BÀI 6
LẬP TRÌNH TRÊN SQL SERVER

MÃ BÀI: ITPRG3-17.6

Giới thiệu:

Hệ quản trị cơ sở dữ liệu SQL Server cung cấp cho ta một công cụ khá mạnh cho phép người quản trị có thể lập trình quản lý dữ liệu SQL nhờ ngôn ngữ SQL, người ta gọi là ngôn ngữ truy vấn có cấu trúc. Việc lập trình trên SQL Server cho phép chúng ta thực hiện các công việc có hiệu quả hơn, chẳng hạn các ràng buộc được chặc chẽ và toàn vẹn hơn, các thao tác được bảo mật hơn, …

Mục tiêu thực hiện:

Học xong bài này học viên sẽ có khả năng:

- Sử dụng được các câu lệnh SQL trong Analysis
- Sử dụng được các tập tin batch
- Sử dụng được thủ tục lưu trữ trong lập trình ứng dụng
- Tạo được Trigger cho các bảng cơ sở dữ liệu

Nội dung:

6.1 Các câu lệnh SQL Server

6.1.1 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]]
```

1) Mệnh đề FROM

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

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

```
SELECT * FROM khoa
```

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.

a. Chọn tất cả các cột trong bảng

Ví dụ: Câu lệnh

SELECT * FROM lop

b. Tên cột trong danh sách chọn

Ví dụ: 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:

SELECT malop, tenlop, lop.makhoa, tenkhoa

FROM lop, khoa

WHERE lop.malop = khoa.makhoa

c. Thay đổi tiêu đề các cột

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: Câu lệnh dưới đây:

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

FROM lop

d. Sử dụng cấu trúc CASE trong danh sách chọn

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: 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 như nhau.

e. Hằng và biểu thức trong danh sách chọn

Ví dụ: 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: 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

Ta chỉ định thêm từ khóa DISTINCT ngay sau từ khoá SELECT.

Ví dụ: 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

Ta chỉ định thêm mệnh đề TOP ngay trước danh sách chọn của câu lệnh SELECT.

Ví dụ: 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: 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

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.

* 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)o 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

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: 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)

Ví dụ: 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:

% :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: Câu lệnh dưới đây

```
SELECT hodem,ten FROM sinhvien
WHERE hodem LIKE 'Lê%'
```

Câu lệnh:

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

e. Giá trị NULL: là một giá trị đặc biệt trong SQL Server dùng để chỉ một giá trị dữ liệu không chứa gì cả.

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:

```
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.

5) Sắp xếp kết quả truy vấn

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
```

6) Phép hợp

Cú pháp như sau:

```
Câu_lệnh_1
UNION [ALL] Câu_lệnh_2
[UNION [ALL] Câu_lệnh_3]
...
[UNION [ALL] Câu_lệnh_n]
[ORDER BY cột_sắp_xếp]
[COMPUTE danh_sách_hàm_gộp [BY danh_sách_cột]]
```

Trong đó

```
Câu_lệnh_1 có dạng
SELECT danh_sách_cột
[INTO tên_bảng_mới]
[FROM danh_sách_bảng|khung_nhìn]
[WHERE điều_kiện]
[GROUP BY danh_sách_cột]
[HAVING điều_kiện]
```

và Câu_lệnh_i (i = 2,..,n) có dạng

```
SELECT danh_sách_cột
[FROM danh_sách_bảng|khung_nhìn]
[WHERE điều_kiện]
[GROUP BY danh_sách_cột]
[HAVING điều_kiện]
```

7) Phép nối

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

Xét hai bảng sau đây:

Bảng KHOA

Bảng LOP

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

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

Như vậy, để thực hiện được yêu cầu truy vấn dữ liệu trên, ta phải thực hiện phép nối giữa hai bảng KHOA và LOP với điều kiện nối là MAKHOA của KHOA bằng với MAKHOA của LOP.

a) Sử dụng phép nối

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

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

b) Các loại phép nối

Phép nối bằng và phép nối tự nhiên

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

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

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

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

hoặc viết dưới dạng ngắn gọn hơn:

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

WHERE lop.makhoa=khoa.makhoa

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:

o Hàm SUM và AVG chỉ làm việc với các biểu thức số.

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

o Hàm COUNT(*) không bỏ qua các giá trị NULL.

Ví dụ: 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 Dalat:

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

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: 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: 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.

6.1.2 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:

o Lệnh **INSERT**

o Lệnh **UPDATE**

o Lệnh **DELETE**

1) Bổ sung dữ liệu

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

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

o + 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: 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 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: Câu lệnh dưới đây bổ sung một bản ghi mới cho bảng SINHVIEN

