Chương 1:

TỔNG QUAN VỀ SQL

Ngôn ngữ hỏi có cấu trúc (SQL) và các hệ quản trị cơ sở dữ liệu quan hệ là một trong những nền tảng kỹ thuật quan trọng trong công nghiệp máy tính. Cho đến nay, có thể nói rằng SQL đã được xem là ngôn ngữ chuẩn trong cơ sở dữ liệu. Các hệ quản trị cơ sở dữ liệu quan hệ thương mại hiện có như Oracle, SQL Server, Informix, DB2,... đều chọn SQL làm ngôn ngữ cho sản phẩm của mình. Vậy thực sự SQL là gì? Tại sao nó lại quan trọng trong các hệ quản trị cơ sở dữ liệu? SQL có thể làm được những gì và như thế nào? Nó được sử dụng ra sao trong các hệ quản trị cơ sở dữ liệu quan hệ? Nội dung của chương này sẽ cung cấp cho chúng ta cái nhìn tổng quan về SQL và một số vấn đề liên quan.

1.1 SQL là ngôn ngữ cơ sở dữ liệu quan hệ

SQL, viết tắt của *Structured Query Language* (ngôn ngữ hỏi có cấu trúc), là công cụ sử dụng để tổ chức, quản lý và truy xuất dữ liệu được lưu trữ trong các cơ sở dữ liệu. SQL là một hệ thống ngôn ngữ bao gồm tập các câu lệnh sử dụng để tương tác với cơ sở dữ liệu quan hệ. Tên gọi *ngôn ngữ hỏi có cấu trúc* phần nào làm chúng ta liên tưởng đến một công cụ (ngôn ngữ) dùng để truy xuất dữ liệu trong các cơ sở dữ liệu. Thực sự mà nói, khả năng của SQL vượt xa so với một công cụ truy xuất dữ liệu, mặc dù đây là mục đích ban đầu khi SQL được xây dựng nên và truy xuất dữ liệu vẫn còn là một trong những chức năng quan trọng của nó. SQL được sử dụng để điều khiển tất cả các chức năng mà một hệ quản trị cơ sở dữ liệu cung cấp cho người dùng bao gồm:

Định nghĩa dữ liệu: SQL cung cấp khả năng định nghĩa các cơ sở dữ liệu, các cấu trúc lưu trữ và tổ chức dữ liệu cũng như mối quan hệ giữa các thành phần dữ liệu.

Truy xuất và thao tác dữ liệu: Với SQL, người dùng có thể dễ dàng thực hiện các thao tác truy xuất, bổ sung, cập nhật và loại bỏ dữ liệu trong các cơ sở dữ liệu.

Điều khiển truy cập: SQL có thể được sử dụng để cấp phát và kiểm soát các thao tác của người sử dụng trên dữ liệu, đảm bảo sự an toàn cho cơ sở dữ liệu.

Đảm bảo toàn vẹn dữ liệu: SQL định nghĩa các ràng buộc toàn vẹn trong cơ sở dữ liệu nhờ đó đảm bảo tính hợp lệ và chính xác của dữ liệu trước các thao tác cập

nhật cũng như các lỗi của hệ thống.

Như vậy, có thể nói rằng SQL là một ngôn ngữ hoàn thiện được sử dụng trong các hệ thống cơ sở dữ liệu và là một thành phần không thể thiếu trong các hệ quản trị cơ sở dữ liệu. Mặc dù SQL không phải là một ngôn ngữ lập trình như C, C++, Java,... song các câu lệnh mà SQL cung cấp có thể được nhúng vào trong các ngôn ngữ lập trình nhằm xây dựng các ứng dụng tương tác với cơ sở dữ liệu. Khác với các ngôn ngữ lập trình quen thuộc như C, C++, Java,... SQL là ngôn ngữ có tính khai báo. Với SQL, người dùng chỉ cần mô tả các yêu cầu cần phải thực hiện trên cơ sở dữ liệu mà không cần phải chỉ ra cách thức thực hiện các yêu cầu như thế nào. Chính vì vậy, SQL là ngôn ngữ dễ tiếp cận và dễ sử dụng.

1.2 Vai trò của SQL

Bản thân SQL không phải là một hệ quản trị cơ sở dữ liệu, nó không thể tồn tại độc lập. SQL thực sự là một phần của hệ quản trị cơ sở dữ liệu, nó xuất hiện trong các hệ quản trị cơ sở dữ liệu với vai trò ngôn ngữ và là công cụ giao tiếp giữa người sử dụng và hệ quản trị cơ sở dữ liệu. Trong hầu hết các hệ quản trị cơ sở dữ liệu quan hệ, SQL có những vai trò như sau:

SQL là ngôn ngữ hỏi có tính tương tác: Người sử dụng có thể dễ dàng thông qua các trình tiện ích để gởi các yêu cầu dưới dạng các câu lệnh SQL đến cơ sở dữ liệu và nhận kết quả trả về từ cơ sở dữ liệu.

SQL là ngôn ngữ lập trình cơ sở dữ liệu: Các lập trình viên có thể nhúng các câu lệnh SQL vào trong các ngôn ngữ lập trình để xây dựng nên các chương trình ứng dụng giao tiếp với cơ sở dữ liệu.

SQL là ngôn ngữ quản trị cơ sở dữ liệu: Thông qua SQL, người quản trị cơ sở dữ liệu có thể quản lý được cơ sở dữ liệu, định nghĩa các cấu trúc lưu trữ dữ liệu, điều khiển truy cập cơ sở dữ liệu,...

SQL là ngôn ngữ cho các hệ thống khách/chủ (client/server): Trong các hệ thống cơ sở dữ liệu khách/chủ, SQL được sử dụng như là công cụ để giao tiếp giữa các trình ứng dụng phía máy khách với máy chủ cơ sở dữ liệu.

SQL là ngôn ngữ truy cập dữ liệu trên Internet: Cho đến nay, hầu hết các máy chủ Web cũng như các máy chủ trên Internet sử dụng SQL với vai trò là ngôn ngữ

để tương tác với dữ liệu trong các cơ sở dữ liệu.

SQL là ngôn ngữ cơ sở dữ liệu phân tán: Đối với các hệ quản trị cơ sở dữ liệu phân tán, mỗi một hệ thống sử dụng SQL để giao tiếp với các hệ thống khác trên mạng, gửi và nhận các yêu cầu truy xuất dữ liệu với nhau.

SQL là ngôn ngữ sử dụng cho các cổng giao tiếp cơ sở dữ liệu: Trong một hệ thống mạng máy tính với nhiều hệ quản trị cơ sở dữ liệu khác nhau, SQL thường được sử dụng như là một chuẩn ngôn ngữ để giao tiếp giữa các hệ quản trị cơ sở dữ liệu.

1.3 Tổng quan về cơ sở dữ liệu quan hệ

1.3.1 Mô hình dữ liệu quan hệ

Mô hình dữ liệu quan hệ được Codd đề xuất năm 1970 và đến nay trở thành mô hình được sử dụng phổ biến trong các hệ quản trị cơ sở dữ liệu thương mại. Nói một cách đơn giản, một cơ sở dữ liệu quan hệ là một cơ sở dữ liệu trong đó tất cả dữ liệu được tổ chức trong các bảng có mối quan hệ với nhau. Mỗi một bảng bao gồm các dòng và các cột: mỗi một dòng được gọi là một bản ghi (bộ) và mỗi một cột là một trường (thuộc tính).

1.3.2 Bảng (Table)

Như đã nói ở trên, trong cơ sở dữ liệu quan hệ, bảng là đối tượng được sử dụng để tổ chức và lưu trữ dữ liệu. Một cơ sở dữ liệu bao gồm nhiều bảng và mỗi bảng được xác định duy nhất bởi tên bảng. Một bảng bao gồm một tập các dòng và các cột: mỗi một dòng trong bảng biểu diễn cho một thực thể.

Tên của bảng: được sử dụng để xác định duy nhất mỗi bảng trong cơ sở dữ liệu.

Cấu trúc của bảng: Tập các cột trong bảng. Mỗi một cột trong bảng được xác định bởi một *tên cột* và phải có một kiểu dữ liệu nào đó. Kiểu dữ liệu của mỗi cột qui định giá trị dữ liệu có thể được chấp nhận trên cột đó.

Dữ liệu của bảng: Tập các dòng (bản ghi) hiện có trong bảng.

1.3.3 Khoá của bảng

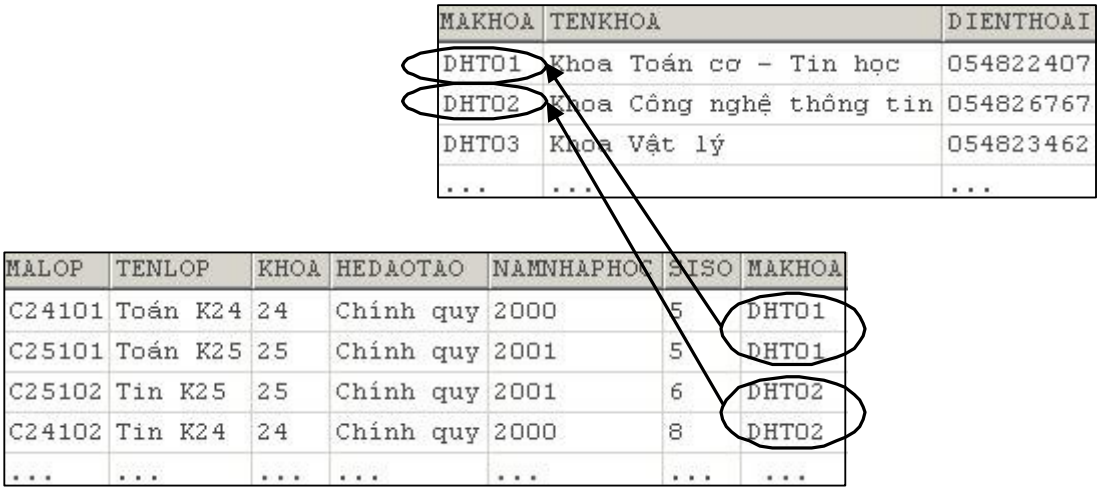
Trong một cơ sở dữ liệu được thiết kế tốt, mỗi một bảng phải có một hoặc một tập các cột mà giá trị dữ liệu của nó xác định duy nhất một dòng trong một tập các dòng của bảng. Tập một hoặc nhiều cột có tính chất này được gọi là khoá của

bảng. Việc chọn khoá của bảng có vai trò quan trọng trong việc thiết kế và cài đặt các cơ sở dữ liệu quan hệ. Các dòng dữ liệu trong một bảng phải có giá trị khác nhau trên khoá.

Một bảng có thể có nhiều tập các cột khác nhau có tính chất của khoá (tức là giá trị của nó xác định duy nhất một dòng dữ liệu trong bảng). Trong trường hợp này, khoá được chọn cho bảng được gọi là *khoá chính* (*primary key*) và những khoá còn lại được gọi là khoá phụ hay là *khoá dự tuyển* (*candidate key*/*unique key*).

1.3.4 Mối quan hệ và khoá ngoài

Các bảng trong một cơ sở dữ liệu không tồn tại độc lập mà có mối quan hệ mật thiết với nhau về mặt dữ liệu. Mối quan hệ này được thể hiện thông qua ràng buộc *giá trị dữ liệu xuất hiện ở bảng này phải có xuất hiện trước trong một bảng khác*. Mối quan hệ giữa các bảng trong cơ sở dữ liệu nhằm đảm bảo được tính đúng đắn và hợp lệ của dữ liệu trong cơ sở dữ liệu. Trong hình 1.1, hai bảng LOP và KHOA có mối quan hệ với nhau. Mối quan hệ này đòi hỏi giá trị cột MAKHOA của một dòng (tức là một lớp) trong bảng LOP phải được xác định từ cột MAKHOA của bảng KHOA.



Hình 1.1: *Mối quan hệ giữa hai bảng LOP và KHOA trong cơ sở dữ liệu*

Mối quan hệ giữa các bảng trong một cơ sở dữ liệu thể hiện đúng mối quan hệ giữa các thực thể trong thế giới thực. Trong hình 1.3, mối quan hệ giữa hai bảng LOP và KHOA không cho phép một lớp nào đó tồn tại mà lại thuộc vào một khoa

không có thật. Khái niệm *khoá ngoài* (*Foreign Key*) trong cơ sở dữ liệu quan hệ được sử dụng để biểu diễn mối quan hệ giữa các bảng dữ liệu. Một hay một tập các cột trong một bảng mà giá trị của nó được xác định từ khóa chính của một bảng khác được gọi là khoá ngoài. Trong hình 1.1, cột MAKHOA của bảng LOP được gọi là khoá ngoài của

bảng này, khoá ngoài này tham chiếu đến khoá chính của bảng KHOA là cột MAKHOA.

1.4 Sơ lược về SQL

1.4.1 Câu lệnh SQL

SQL chuẩn bao gồm khoảng 40 câu lệnh. Bảng 1.1 liệt kê danh sách các câu lệnh thường được sử dụng nhất trong số các câu lệnh của SQL. Trong các hệ quản trị cơ sở dữ liệu khác nhau, mặc dù các câu lệnh đều có cùng dạng và cùng mục đích sử dụng song mỗi một hệ quản trị cơ sở dữ liệu có thể có một số thay đổi nào đó. Điều này đôi khi dẫn đến cú pháp chi tiết của các câu lệnh có thể sẽ khác nhau trong các hệ quản trị cơ cơ sở dữ liệu khác nhau.

Câu lệnh	Chức năng
Thao tác dữ liệu	
SELECT	Truy xuất dữ liệu
INSERT	Bổ sung dữ liệu
UPDATE	Cập nhật dữ liệu
DELETE	Xóa dữ liệu
TRUNCATE	Xoá toàn bộ dữ liệu trong bảng
Định nghĩa dữ liệu	
CREATE TABLE	Tạo bảng
DROP TABLE	Xóa bảng
ALTER TABLE	Sửa đổi bảng
CREATE VIEW	Tạo khung nhìn
ALTER VIEW	Sửa đổi khung nhìn
DROP VIEW	Xoá khung nhìn
CREATE INDEX	Tạo chỉ mục
DROP INDEX	Xoá chỉ mục
CREATE SCHEMA	Tạo lược đồ cơ sở dữ liệu
DROP SCHEMA	Xoá lược đồ cơ sở dữ liệu
CREATE PROCEDURE	Tạo thủ tục lưu trữ
ALTER PROCEDURE	Sửa đổi thủ tục lưu trữ
DROP PROCEDURE	Xoá thủ tục lưu trữ
CREATE FUNCTION	Tạo hàm (do người sử dụng định nghĩa)

ALTER FUNCTION	Sửa đổi hàm
DROP FUNCTION	Xoá hàm
CREATE TRIGGER	Tạo trigger
ALTER TRIGGER	Sửa đổi trigger
DROP TRIGGER	Xoá trigger
Điều khiển truy cập	
GRANT	Cấp phát quyền cho người sử dụng
REVOKE	Thu hồi quyền từ người sử dụng
Quản lý giao tác	
COMMIT	Uỷ thác (kết thúc thành công) giao tác
ROLLBACK	Quay lui giao tác
SAVE TRANSACTION	Đánh dấu một điểm trong giao tác
Lập trình	
DECLARE	Khai báo biến hoặc định nghĩa con trỏ
OPEN	Mở một con trỏ để truy xuất kết quả truy vấn
FETCH	Đọc một dòng trong kết quả truy vấn (sử dụng con trỏ)
CLOSE	Đóng một con trỏ
EXECUTE	Thực thi một câu lệnh SQL

Các câu lệnh của SQL đều được bắt đầu bởi các từ lệnh, là một từ khoá cho biết chức năng của câu lệnh (chẳng hạn SELECT, DELETE, COMMIT). Sau từ lệnh là các mệnh đề của câu lệnh. Mỗi một mệnh đề trong câu lệnh cũng được bắt đầu bởi một từ

khoá (chẳng hạn FROM, WHERE,...).

Ví dụ 1.1: Câu lệnh: dùng để truy xuất dữ liệu trong bảng SINHVIEN được bắt đầu bởi từ lệnh SELECT, trong câu lệnh bao gồm hai mệnh đề: mệnh đề FROM chỉ định tên của bảng cần truy xuất dữ liệu và mệnh đề WHERE chỉ định điều kiện truy vấn dữ liệu.

SELECT **masv,hodem,ten**

FROM **sinhvien**

WHERE *malop='C24102'*

1.4.2 Qui tắc sử dụng tên trong SQL.

Các đối tượng trong cơ sở dữ liệu dựa trên SQL được xác định thông qua tên của đối tượng. Tên của các đối tượng là duy nhất trong mỗi cơ sở dữ liệu. Tên được sử dụng nhiều nhất trong các truy vấn SQL và được xem là nền tảng trong cơ

sở dữ liệu quan hệ là tên bảng và tên cột. Trong các cơ sở dữ liệu lớn với nhiều người sử dụng, khi ta chỉ định tên của một bảng nào đó trong câu lệnh SQL, hệ quản trị cơ sở dữ liệu hiểu đó là tên của bảng do ta sở hữu (tức là bảng do ta tạo ra). Thông thường, trong các hệ quản trị cơ sở dữ liệu này cho phép những người dùng khác nhau tạo ra những bảng trùng tên với nhau mà không gây ra xung đột về tên. Nếu trong một câu lệnh SQL ta cần chỉ đến một bảng do một người dùng khác sở hữu (hiển nhiên là phải được phép) thì tên của bảng phải được viết sau tên của người sở hữu và phân cách với tên người sở hữu bởi dấu chấm: tên_người_sở_hữu.tên_bảng

Một số đối tượng cơ sở dữ liệu khác (như khung nhìn, thủ tục, hàm), việc sử dụng tên cũng tương tự như đối với bảng.

Ta có thể sử dụng tên cột một cách bình thường trong các câu lệnh SQL bằng cách chỉ cần chỉ định tên của cột trong bảng. Tuy nhiên, nếu trong câu lệnh có liên quan đến hai cột trở lên có cùng tên trong các bảng khác nhau thì bắt buộc phải chỉ định thêm tên bảng trước tên cột; tên bảng và tên cột được phân cách nhau bởi dấu chấm.

Ví dụ: Ví dụ dưới đây minh hoạ cho ta thấy việc sử dụng tên bảng và tên cột trong câu

lệnh SQL

```
SELECT  masv,hodem,ten,sinhvien.malop,tenlop
FROM dbo.sinhvien,dbo.lop
WHERE sinhvien.malop = lop.malop
```

1.4.3 Kiểu dữ liệu

Chuẩn ANSI/ISO SQL cung cấp các kiểu dữ liệu khác nhau để sử dụng trong các cơ sở dữ liệu dựa trên SQL và trong ngôn ngữ SQL. Dựa trên cơ sở các kiểu dữ liệu do chuẩn ANSI/ISO SQL cung cấp, các hệ quản trị cơ sở dữ liệu thương mại hiện nay có thể sử dụng các dạng dữ liệu khác nhau trong sản phẩm của mình.

Bảng dưới đây liệt kê một số kiểu dữ liệu thông dụng được sử dụng trong SQL.

Tên kiểu	Mô tả
CHAR (*n*)	Kiểu chuỗi với độ dài cố định
NCHAR (*n*)	Kiếu chuỗi với độ dài cố định hỗ trợ UNICODE
VARCHAR (*n*)	Kiểu chuỗi với độ dài chính xác
NVARCHAR (*n*)	Kiểu chuỗi với độ dài chính xác hỗ trợ UNICODE
INTEGER	Số nguyên có giá trị từ -2^{31} đến $2^{31} - 1$
INT	Như kiểu Integer
TINYTINT	Số nguyên có giá trị từ 0 đến 255.
SMALLINT	Số nguyên có giá trị từ -2^{15} đến $2^{15} – 1$
BIGINT	Số nguyên có giá trị từ -2^{63} đến $2^{63}-1$
NUMERIC (*p,s*)	Kiểu số với độ chính xác cố định.
DECIMAL (*p,s*)	Tương tự kiểu Numeric
FLOAT	Số thực có giá trị từ -1.79E+308 đến 1.79E+308
REAL	Số thực có giá trị từ -3.40E + 38 đến 3.40E + 38
MONEY	Kiểu tiền tệ
BIT	Kiểu bit (có giá trị 0 hoặc 1)
DATETIME	Kiểu ngày giờ (chính xác đến phần trăm của giây)
SMALLDATETIME	Kiểu ngày giờ (chính xác đến phút)
BINARY	Dữ liệu nhị phân với độ dài cố định (tối đa 8000 bytes)
VARBINARY	Dữ liệu nhị phân với độ dài chính xác (tối đa 8000 bytes)
IMAGE	Dữ liệu nhị phân với độ dài chính xác (tối đa 2,147,483,647 bytes)
TEXT	Dữ liệu kiếu chuỗi với độ dài lớn (tối đa 2,147,483,647 ký tự)
NTEXT	Dữ liệu kiếu chuỗi với độ dài lớn và hỗ trợ UNICODE (tối đa 1,073,741,823 ký tự)

Ví dụ 1.2: Câu lệnh dưới đây định nghĩa bảng với kiểu dữ liệu được qui định cho các
cột trong bảng

```
CREATE TABLE  NHANVIEN
(MANV NVARCHAR(10)NOT NULL,
HOTEN NVARCHAR(30)NOT NULL,
GIOITINH BIT,
NGAYSINH SMALLDATETIME,
NOISINH NCHAR(50),
HSLUONG DECIMAL(4,2),
MADV INT)
```

1.4.4 Giá trị NULL

Một cơ sở dữ liệu là sự phản ánh của một hệ thống trong thế giới thực, do đó các giá trị dữ liệu tồn tại trong cơ sở dữ liệu có thể không xác định được. Một giá trị không xác định được xuất hiện trong cơ sở dữ liệu có thể do một số nguyên nhân sau:

Giá trị đó có tồn tại nhưng không biết.

