

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN

KHOA CÔNG NGHỆ THÔNG TIN



BÀI TẬP THỰC HÀNH HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Trình độ đào tạo : ĐẠI HỌC

Chuyên ngành : KTPM/KHMT/CNTT

Năm học: 2024-2025

BÀI 9: THIẾT KẾ CSDL, ĐẢM BẢO DỮ LIỆU, CHỈ MỤC TRONG CSDL SQL SERVER.....	3
9.1. Mục tiêu.....	3
9.2. Tóm tắt lý thuyết.....	3
9.3. Hướng dẫn thực hành.....	9
9.4. Bài tập tự làm.....	17
BÀI 10. NGÔN NGỮ T - SQL	19
10.1. Mục tiêu	19
10.2. Tóm tắt lý thuyết.....	19
10.3. Hướng dẫn thực hành.....	22
10.4. Bài tập tự làm.....	23
BÀI 11. TRUY VẤN NÂNG CAO; VIEW, CON TRỎ, TRANSACTION	27
11.1. Mục tiêu	27
11.2. Tóm tắt lý thuyết.....	27
11.2.1. Mệnh đề TOP	27
11.2.2. Các phép nối.....	28
11.2.3. Toán tử PIVOT và UNPIVOT	28
11.2.4. Toán tử OUTPUT	29
11.2.5. Thống kê dữ liệu	29
11.2.6. Phân hạng dữ liệu	30
11.2.7. Một số hàm cơ bản trong SQL Server	31
11.3. Hướng dẫn thực hành.....	34
11.4. Bài tập tự làm.....	45
BÀI 12: THỦ TỤC LƯU TRỮ VÀ HÀM DO NGƯỜI DÙNG ĐỊNH NGHĨA	49
12.1. Mục tiêu	49
12.2. Tóm tắt lý thuyết.....	49
12.3. Hướng dẫn thực hành.....	50
12.4. Bài tập tự làm.....	61
BÀI 13: TRIGGER	62
13.1. Mục tiêu	62
13.2. Tóm tắt lý thuyết.....	62
13.3. Hướng dẫn thực hành.....	63
13.4. Bài tập tự làm.....	68
BÀI 14: THỰC HÀNH: KIỂM TRA TỔNG HỢP.....	71
14.1. Mục tiêu	71
14.2. Đề bài tập (Giáo viên phát trực tiếp)	71

BÀI 9: THIẾT KẾ CSDL, ĐẢM BẢO DỮ LIỆU, CHỈ MỤC TRONG CSDL SQL SERVER

9.1. Mục tiêu

Sau bài thực hành này, người học có khả năng:

- ✓ Tạo được một cơ sở dữ liệu (Databases).
- ✓ Thiết kế các bảng (Table) trong cơ sở dữ liệu.
- ✓ Tạo và thiết lập được các ràng buộc (Constraint) trên các bảng (Table) và các cột (Column); thêm, sửa đổi và xóa dữ liệu trong bảng.
- ✓ Quản lý được người dùng trong SQL Server.
- ✓ Thiết kế các chỉ mục, chỉ mục toàn văn và ứng dụng tìm kiếm dữ liệu.

9.2. Tóm tắt lý thuyết

9.2.1 Đảm bảo dữ liệu trong CSDL SQL Server

9.2.1.1. Tạo người dùng, nhóm người dùng

Có thể tạo bằng công cụ hoặc câu lệnh, sau đây là hướng dẫn cách tạo bằng câu lệnh:

USE QLSV -- Mở CSDL có tên QLSV để thực hiện:

-- Tạo các login

Exec SP_ADDLOGIN Sv1, '1'

Exec SP_ADDLOGIN Sv2, '2'

Exec SP_ADDLOGIN Sv3, '3'

-- Tạo user từ login

Exec sp_adduser Sv1, u1

Exec sp_adduser Sv2, u2

Exec sp_adduser Sv3, LanAnh

--Tạo nhóm người dùng (database role)

Exec sp_addrole R1

Exec sp_addrole R2

*/

--thêm các user vào Database role

Exec sp_addrolemember R2, u1

Exec sp_addrolemember R2, u2

9.2.1.2. Cấp phát quyền

a. Cấp phát quyền cho người dùng trên các đối tượng cơ sở dữ liệu

Chỉ có người sở hữu cơ sở dữ liệu hoặc người sở hữu đối tượng cơ sở dữ liệu mới có thể cấp phát quyền cho người dùng trên các đối tượng cơ sở dữ liệu. Câu lệnh GRANT trong trường hợp này có cú pháp như sau:

```
GRANT [PRIVILEGES]| các_quyền_cấp_phát  
[(danh_sách_cột)] ON tên_bảng | tên_khung_nhìn  
|ON tên_bảng | tên_khung_nhìn [(danh_sách_cột)]  
|ON tên_thủ_tục  
|ON tên_hàm  
TO danh_sách_người_dùng | nhóm_người_dùng  
[WITH GRANT OPTION]
```

Trong đó:

- **WITH GRANT OPTION** Cho phép người dùng chuyển tiếp quyền cho người dùng khác.

Privilege	Description
SELECT	Ability to perform SELECT statements on the table.
INSERT	Ability to perform INSERT statements on the table.
UPDATE	Ability to perform UPDATE statements on the table.
DELETE	Ability to perform DELETE statements on the table.
REFERENCES	Ability to create a constraint that refers to the table.
ALTER	Ability to perform ALTER TABLE statements to change the table definition.
EXEC	Procedure
ALL	ALL does not grant all permissions for the table. Rather, it grants the ANSI-92 permissions which are SELECT, INSERT, UPDATE, DELETE, and REFERENCES.

Ví dụ 1: Cấp phát cho người dùng có tên *thuchanh* quyền thực thi các câu lệnh SELECT, INSERT và UPDATE trên bảng LOP

```
GRANT SELECT, INSERT, UPDATE ON lop TO thuchanh
```

Cho phép người dùng *thuchanh* quyền xem họ tên và ngày sinh của các sinh viên (cột Hoten, ngaysinh của bảng SINHVIEN)

```
GRANT SELECT (Hoten, ngaysinh) ON sinhvien TO thuchanh
```

hoặc:

```
GRANT SELECT ON sinhvien (Hoten, ngaysinh) TO thuchanh
```

Với quyền được cấp phát như trên, người dùng *thuchanh* có thể thực hiện câu lệnh sau trên bảng SINHVIEN

```
SELECT Hoten, ngaysinh
```

```
FROM sinhvien
```

Nhưng câu lệnh dưới đây lại không thể thực hiện được

```
SELECT * FROM sinhvien
```

Ví dụ 2: Cho phép người dùng *thuchanh* quyền xem dữ liệu trên bảng SINHVIEN đồng thời có thể chuyển tiếp quyền này cho người dùng khác

```
GRANT SELECT ON sinhvien TO thuchanh
```

```
WITH GRANT OPTION
```

b. Cấp phát quyền thực thi các câu lệnh

Những quyền có thể cấp phát trong trường hợp này bao gồm:

- Tạo cơ sở dữ liệu: CREATE DATABASE.
- Tạo bảng: CREATE TABLE
- Tạo khung nhìn: CREATE VIEW
- Tạo thủ tục lưu trữ: CREATE PROCEDURE
- Tạo hàm: CREATE FUNCTION
- Sao lưu cơ sở dữ liệu: BACKUP DATABASE

Câu lệnh GRANT sử dụng trong trường hợp này có cú pháp như sau:

```
GRANT danh_sách_câu_lệnh
```

```
TO danh_sách_người_dùng
```

Ví dụ 3: Để cấp phát quyền tạo bảng và khung nhìn cho người dùng có tên là *thuchanh*, ta sử dụng câu lệnh như sau:

```
GRANT CREATE TABLE, CREATE VIEW TO thuchanh
```

Khác với trường hợp sử dụng câu lệnh GRANT để cấp phát quyền trên đối tượng cơ sở dữ liệu, câu lệnh GRANT trong trường hợp này không thể sử dụng tùy chọn WITH GRANT OPTION, tức là người dùng không thể chuyển tiếp được các quyền thực thi các câu lệnh đã được cấp phát.

9.2.1.3. Thu hồi quyền

a. Thu hồi quyền trên đối tượng cơ sở dữ liệu:

Cú pháp như sau:

```
REVOKE [GRANT OPTION FOR]
```

```
[PRIVILEGES] | các_quyền_cần_thu_hồi
```

```
[(danh_sách_cột)] ON tên_bảng | tên_khung_nhìn
```

```
| ON tên_bảng | tên_khung_nhìn [(danh_sách_cột)]
```

```
|ON tên_thủ_tục  
|ON tên_hàm  
FROM danh_sách_người_dùng  
[CASCADE]
```

Ví dụ 4: Thu hồi quyền thực thi lệnh INSERT trên bảng LOP đối với người dùng *thuchanh*.

```
REVOKE INSERT ON lop  
FROM thuchanh
```

Giả sử người dùng *thuchanh* đã được cấp phát quyền xem dữ liệu trên các cột HOTEN và NGAYSINH của bảng SINHVIEN, câu lệnh dưới đây sẽ thu hồi quyền đã cấp phát trên cột NGAYSINH (chỉ cho phép xem dữ liệu trên cột HOTEN)

```
REVOKE SELECT ON sinhvien (ngaysinh) FROM thuchanh
```

Nếu hai người dùng khác nhau cấp phát cùng các quyền trên cùng một đối tượng cơ sở dữ liệu cho một người dùng khác, sau đó người thu nhất thu hồi lại quyền đã cấp phát thì những quyền mà người dùng thứ hai cấp phát vẫn có hiệu lực.

Lưu ý: Nếu ta đã cấp phát quyền cho người dùng nào đó bằng câu lệnh GRANT với tùy chọn WITH GRANT OPTION thì khi thu hồi quyền bằng câu lệnh REVOKE phải chỉ định tùy chọn CASCADE. Trong trường hợp này, các quyền được chuyển tiếp cho những người dùng khác cũng đồng thời được thu hồi.

Ví dụ 5: Ta cấp phát cho người dùng A trên bảng R với câu lệnh GRANT như sau:

```
GRANT SELECT ON R TO A  
WITH GRANT OPTION
```

sau đó người dùng A lại cấp phát cho người dùng B quyền xem dữ liệu trên R với câu lệnh:

```
GRANT SELECT ON R TO B
```

Nếu muốn thu hồi quyền đã cấp phát cho người dùng A, ta sử dụng câu lệnh REVOKE như sau:

```
REVOKE SELECT ON R FROM A CASCADE
```

Câu lệnh trên sẽ đồng thời thu hồi quyền mà A đã cấp cho B và như vậy cả A và B đều không thể xem được dữ liệu trên bảng R.

Trong trường hợp cần thu hồi các quyền đã được chuyển tiếp và khả năng chuyển tiếp các quyền đối với những người đã được cấp phát quyền với tùy chọn WITH GRANT OPTION, trong câu lệnh REVOKE ta chỉ định mệnh đề GRANT OPTION FOR.

Ví dụ 6: Trong ví dụ trên, nếu ta thay câu lệnh:

```
REVOKE SELECT ON R FROM A CASCADE
```

bởi câu lệnh:

```
REVOKE GRANT OPTION FOR SELECT ON R  
FROM A CASCADE
```

Thì B sẽ không còn quyền xem dữ liệu trên bảng R đồng thời A không thể chuyển tiếp quyền mà ta đã cấp phát cho những người dùng khác (tuy nhiên A vẫn còn quyền xem dữ liệu trên bảng R).

b. Thu hồi quyền thực thi các câu lệnh:

Việc thu hồi quyền thực thi các câu lệnh trên cơ sở dữ liệu (CREATE DATABASE, CREATE TABLE, CREATE VIEW,...) được thực hiện đơn giản với câu lệnh REVOKE có cú pháp:

```
REVOKE các_câu_lệnh_cần_thu_hồi
FROM danh_sách_người_dùng
```

Ví dụ 7: Để không cho phép người dùng *thuchanh* thực hiện lệnh CREATE TABLE trên cơ sở dữ liệu, ta sử dụng câu lệnh:

REVOKE CREATE TABLE FROM thuchanh

9.2.2. Chỉ mục

- Cú pháp của lệnh CREATE INDEX là:

```
CREATE [UNIQUE] [CLUSTERED|NONCLUSTERED] INDEX index_name
ON table_name (column_name [, column_name]...)
[WITH
    [PAD_INDEX]
    [[,]FILLFACTOR=x]
    [[,]DROP_EXISTING]
]
```

Trong đó:

- ✓ UNIQUE chỉ ra rằng không bao giờ hai hàng có chung một giá trị chỉ mục.
- ✓ [CLUSTERED][NONCLUSTERED] là các kiểu chỉ mục khác nhau.
- ✓ PAD_INDEX chỉ ra khoảng trống còn lại mở ra trong mỗi trang ở mức trung của chỉ mục.

Tuỳ chọn PAD_INDEX chỉ hữu ích khi FILLFACTOR (hệ số điền) được xác định, bởi vì PAD_INDEX sử dụng tỷ lệ được xác định bởi hệ số điền.

- ✓ FILLFACTOR là một giá trị từ 0 đến 100 mà xác định phần trăm để trống của trang chỉ mục.
- ✓ DROP_EXISTING xóa một chỉ mục nào đó có cùng tên trong hệ thống.

- Nguyên tắc tạo chỉ mục

- Cột được sử dụng cho việc tìm kiếm một cách thường xuyên.
- Cột được sử dụng cho việc sắp xếp dữ liệu.

Không đánh chỉ mục các cột trong các trường hợp sau:

- Cột chỉ chứa một vài các giá trị duy nhất.
- Bảng chỉ chứa ít bản ghi (hàng). Việc đánh chỉ mục trên các bảng nhỏ có thể không là tối ưu nó khiến cho SQL Server mất nhiều thời gian để duyệt chỉ mục để tìm kiếm dữ liệu hơn là thực hiện việc tìm kiếm tuần tự từng hàng trong bảng.

Hệ số điền (Fill Factor)

Khi một index được tạo ra hoặc tổ chức lại, giá trị fill-factor sẽ xác định phần trăm dung lượng ở mỗi nút lá sẽ chứa dữ liệu, phần còn lại là không gian trống. Ví dụ, fill-factor là 80 nghĩa là 80% không gian trong mỗi trang dữ liệu sẽ được lấp đầy và 20% còn lại sẽ được để trống để chờ các bản ghi mới.

Khi có bản ghi mới được thêm vào hoặc cập nhật, hệ thống sẽ cố gắng sử dụng không gian trống này trước khi phải chia tách trang, giúp giảm phân mảnh.

9.2.2.1. Chỉ mục thông thường

- Chỉ mục Clustered

Một chỉ mục clustered xác định thứ tự lưu trữ của dữ liệu trong một bảng. Một bảng có thể có chỉ một chỉ mục clustered bởi vì chỉ mục clustered xác định thứ tự lưu trữ vật lý của dữ liệu.

