MỤC LỤC

ĐỀ MỤC **TRANG**

MÔ ĐUN QUẢN TRỊ CƠ SỞ DỮ LIỆU NÂNG CAO

Mã mô đun: MĐ 17

Vị trí, ý nghĩa, vai trò mô đun:

Mục tiêu của mô đun:

- Mô tả các thành phần hệ quản trị cơ sở dữ liệu, các khái niệm về cơ sở dữ liệu quan hệ hướng đối tượng và cơ sở dữ liệu quan hệ, ngôn ngữ MS SQL.

- Trình bày các kiến trúc của hệ quản trị cơ sở dữ liệu MS SQL Server, cách làm việc và tương tác giữa các thành phần kiến trúc trong hệ thống.

- Kết nối hệ thống mạng để sử dụng hệ thống cơ sở dữ liệu

- Thực hiện thành thạo các thao tác quản trị tài khoản người dùng và tài khoản nhóm đối với hệ thống MS SQL Server .

- Thiết lập cấu hình và giải quyết các vấn đề thường xảy ra trên mạng khi sử dụng truy cập cơ sở dữ liệu.

- Bảo vệ tài nguyên dữ liệu trên các hệ thống MS SQL Server.

- Bố trí làm việc khoa học đảm bảo an toàn cho người và phương tiện học tập.

Mã bài	Tên chương mục/bài	Loại bài dạy	Địa điểm	Thời lượng			
				Tổng số	Lý thuyết	Thực hành	Kiểm tra
Bài 1	Giới thiệu lịch sử phát triển	LT	Lớp học	4	2	2	
Bài 2	Các thành phần cơ bản của SQL server	LT+TH	Lớp học	8	4	4	
Bài 3	Giới thiệu một số công cụ SQL server	LT+TH	Lớp học	10	4	6	
Bài 4	Phát biểu cơ bản T-SQL	LT+TH	Lớp học	17	5	11	1
Bài 5	Tạo và sửa đổi bảng dữ liệu	LT+TH	Lớp học	17	5	12	0
Bài 6	Khóa và ràng buộc dữ liệu	LT+TH	Lớp học	11	3	7	1
Bài 7	Chuẩn hóa quan hệ	LT+TH	Lớp học	9	3	6	0
Bài 8	Bảng ảo (view)	LT+TH	Lớp học	8	2	5	1
Bài 9	Thiết kế cơ sở dữ liệu	LT+TH	Lớp học	6	2	3	1

YÊU CẦU VỀ ĐÁNH GIÁ HOÀN THÀNH MÔN HỌC/MÔ ĐUN

1. Phương pháp đánh giá

+ Hình thức kiểm tra hết môn có thể chọn một trong các hình thức sau:

- Đối với lý thuyết :Viết, vấn đáp, trắc nghiệm

- Đối với thực hành : Bài tập thực hành trên máy tính.

+ Thời gian kiểm tra:

- Lý thuyết: Không quá 150 phút

- Thực hành: Không quá 4 giờ

+ Thực hiện theo đúng qui chế thi, kiểm tra và công nhận tốt nghiệp trong dạy nghề hệ chính qui ở quyết định 14/2007/BLĐTB&XH ban hành ngày 24/05/2007 của Bộ trưởng Bộ LĐ-TB&XH.

2. Nội dung đánh giá

+ Về kiến thức: Được đánh giá qua bài kiểm tra viết, trắc nghiệm đạt được các yêu cầu sau:

- Hiểu được các kiểu dữ liệu trong MS SQL Server

- Sử dụng được các tiện ích trong MS SQL Server

- Hiểu được các phát biểu cơ bản của T-SQL

- Hiểu và tạo được các khoá và ràng buộc dữ liệu

- Chuẩn hóa được các loại quan hệ

- Thiết kế được một CSDL

- Thiết lập được các bảo mật trên CSDL

+Về kỹ năng: Đánh giá kỹ năng thực hành của sinh viên trong bài thực hành Tạo CSDL, truy vấn dữ liệu, tạo quan hệ và bảo mật dữ liệu

+ Về thái độ: Cẩn thận, tự giác.

BÀI 1
TỔNG QUAN

Mục tiêu:

- Hiểu được lịch sử phát triển và sự cần thiết của SQL SERVER trong thời đại ngày nay;

- Xác định được các cấu trúc CSDL cơ sở nhằm đảm bảo thao tác dữ liệu hiệu quả.

- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

Giới thiệu

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, gởi 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. (mỗi một dòng trong bảng SINHVIEN tương ứng với một sinh viên); và mỗi một cột biểu diễn cho một tính chất của thực thể (chẳng hạn cột NGAYSINH trong bảng SINHVIEN biểu diễn cho ngày sinh của các sinh viên được lưu trữ trong bảng).

Như vậy, liên quan đến mỗi một bảng bao gồm các yếu tố sau:

• 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 (chẳng hạn cột NGAYSINH trong bảng SINHVIEN ở hình 1.1 có kiểu là DATETIME). 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. Bảng MONHOC trong hình dưới đây có khoá là cột MAMONHOC

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 đàm bảo được tính đúng đắn và hợp lệ của dữ liệu 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, 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. 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. 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.

Các câu lệnh của SQL chuẩn:

Câu lệnh	Chức năng
Thao tác dữ liệu	
SELECT	Truy xuất dữ liệu
INSERT	Bổ sung dữ liệu
UPDATE	Cập nhật dữ liệu
DELETE	Xoá dữ liệu
TRUNCATE	Xoá toàn bộ dữ liệu trong bảng
Định nghĩa dữ liệu	
CREATE TABLE	Tạo bảng
DROP TABLE	Xoa 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

Bảng 1.1: *Một số câu lệnh thông dụng trong 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:

```
SELECT masv,hodem,ten
FROM sinhvien
WHERE malop='10T1a'
```

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.

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 1.2 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 2512.
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)
TIMESTAMP	
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 kiểu chuỗi với độ dài lớn (tối đa 2,147,483,647 ký tự)
NTEXT	Dữ liệu kiểu chuỗi với độ dài lớn và hỗ trợ UNICODE (tối đa 1,073,741,823 ký tự)

Bảng 1.2: *Một số kiểu dữ liệu thông dụng trong SQL*

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.

BÀI 2 CÁC THÀNH PHẦN CƠ BẢN CỦA SQL SERVER

Mục tiêu:

- Hiểu được mô hình quan hệ;

- Hiểu được cấu trúc CSDL trong SQL SERVERnhư: Master, Model, msdl, Tempdb, pubs, tempdb, Northwind.

Nội dung chính:

2.1. Khái niệm cơ bản về mô hình quan hệ

Trong hầu hết các cơ sở dữ liệu hiện nay, RDBMS không những lưu trữ dữ liệu mà còn quản trị hệ cơ sở dữ liệu bằng cách kiểm soát những dữ liệu nào được nhập vào và những kiểu dữ liệu nào có thể truy xuất ra khỏi hệ thống. Nếu muốn tất cả dữ liệu đều an toàn thì cần phải sử dụng đến hệ thống lưu trữ.

RDBMS cho phép lưu trữ dữ liệu cùng với những nguyên tắt ràng buộc dữ liệu do người dùng hay hệ thống định nghĩa, trong chương này chúng ta sẽ xem xét những thành phần của SQL Server, Kiểu dữ liệu, và các loại dữ liệu quan hệ

2.2. Các thành phần cấu thành của SQL Server

RDBMS cũng như SQL SERVER chứa dựng nhiều đối tượng bao gồm :

• Database : cơ sở dữ liệu của SQL SERVER

• Tập tin log : tập tin lưu trữ những chuyển tác của SQL

• Tables : bảng dữ liệu.

• Filegroups : tập tin nhóm

• Diagrams : sơ đồ quan hệ

• Views : Khung nhìn (hay bảng ảo) số liệu dựa trên bảng.

• Stored Procedure : Thủ tục và hàm nội

• User defined Function : Hàm do người dùng định nghĩa

• Users : Người sử dụng cơ sở dữ liệu

• Roles : Các qui định vai trò và chức năng trong hệ thống SQL SERVER

• Rules : Những qui tắc

• Defaults : Các giá trị mặc nhiên

• User defined data types : Kiểu dữ liệu do người dùng tự định nghĩa

• Full text catalogs : Tập phân loại dữ liệu Text

2.3. Đối tượng cơ sở dữ liệu

Cơ sở dữ liệu là đối tượng có ảnh hưởng cao nhất khi chúng ta làm việc với SQL SERVER, tuy nhiên những đối tượng con của cơ sở dữ liệu mới là thành phần chính của cơ sở dữ liệu.

Bản thân SQL Server là một cơ sở dữ liệu, chúng bao gồm các đối tượng như database, table, view, stored procedure nêu trên cùng một số cơ sở dữ liệu hỗ trợ khác.

Cơ sở dữ liệu SQL SERVER là cơ sở dữ liệu đa người dùng, với mỗi server chỉ có một hệ quản trị cơ sở dữ liệu. Nếu muốn có nhiều hệ quản trị cơ sở dữ liệu cần có nhiều Server tương ứng.

Truy cập cơ sở dữ liệu của SQL SERVER dựa vào những tài khoản người dùng riêng biệt và ứng với quyền truy cập nhất định. Khi cài đặt SQL SERVER chúng ta có 6 cơ sở dữ liệu mặc định sau :

- Master
- Model
- Msdb
- Tempdb
- Pubs
- Northwind

2.3.1 Cơ sở dữ liệu Master

Bất kỳ hệ SQL SERVER nào đều có cơ sở dữ liệu master (còn gọi là master file), cơ sở dữ liệu này chứa đựng tất cả các bảng dữ liệu đặc biệt (bảng hệ thống), chúng kiểm soát tất cả các hoạt động của hệ SQL Server.

Ví dụ : Khi người dùng tạo cơ sở dữ liệu mới trong SQL Server, thêm hay xoá một Store Procedure, tất cả những thông tin này đều được lưu trữ trong cơ sở dữ liệu master của hệ thống

2.3.2 Cơ sở dữ liệu model

Cơ sở dữ liệu này chứa tất cả các Template dùng làm mẫu để tạo cơ sở dữ liệu mới. Khi bạn tạo mới một cơ sở dữ liệu thì SQL Server lấy tất cả các mẫu (bao gồm bảng, view…) từ cơ sở dữ liệu model này.

Xuất phát từ tính chất cơ sở dữ liệu mẫu giúp SQL server thực hiện việc tạo mới Cơ sở dữ liệu cho người dùng khi có yêu cầu, bạn không được xoá CSDL mẫu này.

Khi một CSDL được tạo ra thì CSDL mới này ít nhất cũng bằng và giống như cơ sở dữ liệu model.