Không xác định được giá trị đó có tồn tại hay không.

Tại một thời điểm nào đó giá trị chưa có nhưng rồi có thể sẽ có.

Giá trị bị lỗi do tính toán (tràn số, chia cho không,...)

Những giá trị không xác định được biểu diễn trong cơ sở dữ liệu quan hệ bởi các giá trị NULL. Đây là giá trị đặc biệt và không nên nhầm lẫn với chuỗi rỗng (đối với dữ liệu kiểu chuỗi) hay giá trị không (đối với giá trị kiểu số). Giá trị NULL đóng một vai trò quan trọng trong các cơ sở dữ liệu và hầu hết các hệ quản trị cơ sở dữ liệu quan hệ hiện nay đều hỗ trợ việc sử dụng giá trị này.

1.5 Kết chương

Như vậy, SQL (viết tắt của Structured Query Language) là hệ thống ngôn ngữ được sử dụng cho các hệ quản trị cơ sở dữ liệu quan hệ. Thông qua SQL có thể thực hiện được các thao tác trên cơ sở dữ liệu như định nghĩa dữ liệu, thao tác dữ liệu, điều khiển truy cập, quản lý toàn vẹn dữ liệu... SQL là một thành phần quan trọng và không
thể thiếu trong hệ quản trị cơ sở dữ liệu quan hệ.

SQL ra đời nhằm sử dụng cho các cơ sở dữ liệu theo mô hình quan hệ. Trong một cơ sở dữ liệu quan hệ, dữ liệu được tổ chức và lưu trữ trong các bảng. Mỗi một bảng là một tập hợp bao gồm các dòng và các cột; mỗi một dòng là một bản ghi và mỗi một cột tương ứng với một trường, tập các tên cột cùng với kiểu dữ liệu và các tính chất khác tạo nên cấu trúc của bảng, tập các dòng trong bảng chính là dữ liệu của bảng.

Các bảng trong một cơ sở dữ liệu có mối quan hệ với nhau. Các mối quan hệ được biểu diễn thông qua khoá chính và khoá ngoài của các bảng. Khoá chính của

bảng là tập một hoặc nhiều cột có giá trị duy nhất trong bảng và do đó giá trị của nó xác định duy nhất một dòng dữ liệu trong bảng. Một khoá ngoài là một tập một hoặc nhiều cột có giá trị được xác định từ khoá chính của các bảng khác.

Chương 2

NGÔN NGỮ THAO TÁC DỮ LIỆU

Đối với đa số người sử dụng, SQL được xem như là công cụ hữu hiệu để thực hiện các yêu cầu truy vấn và thao tác trên dữ liệu. Trong chương này, ta sẽ bàn luận đến nhóm các câu lệnh trong SQL được sử dụng cho mục đích này. Nhóm các câu lệnh này được gọi chung là ngôn ngữ thao tác dữ liệu (DML: Data Manipulation Language)bao gồm các câu lệnh sau:

SELECT: Sử dụng để truy xuất dữ liệu từ môt hoặc nhiều bảng.

INSERT: Bổ sung dữ liệu.

UPDATE: Cập nhật dữ liệu

DELETE: Xoá dữ liệu

Trong số các câu lệnh này, có thể nói SELECT là câu lệnh tương đối phức tạp và được sử dụng nhiều trong cơ sở dữ liệu. Với câu lệnh này, ta không chỉ thực hiện các yêu cầu truy xuất dữ liệu đơn thuần mà còn có thể thực hiện được các yêu cầu thống kê dữ liệu phức tạp. Cũng chính vì vậy, phần đầu của chương này sẽ tập trung tương đối nhiều đến câu lệnh SELECT. Các câu lệnh INSERT, UPDATE và DELETE được bàn luận đến ở cuối chương.

2.1 Truy xuất dữ liệu với câu lệnh SELECT

Câu lệnh SELECT được sử dụng để truy xuất dữ liệu từ các dòng và các cột của một hay nhiều bảng, khung nhìn. Câu lệnh này có thể dùng để thực hiện phép chọn (tức là truy xuất một tập con các dòng trong một hay nhiều bảng), phép chiếu (tức là truy xuất một tập con các cột trong một hay nhiều bảng) và phép nối (tức là liên kết các dòng trong hai hay nhiều bảng để truy xuất dữ liệu). Ngoài ra, câu lệnh này còn cung cấp khả năng thực hiện các thao tác truy vấn và thống kê dữ liệu phức tạp khác.

Cú pháp chung của câu lệnh SELECT có dạng:

SELECT [ALL | DISTINCT][TOP n] danh_sách_chọn

[INTO tên_bảng_mới]

FROM danh_sách_bảng/khung_nhìn

[WHEREđiều_kiện]

[GROUP BY danh_sách_cột]

[HAVING điều_kiện]

[ORDER BY cột_sắp_xếp]

[COMPUTE danh_sách_hàm_gộp [BY danh_sách_cột]]

Điều cần lưu ý đầu tiên đối với câu lệnh này là các thành phần trong câu lệnh SELECT nếu được sử dụng phải tuân theo đúng thứ tự như trong cú pháp. Nếu không, câu lệnh sẽ được xem là không hợp lệ.

Câu lệnh SELECT được sử dụng để tác động lên các bảng dữ liệu và kết quả của câu lệnh cũng được hiển thị dưới dạng bảng, tức là một tập hợp các dòng và các cột (ngoại trừ trường hợp sử dụng câu lệnh SELECT với mệnh đề COMPUTE).

Ví dụ 2.1: Kết quả của câu lệnh sau đây cho biết mã lớp, tên lớp và hệ đào tạo của các lớp hiện có

```
SELECT malop,tenlop,hedaotao
FROM lop
```

MALOP	TENLOP	HEDAOTAO
C24101	Toán K24	Chính quy
C24102	Tin K24	Chính quy
C24103	Lý K24	Chính quy
C24301	Sinh K24	Chính quy
C25101	Toán K25	Chính quy
C25102	Tin K25	Chính quy
C25103	Lý K25	Chính quy
C25301	Sinh K25	Chính quy
C26101	Toán K26	Chính quy
C26102	Tin K26	Chính quy

2.1.1 Mệnh đề FROM

Mệnh đề FROM trong câu lệnh SELECT được sử dung nhằm chỉ định các bảng và khung nhìn cần truy xuất dữ liệu. Sau FROM là danh sách tên của các bảng và khung nhìn tham gia vào truy vấn, tên của các bảng và khung nhìn được phân cách nhau bởi dấu phẩy.

Ví dụ 2.2: Câu lệnh dưới đây hiển thị danh sách các khoa trong trường

SELECT * FROM khoa

kết quả câu lệnh như sau:

Ta có thể sử dụng các bí danh cho các bảng hay khung nhìn trong câu lệnh SELECT. Bí danh được gán trong mệnh đề FROM bằng cách chỉ định bí danh ngay sau tên bảng.

Ví dụ 2.3: câu lệnh sau gán bí danh là a cho bảng khoa

SELECT * FROM khoa a

2.1.2 Danh sách chọn trong câu lệnh SELECT

Danh sách chọn trong câu lệnh SELECT được sử dụng để chỉ định các trường, các biểu thức cần hiển thị trong các cột của kết quả truy vấn. Các trường, các biểu thức được chỉ định ngay sau từ khoá SELECT và phân cách nhau bởi dấu phẩy. Sử dụng danh sách chọn trong câu lệnh SELECT bao gồm các trường hợp sau:

a. Chọn tất cả các cột trong bảng

Khi cần hiển thị tất cả các trường trong các bảng, sử dụng ký tự * trong danh sách chọn thay vì phải liệt kê danh sách tất cả các cột. Trong trường hợp này, các cột được hiển thị trong kết quả truy vấn sẽ tuân theo thứ tự mà chúng đã được tạo ra khi bảng được định nghĩa.

Ví dụ 2.4: Câu lệnh

SELECT * FROM lop

cho kết quả bao như sau:

MALOP	TENLOP	KHOA	HEDAOTAO	NAMNHAPHOC	MAKHOA
C24101	Toán K24	24	Chính quy	2000	DHTO1
C24102	Tin K24	24	Chính quy	2000	DHTO2
C24103	Lý K24	24	Chính quy	2000	DHTO3
C24301	Sinh K24	24	Chính quy	2000	DHTO5
C25101	Toán K25	25	Chính quy	2001	DHTO1
C25102	Tin K25	25	Chính quy	2001	DHTO2
C25103	Lý K25	25	Chính quy	2001	DHTO3
C25301	Sinh K25	25	Chính quy	2001	DHTO5
C26101	Toán K26	26	Chính quy	2002	DHTO1
C26102	Tin K26	26	Chính quy	2002	DHTO2

	DIENTHOAI
học	054822407
g tin	054826767
	054823462
	054823951
	054822934
hất	054823837
	054821133
	054823833
	054825698
	054821135

b. Tên cột trong danh sách chọn

Trong trường hợp cần chỉ định cụ thể các cột cần hiển thị trong kết quả truy vấn, ta chỉ định danh sách các tên cột trong danh sách chọn. Thứ tự của các cột trong kết quả truy vấn tuân theo thứ tự của các trường trong danh sách chọn.

Ví dụ 2.5: Câu lệnh

```
SELECT malop,tenlop,namnhaphoc,khoa
FROM lop
```

cho biết mã lớp, tên lớp, năm nhập học và khoá của các lớp và có kết quả như sau:

MALOP	TENLOP	NAMNHAPHOC	KHOA
C24101	Toán K24	2000	24
C24102	Tin K24	2000	24
C24103	Lý K24	2000	24
C24301	Sinh K24	2000	24
C25101	Toán K25	2001	25
C25102	Tin K25	2001	25
C25103	Lý K25	2001	25
C25301	Sinh K25	2001	25
C26101	Toán K26	2002	26
C26102	Tin K26	2002	26

Lưu ý: Nếu truy vấn được thực hiện trên nhiều bảng/khung nhìn và trong các bảng/khung nhìn có các trường trùng tên thì tên của những trường này nếu xuất hiện trong danh sách chọn phải được viết dưới dạng: tên_bảng.tên_trường

Ví dụ 2.6:

```
SELECT malop, tenlop, lop.makhoa, tenkhoa
FROM lop, khoa
WHERE lop.malop = khoa.makhoa
```

c. Thay đổi tiêu đề các cột

Trong kết quả truy vấn, tiêu đề của các cột mặc định sẽ là tên của các trường

tương ứng trong bảng. Tuy nhiên, để các tiêu đề trở nên thân thiện hơn, ta có thể đổi tên các tiêu đề của các cột. Để đặt tiêu đề cho một cột nào đó, ta sử dụng cách viết:

tiêu_đề_cột = tên_trường

hoặc tên_trường AS tiêu_đề_cột

hoặc tên_trường tiêu_đề_cột

Ví dụ 2.7: Câu lệnh dưới đây:

```
SELECT 'Mã lớp'= malop,tenlop 'Tên lớp',khoa AS 'Khoá'
FROM lop
```

cho biết mã lớp, tên lớp và khoá học của các lớp trong trường. Kết quả của câu lệnh

như sau:

Mã lớp	Tên Lớp	Khoá
C24101	Toán K24	24
C24102	Tin K24	24
C24103	Lý K24	24
C24301	Sinh K24	24
C25101	Toán K25	25
C25102	Tin K25	25
C25103	Lý K25	25
C25301	Sinh K25	25
C26101	Toán K26	26
C26102	Tin K26	26

d. Sử dụng cấu trúc CASE trong danh sách chọn

Cấu trúc CASE được sử dụng trong danh sách chọn nhằm thay đổi kết quả của truy vấn tuỳ thuộc vào các trường hợp khác nhau. Cấu trúc này có cú pháp như sau:

```
CASE biểu_thức
  WHEN biểu_thức_kiểm_tra THEN kết_quả
  [ ... ]
  [ELSE kết_quả_của_else]
END
```

hoặc:

```
CASE
```

WHEN điều_kiện THEN kết_quả

[...]

[ELSE kết_quả_của_else]

END

Ví dụ 2.8: Để hiển thị mã, họ tên và giới tính (nam hoặc nữ) của các sinh viên, ta sử

dụng câu lệnh

```
SELECT masv,hodem,ten,
  CASE gioitinh
WHEN 1 THEN 'Nam'
ELSE 'Nữ'
  END AS gioitinh
FROM sinhvien
```

hoặc:

```
SELECT masv,hodem,ten,
  CASE
          WHEN gioitinh=1 THEN 'Nam'
ELSE 'Nữ'
  END AS gioitinh
FROM sinhvien
```

MASV	HODEM	TEN	GIOITINH
0241010001	Ngô Thị Nhật	Anh	Nữ
0241010002	Nguyễn Thị Ngọc	Anh	Nữ
0241010003	Ngô Việt	Bắc	Nam
0241010004	Nguyễn Đình	Bình	Nam
0241010005	Hồ Đăng	Chiến	Nam
0241020001	Nguyễn Tuấn	Anh	Nam
0241020002	Trần Thị Kim	Anh	Nữ
0241020003	Võ Đức	Ân	Nam
0241020004	Nguyễn Công	Bình	Nam
0241020005	Nguyễn Thanh	Bình	Nam
0241020006	Lê Thị Thanh	Châu	Nữ
0241020007	Bùi Đình	Chiến	Nam
0241020008	Nguyễn Công	Chính	Nam
...	...	...	...

e. Hằng và biểu thức trong danh sách chọn

Ngoài danh sách trường, trong danh sách chọn của câu lệnh SELECT còn có thể sử dụng các biểu thức. Mỗi một biểu thức trong danh sách chọn trở thành một cột trong kết quả truy vấn.

Ví dụ 2.9: câu lệnh dưới đây cho biết tên và số tiết của các môn học

```
SELECT tenmonhoc,sodvht*15 AS sotiet
FROM monhoc
```

TENMONHOC	SOTIET
Hoá đại cương	45
Tin học đại cương	60
Ngôn ngữ C	75
Lý thuyết hệ điều hành	60
Cấu trúc dữ liệu và ...	60
Đại số tuyến tính	60
Giải tích 1	60
Bài tập Đại số	30
Bài tập Giải tích 1	30
Vật lý đại cương	45

Nếu trong danh sách chọn có sự xuất hiện của giá trị hằng thì giá trị này sẽ

TENMONHOC	(No column name)	SOTIET
Hoá đại cương	Số tiết:	45
Tin học đại cương	Số tiết:	60
Ngôn ngữ C	Số tiết:	75
Lý thuyết hệ điều hành	Số tiết:	60
Cấu trúc dữ liệu và giải thuật	Số tiết:	60
Đại số tuyến tính	Số tiết:	60
Giải tích 1	Số tiết:	60
Bài tập Đại số	Số tiết:	30
Bài tập Giải tích 1	Số tiết:	30
Vật lý đại cương	Số tiết:	45

f. Loại bỏ các dòng dữ liệu trùng nhau trong kết quả truy vấn

Trong kết quả của truy vấn có thể xuất hiện các dòng dữ liệu trùng nhau. Để loại bỏ bớt các dòng này, ta chỉ định thêm từ khóa DISTINCT ngay sau từ khoá SELECT.

Ví dụ 2.11: Hai câu lệnh dưới đây

SELECT khoa FROM lop

và:

SELECT DISTINCT khoa FROM lop

có kết quả lần lượt như sau:

KHOA
24
24
24
24
25
25
25
25
26
26

KHOA
24
25
26

g. Giới hạn số lượng dòng trong kết quả truy vấn

Kết quả của truy vấn được hiển thị thường sẽ là tất cả các dòng dữ liệu truy

vấn được. Trong trường hợp cần hạn chế số lượng các dòng xuất hiện trong kết quả truy vấn, ta chỉ định thêm mệnh đề TOP ngay trước danh sách chọn của câu lệnh SELECT.

Ví dụ 2.12: Câu lệnh dưới đây hiển thị họ tên và ngày sinh của 5 sinh viên đầu tiên trong danh sách

```
SELECT TOP 5 hodem,ten,ngaysinh
FROM sinhvien
```

Ngoài cách chỉ định cụ số lượng dòng cần hiển thị trong kết quả truy vấn, ta có thể chỉ định số lượng các dòng cần hiển thị theo tỷ lệ phần trăm bằng cách sử dụng thêm từ khoá PERCENT như ở ví dụ dưới đây.

Ví dụ 2.13: Câu lệnh dưới đây hiển thị họ tên và ngày sinh của 10% số lượng sinh viên hiện có trong bảng SINHVIEN

```
SELECT TOP 10 PERCENT hodem,ten,ngaysinh
FROM sinhvien
```

2.1.3 Chỉ định điều kiện truy vấn dữ liệu

Mệnh đề WHERE trong câu lệnh SELECT được sử dụng nhằm xác định các điều kiện đối với việc truy xuất dữ liệu. Sau mệnh đề WHERE là một biểu thức logic và chỉ những dòng dữ liệu nào thoả mãn điều kiện được chỉ định mới được hiển thị trong kết quả truy vấn.

Ví dụ 2.14: Câu lệnh dưới đây hiển thị danh sách các môn học có số đơn vị học trình

lớn hơn 3

```
SELECT * FROM  monhoc
WHERE sodvht>3
```

Kết quả của câu lệnh này như sau:

MAMONHOC	TENMONHOC	SODVHT
TI-001	Tin học đại cương	4
TI-002	Ngôn ngữ C	5
TI-003	Lý thuyết hệ điều hành	4
TI-004	Cấu trúc dữ liệu và giải thuật	4
TO-001	Đại số tuyến tính	4
TO-002	Giải tích 1	4

Trong mệnh đề WHERE thường sử dụng:

Các toán tử kết hợp điều kiện (AND, OR)

Các toán tử so sánh

Kiểm tra giới hạn của dữ liệu (BETWEEN/ NOT BETWEEN)

Danh sách

Kiểm tra khuôn dạng dữ liệu.

Các giá trị NULL

a) Các toán tử so sánh

Toán tử	Ý nghĩa
=	Bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
<>	Khác
!>	Không lớn hơn
!<	Không nhỏ hơn

Ví dụ 2.15: Câu lệnh

```
SELECT masv,hodem,ten,ngaysinh
FROM sinhvien
WHERE (ten='Anh')
 AND (YEAR(GETDATE())-YEAR(ngaysinh)<=20)
```

cho biết mã, họ tên và ngày sinh của các sinh viên có tên là Anh và có tuổi nhỏ hơn hoặc bằng 20.

MASV	HODEM	TEN	NGAYSINH
0261010001	Lê Hoàng Phương	Anh	1984-03-04 00:00:00
0261010002	Lê Thị Vân	Anh	1984-10-14 00:00:00
0261020002	Lê Thúc Quốc	Anh	1984-12-04 00:00:00
0261020004	Nguyễn Thị Lan	Anh	1984-08-23 00:00:00
0261020005	Nguyễn Thị Lan	Anh	1984-07-25 00:00:00

b) Kiểm tra giới hạn của dữ liệu

Để kiểm tra xem giá trị dữ liệu nằm trong (ngoài) một khoảng nào đó, ta sử dụng toán tử BETWEEN (NOT BETWEEN) như sau:

Cách sử dụng	Ý nghĩa
giá_trị BETWEEN a AND b	a ≤ giá_trị ≤ b
giá_trị NOT BETWEEN a AND b	(giá_trị < a) AND (giá_trị>b)

Ví dụ 2.16: Câu lệnh dưới đây cho biết họ tên và tuổi của các sinh viên có tên là Bình và có tuổi nằm trong khoảng từ 20 đến 22

```
SELECT hodem,ten,year(getdate())-year(ngaysinh) AS tuoi
FROM sinhvien
WHERE ten='Bình' AND YEAR(GETDATE())-YEAR(ngaysinh) BETWEEN 20 AND 22
```

c) Danh sách (IN và NOT IN)

Từ khoá IN được sử dụng khi ta cần chỉ định điều kiện tìm kiếm dữ liệu cho câu lệnh SELECT là một danh sách các giá trị. Sau IN (hoặc NOT IN) có thể là một danh sách các giá trị hoặc là một câu lệnh SELECT khác.

Ví dụ 2.17: Để biết danh sách các môn học có số đơn vị học trình là 2, 4 hoặc 5, thay vì sử dụng câu lệnh

```
SELECT * FROM monhoc
WHERE sodvht=2 OR sodvht=4 OR sodvht=5
```

ta có thể sử dụng câu lệnh

```
SELECT * FROM monhoc
WHERE sodvht IN (2,4,5)
```

d) Toán tử LIKE và các ký tự đại diện

Từ khoá LIKE (NOT LIKE) sử dụng trong câu lệnh SELECT nhằm mô tả khuôn dạng của dữ liệu cần tìm kiếm. Chúng thường được kết hợp với các ký tự đại

diện sau đây:

Ký tự đại diện	Ý nghĩa
%	Chuỗi ký tự bất kỳ gồm không hoặc nhiều ký tự
_	Ký tự đơn bất kỳ
[]	Ký tự đơn bất kỳ trong giới hạn được chỉ định (ví dụ

	[a-f]) hay một tập (ví dụ [abcdef])
[^]	Ký tự đơn bất kỳ không nằm trong giới hạn được chỉ định (ví dụ [^a-f] hay một tập (ví dụ [^abcdef]).