Vd1: Để tạo một chỉ mục clustered trên cột **masv**, câu lệnh sẽ là:

```
CREATE CLUSTERED  
INDEX CLINDX_Masv ON Sinhvien2 (Masv)
```

Vd2: Tạo chỉ mục dạng clustered trên cột **Hoten** của bảng có tên **sinhvien2**

```
CREATE CLUSTERED  
INDEX INDX_ht ON Sinhvien2 (hoten)
```

- Chỉ mục Non-Clustered

Một chỉ mục non-clustered xác định cách sắp xếp logic của bảng. Vì vậy, một bảng có thể có rất nhiều chỉ mục non-clustered (có thể > 249). Một chỉ mục non-clustered giống như một chỉ mục trong một cuốn sách giáo khoa. Dữ liệu được lưu trữ trong một nơi, chỉ mục ở một nơi khác cùng với các con trỏ tới vị trí lưu trữ của dữ liệu.

Vd3: Tạo một chỉ mục non-clustered trên cột **Tenmonhoc** trong bảng **Monhoc** của cơ sở dữ liệu **QLSV**, lệnh sẽ là:

```
CREATE NONCLUSTERED INDEX NCLINDX_Tenmonhoc ON Monhoc  
(Tenmonhoc)
```

- Xem các chỉ mục

Sau khi tạo các chỉ mục, chúng ta có thể cần xem thông tin về các chỉ mục. Ví dụ, chúng ta có thể muốn xem danh sách các chỉ mục được tạo ra trên bảng chỉ định hoặc các cột được đánh chỉ mục trong một bảng.

Câu lệnh T-SQL sử dụng để xem các chỉ mục của một bảng là:

```
sp_helpindex <Table_name>
```

Trong đó, **sp_helpindex** là một thủ tục lưu hệ thống. Nó báo cáo các thông tin về các chỉ mục trong một bảng.

- Ứng dụng chỉ mục để tìm kiếm dữ liệu

Cú pháp:

```
[(INDEX=index_name)] hoặc sử dụng câu truy vấn thông thường. Điều kiện của  
2 câu lệnh này đều có điều kiện liên quan đến cột được chọn trong chỉ mục.
```

Trong đó, **index_name** là tên của một chỉ mục được tạo ra trên bảng.

Bây giờ để tìm kiếm trên cột **Tenmonhoc** trong bảng **Monhoc**, SQL Server thông qua chỉ mục **NCLINDEX_Tenmonhoc** sẽ tìm kết quả chính xác trong bảng tương ứng với điều kiện giá trị trên cột **Tenmonhoc**.

```
SELECT MaMonhoc, Tenmonhoc, Sotc
FROM Monhoc WITH (INDEX = NCLINDEX_Tenmonhoc)
WHERE Tenmonhoc= N'Cơ sở dữ liệu'
```

- Xóa các chỉ mục

Các chỉ mục đã tồn tại có thể được xóa khỏi cơ sở dữ liệu để giải phóng tài nguyên bằng câu lệnh DROP INDEX. Cú pháp là:

DROP INDEX table_name.index_name

Ví dụ, câu lệnh để xóa chỉ mục **NCLINDEX_Tenmonhoc** được tạo trên bảng **Monhoc** là:

```
DROP INDEX Monhoc. NCLINDEX_Tenmonhoc
```

Lệnh DROP INDEX không áp dụng cho các chỉ mục được tạo bởi các ràng buộc PRIMARY KEY hoặc UNIQUE. Nó cũng không thể được sử dụng để xóa một chỉ mục trên một bảng hệ thống.

9.2.2.2. Tìm kiếm toàn văn – Fulltext Search

- Sử dụng chỉ mục toàn văn

Các truy vấn này sử dụng hai từ khóa CONTAINS và FREETEXT.

✓ **Từ khóa CONTAINS**

Từ khóa CONTAINS tìm kiếm các cột chứa các kiểu dữ liệu ký tự phù hợp với các từ đơn và các cụm từ, các từ giống một phần một từ khác, và những sự phù hợp chính xác. Các câu truy vấn trong trường hợp này có thể nh.n giống như những truy vấn được viết với từ khóa LIKE.

Tuy nhiên, CONTAINS có thể cung cấp khả năng truy vấn tốt hơn nhiều so với từ khóa LIKE. Cũng như vậy, không giống từ khóa LIKE, CONTAINS là một từ khóa tìm kiếm phân biệt **hoa thường**.

✓ **Từ khóa FREETEXT**

Từ khóa FREETEXT tìm kiếm các cột có chứa giá trị đồng nghĩa hoặc gần nghĩa với các từ được cung cấp trong điều kiện tìm kiếm. Khi sử dụng FREETEXT, chuỗi tìm kiếm được chia thành một số các nhãn tìm kiếm và sau đó tiến hành tìm. Chuỗi tìm kiếm có thể là một tập hợp các từ hoặc các cụm từ, hoặc thậm chí một câu đầy đủ.

9.3. Hướng dẫn thực hành

9.3.1. Tạo một cơ sở dữ liệu.

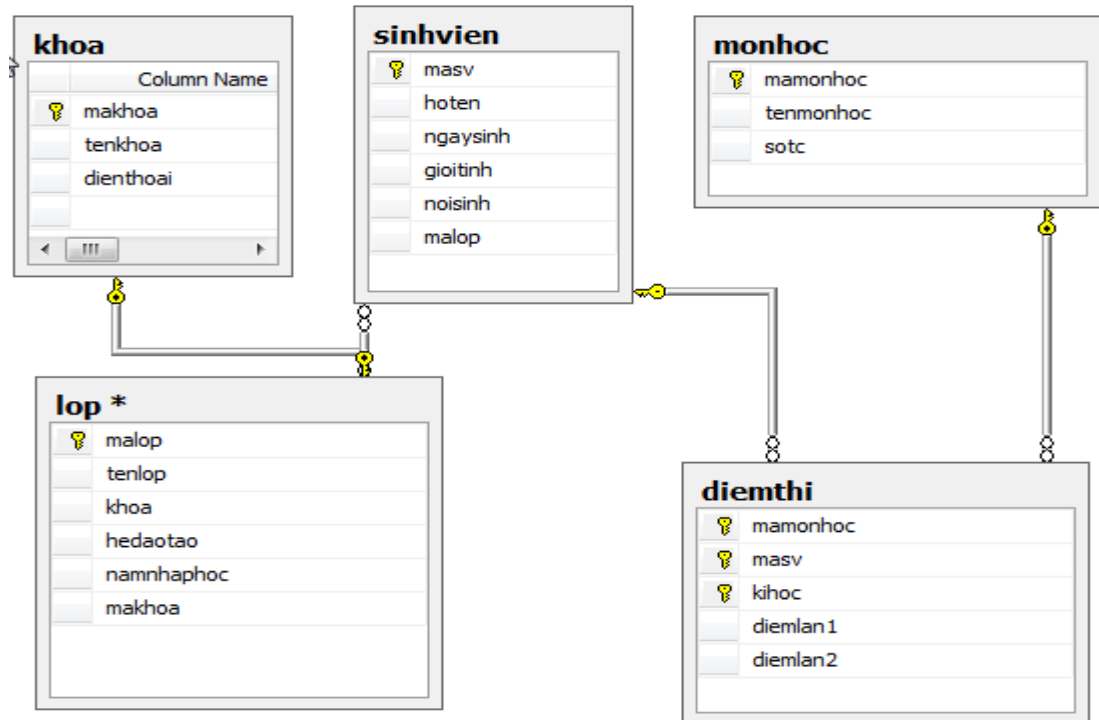
Đề bài: Tạo một CSDL QLSV đơn giản để quản lý điểm của sinh viên như sau (bảng công cụ và bằng ngôn ngữ SQL):

Phân tích bài toán:

Để tạo một CSDL trong SQL Server 2019 chúng ta có thể dùng hai cách sau:

- Dùng công cụ SQL Server Object Explorer

Hướng dẫn thực hiện:



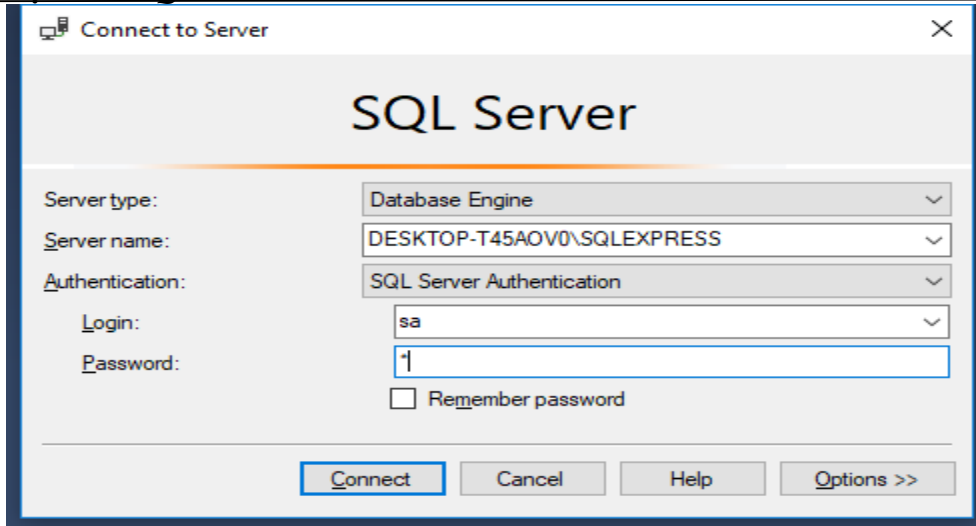
- Thiết lập các ràng buộc khóa chính, khóa ngoại như trên hình
- Tạo ràng buộc để cột diemLan1, diemlan2 trong bảng DiemThi nhận giá trị trong đoạn [0,10]
- Tạo ràng buộc trên cột sotc trong bảng monhoc nhận giá trị trong khoảng [1,8]
- Tạo ràng buộc trên cột ngaysinh sao cho năm sinh của sinh viên phải nằm trong khoảng [1990, 1995]
- Chèn dữ liệu vào mỗi bảng ít nhất 5 dòng dữ liệu.
- Cột gioitinh trong bảng SinhVien kiểu BIT.

9.3.2. Quản lý người dùng

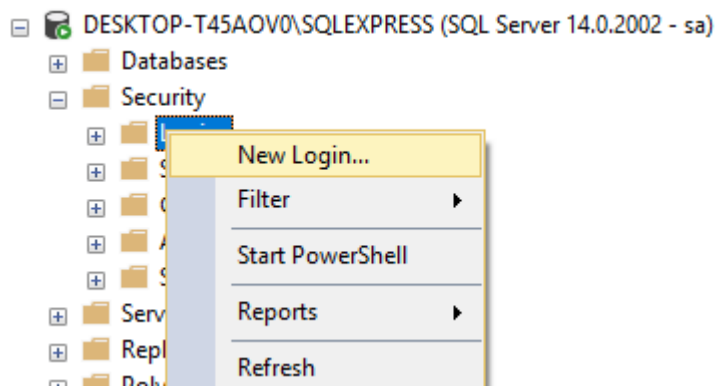
Tạo ra các tài khoản đăng nhập (bằng cách sử dụng câu lệnh SQL và Enterprise Manager): 'sv1', 'sv2'

Sử dụng Enterprise Manager tạo tài khoản và các user

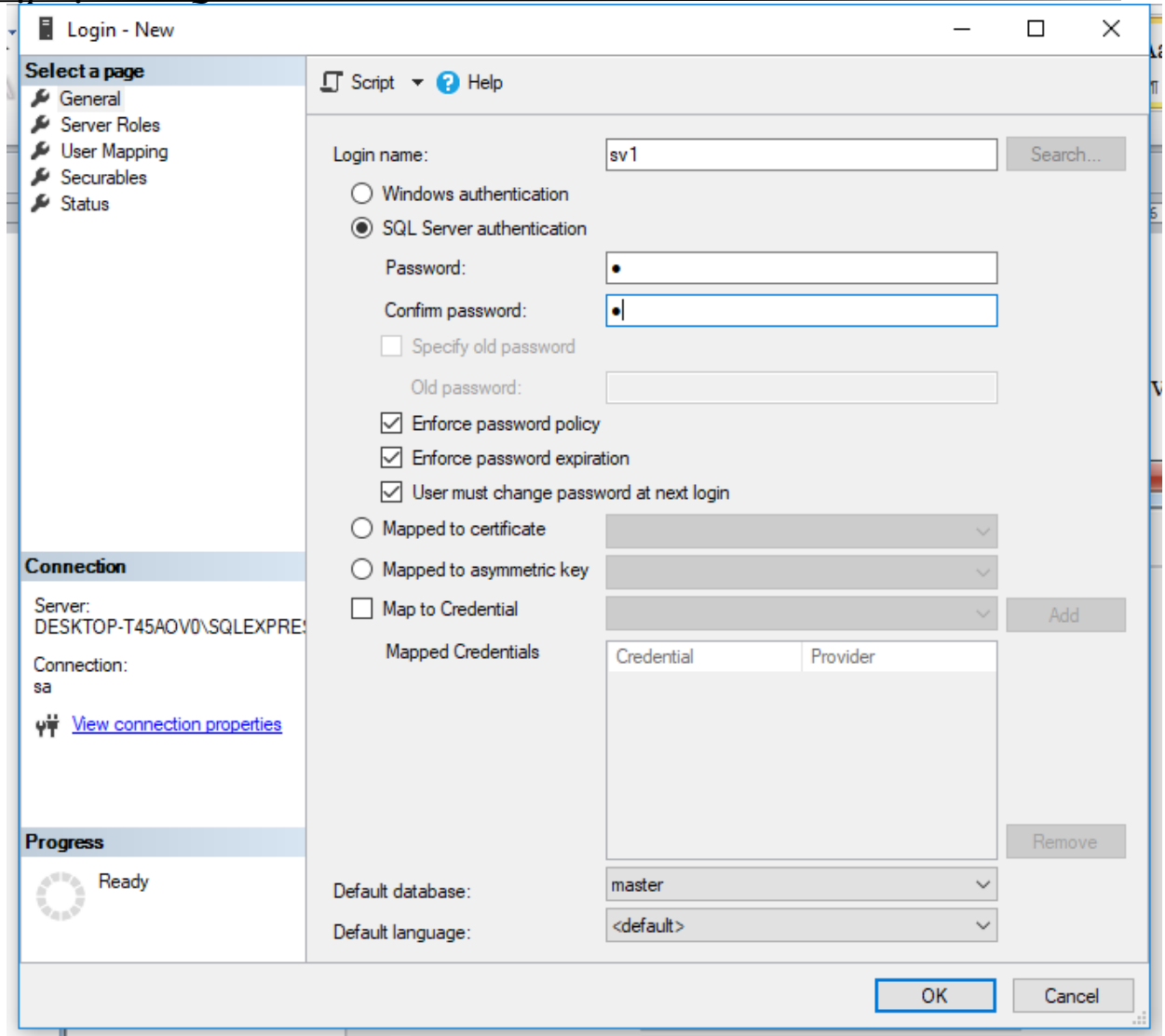
- Bước 1: Truy nhập vào SQL Server sử dụng tài khoản sa.