```
INSERT INTO sinhvien(masv,hodem,ten,gioitinh,malop)
VALUES('0241020008','Nguyễn Công','Chính',1,'C24102')
```

câu lệnh trên còn có thể được viết như sau:

```
INSERT INTO sinhvien
VALUES('0241020008','Nguyễn Công','Chính',
NULL,1,NULL,'C24102')
```

Bổ sung nhiều dòng dữ liệu từ bảng khác

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

Cú pháp câu lệnh INSERT có dạng như sau:

```
INSERT INTO tên_bảng[(danh_sách_cột)] câu_lệnh_SELECT
```

Ví dụ: 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:

o 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.

2) Cập nhật dữ liệu

Câu lệnh UPDATE trong SQL được sử dụng để cập nhật dữ liệu trong các bảng.

Câu lệnh này có cú pháp như sau:

UPDATE tên_bảng

SET tên_cột = biểu_thức

[, ..., tên_cột_k = biểu_thức_k]

[FROM danh_sách_bảng]

[WHERE điều_kiện]

Sau UPDATE là tên của bảng cần cập nhật dữ liệu. Một câu lệnh UPDATE có thể cập nhật dữ liệu cho nhiều cột bằng cách chỉ định các danh sách tên cột và biểu thức tương ứng sau từ khoá SET. Mệnh đề WHERE trong câu lệnh UPDATE thường được sử dụng để chỉ định các dòng dữ liệu chịu tác động của câu lệnh (nếu không chỉ định, phạm vi tác động của câu lệnh được hiểu là toàn bộ các dòng trong bảng)

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

UPDATE monhoc

SET sodvht = 3

WHERE sodvht = 2

Sử dụng cấu trúc CASE trong câu lệnh UPDATE

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

Ví dụ: Giả sử ta có bảng NHATKYPHONG sau đây

Sau khi thực hiện câu lệnh:

UPDATE NHATKYPHONG

SET tienphong=songay*CASE WHEN loaiphong='A' THEN 100

WHEN loaiphong='B' THEN 70

ELSE 50

END

Điều kiện cập nhật dữ liệu liên quan đến nhiều bảng

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

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

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

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

Câu lệnh UPDATE với truy vấn con

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

Ví dụ: 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)
```

3) Xoá dữ liệu

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

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

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

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

Ví dụ: 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: Câu lệnh dưới đây xoá ra khỏi bảng SINHVIEN những sinh viên lớp Tin K3

```
DELETE FROM sinhvien
FROM lop
WHERE lop.malop=sinhvien.malop AND tenlop='Tin K3'
```

Thay vì sử dụng câu lệnh DELETE, ta có thể sử dụng câu lệnh TRUNCATE có cú pháp như sau:

```
TRUNCATE TABLE tên_bảng
```

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

```
DELETE FROM diemthi
```

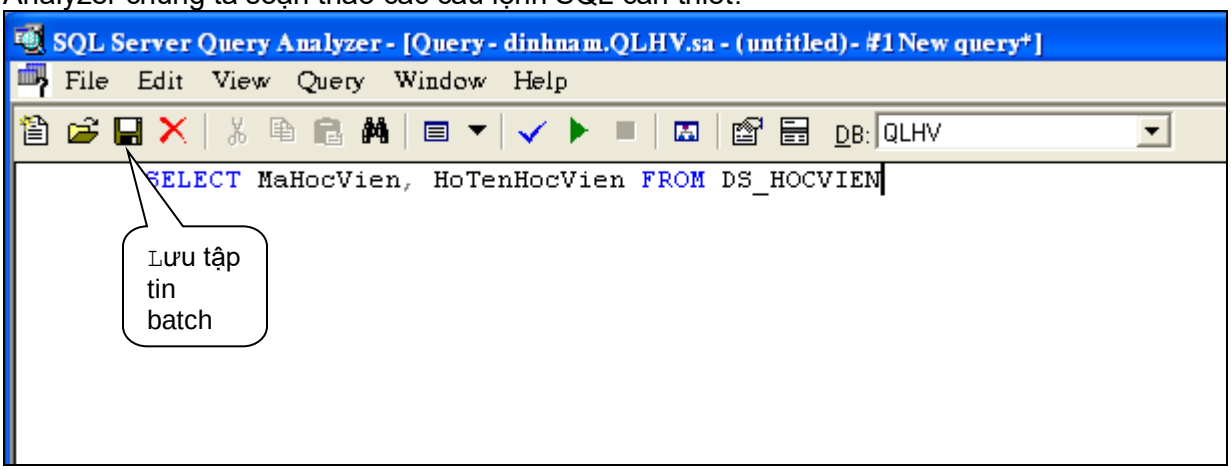
có tác dụng tương tự với câu lệnh

```
TRUNCATE TABLE diemthi
```