Vì lý do này, nếu CSDL model có dung lượng là 100MB thì CSDL mới do SQL Server tạo ra cũng phải có dữ liệu lớn hơn hoặc bằng 100MB.

2.3.3 Cơ sở dữ liệu msdb

Như đã nêu, chũng ta có hai CSDL hệ thống master và model, nếu xoá một trong hai CSDL trên thì hệ thống SQL Server sẽ bị lỗi, nhưng với CSDL msdl thì khác. msdl chính là quá trình SQL Agent lưu trữ tất cả các tác vụ xảy ra trong SQL Server.

Ví dụ khi tạo ra lịch trình cho backup dữ liệu hay lịch trình để thực hiện store procedure, tất cả các tác vụ này đều được lưu trữ msdl.

Nếu xoá CSDL này, phải cài đặt lại nó khi cần dùng hoặc khi hệ thống yêu cầu.

2.3.4 Cơ sở dữ liệu Tempdb

Cơ sở dữ liệu Tempdb là một trong những CSDL chính của SQL Server. Cơ sở dữ liệu này cho phép người dùng tạo những ứng dụng tham khảo hay thực tập trước khi bắt đầu với cơ sở dữ liệu thực.

Không những cơ sở dữ liệu tempdb này dùng làm bộ đệm cơ sở dữ liệu cho các cơ sở dữ liệu khác trong SQL Server, mà chúng còn giúp thực hiện những thao tác về cơ sở dữ liệu mỗi khi SQL khởi động.

2.3.5 Cơ sở dữ liệu pubs

Cơ sở dữ liệu pubs chứa hầu hết nội dung về hướng dẫn, trợ giúp và cả sách tham khảo về SQL Server.

Có thể xoá CSDL này mà không cần xác nhận với SQL Server.

2.3.6 Cơ sở dữ liệu Northwind

Cũng giống như cơ sở dữ liệu pubs, đây là cơ sở dữ liệu mẫu cho người dùng tham khảo, hoặc các lập trình viên Visual Basic hay Access dùng để truy cập dữ liệu SQL Server.

Northwind và pubs là hai CSDL được cài đặt như là một phần của SQL Server nếu cần dùng cấu trúc của hai CSDL này có thể sử dụng hai file kịch bản script mang tên inspubs.sql và insnwnd.sql.

CSDL này chứa đựng những đối tượng mẫu, và một số dữ liệu nhằm giúp cho việc xử lý thử nghiệm trên SQL Server thông qua các ứng dụng khác nhau như Visual Basic, Java, C++.

2.3.7. Tập tin chuyển tác log

Tập tin chứa đựng những hoạt động, hay cả những chuyển tác của CSDL theo thời gian. Thông thường khi cần tìm hiểu sự cố xảy ra với CSDL người dùng chỉ cần tham khảo tập tin log sẽ biết được nguyên nhân.

BÀI 3 GIỚI THIỆU MỘT SỐ CÔNG CỤ TRONG SQL SERVER

Mục tiêu:

- Hiểu được và sử dụng tốt các công cụ như: Enterprise manager, Query Analyzer

- Hiểu và thiết lập được các dịch vụ mạng và một số dịch vụ khác có liên quan.

- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

3.1. Các thành phần quan trọng trong SQL Server 2000

SQL Server 2000 được cấu tạo bởi nhiều thành phần như Relational Database Engine, Analysis Service và English Query.... Các thành phần này khi phối hợp với nhau tạo thành một giải pháp hoàn chỉnh giúp cho việc lưu trữ và phân tích dữ liệu một cách dễ dàng.

3.2. Relational Database Engine - Cái lõi của SQL Server:

Đây là một engine có khả năng chứa data ở các quy mô khác nhau dưới dạng table và support tất cả các kiểu kết nối (data connection) thông dụng của Microsoft như ActiveX Data Objects (ADO), OLE DB, and Open Database Connectivity (ODBC). Ngoài ra nó còn có khả năng tự điều chỉnh (tune up) *Ví dụ* như sử dụng thêm các tài nguyên (resource) của máy khi cần và trả lại tài nguyên cho hệ điều hành khi một user log off.

3.3. Replication - Cơ chế tạo bản sao (Replica):

Giả sử có một database dùng để chứa dữ liệu được các ứng dụng thường xuyên cập nhật. Muốn có một cái database giống y hệt như thế trên một server khác để chạy báo cáo (report database) (cách làm này thường dùng để tránh ảnh hưởng đến performance của server chính). Vấn đề là report server cũng cần phải được cập nhật thường xuyên để đảm bảo tính chính xác của các báo cáo. Không thể dùng cơ chế back up and restore trong trường hợp này. Thế thì phải làm sao? Lúc đó cơ chế replication của SQL Server sẽ được sử dụng để bảo đảm cho dữ liệu ở 2 database được đồng bộ (synchronized). Replication sẽ được học trong bài 12

3.4. Data Transformation Service (DTS)

Một dịch vụ chuyển dịch data vô cùng hiệu quả. Nếu làm việc trong một công ty lớn trong đó data được chứa trong nhiều nơi khác nhau và ở các dạng khác nhau cụ thể như chứa trong Oracle, DB2 (của IBM), SQL Server, Microsoft Access....chắc chắn sẽ có nhu cầu di chuyển data giữa các server

này (migrate hay transfer) và không chỉ di chuyển m à còn muốn định dạng (format) nó trước khi lưu vào database khác, khi đó sẽ thấy DTS giúp bạn giải quyết công việc trên dễ dàng như thế nào. DTS sẽ được h ọc trong bài 8.

3.5. Analysis Service

Một dịch vụ phân tích dữ liệu rất hay của Microsoft, Dữ liệu (Data) chứa trong database sẽ chẳng có ý nghĩa gì nhiều nếu như bạn không thể lấy được những thông tin (Information) bổ ích từ đó. Do đó Microsoft cung cấp cho bạn một công cụ rất mạnh giúp cho việc phân tích dữ liệu trở nên dễ dàng và hiệu quả bằng cách dùng khái niệm hình khối nhiều chiều (multi-dimension cubes) và kỹ thuật "đào mỏ dữ liệu" (data mining) sẽ được chúng tôi giới thiệu trong phần tiếp theo.

3.6. English Query - Một dịch vụ truy vấn

Đây là một dịch vụ giúp cho việc query data bằng tiếng Anh (English).

3.7. Meta Data Service:

Dịch vụ này giúp cho việc chứa đựng và "xào nấu" Meta data dễ dàng hơn. Thế thì Meta Data là cái gì vậy? Meta data là những thông tin mô tả về cấu trúc của data trong database như data thuộc loại nào String hay Integer..., một cột nào đó có phải là Primary key hay không....Bởi vì những thông tin này cũng được chứa trong database nên cũng là một dạng data nhưng để phân biệt với data "chính thống" người ta gọi nó là Meta Data. Phần này chắc là bạn phải xem thêm trong một thành phần khác của SQL Server sắp giới thiệu sau đây là SQL Server Books Online vì không có bài nào trong loạt bài này nói rõ về dịch vụ này cả.

3.8. SQL Server Books Online :

Cho dù bạn có đọc các sách khác nhau dạy về SQL server thì bạn cũng sẽ thấy books online này rất hữu dụng và không thể thiếu được (cho nên Microsoft mới hào phóng đính kèm theo SQL Server).

BÀI 4 PHÁT BIỂU CƠ BẢN T-SQL

Mục tiêu:

- Trình bày cú pháp và công dụng của các phát biểu.

- Thực hiện được việc truy vấn dữ liệu trên câu lệnh T-SQL đúng yêu cầu.

- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

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.

4.1 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

[WHERE điều_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 4.1: 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

4.1.1 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 4.2: Câu lệnh dưới đây hiển thị danh sách các khoa trong trường

```
SELECT * FROM khoa
```

kết quả câu lệnh như sau:

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
DHT07	Khoa Ngữ văn	054821133
DHT08	Khoa Lịch sử	054823833
DHT09	Khoa Mác - Lê Nin	054825698
DHT10	Khoa Luật	054821135

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 4.3: câu lệnh sau gán bí danh là a cho bảng khoa

```
SELECT * FROM khoa a
```

4.1.2 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 4.4: Câu lệnh

```
SELECT * FROM lop
```

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
```

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 4.6:

```
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 4.7: 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.

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 4.8: 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
```

Kết quả của hai câu lệnh trên đều có dạng như sau

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 4.9: câu lệnh dưới đây cho biết tên và số tiết của các môn học

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

Nếu trong danh sách chọn có sự xuất hiện của giá trị hằng thì giá trị này sẽ xuất hiện trong một cột của kết quả truy vấn ở tất cả các dòng

Ví dụ 4.10: Câu lệnh

```
SELECT tenmonhoc,'Số tiết: ',sodvht*15 AS sotiet
FROM monhoc
```

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 4.11: Hai câu lệnh dưới đây

```
SELECT khoa FROM lop
```

và:

```
SELECT DISTINCT khoa FROM lop
```

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 4.12: 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 4.13: 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
```

4.1.3 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 4.14: 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 4.15: 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 4.16: 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 4.17: 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 4.18: 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

```
SELECT hodem,ten FROM sinhvien
WHERE hodem LIKE 'Lê%' AND ten LIKE '[AB]%'
```

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
```

4.1.4 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 4.19: 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.

4.1.5 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 được sắp xếp có thể theo chiều tăng (ASC) hoặc giảm (DESC), mặc định là sắp xếp theo chiều tăng.

Ví dụ 4.20: Câu lệnh dưới đây hiển thị danh sách các môn học và sắp xếp theo chiều giảm dần của số đơn vị học trình

```
SELECT * FROM monhoc
ORDER BY sodvht DESC
```

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 4.21: Câu lệnh

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

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
```

4.1.8 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 4.35: 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 4.36: 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
```

còn câu lệnh:

```
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 4.37: 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 4.38: 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
```

4.1.9 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 4.39: Câu lệnh:

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

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 4.40: 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
```

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 4.50: 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

4.2 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

4.2 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 4.48: 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 4.49: 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')
```

4.3 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 4.51: 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

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 4.54: 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)
```

4.4 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 4.55: 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 4.56: 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 4.57: 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 4.58: 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

4.5 Bài tập

BÀI 5 TẠO VÀ SỬA ĐỔI BẢNG DỮ LIỆU

Mục tiêu:

Tạo được bảng dữ liệu, tạo được các khóa, ràng buộc dữ liệu;

Sửa đổi bảng dữ liệu;

Thực hiện các phát biểu tạo và sửa đổi trên bảng dữ liệu.

Thực hiện các thao tác an toàn với máy tính

Nội dung chính:

Các câu lệnh SQL đã đề cập đến trong các chương trên 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

5.1 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.

