```

Gọi hàm:

```
Select MaNV, Hoten, dbo.dngayDDMMYYYY(Ngaysinh), Phai
From NHANVIEN
```

- Cú pháp hàm người dùng trả về một bảng dữ liệu (Table-valued Functions):

```
CREATE FUNCTION [owner_name.] Tên_hàm
(@tên_biến [AS] Kiểu_giá_trị_biến [= default ],...)
RETURNS TABLE
AS
    RETURN (1 câu_lệnh_select)
```

Chú ý: phần thân của hàm chỉ cho phép sự xuất hiện duy nhất của câu lệnh RETURN. Ngoài ra, không sử dụng bất kỳ câu lệnh nào khác trong phần thân của hàm.

Ví dụ 1:

```
create function dbo.HienthiNV()
returns table
as
    return (Select * from NHANVIEN)
```

Gọi hàm:

```
Select * from dbo.HienthiNV()
```

Ví dụ 2: Ứng dụng hàm người dùng trả về bảng dữ liệu (trường hợp có tham số truyền vào):

```
create function dbo.HienthiNV(@MaNV varchar(4))
returns Table
as
return
    (select MaNV, Hoten, Ngaysinh, Phai
    from NHANVIEN
    where MaNV like @MaNV)
```

Trong trường hợp cần phải sử dụng đến nhiều câu lệnh trong phần thân của hàm, ta sử dụng cú pháp như sau để định nghĩa hàm:

```
CREATE FUNCTION tên_hàm([danh_sách_tham_số])
RETURNS @biến_bảng TABLE định_nghĩa_bảng
AS
BEGIN
    Các_câu_lệnh_trong_thân_hàm
    RETURN
END
```

Khi định nghĩa hàm dạng này cần lưu ý một số điểm sau:

- Cấu trúc của bảng trả về bởi hàm được xác định dựa vào định nghĩa của bảng trong mệnh đề RETURNS.
- Biến @biến_bảng trong mệnh đề RETURNS có phạm vi sử dụng trong hàm và được sử dụng như là một tên bảng.
- Câu lệnh RETURN trong thân hàm không chỉ định giá trị trả về.
- Giá trị trả về của hàm chính là các dòng dữ liệu trong bảng có tên là @biến_bảng được định nghĩa trong mệnh đề RETURNS.

Ví dụ 3:

```
Create function F4(@MaNV varchar(5))
returns @Tonghop table
(
    MaPX int,
    MaNV varchar(5)
)
as
begin
    if (@MaNV is NULL) INSERT INTO @Tonghop
        select MaPX, MaNV from PHIEUXUAT --lấy tất cả
    else INSERT INTO @Tonghop
        select MaPX, N.MaNV from NHANVIEN N, PHIEUXUAT P
        where N.MaNV=P.MaNV and N.MaNV=@MaNV
    return
end
```

Thực thi:

```
select * from F4('NV01')
select * from F4(NULL)
```

10.4 HIỆU CHỈNH FUNCTION

Cú pháp (chung cho cả 3 dạng):

```
ALTER FUNCTION <Tên_Function>([@<Tham_số> <Kiểu_dữ_liệu>
[ = <Giá_trị_mặc_định> ]]
[ ,...n ] )
RETURNS <Kiểu_dữ_liệu_trả_về>
[ AS ]
BEGIN
    <Thân_hàm>
    RETURN <giá_trị_trả_về>
END
```

10.5 XÓA FUNCTION

Cú pháp:

```
DROP FUNCTION <Tên_Function>
```

Ví dụ:

Drop function TONG

MỘT VÀI VÍ DỤ:

Tính tổng lương theo Mã phòng:

```
CREATE FUNCTION TONG_LUONG_PHONG(@PHG INT) RETURNS INT
BEGIN
  DECLARE @TONGLUONG INT
  SET @TONGLUONG =
      (SELECT SUM(LUONG) FROM NHANVIEN
      WHERE PHG = @PHG)
  RETURN @TONGLUONG
END
```

Tính tổng thời gian theo tên Đề án:

```
CREATE FUNCTION TONG_TG_DA(@TENDA NVARCHAR(40))
RETURNS FLOAT
BEGIN
  DECLARE @TONG_TG FLOAT
  SET @TONG_TG =
      (SELECT SUM(THOIGIAN) FROM PHANCONG, DEAN
      WHERE TENDA = @TENDA AND SODA = MADA)
  RETURN @TONG_TG
END
```

TÓM TẮT

Cũng giống như Stored Procedure, Function (Hàm) là một đối tượng trong CSDL bao gồm một tập nhiều câu lệnh SQL được nhóm lại với nhau thành một nhóm. Điểm khác biệt giữa hàm và thủ tục là hàm trả về một giá trị thông qua tên hàm. Điều này cho phép ta sử dụng hàm như là một thành phần của một biểu thức chẳng hạn như trong các câu lệnh truy vấn hay các câu lệnh thực hiện cập nhật dữ liệu.

Hàm người dùng cũng có thể truyền tham số nhưng không được mang thuộc tính OUTPUT, thay vào đó chúng ta dùng câu lệnh RETURN.

Trong SQL có rất nhiều các hàm được định nghĩa sẵn (Được chia theo nhóm - Trong một Database, ta chọn Programmability/Functions/System Functions) như các hàm về chuỗi (String Functions), các hàm về ngày tháng (Date and Time Functions), Các hàm toán học (Mathematical Functions)... Ngoài những hàm do Hệ quản trị CSDL cung cấp sẵn, ta có thể tự xây dựng các hàm nhằm phục vụ cho mục đích riêng của mình - Các hàm do người dùng định nghĩa. Các hàm do người dùng định nghĩa thường có 2 loại: Loại 1 là Hàm trả về là một giá trị (Scalar-valued Functions) và các hàm này cũng sẽ được Hệ quản trị phân thành 2 nhóm. Loại 2 là Hàm với giá trị trả về là "dữ liệu kiểu bảng" (Table-valued Functions).

Function cho phép nhận tham số vào, gọi hàm bằng lệnh EXECUTE giống như gọi PROCEDURE (đối với hàm trả về kiểu vô hướng – Scalar-valued Functions).

Chú ý:

- *Hàm do người dùng định nghĩa không dùng giá trị với kiểu dữ liệu ntext, text, image, cursor, timestamp làm giá trị trả về.*
- *Có thể cung cấp thông tin về lỗi nếu phát sinh.*
- *Có thể sử dụng các hàm do người dùng định nghĩa trong các câu lệnh SQL như SELECT.*

CÂU HỎI ÔN TẬP

Câu 1: Function là gì? Nêu những điểm giống nhau và khác nhau giữa Function và Stored Procedure? Cho biết nguyên lý hoạt động của Function? Hãy phân loại Function?

Câu 2: Trong function, ta có dùng từ khóa OUTPUT phía sau tham số không? Giải thích?

Câu 3: Thực hiện các câu hỏi trong bài thực hành số 2 và 3 liên quan đến Function.

BÀI 11: BẪY LỖI (TRIGGER)

Sau khi học xong bài này, sinh viên có thể:

- *Nắm được cơ chế hoạt động của một Trigger.*
- *Các biến cố tác động lên một table. Liên kết hoạt động giữa 2 hay nhiều table có mối kết hợp với nhau.*
- *Hiểu rõ tính ràng buộc toàn vẹn của CSDL, từ đó có thể xây dựng và vận hành các trigger nhằm đảm bảo sự toàn vẹn dữ liệu và an toàn cho hệ thống.*

11.1 KHÁI NIỆM

Trigger là một Stored procedure đặc biệt tự động được chạy mỗi khi có một hành động nào đó xảy ra có liên quan đến nó. Các hoạt động xảy ra giúp trigger hoạt động: Insert, Delete, Update (DML). TRIGGER không được thực thi một cách tường minh nên cần thận trọng khi dùng trigger.

Trigger hoạt động gần giống các hàm bắt sự kiện trong javascript. Nó chỉ được kích hoạt khi xảy ra một hành động mà cụ thể là các thao tác Insert, Delete, Update trên một table. Nó hoạt động một các thụ động nên chúng ta không thể biết trigger thực thi khi nào.

11.2 SỬ DỤNG TRIGGER

Sử dụng trigger nhằm mục đích:

- Đảm bảo tính ràng buộc toàn vẹn cho CSDL.
- Kiểm soát dữ liệu hiện có trong CSDL khi có thay đổi giá trị của một mẩu tin trong bảng.
- Kiểm tra dữ liệu mới nhập vào có thỏa mãn điều kiện không.

- Kiểm chứng khi xóa mẩu tin trong bảng.
- Tự động cập nhật dữ liệu cho bảng B khi dữ liệu bảng A thay đổi (khi 2 bảng có mối quan hệ với nhau).

11.3 NGUYÊN TẮC HOẠT ĐỘNG

- Triggers được thực hiện tự động sau khi lệnh INSERT, UPDATE, hoặc DELETE được thực hiện trên một table mà trigger đó được định nghĩa. Còn các constraints và INSTEAD OF trigger sẽ được kiểm tra trước khi lệnh INSERT, UPDATE, hoặc DELETE thực hiện.
- Constraints sẽ được kiểm tra trước trigger.
- Một table có thể có nhiều Triggers cho một action. Một trigger có thể được định nghĩa cho nhiều action.
- Thứ tự thi hành sẽ là: trigger INSTEAD OF, các constraint, và sau cùng là trigger AFTER. Khi có nhiều trigger trong một table, thì table owner có thể dùng procedure hệ thống sp_settriggerorder để chỉ định trigger đầu và trigger cuối để thực thi. Thứ tự của các trigger còn lại không thể sắp xếp được.
- User phải có quyền để thực hiện tất cả các lệnh mà được định nghĩa trong Triggers
- Table Owners không thể tạo ra các Triggers trên Views hoặc Temporary Tables nhưng có thể tham chiếu đến view và temporary.
- Khi có nhiều trigger trong một table, thì table owner có thể dùng procedure hệ thống **sp_settriggerorder** để chỉ định trigger đầu và trigger cuối để thực thi. Thứ tự của các trigger còn lại không thể sắp xếp được.
- User phải có quyền để thực hiện tất cả các lệnh mà được định nghĩa trong Triggers.
- Table Owners không thể tạo ra các Triggers trên Views hoặc Temporary Tables nhưng có thể tham chiếu đến view và temporary.
- Triggers không trả kết quả về.
- Triggers có thể điều khiển multi-row actions: một hành động INSERT, UPDATE, hoặc DELETE gọi một trigger có thể ảnh hưởng lên nhiều dòng dữ liệu, Ta có thể chọn:

- Xử lý tất cả các dòng cùng với nhau trong trường hợp các dòng ảnh hưởng phải thỏa điều kiện của trigger.
- Xử lý từng dòng thỏa điều kiện.

11.4 PHÂN LOẠI

Có 4 loại trigger sau:

1. Insert trigger:

Được thực hiện mỗi khi mẩu tin mới được chèn vào bảng. Một bảng tạm Inserted sẽ được sinh ra để chứa mẩu tin cần chèn.

2. Delete trigger:

Được thực hiện mỗi khi các mẩu tin trong bảng bị xóa. Một bảng tạm Deleted được sinh ra để lưu các mẩu tin bị xóa.

3. Update trigger:

Được thực hiện khi các bản các mẩu tin của bảng được cập nhật. Hai bảng Inserted và Deleted sẽ được sinh ra. Bảng Inserted sẽ lưu thông tin các mẩu tin mới được sửa, bảng Deleted sẽ lưu thông tin các mẩu tin cũ.

4. Instead of trigger:

Trigger cho phép cập nhật dữ liệu các bảng thông qua view có liên kết nhiều bảng, hoặc bẫy trước khi các constraint có tác dụng.

Chú ý:

- Một bảng có nhiều trigger.
- Mỗi một trigger có tên duy nhất.
- Trong trigger thường dùng mệnh đề IF EXISTS.
- Sử dụng trigger trong toàn vẹn dữ liệu.
- Sử dụng trigger trong ràng buộc tham chiếu.

11.5 TẠO TRIGGER

Cú pháp:

```
CREATE TRIGGER <Tên_ Trigger> ON <Tên_bảng>
FOR < [INSERT] [,] [UPDATE] [,] [DELETE]>
AS
    <Các_câu_lệnh_SQL>
```

Chú thích:

<Tên_ Trigger>: tên Trigger.

<Tên_bảng>: tên bảng cần định nghĩa Trigger.

INSERT, UPDATE, DELETE: định nghĩa thao tác để kích hoạt Trigger.

Ví dụ 1: Tạo ràng buộc lương nhân viên phải là số dương.

```
CREATE TRIGGER LUONG_DUONG ON NHANVIEN
FOR INSERT, UPDATE
AS
BEGIN
  IF (SELECT COUNT(*) FROM INSERTED WHERE LUONG < 0) > 0
  BEGIN
     Print N'Luong phải là số dương'
     Rollback Tran
  END
END
```

Ví dụ 2: Ràng buộc 0 <= LUONG <= 100000

```
CREATE TRIGGER LUONG_DUONG ON NHANVIEN
FOR INSERT, UPDATE
AS
BEGIN
 IF (SELECT COUNT(*) FROM INSERTED
      WHERE LUONG < 0 OR LUONG > 100000) > 0
 BEGIN
     Print '0 <= LUONG <= 100000'
     Rollback Tran
 END
END
```

11.6 HIỆU CHỈNH TRIGGER

Cú pháp:

```
ALTER TRIGGER <Tên_ Trigger> ON <Tên_bảng>
FOR <[INSERT] [,] [UPDATE] [,] [DELETE]>
AS
    <Các_câu_lệnh_SQL>
```

11.7 XÓA TRIGGER

Cú pháp:

```
DROP TRIGGER <Ten_ Trigger>
```

Ví dụ:

```
DROP TRIGGER LUONG_DUONG
```

11.8 MỘT VÀI VÍ DỤ

- **Ràng buộc lương nhân viên phải tăng:**

```
CREATE TRIGGER LUONG_TANG ON NHANVIEN
FOR UPDATE
AS
IF UPDATE(LUONG)
BEGIN
  IF (SELECT COUNT(*) FROM INSERTED I, DELETED D
          WHERE D.LUONG >= I.LUONG AND I.MANV=D.MANV) > 0
         BEGIN
            Print N'Lương nhân viên phải tăng'
            Rollback Tran
         END
END
```

- **Ràng buộc nhân viên phải từ 18 tuổi trở lên:**

```
CREATE TRIGGER TUOI_18_TROLEN ON NHANVIEN
FOR INSERT, UPDATE
AS
IF EXISTS (SELECT * FROM INSERTED
    WHERE DATEADD(YY, 18, NGSINH) > GETDATE())
BEGIN
     Print N'Nhân viên phải từ 18 tuổi trở lên'
     Rollback Tran
END
```

- **Ràng buộc không cho phép thêm, sửa bảng NHANVIEN nếu PHG không có trong bảng PHONGBAN:**