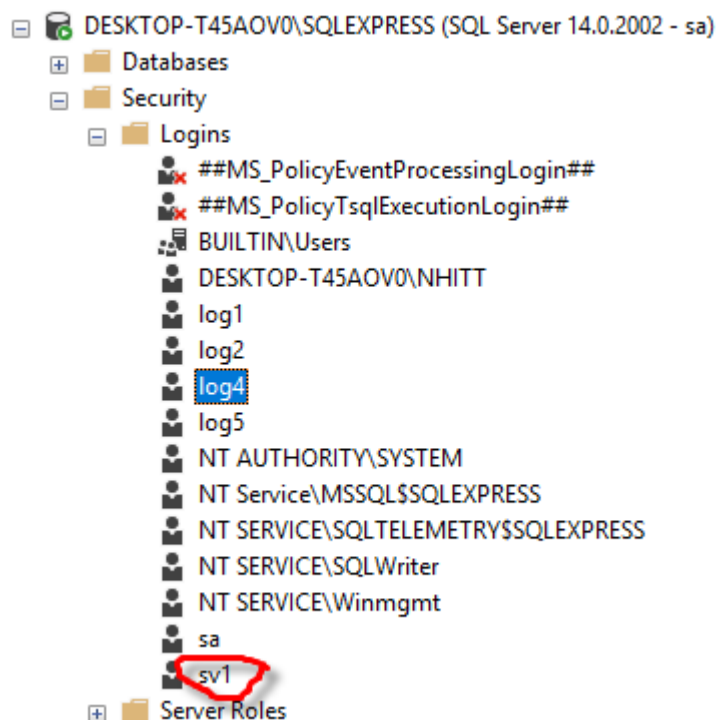
- Bước 2: Tạo ra 1 login đăng nhập có tên là: sv1



- Bước 3: Đặt tên cho Login, chọn quyền xác thực SQL Server authentication và đặt password. Chọn CSDL cần truy xuất trong mục: Default Database

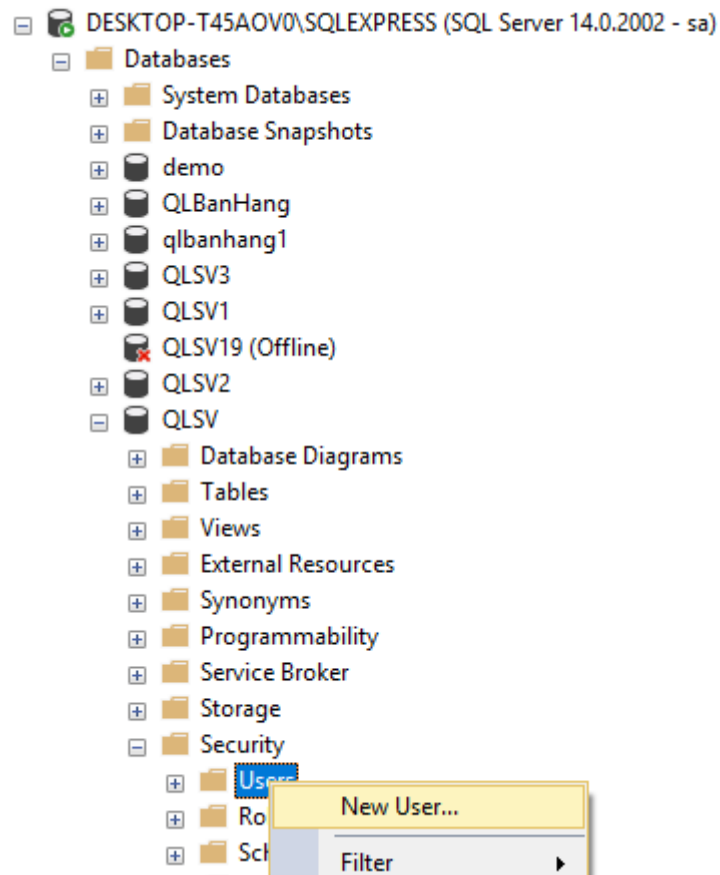


Nhấn OK là hoàn thành việc tạo một login tên là sv1.

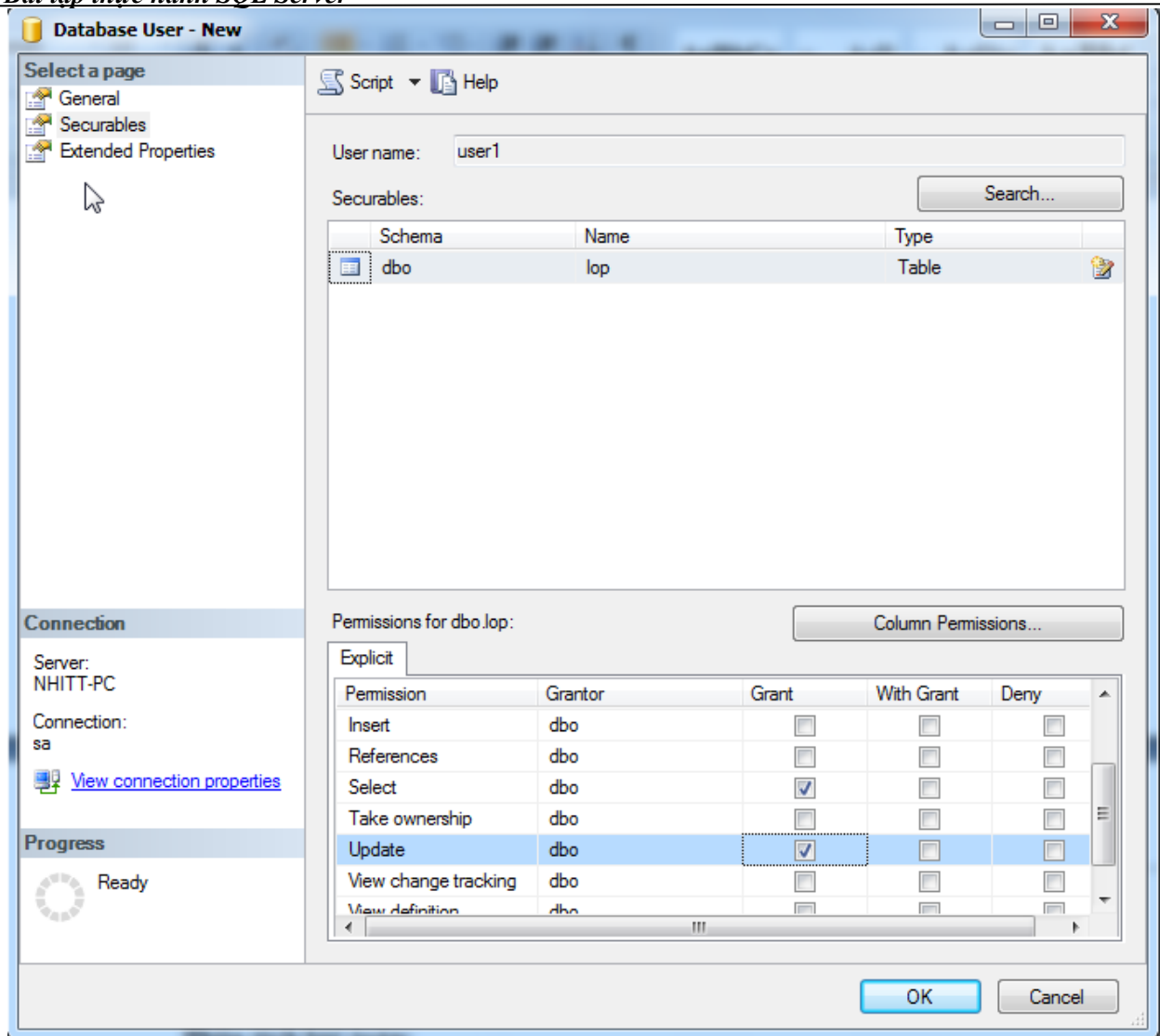


Bước 3: Tạo ra 1 user có tên là user1 đăng nhập từ login sv1trên và truy xuất được vào CSDL QLdiem.

Chọn CSDL QLSV, Vào Security. Nháy chuột phải chọn New User...



- Bước 4: Nhập tên user Name, Login đăng nhập và chọn quyền



Sử dụng ngôn ngữ T- SQL tạo tài khoản và các user

USE QLSV

-- Tạo login SV1 có password là 1 như sau:

```
EXEC SP_ADDLOGIN SV1,'1'
```

-- Tạo user USER1 từ login trên

```
EXEC SP_ADDUSER SV1, USER1
```

-- Cấp quyền cho user1 trên bảng LOP

```
GRANT SELECT, UPDATE ON LOP
```

```
TO USER1
```

-- Thu hồi quyền thực thi lệnh SELECT trên bảng LOP đối với người dùng USER1

```
REVOKE SELECT ON LOP
```

```
FROM USER1
```

Thường xuyên sao lưu cơ sở dữ liệu của bạn để đảm bảo an toàn dữ liệu và có kế hoạch khôi phục khi cần thiết.

-- Sao lưu cơ sở dữ liệu

```
BACKUP DATABASE QLSV TO DISK = 'C:\ backup\QLSV.bak'
```

```
GO
```

```
-- Khôi phục cơ sở dữ liệu
```

```
RESTORE DATABASE QLSV FROM DISK = 'C:\ backup\QLSV.bak'
```

```
GO
```

9.3.3. Chỉ mục

✓ Chỉ mục thông thường:

Ta cũng có thể tạo bằng 2 cách: Công cụ và ngôn ngữ.

Tạo chỉ mục bằng câu lệnh

Tạo chỉ mục Nonclustered trên cột Hoten

```
CREATE NONCLUSTERED INDEX NCLINDX_Sinhvien_Tensv ON SinhVien  
(Hoten)
```

Ứng dụng chỉ mục để tìm kiếm dữ liệu

Tìm kiếm tất cả Sinh viên có tên là “Mai”

```
SELECT MaSV, Hoten, Noisinh  
FROM SinhVien WITH (INDEX = NCLINDX_Sinhvien_HT)  
WHERE RIGHT(Hoten,3)=N'Mai'
```

Hoặc sử dụng:

```
SELECT MaSV, Hoten, Noisinh  
FROM SinhVien  
WHERE RIGHT(Hoten,3)=N'Mai'
```

✓ Chỉ mục toàn văn (FULLTEXT)

Các chỉ mục toàn văn có thể được tạo ra bằng công cụ hoặc tạo bằng câu lệnh. Sau đây là hướng dẫn cách tạo bằng câu lệnh.

```
EXEC SP_FULLTEXT_DATABASE 'ENABLE'
```

Tạo chỉ mục toàn văn

```
--B1: Tạo ra danh mục các chỉ mục
```

```
CREATE FULLTEXT CATALOG HT_cat
```

```
with accent_sensitivity = off
```

```
as default
```

```
--B2: Tạo ra chỉ mục trong danh mục các fulltext (phải dữ trên ràng buộc khóa chính)
```

CREATE FULLTEXT INDEX ON sinhvien (Hoten)

KEY INDEX PK_ sinhvien on HT_cat

Lưu ý:

Chỉ mục toàn văn yêu cầu phải dựa trên khóa chính của bảng. Trong hướng dẫn tạo chỉ mục toàn văn trên **PK_ sinhvien** là tên ràng buộc khóa chính của bảng sinhvien.

Ứng dụng chỉ mục toàn văn

Ví dụ: Hiển thị thông tin của các sinh viên có họ tên chứa các từ là *Anh, Mai, Hoa*:

```
SELECT MaSV, Hoten, Noisinh
FROM sinhvien
WHERE CONTAINS (hoten, '"Anh" or "Mai" or "Hoa"')
--hàm freetext
SELECT MaSV, Hoten, Noisinh
FROM sinhvien
WHERE FREETEXT(hoten, '"MAI", "Anh", "Hoa"')
```

Lưu ý sự khác nhau giữa TRUNCATE và DELETE

Lệnh **TRUNCATE** và **DELETE** đều được dùng để xóa dữ liệu trong bảng. Cả hai lệnh này chỉ xóa dữ liệu mà không xóa cấu trúc bảng.

Lệnh TRUNCATE

Lệnh **TRUNCATE** xóa tất cả dòng dữ liệu trong bảng mà **KHÔNG ghi nhận lại trong transaction log**. Lệnh **TRUNCATE** có cùng chức năng như lệnh **DELETE** là xóa dữ liệu trong bảng mà không làm thay đổi cấu trúc của bảng, tuy nhiên, bạn không dùng mệnh đề **WHERE** với lệnh **TRUNCATE**.

Cú pháp:

TRUNCATE TABLE [{database_name.[schema_name].}schema_name.]table_name

Trong đó: **table_name**: là tên của bảng cần xóa tất cả các dòng dữ liệu.

VD: **TRUNCATE TABLE** Products;

Lệnh trên sẽ xóa tất cả dữ liệu trong bảng Products.

Lệnh DELETE

Lệnh **DELETE** cũng xóa các dòng trong bảng, nhưng nó **ghi lại các dòng được xóa trong transaction log**. Bạn có thể dùng mệnh đề **WHERE** với lệnh **DELETE** để giới hạn các dòng được xóa.

Cú pháp:

DELETE FROM TABLE_NAME
[{database_name. [schema_name] . | schema_name. }] table_name

Trong đó:

- **database_name:** Tên của cơ sở dữ liệu chứa bảng dữ liệu cần xóa. Đây là tùy chọn, nếu không chỉ định, cơ sở dữ liệu hiện tại được chọn.
- **schema_name:** Tên của schema chứa bảng dữ liệu cần xóa. Đây là tùy chọn, nếu không chỉ định, schema hiện tại được chọn.

table_name: Tên của bảng dữ liệu cần xóa.

9.4. Bài tập tự làm

Bài 1: Tạo ra một CSDL có tên là QLSV với các ràng buộc trong mục 9.3 và thực hiện bài tập có hướng dẫn ở mục

Bài 2: Chèn dữ liệu vào cho tất cả các bảng đó (bằng hai cách) ít nhất 10 bản ghi/bảng. Trong đó, bảng để thử nghiệm tìm kiếm chỉ mục nhập dữ liệu với số lượng lớn.

Bài 3: Quản lý người dùng

1. Hãy đăng nhập vào SQL Server bằng tài khoản sa (tài khoản có đặc quyền lớn nhất)

2. Tạo ra các tài khoản đăng nhập (bằng cách sử dụng câu lệnh SQL và công cụ): ‘sv1’, ‘sv2’, ‘sv3’, ‘gv1’, ‘gv2’ với mật khẩu tùy ý, sau đó lần lượt đăng nhập thử bằng các tài khoản đã tạo trong CSDL QLSV.

3. Trong CSDL QLSV, tạo ra các tài khoản NSD có tên tương ứng với tài khoản sau:

Tên đăng nhập	Tên NSD trong CSDL
SV1	SinhVien1
SV2	SinhVien2
SV3	SinhVien3
Gv1	GiaoVien1
Gv2	GiaoVien2

4. Thiết lập quyền hạn cho GiaoVien1 có toàn quyền thao tác trên bảng SINHVIEN.

5. Thêm NSD ‘GiaoVien1’, ‘GiaoVien1’ vào nhóm ‘GiaoVien’ bằng cả 2 cách

6. Phân quyền cho nhóm ‘GiaoVien’ trong CSDL quyền tạo lập bảng mới.

7. Phân quyền cho SinhVien1 trong CSDL có quyền chèn, thêm dữ liệu, đọc dữ liệu nhưng không có quyền sửa và xóa dữ liệu trên bảng Khoa.