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* 6.1:** 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 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 đùng 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 6.2: 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:

5.1.1 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 6.3: 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 6.4: 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                     NULL ,
makhoa          NVARCHAR(5),
CONSTRAINT chk_lop
CHECK (namnhaphoc<=YEAR(GETDATE()) AND
        hedaotao IN ('chính quy','tại chức'))
      )
```

5.1.2 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 6.5: 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 6.6: 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.

5.1.3 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 6.7: 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:

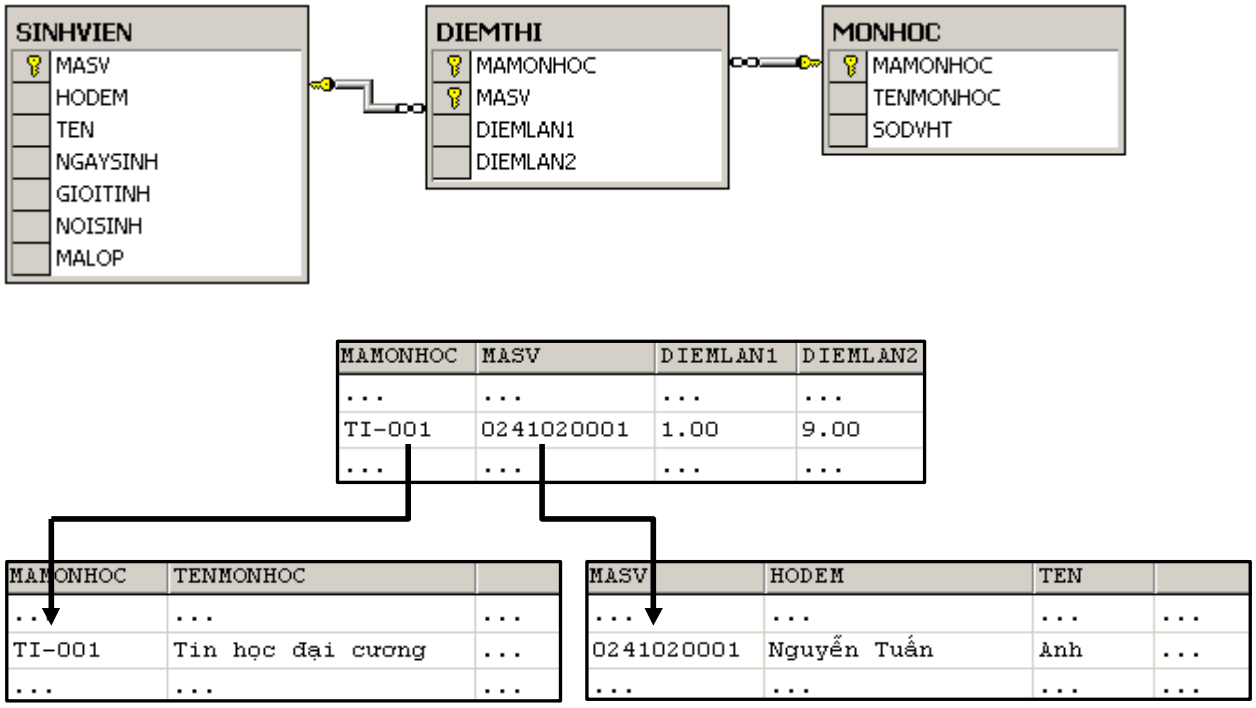
```
CREATE TABLE lop
(
malop          NVARCHAR(10)        NOT NULL,
tenlop         NVARCHAR(30)        NOT NULL,
khoa    SMALLINT                   NULL,
hedaotao       NVARCHAR(25)        NULL,
namnhaphoc     INT                 NULL,
makhoa         NVARCHAR(5),
CONSTRAINT pk_lop PRIMARY KEY (malop),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

5.1.4 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.



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: (Mặc định) Nếu bản ghi trong bảng tham chiếu đang được 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 6.8: 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.

5.2 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 COLUMN tên_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 6.9: 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.

5.3 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

BÀI 6 KHÓA VÀ RÀNG BUỘC DỮ LIỆU

Mục tiêu

Tạo được các loại khóa: Khóa chính, khóa phụ, khóa ngoài.

Ràng buộc dữ liệu: Check.

Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

Nói đến Data Integrity là ta nói đến tính toàn vẹn của một database hay nói một cách khác là data chứa trong database phải chính xác và đáng tin cậy. Nếu data chứa trong database không chính xác ta nói database mất tính toàn vẹn (lost data integrity). Trong bài này chúng ta sẽ bàn qua các phương pháp để giữ cho database được toàn vẹn thông qua việc dùng các khoá và các ràng buộc dữ liệu.

6.1. Các Phương Pháp Đảm Bảo Data Integrity

SQL Server dùng một số cách để đảm bảo Data Integrity. Một số cách như Triggers hay Index sẽ được bàn đến trong các bài sau tuy nhiên trong phạm vi bài này chúng ta cũng nói sơ qua các cách trên.

• Data Type : Data type cũng có thể đảm bảo tính toàn vẹn của data *Ví dụ* bạn khai báo data type của một cột là Integer thì bạn không thể đưa giá trị thuộc dạng String vào được.

• Not Null Definitions : Null là một loại giá trị đặc biệt, nó không tương đương với zero, blank hay empty string " " mà có nghĩa là không biết (unknown) hay chưa được định nghĩa (undefined). Khi thiết kế database ta nên luôn cẩn thận trong việc cho phép một cột được Null hay Not Null vì việc chứa Null data có thể làm cho một số ứng dụng vốn không xửa lý null data kỹ lưỡng bị "té".

• Default Definitions : Nếu một cột được cho một giá trị default thì khi bạn không đưa vào một giá trị cụ thể nào thì SQL Server sẽ dùng giá trị mặc định này. Bạn phải dùng Default đối với Not Null definition.

• Identity Properties : Data thuộc dạng ID sẽ đảm bảo tính duy nhất của data trong table.

• Constraints : Đây sẽ là phần mà ta đào sâu trong bài này. Constraint tạm dịch là những ràng buộc mà ta dùng để đảm bảo tính toàn vẹn của data. Constraints là những quy luật mà ta áp đặt lên một cột để đảm bảo tính chính xác của dữ liệu được nhập vào.

• Rules : Đây là một object mang tính backward-compatible chủ yếu để tương thích với các version trước đây của SQL Server. Rules tương đương với

CHECK Constraint trong SQL Server 2000 nhưng người ta có xu hướng sử dụng CHECK Constraint vì nó chính xác hơn và có thể đặt nhiều Constraints lên một cột trong khi đó chỉ có một rule cho một cột mà thôi. Chú ý rule là một object riêng và sau đó liên kết với một cột nào đó của table trong khi CHECK constraint là một thuộc tính của table nên có thể được tạo ra với lệnh CREATE TABLE.

• Triggers : Một loại stored procedure đặc biệt được thực thi một cách tự động khi một table được Update, Insert, hay Delete. *Ví dụ* ta muốn khi một món hàng được bán ra thì tổng số hàng hóa trong kho phải được giảm xuống (-1) chẳng hạn khi đó ta có thể dùng trigger để đảm bảo chuyện đó. Triggers sẽ được bàn kỹ trong các bài sau.

6.2. Các loại ràng buộc (Constraints)

Constraints là những thuộc tính (property) mà ta áp đặt lên một table hay một cột để tránh việc lưu dữ liệu không chính xác vào database (invalid data). Thật ra NOT NULL hay DEFAULT cũng được xem là một dạng constraint nhưng chúng ta không bao gồm hai loại này ở đây mà chỉ trình bày 4 loại constraints là Primary Key Constraint, Unique Constraint, Foreign Key Constraint và Check Constraint.

6.2.1 Primary Key Constraint:

Một table thường có một hay nhiều cột có giá trị mang tính duy nhất để xác định một hàng bất kỳ trong table. Ta thường gọi là Primary Key và được tạo ra khi ta Create hay Alter một table với Primary Key Constraint.

Một table chỉ có thể có một Primary Key constraint. Có thể có nhiều cột tham gia vào việc tạo nên một Primary Key, các cột này không thể chứa Null và giá trị trong các cột thành viên có thể trùng nhau nhưng giá trị của tất cả các cột tạo nên Primary Key phải mang tính duy nhất.

Khi một Primary Key được tạo ra một Unique Index sẽ được tự động tạo ra để duy trì tính duy nhất. Nếu trong table đó chưa có Clustered Index thì một Unique + Clustered Index sẽ được tạo ra.

Có thể tạo ra Primary Key Constraints như sau:

```
CREATE TABLE Table1
        (Col1 INT PRIMARY KEY,
         Col2 VARCHAR(30)
        )
```

hay

```
CREATE TABLE Table1
        (Col1 INT,
```

```
Col2 VARCHAR(30),
CONSTRAINT table_pk PRIMARY KEY (Col1)
)
```

6.2.2 Unique Constraint

Bạn có thể tạo Unique Constraint để đảm bảo giá trị của một cột nào đó không bị trùng lập. Tuy Unique Constraint và Primary Key Constraint đều đảm bảo tính duy nhất nhưng bạn nên dùng Unique Constraint trong những trường hợp sau:

- Nếu một cột (hay một sự kết hợp giữa nhiều cột) không phải là primary key. Nên nhớ chỉ có một Primary Key Constraint trong một table trong khi ta có thể có nhiều Unique Constraint trên một table.

- Nếu một cột cho phép chứa Null. Unique constraint có thể áp đặt lên một cột chứa giá trị Null trong khi primary key constraint thì không.

Cách tạo ra Unique Constraint cũng tương tự như Primary Key Constraint chỉ việc thay chữ Primary Key thành Unique. SQL Server sẽ tự động tạo ra một non-clustered unique index khi ta tạo một Unique Constraint.

6.2.3 Foreign Key Constraint

Foreign Key là một cột hay một sự kết hợp của nhiều cột được sử dụng để áp đặt mối liên kết data giữa hai table. Foreign key của một table sẽ giữ giá trị của Primary key của một table khác và chúng ta có thể tạo ra nhiều Foreign key trong một table.

Foreign key có thể reference (tham chiếu) vào Primary Key hay cột có Unique Constraints. Foreign key có thể chứa Null. Mặc dù mục đích chính của Foreign Key Constraint là để kiểm soát data chứa trong table có Foreign key (tức table con) nhưng thực chất nó cũng kiểm soát luôn cả data trong table chứa Primary key (tức table cha). *Ví dụ* nếu ta delete data trong table cha thì data trong table con trở nên "mồ côi" (orphan) vì không thể reference ngược về table cha. Do đó Foreign Key constraint sẽ đảm bảo điều đó không xảy ra. Nếu bạn muốn delete data trong table cha thì trước hết bạn phải drop hay disable Foreign key trong table con trước.