Ví dụ 2.18: Câu lệnh dưới đây

SELECT hodem,ten FROM sinhvien

WHERE hodem LIKE 'Lê%'

cho biết họ tên của các sinh viên có họ là Lê và có kết quả như sau

HODEM	TEN
Lê Thị Thanh	Châu
Lê Thị	Anh
Lê Văn Khoa	Bảo
Lê Thị	Dí
Lê Tất Uyên	Châu
Lê Hoàng Phương	Anh
Lê Thị Vân	Anh
Lê Đăng	Ánh
Lê Huy	Đan
Lê Thúc Quốc	Anh

Câu lệnh:

SELECT hodem,ten FROM sinhvien

WHERE hodem LIKE 'Lê%' AND ten LIKE '[AB]%'

Có kết quả là:

HODEM	TEN
Lê Thị	Anh
Lê Văn Khoa	Bảo
Lê Hoàng Phương	Anh
Lê Thị Vân	Anh
Lê Thúc Quốc	Anh

e) Giá trị NULL

Dữ liệu trong một cột cho phép NULL sẽ nhận giá trị NULL trong các trường hợp sau:

Nếu không có dữ liệu được nhập cho cột và không có mặc định cho cột hay kiểu dữ liệu trên cột đó.

Người sử dụng trực tiếp đưa giá trị NULL vào cho cột đó.

Một cột có kiểu dữ liệu là kiểu số sẽ chứa giá trị NULL nếu giá trị được chỉ định gây tràn số.

Trong mệnh đề WHERE, để kiểm tra giá trị của một cột có giá trị NULL hay không, ta sử dụng cách viết:

WHERE tên_cột IS NULL

hoặc:

WHERE tên_cột IS NOT NULL

2.1.4 Tạo mới bảng dữ liệu từ kết quả của câu lệnh SELECT

Câu lệnh SELECT ... INTO có tác dụng tạo một bảng mới có cấu trúc và dữ liệu được xác định từ kết quả của truy vấn. Bảng mới được tạo ra sẽ có số cột bằng số cột được chỉ định trong danh sách chọn và số dòng sẽ là số dòng kết quả của truy vấn.

Ví dụ 2.19: Câu lệnh dưới đây truy vấn dữ liệu từ bảng SINHVIEN và tạo một bảng

TUOISV bao gồm các trường HODEM, TEN và TUOI

```
SELECT hodem,ten,YEAR(GETDATE())-YEAR(ngaysinh) AS tuoi
INTO tuoisv
FROM sinhvien
```

Lưu ý: Nếu trong danh sách chọn có các biểu thức thì những biểu thức này phải được đặt tiêu đề.

2.1.5 Sắp xếp kết quả truy vấn

Mặc định, các dòng dữ liệu trong kết quả của câu truy vấn tuân theo thứ tự của chúng trong bảng dữ liệu hoặc được sắp xếp theo chỉ mục (nếu trên bảng có chỉ mục).

Trong trường hợp muốn dữ liệu được sắp xếp theo chiều tăng hoặc giảm của giá trị của một hoặc nhiều trường, ta sử dụng thêm mệnh đề ORDER BY trong câu lệnh SELECT; Sau ORDER BY là danh sách các cột cần sắp xếp (tối đa là 16 cột). Dữ liệu

đặc định là sắp

MAMONHOC	TENMONHOC	SODVHT
TI-002	Ngôn ngữ C	5
TI-003	Lý thuyết hệ điều hành	4
TI-001	Tin học đại cương	4
TI-004	Cấu trúc dữ liệu và giải thuật	4
TO-001	Đại số tuyến tính	4
TO-002	Giải tích 1	4
HO-001	Hoá đại cương	3
VL-001	Vật lý đại cương	3
TO-004	Bài tập Giải tích 1	2
TO-003	Bài tập Đại số	2

à sắp xếp theo

Nếu sau ORDER BY có nhiều cột thì việc sắp xếp dữ liệu sẽ được ưu tiên theo thứ tự từ trái qua phải.

Ví dụ 2.21: Câu lệnh

```
SELECT hodem,ten,gioitinh, YEAR(GETDATE())-YEAR(ngaysinh) AS tuoi
FROM sinhvien
WHERE ten='Bình'
ORDER BY gioitinh,tuoi
```

có kết quả là:

HODEM	TEN	GIOITINH	TUOI
Nguyễn Thị	Bình	0	23
Hoàng Văn	Bình	1	21
Châu Văn Quốc	Bình	1	21
Nguyễn Thanh	Bình	1	22
Nguyễn Đình	Bình	1	22
Nguyễn Công	Bình	1	25

Thay vì chỉ định tên cột sau ORDER BY, ta có thể chỉ định số thứ tự của cột cần được sắp xếp. Câu lệnh ở ví dụ trên có thể được viết lại như sau:

```
SELECT hodem,ten,gioitinh, YEAR(GETDATE())-YEAR(ngaysinh) AS tuoi
FROM sinhvien
WHERE ten='Bình'
ORDER BY 3, 4
```

2.1.6 Phép hợp

Phép được sử dụng trong trường hợp ta cần gộp kết quả của hai hay nhiều truy vấn thành một tập kết quả duy nhất. SQL cung cấp toán tử UNION để thực hiện phép hợp. Cú pháp như sau

Câu_lệnh_1

UNION [ALL] Câu_lệnh_2

[UNION [ALL] Câu_lệnh_3]

...

[UNION [ALL] Câu_lệnh_n]

[ORDER BY cột_sắp_xếp]

[COMPUTEdanh_sách_hàm_gộp [BY danh_sách_cột]]

Trong đó

Câu_lệnh_1 có dạng

SELECT danh_sách_cột

[INTOtên_bảng_mới]

[FROM danh_sách_bảng|khung_nhìn]

[WHERE điều_kiện]

[GROUP BY danh_sách_cột]

[HAVING điều_kiện]

và Câu_lệnh_i (i = 2,..,n) có dạng

SELECT danh_sách_cột

[FROM danh_sách_bảng|khung_nhìn]

[WHERE điều_kiện]

[GROUP BY danh_sách_cột]

[HAVING điều_kiện]

Ví dụ 2.22: Giả sử ta có hai bảng Table1 và Table2 lần lượt như sau:

A	B	C
a	1	10
b	2	20
c	3	30
d	4	40
a	5	50
b	6	60

D	E
a	1
b	
d	
e	

câu lệnh

SELECT A,B FROM Table1

A	B
a	1
a	5
b	2
b	6
c	3
d	3
d	4
e	4

UNION

SELECT D,E FROM table2

Cho kết quả như sau:

Mặc định, nếu trong các truy vấn thành phần của phép hợp xuất hiện những dòng dữ liệu giống nhau thì trong kết quả truy vấn chỉ giữ lại một dòng. Nếu muốn giữ lại các dòng này, ta phải sử dụng thêm từ khoá ALL trong truy vấn thành phần.

Ví dụ 2.23: Câu lệnh

SELECT A,B FROM Table1

UNION ALL

SELECT D,E FROM table2

Cho kết quả như sau:

A	B
a	1
b	2
c	3
d	4
a	5
b	6
a	1
b	2
d	3
e	4

Khi sử dụng toán tử UNION để thực hiện phép hợp, ta cần chú ý các nguyên tắc sau:

Danh sách cột trong các truy vấn thành phần phải có cùng số lượng.

Các cột tương ứng trong tất cả các bảng, hoặc tập con bất kỳ các cột được sử dụng trong bản thân mỗi truy vấn thành phần phải cùng kiểu dữ liệu.

Các cột tương ứng trong bản thân từng truy vấn thành phần của một câu lệnh UNION phải xuất hiện theo thứ tự như nhau. Nguyên nhân là do phép hợp

so sánh các cột từng cột một theo thứ tự được cho trong mỗi truy vấn.

Khi các kiểu dữ liệu khác nhau được kết hợp với nhau trong câu lệnh UNION, chúng sẽ được chuyển sang kiểu dữ liệu cao hơn (nếu có thể được).

Tiêu đề cột trong kết quả của phép hợp sẽ là tiêu đề cột được chỉ định trong truy vấn đầu tiên.

Truy vấn thành phần đầu tiên có thể có INTO để tạo mới một bảng từ kết quả của chính phép hợp.

Mệnh đề ORDER BY và COMPUTE dùng để sắp xếp kết quả truy vấn hoặc tính toán các giá trị thống kê chỉ được sử dụng ở cuối câu lệnh UNION. Chúng không được sử dụng ở trong bất kỳ truy vấn thành phần nào.

Mệnh đề GROUP BY và HAVING chỉ có thể được sử dụng trong bản thân từng truy vấn thành phần. Chúng không được phép sử dụng để tác động lên kết quả chung của phép hợp.

Phép toán UNION có thể được sử dụng bên trong câu lệnh INSERT.

Phép toán UNION không được sử dụng trong câu lệnh CREATE VIEW.

2.1.7 Phép nối

Khi cần thực hiện một yêu cầu truy vấn dữ liệu từ hai hay nhiều bảng, ta phải sử dụng đến phép nối. Một câu lệnh nối kết hợp các dòng dữ liệu trong các bảng khác nhau lại theo một hoặc nhiều điều kiện nào đó và hiển thị chúng trong kết quả truy vấn.

Xét hai bảng sau đây:

MALOP	TENLOP	KHOA	HEDAOTAO	NAMNHAPHOC	MAKHOA
C24101	Toán K24	24	Chính quy	2000	DHT01
C24102	Tin K24	24	Chính quy	2000	DHT02
C24103	Lý K24	24	Chính quy	2000	DHT03
C24301	Sinh K24	24	Chính quy	2000	DHT05
C25101	Toán K25	25	Chính quy	2001	DHT01
C25102	Tin K25	25	Chính quy	2001	DHT02
C25103	Lý K25	25	Chính quy	2001	DHT03
C25301	Sinh K25	25	Chính quy	2001	DHT05
C26101	Toán K26	26	Chính quy	2002	DHT01
C26102	Tin K26	26	Chính quy	2002	DHT02

Bảng LOP

Bảng KHOA

MAKHOA	TENKHOA	DIENTHOAI
DHT01	Khoa Toán cơ - Tin học	054822407
DHT02	Khoa Công nghệ thông tin	054826767
DHT03	Khoa Vật lý	054823462
DHT04	Khoa Hoá học	054823951
DHT05	Khoa Sinh học	054822934
DHT06	Khoa Địa lý - Địa chất	054823837

Giả sử ta cần biết mã lớp và tên lớp của các lớp thuộc Khoa Công nghệ Thông tin, ta phải làm như sau:

Chọn ra dòng trong bảng KHOA có tên khoa là Khoa Công nghệ Thông tin, từ đó xác định được mã khoa (MAKHOA) là DHT02.

Tìm kiếm trong bảng LOP những dòng có giá trị trường MAKHOA là DHT02 (tức là bằng MAKHOA tương ứng trong bảng KHOA) và đưa những dòng này vào kết quả truy vấn

MAKHOA	TENKHOA	DIENTHOAI
DHT01	Khoa Toán cơ - Tin học	054822407
DHT02	Khoa Công nghệ thông tin	054826767
DHT03	Khoa Vật lý	054823462
DHT04	Khoa Hoá học	054823951
DHT05	Khoa Sinh học	054822934
DHT06	Khoa Địa lý - Địa chất	054823837

MALOP	TENLOP	KHOA	HEDAOTAO	NAMNHAPHOC	MAKHOA
C24101	Toán K24	24	Chính quy	2000	DHT01
C24102	Tin K24	24	Chính quy	2000	DHT02
C24103	Lý K24	24	Chính quy	2000	DHT03
C24301	Sinh K24	24	Chính quy	2000	DHT05
C25101	Toán K25	25	Chính quy	2001	DHT01
C25102	Tin K25	25	Chính quy	2001	DHT02
C25103	Lý K25	25	Chính quy	2001	DHT03
C25301	Sinh K25	25	Chính quy	2001	DHT05
C26101	Toán K26	26	Chính quy	2002	DHT01
C26102	Tin K26	26	Chính quy	2002	DHT02

MALOP	TENLOP
C24102	Tin K24
C25102	Tin K25
C26102	Tin K26

Như vậy, để thực hiện được yêu cầu truy vấn dữ liệu trên, ta phải thực hiện

phép nối giữa hai bảng KHOA và LOP với điều kiện nối là MAKHOA của KHOA bằng với MAKHOA của LOP. Câu lệnh sẽ được viết như sau:

```
SELECT malop,tenlop
FROM khoa,lop
WHERE khoa.makhoa = lop.makhoa AND   tenkhoa='Khoa Công nghệ Thông tin'
```

2.1.7.1 Sử dụng phép nối

Phép ơ sở để thực hiện các yêu cầu truy vấn dữ liệu liên quan đến nhiều bảng. Một câu lệnh nối thực hiện lấy các dòng dữ liệu trong các bảng tham gia truy vấn, so sánh giá trị của các dòng này trên một hoặc nhiều cột được chỉ định trong điều kiện nối và kết hợp các dòng thoả mãn điều kiện thành những dòng trong kết quả truy vấn.

Để thực hiện được một phép nối, cần phải xác định được những yếu tố sau:

Những cột nào cần hiển thị trong kết quả truy vấn

Những bảng nào có tham gia vào truy vấn.

Điều kiện để thực hiện phép nối giữa các bảng dữ liệu là gì

Trong các yếu tố kể trên, việc xác định chính xác điều kiện để thực hiện phép nối giữa các bảng đóng vai trò quan trọng nhất. Trong đa số các trường hợp, điều kiện của phép nối được xác định nhờ vào mối quan hệ giữa các bảng cần phải truy xuất dữ liệu. Thông thường, đó là điều kiện bằng nhau giữa khoá chính và khoá ngoài của hai bảng có mối quan hệ với nhau. Như vậy, để có thể đưa ra một câu lệnh nối thực hiện chính xác yêu cầu truy vấn dữ liệu đòi hỏi phải hiểu được mối quan hệ cũng như ý nghĩa của chúng giữa các bảng dữ liệu.

Danh sách chọn trong phép nối

Một câu lệnh nối cũng được bắt đầu với từ khóa SELECT. Các cột được chỉ định tên sau từ khoá SELECT là các cột được hiển thị trong kết quả truy vấn. Việc sử dụng tên các cột trong danh sách chọn có thể là:

Tên của một số cột nào đó trong các bảng có tham gia vào truy vấn. Nếu tên cột trong các bảng trùng tên nhau thì tên cột phải được viết dưới dạng tên_bảng.tên_cột

Dấu sao (*) được sử dụng trong danh sách chọn khi cần hiển thị tất cả các

cột của các bảng tham gia truy vấn.

Trong trường hợp cần hiển thị tất cả các cột của một bảng nào đó, ta sử dụng cách viết: tên_bảng.*

Mệnh đề FROM trong phép nối

Sau mệnh đề FROM của câu lệnh nối là danh sách tên các bảng (hay khung nhìn) tham gia vào truy vấn. Nếu ta sử dụng dấu * trong danh sách chọn thì thứ tự củacác bảng liệt kê sau FROM sẽ ảnh hưởng đến thứ tự các cột được hiển thị trong kết quả truy vấn.

Mệnh đề WHERE trong phép nối

Khi hai hay nhiều bảng được nối với nhau, ta phải chỉ định điều kiện để thực hiện phép nối ngay sau mệnh đề WHERE. Điều kiện nối được biểu diễn dưới dạng biểu thức logic so sánh giá trị dữ liệu giữa các cột của các bảng tham gia truy vấn.

Các toán tử so sánh dưới đây được sử dụng để xác định điều kiện nối.

Phép toán	Ý nghĩa
=	Bằng
>	Lớn hơn
<	Nhỏ hơn
>=	Lớn hơn hoặc bằng
<=	Nhỏ hơn hoặc bằng
<>	Khác
!>	Không lớn hơn
!<	Không nhỏ hơn

Ví dụ 2.24: Câu lệnh dưới đây hiển thị danh sách các sinh viên với các thông tin: mã sinh viên, họ và tên, mã lớp, tên lớp và tên khoa

```
SELECT masv,hodem,ten,sinhvien.malop,tenlop,tenkhoa
FROM sinhvien,lop,khoa
WHERE sinhvien.malop = lop.malop AND  lop.makhoa=khoa.makhoa
```

Trong câu lệnh trên, các bảng tham gia vào truy vấn bao gồm SINHVIEN, LOP và KHOA. Điều kiện để thực hiện phép nối giữa các bảng bao gồm hai điều kiện:

sinhvien.malop = lop.malop

và

lop.malop = khoa.malop

Điều kiện nối giữa các bảng trong câu lệnh trên là điều kiện bằng giữa khoá ngoài và khoá chính của các bảng có mối quan hệ với nhau. Hay nói cách khác, điều kiện của phép nối được xác định dựa vào mối quan hệ giữa các bảng trong cơ sở dữ liệu.

2.1.7.2 Các loại phép nối

Phép nối bằng và phép nối tự nhiên

Một phép nối bằng (equi-join) là một phép nối trong đó giá trị của các cột được sử dụng để nối được so sánh với nhau dựa trên tiêu chuẩn bằng và tất cả các cột trong các bảng tham gia nối đều được đưa ra trong kết quả.

Ví dụ 2.25: Câu lệnh dưới đây thực hiện phép nối bằng giữa hai bảng LOP và KHOA

```
SELECT *
FROM lop,khoa
WHERE lop.makhoa=khoa.makhoa
```

Trong kết quả của câu lệnh trên, cột makhoa (mã khoa) xuất hiện hai lần trong kết quả phép nối (cột makhoa của bảng khoa và cột makhoa của bảng lop) và như vậy là không cần thiết. Ta có thể loại bỏ bớt đi những cột trùng tên trong kết quả truy vấn bằng cách chỉ định danh sách cột cần được hiển thị trong danh sách chọn của câu lệnh.

Một dạng đặc biệt của phép nối bằng được sử dụng nhiều là phép nối tự nhiên (natural-join). Trong phép nối tự nhiên, điều kiện nối giữa hai bảng chính là điều kiện bằng giữa khoá ngoài và khoá chính của hai bảng; Và trong danh sách chọn của câu lệnh chỉ giữ lại một cột trong hai cột tham gia vào điều kiện của phép nối.

Ví dụ 2.26: Để thực hiện phép nối tự nhiên, câu lệnh trong ví dụ 2.25 được viết lại như sau:

```
SELECT malop,tenlop,khoa,hedaotao,namnhaphoc, siso,lop.makhoa,tenkhoa,dienthoai
FROM lop,khoa
WHERE lop.makhoa=khoa.makhoa
```

hoặc viết dưới dạng ngắn gọn hơn:

```
SELECT lop.*,tenkhoa,dienthoai
```

```
FROM lop,khoa
WHERE lop.makhoa=khoa.makhoa
```

Phép nối với các điều kiện bổ sung

Trong các câu lệnh nối, ngoài điều kiện của phép nối được chỉ định trong mệnh đề WHERE còn có thể chỉ định các điều kiện tìm kiếm dữ liệu khác (điều kiện chọn). Thông thường, các điều kiện này được kết hợp với điều kiện nối thông qua toán tử AND.

Ví dụ 2.27: Câu lệnh dưới đây hiển thị họ tên và ngày sinh của các sinh viên Khoa Công nghệ Thông tin

```
SELECT hodem,ten,ngaysinh
FROM sinhvien,lop,khoa
WHERE tenkhoa='Khoa Công nghệ Thông tin' AND  sinhvien.malop = lop.malop AND  lop.makhoa = khoa.makhoa
```

Phép tự nối và các bí danh

Phép tự nối là phép nối mà trong đó điều kiện nối được chỉ định liên quan đến các cột của cùng một bảng. Trong trường hợp này, sẽ có sự xuất hiện tên của cùng một bảng nhiều lần trong mệnh đề FROM và do đó các bảng cần phải được đặt bí danh.

Ví dụ 2.28: Để biết được họ tên và ngày sinh của các sinh viên có cùng ngày sinh với sinh viên Trần Thị Kim Anh, ta phải thực hiện phép tự nối ngay trên chính bảng sinhvien. Trong câu lệnh nối, bảng sinhvien xuất hiện trong mệnh đề FROM với bí danh là a và b. Bảng sinhvien với bí danh là a sử dụng để chọn ra sinh viên có họ tên là Trần Thị Kim Anh và bảng sinhvien với bí danh là b sử dụng để xác định các sinh viên trùng ngày sinh với sinh viên Trần Thị Kim Anh. Câu lệnh được viết như sau:

```
SELECT b.hodem,b.ten,b.ngaysinh
FROM sinhvien a, sinhvien b
WHERE a.hodem='Trần Thị Kim' AND a.ten='Anh' AND
    a.ngaysinh=b.ngaysinh AND a.masv<>b.masv
```

Phép nối không dựa trên tiêu chuẩn bằng

Trong phép nối này,điều kiện để thực hiện phép nối giữa các bảng dữ liệu không phải là điều kiện so sành bằng giữa các cột. Loại phép nối này trong thực tế thường ít được sử dụng.