8. Phân quyền cho ‘SinhVien2’ trong CSDL được sửa và xóa dữ liệu trên Monhoc.

9. Phân quyền cho ‘SinhVien3’ trong CSDL được truy vấn dữ liệu trên Monhoc.

10. Đăng nhập bằng các tài khoản trên sau đó thử thực hiện một thao tác mà NSD đó không có quyền.

11. Đăng nhập với tài khoản sa để thu hồi các quyền của các người dùng mà sa đã cấp phát.

Bài 4: Hãy thực hiện các yêu cầu sau:

1. Tạo một bảng SVtam từ bảng sinhvien (thiết lập lại khóa chính là Masv)

2. Tạo chỉ mục Nonclustered trên cột Hoten
 3. Tạo chỉ mục Nonclustered phức hợp trên hai cột hoten và noisinh
 4. Xem lại các chỉ mục đang có trên bảng Sinhvien
 5. Tìm kiếm Thông tin các sinh viên có họ “Nguyễn” bằng cách sử dụng chỉ mục được tạo trong câu 2.
 6. Tìm kiếm tất cả các Sinh viên tên là “Lan” và nơi sinh ở “Hưng Yên” bằng cách sử dụng chỉ mục được tạo trong câu 3
 7. Xóa các chỉ mục được tạo trong câu 1-3
 8. Xóa ràng buộc khóa chính khỏi bảng **svtam**
 9. Tạo lại khoá chính trên bảng **svtam**, trong quá trình tạo khoá chính sử dụng tùy chọn NonClustered để SQL Server không tạo chỉ mục Clustered trên chỉ mục khoá chính (vì mặc định khi tạo khoá chính SQL sẽ tự động tạo chỉ mục Clustered trên cột khoá)
 10. Tạo chỉ mục Clustered và Unique trên cột Masv với hệ số điền đầy bằng 60
 11. Thiết lập chỉ mục tìm kiếm toàn văn trên cột Hoten.
 12. Tìm kiếm tất cả các cuốn sách có tên sách chứa từ ‘Văn’ hoặc ‘Nguyễn’
- Chú ý:** Tất cả các công việc trên được thực hiện bằng cả hai cách: Enterprise Manager (Công cụ) và Query Analyzer (Câu lệnh)

10.1. Mục tiêu

Sau bài thực hành này, người học có khả năng:

- ✓ Trình bày được quá trình chạy (batch - xử lý bó) trong T-SQL
- ✓ Sử dụng được biến trong T-SQL.
- ✓ Sử dụng các hàm cơ bản trong lập trình T-SQL
- ✓ Vận dụng thành thạo các cấu trúc điều khiển trong các truy vấn và trong các đoạn chương trình.
- ✓ Rèn luyện tư duy trong việc lựa chọn cấu trúc điều khiển trong lập trình T-SQL.

10.2. Tóm tắt lý thuyết

Biến(variables)

- Biến trong T-SQL cũng có chức năng tương tự như trong các ngôn ngữ lập trình khác nghĩa là cần khai báo trước loại dữ liệu trước khi sử dụng. Biến được bắt đầu bằng dấu @ (Đối với các global variable thì có hai dấu @@)
- Cú pháp:
- Declare @tên_biến (kiểu_dữ_liệu)
- Gán giá trị cho biến
 - Cú pháp 1: SET @tên_biến=giá_trị
 - Cú pháp 2:


```
SELECT @tên_biến= <tên cột>
FROM <tên bảng>
[WHERE <BTĐK>]
```

Toán tử (Operator)

- *Toán tử gán*: Ký hiệu là dấu '=' được dùng để gán giá trị cho một biến hoặc một cột.

```
DECLARE @intValue int
SELECT @intValue = 1

PRINT @intValue
```

hoặc

```
DECLARE @intValue int

SET @intValue = 1

PRINT @intValue
```

- *Toán tử số học*: Đó là các phép toán cộng (+), trừ (-), nhân (*), chia (/) và chia modul (%).

- *Toán tử so sánh*: Đó là các phép toán so sánh giữa hai biểu thức và trả về giá TRUE hoặc FALSE. Đó là các phép so sánh: = (bằng), <> (khác), > (lớn hơn), >= (lớn hơn hoặc bằng), < (nhỏ hơn), <= (nhỏ hơn hoặc bằng).

- *Toán tử logic*: Kiểm tra điều kiện đúng của hai biểu thức, chúng thường được sử dụng cùng với các toán tử so sánh để trả về giá trị TRUE hoặc FALSE: ALL, ANY, AND, BETWEEN, EXISTS, IN, LIKE, NOT, OR, SOME.

- *Toán tử ghép chuỗi (+)*: Dùng để ghép hai chuỗi với nhau thành một chuỗi. Toán tử ghép chuỗi được dùng với các kiểu dữ liệu char, varchar, nchar, nvarchar, text, và ntext.

SELECT 'This' + ' is a test.'

- *Toán tử bit*: Thực hiện thao tác với các bit-level với các kiểu dữ liệu Integer: AND, OR, XOR, NOT.

Các cấu trúc điều khiển:

➤ **Cấu trúc rẽ nhánh: IF...ELSE**

Cú pháp:

If (điều_kiện_công_việc_1)

Câu lệnh/nhóm lệnh

Else (if (điều_kiện_công_việc_2))

.....

Else công_việc_n

➤ **Cấu trúc Case**

Khi chúng ta sử dụng nhiều If...else thì có thể dùng Case...When để thay thế.

Cú pháp:

CASE Biểu thức

WHEN Giá trị 1 THEN kết quả1

WHEN Giá trị 2 THEN kết quả2

WHEN Giá trị n THEN kết quả3

END

Hoặc

CASE

WHEN *Boolean_expression* THEN *result*

[...n]

ELSE *else_result*

END

➤ **Cấu trúc Waitfor**

- Thực thi một khối lệnh của một stored procedure hay một transaction đến thời gian chỉ định hoặc thời gian được kích hoạt
- Sửa một câu lệnh được chỉ định hoặc trả về ít nhất một dòng.

Cú Pháp:

WAITFOR

```
{
    DELAY <"thời gian">  //--khoảng thời gian mà SQL Server phải chờ
    | TIME <"thời gian">  //-- yêu cầu SQL Server chờ đến thời điểm chỉ
    định
}
```

Ví dụ :

WAITFOR Time '12:30:23'

Print "Câu lệnh này chỉ được chạy sau: 12:30:23"

➤ **Cấu trúc TRY ... CATCH**

- Cú pháp:

```
BEGIN TRY
    { cau_lenh_sql|khởi_cau_lenh }
END TRY
BEGIN CATCH
    { cau_lenh_sql|khởi_cau_lenh }
END CATCH
```

- Mỗi khối TRY phải có một khối CATCH đơn theo ngay sau.
- Khi lỗi xuất hiện từ câu lệnh trong khối TRY, bộ điều khiển được chuyển tới khối CATCH, nơi thực hiện xử lý lỗi. Nếu không có lỗi xảy ra trong khối TRY, khối CATCH được bỏ qua.

➤ **Lặp không xác định: WHILE**

Cú pháp:

WHILE Boolean_expression

```
{ sql_statement | statement_block }
```

```
[ BREAK ]
```

```
{ sql_statement | statement_block }
```

[CONTINUE]

[**BREAK**] : Dừng để kết thúc vòng lặp khi gặp một trường hợp cụ thể nào đó.

[**CONTINUE**] : Lặp lại đầu vòng lặp.

Thông thường 2 từ khóa Break và Continue phải nằm trong trong cấu trúc If ... Else..

10.3. Hướng dẫn thực hành

1. In ra các số nguyên lẻ từ 1 đến 10:

```
✓ Khai báo biến:
DECLARE @t1 INT
✓ Gán giá trị cho biến
SET @t1=1
✓ Thực hiện vòng lặp để in từng giá trị sau mỗi lần kiểm tra điều kiện
WHILE @t1<10
BEGIN
    PRINT @T1--print 'SO CAN IN '+convert(varchar(11),
@t1)--
    SET @t1=@t1+2
END
GO
```

2. Cho bảng Sinhvien trong CSDL QLSV

- Lập trình T-SQL thực hiện công việc sau:

- ✓ Đếm số sinh viên có nơi sinh ở Hưng Yên, nếu số sinh viên lớn hơn 10 thì hiển thị thông tin của sinh viên đó. Ngược lại, hiển thị toàn bộ thông tin của bảng Sinhvien

- Hướng dẫn:

- ✓ Khai báo biến: *DECLARE @so_luong int*

- ✓ Gán giá trị:

```
SELECT @so_luong= COUNT(MaSV)
FROM SINHVIEN
WHERE Noisinh = N'Hưng Yên'
```

- ✓ Thực hiện:

```
If (@so_luong >=10)
    SELECT MaSv, Hoten, Gioitinh
    FROM SINHVIEN
    WHERE Noisinh = N'Hưng Yên'
```

Else

```
SELECT MaSv, Hoten, Gioitinh FROM SINHVIEN
```

3. Cho bảng sau:

- Ban(mahoadon, mahang, trigiaban)
- Mua(mahoadon, mahang, trigiamua)
- Lập trình T-SQL thực hiện các công việc sau:
 - ✓ Giảm giá bán 20% đối với những mặt hàng mà có giá bán trên 50.000
 - ✓ Tăng giá mua 20% cho những mặt hàng, tăng cho đến khi trị giá mua lớn nhất lớn hơn 80.000
- Hướng dẫn:
 - If (exists (SELECT trigaiban from Ban where trigiaban> 50000))
UPDATE BAN SET trigiaban=trigiaban-trigiaban*0.2
 - Declare @lon_nhat int
SELECT @lon_nhat = MAX(trigiamua) from Mua
WHILE (@lon_nhat<80000)
UPDATE MUA SET trigiamua =trigiamua+trigiamua*0.2

4. Cho CSDL QLSV trong bài thực hành 9.

---Hiện thị thông tin của sinh viên: masv, hoten, gioitinh
sao cho nếu gioitinh=0 thì hiện thị là 'Nữ', ngược lại
hiện thị 'Nam'

- Hướng dẫn:

Ta sử dụng truy vấn thông thường, thuộc tính tính toán là gt, thiết lập các giá trị cho gioitinh bằng cách sử dụng cấu trúc Case.

```
SELECT masv, hoten , gioitinh=
CASE WHEN gioitinh = 0 THEN 'Nữ'
ELSE 'Nam'
END
FROM sinhvien
```

10.4. Bài tập tự làm

Bài 1. Viết đoạn chương trình thực hiện việc cộng n số nguyên đầu tiên (n tùy ý).

Bài 2. Viết đoạn chương trình dùng để tính giai thừa của một số nguyên n tùy chọn.

Bài 3: Viết đoạn chương trình in ra thời khóa biểu học tập trong một tuần của bạn.

Bài 4. Cho CSDL QLSV trong bài thực hành 9. Thực hiện các yêu cầu sau đây:

- a. Tính số lượng sinh viên trong bảng sinh viên. Nếu số lượng sinh viên >100 thì in ra chuỗi 'Số lượng sinh viên đã vượt 100!'. Ngược lại, in ra chuỗi 'Số lượng sinh viên hợp lệ'
- b. Hiện thị thông tin của các môn học gồm: Mamonhoc, tenmonhoc, tinhchat. Trong đó, giá trị của cột tinhchat dựa vào số tín chỉ của môn học đó:
 - Nếu sotec>=6 thì tính chất môn là 'Thực tập, đồ án'
 - Nếu sotec >=4 and sotec<6 thì tính chất môn là 'Cơ sở ngành, chuyên ngành'
 - Trường hợp còn lại tính chất môn là 'Cơ bản'
- c. Hiện thị thông tin của các sinh viên gồm: Masv, hoten, diemTB11, Xeploai của mỗi sinh viên thuộc kì 1. Biết rằng, diemTB11 của mỗi sinh viên được tính toán bằng

công thức $\sum(\text{Diemlan1} * \text{Sotc}) / \sum(\text{Sotc})$; giá trị của cột Xeploai dựa vào cột diemTB11:

Nếu $\text{DiemTB11} \geq 9$ thì Xeploai là 'Xuất sắc'

Nếu $8 \leq \text{DiemTB11} < 9$ thì Xeploai là 'Giỏi'

Nếu $7 \leq \text{DiemTB11} < 8$ thì Xeploai là 'Khá'

Nếu $6 \leq \text{DiemTB11} < 7$ thì Xeploai là 'Trung bình'

Còn lại Xeploai là 'Yếu'

d. Cho bảng dữ liệu của bảng môn học như sau:

	mamonhoc	tenmonhoc	sotc
►	CSDL	Cơ sở dữ liệu	4
	Csktlt	Cơ sở KTLT	4
	dhnn	Định hướng NN	2
	tcc1	Toán cc1	2
	tcc2	Toán cc2	2
	tdc	Tin đại cương	2
	vl	Vật lý	2

Hãy thực hiện việc in ra các thông tin về lỗi khi thực hiện các câu lệnh sau và chỉ ra lỗi ở dòng lệnh nào:

```
DECLARE @st NVARCHAR(5)
SELECT @st = SOTC FROM monhoc
PRINT CONVERT(DATETIME, @ST)
```

e. Tạo cấu trúc bảng:

```
CREATE TABLE T1(MA INT PRIMARY KEY, TEN NVARCHAR(30))
```

Dùng cấu trúc lặp WHILE thực hiện nhập vào bảng T1, 18 bản ghi có nội dung như sau:

	MA	TEN
1	1	Bản ghi thứ 1
2	2	Bản ghi thứ 2
3	3	Bản ghi thứ 3
4	4	Bản ghi thứ 4
5	5	Bản ghi thứ 5
6	6	Bản ghi thứ 6
7	7	Bản ghi thứ 7
8	8	Bản ghi thứ 8
9	9	Bản ghi thứ 9
10	10	Bản ghi thứ 10
11	11	Bản ghi thứ 11
12	12	Bản ghi thứ 12
13	13	Bản ghi thứ 13
14	14	Bản ghi thứ 14
15	15	Bản ghi thứ 15
16	16	Bản ghi thứ 16
17	17	Bản ghi thứ 17
18	18	Bản ghi thứ 18
19	19	Bản ghi thứ 19
20	20	Bản ghi thứ 20

- f. Thêm 1 cột Diadiem vào bảng KHOA Cập nhật dữ liệu trên cột địa điểm của bảng khoa dựa vào mã khoa như sau:
- Những khoa có mã là 'CNTT', 'KT', 'MTK', 'NN' thì địa điểm là 'CS2'
 - Những khoa có mã là 'CK', 'ĐT', 'ĐL', 'HMT' thì địa điểm là 'CS1'
 - Còn lại các khoa có mã khác thì địa điểm là 'CS3'

Bài tập về nhà

Tạo CSDL có cấu trúc như sau sử dụng ngôn ngữ T-SQL

- Bảng Phòng: lưu thông tin về phòng được quản lý theo thành phố

Tên trường	Kiểu dữ liệu	Kích thước	Chú thích
<u>Maphong</u>	Char	5	Mã của phòng(khóa chính)
Tenphong	Nvarchar	50	Tên phòng
Matp	int		Mã thành phố

- Bảng NhânViên: lưu thông tin về nhân viên

Tên trường	Kiểu dữ liệu	Kích thước	Chú thích
<u>MaNV</u>	Char	5	Mã nhân viên (khóa chính)
Hoten	Nvarchar	50	Họ tên của nhân viên
Gioitinh	Char	5	Giới tính
Maphong	Char	5	Mã đơn vị (khóa ngoại)
Luong	Int		Lương của nhân viên
Ngaysinh	datetime		Ngày sinh
Quequan	Nvarchar	50	Quê quán nhân viên