6.2 Tập tin batch

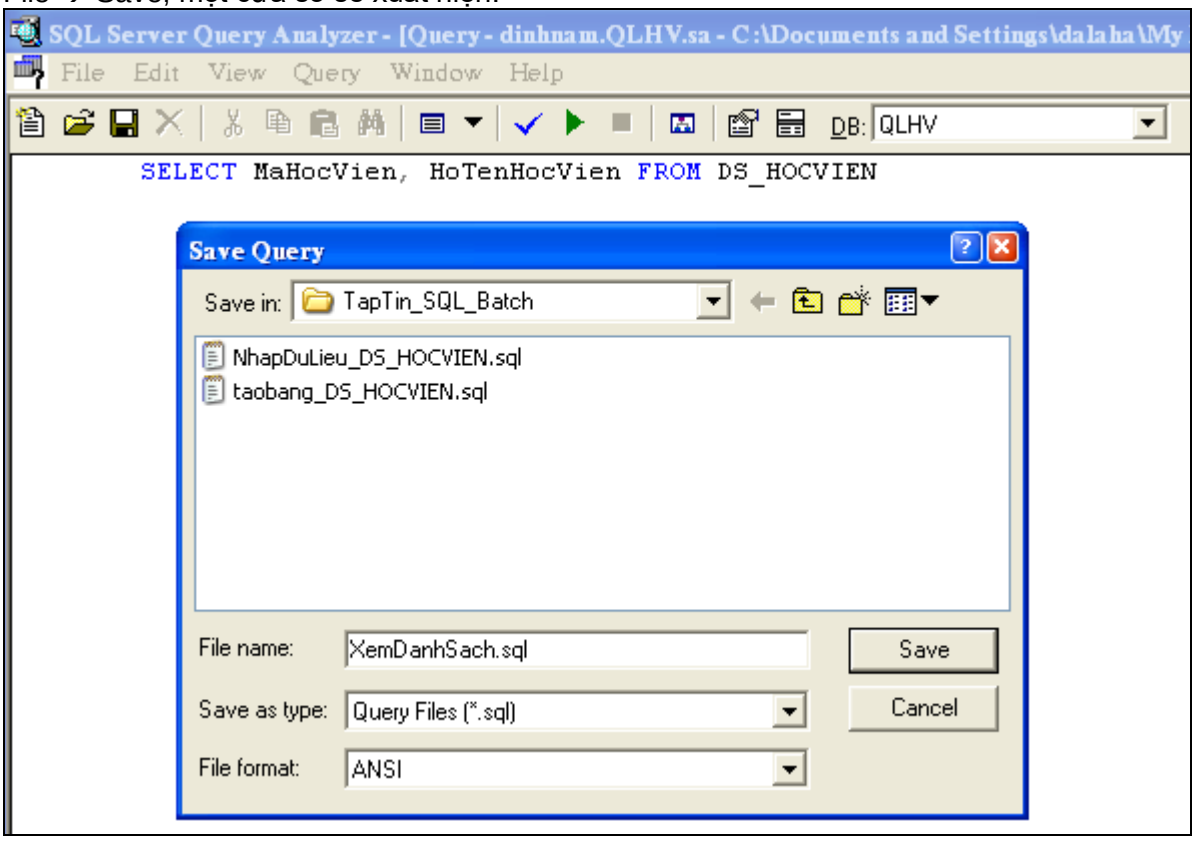
- Tập tin batch là tập tin văn bản chứa các câu lệnh SQL được lập trình sẵn, giúp việc cài đặt và sao chép các ứng dụng sử dụng cơ sở dữ liệu SQL mang lại hiệu quả nhanh chóng và dễ dàng.

+ Để tạo ra một tập tin batch trong môi trường SQL Server, trong cửa sổ Query Analyzer chúng ta soạn thảo các câu lệnh SQL cần thiết:



Hình 6.1 : Tập tin batch

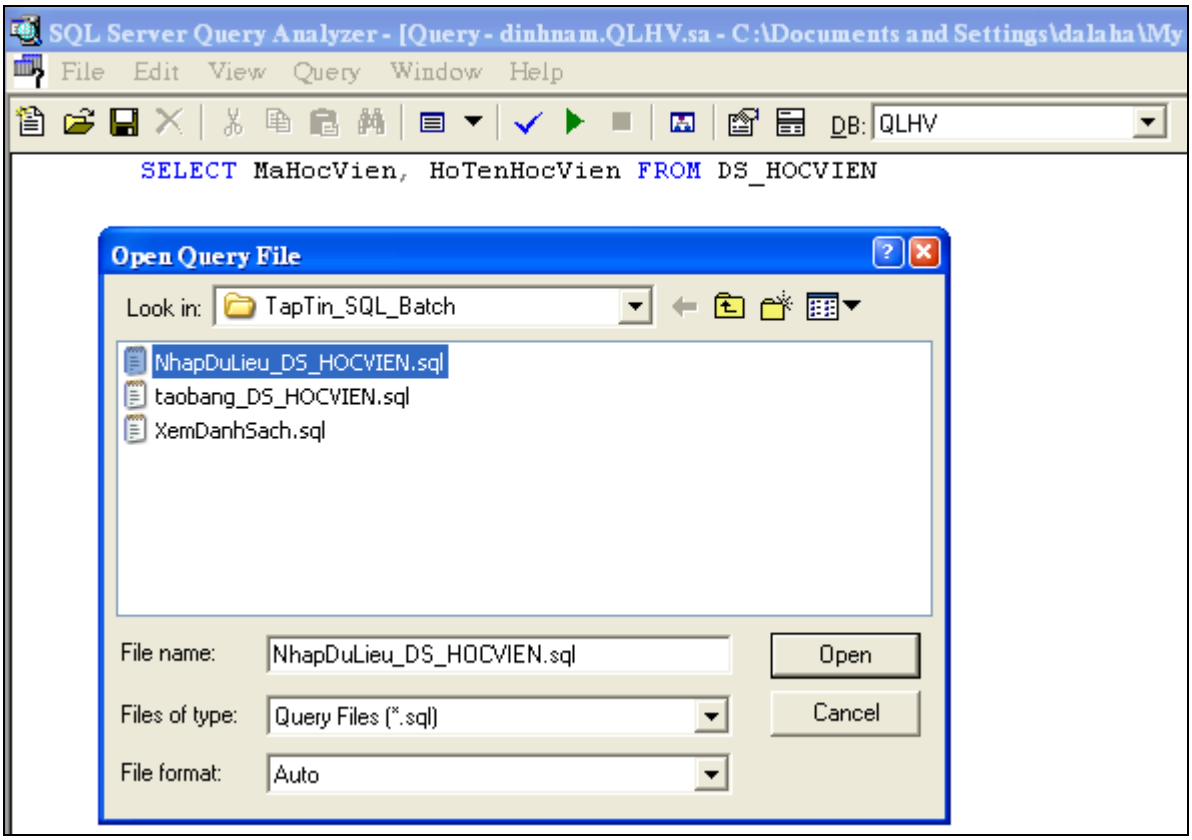
sau đó nhấn vào nút Save (hình đĩa mềm) trên thanh công cụ hoặc vào thực đơn File → Save, một cửa sổ sẽ xuất hiện:



Hình 6.2 : Lưu query

Mặc định, tập tin batch sẽ tự động lưu với phần mở rộng .sql, người sử dụng chỉ cần nhập tên tập tin, sau đó nhấn vào nút Save để lưu tập tin lên thư mục hoặc ổ đĩa.

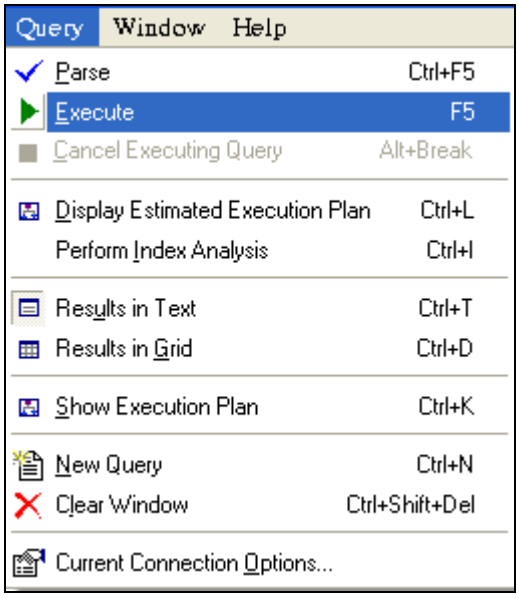
+ Để mở một tập tin batch, chúng ta nhấn vào nút Open trên thanh công cụ hoặc vào thực đơn File→ Open, một cửa sổ sẽ xuất hiện:



Hình 6.3: Mở tập query

Chọn tập tin batch cần mở, sau đó nhấn nút Open.