Phép nối ngoài (outer-join)

Trong các phép nối đã đề cập ở trên, chỉ những dòng có giá trị trong các cột được chỉ định thoả mãn điều kiện kết nối mới được hiển thị trong kết quả truy vấn, và được gọi là phép nối trong (inner join) Theo một nghĩa nào đó, những phép nối này loại bỏ thông tin chứa trong những dòng không thoả mãn điều kiện nối. Tuy nhiên, đôikhi ta cũng cần giữ lại những thông tin này bằng cách cho phép những dòng không thoả mãn điều kiện nối có mặt trong kết quả của phép nối. Để làm điều này, ta có thểsử dụng phép nối ngoài.

SQL cung cấp các loại phép nối ngoài sau đây:

Phép nối ngoài trái (ký hiệu: *=): Phép nối này hiển thị trong kết quả truy vấn tất cả các dòng dữ liệu của bảng nằm bên trái trong điều kiện nối cho dù những dòng này không thoả mãn điều kiện của phép nối.

Phép nối ngoài phải (ký hiệu: =*): Phép nối này hiển thị trong kết quả truy vấn tất cả các dòng dữ liệu của bảng nằm bên phải trong điều kiện nối cho dù những dòng này không thoả điều kiện của phép nối.

Ví dụ 2.29: Giả sử ta có hai bảng DONVI và NHANVIEN như sau:

Bảng DONVI

MADV	TENDV
1	Doi ngoai
2	Hanh chinh
3	Ke toan
4	Kinh doanh

Bảng NHANVIEN

HOTEN	MADV
Thanh	1
Hoa	2
Nam	2
Vinh	1
Hung	5
Phuong	NULL

Câu lệnh:

```
SELECT *
FROM nhanvien,donvi
WHERE nhanvien.madv=donvi.madv
```

có kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai

Nếu thực hiện phép nối ngoài trái giữa bảng NHANVIEN và bảng DONVI:

SELECT *

FROM nhanvien,donvi

WHERE nhanvien.madv*=donvi.madv

kết quả của câu lệnh sẽ là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai
Hung	5	NULL	NULL
Phuong	NULL	NULL	NULL

Và kết quả của phép nối ngoài phải:

select *

from nhanvien,donvi

where nhanvien.madv=*donvi.madv

như sau:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Vinh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
NULL	NULL	3	Ke toan
NULL	NULL	4	Kinh doanh

Phép nối và các giá trị NULL

Nếu trong các cột của các bảng tham gia vào điều kiện của phép nối có các giá trị NULL thì các giá trị NULL được xem như là không bằng nhau.

Ví dụ 2.30: Giả sử ta có hai bảng TABLE1 và TABLE2 như sau:

TABLE1AB1B1NULLB24B3 TABLE 2CDNULLD14D2

Câu lệnh:

SELECT *

FROM table1, table2

WHERE A *= C

Có kết quả là:

A	B	C	D
1	B1	NULL	NULL
NULL	B2	NULL	NULL
4	B3	4	D2

2.1.7.4 Sử dụng phép nối trong SQL2

Ở phần trước đã đề cập đến phương pháp sử dụng phép nối trong và phép nối ngoài trong truy vấn SQL. Như đã trình bày, điều kiện của phép nối trong câu lệnh được chỉ định trong mệnh đề WHERE thông qua các biểu thức so sánh giữa các bảng tham gia truy vấn.

Chuẩn SQL2 (SQL-92) đưa ra một cách khác để biểu diễn cho phép nối, trong cách biểu diễn này, điều kiện của phép nối không được chỉ định trong mệnh đề WHERE mà được chỉ định ngay trong mệnh đề FROM của câu lệnh. Cách sử dụng phép nối này cho phép ta biểu diễn phép nối cũng như điều kiện nối được rõ ràng, đặc biệt là trong trường hợp phép nối được thực hiện trên ba bảng trở lên.

Phép nối trong

Điều kiện để thực hiện phép nối trong được chỉ định trong mệnh đề FROM theo cú pháp như sau:

tên_bảng_1 [INNER] JOIN tên_bảng_2 ON điều_kiện_nối

Ví dụ 2.31: Để hiển thị họ tên và ngày sinh của các sinh viên lớp Tin K24, thay vì sử dụng câu lệnh:

SELECT hodem,ten,ngaysinh

FROM sinhvien,lop

WHERE tenlop='Tin K24' AND sinhvien.malop=lop.malop

ta có thể sử dụng câu lệnh như sau:

SELECT hodem,ten,ngaysinh

FROM sinhvien INNER JOIN lop

ON sinhvien.malop=lop.malop

WHERE tenlop='Tin K24'

Phép nối ngoài

SQL2 cung cấp các phép nối ngoài sau đây:

Phép nối ngoài trái (LEFT OUTER JOIN)

Phép nối ngoài phải (RIGHT OUTER JOIN)

Phép nối ngoài đầy đủ (FULL OUTER JOIN)

Cũng tương tự như phép nối trong, điều kiện của phép nối ngoài cũng được chỉ định ngay trong mệnh đề FROM theo cú pháp:

tên_bảng_1 LEFT|RIGHT|FULL [OUTER] JOIN tên_bảng_2

ON điều_kiện_nối

Ví dụ 2.32: Giả sử ta có hai bảng DONVI, NHANVIEN dữ liệu như sau:

MADV	TENDV
1	Doi ngoai
2	Hanh chinh
3	Ke toan
4	Kinh doanh

HOTEN	MADV
Thanh	1
Hoa	2
Nam	2
Vinh	1
Hung	5
Phuong	NULL

Phé ; N ı bởi câu lệnh:

SELECT *

FROM nhanvien LEFT OUTER JOIN donvi

ON nhanvien.madv=donvi.madv

có kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai
Hung	5	NULL	NULL
Phuong	NULL	NULL	NULL

Câu lệnh:

SELECT *

FROM nhanvien RIGHT OUTER JOIN donvi

ON nhanvien.madv=donvi.madv

thực hiện phép nối ngoài phải giữa hai bảng NHANVIEN và DONVI, và có kết quả là:

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Vinh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
NULL	NULL	3	Ke toan
NULL	NULL	4	Kinh doanh

Nếu phép nối ngoài trái (tương ứng phải) hiển thị trong kết quả truy vấn cả những dòng dữ liệu không thoả điều kiện nối của bảng bên trái (tương ứng phải) trong phép nối thì phép nối ngoài đầy đủ hiển thị trong kết quả truy vấn cả những dòng dữ liệu không thoả điều kiện nối của cả hai bảng tham gia vào phép nối.

Ví dụ 2.33: Với hai bảng NHANVIEN và DONVI như ở trên, câu lệnh

SELECT *

FROM nhanvien FULL OUTER JOIN donvi

ON nhanvien.madv=donvi.madv

HOTEN	MADV	MADV	TENDV
Thanh	1	1	Doi ngoai
Hoa	2	2	Hanh chinh
Nam	2	2	Hanh chinh
Vinh	1	1	Doi ngoai
Hung	5	NULL	NULL
Phuong	NULL	NULL	NULL
NULL	NULL	4	Kinh doanh
NULL	NULL	3	Ke toan

Thực hiện phép nối trên nhiều bảng

Một đặc điểm nổi bật của SQL2 là cho phép biểu diễn phép nối trên nhiều bảng dữ liệu một cách rõ ràng. Thứ tự thực hiện phép nối giữa các bảng được xác định theo nghĩa kết quả của phép nối này được sử dụng trong một phép nối khác.

Ví dụ 2.34: Câu lệnh dưới đây hiển thị họ tên và ngày sinh của các sinh viên thuộcKhoa Công nghệ Thông tin

```
SELECT hodem,ten,ngaysinh
FROM (sinhvien INNER JOIN lop
        ON sinhvien.malop=lop.malop)
      INNER JOIN khoa ON lop.makhoa=khoa.makhoa
WHERE tenkhoa=N'Khoa công nghệ thông tin'
```

Trong câu lệnh trên, thứ tự thực hiện phép nối giữa các bảng được chỉ định rõ ràng: phép nối giữa hai bảng sinhvien và lop được thực hiện trước và kết quả của phép nối này lại tiếp tục được nối với bảng khoa.

2.1.8 Thống kê dữ liệu với GROUP BY

Ngoài khả năng thực hiện các yêu cầu truy vấn dữ liệu thông thường (chiếu, chọn, nối,…) như đã đề cập như ở các phần trước, câu lệnh SELECT còn cho phép thực hiện các thao tác truy vấn và tính toán thống kê trên dữ liệu như: cho biết tổng số tiết dạy của mỗi giáo viên, điểm trung bình các môn học của mỗi sinh viên,…

Mệnh đề GROUP BY sử dụng trong câu lệnh SELECT nhằm phân hoạch các dòng dữ liệu trong bảng thành các nhóm dữ liệu, và trên mỗi nhóm dữ liệu thực hiện tính toán các giá trị thống kê như tính tổng, tính giá trị trung bình,...

Các hàm gộp được sử dụng để tính giá trị thống kê cho toàn bảng hoặc trên mỗi nhóm dữ liệu. Chúng có thể được sử dụng như là các cột trong danh sách chọn của câu lệnh SELECT hoặc xuất hiện trong mệnh đề HAVING, nhưng không được

phép xuất hiện trong mệnh đề WHERE

SQL cung cấp các hàm gộp dưới đây:

Hàm gộp Chức năng

SUM([ALL | DISTINCT] biểu_thức)	Tính tổng các giá trị
AVG([ALL | DISTINCT] biểu_thức)	Tính trung bình của các giá trị
COUNT([ALL | DISTINCT] biểu_thức)	Đếm số các giá trị trong biểu thức.
COUNT(*)	Đếm số các dòng được chọn.
MAX(biểu_thức)	Tính giá trị lớn nhất
MIN(biểu_thức)	Tính giá trị nhỏ nhất

Trong đó: Hàm SUM và AVG chỉ làm việc với các biểu thức số.

Hàm SUM, AVG, COUNT, MIN và MAX bỏ qua các giá trị NULL khi tính toán.

Hàm COUNT(*) không bỏ qua các giá trị NULL. Mặc định, các hàm gộp thực hiện tính toán thống kê trên toàn bộ dữ liệu. Trong trường hợp cần loại bỏ bớt các giá trị trùng nhau (chỉ giữ lại một giá trị), ta chỉ định thêm từ khoá DISTINCT ở trước biểu thức là đối số của hàm

Thống kê trên toàn bộ dữ liệu

Khi cần tính toán giá trị thống kê trên toàn bộ dữ liệu, ta sử dụng các hàm gộp trong danh sách chọn của câu lệnh SELECT. Trong trường hợp này, trong danh sách chọn không được sử dụng bất kỳ một tên cột hay biểu thức nào ngoài các hàm gộp.

Ví dụ 2.35: Để thống kê trung bình điểm lần 1 của tất cả các môn học, ta sử dụng câu lệnh như sau:

```
SELECT AVG(diemlan1)
FROM diemthi
```

còn câu lệnh dưới đây cho biết tuổi lớn nhất, tuổi nhỏ nhất và độ tuổi trung bình của tất cả các sinh viên sinh tại Huế:

```
SELECT MAX(YEAR(GETDATE())-YEAR(ngaysinh)),
  MIN(YEAR(GETDATE())-YEAR(ngaysinh)),
  AVG(YEAR(GETDATE())-YEAR(ngaysinh))
FROM sinhvien
WHERE noisinh='Huế'
```

Thống kê dữ liệu trên các nhóm

Trong trường hợp cần thực hiện tính toán các giá trị thống kê trên các nhóm dữ liệu,

ta sử dụng mệnh đề GROUP BY để phân hoạch dữ liệu vào trong các nhóm. Các hàm gộp được sử dụng sẽ thực hiện thao tác tính toán trên mỗi nhóm và cho biết giá trị thống kê theo các nhóm dữ liệu.

Ví dụ 2.36: Câu lệnh dưới đây cho biết sĩ số (số lượng sinh viên) của mỗi lớp

```
SELECT lop.malop,tenlop,COUNT(masv) AS siso
FROM lop,sinhvien
WHERE lop.malop=sinhvien.malop
GROUP BY lop.malop,tenlop
```

và có kết quả là

MALOP	TENLOP	SISO
C24101	Toán K24	5
C24102	Tin K24	8
C24103	Lý K24	7
C24301	Sinh K24	5
C25101	Toán K25	5
C25102	Tin K25	6
C25103	Lý K25	6
C25301	Sinh K25	8
C26101	Toán K26	5
C26102	Tin K26	5

```
SELECT sinhvien.masv,hodem,ten,
     sum(diemlan1*sodvht)/sum(sodvht)
FROM sinhvien,diemthi,monhoc
WHERE sinhvien.masv=diemthi.masv AND
    diemthi.mamonhoc=monhoc.mamonhoc
GROUP BY sinhvien.masv,hodem,ten
```

cho biết trung bình điểm thi lần 1 các môn học của các sinh viên

Lưu ý: Trong trường hợp danh sách chọn của câu lệnh SELECT có cả các hàm gộp và những biểu thức không phải là hàm gộp thì những biểu thức này phải có mặt đầy đủ trong mệnh đề GROUP BY, nếu không câu lệnh sẽ không hợp lệ.

Ví dụ 2.37: Dưới đây là một câu lệnh sai

```
SELECT lop.malop,tenlop,COUNT(masv)
FROM lop,sinhvien
WHERE lop.malop=sinhvien.malop
```

GROUP BY lop.malop

do thiếu trường TENLOP sau mệnh đề GROUP BY.

Chỉ định điều kiện đối với hàm gộp

Mệnh đề HAVING được sử dụng nhằm chỉ định điều kiện đối với các giá trị thống kê được sản sinh từ các hàm gộp tương tự như cách thức mệnh đề WHERE thiết lập các điều kiện cho câu lệnh SELECT. Mệnh đề HAVING thường không thực sự có nghĩa nếu như không sử dụng kết hợp với mệnh đề GROUP BY. Một điểm khác biệt giữa HAVING và WHERE là trong điều kiện của WHERE không được có các hàm gộp trong khi HAVING lại cho phép sử dụng các hàm gộp trong điều kiện của mình.

Ví dụ 2.38: Để biết trung bình điểm thi lần 1 của các sinh viên có điểm trung bình lớn

hơn hoặc bằng 5, ta sử dụng câu lệnh như sau:

```
SELECT sinhvien.masv,hodem,ten,
     SUM(diemlan1*sodvht)/sum(sodvht)
FROM sinhvien,diemthi,monhoc
WHERE sinhvien.masv=diemthi.masv AND
    diemthi.mamonhoc=monhoc.mamonhoc
GROUP BY sinhvien.masv,hodem,ten
HAVING sum(diemlan1*sodvht)/sum(sodvht)>=5
```

2.1.9 Thống kê dữ liệu với COMPUTE

Khi thực hiện thao tác thống kê với GROUP BY, kết quả thống kê (được sản sinh bởi hàm gộp) xuất hiện dưới một cột trong kết quả truy vấn. Thông qua dạng truy vấn này, ta biết được giá trị thống kê trên mỗi nhóm dữ liệu nhưng không biết được chi tiết dữ liệu trên mỗi nhóm

Ví dụ 2.39: Câu lệnh:

```
SELECT khoa.makhoa,tenkhoa,COUNT(malop) AS solop
FROM khoa,lop
WHERE khoa.makhoa=lop.makhoa
GROUP BY khoa.makhoa,tenkhoa
```

cho ta biết được số lượng lớp của mỗi khoa với kết quả như sau:

MAKHOA	TENKHOA	SOLOP
DHTO1	Khoa Toán cơ - Tin học	3
DHTO2	Khoa Công nghệ thông tin	3
DHTO3	Khoa Vật lý	2
DHTO5	Khoa Sinh học	2

nhưng cụ thể mỗi khoa bao gồm những lớp nào thì chúng ta không thể biết được trong kết quả truy vấn trên.

Mệnh đề COMPUTE sử dụng kết hợp với các hàm gộp (dòng) và ORDER BY trong câu lệnh SELECT cũng cho chúng ta các kết quả thống kê (của hàm gộp) trên các nhóm dữ liệu. Điểm khác biệt giữa COMPUTE và GROUP BY là kết quả thống kê xuất hiện dưới dạng một dòng trong kết quả truy vấn và còn cho chúng ta cả chi tiết về dữ liệu trong mỗi nhóm. Như vậy, câu lệnh SELECT với COMPUTE cho chúng ta cả chi tiết dữ liệu và giá trị thống kê trên mỗi nhóm.

Mệnh đề COMPUTE …BY có cú pháp như sau:

COMPUTE hàm_gộp(tên_cột) [,…, hàm_gộp (tên_cột)]

BY danh_sách_cột

Trong đó:

Các hàm gộp có thể sử dụng bao gồm SUM, AVG, MIN, MAX và COUNT.

danh_sách_cột: là danh sách cột sử dụng để phân nhóm dữ liệu

Ví dụ 2.40: Câu lệnh dưới đây cho biết danh sách các lớp của mỗi khoa và tổng số các lớp của mỗi khoa:

```
SELECT khoa.makhoa,tenkhoa,malop,tenlop FROM khoa,lop
WHERE khoa.makhoa=lop.makhoa
ORDER BY khoa.makhoa
COMPUTE COUNT(malop) BY khoa.makhoa
```

Khi sử dụng mệnh đề COMPUTE ... BY cần tuân theo các qui tắc dưới đây:

Từ khóa DISTINCT không cho phép sử dụng với các hàm gộp dòng

Hàm COUNT(*) không được sử dụng trong COMPUTE.

Sau COMPUTE có thể sử dụng nhiều hàm gộp, khi đó các hàm phải phân cách nhau bởi dấu phẩy.

Các cột sử dụng trong các hàm gộp xuất hiện trong mệnh đề COMPUTE phải có mặt trong danh sách chọn.

Không sử dụng SELECT INTO trong một câu lệnh SELECT có sử dụng COMPUTE.

Nếu sử dụng mệnh đề COMPUTE ... BY thì cũng phải sử dụng mệnh đề ORDER BY. Các cột liệt kê trong COMPUTE … BY phải giống hệt hay là một tập con của những gì được liệt kê sau ORDER BY. Chúng phải có cùng thứ tự từ trái qua phải, bắt đầu với cùng một biểu thức và không bỏ qua bất kỳ một biểu thức nào

Chẳng hạn nếu mệnh đề ORDER BY có dạng:

ORDER BY a, b, c

Thì mệnh đề COMPUTE BY với hàm gộp F trên cột X theo một trong các cách dưới đây là hợp lệ:

COMPUTE F(X) BY a, b, c

COMPUTE F(X) BY a, b

COMPUTE F(X) BY a

Và các cách sử dụng dưới đây là sai:

COMPUTE F(X) BY b, c

COMPUTE F(X) BY a, c

COMPUTE F(X) BY c

Phải sử dụng một tên cột hoặc một biểu thức trong mệnh đề ORDER BY, việc sắp xếp không được thực hiện dựa trên tiêu đề cột.

Trong trường hợp sử dụng COMPUTE mà không có BY thì có thể không cần sử dụng ORDER BY, khi đó phạm vi tính toán của hàm gộp là trên toàn bộ dữ liệu.

Ví dụ 2.41: Câu lệnh dưới đây hiển thị danh sách các lớp và tổng số lớp hiện có:

```
SELECT malop,tenlop,hedaotao
FROM lop
ORDER BY makhoa
COMPUTE COUNT(malop)
```

kết quả của câu lệnh như sau

MALOP	TENLOP	HEDAOTAO
C24101	Toán K24	Chính quy
C25101	Toán K25	Chính quy
C26101	Toán K26	Chính quy
C26102	Tin K26	Chính quy
C25102	Tin K25	Chính quy
C24102	Tin K24	Chính quy
C24103	Lý K24	Chính quy
C25103	Lý K25	Chính quy
C25301	Sinh K25	Chính quy
C24301	Sinh K24	Chính quy
CNT		
10		

Có thể thực hiện việc tính toán hàm gộp dòng trên các nhóm lồng nhau bằng cách sử dụng nhiều mệnh đề COMPUTE … BY trong cùng một câu lệnh SELECT

Ví dụ 2.42: Cho biết danh sách các lớp của mỗi khoa, tổng số lớp theo mỗi khoa và tổng số lớp hiện có:

```
SELECT khoa.makhoa,tenkhoa,malop,tenlop FROM khoa,lop
WHERE khoa.makhoa=lop.makhoa
ORDER BY khoa.makhoa
COMPUTE COUNT(malop) BY khoa.makhoa
COMPUTE COUNT(malop)
```

2.1.10 Truy vấn con (Subquery)

Truy vấn con là một câu lệnh SELECT được lồng vào bên trong một câu lệnh SELECT, INSERT, UPDATE, DELETE hoặc bên trong một truy vấn con khác. Loại truy vấn này được sử dụng để biểu diễn cho những truy vấn trong đó điều kiện truy vấn dữ liệu cần phải sử dụng đến kết quả của một truy vấn khác.

Cú pháp của truy vấn con như sau:

```
(SELECT [ALL | DISTINCT] danh_sách_chọn
FROM  danh_sách_bảng
[WHERE    điều_kiện]
```

[GROUP BY danh_sách_cột]

[HAVING điều_kiện])

Khi sử dụng truy vấn con cần lưu ý một số quy tắc sau:

Một truy vấn con phải được viết trong cặp dấu ngoặc. Trong hầu hết các trường hợp, một truy vấn con thường phải có kết quả là một cột (tức là chỉ có duy nhất một cột trong danh sách chọn).