- Bảng Địa điểm phòng: lưu thông tin vị trí của phòng

Tên trường	Kiểu dữ liệu	Kích thước	Chú thích
Maphong	Char	5	Mã phòng(khóa ngoại)
Diadiem	Varchar	50	Địa điểm của phòng

- Bảng Duan: lưu thông tin về các dự án do phòng quản lý

Tên trường	Kiểu dữ liệu	Kích thước	Chú thích
MaDA	Char	5	Mã dự án(khóa chính)
TenDA	Varchar	5	Tên của dự án
DiadiemDA	Varchar	50	Địa điểm của dự án
Maphong	Char	5	Mã phòng(khóa ngoại)

- Bảng Chấm công: lưu thông tin về thời gian làm việc của nhân viên

Tên trường	Kiểu dữ liệu	Kích thước	Chú thích
MaNV	Char	5	Mã nhân viên(khóa ngoại)
MaDA	Char	5	Mã dự án(khóa ngoại)
Sogio	Int		Số giờ thực hiện

Thực hiện các truy vấn sau trên CSDL Quản lí dự án:

1. Đưa ra danh sách những nhân viên có quê ở Hà Nội
2. Hiển thị thông tin của những nhân viên có tên “An”
3. Hiển thị thông tin của những nhân viên có lương nhỏ hơn 5000000
4. Hiển thị thông tin của những nhân viên có lương cao nhất
5. Tìm tên của nhân viên có lương nhỏ hơn nhân viên có mã “NV1”
6. Đưa ra danh sách nhân viên chưa tham gia dự án nào
7. Tăng lương cho nhân viên. Nếu nhân viên đó có số giờ tham gia 1 dự án <10 giờ thì tăng lương của nhân viên đó lên 10%, nếu số giờ tham gia lớn hơn 10 và nhỏ hơn 50 thì tăng 5% lương
8. Hiện thị thông tin về các dự án, sắp xếp tên các dự án giảm theo số giờ tham gia.
9. Đưa ra tên phòng có nhân viên tham gia dự án trong tháng 6 năm 2020
10. Tăng lương 10% cho nhân viên nhiều tuổi nhất trong công ty nhân dịp Tết dương lịch
11. Tính số nhân viên tham gia mỗi dự án trong năm 2020.
12. Hiện thị thông tin về các phòng có số nhân viên nhiều nhất.

BÀI 11. TRUY VẤN NÂNG CAO; VIEW, CON TRỎ, TRANSACTION

11.1. Mục tiêu

Sau bài thực hành này, người học có khả năng:

- ✓ Thành thạo truy vấn với các phép toán nối, truy vấn con.
- ✓ Thành thạo các mệnh đề Group by, Having, Order by...
- ✓ Sử dụng được các toán tử nâng cao khác: TOP, phân hạng,...
- ✓ Thành thạo các hàm cơ bản trong SQL Server.
- ✓ Rèn luyện tư duy trong việc lựa chọn các toán tử, mệnh đề phù hợp trong truy vấn.
- ✓ Thiết kế được các VIEW và thao tác trên VIEW đó.
- ✓ Lập trình sử dụng con trỏ xử lý từng bản ghi của bảng.
- ✓ Lập trình giao dịch thực hiện các yêu cầu cụ thể của từng ứng dụng.
- ✓ Rèn luyện tính cẩn thận trong việc viết SCRIPS.

11.2. Tóm tắt lý thuyết

11.2.1. Mệnh đề TOP

Mệnh đề TOP chỉ định tập hợp các dòng đầu tiên được trả về trong truy vấn. Tập hợp các dòng đó có thể là một con số hoặc theo tỷ lệ phần trăm (PERCENT) các dòng dữ liệu. Mệnh đề TOP được sử dụng trong các khối câu lệnh SELECT, INSERT, UPDATE và DELETE. Cú pháp:

```
[ TOP (expression) [PERCENT]
  [ WITH TIES ]
]
```

Trong đó:

- Expression: Là biểu thức trả về giá trị kiểu số.
- PERCENT: Chỉ định số dòng trả về là expression phần trăm trong tập kết quả.
- WITH TIES: TOP ...WITH TIES chỉ được chỉ định trên khối câu lệnh SELECT và có mệnh đề ORDER BY. Chỉ định thêm các dòng từ tập kết quả cơ sở có cùng giá trị với các cột trong mệnh đề ORDER BY xuất hiện như là dòng cuối cùng của TOP n (PERCENT).

Ví dụ: Sử dụng mệnh đề TOP

- Trong câu lệnh Insert

INSERT TOP (2) INTO LOP

SELECT * FROM DMLOP ORDER BY Khoa

- Trong câu lệnh SELECT

INSERT INTO LOP

SELECT TOP (2) WITH TIES * FROM DMLOP

ORDER BY Khoa

11.2.2. Các phép nối

Trong khối câu lệnh SELECT, ở mệnh đề FROM ta có thể sử dụng phát biểu JOIN để kết nối các bảng có quan hệ với nhau.

Mệnh đề kết nối Join được phân loại như sau:

- Inner joins (toán tử thường dùng để kết nối thường là các toán tử so sánh = hoặc <>). Inner joins sử dụng một toán tử so sánh để so khớp các dòng từ hai bảng dựa trên các giá trị của các cột so khớp của mỗi bảng. Kết quả trả về của Inner Join là các dòng thỏa mãn điều kiện so khớp.
- Outer joins. Outer joins có thể là left, right, hoặc full outer join.
 - ✓ LEFT JOIN hoặc LEFT OUTER JOIN: Kết quả của left outer join không chỉ bao gồm các dòng thỏa mãn điều kiện so khớp giữa hai bảng mà còn gồm tất cả các dòng của bảng bên trái trong mệnh đề LEFT OUTER. Khi một dòng ở bảng bên trái không có dòng nào của bảng bên phải so khớp đúng thì các giá trị NULL được trả về cho tất cả các cột ở bảng bên phải.
 - ✓ RIGHT JOIN or RIGHT OUTER JOIN: Right outer join là nghịch đảo của left outer join. Tất cả các dòng của bảng bên phải được trả về. Các giá trị Null cho bảng bên trái khi bất cứ một dòng nào bên phải không có một dòng nào bảng bên trái so khớp đúng.
 - ✓ FULL JOIN or FULL OUTER JOIN: full outer join trả về tất cả các dòng trong cả hai bảng bên trái và phải. Bất kỳ một dòng không có dòng so khớp đúng của bảng còn lại thì bảng còn lại nhận các giá trị NULL. Khi có sự so khớp đúng giữa các bảng thì tập kết quả sẽ chứa dữ liệu các bảng cơ sở đó.
- Cross joins: Trả về tất cả các dòng của bảng bên trái và mỗi dòng bên trái sẽ kết hợp với tất cả các dòng của bảng bên phải. Cross joins còn được gọi là tích Đề các (Cartesian products).

11.2.3. Toán tử PIVOT và UNPIVOT

SQL Server đưa ra các toán tử đơn giản hơn cho việc truy vấn cross tab, đó là toán tử PIVOT và UNPIVOT trong mệnh đề FROM của khối câu lệnh SELECT.

- + Toán tử PIVOT thực hiện xoay một biểu thức giá trị bảng (table valued expression) thành một bảng khác bằng việc đưa các giá trị duy nhất của một cột thành các cột và thực hiện các hàm thống kê trên các cột còn lại.
- + Toán tử UNPIVOT thực hiện quá trình ngược lại với quá trình thực hiện của toán tử PIVOT, xoay các cột của biểu thức bảng thành giá trị của một cột.

11.2.4. Toán tử OUTPUT

Sử dụng trong các câu lệnh INSERT, UPDATE, UPDATE

Cú pháp:

```
<OUTPUT_CLAUSE> ::=
{
    [ OUTPUT <dml_select_list> INTO { @table_variable |
output_table } [ ( column_list ) ] ]
    [ OUTPUT <dml_select_list> ]
}
<dml_select_list> ::=
{ <column_name> | scalar_expression } [ [AS]
column_alias_identifier ]
    [ ,...n ]

<column_name> ::=
{ DELETED | INSERTED | from_table_name } . { * |
column_name }
```

11.2.5. Thống kê dữ liệu

Cú pháp chung của câu lệnh SELECT có dạng:

```
SELECT [DISTINCT] [TOP n] <danhsach_tên_cột>/<Biểu_thức_1> [AS <Tên_cột_mới>] [...]
[/ * [INTO <tên_bảng_mới>]
FROM <tên_bảng> [bí_danh_1] [<điều_kiện_nối> <tên_bảng_2> [...]
[WHERE <điều_kiện_lọc1> [AND | OR <điều_kiện_lọc2>]]
[GROUP BY <DS_cột_cần_nhóm>
[HAVING <điều_kiện_nhóm>]
[ORDER BY <biểu_thức_sắp_xếp> [ASC | DESC],...]
[UNION | INTERSECT | EXCEPT <câu_truy_vấn_khác>]
```

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.

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.

- Sử dụng kết hợp với mệnh đề *GROUP BY* và các tùy chọn mở rộng ***ROLLUP, CUBE*** với các hàm tổng hợp dữ liệu trong hệ quản trị CSDL SQL Server.

Trong SQL Server 2022, để tăng cường hiệu suất truy vấn ta sử dụng tính năng Adaptive Query Processing và Intelligent Query Processing.

11.2.6. Phân hạng dữ liệu

Các hàm Ranking cho phép bạn có thể đánh số liên tục (xếp loại) cho các tập hợp kết quả. Các hàm này có thể được sử dụng để cung cấp số thứ tự trong hệ thống đánh số tuần tự khác nhau. Có thể hiểu đơn giản như sau: bạn có từng con số nằm trên từng dòng liên tục, tại dòng thứ nhất xếp loại số 1, dòng thứ 2 xếp loại số là 2... Bạn có thể sử dụng hàm ranking theo các nhóm số tuần tự, mỗi một nhóm sẽ được đánh số theo lược đồ 1,2,3 và nhóm tiếp theo lại bắt đầu bằng 1, 2, 3 ...

Hàm ROW_NUMBER

Hàm đầu tiên tôi muốn nói tới là ROW_NUMBER. Hàm này trả lại một dãy số tuần tự bắt đầu từ 1 cho mỗi dòng hay nhóm trong tập hợp kết quả. Hàm ROW_NUMBER sẽ có cú pháp sau:

ROW_NUMBER () OVER ([])

Trong đó:

là cột hay tập hợp các cột được sử dụng để quyết định việc gộp nhóm cho hàm ROW_NUMBER áp dụng cho việc đánh số tuần tự.

là một cột hay tập hợp các cột được sử dụng để sắp xếp tập hợp kết quả trong nhóm (partition)

Hàm RANK

Đôi khi bạn muốn một dòng có cùng sắp xếp giá trị cột như các dòng khác có cùng một xếp loại. Nếu thế thì hàm RANK () có thể giúp bạn. Hàm RANK có cú pháp như sau:

RANK () OVER ([])

Trong đó:

là một cột hay tập hợp các cột được sử dụng để quyết định việc đánh số liên tục trong

hàm RANK

là một cột hay tập hợp các cột được sử dụng để sắp xếp tập hợp kết quả trong nhóm (partition)

Hàm RANK sẽ đánh số liên tục một tập hợp bản ghi nhưng khi có 2 dòng có cùng giá trị sắp xếp thì hàm sẽ đánh giá là cùng bậc giá trị. Giá trị xếp loại vẫn sẽ tăng kể cả khi có 2 dòng cùng giá trị, vì vậy khi đánh giá một giá trị sắp xếp tiếp theo thì số thứ tự vẫn tiếp tục được đánh nhưng sẽ tăng thêm 1 giá trị vào các dòng tiếp theo trong tập hợp.

Hàm DENSE_RANK

Hàm DENSE_RANK cũng giống như hàm RANK, tuy vậy, hàm này không cung cấp khoảng cách giữa các số xếp loại. Thay vào đó, hàm này sẽ xếp loại liên tục cho từng giá trị ORDER BY cụ thể. Với hàm DENSE_RANK, kể cả khi có hai dòng có cùng giá trị xếp loại thì dòng tiếp theo vẫn chỉ tăng thêm một giá trị so với dòng trên. Hàm DENSE_RANK có cú pháp như hàm RANK.

Hàm NTILE

Hàm cuối cùng là hàm NTILE. Đây là hàm được sử dụng để phá vỡ tập hợp bản ghi trong một số cụ thể của các nhóm. Hàm NTILE cũng sử dụng cú pháp như các hàm ranking khác.

11.2.7. Một số hàm cơ bản trong SQL Server

Thành thạo việc sử dụng các hàm cơ bản ứng với các kiểu dữ liệu khác nhau trong SQL Server.

11.2.8. Khung nhìn (VIEW)

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.

-Tạo khung nhìn

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

CREATE VIEW tên_khung_nhìn[(danh_sách_tên_cột)] AS câu_lệnh_SELECT

- 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.

- Phải đặt tên cho các cột của khung nhìn nếu khung nhìn tại các cột chứa biểu thức tính toán.

- 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.

- 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*

- 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.

11.2.9 Con trỏ (Cursor)

Một con trỏ là một đối tượng cơ sở dữ liệu mà các ứng dụng sử dụng để xử lý dữ liệu theo từng hàng. Sử dụng câu lệnh T-SQL DECLARE CURSOR để định nghĩa các thuộc tính của một con trỏ T-SQL.

Để khai báo một con trỏ chúng ta sử dụng câu lệnh T-SQL trên Query Editor. Cú pháp khai báo con trỏ tổng quát như sau:

DECLARE *tên_cursor* CURSOR

[LOCAL | GLOBAL]

[FORWARD ONLY | SCROLL]

[STATIC | DYNAMIC | KEYSET]

[READ_ONLY | SCROLL_LOCK]

FOR *câu_lệnh_select*

[FOR UPDATE [OF *danh_sách_cột_cập_nhật*]]

Trong đó:

Tên_cursor: tên của biến kiểu cursor

Local | global: phạm vi hoạt động của biến local biến cục bộ (phạm vi là thủ tục, batch(lô) các lệnh, trigger; global biến toàn cục (tham chiếu đến bất kỳ thủ tục nào của kết nối tạo ra biến cursor đó.

FORWARD_ONLY: duyệt từ bản ghi đầu đến cuối cùng, theo chiều đi tới.

SCROLL: Cursor được phép di chuyển tới lui, qua lại các dòng bản ghi bên trong cursor.

STATIC: Cursor tĩnh. Nghĩa là khi có sự thay đổi bên dưới dữ liệu gốc (base table) thì các thay đổi đó không được cập nhật tự động trong dữ liệu của cursor.

DYNAMIC: Cursor động, nghĩa là khi có thay đổi dữ liệu gốc (base table) thì các thay đổi đó tự động cập nhật trong dữ liệu kiểu cursor.

KEYSET: Gần như DYNAMIC. Nghĩa là những thay đổi trên cột không là khoá chính trong bảng gốc (base table) sẽ tự động cập nhật trong dữ liệu cursor. Tuy nhiên, những bản ghi vừa thêm mới hoặc những bản ghi vừa huỷ bỏ sẽ không hiển thị trong dữ liệu cursor có kiểu là keyset.