+ Để thực hiện tập tin batch, chúng ta có thể nhấn phím F5 hoặc vào thực đơn Query → Execute:



Hình 6.4 :Execute

6.3 Stored Procedure (thủ tục lưu trữ)

6.3.1 Các khái niệm

Như đã đề cập ở các chương trước, SQL được thiết kế và cài đặt như là một ngôn ngữ để thực hiện các thao tác trên cơ sở dữ liệu như tạo lập các cấu trúc trong cơ sở dữ liệu, bổ sung, cập nhật, xoá và truy vấn dữ liệu trong cơ sở dữ liệu. Các câu lệnh SQL được người sử dụng viết và yêu cầu hệ quản trị cơ sở dữ liệu thực hiện theo chế độ tương tác.

Các câu lệnh SQL có thể được nhúng vào trong các ngôn ngữ lập trình, thông qua đó chuỗi các thao tác trên cơ sở dữ liệu được xác định và thực thi nhờ vào các câu lệnh, các cấu trúc điều khiển của bản thân ngôn ngữ lập trình được sử dụng.

Với thủ tục lưu trữ, một phần nào đó khả năng của ngôn ngữ lập trình được đưa vào trong ngôn ngữ SQL. Một thủ tục là một đối tượng trong cơ sở dữ liệu bao gồm một tập nhiều câu lệnh SQL được nhóm lại với nhau thành một nhóm với những khả năng sau:

- Các cấu trúc điều khiển (IF, WHILE, FOR) có thể được sử dụng trong thủ tục.
- Bên trong thủ tục lưu trữ có thể sử dụng các biến như trong ngôn ngữ lập trình nhằm lưu giữ các giá trị tính toán được, các giá trị được truy xuất được từ cơ sở dữ liệu.
- Một tập các câu lệnh SQL được kết hợp lại với nhau thành một khối lệnh bên trong một thủ tục. Một thủ tục có thể nhận các tham số truyền vào cũng như có thể trả về các giá trị thông qua các tham số (như trong các ngôn ngữ lập trình). Khi một thủ tục lưu trữ đã được định nghĩa, nó có thể được gọi thông qua tên thủ tục, nhận các tham số truyền vào, thực thi các câu lệnh SQL bên trong thủ tục và có thể trả về các giá trị sau khi thực hiện xong.

Sử dụng các thủ tục lưu trữ trong cơ sở dữ liệu sẽ giúp tăng hiệu năng của cơ sở dữ liệu, mang lại các lợi ích sau:

- Đơn giản hoá các thao tác trên cơ sở dữ liệu nhờ vào khả năng module hoá các thao tác này.
- Thủ tục lưu trữ được phân tích, tối ưu khi tạo ra nên việc thực thi chúng nhanh hơn nhiều so với việc phải thực hiện một tập rời rạc các câu lệnh SQL tương đương theo cách thông thường.

• Thủ tục lưu trữ cho phép chúng ta thực hiện cùng một yêu cầu bằng một câu lệnh đơn giản thay vì phải sử dụng nhiều dòng lệnh SQL. Điều này sẽ làm giảm thiểu sự lưu thông trên mạng.

• Thay vì cấp phát quyền trực tiếp cho người sử dụng trên các câu lệnh SQL và trên các đối tượng cơ sở dữ liệu, ta có thể cấp phát quyền cho người sử dụng thông qua các thủ tục lưu trữ, nhờ đó tăng khả năng bảo mật đối với hệ thống.

6.3.2 Tạo thủ tục lưu trữ

Thủ tục lưu trữ được tạo bởi câu lệnh CREATE PROCEDURE với cú pháp như sau:

```
CREATE PROCEDURE tên_thủ_tục [(danh_sách_tham_số)]
[WITH RECOMPILE|ENCRYPTION|RECOMPILE,ENCRYPTION]
AS
Các_câu_lệnh_của_thủ_tục
```

Trong đó:

tên_thủ_tục	Tên của thủ tục cần tạo. Tên phải tuân theo qui tắc định danh và không được vượt quá 128 ký tự.
danh_sách_tham_số	Các tham số của thủ tục được khai báo ngay sau tên thủ tục và nếu thủ tục có nhiều tham số thì các khai báo phân cách nhau bởi dấu phẩy. Khai báo của mỗi một tham số tối thiểu phải bao gồm hai phần: • tên tham số được bắt đầu bởi dấu @. • kiểu dữ liệu của tham số **Ví dụ:** @mamonhoc nvarchar(10)
RECOMPILE	Thông thường, thủ tục sẽ được phân tích, tối ưu và dịch sẵn ở lần gọi đầu tiên. Nếu tuỳ chọn WITH RECOMPILE được chỉ định, thủ tục sẽ được dịch lại mỗi khi được gọi.