Mệnh đề COMPUTE và ORDER BY không được phép sử dụng trong truy vấn con.

Các tên cột xuất hiện trong truy vấn con có thể là các cột của các bảng trong truy vấn ngoài.

Một truy vấn con thường được sử dụng làm điều kiện trong mệnh đề WHERE hoặc HAVING của một truy vấn khác.

Nếu truy vấn con trả về đúng một giá trị, nó có thể sử dụng như là một thành phần bên trong một biểu thức (chẳng hạn xuất hiện trong một phép so sánh bằng)

Phép so sánh đối với với kết quả truy vấn con

Kết quả của truy vấn con có thể được sử dụng để thực hiện phép so sánh số học với một biểu thức của truy vấn cha. Trong trường hợp này, truy vấn con được sử dụng dưới dạng:

WHERE biểu_thức phép_toán_số_học [ANY|ALL]

(truy_vấn_con)

Trong đó phép toán số học có thể sử dụng bao gồm: =, <>, >, <, >=, <=; Và truy vấn con phải có kết quả bao gồm đúng một cột.

Ví dụ 2.43: Câu lệnh dưới đây cho biết danh sách các môn học có số đơn vị học trình lớn hơn hoặc bằng số đơn vị học trình của môn học có mã là TI-001

```
SELECT *
FROM monhoc
WHERE sodvht>=(SELECT sodvht  FROM monhoc  WHERE mamonhoc='TI-001')
```

Nếu truy vấn con trả về nhiều hơn một giá trị, việc sử dụng phép so sánh như trên sẽ không hợp lệ. Trong trường hợp này, sau phép toán so sánh phải sử dụng thêm lượng từ ALL hoặc ANY. Lượng từ ALL được sử dụng khi cần so sánh giá trị của

biểu thức với tất cả các giá trị trả về trong kết quả của truy vấn con; ngược lai, phép so sánh với lượng từ ANY có kết quả đúng khi chỉ cần một giá trị bất kỳ nào đó trong kết quả của truy vấn con thoả mãn điều kiện. Ví dụ 2.44: Câu lệnh dưới đây cho biết họ tên của những sinh viên lớp Tin K25 sinh trước tất cả các sinh viên của lớp Toán K25

```
SELECT hodem,ten
FROM sinhvien JOIN lop ON sinhvien.malop=lop.malop
WHERE tenlop='Tin K25' AND
ngaysinh<ALL(SELECT ngaysinh
  FROM sinhvien JOIN lop
        ON sinhvien.malop=lop.malop
  WHERE lop.tenlop='Toán K25')
```

và câu lệnh:

```
SELECT hodem,ten
FROM sinhvien JOIN lop on sinhvien.malop=lop.malop
WHERE tenlop='Tin K25' AND
 year(ngaysinh)= ANY(SELECT year(ngaysinh)
 FROM sinhvien JOIN lop
 ON sinhvien.malop=lop.malop
 WHERE lop.tenlop='Toán K25')
```

cho biết họ tên của những sinh viên lớp Tin K25 có năm sinh trùng với năm sinh của bất kỳ một sinh viên nào đó của lớp Toán K25.

Sử dụng truy vấn con với toán tử IN

Khi cần thực hiện phép kiểm tra giá trị của một biểu thức có xuất hiện (không xuất hiện) trong tập các giá trị của truy vấn con hay không, ta có thể sử dụng toán tử IN (NOT IN) như sau:

```
WHERE biểu_thức [NOT] IN (truy_vấn_con)
```

Ví dụ 2.45: Để hiển thị họ tên của những sinh viên lớp Tin K25 có năm sinh bằng với năm sinh của một sinh viên nào đó của lớp Toán K25, thay vì sử dụng câu lệnh như ở ví dụ trên, ta có thể sử dụng câu lệnh như sau:

```
SELECT hodem,ten
```

```
FROM sinhvien JOIN lop on sinhvien.malop=lop.malop
WHERE tenlop='Tin K25' AND
 year(ngaysinh)IN(SELECT year(ngaysinh)
   FROM sinhvien JOIN lop
   ON sinhvien.malop=lop.malop
   WHERE lop.tenlop='Toán K25')
```

Sử dụng lượng từ EXISTS với truy vấn con

Lượng từ EXISTS được sử dụng kết hợp với truy vấn con dưới dạng:

WHERE [NOT] EXISTS (truy_vấn_con)

để kiểm tra xem một truy vấn con có trả về dòng kết quả nào hay không. Lượng từ EXISTS (tương ứng NOT EXISTS) trả về giá trị True (tương ứng False) nếu kết quả của truy vấn con có ít nhất một dòng (tương ứng không có dòng nào). Điều khác biệt của việc sử dụng EXISTS với hai cách đã nêu ở trên là trong danh sách chọn của truy vấn con có thể có nhiều hơn hai cột.

Ví dụ 2.46: Câu lệnh dưới đây cho biết họ tên của những sinh viên hiện chưa có điểm thi của bất kỳ một môn học nào

```
SELECT hodem,ten
FROM sinhvien
WHERE NOT EXISTS(SELECT masv FROM diemthi
  WHERE diemthi.masv=sinhvien.masv)
```

Sử dụng truy vấn con với mệnh đề HAVING

Một truy vấn con có thể được sử dụng trong mệnh đề HAVING của một truy vấn khác. Trong trường hợp này, kết quả của truy vấn con được sử dụng để tạo nên điều kiện đối với các hàm gộp.

Ví dụ 2.47: Câu lệnh dưới đây cho biết mã, tên và trung bình điểm lần 1 của các mônhọc có trung bình lớn hơn trung bình điểm lần 1 của tất cả các môn học

```
SELECT diemthi.mamonhoc,tenmonhoc,AVG(diemlan1)
FROM diemthi,monhoc
WHERE diemthi.mamonhoc=monhoc.mamonhoc
GROUP BY diemthi.mamonhoc,tenmonhoc
HAVING AVG(diemlan1)>
```

(SELECT AVG(diemlan1) FROM diemthi)

2.2 Bổ sung, cập nhật và xoá dữ liệu

Các câu lệnh thao tác dữ liệu trong SQL không những chỉ sử dụng để truy vấn dữ liệu mà còn để thay đổi và cập nhật dữ liệu trong cơ sở dữ liệu. So với câu lệnh SELECT, việc sử dụng các câu lệnh để bổ sung, cập nhật hay xoá dữ liệu đơn giản hơn nhiều. Trong phần còn lại của chương này sẽ đề cập đến 3 câu lệnh:

Lệnh INSERT

Lệnh UPDATE

Lệnh DELETE

2.2.1 Bổ sung dữ liệu

Dữ liệu trong các bảng được thể hiện dưới dạng các dòng (bản ghi). Để bổ sung thêm các dòng dữ liệu vào một bảng, ta sử dụng câu lệnh INSERT. Hầu hết các hệ quản trị CSDL dựa trên SQL cung cấp các cách dưới đây để thực hiện thao tác bổ sung dữ liệu cho bảng:

Bổ sung từng dòng dữ liệu với mỗi câu lệnh INSERT. Đây là các sử dụng thường gặp nhất trong giao tác SQL.

Bổ sung nhiều dòng dữ liệu bằng cách truy xuất dữ liệu từ các bảng dữ liệu khác.

Bổ sung từng dòng dữ liệu với lệnh INSERT

Để bổ sung một dòng dữ liệu mới vào bảng, ta sử dụng câu lệnh INSERT với cú pháp như sau:

INSERT INTO tên_bảng[(danh_sách_cột)]
VALUES(danh_sách_trị)

Trong câu lệnh INSERT, danh sách cột ngay sau tên bảng không cần thiết phải chỉ định nếu giá trị các trường của bản ghi mới được chỉ định đầy đủ trong danh sách trị.

Trong trường hợp này, thứ tự các giá trị trong danh sách trị phải bằng với số lượng các trường của bảng cần bổ sung dữ liệu cũng như phải tuân theo đúng thứ tự của các trường như khi bảng được định nghĩa.

Ví dụ 2.48: Câu lệnh dưới đây bổ sung thêm một dòng dữ liệu vào bảng KHOA

INSERT INTO khoa

VALUES('DHT10','Khoa Luật','054821135')

Trong trường hợp chỉ nhập giá trị cho một số cột trong bảng, ta phải chỉ định danh sách các cột cần nhập dữ liệu ngay sau tên bảng. Khi đó, các cột không được nhập dữ liệu sẽ nhận giá trị mặc định (nếu có) hoặc nhận giá trị NULL (nếu cột cho phép chấp nhận giá trị NULL). Nếu một cột không có giá trị mặc định và không chấp nhận giá trị NULL mà không được nhập dữ liệu, câu lệnh sẽ bị lỗi.

Ví dụ 2.49: Câu lệnh dưới đây bổ sung một bản ghi mới cho bảng SINHVIEN

INSERT INTO sinhvien(masv,hodem,ten,gioitinh,malop)

VALUES('0241020008','Nguyễn Công','Chính',1,'C24102')

câu lệnh trên còn có thể được viết như sau:

INSERT INTO sinhvien

VALUES('0241020008','Nguyễn Công','Chính',

NULL,1,NULL,'C24102')

Bổ sung nhiều dòng dữ liệu từ bảng khác

Một cách sử dụng khác của câu lệnh INSERT được sử dụng để bổ sung nhiều dòng dữ liệu vào một bảng, các dòng dữ liệu này được lấy từ một bảng khác thông qua câu lệnh SELECT. Ở cách này, các giá trị dữ liệu được bổ sung vào bảng không được chỉ định tường minh mà thay vào đó là một câu lệnh SELECT truy vấn dữ liệu từ bảng khác.

Cú pháp câu lệnh INSERT có dạng như sau:

INSERT INTO tên_bảng[(danh_sách_cột)] câu_lệnh_SELECT

Ví dụ 2.50: Giả sử ta có bảng LUUSINHVIEN bao gồm các trường HODEM, TEN, NGAYSINH. Câu lệnh dưới đây bổ sung vào bảng LUUSINHVIEN các dòng dữ liệu có được từ câu truy vấn SELECT:

INSERT INTO luusinhvien

SELECT hodem,ten,ngaysinh

FROM sinhvien

WHERE noisinh like '%Huế%'

Khi bổ sung dữ liệu theo cách này cần lưu ý một số điểm sau:

Kết quả của câu lệnh SELECT phải có số cột bằng với số cột được chỉ

định trong bảng đích và phải tương thích về kiểu dữ liệu.

Trong câu lệnh SELECT được sử dụng mệnh đề COMPUTE ... BY

2.2.2 Cập nhật dữ liệu

Câu lệnh UPDATE trong SQL được sử dụng để cập nhật dữ liệu trong các bảng. Câu lệnh này có cú pháp như sau:

```
UPDATE tên_bảng
SET    tên_cột = biểu_thức
[, ..., tên_cột_k = biểu_thức_k]
[FROM  danh_sách_bảng]
[WHERE      điều_kiện]
```

Sau UPDATE là tên của bảng cần cập nhật dữ liệu. Một câu lệnh UPDATE có thể cập nhật dữ liệu cho nhiều cột bằng cách chỉ định các danh sách tên cột và biểu thức tương

ứng sau từ khoá SET. Mệnh đề WHERE trong câu lệnh UPDATE thường được sử dụng để chỉ định các dòng dữ liệu chịu tác động của câu lệnh (nếu không chỉ định, phạm vi tác động của câu lệnh được hiểu là toàn bộ các dòng trong bảng)

Ví dụ 2.51: Câu lệnh dưới đây cập nhật lại số đơn vị học trình của các môn học có số đơn vị học trình nhỏ hơn 2

```
UPDATE monhoc
SET sodvht = 3
WHERE sodvht = 2
```

Sử dụng cấu trúc CASE trong câu lệnh UPDATE

Cấu trúc CASE có thể được sử dụng trong biểu thức khi cần phải đưa ra các quyết định khác nhau về giá trị của biểu thức

Ví dụ 2.52: Giả sử ta có bảng NHATKYPHONG sau đây

SOPHONG	LOAIPHONG	SONGAY	TIENPHONG
101	A	5	
202	B	5	
101	A	2	
102	C	3	

Sau khi thực hiện câu lệnh:

```
UPDATE nhatkyphong
SET tienphong=songay*CASE WHEN loaiphong='A' THEN 100
 WHEN loaiphong='B' THEN 70
ELSE 50
 END
```

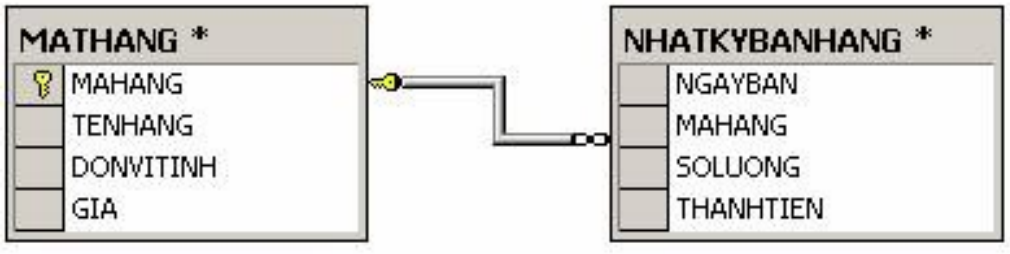
Dữ liệu trong bảng sẽ là:

SOPHONG	LOAIPHONG	SONGAY	TIENPHONG
101	A	5	500
202	B	5	350
101	A	2	200
102	C	3	150

Điều kiện cập nhật dữ liệu liên quan đến nhiều bảng

Mệnh đề FROM trong câu lệnh UPDATE được sử dụng khi cần chỉ định các điều kiện liên quan đến các bảng khác với bảng cần cập nhật dữ liệu. Trong trường hợp này, trong mệnh đề WHERE thường có điều kiện nối giữa các bảng.

Ví dụ 2.53: Giả sử ta có hai bảng MATHANG và NHATKYBANHANG như sau:



Câu lệnh dưới đây sẽ cập nhật giá trị trường THANHTIEN của bảng NHATKYBANHANG theo công thức THANHTIEN = SOLUONG × GIA

```
UPDATE nhatkybanhang
SET  thanhtien = soluong*gia
FROM mathang
```

WHERE nhatkybanhang.mahang = mathang.mahang

Câu lệnh UPDATE với truy vấn con

Tương tự như trong câu lệnh SELECT, truy vấn con có thể được sử dụng trong mệnh đề WHERE của câu lệnh UPDATE nhằm chỉ định điều kiện đối với các dòng dữ liệu cần cập nhật dữ liệu.

Ví dụ 2.54: Câu lệnh ở trên có thể được viết như sau:

```
UPDATE nhatkybanhang
SET  thanhtien = soluong*gia
FROM mathang
WHERE mathang.mahang =(SELECT mathang.mahang
 FROM mathang
 WHERE mathang.mahang=nhatkybanhang.mahang)
```

2.2.3 Xoá dữ liệu

Để xoá dữ liệu trong một bảng, ta sử dụng câu lệnh DELETE. Cú pháp của câu lệnh này như sau:

```
DELETE FROM tên_bảng
[FROM danh_sách_bảng]
[WHERE      điều_kiện]
```

Trong câu lệnh này, tên của bảng cần xoá dữ liệu được chỉ định sau DELETE FROM.

Mệnh đề WHERE trong câu lệnh được sử dụng để chỉ định điều kiện đối với các dòng dữ liệu cần xoá. Nếu câu lệnh DELETE không có mệnh đề WHERE thì toàn bộ các dòng dữ liệu trong bảng đều bị xoá.

Ví dụ 2.55: Câu lệnh dưới đây xoá khỏi bảng SINHVIEN những sinh viên sinh tại Huế

```
DELETE FROM sinhvien
WHERE noisinh LIKE '%Huế%'
```

Xoá dữ liệu khi điều kiện liên quan đến nhiều bảng

Nếu điều kiện trong câu lệnh DELETE liên quan đến các bảng không phải là bảng

cần xóa dữ liệu, ta phải sử dụng thêm mệnh đề FROM và sau đó là danh sách tên các bảng đó. Trong trường hợp này, trong mệnh đề WHERE ta chỉ định thêm điều kiện nối giữa các bảng

Ví dụ 2.56: Câu lệnh dưới đây xoá ra khỏi bảng SINHVIEN những sinh viên lớp Tin K24

```
DELETE FROM sinhvien
FROM lop
WHERE lop.malop=sinhvien.malop AND tenlop='Tin K24'
```

Sử dụng truy vấn con trong câu lệnh DELETE

Một câu lệnh SELECT có thể được lồng vào trong mệnh đề WHERE trong câu lệnh DELETE để làm điều kiện cho câu lệnh tương tự như câu lệnh UPDATE.

Ví dụ 2.57: Câu lệnh dưới đây xoá khỏi bảng LOP những lớp không có sinh viên nào học

```
DELETE FROM lop
WHERE malop NOT IN (SELECT DISTINCT malop
FROM sinhvien)
```

Xoá toàn bộ dữ liệu trong bảng

Câu lệnh DELETE không chỉ định điều kiện đối với các dòng dữ liệu cần xoá trong mệnh đề WHERE sẽ xoá toàn bộ dữ liệu trong bảng. Thay vì sử dụng câu lệnh DELETE trong trường hợp này, ta có thể sử dụng câu lệnh TRUNCATE có cú pháp như sau:

```
TRUNCATE TABLE tên_bảng
```

Ví dụ 2.58: Câu lệnh sau xoá toàn bộ dữ liệu trong bảng diemthi:

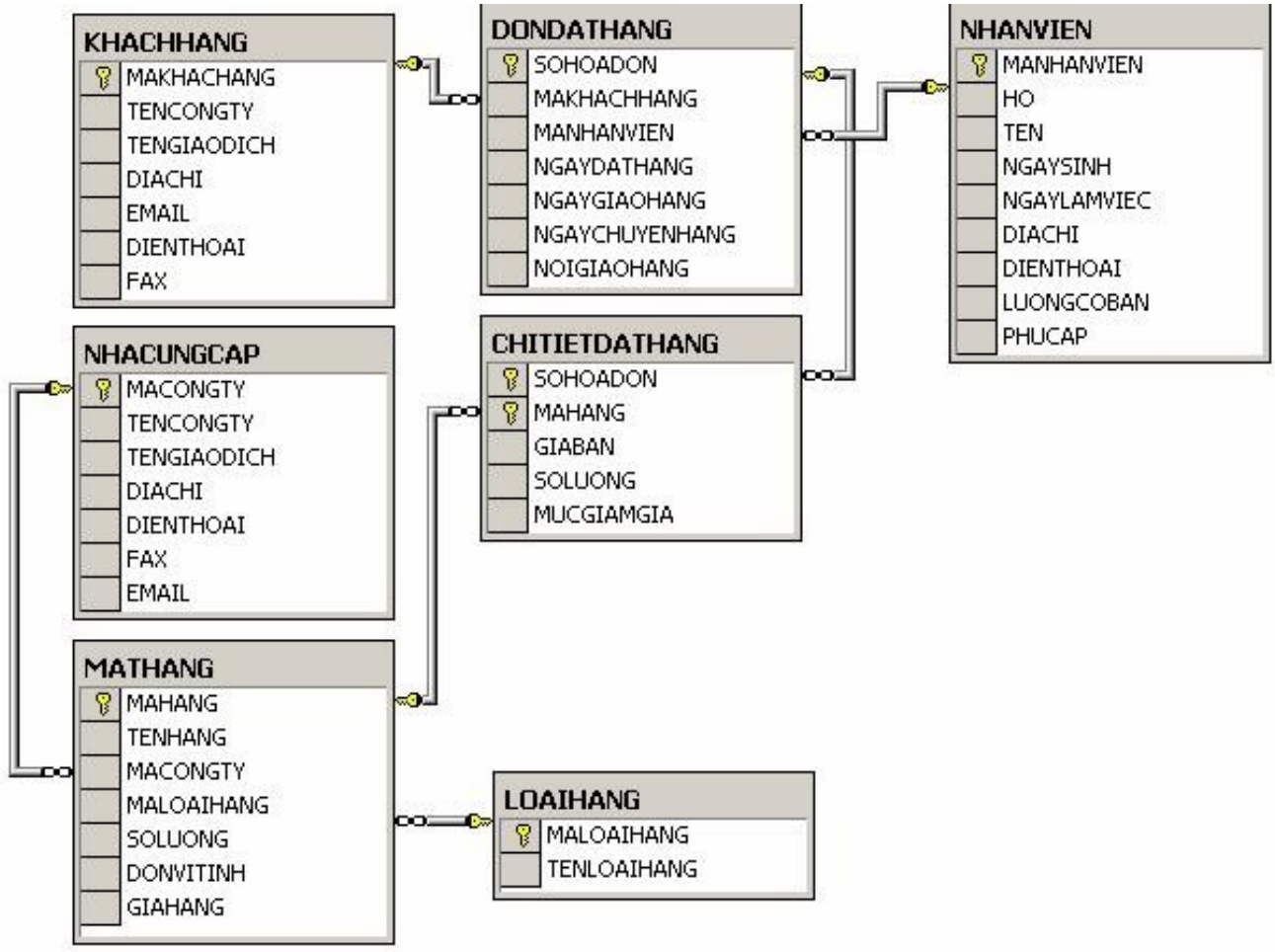
```
DELETE FROM diemthi
```

có tác dụng tương tự với câu lệnh

```
TRUNCATE TABLE diemthi
```

Bài tập chương 2

Cơ sở dữ liệu dưới đây được sử dụng để quản lý công tác giao hàng trong một công ty kinh doanh. Các bảng trong cơ sở dữ liệu này được biểu diễn trong sơ đồ dưới đây:



Trong đó:

Bảng NHACUNGCAP lưu trữ dữ liệu về các đối tác cung cấp hàng cho công ty.