SCROLL LOCK: Chỉ định SQL SERVER khoá các bản ghi cần phải thay đổi giá trị hoặc bị huỷ bỏ bên trong bảng nhằm đảm bảo hành động cập nhật luôn thành công.

SELECT: Lựa chọn đến các cột trong bảng mà ta cần đọc. Nhớ lệnh select trong cursor không chứa các mệnh đề INTO.

DANH_SACH_CAC_COT_CAP_NHAT: Chỉ định danh sách các cột sẽ được phép thay đổi giá trị trong cursor. Mặc định là tất cả các cột trong mệnh đề select sẽ được phép thay đổi giá trị nếu dữ liệu cursor không phải là chỉ đọc.

- Để duyệt con trỏ, có thể có các tùy chọn sau:

FETCH [NEXT| PRIOR | FIRST |LAST| ABSOLUTE n| FETCH RELATIVE n]
FROM <Cursor_name> [INTO <Var_list>]

FETCH FIRST: Truy xuất hàng đầu tiên.

FETCH NEXT: Truy xuất hàng tiếp theo.

FETCH PRIOR: Truy xuất hàng trước hàng hiện tại.

FETCH LAST: Truy xuất hàng cuối cùng.

FETCH ABSOLUTE n: N là một số nguyên dương, truy xuất hàng n trong con trỏ.

FETCH RELATIVE n: truy xuất đến một bản ghi liên quan.

@@FETCH_STATUS: Trả về một số nguyên cho biết kết quả của lệnh truy xuất cuối cùng của con trỏ.

@@CURSOR_ROWS: Trả về tổng số hàng hiện tại trong con trỏ đang mở.

11.2.10. Giao dịch (TRANSACTION)

Giao tác (transaction) là 1 tập hợp có thứ tự các thao tác (statement) truy xuất dữ liệu trên CSDL thành 1 đơn vị công việc logic (xem là 1 thao tác nguyên tố), chuyển CSDL từ trạng thái nhất quán này sang trạng thái nhất quán khác.

Transaction là một đơn vị công việc trong nó bao gồm nhiều việc nhỏ, các việc này được thực hiện thành công thì Transaction thành công, dữ liệu thay đổi trong quá trình thực hiện của Transaction sẽ được cập nhật. Nếu trong quá trình có phát sinh lỗi thì Transaction sẽ lập lại (Roll Back hoặc Cancel), dữ liệu không được cập nhật. Một phiên giao dịch có 4 đặc tính ACID (Atomicity, Consistency, Isolation, Durability).

T-SQL sử dụng bốn câu lệnh để tạo giao dịch:

- ☐ BEGIN TRANSACTION
- ☐ COMMIT TRANSACTION
- ☐ ROLLBACK TRANSACTION
- ☐ SAVE TRANSACTION

Các biến dưới đây được sử dụng trong quá trình thực hiện giao dịch

@@ERROR

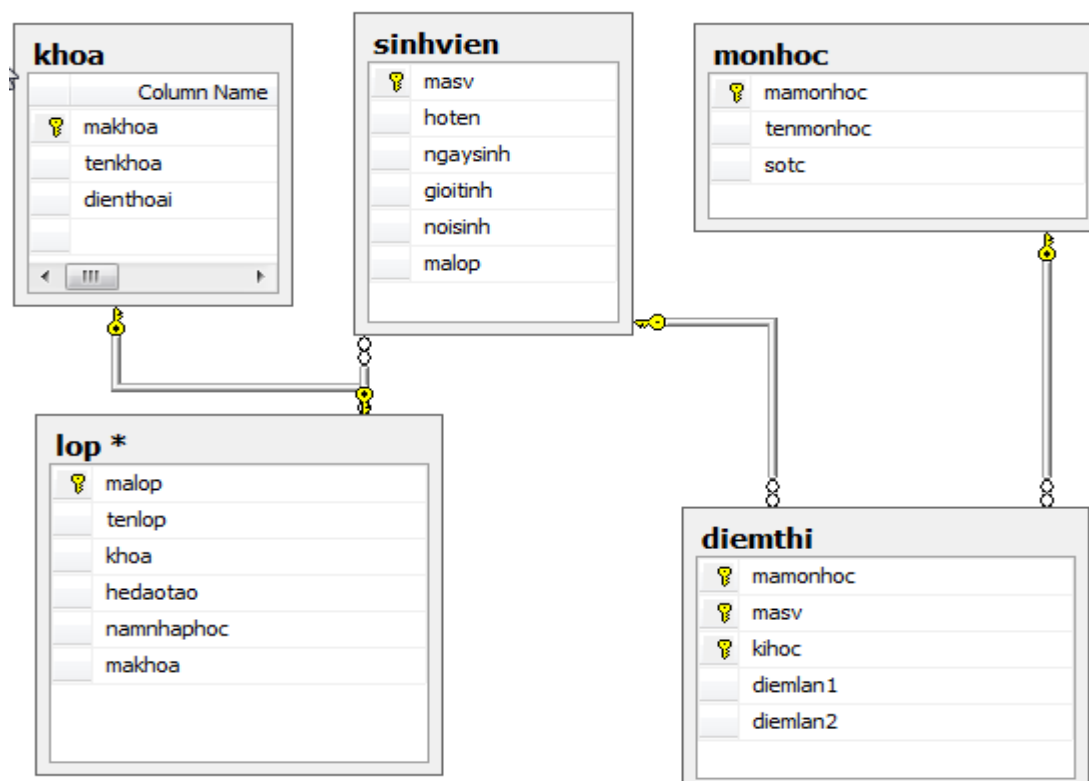
@@TRANCOUNT – Nó được sử dụng để đếm số lần các giao dịch mở đồng thời.

Tùy chọn XACT_ABORT ON

Khi XACT_ABORT được đặt thành ON, SQL Server mới cư xử đúng như mong đợi – khi gặp bất kỳ lỗi nào nó hủy bỏ toàn bộ transaction và quay lui trở lại như lúc ban đầu.

11.3. Hướng dẫn thực hành

Mở CSDL QLSV đã được tạo ở bài thực hành 9.



Yêu cầu 1: Sử dụng ngôn ngữ T- SQL thực hiện các câu lệnh sau:

1. Thực thi truy vấn trên nhiều bảng.
2. Thực thi truy vấn với các toán tử nâng cao.
3. Thống kê dữ liệu bằng cách sử dụng GROUP BY..HAVING, tùy chọn ROLLUP, CUBE trong GROUP BY
4. Thực hiện phân hạng dữ liệu.
5. Sử dụng các hàm cơ bản.

Bài 1:

Hiển thị thông tin của các lớp chưa có sinh viên nào nhập vào.

Hướng dẫn:

- Thông tin hiển thị: các lớp
- Điều kiện: Các lớp này chưa có sinh viên nào nhập vào
- Truy vấn được viết bằng nhiều cách, sau đây là một số cách:

+ Cách 1: Sử dụng phép nối LEFT Excluding JOIN

```
SELECT LOP.malop, TENLOP
FROM lop LEFT JOIN sinhvien ON lop.malop
=sinhvien.malop
WHERE sinhvien.malop IS NULL
```

+ Cách 2: Sử dụng truy vấn con

```
SELECT malop, TENLOP
FROM lop
WHERE malop NOT IN (SELECT malop FROM sinhvien )
```

Bài 2:

Hiện thị thông tin của mỗi sinh viên gồm: MASV, HOTEN, DIEMlan1 thuộc kì học 1 của các môn học có số tín chỉ ≥ 3 và các sinh viên đó quê ở Hưng Yên.

Hướng dẫn:

- Thông tin lấy ở trên các bảng SINHVIEN, DIEMTHI, MONHOC
- Điều kiện tại các cột kihoc, noisinh, sotc
- Thực hiện: Có thể sử dụng truy vấn thông thường hoặc truy vấn sử dụng phép kết nối INNER JOIN.

```
SELECT SINHVIEN.MASV, HOTEN, DIEMlan1
FROM sinhvien INNER JOIN diemthi
ON sinhvien.masv =diemthi.masv
INNER JOIN monhoc ON monhoc.mamonhoc =diemthi.mamonhoc
WHERE kihoc =1 AND sotc $\geq$ 3 AND noisinh=N'Hưng Yên'
```

Bài 3: Hiện thị 3 môn học có số tín chỉ cao nhất tính từ cao xuống thấp.

Hướng dẫn:

- Sử dụng mệnh đề TOP, số tín chỉ của môn học được sắp xếp theo chiều giảm dần.

```
SELECT TOP (3) MAMONHOC, TENMONHOC, SOTC
FROM monhoc
ORDER BY sotc DESC
```

Bài 4: Thống kê số lượng sinh viên của từng lớp theo giới tính.

Hướng dẫn:

- Sử dụng toán tử PIVOT, cột tính toán là masv, cột dùng để PIVOT là gioitinh, cột dùng để sắp xếp là Malop.
- Thông tin hiển thị là Malop, 'Nữ', 'Nam'.

```
SELECT malop, [0] as Nữ, [1] as Nam
FROM
(SELECT malop, masv, gioitinh FROM sinhvien) s
pivot
(
count (masv)
for gioitinh in
([0], [1])
) as pvt
ORDER BY malop
```

Bài 5: Tính tổng số lượng sinh viên của mỗi lớp. Biết rằng thông tin hiển thị gồm: Malop, Tenlop, SLSV. Trong đó, SLSV là thuộc tính tự đặt.

Hướng dẫn:

- Đây là một truy vấn dạng thống kê gom nhóm.
- Nhóm dữ liệu để gom: Malop, Tenlop
- Cột tính toán: MaSv; hàm sử dụng là Count

```
SELECT lop.malop, TENLOP, SLSV=COUNT(MASV)
FROM lop LEFT JOIN sinhvien ON lop.malop
=sinhvien.malop
GROUP BY lop.malop, TENLOP
```

Bài 6: Hiển thị thông tin về các lớp có đông sinh viên nhất. Biết rằng thông tin hiển thị gồm: Malop, Tenlop, SLSV. Trong đó, SLSV là thuộc tính tự đặt.

Hướng dẫn:

- Tương tự như bài 5 nhưng truy vấn này có điều kiện đối với mỗi nhóm.
- Truy vấn con bên trong đưa ra số lượng sinh viên thuộc mỗi nhóm.
- Thực hiện:

+ Cách 1:

```
SELECT lop.malop, TENLOP, SLSV=COUNT(MASV)
FROM lop LEFT JOIN sinhvien ON lop.malop
=sinhvien.malop
GROUP BY lop.malop, TENLOP
HAVING COUNT(MASV) >= ALL (SELECT COUNT(MASV)
FROM sinhvien
GROUP BY malop)
```

+ Cách 2: Sử dụng mệnh đề TOP (1) kèm theo thuộc tính WITH TIES sẽ lấy ra thông tin của các lớp có số lượng sinh viên bằng với lớp có số lượng sinh viên đông nhất.

```
SELECT TOP(1) WITH TIES lop.malop, TENLOP,
SLSV=COUNT(MASV)
FROM lop LEFT JOIN sinhvien ON lop.malop
=sinhvien.malop
GROUP BY lop.malop, TENLOP
```

ORDER BY COUNT (MASV) DESC

Bài 7: Tính điểm trung bình lần 1 của mỗi sinh viên, thông tin hiển thị masv, hoten, diemtbl1

Ta sử dụng truy vấn gom nhóm theo masv, hoten; cột diemtbl1 cần tính theo công thức tính điểm trung bình.

Truy vấn này thực hiện trên 3 bảng Sinhvien, Monhoc, Diemthi.

```
SELECT s.Masv, HOTEN, ROUND(sum(DIEML1*sotc)/sum(sotc),1)
AS diemTbl1
FROM SINHVIEN S INNER JOIN DiemThi D ON S.masv =D.masv
INNER JOIN MONHOC M ON D.mamonhoc =M.mamonhoc
GROUP BY s.masv, HOTEN
```

Kết quả như sau:

	Masv	HOTEN	diemTbl1
1	sv01	Nguyễn Thị Lan	7.1
2	sv02	Trần Văn Hùng	7.7
3	sv03	Trần Thị Anh	6.2
4	sv04	Nguyễn Văn Hùng	4.6
5	sv05	Nguyễn Thị Hằng	8.1

Bài 8: Trong việc quản lý sinh viên, người dùng còn có nhu cầu tính điểm trung bình tất cả các môn của mỗi sinh viên và điểm trung bình chung của tất cả sinh viên như sau:

	Họ tên	tên môn	Diemtbl1
1	Nguyễn Thị Hằng	Cơ sở dữ liệu	9
2	Nguyễn Thị Hằng	Hệ quản trị CSDL	7
3	Nguyễn Thị Hằng	Java nâng cao	8
4	Nguyễn Thị Hằng	All mon hoc	8.1
5	Nguyễn Thị Lan	C# nâng cao	7
6	Nguyễn Thị Lan	Cơ sở dữ liệu	8
7	Nguyễn Thị Lan	Hệ quản trị CSDL	6
8	Nguyễn Thị Lan	All mon hoc	7.1
9	Nguyễn Văn Hú...	C# nâng cao	4
10	Nguyễn Văn Hú...	Cơ sở dữ liệu	4
11	Nguyễn Văn Hú...	Java nâng cao	4
12	Nguyễn Văn Hú...	Lập trình HDT	7
13	Nguyễn Văn Hú...	All mon hoc	4.6
14	Trần Thị Anh	C# nâng cao	4
15	Trần Thị Anh	Java nâng cao	7
16	Trần Thị Anh	Lập trình HDT	8
17	Trần Thị Anh	All mon hoc	6.2
18	Trần Văn Hùng	Cơ sở dữ liệu	8
19	Trần Văn Hùng	Hệ quản trị CSDL	9
20	Trần Văn Hùng	Lập trình HDT	6
21	Trần Văn Hùng	All mon hoc	7.7
22	All sinh vien	All mon hoc	6.6

Khi đó, ta sử dụng thêm các tùy chọn ROLLUP và CUBE vào trong mệnh đề GROUP BY

```
SELECT HOTEN, TenMonHoc,  
ROUND(sum(DIEML1*sotc)/sum(sotc),1)  
FROM SINHVIEN S INNER JOIN diemthi D ON S.masv =D.masv  
INNER JOIN MONHOC M ON D.mamonhoc =M.mamonhoc  
  
GROUP BY HOTEN , TENMONHOC WITH ROLLUP
```

Trong kết quả trên, tùy chọn ROLLUP đã chèn thêm các dòng tính điểm trung bình tất cả môn học của từng sinh viên. Tại ô tên môn học, ROLLUP cho giá trị NULL. Ta có thể sửa các giá trị NULL này thành tên gọi mang ý nghĩa tại các ô đó bằng cách sử dụng Hàm GROUPING() để xác định xem cột[hoten] hoặc [Tenmonhoc] có phải hiện đang được nhóm hay không với lệnh SELECT như sau:

```
SELECT  
CASE when GROUPING([hoten]) = 1 THEN 'All sinh vien'  
ELSE [hoten] END  
AS [Họ tên],  
CASE when GROUPING([tenmonhoc]) = 1 THEN 'All mon hoc'  
ELSE [tenmonhoc] END  
AS [tên môn],  
ROUND(sum(DIEML1*sotc)/sum(sotc),1) as Diemtbl1  
FROM SINHVIEN S INNER JOIN diemthi D ON S.masv =D.masv  
INNER JOIN MONHOC M ON D.mamonhoc =M.mamonhoc  
  
GROUP BY HOTEN , TENMONHOC WITH ROLLUP
```