Có thể tạo ra Foreign Key Constraints như sau:

```
CREATE TABLE Table1
    (Col1 INT PRIMARY KEY,
     Col2 INT REFERENCES Employees(EmployeeID)
     )
```

hay

```
CREATE TABLE Table1
        (Col1 INT PRIMARY KEY,
         Col2 INT,
         CONSTRAINT col2_fk FOREIGN KEY (Col2)
         REFERENCES Employees (EmployeeID)
         )
```

Đôi khi chúng ta cũng cần Disable Foreign Key Constraint trong trường hợp:

• Insert hay Update: Nếu data insert vào sẽ vi phạm những ràng buộc có sẵn (violate constraint) hay constraint của ta chỉ muốn áp dụng cho data hiện thời mà thôi chứ không phải data sẽ insert.

• Tiến hành quá trình replicate. Nếu không disable Foreign Key Constraint khi replicate data thì có thể cản trở quá trình copy data từ source table tới destination table một cách không cần thiết.

6.2.4 Check Constraint

Check Constraint dùng để giới hạn hay kiểm soát giá trị được phép insert vào một cột. Check Constraint giống Foreign Key Constraint ở chỗ nó kiểm soát giá trị đưa vào một cột nhưng khác ở chỗ Foreign Key Constraint dựa trên giá trị ở table cha để cho phép một giá trị được chấp nhận hay không trong khi Check Constraint dựa trên một biểu thức logic (logic expression) để kiểm tra xem một giá trị có hợp lệ không. *Ví dụ* ta có thể áp đặt một Check Constraint lên cột salary để chỉ chấp nhận tiền lương từ $15000 đến $100000/năm.

Ta có thể tạo ra nhiều Check Constraint trên một cột. Ngoài ra ta có thể tạo một Check Constraint trên nhiều cột bằng cách tạo ra Check Constraint ở mức table (table level).

Có thể tạo ra Check Constraint như sau:

```
CREATE TABLE Table1
        (Col1 INT PRIMARY KEY,
         Col2 INT
         CONSTRAINT limit_amount CHECK (Col2 BETWEEN 0
AND 1000),
         Col3 VARCHAR(30)
         )
```

Trong *Ví dụ* này ta giới hạn giá trị chấp nhận được của cột Col2 từ 0 đến 1000. *Ví dụ* sau sẽ tạo ra một Check Constraint giống như trên nhưng ở table level:

```
CREATE TABLE Table1
    (Col1 INT PRIMARY KEY,
     Col2 INT,
     Col3 VARCHAR(30),
     CONSTRAINT limit_amount CHECK (Col2 BETWEEN 0 AND 1000)
     )
```

Tương tự như Foreign Key Constraint đôi khi ta cũng cần disable Check Constraint trong trường hợp Insert hay Update mà việc kiểm soát tính hợp lệ của data không áp dụng cho data hiện tại. Trường hợp thứ hai là replication.

Muốn xem hay tạo ra Constraint bằng Enterprise Manager thì làm như sau:

Click lên trên một table nào đó và chọn Design Table-> Click vào icon bên phải "Manage Constraints..."

6.3 Bài tập

BÀI 7 CHUẨN HÓA QUAN HỆ

Mục tiêu:

- Xác định được các loại quan hệ trong bảng;

- Chuẩn hóa được các mối quan hệ giữa các bảng.

- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

7.1 Khái niệm về chuẩn hoá và quan hệ

Khi thiết kế cơ sở dữ liệu quan hệ ta thường đứng trước vấn đề lựa chọn giữa các lược đồ quan hệ: lược đồ nào tốt hơn ? Tại sao ? Mục này sẽ nghiên cứu một số tiêu chuẩn đánh giá lược đồ quan hệ và các thuật toán giúp chúng ta xây dựng được lược đồ cơ sở dữ liệu quan hệ có cấu trúc tốt.

Có thể nói tổng quát một lược đồ quan hệ có *cấu trúc tốt* là lược đồ không chứa đựng sự *dư thừa dữ liệu*, tức là là sự trùng lặp thông tin trong cơ sở dữ liệu.

7.1.1. Sự dư thừa dữ liệu

Dư thừa dữ liệu là sự trùng lặp thông tin trong cơ sở dữ liệu.

+ *Ví dụ*:

Xét quan hệ EMP(ENO, ENAME, TITLE, SAL, PNO, RESP, DUR). Nếu một nhân viên tham gia trong nhiều dự án, thì các dữ liệu như ENAME, TITLE, SAL phải lặp lại nhiều lần và kéo theo dư thừa dữ liệu.

Ngoài việc gây lãng phí dung lượng lưu trữ, sự dư thừa dữ liệu có thể gây ra những hậu quả nghiêm trọng đối với dữ liệu khi người dùng cập nhật dữ liệu làm cho dữ liệu không tương thích, bất định hoặc mất mát. Các sự cố như vậy gọi là những dị thường.

7.1.2. Các dị thường cập nhật dữ liệu

Ta sẽ minh hoạ các dị thường bằng các lược đồ

EMP(ENO, ENAME, TITLE, SAL, PNO, RESP, DUR)

PROJ(PNO, PNAME, BUDGET)

a. Dị thường do dữ liệu lặp: Một số thông tin có thể được lặp lại một cách vô ích.

+ *Ví dụ*: Trong quan hệ EMP tên (ENAME), chức vụ (TITLE), và lương (SAL) của nhân viên được lặp lại trong mỗi dự án mà họ tham gia. Điều này rõ ràng là làm lãng phí chỗ lưu trữ và đối nghịch với các nguyên lý của cơ sở dữ liệu.

b. Dị thường chèn bộ: Không thể chèn bộ mới vào quan hệ, nếu không có đầy đủ dữ liệu.

+ *Ví dụ*: Xét quan hệ EMP. Giả sử một nhân viên mới được nhận vào công ty và chưa được phân công vào dự án nào cả. Khi đó chúng ta không thể nhập các thông tin về tên, chức vụ, lương của nhân viên này vào quan hệ, vì khoá của EMP là (ENO, PNO).

c. Dị thường xoá bộ: Trường hợp này ngược với dị thường chèn bộ. Việc xoá bộ có thể kéo theo mất thông tin.

+ *Ví dụ*: Xét quan hệ EMP. Giả sử một nhân viên làm việc trong một dự án duy nhất. Khi dự án chấm dứt, chúng ta không thể xoá thông tin về dự án đó trong EMP được, vì nếu làm thế ta sẽ mất luôn thông tin về nhân viên đó.

d. Dị thường sửa bộ: Việc sửa đổi dữ liệu dư thừa có thể dẫn đến sự không tương thích dữ liệu.

+ *Ví dụ*: Xét quan hệ EMP. Giả sử một nhân viên làm việc trong nhiều dự án. Khi có sự thay đổi về lương, rất nhiều bộ phải cập nhật sự thay đổi này. Điều đó gây lãng phí thời gian công sức và là nguy cơ gây ra sự không thống nhất dữ liệu.

Trong các *Ví dụ* trên ta thấy tác hại của sự dư thừa dữ liệu và sự cần thiết phải loại bỏ chúng khỏi các lược đồ quan hệ. Quá trình từng bước thay thế một lược đồ quan hệ bằng các tập lược đồ quan hệ đơn giản và chuẩn tắc hơn gọi là chuẩn hoá. Mục đích của chuẩn hoá là loại bỏ các dị thường (hoặc các khía cạnh không mong muốn khác) để có những quan hệ tốt hơn.

Cơ sở lý thuyết của việc thiết kế lược đồ cơ sở dữ liệu quan hệ tốt là khái niệm phụ thuộc dữ liệu. Phụ thuộc dữ liệu biểu diễn các quan hệ nhân quả giữa các thuộc tính trong quan hệ. *Ví dụ* trong bảng EMP, thuộc tính SAL phụ thuộc vào thuộc tính ENO, vì mỗi nhân viên chỉ có một lương duy nhất.

Cũng dựa trên khái niệm phụ thuộc dữ liệu người ta định nghĩa các dạng chuẩn của lược đồ dữ liệu quan hệ. Mỗi dạng chuẩn đáp ứng một yêu cầu nhất định đối với lược đồ quan hệ.

Quá trình biến đổi một lược đồ thành lược đồ tương đương (bảo toàn thông tin và phụ thuộc dữ liệu) thoả mãn dạng chuẩn gọi là quá trình chuẩn hoá lược đồ quan hệ.

Khái niệm phụ thuộc dữ liệu sẽ được nghiên cứu chi tiết ở phần sau.

7.2. Cấu trúc phụ thuộc dữ liệu

Có ba dạng phụ thuộc dữ liệu, *phụ thuộc hàm* (*functional dependancy*- FD) , *phụ thuộc đa trị* (*multivalued dependancy* - MVD) và *phụ thuộc chiếu nối* (*projection-join dependancy* - PJD)

7.2.1 Phụ thuộc hàm

Cho lược đồ quan hệ R=(A_1, A_2, ..., A_n) và X, Y là các tập con của {A_1, A_2, ..., A_n}. Ta nói rằng X *xác định hàm* Y hay Y *phụ thuộc hàm* X, ký hiệu X Y, nếu mọi quan hệ bất kỳ **r** của lược đồ R thoả mãn:

u, v **r** : u(X) = v(X) u(Y) = v(Y)

Cần nhấn mạnh rằng tính chất phụ thuộc hàm phải thoả với mọi quan hệ **r** của lược đồ R. Ta không thể chỉ xét một quan hệ đặc biệt (quan hệ rỗng chẳng hạn) rồi quy nạp cho toàn lược đồ. Nhưng ta có thể phủ nhận phụ thuộc hàm qua một quan hệ cụ thể nào đó.

Phụ thuộc hàm X Y gọi là phụ thuộc hàm *tầm thường* nếu Y X (hiển nhiên là nếu Y X thì theo định nghĩa ta có X Y).

Phụ thuộc hàm X Y gọi là phụ thuộc hàm *nguyên tố* nếu không có tập con thực sự Z X thoả Z Y.

Tập thuộc tính K R gọi là *khoá* nếu nó xác định hàm tất cả các thuộc tính và K R là phụ thuộc hàm nguyên tố.