Bảng MATHANG lưu trữ dữ liệu về các mặt hàng hiện có trong công ty.

Bảng LOAIHANG phân loại các mặt hàng hiện có.

Bảng NHANVIEN có dữ liệu là thông tin về các nhân viên làm việc trong công ty.

Bảng KHACHHANG được sử dụng để lưu giữ thông tin về các khách hàng

của công ty.

Khách hàng đặt hàng cho công ty thông qua các đơn đặt hàng. Thông tin chung về các đơn đặt hàng được lưu trữ trong bảng DONDATHANG (Mỗi một đơn đặt hàng phải do một nhân viên của công ty lập và do đó bảng này có quan hệ với bảng NHANVIEN)

Thông tin chi tiết của các đơn đặt hàng (đặt mua mặt hàng gì, số lượng, giá cả,...) được lưu trữ trong bảng CHITIETDATHANG. Bảng này có quan hệ với hai bảng DONDATHANG và MATHANG

Sử dụng câu lệnh SELECT để viết các yêu cầu truy vấn dữ liệu sau đây:

1. Cho biết danh sách các đối tác cung cấp hàng cho công ty.
2. Mã hàng, tên hàng và số lượng của các mặt hàng hiện có trong công ty.
3. Họ tên và địa chỉ và năm bắt đầu làm việc của các nhân viên trong công ty.
4. Địa chỉ và điện thoại của nhà cung cấp có tên giao dịch VINAMILK là gì?
5. Cho biết mã và tên của các mặt hàng có giá lớn hơn 100000 và số lượng hiện có ít hơn 50.
6. Cho biết mỗi mặt hàng trong công ty do ai cung cấp.
7. Công ty Việt Tiến đã cung cấp những mặt hàng nào?
8. Loại hàng thực phẩm do những công ty nào cung cấp và địa chỉ của các công ty đó là gì?
9. Những khách hàng nào (tên giao dịch) đã đặt mua mặt hàng Sữa hộp XYZ của công ty?
10. Đơn đặt hàng số 1 do ai đặt và do nhân viên nào lập, thời gian và địa điểm giao hàng là ở đâu?
11. Hãy cho biết số tiền lương mà công ty phải trả cho mỗi nhân viên là bao nhiêu (lương = lương cơ bản + phụ cấp).
12. Trong đơn đặt hàng số 3 đặt mua những mặt hàng nào và số tiền mà khách hàng phải trả cho mỗi mặt hàng là bao nhiêu (số tiền phải trả được tính theo công thức SOLUONG×GIABAN – SOLUONG×GIABAN×MUCGIAMGIA/100)
13. Hãy cho biết có những khách hàng nào lại chính là đối tác cung cấp hàng của công ty (tức là có cùng tên giao dịch).
14. Trong công ty có những nhân viên nào có cùng ngày sinh?

15. Những đơn đặt hàng nào yêu cầu giao hàng ngay tại công ty đặt hàng và những đơn đó là của công ty nào?
16. Cho biết tên công ty, tên giao dịch, địa chỉ và điện thoại của các khách hàng và các nhà cung cấp hàng cho công ty.
17. Những mặt hàng nào chưa từng được khách hàng đặt mua? Những nhân viên nào của công ty chưa từng lập bất kỳ một hoá đơn đặt hàng nào?
18. Những nhân viên nào của công ty có lương cơ bản cao nhất?
19. Tổng số tiền mà khách hàng phải trả cho mỗi đơn đặt hàng là bao nhiêu?
20. Trong năm 2003, những mặt hàng nào chỉ được đặt mua đúng một lần.
21. Hãy cho biết mỗi một khách hàng đã phải bỏ ra bao nhiêu tiền để đặt mua hàng của công ty?
22. Mỗi một nhân viên của công ty đã lập bao nhiêu đơn đặt hàng (nếu nhân viên chưa hề lập một hoá đơn nào thì cho kết quả là 0)
23. Cho biết tổng số tiền hàng mà cửa hàng thu được trong mỗi tháng của năm 2003 (thời được gian tính theo ngày đặt hàng).
24. Hãy cho biết tổng số tiền lời mà công ty thu được từ mỗi mặt hàng trong năm 2003.
25. Hãy cho biết tổng số lượng hàng của mỗi mặt hàng mà công ty đã có (tổng số lượng hàng hiện có và đã bán).
26. Nhân viên nào của công ty bán được số lượng hàng nhiều nhất và số lượng hàng bán được của những nhân viên này là bao nhiêu?
27. Đơn đặt hàng nào có số lượng hàng được đặt mua ít nhất?
28. Số tiền nhiều nhất mà mỗi khách hàng đã từng bỏ ra để đặt hàng trong các đơn đặt hàng là bao nhiêu?
29. Mỗi một đơn đặt hàng đặt mua những mặt hàng nào và tổng số tiền mà mỗi đơn đặt hàng phải trả là bao nhiêu?
30. Hãy cho biết mỗi một loại hàng bao gồm những mặt hàng nào, tổng số lượng hàng của mỗi loại và tổng số lượng của tất cả các mặt hàng hiện có trong công ty là bao nhiêu?
31. Thống kê xem trong năm 2003, mỗi một mặt hàng trong mỗi tháng và trong cả năm bán được với số lượng bao nhiêu

32. Yêu cầu: Kết quả được hiển thị dưới dạng bảng, hai cột cột đầu là mã hàng và tên hàng, các cột còn lại tương ứng với các tháng từ 1 đến 12 và cả năm. Như vậy mỗi dòng trong kết quả cho biết số lượng hàng bán được mỗi tháng và trong cả năm của mỗi mặt hàng.

Sử dụng câu lệnh UPDATE để thực hiện các yêu cầu sau:

1. Cập nhật lại giá trị trường NGAYCHUYENHANG của những bản ghi có NGAYCHUYENHANG chưa xác định (NULL) trong bảng DONDATHANG bằng với giá trị của trường NGAYDATHANG.
2. Tăng số lượng hàng của những mặt hàng do công ty VINAMILK cung cấp lên gấp đôi.
3. Cập nhật giá trị của trường NOIGIAOHANG trong bảng DONDATHANG bằng địa chỉ của khách hàng đối với những đơn đặt hàng chưa xác định được nơi giao hàng (giá trị trường NOIGIAOHANG bằng NULL).
4. Cập nhật lại dữ liệu trong bảng KHACHHANG sao cho nếu tên công ty và tên giao dịch của khách hàng trùng với tên công ty và tên giao dịch của một nhà cung cấp nào đó thì địa chỉ, điện thoại, fax và e-mail phải giống nhau
5. Tăng lương lên gấp rưỡi cho những nhân viên bán được số lượng hàng nhiều hơn 100 trong năm 2003.
6. Tăng phụ cấp lên bằng 50% lương cho những nhân viên bán được hàng nhiều nhất.
7. Giảm 25% lương của những nhân viên trong năm 2003 không lập được bất kỳ đơn đặt hàng nào.
8. Giả sử trong bảng DONDATHANG có thêm trường SOTIEN cho biết số tiền mà khách hàng phải trả trong mỗi đơn đặt hàng. Hãy tính giá trị cho trường này.

Thực hiện các yêu cầu dưới đây bằng câu lệnh DELETE.

1. Xoá khỏi bảng NHANVIEN những nhân viên đã làm việc trong công ty quá 40 năm.
2. Xoá những đơn đặt hàng trước năm 2000 ra khỏi cơ sở dữ liệu.
3. Xoá khỏi bảng LOAIHANG những loại hàng hiện không có mặt hàng.
4. Xoá khỏi bảng KHACHHANG những khách hàng hiện không có bất kỳ đơn đặt hàng nào cho công ty.

5. Xoá khỏi bảng MATHANG những mặt hàng có số lượng bằng 0 và không được đặt mua trong bất kỳ đơn đặt hàng nào.

Chương 3

NGÔN NGỮ ĐỊNH NGHĨA DỮ LIỆU

Các câu lệnh SQL đã đề cập đến trong chương 3 được sử dụng nhằm thực hiện các thao tác bổ sung, cập nhật, loại bỏ và xem dữ liệu. Nhóm các câu lệnh này được gọi là ngôn ngữ thao tác dữ liệu (DML). Trong chương này, chúng ta sẽ tìm hiểu nhóm các câu lệnh được sử dụng để định nghĩa và quản lý các đối tượng CSDL như bảng, khung nhìn, chỉ mục,... và được gọi là ngôn ngữ định nghĩa dữ liệu (DLL).

Về cơ bản, ngôn ngữ định nghĩa dữ liệu bao gồm các lệnh:

CREATE: định nghĩa và tạo mới đối tượng CSDL.

ALTER: thay đổi định nghĩa của đối tượng CSDL.

DROP: Xoá đối tượng CSDL đã có.

3.1 Tạo bảng dữ liệu

Như đã nói đến ở chương 1, bảng dữ liệu là cấu trúc có vai trò quan trọng nhất trong cơ sở dữ liệu quan hệ. Toàn bộ dữ liệu của cơ sở dữ liệu được tổ chức trong các bảng, những bảng này có thể là những bảng hệ thống được tạo ra khi tạo lập cơ sở dữ liệu, và cũng có thể là những bảng do người sử dụng định nghĩa.

MAKHOA	TENKHOA	DIENTHOAI	TRUONGKHOA
DHT01	Khoa Toán cơ - Tin học	054822407	Trần Lộc Hùng
DHT02	Khoa Công nghệ thông tin	054826767	Nguyễn Mậu Hân
DHT03	Khoa Vật lý	054823462	Trương Văn Chương
DHT04	Khoa Hoá học	054823951	Nguyễn Văn Hợp
DHT05	Khoa Sinh học	054822934	Đỗ Quý Hai
DHT06	Khoa Địa lý - Địa chất	054823837	NULL
DHT07	Khoa Ngữ văn	054821133	Hoàng Tất Thắng
DHT08	Khoa Lịch sử	054823833	Nguyễn Văn Mạnh
DHT09	Khoa Mác - Lê Nin	054825698	Đoàn Đức Hiếu
DHT10	Khoa Luật	054821135	Đoàn Đức Lương

Trong các bảng, dữ liệu được tổ chức dưới dạng các dòng và cột. Mỗi một dòng là một bản ghi duy nhất trong bảng và mỗi một cột là một trường. Các bảng trong cơ sở dữ liệu được sử dụng để biểu diễn thông tin, lưu giữ dữ liệu về các đối tượng trong thế giới thực và/hoặc mối quan hệ giữa các đối tượng. Bảng trong

hình 3.1 bao gồm 10 bản ghi và 4 trường là MAKHOA, TENKHOA, DIENTHOAI và TRUONGKHOA.

Câu lệnh CREATE TABLE được sử dụng để định nghĩa một bảng dữ liệu mới trong cơ sở dữ liệu. Khi định nghĩa một bảng dữ liệu mới, ta cần phải xác định được các yêu cầu sau đây:

Bảng mới được tạo ra sử dụng với mục đích gì và có vai trò như thế nào trong cơ sở dữ liệu.

Cấu trúc của bảng bao gồm những trường (cột) nào, mỗi một trường có ý nghĩa như thế nào trong việc biểu diễn dữ liệu, kiểu dữ liệu của mỗi trường là gì và trường đó có cho phép nhận giá trị NULL hay không.

Những trường nào sẽ tham gia vào khóa chính của bảng. Bảng có quan hệ với những bảng khác hay không và nếu có thì quan hệ như thế nào.

Trên các trường của bảng có tồn tại những ràng buộc về khuôn dạng, điều kiện hợp lệ của dữ liệu hay không; nếu có thì sử dụng ở đâu và như thế nào.

Câu lệnh CREATE TABLE có cú pháp như sau

```
CREATE TABLE    tên_bảng
(
tên_cột   thuộc_tính_cột  các_ràng_buộc
[,...,tên_cột_n thuộc_tính_cột_n  các_ràng_buộc_cột_n]
[,các_ràng_buộc_trên_bảng]
)
```

Trong đó:

tên_bảng	Tên của bảng cần tạo. Tên phải tuân theo qui tắc định danh và không được vượt quá 128 ký tự.
tên_cột	Là tên của cột (trường) cần định nghĩa, tên cột phải tuân theo qui tắc định danh và không được trùng nhau trong mỗi một bảng. Mỗi một bảng phải có ít nhất một cột. Nếu bảng có nhiều cột thì định nghĩa của các cột (tên cột, thuộc tính và các ràng buộc) phải phân cách nhau bởi dấu phẩy
thuộc_tính_cột	Mỗi một cột trong một bảng ngoài tên cột còn có các thuộc

	tính bao gồm: Kiểu dữ liệu của cột. Đây là thuộc tính bắt buộc phải có đối với mỗi cột. Giá trị mặc định của cột: là giá trị được tự động gán cho cột nếu như người sử dụng không nhập dữ liệu cho cột một cách tường minh. Mỗi một cột chỉ có thể có nhiều nhất một giá trị mặc định. Cột có tính chất IDENTITY hay không? tức là giá trị của cột có được tự động tăng mỗi khi có bản ghi mới được bổ sung hay không. Tính chất này chỉ có thể sử dụng đối với các trường kiểu số. Cột có chấp nhận giá trị NULL hay không

Ví dụ 3.1: Khai báo dưới đây định nghĩa cột STT có kiểu dữ liệu là int và cột có tính chất IDENTITY:

stt INT IDENTITY hay định nghĩa cột NGAY có kiểu datetime và không cho phép chấp nhận giá trị NULL: ngay DATETIME NOT NULL và định nghĩa cột SOLUONG kiểu int và có giá trị mặc định là 0:

soluong INT DEFAULT (0)

Các ràng buộc được sử dụng trên mỗi cột hoặc trên bảng nhằm các mục đích sau:

Quy định khuôn dạng hay giá trị dữ liệu được cho phép trên cột (chẳng hạn qui định tuổi của một học sinh phải lớn hơn 6 và nhỏ hơn 20, số điện thoại phải là một chuỗi bao gồm 6 chữ số,...). Những ràng buộc kiểu này được gọi là ràng buộc CHECK

Đảm bảo tính toàn vẹn dữ liệu trong một bảng và toàn vẹn tham chiếu giữa các bảng trong cơ sở dữ liệu. Những loại ràng buộc này nhằm đảm bảo tính đúng của dữ liệu như: số chứng minh nhân dân của mỗi một người phải duy nhất, nếu sinh viên học một lớp nào đó thì lớp đó phải tồn tại,... Liên quan đến những

loại ràng buộc này bao gồm các ràng buộc PRIMARY KEY (khoá chính), UNIQUE (khóa dự tuyển) và FOREIGN KEY (khoá ngoài) Các loại ràng buộc này sẽ được trình bày chi tiết hơn ở phần sau.

Ví dụ 3.2: Câu lệnh dưới đây định nghĩa bảng NHANVIEN với các trường MANV (mã nhân viên), HOTEN (họ và tên), NGAYSINH (ngày sinh của nhân viên), DIENTHOAI (điện thoại) và HSLUONG (hệ số lương)

```
CREATE TABLE nhanvien
(
manv NVARCHAR(10)   NOT NULL,
hoten NVARCHAR(50)   NOT NULL,
ngaysinh DATETIME NULL
dienthoai NVARCHAR(10) NULL,
hsluong DECIMAL(3,2)   DEFAULT (1.92)
)
```

Trong câu lệnh trên, trường MANV và HOTEN của bảng NHANVIEN không được NULL (tức là bắt buộc phải có dữ liệu), trường NGAYSINH và DIENTHOAI sẽ nhận giá trị NULL nếu ta không nhập dữ liệu cho chúng còn trường HSLUONG sẽ nhận giá trị mặc định là 1.92 nếu không được nhập dữ liệu.

Nếu ta thực hiện các câu lệnh dưới đây sau khi thực hiện câu lệnh trên để bổ sung dữ liệu cho bảng NHANVIEN

```
INSERT INTO nhanvien
VALUES('NV01','Le Van A','2/4/75','886963',2.14)
INSERT INTO nhanvien(manv,hoten)
VALUES('NV02','Mai Thi B')
INSERT INTO nhanvien(manv,hoten,dienthoai)
VALUES('NV03','Tran Thi C','849290')
```

Ta sẽ có được dữ liệu trong bảng NHANVIEN như sau:

MANV	HOTEN	NGAYSINH	DIENTHOAI	HSLUONG
NV01	Le Van A	1975-02-04 00:00:00.000	886963	2.14
NV02	Mai Thi B	NULL	NULL	1.92
NV03	Tran Thi C	NULL	849290	1.92

3.1.1 Ràng buộc CHECK

Ràng buộc CHECK được sử dụng nhằm chỉ định điều kiện hợp lệ đối với dữ liệu. Mỗi khi có sự thay đổi dữ liệu trên bảng (INSERT, UPDATE), những ràng buộc này sẽ được sử dụng nhằm kiểm tra xem dữ liệu mới có hợp lệ hay không.

Ràng buộc CHECK được khai báo theo cú pháp như sau:

[CONSTRAINT tên_ràng_buộc]

CHECK (điều_kiện)

Trong đó, điều_kiện là một biểu thức logic tác động lên cột nhằm qui định giá trị hoặc khuôn dạng dữ liệu được cho phép. Trên mỗi một bảng cũng như trên mỗi một cột có thể có nhiều ràng buộc CHECK.

Ví dụ 3.3: Câu lệnh dưới đây tạo bảng DIEMTOTNGHIEP trong đó qui định giá trị của cột DIEMVAN và DIEMTOAN phải lớn hơn hoặc bằng 0 và nhỏ hơn hoặc bằng 10.

```
CREATE TABLE diemtotnghiep
(
hoten NVARCHAR(30) NOT NULL,
ngaysinh DATETIME,
diemvan DECIMAL(4,2)
CONSTRAINT chk_diemvan
CHECK(diemvan>=0 AND diemvan<=10),
diemtoan DECIMAL(4,2)
CONSTRAINT chk_diemtoan
CHECK(diemtoan>=0 AND diemtoan<=10),
)
```

Như vậy, với định nghĩa như trên của bảng DIEMTOTNGHIEP.

các câu lệnh dưới đây là hợp lệ:

```
INSERT INTO diemtotnghiep(hoten,diemvan,diemtoan)
  VALUES('Le Thanh Hoang',9.5,2.5)
INSERT INTO diemtotnghiep(hoten,diemvan)
  VALUES('Hoang Thi Mai',2.5)
```

Còn câu lệnh dưới đây là không hợp lệ:

```
INSERT INTO diemtotnghiep(hoten,diemvan,diemtoan)
  VALUES('Tran Van Hanh',6,10.5)
```

do cột DIEMTOAN nhận giá trị 10.5 không thoả mãn điều kiện của ràng buộc

Trong ví dụ trên, các ràng buôc được chỉ định ở phần khai báo của mỗi cột. Thay vì chỉ định ràng buộc trên mỗi cột, ta có thể chỉ định các ràng buộc ở mức bảng bằng cách khai báo các ràng buộc sau khi đã khai báo xong các cột trong bảng.

Ví dụ 3.4: Câu lệnh CREATE TABLE lop

```
(
malop  NVARCHAR(10)  NOT NULL ,
tenlop NVARCHAR(30)  NOT NULL ,
khoa SMALLINT  NULL ,
hedaotao NVARCHAR(25)  NULL
CONSTRAINT chk_lop_hedaotao
CHECK (hedaotao IN ('chính quy','tại chức')),
namnhaphoc  INT  NULL
CONSTRAINT chk_lop_namnhaphoc
CHECK (namnhaphoc<=YEAR(GETDATE())),
makhoa NVARCHAR(5)
)
```

có thể được viết lại như sau:

```
CREATE TABLE lop
(
malop  NVARCHAR(10)  NOT NULL ,
tenlop NVARCHAR(30)  NOT NULL ,
khoa SMALLINT  NULL ,
hedaotao NVARCHAR(25)  NULL
namnhaphoc INT
makhoa NVARCHAR(5),
CONSTRAINT chk_lop
CHECK (namnhaphoc<=YEAR(GETDATE()) AND
hedaotao IN ('chính quy','tại chức'))
```

)

3.1.2 Ràng buộc PRIMARY KEY

Ràng buộc PRIMARY KEY được sử dụng để định nghĩa khoá chính của bảng. Khoá chính của một bảng là một hoặc một tập nhiều cột mà giá trị của chúng là duy nhất trong bảng. Hay nói cách khác, giá trị của khoá chính sẽ giúp cho ta xác định được duy nhất một dòng (bản ghi) trong bảng dữ liệu. Mỗi một bảng chỉ có thể có duy nhất một khoá chính và bản thân khoá chính không chấp nhận giá trị NULL. Ràng buộc PRIMARY KEY là cơ sở cho việc đảm bảo tính toàn vẹn thực thể cũng như toàn vẹn tham chiếu.