Ta có nhận xét là, trong kết quả trên, ta chỉ tính được điểm trung bình từng môn của từng sinh viên cũng như điểm trung bình tất cả các môn của từng sinh viên, (tức là tính trung bình theo sinh viên), nhưng không có điểm trung bình của tất cả sinh viên cho từng môn. (chẳng hạn, điểm trung bình của tất cả sinh viên trong môn Cơ sở dữ liệu). Để tính loại điểm trung bình thứ 3, ta có thể đảo ngược vị trí của [Hoten] và [Tenmonhoc] trong mệnh đề GROUP BY như sau:

```
SELECT  
CASE when GROUPING([Tenmonhoc]) = 1 THEN 'All mon hoc'  
ELSE [Tenmonhoc] END  
AS [tên môn],  
CASE when GROUPING([hoten]) = 1 THEN 'All sinh vien'  
ELSE [hoten] END  
AS [Họ tên],  
ROUND(sum(DIEML1*sotc)/sum(sotc),1) as Diemtbl1  
FROM SINHVIEN S INNER JOIN bangdiem D ON S.masv =D.masv  
INNER JOIN MONHOC M ON D.Mamonhoc =M.Mamonhoc  
GROUP BY TENMONHOC, HOTEN WITH ROLLUP
```

Tuy nhiên, để kết quả có cả 3 loại điểm trung bình trên, tùy chọn ROLLUP không đáp ứng được, mà ta cần sử dụng tùy chọn CUBE. Thay thế tùy chọn CUBE cho ROLLUP

```
SELECT  
CASE when GROUPING([hoten]) = 1 THEN 'All sinh vien' ELSE  
[hoten] END  
AS [hoten],
```

```
CASE when GROUPING ([Tenmonhoc]) = 1 THEN 'All mon hoc' ELSE  
[Tenmonhoc] END  
AS [tên môn],  
ROUND(sum(DIEML1*sotc)/sum(sotc),1) AS DIEMTBL1  
FROM SINHVIEN S INNER JOIN bangdiem D ON S.masv =D.masv  
INNER JOIN MONHOC M ON D.Mamonhoc =M.Mamonhoc  
GROUP BY TENMONHOC, HOTEN WITH CUBE
```

Bài 9: Cho cấu trúc bảng và dữ liệu nhập như sau:

```
CREATE TABLE DBO.Khachhang(  
KhachHang_ID INT IDENTITY PRIMARY KEY,  
Ten NVARCHAR(100),  
Email VARCHAR(100)  
)  
  
CREATE TABLE dbo.KhachHangLuu(  
KhachHang_ID INT,  
Ten NVARCHAR(50),  
Email VARCHAR(100)  
)  
GO  
INSERT INTO dbo.KhachHang VALUES(N'Ý Lan','ylan@ylan.com')  
INSERT INTO dbo.KhachHang VALUES(N'Tuần Ngọc','tuannhoc@tuannhoc.com')  
INSERT INTO dbo.KhachHang VALUES(N'Thái Hiền','thaihien@thaihien.com')  
  
INSERT INTO dbo.KhachHang VALUES(N'Ngọc Hạ','ngocha@ngocha.com')  
  
SELECT * FROM dbo.KhachHang
```

Hãy thực hiện việc lưu thông tin của khách khi xóa khách hàng có mã KhachHang_ID là 4 và khi cập nhật email của khách hàng có KhachHang_ID vào bảng KhachHangLuu.

Hướng dẫn:

Sử dụng mệnh đề OUTPUT trong các lệnh DELETE, INSERT, UPDATE, để trả về các bản ghi bị ảnh hưởng hoặc lưu vào một bảng khác. Tính năng này rất hữu dụng khi bạn muốn lưu lại, chẳng hạn với lệnh DELETE, các bản ghi cần xóa sang một nơi khác để có thể tra cứu về sau.

```
-- xóa KhachHang_ID = 4 (Ngọc Hạ)  
DELETE dbo.KhachHang  
OUTPUT DELETED.* INTO dbo.KhachHangLuu  
WHERE KhachHang_ID = 4  
  
-- cập nhật email của KhachHang_ID = 3  
UPDATE dbo.KhachHang  
SET Email = N'me@thaihien.com'  
OUTPUT DELETED.* INTO dbo.KhachHangLuu  
WHERE KhachHang_ID = 3  
  
SELECT * FROM dbo.KhachHangLuu
```

Yêu cầu 2: Thiết kế và thao tác với View sử dụng công cụ SQL Server Object Explorer hoặc câu lệnh SQL trên Query Editor.

- ✓ Tạo ra các kiểu con trỏ khác nhau và sử dụng chúng trong ứng dụng.
- ✓ Viết các Transaction thực hiện các yêu cầu cụ thể.
- ✓ Vận dụng để sử dụng linh hoạt trong các ứng dụng.

Bài 10:

Thiết kế View có tên View_DSSV để lưu trữ thông tin của sinh viên đó gồm: MaSV, Hoten, Ngaysinh, Gioitinh, Noisinh, Tenlop

Phân tích đề bài

- Để tạo một View trước hết phải hiểu ý nghĩa của View:

View là một bảng dữ liệu ảo được định nghĩa bằng một truy vấn. Một view không tồn tại vật lý như các tập hợp dữ liệu lưu trữ trong cơ sở dữ liệu. Tuy nhiên, nó giống như một bảng dữ liệu thực với tập hợp các hàng, các cột dữ liệu. Dữ liệu của View nhận được từ các bảng thực trong cơ sở dữ liệu.

- Chẳng hạn như view View_DSSV ở đây gồm có 6 cột, lấy thông tin ở 2 bảng là Sinhvien và lop.

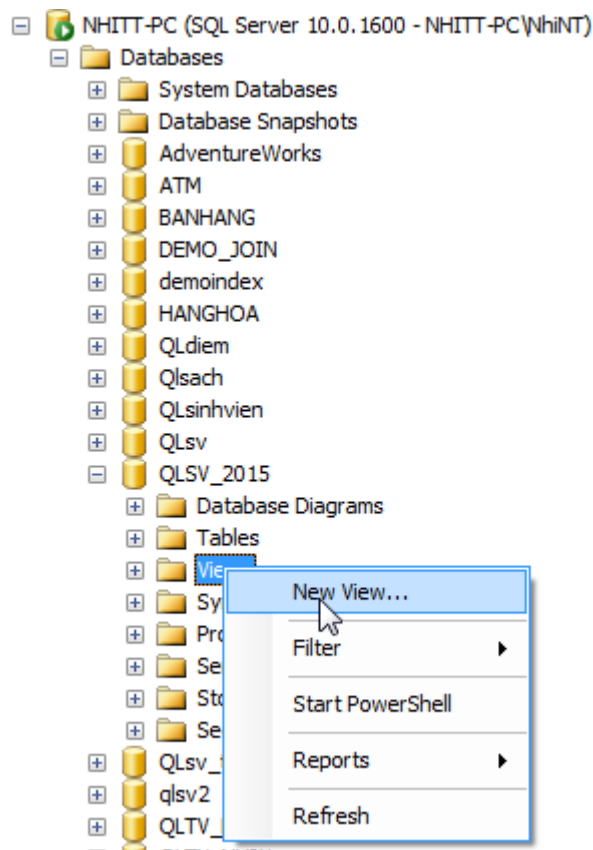
- Để thiết kế View trong SQL Server ta có thể dùng 2 cách sau:

- + Dùng công cụ SQL Server Object Explorer
- + Dùng Query Editor

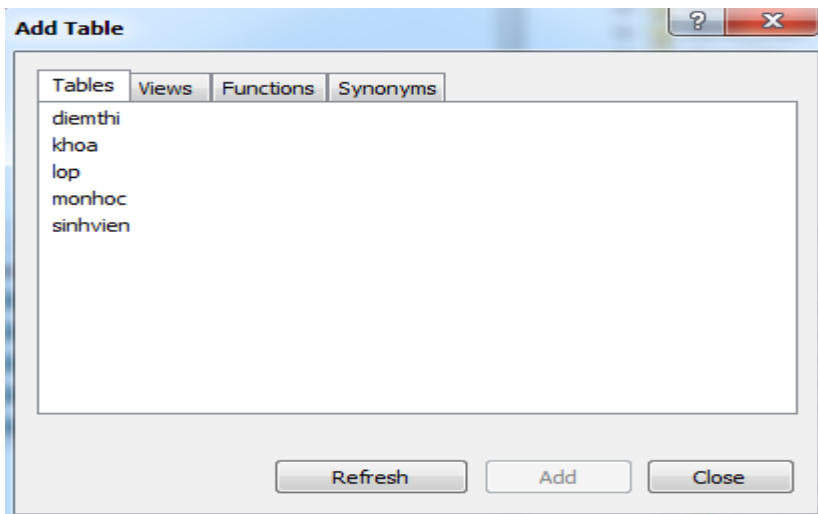
Hướng dẫn thực hiện:

Dùng công cụ SQL Server Object Explorer

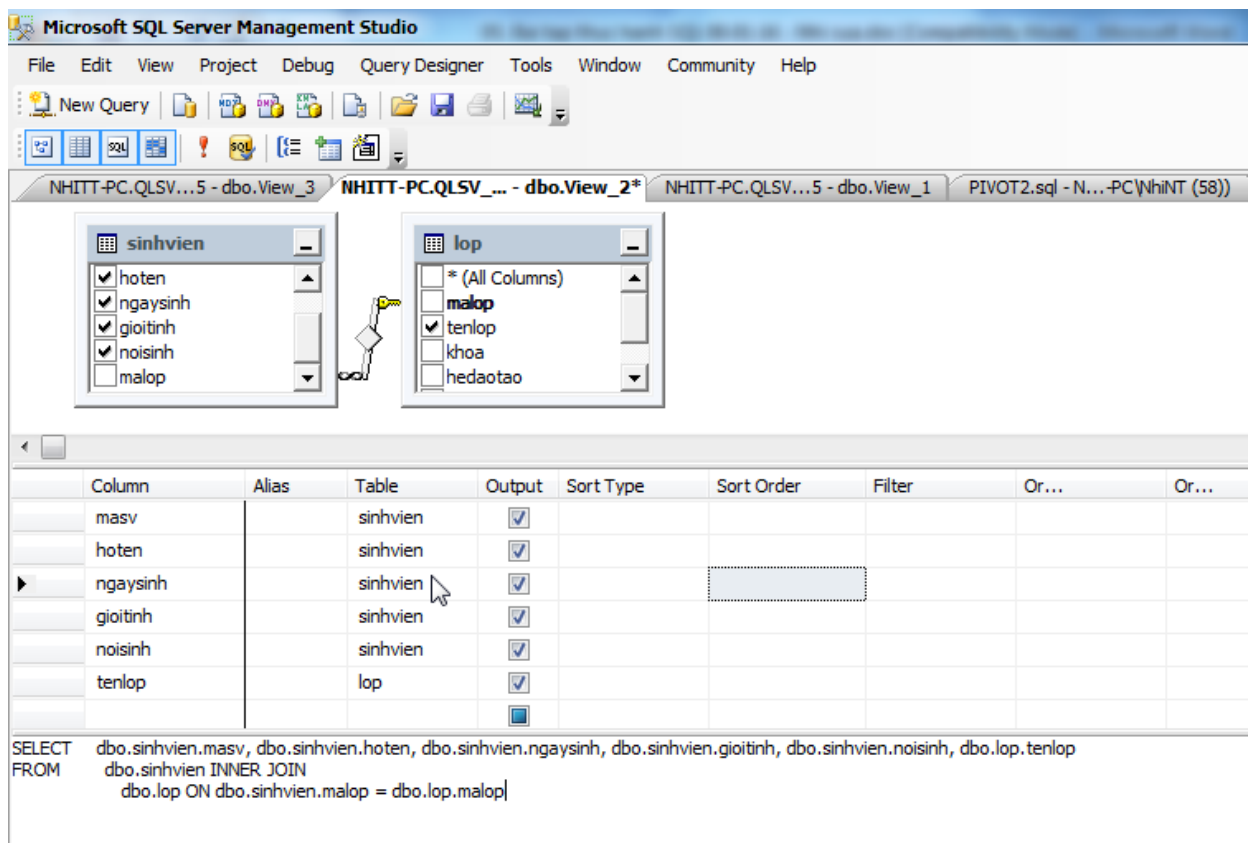
Bước 1: Nhấn đúp chuột vào CSDL cần thao tác (QLSV), khi đó một danh sách các đối tượng sẽ hiện ra




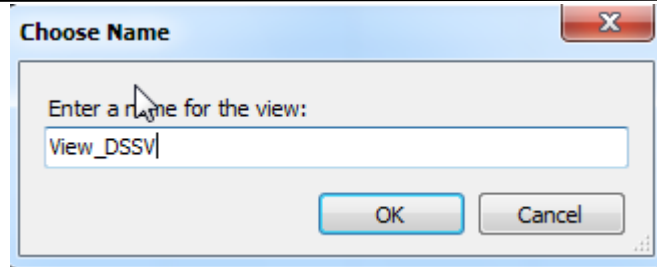
Bước 2: Nhấn chuột phải vào “View” chọn “New view” khi đó một cửa sổ mới sẽ hiện ra cho phép ta chọn các bảng thực chứa dữ liệu của View cần tạo:



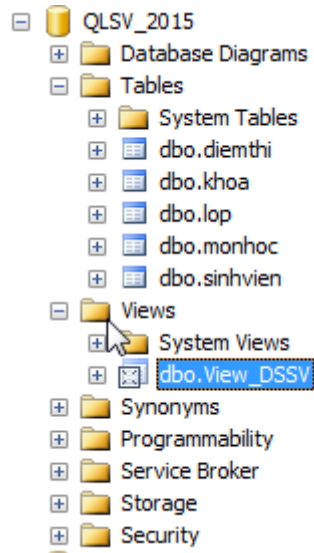
Bước 3: Nhấn “Add” để thêm các bảng. Khi đó ta thấy thiết kế của View hiện ra, đánh dấu các trường cần chọn.



Bước 4: Lưu lại bằng cách chọn biểu tượng hình cái đĩa mềm  hoặc nhấn tổ hợp phím “Ctrl + S”, sau đó nhập tên view vào hộp thoại “Choose Name” (ở đây tên view là View_DSSV) như hình sau:



Nhấn OK để lưu lại. Khi đó view có tên View_DSSV đã được tạo.



Dùng Query Editor

- Cú pháp tạo View là:

```
CREATE VIEW tên_View [danh_sách_tên_cột]
AS
câu_lệnh_ SELECT
```

- Tạo view View_DSSV

```
CREATE VIEW VIEW_DSSV
AS
```

```
SELECT    dbo.sinhvien.masv, dbo.sinhvien.hoten, dbo.sinhvien.ngaysinh,
          dbo.sinhvien.gioitinh, dbo.sinhvien.noisinh, dbo.lop.tenlop
FROM      dbo.sinhvien INNER JOIN
          dbo.lop ON dbo.sinhvien.malop = dbo.lop.malop
```

- Sau khi view View_DSSV được tạo, ta có thể thực thi truy vấn, cập nhật dữ liệu trên View như bảng.