```
CREATE TRIGGER TONTAI_PHONG ON NHANVIEN
FOR INSERT, UPDATE
AS
  IF NOT EXISTS (SELECT * FROM INSERTED, PHONGBAN
     WHERE PHG = MAPHG)
  BEGIN
     Print N'Mã phòng chưa tồn tại'
     Rollback Tran
  END
```

- **Ràng buộc không cho sửa MAPHG trong bảng PHONGBAN:**

```
CREATE TRIGGER KHONG_SUA_KHOACHINH ON PHONGBAN
FOR UPDATE
AS
  IF UPDATE(MAPHG)
  BEGIN
     Print N'Không được sửa khóa chính'
     Rollback Tran
  END
```

- **Sửa MAPHG trong bảng PHONGBAN thì phải sửa luôn những mẫu tin có liên quan trong bảng NHANVIEN:**

```
CREATE TRIGGER SUA_DAYCHUYEN ON PHONGBAN
FOR UPDATE
AS
  IF UPDATE(MAPHG)
  BEGIN
      UPDATE NHANVIEN
      SET PHG = I.MAPHG
      FROM NHANVIEN, DELETED D, INSERTED I
      WHERE PHG = D.MAPHG
  END
```

- **Không cho xóa NHANVIEN nếu nhân viên đó có trong bảng PHANCONG:**

```
CREATE TRIGGER KHONG_XOA_NV_CO_PHANCONG ON NHANVIEN
FOR DELETE
AS
  IF (SELECT COUNT(*) FROM DELETED, PHANCONG
  WHERE MANV = MA_NVIEN) > 0
  BEGIN
      Print N'Nhân viên có trong phân công'
      Rollback Tran
  END
```

TÓM TẮT

Trigger là một loại đặc biệt của thủ tục lưu trữ, nó tự động thực thi khi một sự kiện xảy ra trong các máy chủ CSDL. DML (Data Manipulation Language – ngôn ngữ thao tác dữ liệu) gây nên khi người dùng cố gắng sửa đổi dữ liệu thông qua một ngôn ngữ thao tác dữ liệu (sự kiện INSERT, UPDATE, hoặc DELETE trên một Table hoặc View). Trigger không được thực thi một cách tường minh nên cần thận trọng khi dùng trigger.

Trigger là một thủ tục không có tham số. Một table có thể có nhiều trigger. Trigger thường dùng để kiểm tra các ràng buộc mà không thể khai báo trên table như các ràng buộc liên thuộc tính-liên quan hệ, liên bộ-liên quan hệ, ràng buộc do sự hiện diện của chu trình.

Sử dụng lệnh ROLLBACK TRAN trong trigger để bãi bỏ phát biểu cập nhật khi cần thiết.

Các loại trigger: Insert trigger, Delete trigger, Update trigger và Istead of trigger.

CÂU HỎI ÔN TẬP

Câu 1: Khái niệm về trigger? Tại sao phải dùng trigger trong một hệ thống CSDL?

Câu 2: Phân loại trigger và cho biết nguyên lý hoạt động của từng loại?

Câu 3: Ta sử dụng Check Constraint lúc tạo field cho table khi nào? Trong ngữ cảnh đó, ta dùng trigger được không? Giải thích?

Câu 4: Thực hiện các câu hỏi trong bài thực hành số 2 và 3 liên quan đến trigger.

BÀI 12: MỘT SỐ KIẾN THỨC NÂNG CAO

Sau khi học xong bài này, sinh viên có thể:

- *Nắm được khái niệm, cách dùng, ý nghĩa của kiểu dữ liệu cursor.*
- *Hiểu và thực hiện được việc mã hóa dữ liệu trong SQL Server .*
- *Hiểu và thực hiện được việc bảo mật, phân quyền trong SQL Server.*

12.1 KIỂU DỮ LIỆU CURSOR

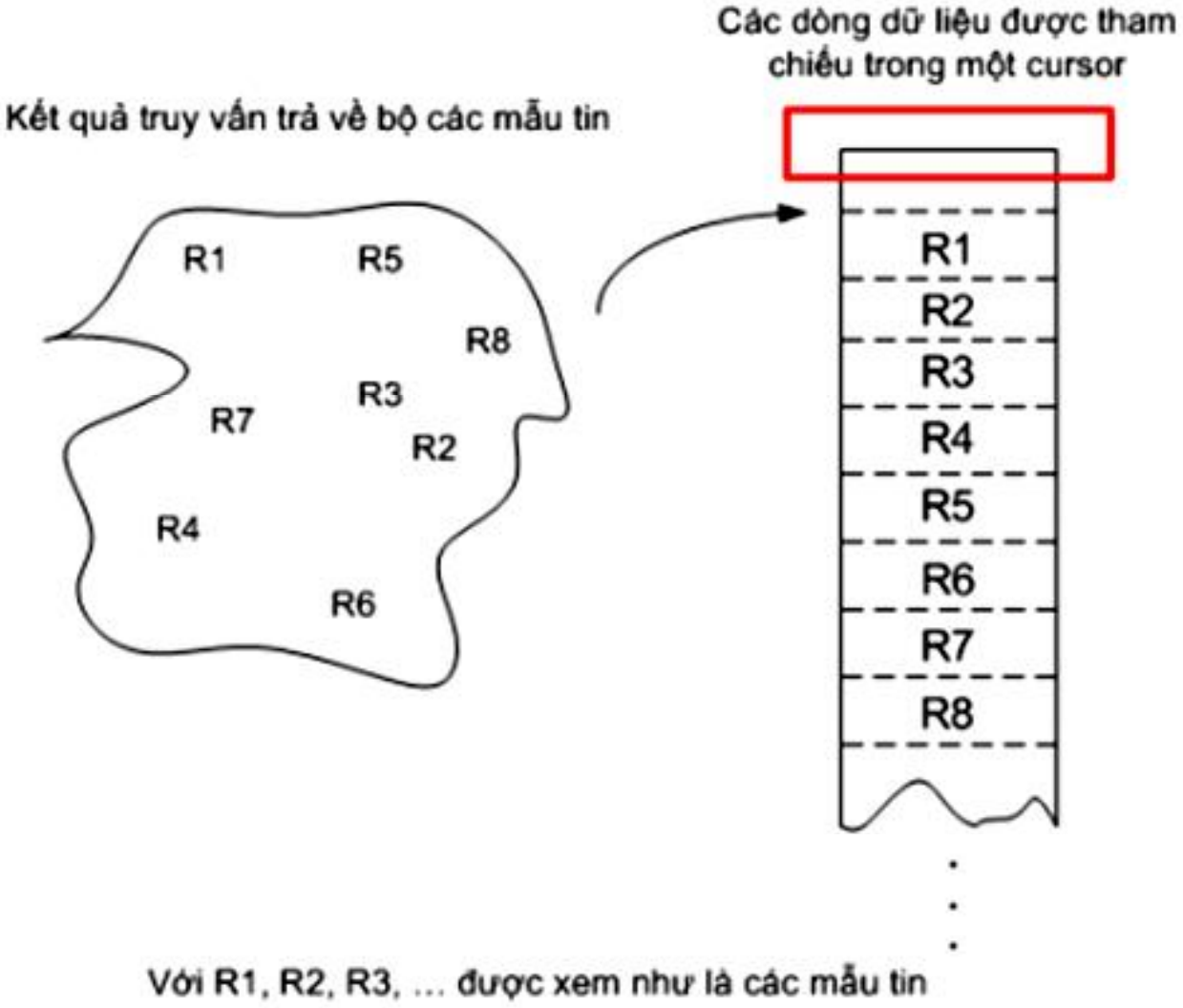
Phần lớn các CSDL quan hệ thường làm việc trên dữ liệu của nhiều dòng mẫu tin, còn gọi là một bộ các mẫu tin. Ví dụ lệnh SELECT kết quả luôn trả về nhiều dòng dữ liệu hơn là một dòng dữ liệu.

Ngược lại đối với một số ngôn ngữ lập trình hoặc bên trong các ứng dụng thì người lập trình vẫn còn các thói quen xử lý và tính toán dữ liệu trên từng dòng riêng lẻ. Để đáp ứng được yêu cầu này của các người lập trình, muốn làm việc chỉ trên từng dòng dữ liệu tại thời điểm hiện hành, Microsoft SQL Server tạo ra một kiểu dữ liệu đó chính là kiểu cursor (kiểu con trỏ).

Có thể hình dung kiểu dữ liệu cursor giống như một cuốn sổ danh bạ chứa thông tin liên lạc của các khách hàng giao tác trong một công ty. Bằng cách dò tìm thủ công, ta sẽ phải sử dụng đến mắt và tay để tham chiếu đến tên của các khách hàng bất bỳ trong sổ danh bạ đó. Ta có thể di chuyển lên, xuống hoặc qua trang để tìm ra các khách hàng mong muốn, nhưng tại thời điểm hiện hành, tay và mắt của mình chỉ đứng tại một khách hàng mà thôi.

Hoạt động của kiểu dữ liệu cursor trong Transaction-SQL hoàn toàn giống như ví dụ minh họa ở trên. Tuy nhiên, cursor có nhiều kiểu khác nhau cho phép ta có thể

chọn lựa để định nghĩa theo đúng yêu cầu mà mình mong muốn. Tùy thuộc vào kiểu cursor đã định nghĩa mà việc đọc và cập nhật dữ liệu sẽ có hiệu lực như thế nào.



Hình 12.1: So sánh cơ chế cursor và bộ các mẫu tin

12.1.1 Khái niệm

Cursor là một kiểu dữ liệu ánh xạ đến một tập các dòng dữ liệu được trả về từ kết quả của một câu truy vấn (select). Cursor cho phép duyệt tuần tự qua các dòng dữ liệu và đọc giá trị từng dòng. Thể hiện của cursor là 1 biến, nhưng biến này có thể không bắt đầu bằng ký tự '@'.

Vị trí hiện hành của cursor có thể được dùng như điều kiện trong mệnh đề where của lệnh update hoặc delete: cho phép cập nhật/xóa dữ liệu (dữ liệu thật sự trong CSDL) tương ứng với vị trí hiện hành của cursor.

12.1.2 Khai báo Cursor

Cú pháp SQL92 chuẩn:

```
Declare cursor_name [Insensitive] [Scroll] Cursor
For select_statement
[For {Read only | Update [of column_name [,…n]]}]
```

Cú pháp T_SQL mở rộng:

```
Declare cursor_name Cursor
[Local | Global]
[Forward_only | Scroll]
[Static | Dynamic]
[Read_only]
For select_statement
[For Update [of column_name [,…n]]]
```

Lưu ý: *Tên cursor trong các cách khai báo không bắt đầu bằng ký tự '@'.*

Giải thích:

- Insensitive/static: nội dung của cursor không thay đổi trong suốt thời gian tồn tại, trong trường hợp này, cursor chỉ là read only.
- Dynamic: trong thời gian tồn tại, nội dung của cursor có thể thay đổi nếu dữ liệu trong các bảng liên quan có thay đổi.
- Local: cursor cục bộ, chỉ có thể sử dụng trong phạm vi một khối (query batch) hoặc một thủ tục/hàm.
- Global: cursor toàn cục, có thể sử dụng trong một thủ tục/hàm hay một query batch bất kỳ hoặc đến khi bị hủy một cách tường minh.
- Forward_only: cursor chỉ có thể duyệt một chiều từ đầu đến cuối.

- Scroll: có thể duyệt lên xuống cursor tùy ý (duyệt theo đa chiều).
- Read only: có thể đọc từ cursor, không thể sử dụng cursor để update dữ liệu trong các bảng liên quan (ngược lại với "for update…").
- Update [of column_name [,…n]]: danh sách các field có thể update dữ liệu. Nếu chỉ có update thì tất cả các field đều có thể cập nhật dữ liệu.
- Mặc định khai báo cursor nếu không chỉ ra các tùy chọn thì cursor có các tính chất:
  - Global , Forward_only, Read only hay "for update" tùy thuộc vào câu truy vấn, Dynamic

12.1.3 Phân loại Cursor

Có 3 loại cursor:

- Static – Con trỏ tĩnh
- Keyset – Con trỏ keyset
- Dynamic – Con trỏ động

Static cursor:

Khi con trỏ này được tạo ra, những mẫu tin được copy vào một bảng tạm thời trong CSDL temdp. Con trỏ làm việc với bảng tạm thời, do đó những thay đổi dữ liệu trong bảng gốc không có tác động đến con trỏ. Con trỏ static không cho phép cập nhật dữ liệu.

Keyset cursor:

Tập dữ liệu trong khóa được copy vào bảng tạm trong temdb, tập dữ liệu trong khóa phải duy nhất. Cho phép thay đổi các mục dữ liệu không phải là khóa. Việc thay đổi các mục không phải là khóa sẽ có tác động ngay đến con trỏ. Việc xóa mẫu tin hoặc thay đổi khóa có thể làm con trỏ bị lỗi. Việc thêm mới mẫu tin vào bảng gốc không tác động đến con trỏ.

Dynamic cursor:

Con trỏ Dynamic không sử dụng đến bảng tạm trong temdp mà thao tác trực tiếp với bảng gốc. Khác với con trỏ Keyset, dynamic không yêu cầu tính duy nhất của dữ liệu. Mọi thao tác thêm, sửa, xóa đều có tác động ngay lập tức đến con trỏ Dyanamic.

12.1.4 Thay đổi dữ liệu tại vị trí Cursor

Có thể sử dụng Update hoặc Delete để sửa đổi hoặc xóa mẫu tin tại vị trí hiện hành của con trỏ với mệnh đề:

where current of Tên_Cursor

Lưu ý: Chỉ áp dụng với con trỏ dạng Keyset và Dynamic.

12.1.5 Duyệt Cursor

Dùng lệnh Fetch để duyệt tuần tự cursor qua các dòng dữ liệu theo cú pháp:

```
Fetch
[[Next | Prior | First | Last | Absolute n | Relative n]
    From] Tên_cursor
[Into Tên_biến [,…n]]
```

Các từ khóa NEXT, PRIOR, FIRST, LAST: dùng để đọc dữ liệu của dòng dữ liệu kế tiếp (next), dòng dữ liệu phía trước (prior), dòng dữ liệu đầu tiên (first), dòng dữ liệu cuối cùng (last) so với dòng dữ liệu hiện hành mà cursor đang chỉ đến. Sau khi đọc dữ liệu thành công, dòng dữ liệu hiện hành sẽ bị thay đổi chính là dòng vừa mới được đọc.

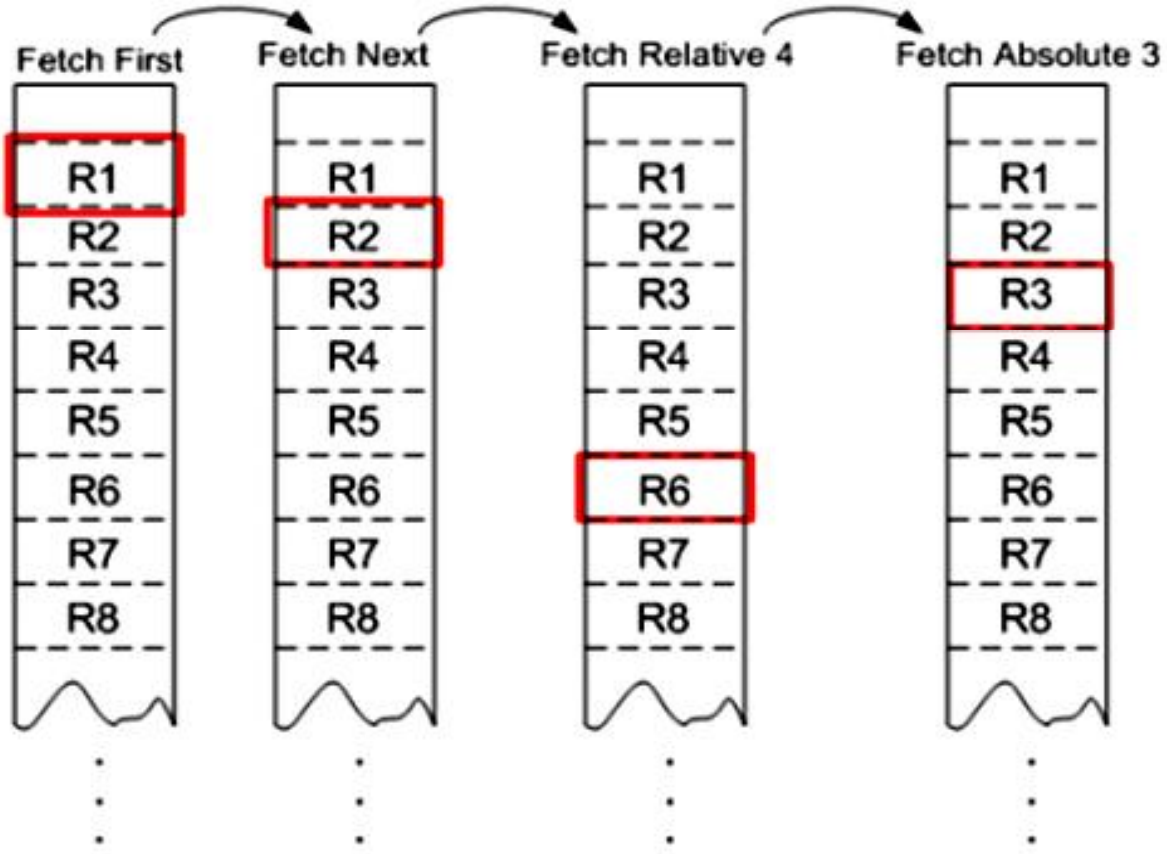
Từ khóa ABSOLUTE: dùng để chỉ định dữ liệu chính xác thứ n bên trong cursor. Với n là số nguyên dương dùng chỉ định việc đọc dữ liệu tại dòng thứ n được đếm từ dòng đầu tiên, với n là số nguyên âm dùng chỉ định việc đọc dữ liệu tại dòng thứ n được đếm ngược từ dòng cuối cùng trở lên.

Từ khóa RELATIVE: dùng để chỉ định việc đọc dữ liệu tại một dòng tương đối so với dòng dữ liệu hiện hành. Với n là một số nguyên có thể dương hoặc âm để chỉ định việc đọc theo chiều tới hoặc lui so với dòng dữ liệu hiện hành.

Mặc định: fetch next.

Đối với cursor dạng forward_only, chỉ có thể fetch next.

Biến hệ thống @@fetch_status cho biết lệnh fetch vừa thực hiện có thành công hay không, giá trị của biến này là cơ sở để biết đã duyệt đến cuối cursor hay chưa.



Hình 12.2: Minh họa việc đọc dữ liệu theo các thứ tự khác nhau

12.1.6 Quy trình sử dụng Cursor

- Khai báo cursor.
- "Mở" cursor bằng lệnh Open: Open Tên_cursor
- Khai báo các biến tạm để chứa phần tử hiện hành (đang được xử lý) của cursor:
- Các biến tạm phải có cùng kiểu dữ liệu với các field tương ứng của phần tử trong cursor.
- Có n field trong phần tử của cursor thì có đủ n biến tạm tương ứng
- Fetch (next,…) cursor để chuyển đến vị trí phù hợp:
    - Có thể đưa các giá trị của dòng hiện hành vào các biến thông qua mệnh đề into của lệnh fetch.

  - Có thể sử dụng vị trí hiện tại như là điều kiện cho mệnh đề where của câu delete/update (nếu cursor không là read_only).

- Lặp lại việc duyệt và sử dụng cursor, có thể sử dụng biến @@fetch_status để biết đã duyệt qua hết cursor hay chưa:
  - @@FETCH_STATUS = 0: lấy mẫu tin thành công.
  - @@FETCH_STATUS = -1: lấy mẫu tin thất bại với lý do là con trỏ đã ra quá vùng giới hạn BOF hoặc EOF.
  - @@FETCH_STATUS = -2: lấy mẫu tin thất bại với lý do là mẫu tin không tồn tại.
- Đóng cursor bằng lệnh Close: Close Tên_cursor

  Lưu ý: Sau khi đóng, vẫn có thể mở lại nếu cursor chưa bị hủy.
- Hủy cursor bằng lệnh deallocate: Deallocate Tên_cursor

  Ví dụ 1:

  Xét loại quan hệ: SANPHAM(MASP,TENSP)

```
Declare cur_SP cursor
For select MASP, TENSP from SANPHAM
Declare @Masp varchar(10), @Tensp nvarchar(30)
Open cur_SP
Fetch Next from cur_SP into @Masp, @Tensp
Print N'Mã sản phẩm: ' + @Masp +
     N' Tên sản phẩm: ' + @Tensp
Close cur_SP
Deallocate cur_SP
```

  Có nhận xét gì trong ví dụ trên?

  Cải tiến Ví dụ 1 ở trên:

  Xét loại quan hệ: SANPHAM(MASP,TENSP)

```
Declare cur_SP cursor
For select MASP, TENSP from SANPHAM
Declare @Masp varchar(10), @Tensp nvarchar(30)
Open cur_SP
Fetch Next from cur_SP into @Masp, @Tensp
While (@@fetch_status = 0)
begin
 Print N'Mã sản phẩm: ' + @Masp +
       N' Tên sản phẩm: ' + @Tensp
 Fetch Next from cur_SP into @Masp, @Tensp
end
Close cur_SP
Deallocate cur_SP
```

Ví dụ 2:

```
--1. Khai báo biến Cursor
Declare cur_Vattu CURSOR
KEYSET
For
 select * from VT where TENVT Like 'DA%'
 order by MAVT
--2. Mở cursor
Open cur_Vattu
Fetch next from cur_Vattu
--3. Đọc dữ liệu
While @@FETCH_STATUS=0
Begin
 --Đọc các dòng kế tiếp
 Fetch next from cur_Vattu
End
```

--4. Đóng cursor

```
Close cur_Vattu
Deallocate cur_Vattu
```

Ví dụ 3:

Xét CSDL gồm 2 bảng sau:

PHIEUNHAP(SOPN, TGNHAP),

CTPHIEUNHAP(SOPN, SLNHAP, DGNHAP).

Để cập nhật giá trị dữ liệu cho cột TGNHAP (trị giá nhập) trong bảng PHIEUNHAP bằng cách duyệt qua từng phiếu nhập, tính ra trị giá nhập của từng phiếu căn cứ vào số lượng nhập và đơn giá nhập của từng vật tư trong bảng CTPNHAP, sau cùng cập nhật vào cột TGNHAP.

--1. Khai báo biến cursor, các biến cục bộ