Để khai báo một ràng buộc PRIMARY KEY, ta sử dụng cú pháp như sau:

[CONSTRAINT tên_ràng_buộc]

PRIMARY KEY [(danh_sách_cột)]

Nếu khoá chính của bảng chỉ bao gồm đúng một cột và ràng buộc PRIMARY KEY được chỉ định ở mức cột, ta không cần thiết phải chỉ định danh sách cột sau từ khoá PRIMARY KEY. Tuy nhiên, nếu việc khai báo khoá chính được tiến hành ở mức bảng (sử dụng khi số lượng các cột tham gia vào khoá là từ hai trở lên) thì bắt buộc phải chỉ định danh sách cột ngay sau từ khóa PRIMARY KEY và tên các cột được phân cách nhau bởi dấu phẩy.

Ví dụ 3.5: Câu lệnh dưới đây định nghĩa bảng SINHVIEN với khoá chính là MASV

```
CREATE TABLE sinhvien
(
Masv NVARCHAR(10) CONSTRAINT pk_sinhvien_masv PRIMARY KEY,
hodem NVARCHAR(25)   NOT NULL ,
ten NVARCHAR(10)   NOT NULL ,
ngaysinh  DATETIME,
gioitinh  BIT,
noisinh   NVARCHAR(255),
malop NVARCHAR(10)
)
```

Với bảng vừa được tạo bởi câu lệnh ở trên, nếu ta thực hiện câu lệnh:

```
INSERT INTO sinhvien(masv,hodem,ten,gioitinh,malop)
```

VALUES('0261010001','Lê Hoàng Phương','Anh',0,'C26101')

một bản ghi mới sẽ được bổ sung vào bảng này. Nhưng nếu ta thực hiện tiếp câu lệnh:

INSERT INTO sinhvien(masv,hodem,ten,gioitinh,malop)

VALUES('0261010001','Lê Huy','Đan',1,'C26101')

thì câu lệnh này sẽ bị lỗi do trùng giá trị khoá với bản ghi đã có.

Ví dụ 3.6: Câu lệnh dưới đây tạo bảng DIEMTHI với khoá chính là tập bao gồm hai cột MAMONHOC và MASV

```
CREATE TABLE diemthi
(
Mamonhoc NVARCHAR(10) NOT NULL ,
masv NVARCHAR(10) NOT NULL ,
diemlan1  NUMERIC(4, 2),
diemlan2  NUMERIC(4, 2),
CONSTRAINT pk_diemthi PRIMARY KEY(mamonhoc,masv)
)
```

Lưu ý:

Mỗi một bảng chỉ có thể có nhiều nhất một ràng buộc PRIMARY KEY.

Một khoá chính có thể bao gồm nhiều cột nhưng không vượt quá 16 cột.

3.1.3 Ràng buộc UNIQUE

Trên một bảng chỉ có thể có nhiều nhất một khóa chính nhưng có thể có nhiều cột hoặc tập các cột có tính chất như khoá chính, tức là giá trị của chúng là duy nhất trong bảng. Tập một hoặc nhiều cột có giá trị duy nhất và không được chọn làm khoá chính được gọi là khoá phụ (khoá dự tuyển) của bảng. Như vậy, một bảng chỉ có nhiều nhất một khoá chính nhưng có thể có nhiều khoá phụ.

Ràng buộc UNIQUE được sử dụng trong câu lệnh CREATE TABLE để định nghĩa khoá phụ cho bảng và được khai báo theo cú pháp sau đây:

```
[CONSTRAINT      tên_ràng_buộc]
 UNIQUE [(danh_sách_cột)]
```

Ví dụ 3.7: Giả sử ta cần định nghĩa bảng LOP với khoá chính là cột MALOP nhưng

đồng thời lại không cho phép các lớp khác nhau được trùng tên lớp với nhau, ta sử dụng câu lệnh như sau:

```
CREATE TABLE lop
(
malop NVARCHAR(10)  NOT NULL,
tenlop  NVARCHAR(30)  NOT NULL,
khoa  SMALLINT  NULL,
hedaotao NVARCHAR(25)  NULL,
namnhaphoc  INT
makhoa          NVARCHAR(5),
CONSTRAINT pk_lop PRIMARY KEY (malop),
CONSTRAINT unique_lop_tenlop UNIQUE(tenlop)
)
```

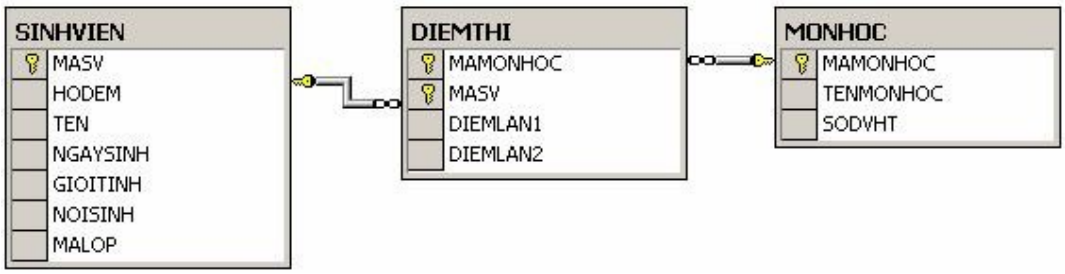
3.1.4 Ràng buộc FOREIGN KEY

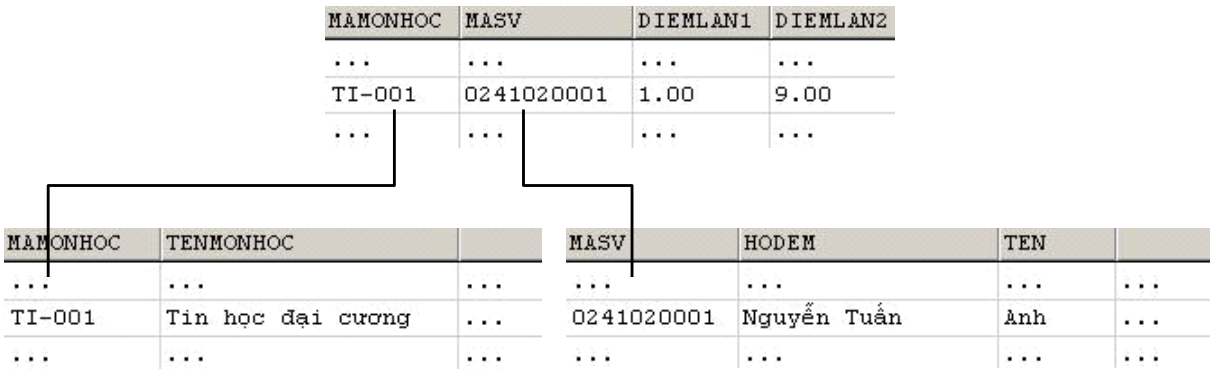
Các bảng trong một cơ sở dữ liệu có mối quan hệ với nhau. Những mối quan hệ này biểu diễn cho sự quan hệ giữa các đối tượng trong thế giới thực. Về mặt dữ liệu, những mối quan hệ được đảm bảo thông qua việc đòi hỏi sự có mặt của một giá trị dữ

liệu trong bảng này phải phụ thuộc vào sự tồn tại của giá trị dữ liệu đó ở trong một bảng khác.

Ràng buộc FOREIGN KEY được sử dụng trong định nghĩa bảng dữ liệu nhằm tạo nên mối quan hệ giữa các bảng trong một cơ sở dữ liệu. Một hay một tập các cột trong một bảng được gọi là khoá ngoại, tức là có ràng buộc FOREIGN KEY, nếu giá trị của nó được xác định từ khoá chính (PRIMARY KEY) hoặc khoá phụ (UNIQUE) của một bảng dữ liệu khác.

Hình dưới đây cho ta thấy được mối quan hệ giữa 3 bảng DIEMTHI, SINHVIEN và MONHOC. Trong bảng DIEMTHI, MASV là khoá ngoài tham chiếu đến cột MASV của bảng SINHVIEN và MAMONHOC là khoá ngoài tham chiếu đến cột MAMONHOC của bảng MONHOC.





MAMONHOC	MASV	DIEMLAN1	DIEMLAN2
...	...	...	...
TI-001	0241020001	1.00	9.00
...	...	...	...

MAMONHOC	TENMONHOC	
...	...	...
TI-001	Tin học đại cương	...
...	...	...

MASV	HODEM	TEN	
...	...	...	...
0241020001	Nguyễn Tuấn	Anh	...
...	...	...	...

Với mối quan hệ được tạo ra như hình trên, hệ quản trị cơ sở dữ liệu sẽ kiểm tra tính hợp lệ của mỗi bản ghi trong bảng DIEMTHI mỗi khi được bổ sung hay cập nhật. Một bản ghi bất kỳ trong bảng DIEMTHI chỉ hợp lệ (đảm bảo ràng buộc FOREIGN KEY) nếu giá trị của cột MASV phải tồn tại trong một bản ghi nào đó của bảng SINHVIEN và giá trị của cột MAMONHOC phải tồn tại trong một bản ghi nào đó của bảng MONHOC.

Ràng buộc FOREIGN KEY được định nghĩa theo cú pháp dưới đây:

[CONSTRAINT tên_ràng_buộc]
FOREIGN KEY [(danh_sách_cột)]
REFERENCES tên_bảng_tham_chiếu(danh_sách_cột_tham_chiếu)
[ON DELETE CASCADE | NO ACTION | SET NULL | SET DEFAULT]
[ON UPDATE CASCADE | NO ACTION | SET NULL | SET DEFAULT]

Việc định nghĩa một ràng buộc FOREIGN KEY bao gồm các yếu tố sau:

Tên cột hoặc danh sách cột của bảng được định nghĩa tham gia vào khoá ngoài.

Tên của bảng được tham chiếu bởi khoá ngoài và danh sách các cột được tham chiếu đến trong bảng tham chiếu.

Cách thức xử lý đối với các bản ghi trong bảng được định nghĩa trong trường

hợp các bản ghi được tham chiếu trong bảng tham chiếu bị xoá (ON DELETE) hay cập nhật (ON UPDATE). SQL chuẩn đưa ra 4 cách xử lý:

CASCADE: Tự động xoá (cập nhật) nếu bản ghi được tham chiếu bị xoá (cập nhật).

NO ACTION: (Mặcđịnh) Nếu bản ghi trong bảng tham chiếu đangđược tham chiếu bởi một bản ghi bất kỳ trong bảng được định nghĩa thì bàn ghi đó không được phép xoá hoặc cập nhật (đối với cột được tham chiếu).

SET NULL: Cập nhật lại khoá ngoài của bản ghi thành giá trị NULL (nếu cột cho phép nhận giá trị NULL).

SET DEFAULT: Cập nhật lại khoá ngoài của bản ghi nhận giá trị mặc định (nếu cột có qui định giá trị mặc định).

Ví dụ 3.8: Câu lệnh dưới đây định nghĩa bảng DIEMTHI với hai khoá ngoài trên cột MASV và cột MAMONHOC (giả sử hai bảng SINHVIEN và MONHOC đã được định nghĩa)

```
CREATE TABLE diemthi
(
mamonhoc NVARCHAR(10) NOT NULL ,
masv NVARCHAR(10) NOT NULL ,
diemlan1  NUMERIC(4, 2),
diemlan2  NUMERIC(4, 2),
CONSTRAINT pk_diemthi PRIMARY KEY(mamonhoc,masv),
CONSTRAINT fk_diemthi_mamonhoc
FOREIGN KEY(mamonhoc)
REFERENCES monhoc(mamonhoc)
ON DELETE CASCADE
ON UPDATE CASCADE,
CONSTRAINT fk_diemthi_masv
FOREIGN KEY(masv)
REFERENCES sinhvien(masv)
ON DELETE CASCADE
ON UPDATE CASCADE
```

)

Lưu ý:

Cột được tham chiếu trong bảng tham chiếu phải là khoá chính (hoặc là khoá phụ).

Cột được tham chiếu phải có cùng kiểu dữ liệu và độ dài với cột tương ứng trong khóa ngoài.

Bảng tham chiếu phải được định nghĩa trước. Do đó, nếu các bảng có mối quan hệ vòng, ta có thể không thể định nghĩa ràng buộc FOREIGN KEY ngay trong câu lệnh CREATE TABLE mà phải định nghĩa thông qua lệnh ALTER TABLE.

3.2 Sửa đổi định nghĩa bảng

Một bảng sau khi đã được định nghĩa bằng câu lệnh CREATE TABLE có thể được sửa đổi thông qua câu lệnh ALTER TABLE. Câu lệnh này cho phép chúng ta thực hiện được các thao tác sau:

Bổ sung một cột vào bảng.

Xoá một cột khỏi bảng.

Thay đổi định nghĩa của một cột trong bảng.

Xoá bỏ hoặc bổ sung các ràng buộc cho bảng

Cú pháp của câu lệnh ALTER TABLE như sau:

```
ALTER TABLE      tên_bảng
 ADD          định_nghĩa_cột |
 ALTER COLUMNtên_cột  kiểu_dữ_liêu [NULL | NOT NULL] |
 DROP COLUMN  tên_cột |
 ADD CONSTRAINT       tên_ràng_buộc  định_nghĩa_ràng_buộc |
 DROP CONSTRAINT     tên_ràng_buộc
```

Ví dụ 3.9: Các ví dụ dưới đây minh hoạ cho ta cách sử dụng câu lệnh ALTER TABLE trong các trường hợp.

Giả sử ta có hai bảng DONVI và NHANVIEN với định nghĩa như sau:

```
CREATE TABLE donvi
(
```

```
madv INT NOT NULL PRIMARY KEY,
tendv  NVARCHAR(30)   NOT NULL
)
CREATE TABLE nhanvien
(
manv NVARCHAR(10)  NOT NULL,
hoten NVARCHAR(30)   NOT NULL,
ngaysinh DATETIME,
diachi CHAR(30)  NOT NULL
)
```

Bổ sung vào bảng NHANVIEN cột DIENTHOAI với ràng buộc CHECK nhằm qui định điện thoại của nhân viên là một chuỗi 6 chữ số:

```
ALTER TABLE nhanvien
ADD
  dienthoai NVARCHAR(6)
  CONSTRAINT chk_nhanvien_dienthoai
  CHECK (dienthoai LIKE '[0-9][0-9][0-9][0-9][0-9][0-9]')
```

Bổ sung thêm cột MADV vào bảng NHANVIEN:

```
ALTER TABLE nhanvien
ADD
madv INT NULL
```

Định nghĩa lại kiểu dữ liệu của cột DIACHI trong bảng NHANVIEN và cho phép cột

này chấp nhận giá trị NULL:

```
ALTER TABLE nhanvien
ALTER COLUMN diachi NVARCHAR(100) NULL
```

Xoá cột ngày sinh khỏi bảng NHANVIEN:

```
ALTER TABLE nhanvien
DROP COLUMN ngaysinh
```

Định nghĩa khoá chính (ràng buộc PRIMARY KEY) cho bảng NHANVIEN là cột MANV:

ALTER TABLE nhanvien

ADD

CONSTRAINT pk_nhanvien PRIMARY KEY(manv)

Định nghĩa khoá ngoài cho bảng NHANVIEN trên cột MADV tham chiếu đến cột MADV của bảng DONVI:

ALTER TABLE nhanvien

ADD

CONSTRAINT fk_nhanvien_madv

FOREIGN KEY(madv) REFERENCES donvi(madv)

ON DELETE CASCADE

ON UPDATE CASCADE

Xoá bỏ ràng buộc kiểm tra số điện thoại của nhân viên

ALTER TABLE nhanvien

DROP CONSTRAINT CHK_NHANVIEN_DIENTHOAI

Lưu ý:

Nếu bổ sung thêm một cột vào bảng và trong bảng đã có ít nhất một bản ghi thì cột mới cần bổ sung phải cho phép chấp nhận giá trị NULL hoặc phải có giá trị mặc định.

Muốn xoá một cột đang được ràng buộc bởi một ràng buộc hoặc đang được tham chiếu bởi một khoá ngoài, ta phải xoá ràng buộc hoặc khoá ngoài trước sao cho trên cột không còn bất kỳ một ràng buộc và không còn được tham chiếu bởi bất kỳ khoá ngoài nào.

Nếu bổ sung thêm ràng buộc cho một bảng đã có dữ liệu và ràng buộc cần bổ sung không được thoả mãn bởi các bản ghi đã có trong bảng thì câu lệnh ALTER TABLE không thực hiện được.

3.3 Xoá bảng

Khi một bảng không còn cần thiết , ta có thể xoá nó ra khỏi cơ sở dữ liệu bằng câu lệnh DROP TABLE. Câu lệnh này cũng đồng thời xoá tất cả những ràng buộc, chỉ mục, trigger liên quan đến bảng đó.

Câu lệnh có cú pháp như sau:

DROP TABLE tên_bảng

Trong các hệ quản trị cơ sở dữ liệu, khi đã xoá một bảng bằng lệnh DROP TABLE, ta không thể khôi phục lại bảng cũng như dữ liệu của nó. Do đó, cần phải cẩn thận khi sử dụng câu lệnh này.

Câu lệnh DROP TABLE không thể thực hiện được nếu bảng cần xoá đang được tham chiếu bởi một ràng buộc FOREIGN KEY. Trong trường hợp này, ràng buộc FOREIGN KEY đang tham chiếu hoặc bảng đang tham chiếu đến bảng cần xoá phải được xoá trước.

Khi một bảng bị xoá, tất cả các ràng buộc, chỉ mục và trigger liên quan đến bảng cũng đồng thời bị xóa theo. Do đó, nếu ta tạo lại bảng thì cũng phải tạo lại các đối tượng này.

Ví dụ 3.10: Giả sử cột MADV trong bảng DONVI đang được tham chiếu bởi khoá ngoài fk_nhanvien_madv trong bảng NHANVIEN. Để xoá bảng DONVI ra khỏi cơ sở dữ liệu, ta thực hiện hai câu lệnh sau:

Xoá bỏ ràng buộc fk_nhanvien_madv khỏi bảng NHANVIEN:

ALTER TABLE nhanvien

DROP CONSTRAINT fk_nhanvien_madv

Xoá bảng DONVI:

DROP TABLE donvi

3.4 Khung nhìn

Các bảng trong cơ sở dữ liệu đóng vai trò là các đối tượng tổ chức và lưu trữ dữ liệu. Như vậy, ta có thể quan sát được dữ liệu trong cơ sở dữ liệu bằng cách thực hiện các truy vấn trên bảng dữ liệu. Ngoài ra, SQL còn cho phép chúng ta quan sát được dữ liệu thông qua việc định nghĩa các khung nhìn.

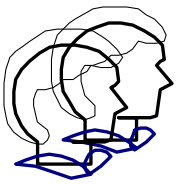
Một khung nhìn (view) có thể được xem như là một bảng "ảo" trong cơ sở dữ liệu có nội dung được định nghĩa thông qua một truy vấn (câu lệnh SELECT). Như vậy, một khung nhìn trông giống như một bảng với một tên khung nhìn và là một tập bao gồm các dòng và các cột. Điểm khác biệt giữa khung nhìn và bảng là khung nhìn không được xem là một cấu trúc lưu trữ dữ liệu tồn tại trong cơ sở dữ liệu. Thực chất dữ liệu quan sát được trong khung nhìn được lấy từ các bảng thông qua câu lệnh truy vấn dữ liệu.

Hình 3.3 dưới đây minh hoạ cho ta thấy khung nhìn có tên DSSV được định nghĩa thông qua câu lệnh SELECT truy vấn dữ liệu trên hai bảng SINHVIEN và LOP:

```
SELECT masv,hodem,ten, DATEDIFF(YY,ngaysinh,GETDATE()) AS tuoi,tenlop
FROM sinhvien,lop
WHERE sinhvien.malop=lop.malop
```

MASV	HODEM	TEN	NGAYSINH
0241010001	Ngô Thị Nhật	Anh	Nov 27 1982 ...
0241010002	Nguyễn Thị Ngọc	Anh	Mar 21 1983 ...
0241010003	Ngô Việt	Bắc	May 11 1982 ...
0241010004	Nguyễn Đình	Bình	Oct 6 1982 ...
0241010005	Hồ Đăng	Chiến	Jan 20 1982 ...
0241020001	Nguyễn Tuấn	Anh	Jul 15 1979 ...
0241020002	Trần Thị Kim	Anh	Nov 4 1982 ...
...	...	...	...

MALOP	TENLOP	
C24101	Toán K24	...
C24102	Tin K24	...
C24103	Lý K24	...
..	...	....



MASV	HODEM	TEN	TUOI	TENLOP
0241010001	Ngô Thị Nhật	Anh	22	Toán K24
0241010002	Nguyễn Thị Ngọc	Anh	21	Toán K24
0241010003	Ngô Việt	Bắc	22	Toán K24
0241010004	Nguyễn Đình	Bình	22	Toán K24
0241010005	Hồ Đăng	Chiến	22	Toán K24
0241020001	Nguyễn Tuấn	Anh	25	Tin K24
0241020002	Trần Thị Kim	Anh	22	Tin K24
...	...	...	...	...

Hình 3.3 *Khung nhìn DSSV với dữ liệu được lấy từ bảng SINHVIEN và LOP*

Khi khung nhìn DSSV đã được định nghĩa, ta có thể sử dụng câu lệnh SELECT để truy vấn dữ liệu từ khung nhìn như đối với các bảng. Khi trong câu truy vấn xuất hiện khung nhìn, hệ quản trị cơ sở dữ liệu sẽ dựa vào đinh nghĩa của khung nhìn để chuyển yêu cầu truy vấn dữ liệu liên quan đến khung nhìn thành yêu cầu tương tự trên các bảng cơ sở và việc truy vấn dữ liệu được thực hiện bởi yêu cầu tương đương trên các bảng.