+ *Ví dụ*: Xét quan hệ PROJ. Ta có thể chấp nhận rằng mỗi dự án có tên và kinh phí xác định. Vậy có thể khẳng định

PNO (PNAME, BUDGET)

Trong quan hệ EMP ta có

(ENO, PNO) (ENAME, TITLE, SAL, RESP, DUR)

ENO (ENAME, TITLE, SAL)

Hoàn toàn hợp lý khi chúng ta khẳng định rằng lương của mỗi chức vụ là cố định, do đó sẽ tồn tại phụ thuộc hàm

TITLE SAL

7.2.2 Phụ thuộc đa trị

Cho lược đồ quan hệ R=(A_1, A_2, ..., A_n) và X, Y là các tập con của {A_1, A_2, ..., A_n}. Ta nói rằng X *xác định đa trị* Y hay Y *phụ thuộc đa trị* vào X, ký hiệu X Y, nếu mọi quan hệ bất kỳ **r** của lược đồ R thoả mãn:

ứng với mỗi giá trị của miền giá trị các thuộc tính trong X, có một tập giá trị các thuộc tính trong Y liên quan và tập này độc lập với các thuộc tính trong Z=R\(X Y), tức là:

x D(X) y, y' D(Y) z, z' D(Z):(x,y,z), (x, y', z') **r** (x, y, z'), (x, y', z) **r**

Chú ý rằng phụ thuộc hàm là trường hợp riêng của phụ thuộc đa trị, tức là

X Y X Y

Thật vậy, nếu (x,y,z), (x, y', z') **r** và X Y

thì y=y', và kéo theo (x, y, z'), (x, y', z) **r**

+ *Ví dụ*:

Trở lại *Ví dụ* đang xét. Giả sử ta muốn duy trì thông tin về tập nhân viên và về tập dự án có liên quan đến công ty cũng như về chi nhánh (PLACE) thực hiện dự án. Yêu cầu này có thể được thực hiện bằng cách định nghĩa quan hệ

SKILL(ENO, PNO, PLACE)

Ta giả sử (có thể không thực tế) (1) mỗi nhân viên đều có thể làm việc cho mọi dự án, (2) mỗi nhân viên đều có thể làm việc tại mọi chi nhánh và (3) mỗi dự án đều có thể được thực hiện tại bất kỳ chi nhánh nào. Một quan hệ mẫu thoả các điều kiện này cho ở bảng sau:

SKILL

ENO	*PNO*	PLACE
E1	P1	Toronto
E1	P1	New York
E1	P1	London
E1	P2	Toronto
E1	P2	New York
E1	P2	London
E2	P1	Toronto
E2	P1	New York
E2	P1	London
E2	P2	Toronto
E2	P2	New York
E2	P2	London

Chú ý rằng không có phụ thuộc hàm nào trong quan hệ SKILL; tất cả thuộc tính là thuộc tính khoá. Quan hệ SKILL có hai phụ thuộc đa trị

ENO PNO

ENO PLACE

7.2.3 Phụ thuộc chiếu-nối

Cho lược đồ quan hệ R=(A_1, A_2, ..., A_n) và R_1, R_2 ,..., R_k là các tập con của {A_1, A_2, ..., A_n}. Ta nói rằng {R_1, R_2 ,..., R_k } xác định một *phụ thuộc chiếu-nối* của R nếu mọi quan hệ **r** của R là nối tự nhiên của các chiếu của nó lên R_1, R_2 ,..., R_k , tức là

$$\mathbf{r} = {}_{R1}(\mathbf{r}) \bowtie {}_{R2}(\mathbf{r}) \bowtie \ldots \bowtie {}_{Rk}(\mathbf{r})$$

Chú ý rằng phụ thuộc đa trị là trường hợp riêng của phụ thuộc chiếu-nối, tức là

X Y {XY, XZ} tạo phụ thuộc chiếu-nối

7.3. Chuẩn hoá lược đồ quan hệ

Chúng ta đã chỉ ra rằng sự dư thừa dữ liệu là nguyên nhân của các dị thường khi cập nhật dữ liệu dẫn đến sự không tương thích dữ liệu và các hậu quả nghiêm trọng khác. Một lược đồ cơ sở dữ liệu được cho là *tốt* là phải loại bỏ được sự dư thừa dữ liệu. Tuy nhiên ta cần đưa ra định nghĩa chính xác thế nào là lược đồ cơ sở dữ liệu tốt cùng với quá trình thiết kế chúng. Quá trình biến đổi một lược đồ cơ sở dữ liệu thành *lược đồ tương đương*, tức phải bảo toàn thông tin và bảo toàn phụ thuộc dữ liệu, thoả mãn những tiêu chuẩn nhất định gọi là quá trình *chuẩn hoá lược đồ quan hệ*.

Chuẩn hoá lược đồ quan hệ thường được thực hiện qua các giai đoạn tương ứng với các dạng chuẩn (xem sơ đồ dưới). *Dạng chuẩn* là trạng thái quan hệ được xác định bằng cách áp dụng các quy tắc đối với phụ thuộc hàm của quan hệ.

7.3.1 Dạng chuẩn thứ nhất (1NF)

Quan hệ gọi là ở ***dạng chuẩn thứ nhất*** hay ***quan hệ chuẩn hoá*** nếu miền giá trị của mỗi thuộc tính chỉ chứa những giá trị *nguyên tử*, tức là không phân chia được nữa. Như vậy mỗi giá trị trong quan hệ cũng là nguyên tử.

Dạng chuẩn 1 chỉ có ý nghĩa ở mức thể hiện của lược đồ quan hệ, vì chỉ liên quan đến giá trị các thuộc tính của các bộ trong một quan hệ được định nghĩa trên lược đồ quan hệ đó.

7.3.2 Dạng chuẩn thứ 2 (2NF)

Thuộc tính A gọi là ***phụ thuộc đầy đủ*** vào tập thuộc tính X nếu X A là phụ thuộc hàm nguyên tố.

Giả sử K là khoá của lược đồ R. Khi đó mọi thuộc tính không khoá A của R đều phụ thuộc hàm vào khoá K: K A. Nếu A không phụ thuộc đầy đủ

vào K thì tồn tại tập con thực sự H của K xác định hàm A, tức H A. Khi đó phụ thuộc hàm H A gọi là ***phụ thuộc hàm bộ phận***.

Một lược đồ quan hệ gọi là ở ***dạng chuẩn thứ 2*** nếu nó ở dạng chuẩn thứ 1 và không có phụ thuộc hàm bộ phận, tức là mọi thuộc tính không khoá đều phụ thuộc đầy đủ vào các khoá của lược đồ.

+ *Ví dụ*:

- Xét các quan hệ sau:

EMP(ENO, ENAME, TITLE, SAL, PNO, RESP, DUR)

PROJ(PNO, PNAME, BUDGET)

Lược đồ của EMP có khoá là (ENO, PNO).

Phụ thuộc hàm ENO (ENAME, TITLE) là phụ thuộc hàm bộ phận vì vế phải là tập con thực sự của khoá. Vậy EMP không ở dạng chuẩn thứ 2.

Lược đồ của PROJ không có phụ thuộc hàm bộ phận, vậy nó ở dạng chuẩn 2.

- Xét quan hệ KHO_HANG(Kho, Hang, QuayHang, NhanVien). Lược đồ của quan hệ này có hai phụ thuộc hàm sau:

Kho,Hang QuayHang: Mỗi mặt hàng ở mỗi kho chỉ được bán ở 1 quầy hàng;

Kho,QuayHang NhanVien: Mỗi quầy hàng của mỗi kho chỉ có 1 nhân viên phụ trách.

Khoá của lược đồ này là (Kho, Hang).

Vậy lược đồ này ở dạng chuẩn thứ 2 vì không có phụ thuộc hàm bộ phận.

7.3.3 Dạng chuẩn thứ 3 (3NF)

Phụ thuộc hàm X A gọi là ***phụ thuộc hàm bắc cầu***, nếu nó là phụ thuộc hàm nguyên tố, A là thuộc tính không khoá, A X, và X chứa thuộc tính không khoá.

Khi đó với mọi khoá K ta có các phụ thuộc hàm không tầm thường K X & X A. Mặt khác không thể có X K vì X chứa các thuộc tính không khoá và không chứa khoá (vì X A là nguyên tố).

Nói một cách khác phụ thuộc hàm bắc cầu là sự phụ thuộc không tầm thường giữa các thuộc tính không khoá.

Một lược đồ quan hệ gọi là ở ***dạng chuẩn thứ 3*** nếu nó ở dạng chuẩn thứ 2 và không có phụ thuộc hàm bắc cầu.

+ *Ví dụ*:

- Lược đồ của quan hệ

EMP(ENO, ENAME, TITLE, SAL, PNO, RESP, DUR)

EMP có khoá là (ENO, PNO).

Phụ thuộc hàm TITLE SAL là phụ thuộc hàm bắc cầu. Vậy EMP không ở dạng chuẩn thứ 3.

- Lược đồ của quan hệ

PROJ(PNO, PNAME, BUDGET)

không có phụ thuộc hàm bắc cầu, vậy nó ở dạng chuẩn 3.

- Xét quan hệ KHO_HANG(Kho, Hang, QuayHang, NhanVien). Ta có hai phụ thuộc hàm sau:

Kho,Hang QuayHang: Mỗi mặt hàng ở mỗi kho chỉ được bàn ở 1 quầy hàng;

Kho,QuayHang NhanVien: Mỗi quầy hàng của mỗi kho chỉ có 1 nhân viên phụ trách.

Khoá của lược đồ này là (Kho, Hang).

Phụ thuộc hàm thứ hai là phụ thuộc hàm bắc cầu, vì thế lược đồ không ở dạng chuẩn thứ 3, mặc dù nó ở dạng chuẩn thứ 2.

7.3.4. Dạng chuẩn Boyce-Codd (BCNF)

Một lược đồ quan hệ gọi là ở ***dạng chuẩn Boyce-Codd*** nếu mọi phụ thuộc hàm không tầm thường đều có vế trái là siêu khoá

+ *Ví dụ*:

- Lược đồ của quan hệ

PROJ(PNO, PNAME, BUDGET)

chỉ có phụ thuộc hàm duy nhất PNO (PNAME, BUDGET), vậy nó ở dạng chuẩn Boyce-Codd.

- Xét lại lược đồ

LOPHOC(Lop,MonHoc,GiaoVien) với 2 phụ thuộc hàm sau:

GiaoVien MonHoc và (Lop,MonHoc) GiaoVien

Lược đồ có 2 khoá

K_1 = (Lop, MonHoc) và K_2 = (Lop, GiaoVien),

nên tất cả thuộc tính đều là thuộc tính khoá. Như vậy lược đồ ở dạng chuẩn thứ 3. Tuy nhiên lược đồ không ở dạng chuẩn Boyce-Codd vì phụ thuộc hàm

GiaoVien MonHoc

không thoả yêu cầu vế trái phải là siêu khoá.

Sự dị thường khi thêm bộ hay sửa bộ thể hiện ở chỗ nếu một giáo viên dạy nhiều lớp (cùng một môn học) thì thông tin về giáo viên đó lặp lại nhiều lần gây dư thừa dữ liệu.

Sự dị thường khi xoá bộ thể hiện ở chỗ nếu giáo viên T chỉ dạy lớp C nào đó, thì thông tin về giáo viên T (môn học mà giáo viên đó dạy) sẽ bị mất nếu ta xoá bản ghi tương ứng (chẳng hạn vì giáo viên T thôi không dạy lớp C nữa).