```
Declare cur_PNHAP CURSOR
Forward_only FOR
 select SOPN from PHIEUNHAP
Declare @SoPN char(4), @TongTG money
```

--2. Mở cursor

```
Open cur_PNHAP
```

--3. Đọc dữ liệu và cập nhật giá trị

```
While 0=0
Begin
  Fetch next from cur_PNHAP INTO @SoPN
  If @@FETCH_STATUS<>0 Break
 Select @TongTG=SUM(SLNHAP*DGNHAP) from CTPHIEUNHAP where
 SOPN=@SoPN
  Print N'Đang nhập số phiếu nhập: ' + @SoPN + '…'
  Update PHIEUNHAP
  Set TGNHAP = @TongTG
  Where CURRENT OF cur_PNHAP
End
```

--4. Đóng cursor

```
Close cur_PNHAP
Deallocate cur_PNHAP
```

Nhận xét:

Trong ví dụ 2 ở trên, sử dụng vòng lặp WHILE mà điều kiện lặp là 0 = 0 để chỉ định điều kiện so sánh vòng lặp là luôn luôn đúng. Do đó, bên trong vòng lặp này bạn bắt buộc phải thoát khỏi vòng lặp bằng lệnh BREAK và điều kiện thoát là khi việc đọc dữ liệu bị lỗi (@@FETCH_STATUS <> 0). Ngoài việc cập nhật dữ liệu bằng lệnh UPDATE mà trong mệnh đề WHERE có sử dụng từ khóa CURRENT OF <Tên cursor> dùng để chỉ định việc cập nhật dữ liệu trên dòng dữ liệu hiện hành của cursor.

12.2 MÃ HÓA DỮ LIỆU TRONG SQL SERVER

12.2.1 Khái niệm

Mã hóa là một phương pháp quan trọng nhằm bảo mật dữ liệu.

Những dữ liệu nhạy cảm như số CMND, số thẻ tín dụng, mật khẩu… cần phải được bảo vệ trước vô vàn mối nguy hiểm tấn công hiện nay. Phần lớn các CSDL quan hệ thường làm việc trên dữ liệu của nhiều dòng mẫu tin, còn gọi là một bộ các mẫu tin. Ví dụ lệnh SELECT kết quả luôn trả về nhiều dòng dữ liệu hơn là một dòng dữ liệu.

Trong hệ QTCSDL SQL Server các bạn có thể tự tạo các hàm của riêng mình hoặc sử dụng các DLL ngoài để mã hóa dữ liệu.

Trong phiên bản từ SQL Server 2005 trở về sau, các hàm và các phương thức mã hóa đã được mặc định có sẵn để sử dụng.

12.2.2 Các kỹ thuật mã hóa

Hệ QTCSDL SQL Server cung cấp các kỹ thuật mã hóa dữ liệu:

- Mã hóa bằng mật khẩu.
- Mã hóa khóa đối xứng.
- Mã hóa khóa không đối xứng.

- Mã hóa chứng nhận.

Trong bài học này, giới thiệu cách sử dụng kỹ thuật mã hóa bằng mật khẩu và phương pháp giải mã nó. SQL Server cung cấp 2 hàm cho việc mã hóa: một cho việc mã hóa và một cho việc giải mã. "Mã hóa bằng mật khẩu" là phương pháp mã hóa dữ liệu cơ bản thông qua mật khẩu. Dữ liệu có thể được giải mã nếu nhập đúng mật khẩu đã sử dụng khi mã hóa. Chúng ta sẽ thử một ví dụ về việc mã hóa và giải mã dữ liệu bằng kỹ thuật mã hóa thông qua mật khẩu.

Hàm mã hóa **EncryptByPassPhrase** để mã hóa dữ liệu, ta sử dụng hàm này như sau:

```
select MaHoaData = EncryptByPassPhrase(`B@234','dulieucầnmahoa')
```

Trong đó: `B@234' là mật khẩu còn 'dulieucầnmahoa' sẽ là thông tin mà chúng ta sẽ mã hóa cho người khác không đọc được. Như vậy, sau khi thực hiện lệnh select ở trên, dữ liệu hiển thị ra màn hình sẽ là chuối số đã được mã hóa. Với cùng một dữ liệu cần mã hóa, mỗi lần chạy sẽ cho ra một chuỗi số mã hóa khác nhau, điều này làm tăng khả năng bảo mật của dữ liệu cần mã hóa. Tuy nhiên khi sử dụng hàm giải mã **DecryptByPassPhrase** thì với bất kỳ chuỗi số đã mã hóa nào cũng sẽ cho kết quả như ban đầu.

```
Select GiaiMaDat = DecryptByPassPhrase(`B@234','chuoisodamahoa')
```

Sau khi thực hiện lệnh select có hàm giải mã và cung cấp đúng mật khẩu, dữ liệu sẽ trở về trang thái như ban đầu là `dulieucầnmahoa'. Tuy nhiên, nếu chúng ta cung cấp sai mật khẩu, dữ liệu trả về sẽ là NULL.

Bây giờ, chúng ta sẽ thử tạo một bảng chứa số thẻ tín dụng và số CMND, sau đó sẽ mã hóa dữ liệu này thông qua phương pháp mã hóa mật khẩu:

```
USE [master]
Go
IF EXISTS (SELECT name FROM sys.databases WHERE name = N'Customer DB')
DROP DATABASE [Customer DB]
Go
Create database [Customer DB]
```

```
Go
use [Customer DB]
Go
Create table [Customer data]
      ([customer id] int,
      [Credit Card Number] bigint,
      [Social Security Number] bigint)
Go
```

Chèn dữ liệu vào bảng:

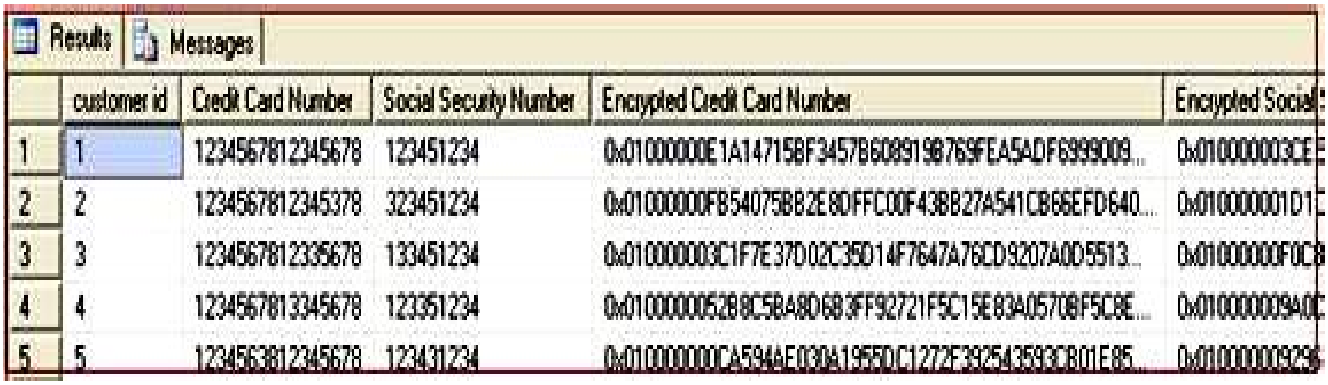
```
insert into [Customer data] values (1, 1234567812345678, 123451234)
insert into [Customer data] values (2, 1234567812345378, 323451234)
insert into [Customer data] values (3, 1234567812335678, 133451234)
insert into [Customer data] values (4, 1234567813345678, 123351234)
insert into [Customer data] values (5, 1234563812345678, 123431234)
Go
--Tạo hai cột để lưu dữ liệu đã được mã hóa:
use [Customer DB]
Go
Alter table [Customer Data]
  Add [Encrypted Credit Card Number] varbinary(MAX)
Go
Alter table [Customer Data
  Add [Encrypted Social Security Number] varbinary(MAX)
Go
--Cập nhật dữ liệu đã được mã hóa vào hai cột vừa tạo:
use [Customer DB]
Go
Update [Customer Data]
  set [Encrypted Credit Card Number] =
```