Việc sử dụng khung nhìn trong cơ sở dữ liệu đem lại các lợi ích sau đây:

Bảo mật dữ liệu: Người sử dụng được cấp phát quyền trên các khung nhìn

với những phần dữ liệu mà người sử dụng được phép. Điều này hạn chế được phần nào việc người sử dụng truy cập trực tiếp dữ liệu.

Đơn giản hoá các thao tác truy vấn dữ liệu: Một khung nhìn đóng vai trò như là một đối tượng tập hợp dữ liệu từ nhiều bảng khác nhau vào trong một "bảng". Nhờ vào đó, người sử dụng có thể thực hiện các yêu cầu truy vấn dữ liệu một cách đơn giản từ khung nhìn thay vì phải đưa ra những câu truy vấn phức tạp.

Tập trung và đơn giản hoà dữ liệu: Thông qua khung nhìn ta có thể cung cấp cho người sử dụng những cấu trúc đơn giản, dễ hiểu hơn về dữ liệu trong cơ sở dữ liệu đồng thời giúp cho người sử dụng tập trung hơn trên những phần dữ liệu cần thiết.

Độc lập dữ liệu: Một khung nhìn có thể cho phép người sử dụng có được cái nhìn về dữ liệu độc lập với cấu trúc của các bảng trong cơ sở dữ liệu cho dù các bảng cơ sở có bị thay đổi phần nào về cấu trúc

Tuy nhiên, việc sử dụng khung nhìn cũng tồn tại một số nhược điểm sau:

Do hệ quản trị cơ sở dữ liệu thực hiện việc chuyển đổi các truy vấn trên khung nhìn thành những truy vấn trên các bảng cơ sở nên nếu một khung nhìn được định nghĩa bởi một truy vấn phức tạp thì sẽ dẫn đến chi phí về mặt thời gian khi thực hiện truy vấn liên quan đến khung nhìn sẽ lớn.

Mặc dù thông qua khung nhìn có thể thực hiện được thao tác bổ sung và cập nhật dữ liệu cho bảng cơ sở nhưng chỉ hạn chế đối với những khung nhìn đơn giản. Đối với những khung nhìn phức tạp thì thường không thực hiện được; hay nói cách khác là dữ liệu trong khung nhìn là chỉ đọc.

3.4.1 Tạo khung nhìn

Câu lệnh CREATE VIEW được sử dụng để tạo ra khung nhìn và có cú pháp như sau:

CREATE VIEW tên_khung_nhìn[(danh_sách_tên_cột)]

AS

câu_lệnh_SELECT

Ví dụ 3.11: Câu lệnh dưới đây tạo khung nhìn có tên DSSV từ câu lệnh SELECT truy vấn dữ liệu từ hai bảng SINHVIEN và LOP

```
CREATE VIEW dssv
AS
SELECT masv,hodem,ten,
  DATEDIFF(YY,ngaysinh,GETDATE()) AS tuoi,tenlop
FROM sinhvien,lop
WHERE sinhvien.malop=lop.malop
```

và nếu thực hiện câu lệnh:

```
SELECT * FROM dssv
```

ta có được kết quả như sau:

MASV	HODEM	TEN	TUOI	TENLOP
0241010001	Ngô Thị Nhật	Anh	22	Toán K24
0241010002	Nguyễn Thị Ngọc	Anh	21	Toán K24
0241010003	Ngô Việt	Bắc	22	Toán K24
0241010004	Nguyễn Đình	Bình	22	Toán K24
0241010005	Hồ Đăng	Chiến	22	Toán K24
0241020001	Nguyễn Tuấn	Anh	25	Tin K24
0241020002	Trần Thị Kim	Anh	22	Tin K24
0241020003	Võ Đức	Ân	22	Tin K24
0241020004	Nguyễn Công	Bình	25	Tin K24
0241020005	Nguyễn Thanh	Bình	22	Tin K24
...	...	...	...	...

Nếu trong câu lệnh CREATE VIEW, ta không chỉ định danh sách các tên cột cho khung nhìn, tên các cột trong khung nhìn sẽ chính là tiêu đề các cột trong kết quả của câu lệnh SELECT. Trong trường hợp tên các cột của khung nhìn được chỉ định, chúng phải có cùng số lượng với số lượng cột trong kết quả của câu truy vấn.

Ví dụ 3.12: Câu lệnh dưới đây tạo khung nhìn từ câu truy vấn tương tự như ví dụ trên

nhưng có đặt tên cho các cột trong khung nhìn:

```
CREATE VIEW dssv(ma,ho,ten,tuoi,lop)
AS
SELECT  masv,hodem,ten,  DATEDIFF(YY,ngaysinh,GETDATE()),tenlop
FROM sinhvien,lop
WHERE sinhvien.malop=lop.malop
```

và câu lệnh:

SELECT * FROM dssv

MASV	HODEM	TEN	TUOI	TENLOP
0241010001	Ngô Thị Nhật	Anh	22	Toán K24
0241010002	Nguyễn Thị Ngọc	Anh	21	Toán K24
0241010003	Ngô Việt	Bắc	22	Toán K24
0241010004	Nguyễn Đình	Bình	22	Toán K24
0241010005	Hồ Đăng	Chiến	22	Toán K24
0241020001	Nguyễn Tuấn	Anh	25	Tin K24
0241020002	Trần Thị Kim	Anh	22	Tin K24
0241020003	Võ Đức	Ân	22	Tin K24
0241020004	Nguyễn Công	Bình	25	Tin K24
0241020005	Nguyễn Thanh	Bình	22	Tin K24
...	...	...	...	...

Khi tạo khung nhìn với câu lệnh CREATE VIEW, ta cần phải lưu ý một số nguyên tắc sau:

Tên khung nhìn và tên cột trong khung nhìn, cũng giống như bảng, phải tuân theo qui tắc định danh

Không thể qui định ràng buộc và tạo chỉ mục cho khung nhìn

Câu lệnh SELECT với mệnh đề COMPUTE ... BY không được sử dụng để định nghĩa khung nhìn.

Phải đặt tên cho các cột của khung nhìn trong các trường hợp sau đây:

m Trong kết quả của câu lệnh SELECT có ít nhất một cột được sinh ra bởi một biểu thức (tức là không phải là một tên cột trong bảng cơ sở) và cột đó không được đặt tiêu đề.

m Tồn tại hai cột trong kết quả của câu lệnh SELECT có cùng tiêu đề cột.

Ví dụ 3.13: Câu lệnh dưới đây là câu lệnh sai do cột thứ 4 không xác định được tên cột

```
CREATE VIEW tuoisinhvien
AS
SELECT masv,hodem,ten,DATEDIFF(YY,ngaysinh,GETDATE())
FROM sinhvien
```

3.4.2 Cập nhật, bổ sung và xoá dữ liệu thông qua khung nhìn

Đối với một số khung nhìn, ta có thể tiến hành thực hiện các thao tác cập nhập, bổ sung và xoá dữ liệu. Thực chất, những thao tác này sẽ được chuyển thành những thao

tác tương tự trên các bảng cơ sở và có tác động đến những bảng cơ sở.

Về mặt lý thuyết, để có thể thực hiện thao tác bổ sung, cập nhật và xoá, một khung nhìn trước tiên phải thoả mãn các điều kiện sau đây:

Trong câu lệnh SELECT định nghĩa khung nhìn không được sử dụng từ khoá DISTINCT, TOP, GROUP BY và UNION.

Các thành phần xuất hiện trong danh sách chọn của câu lệnh SELECT phải là các cột trong các bảng cơ sở. Trong danh sách chọn không được chứa các biểu thức tính toán, các hàm gộp.

Ngoài những điều kiện trên, các thao tác thay đổi đến dữ liệu thông qua khung nhìn còn phải đảm bảo thoả mãn các ràng buộc trên các bảng cơ sở, tức là vẫn đảm bảo tính toàn vẹn dữ liệu. Ví dụ dưới đây sẽ minh hoạ cho ta thấy việc thực hiện các thao tác bổ sung, cập nhật và xoá dữ liệu thông qua khung nhìn.

Ví dụ 3.14: Xét định nghĩa hai bảng DONVI và NHANVIEN như sau:

```
CREATE TABLE donvi
(
madv  INT  PRIMARY KEY,
endv NVARCHAR(30) NOT NULL,
dienthoai NVARCHAR(10) NULL,
)
CREATE TABLE  nhanvien
(
manv NVARCHAR(10)  PRIMARY KEY,
hoten   NVARCHAR(30)   NOT NULL,
ngaysinh        DATETIME          NULL,
diachi  NVARCHAR(50)    NULL,
madv INT FOREIGN KEY
ON DELETE CASCADE
ON UPDATE CASCADE
REFERENCES donvi(madv)
)
```

Giả sử trong hai bảng này đã có dữ liệu như sau:

MADV	TENDV	DIENTHOAI
1	P. Kinh doanh	822321
2	P. Tiep thi	822012

Bảng DONVI

MANV	HOTEN	NGAYSINH	DIACHI	MADV
NV01	Tran Van A	1975-02-03 00:00:00	77 Tran Phu	1
NV02	Mai Thi B	1977-05-04 00:00:00	34 Nguyen Hue	2
NV03	Nguyen Van C	NULL	NULL	2

Bảng NHANVIEN

Câu lệnh dưới đây định nghĩa khung nhìn NV1 cung cấp các thông tin về mã nhân viên, họ tên và mã đơn vị nhân viên làm việc:

CREATE VIEW nv1

AS

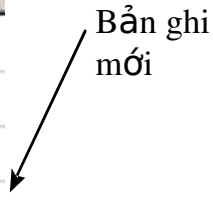
SELECT manv,hoten,madv FROM nhanvien

Nếu ta thực hiện câu lệnh

INSERT INTO nv1 VALUES('NV04','Le Thi D',1)

Một bản ghi mới sẽ được bổ sung vào bảng NHANVIEN và dữ liệu trong bảng này sẽ là:

MANV	HOTEN	NGAYSINH	DIACHI	MADV
NV01	Tran Van A	1975-02-03 00:00:00	77 Tran Phu	1
NV02	Mai Thi B	1977-05-04 00:00:00	34 Nguyen Hue	2
NV03	Nguyen Van C	NULL	NULL	2
NV04	Le Thi D	NULL	NULL	1



Thông qua khung nhìn này, ta cũng có thể thực hiện thao tác cập nhật và xoá dữ liệu.

Chẳng hạn, nếu ta thực hiện câu lệnh:

DELETE FROM nv1 WHERE manv='NV04'

Thì bản ghi tương ứng với nhân viên có mã NV04 sẽ bị xoá khỏi bảng NHANVIEN

Nếu trong danh sách chọn của câu lệnh SELECT có sự xuất hiện của biểu thức tính toán đơn giản, thao tác bổ sung dữ liệu thông qua khung nhìn không thể thực hiện được. Tuy nhiên, trong trường hợp này thao tác cập nhật và xoá dữ liệu vẫn có thể có khả năng thực hiện được (hiển nhiên không thể cập nhật dữ liệu đối với một cột có được từ một biểu thức tính toán).

Ví dụ 3.15: Xét khung nhìn NV2 được định nghĩa như sau:

```
CREATE VIEW nv2
AS
SELECT manv,hoten,YEAR(ngaysinh) AS namsinh,madv
FROM nhanvien
```

Đối với khung nhìn NV2, ta không thể thực hiện thao tác bổ sung dữ liệu nhưng có thể cập nhật hoặc xoá dữ liệu trên bảng thông qua khung nhìn này. Câu lệnh dưới đây là không thể thực hiện được trên khung nhìn NV2

```
INSERT INTO nv2(manv,hoten,madv)
VALUES('NV05','Le Van E',1)
```

Nhưng câu lệnh:

```
UPDATE nv2 SET hoten='Le Thi X' WHERE manv='NV04'
```

hoặc câu lệnh

```
DELETE FROM nv2 WHERE manv='NV04'
```

lại có thể thực hiện được và có tác động đối với dữ liệu trong bảng NHANVIEN

Trong trường hợp khung nhìn được tạo ra từ một phép nối (trong hoặc ngoài) trên nhiều bảng, ta có thể thực hiện được thao tác bổ sung hoặc cập nhật dữ liệu nếu thao tác này chỉ có tác động đến đúng một bảng cơ sở (câu lệnh DELETE không thể thực hiện được trong trường hợp này).

Ví dụ 3.16: Với khung nhìn được định nghĩa như sau:

```
CREATE VIEW nv3
AS
SELECT manv,hoten,ngaysinh, diachi,nhanvien.madv AS noilamviec,  donvi.madv, tendv, dienthoai
FROM nhanvien FULL OUTER JOIN donvi  ON nhanvien.madv=donvi.madv
```

Câu lệnh:

INSERT INTO nv3(manv,hoten,noilamviec)

VALUES('NV05','Le Van E',1)

sẽ bổ sung thêm vào bảng NHANVIEN một bản ghi mới.

Hoặc câu lệnh:

INSERT INTO nv3(madv,tendv) VALUES(3,'P. Ke toan')

bổ sung thêm vào bảng DONVI một bản ghi do cả hai câu lệnh này chỉ có tác động đến đúng một bảng cơ sở.

Câu lệnh dưới đây không thể thực hiện được do có tác động một lúc đến hai bảng cơ sở.

INSERT INTO nv3(manv,hoten,noilamviec,madv,tendv)

VALUES('NV05','Le Van E',1,3,'P. Ke toan')

3.4.3 Sửa đổi khung nhìn

Câu lệnh ALTER VIEW được sử dụng để định nghĩa lại khung nhìn hiện có nhưng không làm thay đổi các quyền đã được cấp phát cho người sử dụng trước đó.

Câu lệnh này sử dụng tương tự như câu lệnh CREATE VIEW và có cú pháp như sau:

ALTER VIEW tên_khung_nhìn [(danh_sách_tên_cột)]

AS

Câu_lệnh_SELECT

Ví dụ 3.17: Ta định nghĩa khung nhìn như sau:

CREATE VIEW viewlop

AS

SELECT malop,tenlop,tenkhoa

FROM lop INNER JOIN khoa ON lop.makhoa=khoa.makhoa

WHERE tenkhoa='Khoa Vật lý'

và có thể định nghĩa lại khung nhìn trên bằng câu lệnh:

ALTER VIEW view_lop

AS

SELECT malop,tenlop,hedaotao

FROM lop INNER JOIN khoa ON lop.makhoa=khoa.makhoa

WHERE tenkhoa='Khoa Công nghệ thông tin'

3.4.4 Xoá khung nhìn

Khi một khung nhìn không còn sử dụng, ta có thể xoá nó ra khỏi cơ sở dữ liệu thông qua câu lệnh:

DROP VIEWtên_khung_nhìn

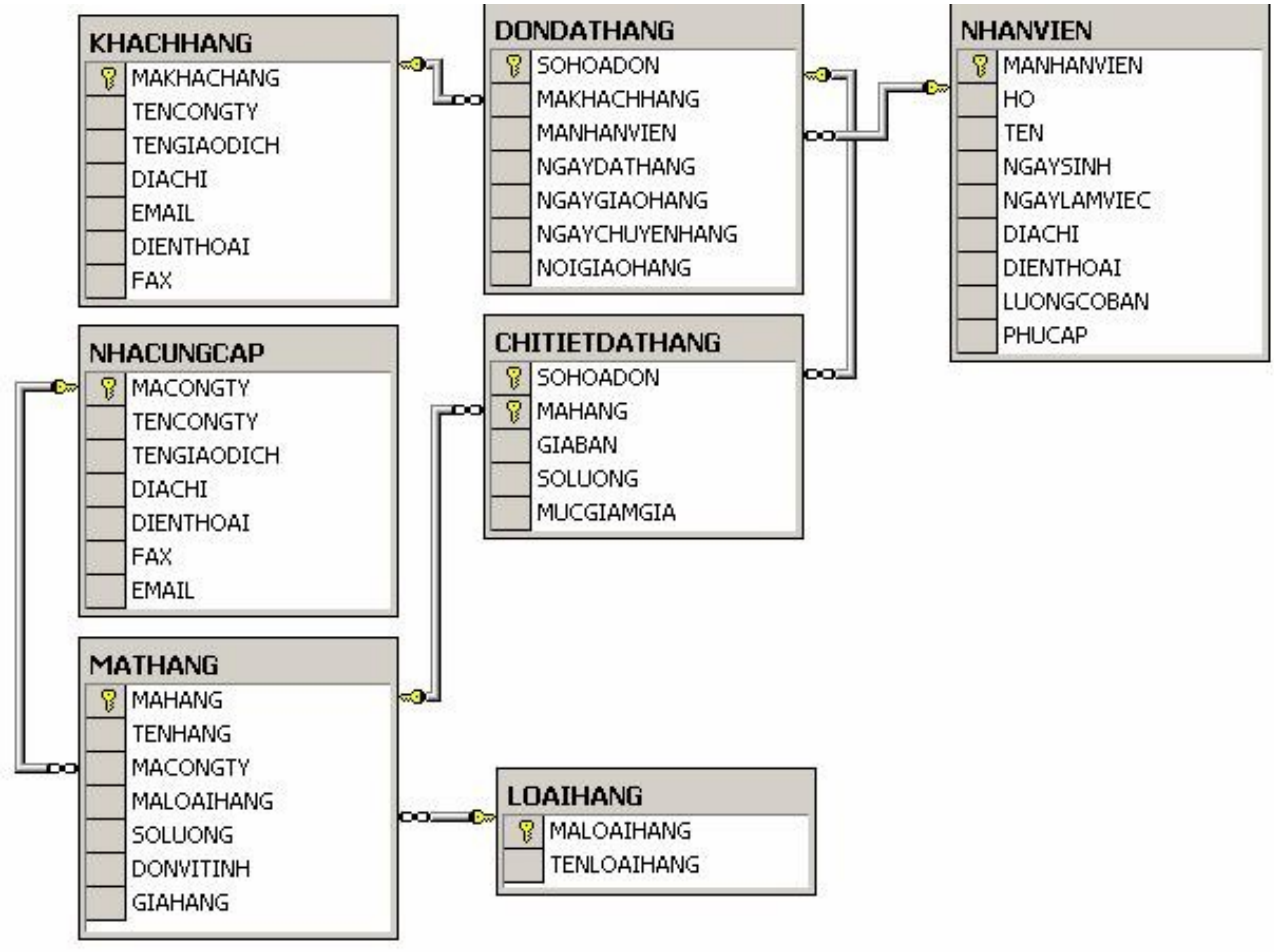
Nếu một khung nhìn bị xoá, toàn bộ những quyền đã cấp phát cho người sử dụng trên khung nhìn cũng đồng thời bị xoá. Do đó, nếu ta tạo lại khung nhìn thì phải tiến hành cấp phát lại quyền cho người sử dụng.

Ví dụ 3.18: Câu lệnh dưới đây xoá khung nhìn VIEW_LOP ra khỏi cơ sở dữ liệu

DROP VIEW view_lop

Bài tập chương 3

1. Sử dụng câu lệnh CREATE TABLE để tạo các bảng trong cơ sở dữ liệu như sơ đồ dưới đây (bạn tự lựa chọn kiểu dữ liệu cho phù hợp)



2. Bổ sung ràng buộc thiết lập giá trị mặc định bằng 1 cho cột SOLUONG và bằng 0 cho cột MUCGIAMGIA trong bảng CHITIETDATHANG
3. Bổ sung cho bảng DONDATHANG ràng buộc kiểm tra ngày giao hàng và ngày chuyển hàng phải sau hoặc bằng với ngày đặt hàng.
4. Bổ sung ràng buộc cho bảng NHANVIEN để đảm bảo rằng một nhân viên chỉ có thể làm việc trong công ty khi đủ 18 tuổi và không quá 60 tuổi.
5. Với các bảng đã tạo được, câu lệnh:

   DROP TABLE nhacungcap

   có thể thực hiện được không? Tại sao?
6. Cho khung nhìn được định nghĩa như sau:

   CREATE VIEW view_donhang

AS

SELECT dondathang.sohoadon, makhachhang, manhanvien, ngaydathang, ngaygiaohang, ngaychuyenhang, noigiaohang,mahang, giaban,soluong,mucgiamgia

FROM dondathang INNER JOIN chitietdathang

ON dondathang.sohoadon = chitietdathang.sohoadon

a. Có thể thông qua khung nhìn này để bổ sung dữ liệu cho bảng DONDATHANG được không?

b. Có thể thông qua khung nhìn này để bổ sung dữ liệu cho bảng CHITIETDATHANG được không?

7. Với khung nhìn được định nghĩa như sau:

CREATE VIEW view_donhang

AS

SELECT dondathang.sohoadon, makhachhang, manhanvien, ngaydathang, ngaygiaohang, ngaychuyenhang, noigiaohang, mahang, giaban * soluong as thanhtien, mucgiamgia

FROM dondathang INNER JOIN chitietdathang

ON dondathang.sohoadon = chitietdathang.sohoadon

a. Có thể thông qua khung nhìn này để xoá hay cập nhật dữ liệu trong bảng DONDATHANG được không?

b. Có thể thông qua khung nhìn này để cập nhật dữ liệu trong bảng CHITIETDATHANG được không?