7.3.5. Dạng chuẩn thứ 4 (4NF)

Một quan hệ R được gọi là ở ***dạng chuẩn thứ 4*** nếu với mỗi phụ thuộc đa trị X Y trong R, X cũng xác định hàm tất cả thuộc tính của R.

Như vậy, nếu quan hệ ở dạng chuẩn BCNF và các phụ thuộc đa trị cũng là phụ thuộc hàm thì quan hệ này ở dạng chuẩn 4.

+ *Ví dụ*:

Xét quan hệ

SKILL(ENO, PNO, PLACE)

ENO	PNO	PLACE
E1	P1	Toronto
E1	P1	New York
E1	P1	London
E1	P2	Toronto
E1	P2	New York
E1	P2	London
E2	P1	Toronto
E2	P1	New York
E2	P1	London
E2	P2	Toronto
E2	P2	New York
E2	P2	London

Chú ý rằng không có phụ thuộc hàm nào trong quan hệ SKILL; tất cả thuộc tính là thuộc tính khoá. Quan hệ SKILL có hai phụ thuộc đa trị

ENO PNO

ENO PLACE

Vì quan hệ không có phụ thuộc hàm nên nó ở dạng BCNF. Tuy nhiên nó không ở dạng chuẩn 4, vì ENO không phải là khoá.

Để đạt dạng chuẩn 4, cần phân rã SKILL thành hai quan hệ

EP(ENO, PNO) và **EL**(ENO, PLACE)

7.3.5.. Dạng chuẩn thứ 5 (5NF)

Một quan hệ R được gọi là ở ***dạng chuẩn thứ 5*** , còn gọi là ***dạng chuẩn chiếu-nối*** **PJNF**, nếu mỗi phụ thuộc chiếu nối được xác định bởi các khoá dự tuyển của R.

+ *Ví dụ*:

Với quan hệ **PROJ**(PNO, PNAME, BUDGET) ta có phụ thuộc chiếu-nối

{(PNO, PNAME), (PNO, BUDGET)}

và mỗi thành phần đều có khoá chính PNO.

BÀI 8
BẢNG ẢO - VIEW

Mục tiêu:

- Hiểu được thế nào là view, sự giống nhau giữa table và view;
- Dùng view để lọc dữ liệu;
- Các phép tạo, cập nhật, thêm dữ liệu vào view.
- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

8.1. Khái niệm về View

Định nghĩa một cách đơn giản thì view trong SQL Server tương tự như Query trong Access database. View có thể được xem như một table ảo mà data của nó được select từ một stored query. Đối với programmer thì view không khác chi so với table và có thể đặt ở vị trí của table trong các câu lệnh SQL. Đặc điểm của View là ta có thể join data từ nhiều table và trả về một recordset đơn. Ngoài ra ta có thể "xào nấu" data (manipulate data) trước khi trả về cho user bằng cách dùng một số logic checking như (if, case...).

Ví dụ:

```
Create View OrderReport
As
Select OrderID,
    (case when [Name] is null then 'New Customer'
      else [Name]
      end )As CustomerName,
      ProductName,
      DateProcessed
From Customers Right Outer Join Orders on
            Customers.CustomerID=Orders.CustomerID
```

Trong *Ví dụ* trên ta chủ yếu trả về data từ Orders table trong PracticeDB nhưng thay vì display CustomerID vốn không có ý nhiều ý nghĩa đối với user ta sẽ display tên của customer bằng cách join với Customer table. Nếu Customer Name là Null nghĩa là tên của customer đã đặt order không tồn tại trong system. Thay vì để Null ta sẽ display "New Customer" để dễ nhìn hơn cho user.

Nói chung câu lệnh SQL trong View có thể từ rất đơn giản như select toàn bộ data từ một table cho đến rất phức tạp với nhiều tính năng programming của T-SQL.

View Thường Được Dùng Vào Việc Gì?

View thường được sử dùng vào một số công việc sau:

Tập trung vào một số data nhất định : ta thường dùng view để select một số data mà user quan tâm hay chịu trách nhiệm và loại bỏ những data không cần thiết.

Ví dụ: Giả sử trong table ta có column "Deleted" với giá trị là True hay False để đánh dấu một record bị delete hay không. Việc này đôi khi được dùng cho việc Audit. Nghĩa là trong một ứng dụng nào đó khi user delete một record nào đó, thay vì ta physically delete record ta chỉ logically delete bằng cách đánh dấu record là đã được "Deleted" để đề phòng user yêu cầu roll back. Như vậy chủ yếu ta chỉ quan tâm đến data chưa delete còn data đã được đánh dấu deleted chỉ được để ý khi nào cần roll back hay audit mà thôi. Trong trường hợp này ta có thể tạo ra một view select data mà Deleted=False và làm việc chủ yếu trên view thay vì toàn bộ table.

Đơn giản hóa việc xử lý data: Đôi khi ta có những query rất phức tạp và sử dụng thường xuyên ta có thể chuyển nó thành View và đối xử nó như một table, như vậy sẽ làm cho việc xử lý data dễ dàng hơn.

Customize data: Ta có thể dùng view để làm cho users thấy data từ những góc độ khác nhau mặc dù họ đang dùng một nguồn data giống nhau. *Ví dụ*: Ta có thể tạo ra views trong đó những thông tin về customer được thể hiện khác nhau tùy login ID là normal user hay manager.

Export và Import data: Đôi khi ta muốn export data từ SQL Server sang các ứng dụng khác như Excel chẳng hạn ta có thể dùng view để join nhiều table và export dùng bcp.

Khi sử dụng view ta có thể select,insert, update, delete data bình thường như với một table.

Ví dụ:

```
Select * From OrderReport
Where DateProcessed <'2003-01-01'
```

Lưu ý: Trong Enterprise Edition (và Developer Edition) ta có thể tạo Index cho View như cho table. Index sẽ được bàn đến trong các bài sau.

8.2 Khung nhìn đơn giản

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.

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

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.

8.3 Tạo khung nhìn - Khung nhìn như bộ lọc

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: 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
```

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: 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
```

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:

§ 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.

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

Ví dụ: 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
```

8.4. 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: Xét định nghĩa hai bảng DONVI và NHANVIEN như sau:

```
CREATE TABLE donvi
(
madv        INT                 PRIMARY KEY,
tendv  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
                                REFERENCES donvi(madv)
                                ON DELETE CASCADE
                                ON UPDATE CASCADE
)
```

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

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 9.6: 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 : 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')
```