```
 EncryptByPassPhrase('Credit Card', convert(varchar(100),[Credit Card
 Number]))
Go
Update [Customer Data] set [Encrypted Social Security Number] =
EncryptByPassPhrase('Social Security', convert(varchar(100),[Social Security
Number]))
Go
```

Truy vấn bảng bằng các lệnh sau:

```
Use [Customer DB]
Go
Select * from [customer data]
Go
```

Kết quả:



Results | Messages

	customer id	Credit Card Number	Social Security Number	Encrypted Credit Card Number	Encrypted Social
1	1	1234567812345678	123451234	0x01000000E1A14715BF3457B60891987659FEA5ADF6999009...	0x010000003CE
2	2	1234567812345378	323451234	0x01000000FB54075BB2E8DFFC00F43BB27A541CB66EFD640...	0x010000001D1
3	3	1234567812335678	133451234	0x01000000C1F7E37D02C35D14F7647A76CD9207A0D5513...	0x010000000F0C8
4	4	1234567813345678	123351234	0x01000000528BC5BA8D683FF92721F5C15E83A0570BF5CBE...	0x010000009A0C
5	5	1234563812345678	123431234	0x01000000CA594AE030A1955DC1272F392543593CB01E85...	0x010000005296

Hình 12.3 Kết quả khi truy vấn 2 cột lưu dữ liệu mã hóa

--Xóa bỏ cột chứa dữ liệu chưa được mã hóa để các users khác không xem được dữ liệu gốc:

```
Use [Customer DB]
Go
Alter table [Customer Data]
 drop column [Credit Card Number]
Go
Alter table [Customer Data]
 drop column [Social Security Number]
Go
```

-- Giải mã dữ liệu trên bảng thông qua hàm Decryptbypassphrase như sau:

```
Use [Customer DB]
Go
Select [customer id],
convert(bigint,convert(varchar(100),
decryptbypassphrase('Credit Card',
[Encrypted Credit Card Number]) )) as [Credit Card Number],
convert(bigint,convert(varchar(100),
decryptbypassphrase('Social Security',
[Encrypted Social Security Number])))
as [Social Security Number] from [customer data]
Go
```

```
customer id,Credit Card Number,Social Security Number
1, 1234567812345678, 123451234
2, 1234567812345378, 323451234
3, 1234567812335678, 133451234
4, 1234567813345678, 123351234
5, 1234563812345678, 123431234
```

Results | Messages

	customer id	Credit Card Number	Social Security Number
1	1	1234567812345678	123451234
2	2	1234567812345378	323451234
3	3	1234567812335678	133451234
4	4	1234567813345678	123351234
5	5	1234563812345678	123431234

Hình 12.4: Kết quả sau khi giải mã dữ liệu thông qua hàm Decryptbypassphrase

Mặc dù mã hóa là một công cụ có giá trị để bảo đảm an ninh,an toàn cho dữ liệu nhưng nó không nên được xem xét cho tất cả các loại dữ liệu hoặc các kết nối khác

nhau. Khi ta quyết định có nên thực hiện mã hóa hay không, trước tiên hãy xem xét cách thức người dùng truy cập dữ liệu. Nếu người dùng truy cập dữ liệu qua một mạng công cộng, mã hóa dữ liệu có thể được yêu cầu nhằm tăng cường an ninh. Tuy nhiên, nếu tất cả các truy cập chỉ liên quan trong mạng nội bộ (đã đảm bảo an toàn), mã hóa có thể không được thực hiện để tăng tốc độ truy xuất dữ liệu.

12.3 BẢO MẬT VÀ QUẢN TRỊ

12.3.1 Khái niệm

Bảo mật và quản trị nhằm kiểm soát ai truy cập và truy cập dữ liệu gì. Người quản trị phải làm cho dễ dàng và thuận lợi để người dùng chỉ truy xuất được phần dữ liệu mà họ được phép, đồng thời phải ngăn chặn được những người không được phép truy xuất vào dữ liệu hoặc cố tình phá hoại dữ liệu.

Mỗi database nên có 1 hệ thống bảo mật đáng tin cậy (reliable security system) để giám sát mọi hoạt động cũng như các thông tin cần được xem và chỉnh sửa

Một hệ thống bảo mật đáng tin cậy phải bảo đảm được việc bảo vệ dữ liệu bất kể việc user đã dùng cách nào để truy xuất vào database. SQL áp dụng các quyền bảo mật vào các mức: mức CSDL, mức các đối tượng và mức các cột của bảng

12.3.2 LoginID và UserID

Người dùng muốn truy xuất vào Microsoft SQL Server, thì phải có login ID và password. Nhưng login ID chính nó không cho phép người dùng quyền truy xuất đến các cơ sở dữ liệu. Muốn tạo login ID ta dùng lệnh **sp_addlogin**

Cú pháp: sp_addlogin [**@loginame =**] '*login*'
[**,** [**@passwd =**] '*password*']
[**,** [**@defdb =**] '*database*']

Ví dụ:

```
EXEC sp_addlogin 'student1','Password','master'
```

Chỉ có người quản tri (administrator) mới có quyền tạo login ID mới.

User ID nhận dạng người dùng trong một CSDL. Tất cả các quyền và chủ quyền của các đối tượng trong CSDL đều được điều khiển bởi user ID. Ví dụ user ID là xyz trong CSDL QuanLyVatTu khác với user ID cũng tên là xyz trong CSDL QuanLyKho.

Để thêm mới một user ID ta dùng lệnh **sp_adduser**

sp_adduser [**@loginame =**] '*login*' [**,** [**@name_in_db =**] '*user*'] [**,** [**@grpname =**] '*group*']

Trong đó:

'login': xác định login id của user

'***user'*:** là tên của user mới. Nếu tuỳ chọn này không được xác định, tên của user sẽ chính là tên login id của user đó. Có thể tạo ra tài khoản user khác với tên login id của user đó.

'***group***': là nhóm hay role mà user mới này sẽ tự động trở thành thành viên của nhóm.

Lưu ý: chỉ có thể tạo user mới cho những user nào đã có tài khoản đăng nhập (login ID).

12.3.3 Nhóm quyền (Role)

Role là một công cụ cho phép ta tập hợp các user vào cùng 1 nhóm nhờ đó ta có thể gán quyền chung cho cả nhóm đó mà không phải cấp quyền cho từng user.

Ví dụ như, trong 1 công ty, ta có thể tập hợp các công nhân làm cùng 1 công việc lại thành 1 nhóm, nhờ đó ta có thể giao việc, cấp quyền cho cả nhóm. Mọi thành viên trong nhóm sẽ có quyền như nhau. Nếu chức năng của nhóm thay đổi, ta chỉ đơn giản thay đổi quyền của nhóm, khi đó những thay đổi này sẽ được tự động áp dụng cho tất cả các thành viên trong nhóm.

Để dễ quản lý một CSDL, ta nên xác định 1 tập hợp các role dựa theo yêu cầu công việc và gán cho mỗi role những quyền hạn (permission) khác nhau. Sau đó, chỉ cần chuyển các user vào các role thích hợp. Như vậy sẽ tốt hơn là phải cấp quyền cho mỗi user riêng lẻ. Nếu công việc thay đổi. chỉ cần thay đổi quyền trong mỗi role thì những thay đổi này sẽ tự động được áp dụng cho toàn bộ các thành viên của role đó.

Các user có thể là thành viên của nhiều role.

Để tạo nhóm quyền ta dùng lệnh **sp_addrole**

Cú pháp:

sp_addrole [**@rolename =**] '*role*'
[**,** [**@ownername =**] '*owner*']

Chỉ có những thành viên của role **sysadmin, db_securityadmin** và **db_owner** mới có quyền chạy lệnh này

Ví dụ: tạo 1 role mới tên **Managers** cho CSDL hiện hành.