8.5. 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 : 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'
```

8.6. 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 VIEW tên_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 : Câu lệnh dưới đây xoá khung nhìn VIEW_LOP ra khỏi cơ sở dữ liệu

DROP VIEW view_lop

Bài 9
THIẾT KẾ CƠ SỞ DỮ LIỆU

Mục tiêu:

- Thiết và tạo được CSDL;
- Xây dựng được mô hình CSDL;
- Backup và restore được CSDL.
- Thực hiện các thao tác an toàn với máy tính.

Nội dung chính:

9.1. Cấu Trúc Của SQL Server

Như đã trình bày ở các bài trước một trong những đặc điểm của SQL Server 2000 là Multiple-Instance nên khi nói đến một (SQL) Server nào đó là ta nói đến một Instance của SQL Server 2000, thông thường đó là Default Instance. Một Instance của SQL Server 2000 có 4 system databases và một hay nhiều user database. Các system databases bao gồm:

• Master : Chứa tất cả những thông tin cấp hệ thống (system-level information) bao gồm thông tin về các database khác trong hệ thống như vị trí của các data files, các login account và các thiết đặt cấu hình hệ thống của SQL Server (system configuration settings).

• Tempdb : Chứa tất cả những table hay stored procedure được tạm thời tạo ra trong quá trình làm việc bởi user hay do bản thân SQL Server engine. Các table hay stored procedure này sẽ biến mất khi khởi động lại SQL Server hay khi ta disconnect.

• Model : Database này đóng vai trò như một bảng kẻm (template) cho các database khác. Nghĩa là khi một user database được tạo ra thì SQL Server sẽ copy toàn bộ các system objects (tables, stored procedures...) từ Model database sang database mới vừa tạo.

• Msdb : Database này được SQL Server Agent sử dụng để hoạch định các báo động và các công việc cần làm (schedule alerts and jobs).

9.2. Cấu Trúc Vật Lý Của Một SQL Server Database

Mỗi một database trong SQL Server đều chứa ít nhất một data file chính (primary), có thể có thêm một hay nhiều data file phụ (Secondary) và một transaction log file.

• Primary data file (thường có phần mở rộng .mdf) : đây là file chính chứa data và những system tables.

• Secondary data file (thường có phần mở rộng .ndf) : đây là file phụ thường chỉ sử dụng khi database được phân chia để chứa trên nhiều dĩa.

• Transaction log file (thường có phần mở rộng .ldf) : đây là file ghi lại tất cả những thay đổi diễn ra trong một database và chứa đầy đủ thông tin để có thể roll back hay roll forward khi cần.

Data trong SQL Server được chứa thành từng Page 8KB và 8 page liên tục tạo thành một Extent.

Trước khi SQL Server muốn lưu data vào một table nó cần phải dành riêng một khoảng trống trong data file cho table đó. Những khoảng trống đó chính là các extents. Có 2 loại Extents: Mixed Extents (loại hỗn hợp) dùng để chứa data của nhiều tables trong cùng một Extent và Uniform Extent (loại thuần nhất) dùng để chứa data của một table. Đầu tiên SQL Server dành các Page trong Mixed Extent để chứa data cho một table sau đó khi data tăng trưởng thì SQL dành hẳn một Uniform Extent cho table đó.

9.3. Nguyên Tắc Hoạt Động Của Transaction Log Trong SQL Server

Transaction log file trong SQL Server dùng để ghi lại các thay đổi xảy ra trong database. Quá trình này diễn ra như sau: đầu tiên khi có một sự thay đổi data như Insert, Update, Delete được yêu cầu từ các ứng dụng, SQL Server sẽ tải (load) data page tương ứng lên memory (vùng bộ nhớ này gọi là data cache), sau đó data trong data cache được thay đổi(những trang bị thay đổi còn gọi là dirty-page). Tiếp theo mọi sự thay đổi đều được ghi vào transaction log file cho nên người ta gọi là write-ahead log. Cuối cùng thì một quá trình gọi là Check Point Process sẽ kiểm tra và viết tất cả những transaction đã được commited (hoàn tất) vào dĩa cứng (flushing the page).

Ngoài Check Point Process những dirty-page còn được đưa vào dĩa bởi một Lazy writer. Đây là một anh chàng làm việc âm thầm chỉ thức giấc và quét qua phần data cache theo một chu kỳ nhất định sau đó lại ngủ yên chờ lần quét tới.

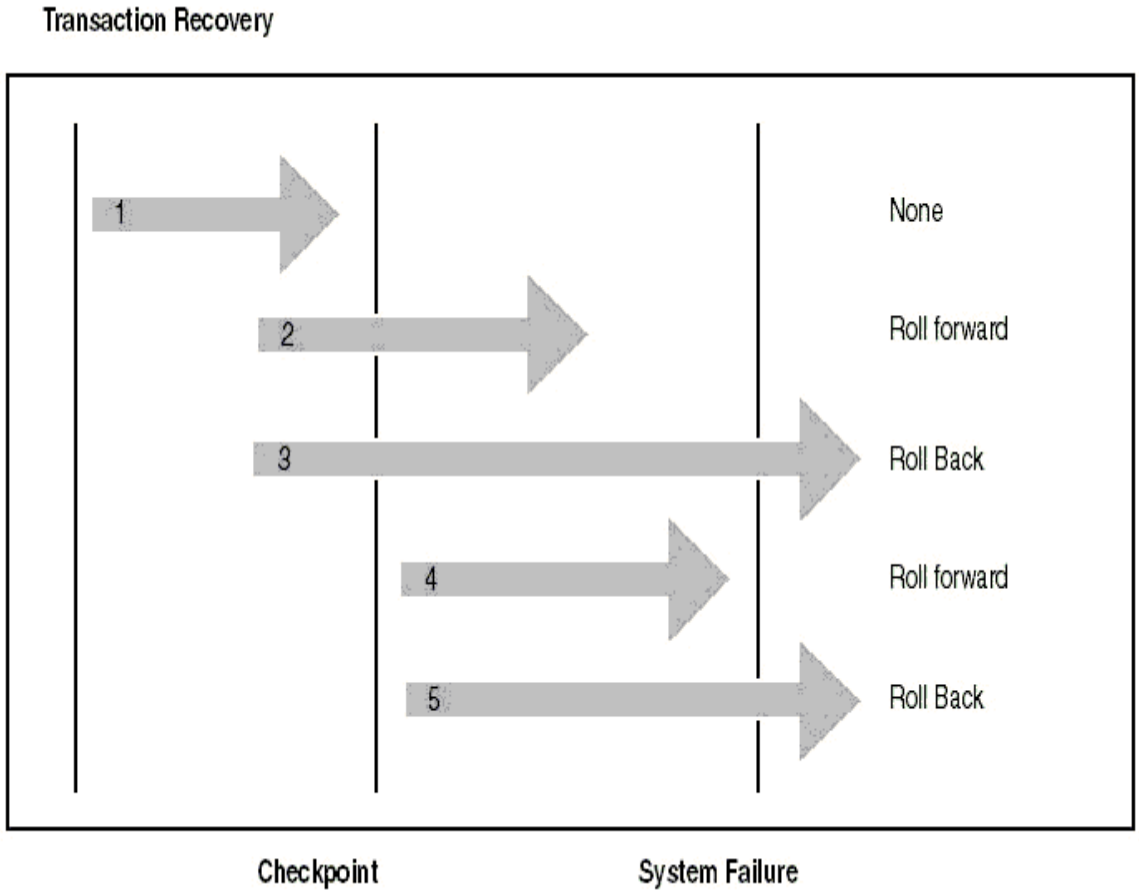
Xin giải thích thêm một chút về khái niệm transaction trong database. Một transaction hay một giao dịch là một loạt các hoạt động xảy ra được xem như một công việc đơn (unit of work) nghĩa là hoặc thành công toàn bộ hoặc không làm gì cả (all or nothing). Sau đây là một *Ví dụ* cổ điển về transaction:

Chúng ta muốn chuyển một số tiền $500 từ account A sang account B như vậy công việc này cần làm các bước sau:

1. Trừ $500 từ account A
2. Cộng $500 vào account B

Tuy nhiên việc chuyển tiền trên phải được thực hiện dưới dạng một transaction nghĩa là giao dịch chỉ được xem là hoàn tất (commited) khi cả hai bước trên đều thực hiện thành công. Nếu vì một lý do nào đó ta chỉ có thể thực hiện được bước 1 (chẳng hạn như vừa xong bước 1 thì điện cúp hay máy bị treo) thì xem như giao dịch không hoàn tất và cần phải được phục hồi lại trạng thái ban đầu (roll back).

Thế thì Check Point Process hoạt động như thế nào để có thể đảm bảo một transaction được thực thi mà không làm "dơ" database.



Trong hình vẽ trên, một transaction được biểu diễn bằng một mũi tên. Trục nằm ngang là trục thời gian. Giả sử một Check Point được đánh dấu vào thời điểm giữa transaction 2 và 3 như hình vẽ và sau đó sự cố xãy ra trước khi gặp một Check point kế tiếp. Như vậy khi SQL Server được restart nó sẽ dựa trên những gì ghi trong transaction log file để phục hồi data (xem hình vẽ).

Điều đó có nghĩa là SQL Server sẽ không cần làm gì cả đối với transaction 1 vì tại thời điểm Check point data đã được lưu vào đĩa rồi. Trong khi đó transaction 2 và 4 sẽ được roll forward vì tuy đã được commited nhưng do sự cố xảy ra trước thời điểm check point kế tiếp nên data chưa kịp lưu vào đĩa. Tức là dựa trên những thông tin được ghi trên log file SQL Server hoàn

toàn có đầy đủ cơ sở để viết vào dĩa cứng. Còn transaction 3 và 5 thì chưa được commited (do bị down bất ngờ) cho nên SQL Server sẽ roll back hai transaction này dựa trên những gì được ghi trên log file.

9.4. Cấu Trúc Logic Của Một SQL Server Database

Hầu như mọi thứ trong SQL Server được tổ chức thành những objects ví dụ như tables, views, stored procedures, indexes, constraints.... Những system objects trong SQL Server thường có bắt đầu bằng chữ sys hay sp. Các objects trên sẽ được nghiên cứu lần lượt trong các bài sau do đó trong phần này chúng ta chỉ bàn sơ qua một số system object thông dụng trong SQL Server database mà thôi.

Một số Sytem objects thường dùng:

System Stored Procedure	Ứng dụng
Sp_help *['object']*	Cung cấp thông tin về một database object (table, view...) hay một data type.
Sp_helpdb *['database']*	Cung cấp thông tin về một database cụ thể nào đó.
Sp_monitor	Cho biết độ bận rộn của SQL Server
Sp_spaceused *['object', 'updateusage']*	Cung cấp thông tin về các khoảng trống đã được sử dụng cho một object nào đó
Sp_who ['*login*']	Cho biết thông tin về một SQL Server user

9.5. Tạo Một User Database

Chúng ta có thể tạo một database dễ dàng dùng SQL Server Enterprise bằng cách right-click lên trên "database" và chọn "New Database".

Ngoài ra đôi khi chúng ta cũng dùng SQL script để tạo một database. Khi đó ta phải chỉ rõ vị trí của primary data file và transaction log file.

Ví dụ:

```
USE master
GO
CREATE DATABASE Products
ON
( NAME = prods_dat,
        FILENAME = 'c:\program files\microsoft SQL server\mssql\data\prods.mdf',
  SIZE = 4,
```

```
    MAXSIZE = 10,
    FILEGROWTH = 1
)
GO
```

Trong *Ví dụ* trên ta tạo một database tên là Products với logical file name là prods_dat và physical file name là prods.mdf, kích thước ban đầu là 4 MB và data file sẽ tự động tăng lên mỗi lần 1 MB cho tới tối đa là 10 MB. Nếu ta không chỉ định một transaction log file thì SQL sẽ tự động tạo ra 1 log file với kích thước ban đầu là 1 MB.

Lưu Ý:

Khi tạo ra một database chúng ta cũng phải lưu ý một số điểm sau: Đối với các hệ thống nhỏ mà ở đó vấn đề tốc độ của server không thuộc loại nhạy cảm thì chúng ta thường chọn các giá trị mặc định (default) cho Initial size, Automatically growth file. Nhưng trên một số production server của các hệ thống lớn kích thước của database phải được người DBA ước lượng trước tùy theo tầm cỡ của business, và thông thường người ta không chọn Autogrowth(tự động tăng trưởng) và Autoshrink(tự động nén). Câu hỏi được đặt ra ở đây là vì sao ta không để SQL Server chọn một giá trị khởi đầu cho datafile và sau đó khi cần thì nó sẽ tự động nới rộng ra mà lại phải ước lượng trước? Nguyên nhân là nếu chọn Autogrowth (hay Autoshrink) thì chúng ta có thể sẽ gặp 2 vấn đề sau:

• Performance hit: Ảnh hưởng đáng kể đến khả năng làm việc của SQL Server. Do nó phải thường xuyên kiểm tra xem có đủ khoảng trống cần thiết hay không và nếu không đủ nó sẽ phải mở rộng bằng cách dành thêm khoảng trống từ dĩa cứng và chính quá trình này sẽ làm chậm đi hoạt động của SQL Server.

• Disk fragmentation : Việc mở rộng trên cũng sẽ làm cho data không được liên tục mà chứa ở nhiều nơi khác nhau trong dĩa cứng điều này cũng gây ảnh hưởng lên tốc độ làm việc của SQL Server.

Trong các hệ thống lớn người ta có thể dự đoán trước kích thước của database bằng cách tính toán kích thước của các tables, đây cũng chỉ là kích thước ước đoán mà thôi (xin xem "Estimating the size of a database" trong SQL Books Online để biết thêm về cách tính) và sau đó thường xuyên dùng một số câu lệnh SQL (thường dùng các câu lệnh bắt đầu bằng DBCC .Phần này sẽ được bàn qua trong các bài sau) kiểm tra xem có đủ khoảng trống hay không nếu không đủ ta có thể chọn một thời điểm mà SQL server ít bận rộn nhất (như ban đêm hay sau giờ làm việc) để nới rộng data file như thế sẽ không làm ảnh hưởng tới performance của Server.

Chú ý giả sử ta dành sẵn 2 GB cho datafile, khi dùng Window Explorer để xem ta sẽ thấy kích thước của file là 2 GB nhưng data thực tế có thể chỉ chiếm vài chục MB mà thôi.

9.6. Những Điểm Cần Lưu Ý Khi Thiết Kế Một Database

Trong phạm vi bài này chúng ta không thể nói sâu về lý thuyết thiết kế database mà chỉ đưa ra một vài lời khuyên mà bạn nên tuân theo khi thiết kế.

Trước hết bạn phải nắm vững về các loại data type. *Ví dụ* bạn phải biết rõ sự khác biệt giữa char(10), nchar(10) varchar(10), nvarchar(10). Loại dữ liệu Char là một loại string có kích thước cố định nghĩa là trong *Ví dụ* trên nếu data đưa vào "This is a really long character string" (lớn hơn 10 ký tự) thì SQL Server sẽ tự động cắt phần đuôi và ta chỉ còn "This is a". Tương tự nếu string đưa vào nhỏ hơn 10 thì SQL sẽ thêm khoảng trống vào phía sau cho đủ 10 ký tự. Ngược lại loại varchar sẽ không thêm các khoảng trống phía sau khi string đưa vào ít hơn 10. Còn loại data bắt đầu bằng chữ n chứa dữ liệu dạng unicode.

Một lưu ý khác là trong SQL Server ta có các loại Integer như : tinyint, smallint, int, bigint. Trong đó kích thước từng loại tương ứng là 1,2,4,8 bytes. Nghĩa là loại smallint tương đương với Integer và loại int tương đương với Long trong VB.

Khi thiết kế table nên:

• Có ít nhất một cột thuộc loại ID dùng để xác định một record dễ dàng.

• Chỉ chứa data của một entity (một thực thể)

Ngoài ra một trong những điều quan trọng là phải biết rõ quan hệ (Relationship) giữa các table:

• One-to-One Relationships : trong mối quan hệ này thì một hàng bên table A không thể liên kết với hơn 1 hàng bên table B và ngược lại.

• One-to-Many Relationships : trong mối quan hệ này thì một hàng bên table A có thể liên kết với nhiều hàng bên table B.

• Many-to-Many Relationships : trong mối quan hệ này thì một hàng bên table A có thể liên kết với nhiều hàng bên table B và một hàng bên table B cũng có thể liên kết với nhiều hàng bên table A. Như ta thấy trong *Ví dụ* trên một cuốn sách có thể được viết bởi nhiều tác giả và một tác giả cũng có thể viết nhiều cuốn sách. Do đó mối quan hệ giữa Books và Authors là quan hệ Many to Many. Trong trường hợp này người ta thường dùng một table trung gian để giải quyết vấn đề (table AuthorBook).

Để có một database tương đối hoàn hảo nghĩa là thiết kế sao cho data chứa trong database không thừa không thiếu bạn cần biết thêm về các thủ

thuật Normalization. Tuy nhiên trong phạm vi khóa học này chúng tôi không muốn bàn sâu hơn về đề tài này, bạn có thể xem thêm trong các sách dạy lý thuyết cơ sở dữ liệu.

Tóm lại trong bài này chúng ta đã tìm hiểu về cấu trúc của một SQL Server database và một số vấn đề cần biết khi thiết kế một database.

9.7 bài tập

CÁC THUẬT NGỮ CHUYÊN MÔN

Thuật ngữ	Giải thích
Database	Cơ sở dữ liệu

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] Phạm Hữu Khang, Lập trình ứng dụng chuyên nghiệp SQL Server, Nhà xuất bản Lao động xã hội, 2003.

[2] Microsoft SQL Server 2005 Book Online.

[3] Đỗ Phúc, Tài liệu môn học CSDL nâng cao , Đại học Công nghệ thông tin, Đại học Quốc gia Tp HCM, 2006.

[4] Trần Đức Quang, Nguyên lý các hệ cơ sở dữ liệu phân, Nhà xuất bản Thống kê, tập 1 và 2, 1999.

[5] Phạm Thế Quế, Giáo trình cơ sở dữ liệu phân tán, Học viên công nghệ bưu chính viễn thông, 2010.

[6] Lê Văn Sơn, Giáo trình Hệ tin học phân tán cho học viên Cao học CNTT, Đại học Đà Nẵng, Đà Nẵng, 1999.

[7] Lê Văn Sơn, Hệ tin học phân tán, Nhà xuất bản đại học quốc gia TP HCM, 2002.

[8] Nguyễn Bá Tường, Nhập môn cơ sở dữ liệu phân tán, Nhà xuất bản Khoa học Kỹ thuật, 2005.

[9] Trần Quốc Chiến, Giáo trình cơ sở dữ liệu nâng cao, Đại học Đà Nẵng, 2007.

Tiếng Anh

[10] M.Tamer Ozsu and Patrick Vaduries, Principles of Distributed Database System, Frentice-Hall, 2006.

Trang Web

[11] http://www.lirc.udn.vn/

[12] http://msdn.microsoft.com

DANH SÁCH BAN BIÊN SOẠN GIÁO TRÌNH DẠY NGHỀ

TRÌNH ĐỘ TRUNG CẤP, CAO ĐẲNG

Tên giáo trình: QUẢN TRỊ CƠ SỞ DỮ LIỆU NÂNG CAO
Tên nghề: QUẢN TRỊ MẠNG

1. Ông (bà).....	Chủ nhiệm
2. Ông (bà).....	Phó chủ nhiệm
3. Ông (bà).....	Thư ký
4. Ông (bà).....	Thành viên
5. Ông(bà).....	Thành viên
6. Ông(bà).....	Thành viên
7. Ông(bà).....	Thành viên
8. Ông(bà).....	Thành viên
9. Ông(bà).....	Thành viên

DANH SÁCH HỘI ĐỒNG NGHIỆM THU

GIÁO TRÌNH DẠY NGHỀ TRÌNH ĐỘ TRUNG CẤP, CAO ĐẲNG

1. Ông (bà).....	Chủ tịch
2. Ông (bà).....	Phó chủ tịch
3. Ông (bà).....	Thư ký
4. Ông (bà).....	Thành viên
5. Ông(bà).....	Thành viên
6. Ông(bà).....	Thành viên
7. Ông(bà).....	Thành viên
8. Ông(bà).....	Thành viên
9. Ông(bà).....	Thành viên

Phụ lục 3.2

CÁC TIÊU CHÍ VÀ TIÊU CHUẨN ĐÁNH GIÁ CHẤT LƯỢNG GIÁO TRÌNH DẠY NGHỀ TRÌNH ĐỘ TRUNG CẤP VÀ CAO ĐẲNG

Số TT	Các tiêu chí đánh giá	Mức độ đánh giá			Ghi chú
		Đạt yêu cầu đề nghị ban hành ngay	Đạt yêu cầu nhưng phải chỉnh sửa	Chưa đạt yêu cầu phải xây dựng lại	
A	Sự tương ứng với chương trình				
1	Giáo trình có đủ các đề mục và thể hiện nội dung theo đúng mẫu định dạng				
2*	Giáo trình có đầy đủ các nội dung theo chương trình chi tiết các mô đun trong chương trình đào tạo				
3*	Nội dung các chương/bài đảm bảo mục tiêu kiến thức, kỹ năng đã đề ra không?				
4*	Khối lượng các thông tin trong các mô đun có phù hợp với thời lượng của chương trình không?				
B	Tính logíc				
5*	Nội dung từng chương/bài có được trình bày một cách logic với quá trình nhân thức không? (tức là: Mức độ từ dễ đến khó, tính trình tự cho các khái niệm từ đơn giản đến phức tạp)				
6*	Các bước hình thành kỹ năng có hợp lý và vừa phải không? (tức là quan sát mẫu - bắt trước - làm được - làm độc lập - làm thuần thục hoặc theo đường xoắn ốc để hình thành các kỹ xảo)				
7*	Mối quan hệ giữa lý thuyết và thực hành có hợp lý để bảo đảm				

	được sự nhận thức và kiến thức, sự hình thành kỹ năng không?				
8*	Hình thức học tập và các giải pháp sư phạm cho từng chủ đề có thích hợp so với mục tiêu đã đề ra không?				
C	Mức đầy đủ/bao quát đối với mục tiêu				
9*	Nội dung có đầy đủ để đảm bảo đào tạo có kết quả theo các mục tiêu thực hiện không				
10*	Nội dung có được nhấn mạnh để rèn luyện, hình thành các kỹ năng cần thiết không? (Tức là có các quy trình rèn luyện/thực hành bao gồm cả các khía cạnh khác như: tinh thần trách nhiệm, tuân thủ kỷ luật, ý thức an toàn, ứng xử trong nhóm, tác phong công nghiệp…)				
11*	Các cấu phần tạo sự chủ động và học tích cực có đầy đủ không? (tức là đủ các mục: Giới thiệu, hướng dẫn, tự đánh giá, giải thích thuật ngữ, tài liệu tham khảo…)				
12	Có vận dụng được sự hỗ trợ của các trang thiết bị, nguồn học liệu, nguồn l;ực khác cho quá trình học tập của học viên không?				
13	Các hành ảnh minh họa, bảng biểu, bản vẽ, quy trình thực hiện…có đủ ở mức cần thiết, rõ ràng và ăn nhập với đoạn viết không?				
D	Tính chuẩn xác				
14	Nội dung khoa học của thông tin có chính xác không? (về bản chất vấn đề, về các số liệu, về các sự kiện và đường nét…được đề cập trên các đoạn viêt, các bảng biểu và các hình minh họa, bản vẽ..)				
15	Các thuật ngũ có đảm bảo tính phổ thông và nhất quán không?				

E	Phong cách biên soạn				
16	Ý tứ trình bày rõ ràng, sáng sủa, đơn giản và dễ hiểu không?				
17	Cân đối và phù hợp giữa kênh hình và kênh chữ				
18	Có vi phạm gì về văn hóa tập quán của các dân tộc Việt Nam không?				
19	Có sai phạm gì đối với Luật bản quyền không?				
20	Phong cách trình bày có thể hiện tính gợi mở, lôi kéo người học thực hiện công việc không?				
F	Cấu trúc và các chuyên mục				
21	Bố cục có nhất quán trong toàn bộ tài liệu không?				
22	Mối liên hệ giữa các chuyên mục có chặt chẽ và tương ứng với nhau không? (đặc biệt là mục tiêu, kiểm tra đánh giá và các hướng dẫn trả lời)				
23	Mã các chuyên mục, hình vẽ, bảng biểu, bản vẽ…có nhất quán và chính xác và tạo điều kiện thuận lợi cho việc tìm kiếm và liên hệ				

Ghi chú:

1. Các tiêu chí có đánh dấu * có ý nghĩa rất quan trọng đối với chất lượng giáo trình đã biên soạn

2. Các mức độ đánh giá:

- Đạt yêu cầu: Không phải sửa chữa gì hoặc chỉ cần sửa chữa vài lỗi nhỏ về biên tập;
- Đạt yêu cầu nhưng phải chỉnh sửa: Phải sửa chữa một số lỗi về nội dung chuyên môn và biên tập, chỉnh lý, bổ sung; sau đó trình chủ tịch, phó chủ tịch và thư ký hội đồng xem xét, nếu thông qua được thì đạt yêu cầu đề nghị phê duyệt;
- Không đạt yêu cầu: Có nhiều lỗi về nội dung chuyên môn và biên tập, phải biên soạn lại để trình Hội đồng thẩm định lại.