```
EXEC sp_addrole 'Managers'
```

Để thêm thành viên vào nhóm ta dùng lệnh:

sp_addrolemember [**@rolename =**] '*role*' **,**
[**@membername =**] '*security_account*'

Để xóa một thành viên khỏi nhóm ta dùng lệnh:

sp_droprolemember [**@rolename =**] '*role*' **,**

[**@membername =**] '*security_account*'

12.3.4 Quyền trong SQL server

Khi 1 đối tượng được tạo ra, chỉ có owner (người tạo đối tượng) mới có quyền truy xuất đối tượng. Owner phải cấp quyền cho các user khác.

User phải có quyền thích hợp để thực thi bất kỳ hoạt động nào liên quan đến việc thay đổi định nghĩa CSDL hay truy xuất dữ liệu trong CSDL/

Có 3 loại quyền:

- Object Permissions: cho phép người dùng thực hiện các tác vụ trên từng đối tượng như: SELECT, INSERT, UPDATE, DELETE…
- Statement Permissions: cho phép người dùng tạo CDSL mới, tạo các đối tượng trong phạm vi CSDL như: CREATE DATABASE, CREATE TABLE, CREATE RULE, CREATE PROCEDURE…

- Implied Permissions: đây là những quyền người dùng có được do là thành viên của nhóm hoặc bởi vì người dùng là chủ đối tượng.

Tất cả các quyền trong SQL server có thể tồn tại dưới 1 trong 3 trạng thái sau: granted (được cấp quyền), revoked (bị thu hồi) và denied (bị từ chối).

12.3.5 Thay đổi quyền

Chúng ta có thể sử dụng các lệnh DCL(Data Control Language) để thay đổi các quyền có liên quan đến user hay role của CSDL.

Lệnh **GRANT** dùng để cấp quyền cho phép user, group hay role làm việc với lệnh về dữ liệu hay thực thi các lệnh T-SQL

Cú pháp:

Statement permissions:

```
GRANT { ALL | statement [ ,...n ] }
TO security_account [ ,...n ]
```

Object permissions:

```
GRANT
   { ALL | permission [ ,...n ] }
   {
      [ ( column [ ,...n ] ) ] ON { table | view }
      | ON { table | view } [ ( column [ ,...n ] ) ]
      | ON { stored_procedure | extended_procedure }
      | ON { user_defined_function }
   }
TO security_account [ ,...n ]
[ WITH GRANT OPTION ]
[ AS { group | role } ]
```

Trong đó:

ALL: đối với quyền về lệnh (statement permission), thì ALL chỉ có thể được dùng bởi các thành viên của **sysadmin**. Đối với quyền về đối tượng (object

permissions), ALL chỉ được dùng bởi các thành viên của **sysadmin, db_owner** và owner của đối tượng CSDL.

WITH GRANT OPTION (chỉ dùng cho quyền về đối tượng) cho quyền các tài khoản sau khi được gán quyền này được phép cấp quyền này cho các tài khoản khác

Ví dụ:

Use pubs

GRANT select, insert, update ON titles TO faculty

Lệnh **Deny** để loại bỏ quyền khỏi 1 tài khoản trong CSDL hiện hành và để tránh tài khoản này được thừa hưởng quyền của nhóm hay role của tài khoản đó.

Cú pháp:

DENY <permissions>[ON <object>]TO <user/role>

Ví dụ:

Use pubs

DENY select, insert, update ON titles TO faculty

Lệnh **REVOKE** dùng để thu hồi lại quyền đã được cấp hay từ chối từ 1 user của CSDL hiện hành

Cú pháp

REVOKE [GRANT OPTION FOR] <permissions>

[ON <object>]

TO <user/role>

Ví dụ:

REVOKE select, insert, update ON titles TO faculty

Lưu ý: quyền deny luôn ưu tiên hơn. Quyền deny ở bất kỳ mức nào cũng sẽ cấm các quyền trên các đối tượng bất kể có tồn tại quyền grant hay revoke dành cho user đó hay không. Ví dụ nếu John là thành viên của nhóm sales, nhóm này đã được cấp quyền SELECT trên bảng Customer, nếu John bị cấm (deny) 1 cách tường minh quyền

SELECT trên bảng Customer thì John sẽ không còn được truy xuất vào bảng đó nữa. Tương tự, nếu nhóm sales bị cấm truy xuất bảng customer thì dù cho John được cấp quyền truy xuất cũng sẽ bị cấm theo.

TÓM TẮT

Phân quyền là sự phân chia khả năng quản trị và sử dụng hệ quản trị CSDL SQL Server. Hình thành theo cơ cấu:

- *Người đăng nhập (login)*
- *Người dùng (user)*
- *Quyền hạn (permission)*
- *Nhóm quyền (role)*
- *Người đăng nhập được thể hiện là mỗi một người dùng với một số quyền hạn ứng với một dữ liệu.*

Mỗi CSDL nên có một hệ thống bảo mật đáng tin cậy (reliable security system) để giám sát mọi hoạt động cũng như các thông tin cần được xem và chỉnh sửa. Một hệ thống bảo mật đáng tin cậy phải bảo đảm được việc bảo vệ dữ liệu bất kể việc user đã dùng cách nào để truy xuất vào CSDL. SQL áp dụng các quyền bảo mật vào các mức: mức CSDL, mức các đối tượng và mức các cột của bảng.

User có thể truy xuất vào Databasethông qua 1 account đăng nhập (login ID) hợp lệ, nhờ đó user có khả năng kết nối vào database server. Quá trình này được gọi là authentication (xác thực).

Tài khoản đăng nhập (Login ID) sẽ được ánh xạ với tài khoản user (user ID) để cho phép user được quyền truy xuất trong một database. Quá trình này gọi là authorization (cấp phép) (hay permission validation). User sẽ không thể truy xuất vào database ngay cả khi họ có tài khoản đăng nhập hợp lệ.

CÂU HỎI ÔN TẬP

Câu 1: Tại sao cần thiết phải tạo người dùng và phân quyền cho người dùng?

Từ kiến thức đã học, học viên áp dụng làm bài thực hành số 4.

TÀI LIỆU THAM KHẢO

1. Jeffrey Ullman - dịch giả Trần Đức Quang(2002) Lý thuyết thiết kế cơ sở dữ liệu NXB Thống Kê, Hà Nội.
2. David Maier (1983). The Theory of Relational Database, Communications of the ACM, Volume 26, Number 2.
3. Đồng Thị Bích Thủy (2000), Nhập môn cơ sở dữ liệu, ĐH khoa học tự nhiên, ĐH Quốc Gia TP. Hồ Chí Minh.
4. Nguyễn Bá Tường (1996), Cơ sở dữ liệu – Lý thuyết và thực hành, nhà xuất bản Thống Kê, Hà Nội.
5. Giáo trình hệ QTCSDL, ĐH Khoa học Tự nhiên, ĐH Quốc Gia TP.HCM, 2010.
6. Tạ Thị Thu Phượng. *Hệ quản trị CSDL (Bài giảng tóm tắt)*. ĐH Đà Lạt, 2007.