Bài 11:

Sửa đổi view View_DSSV trên thành View mới không có cột gioitinh.

Phân tích đề bài

- Để sửa đổi một View trong SQL Server, ta cũng có thể dùng 2 cách:
 - + Dùng SQL Server Object Explorer
 - + Dùng Query Editor

Hướng dẫn thực hiện

Dùng công cụ SQL Server Object Explorer

- Với cách này chỉ cần chọn View muốn sửa, click chuột phải để mở VIEW ở chế độ thiết kế. Sau đó, tiến hành sửa đổi VIEW theo yêu cầu và lưu lại.
- Thực thi VIEW để xem kết quả.

Dùng Query Editor

- Cú pháp sửa đổi View là:

```
ALTER VIEW tên_View  
AS  
câu_lệnh_SELECT
```

- Sửa view View_DSSV

```
ALTER VIEW VIEW_DSSV  
AS
```

```
SELECT          dbo.sinhvien.masv,          dbo.sinhvien.hoten,  
                dbo.sinhvien.ngaysinh, dbo.sinhvien.gioitinh, dbo.sinhvien.noisinh,  
                dbo.lop.tenlop  
FROM            dbo.sinhvien INNER JOIN  
                dbo.lop ON dbo.sinhvien.malop = dbo.lop.malop
```

Bài 12: Sử dụng con trỏ để truy vấn lấy từng bản ghi của bảng sinh viên trong bảng sinh viên.

Phân tích đề bài

- Theo yêu cầu của bài, dữ liệu trả về không phải một tập các bản ghi mà từng bản ghi nên ta phải giải quyết yêu cầu bằng cách dùng con trỏ.

- **Bước 1:** Khai báo một con trỏ có tên CUR_SV cho câu truy vấn SELECT Masv, Hoten, Ngaysinh, Gioitinh, Noisinh FROM SINHVIEN. Sau câu lệnh khai báo này biến con trỏ sẽ chứa tất cả các hàng dữ liệu của bảng.

- **Bước 2:** Sử dụng con trỏ, phải tuân theo các bước sau:

+ Mở con trỏ

+ Truy xuất đến bản ghi đầu tiên của con trỏ.

+ Duyệt con trỏ: Sử dụng biến toàn cục @@FETCH_STATUS=0 kiểm tra xem trạng thái của con trỏ đã ở bản ghi cuối của bảng chưa để truy xuất đến bản ghi tiếp theo của bảng.

+ Đóng con trỏ.

+ Giải phóng con trỏ.

Đoạn chương trình minh họa

```
--1.Khai báo con trỏ
```

```
DECLARE cur_SV CURSOR
```

```
--DECLARE cur_SV CURSOR SCROLL
```

```
FOR SELECT Masv, Hoten, Ngaysinh, Gioitinh, Noisinh  
FROM sinhvien
```

```
--2.Sử dụng con trỏ
```

```
--Bước 1:Mở con trỏ
OPEN cur_SV
--Bước 2:Truy xuất đến hàng đầu tiên của con trỏ
FETCH NEXT FROM cur_SV
--FETCH FIRST FROM cur_SV
--Bước 3:Duyệt con trỏ
WHILE @@fetch_status=0
    FETCH NEXT FROM cur_SV
--Bước 4:Đóng con trỏ
CLOSE cur_SV
--Bước 5: Giải phóng con trỏ
DEALLOCATE cur_SV
```

Bài 13:

Viết một đoạn chương trình gồm hai giao dịch (1 giao dịch thực hiện sai và một giao dịch thực hiện đúng) thực hiện việc nhập dữ liệu vào bảng t2 chứa khóa ngoại tham chiếu đến bảng t1 của CSDL có tên UTE_TEST .

Phân tích đề bài

- Theo đề bài, ta tạo CSDL có tên UTE_TEST, tạo cấu trúc 2 bảng t1, t2 đơn giản.
- Viết 1 giao dịch có tùy chọn SET XACT_ABORT OFF, giao dịch này key 2 bị lỗi và đã Roll back, nhưng do có tùy chọn này mà giao dịch vẫn thành công. Điều này làm cho giao dịch hoạt động không đúng như mong đợi.
- Viết 1 giao dịch có tùy chọn SET XACT_ABORT ON, giao dịch này key 5 bị lỗi và do có tùy chọn này mà giao dịch bị thất bại.

Đoạn chương trình mẫu

```
IF EXISTS (SELECT * FROM sys.databases WHERE name =
'UTE_TEST')
    DROP DATABASE UTE_TEST
GO
CREATE DATABASE UTE_TEST
GO
USE UTE_TEST
GO
CREATE TABLE t1 (a int PRIMARY KEY)
CREATE TABLE t2 (a int REFERENCES t1(a))
GO

INSERT INTO t1 VALUES (1)
INSERT INTO t1 VALUES (3)
INSERT INTO t1 VALUES (4)
```

```
INSERT INTO t1 VALUES (6)
GO
SET XACT_ABORT OFF
GO
BEGIN TRAN
INSERT INTO t2 VALUES (1)
INSERT INTO t2 VALUES (2) /* Foreign key error */
INSERT INTO t2 VALUES (3)
COMMIT TRAN
GO

SET XACT_ABORT ON
GO

BEGIN TRAN
INSERT INTO t2 VALUES (4)
INSERT INTO t2 VALUES (5) /* Foreign key error */
INSERT INTO t2 VALUES (6)
COMMIT TRAN
GO
```

Lưu ý:

TRY CATCH không bắt được một số loại lỗi, như lỗi timeout hoặc tên bảng/tên cột không tồn tại. Khi gặp những lỗi này chương trình sẽ gục ngã ngay tại đó và không nhảy được tới phần BEGIN CATCH để mà rollback. Nếu SET XACT_ABORT là OFF, các lệnh chạy trước lệnh chứa lỗi vẫn thực hiện thành công. Chỉ khi SET XACT_ABORT là ON thì toàn bộ các lệnh mới được tự động ROLLBACK.

Ví dụ:

```
CREATE TABLE #t2(i INT, CONSTRAINT ck2 CHECK (i<>2) )
SET XACT_ABORT OFF
-- SET XACT_ABORT ON
BEGIN TRAN
BEGIN TRY
    INSERT #t2 SELECT 1
    INSERT #t2 SELECT 3
    INSERT #t3 SELECT 100 -- bảng #t3 không tồn tại
COMMIT
END TRY
BEGIN CATCH
    ROLLBACK TRAN
    DECLARE @ErrorMessage VARCHAR(1800)
    SELECT @ErrorMessage = 'Lỗi: ' + ERROR_MESSAGE()
    RAISERROR(@ErrorMessage, 14, 1)
END CATCH
SELECT * FROM #t2
```

11.4. Bài tập tự làm

Bài 1: Cho CSDL QLSV gồm các bảng như trên.

Nhập dữ liệu cho các bảng, tối thiểu 10 bản ghi/1 bảng.

Thực hiện các yêu cầu sau đây:

1. Hiển thị thông tin về các sinh viên sinh ở Hưng Yên và chưa có điểm môn nào. Thông tin gồm MaSV, Hoten, GioiTinh.
2. Hiển thị thông tin của các sinh viên quê ở Thái Nguyên có tuổi lớn hơn 18 và điểm lần 1 của môn 'CSDL' lớn hơn 8. Thông tin hiển thị gồm: Mã sinh viên, họ tên, giới tính, nơi sinh, tuổi. (Biết rằng, tuổi của sinh viên được tính đến năm hiện tại)
3. Hiển thị thông tin về các khoa có các sinh viên đã từng học môn học có tên "Hệ quản trị CSDL".
4. Hiển thị thông tin của 50% sinh viên đầu tiên có tuổi cao nhất.
Gợi ý: Sử dụng từ khóa Top n percent để lấy ra nửa số bản ghi
5. Thống kê số lượng sinh viên theo từng khoa (Sử dụng GROUP BY, GROUP BY với một số toán tử CUBE,...)
6. Thực hiện việc xếp loại lực học cho mỗi sinh viên thuộc kì 1. Thông tin gồm: Masv, Hoten, diemtbl1, Xeploai. (diemTbl1 tính theo công thức, sử dụng cấu trúc CASE để thiết lập giá trị cho cột Xeploai theo DiemTbl1)
7. Hiển thị thông tin của các sinh viên có DiemTBL1 lớn hơn DiemTBL1 của các sinh viên sinh ở Thái Bình. Thông tin hiển thị gồm: MaSV, Hoten, NamSinh, DiemTBL1. (có 2 cột tính toán là Namsinh theo ngaysinh và DiemTBL1)
8. Liệt kê các sinh viên có điểm cao nhất của môn Lập trình C#, thông tin gồm MaSV, Hoten, Diemlan1 (sử dụng hàm Max() trong truy vấn con)
9. Hiển thị thông tin của sinh viên có điểm trung bình lần 1 cao nhất thuộc kì 1, thông tin gồm: maSV, Hoten, DiemTBL1.
10. Hiển thị 5 sinh viên đầu tiên có tuổi được sắp xếp từ thấp -> cao thuộc khoa có mã là 'CNTT'.
11. Hiển thị thông tin các sinh viên bị trượt ít nhất một môn (sử dụng truy vấn con và từ khóa EXISTS hoặc IN)
12. Cho biết danh sách sinh viên đã học hết tất các môn trong bảng monhoc.
13. Thực hiện việc phân hạng cho môn học theo sotc (sử dụng tất cả các hàm phân hạng)
14. Thực hiện việc thống kê số lượng sinh viên của từng khoa theo giới tính (sử dụng toán tử PIVOT).
15. Thử nghiệm một số hàm mới từ phiên bản SQL Server mới
16. Tạo View có tên là w_cau1 lưu trữ các thông tin: Họ tên sinh viên, giới tính, tên môn học, điểm thi lần 1 của sinh viên đó (bằng 2 cách)
17. Sửa đổi view vừa tạo thành view chỉ lưu trữ Họ tên sinh viên, giới tính, điểm thi lần 1 (bằng 2 cách)
18. Tạo view có tên là w_cau3 lưu trữ thông tin gồm mã môn học và số lượng sinh viên đã học môn học đó.
19. Tạo view có tên là w_cau4 lưu trữ thông tin gồm mã lớp, tên lớp, số lượng sinh viên của lớp tương ứng (được tính thông qua cột masv)

20. Tạo view có tên là w_cau5 cho biết tên của sinh viên chưa học môn nào (chưa có điểm môn học đó)
 21. Tạo ra view View_Ttkhoa gồm các thông tin: mã khoa, tên khoa, địa điểm. Sau đó thực hiện các thao tác truy vấn, cập nhật dữ liệu thông qua view View_Ttkhoa đã tạo.
 22. Sử dụng con trỏ để hiển thị thông tin của sinh viên thứ 1 và sinh viên cuối cùng trong bảng sinhvien.
 23. Khai báo một con trỏ có tên là cur_cau7 để hiển thị mã sinh viên và điểm thi lần 1 cao nhất trong tất cả các môn học mà sinh viên đó đã có điểm.
 24. Sử dụng con trỏ để cập nhật số lượng sinh viên của mỗi lớp vào cột siso của bảng Lop (trong bảng Lop ta thêm một cột siso).
 25. Thêm cột XepLoai vào bảng SINHVIEN
- Tính điểm trung bình lần 1 (DiemTBL1) của mỗi sinh viên và thực hiện sử dụng con trỏ để cập nhật dữ liệu cho cột XepLoai theo công thức sau:
- Nếu DiemTBL1 < 5 thì XepLoai = Yếu
- Nếu DiemTBL1 ≥ 5 và DiemTBL1 < 7 thì XepLoai = Trung bình
- Nếu DiemTBL1 ≥ 7 và DiemTBL1 < 8 thì XepLoai = Khá
- Nếu DiemTBL1 ≥ 8 thì DiemTBL1 = Giỏi
26. Viết một giao dịch dùng để kiểm soát dữ liệu nhập vào vào bảng lớp (một số bản ghi có giá trị cụ thể).
 27. Viết giao dịch kiểm soát tính toàn vẹn của dữ liệu khi bạn thêm một sinh viên vào một lớp học bạn cập nhật lại sĩ số của lớp học. (khắc phục được trường hợp nếu việc chèn thông tin sinh viên không thành công mà sĩ số lại được cộng thêm 1, tính toàn vẹn của dữ liệu bị hỏng)

Bài tập về nhà

Cho lược đồ CSDL của một cửa hàng bán linh kiện máy tính như sau:

PhânLoai (MãLoai, TênLoai)

NhàCungCấp (MãNCC, TênNCC, ĐịaChỉ, ĐiệnThoại)

KháchHàng (MãKH, TênKH, ĐịaChỉ, ĐiệnThoại)

HangHoa (MãHH, TênHH, ĐơnViTinh, MãLoai)

BangBaoGia (MãNCC, MãHH, GiáBán)

CungCap (MãNCC, MãHH, Ngày, SốLượng)

HoaDonBan (MãHĐ, MãKH, MãHH, Ngày, SốLượng, ĐơnGiá)

Hãy thực hiện các truy vấn:

1. Liệt kê tất cả các mặt hàng thuộc loại có tên là “Màn Hình”
2. Liệt kê tất cả các nhà cung cấp ở thành phố Hồ Chí Minh
3. Liệt kê những khách hàng có tên bắt đầu bằng chữ “H”
4. Liệt kê đầy đủ thông tin của các nhà cung cấp đã từng cung cấp mặt hàng có tên là “CD LG 52X”
5. Liệt kê mã số và tên của các mặt hàng hiện không có nhà cung cấp nào cung cấp

6. Liệt kê những khách hàng nào mua nhiều hơn một lần trong một ngày (có từ hai hoá đơn trở lên) trong khoảng thời gian từ 01/01/2021 đến 31/10/2021
7. Liệt kê đầy đủ các thông tin của các nhà cung cấp bán mặt hàng có tên là: “CPU PIII – 933EB” với giá rẻ nhất
8. Liệt kê tất cả các khách hàng đã có giao dịch với cửa hàng
9. Liệt kê tất cả các khách hàng thân thuộc trong năm 2021. Biết rằng khách hàng thân thuộc được đánh giá theo mỗi năm và là khách hàng mà có nhiều hơn 10 lần mua trong năm đó
10. Liệt kê tất cả các khách hàng tiềm năng: Là những khách hàng có số tiền mua từ đầu năm đến nay nhiều hơn 50 triệu đồng
11. Liệt kê những nhà cung cấp nào cung cấp hàng cho cửa hàng nhiều nhất trong năm 2021. Biết rằng tiêu chuẩn đánh giá dựa trên tổng giá trị của hàng hoá
12. Liệt kê các mặt hàng bán đắt nhất trong năm 2020

BÀI 12: THỦ TỤC LƯU TRỮ VÀ HÀM DO NGƯỜI DÙNG ĐỊNH NGHĨA

12.1. Mục tiêu

Sau bài thực hành này, người học có khả năng:

- ✓ Viết thủ tục, biên dịch, thực thi các thủ tục người dùng không tham số và có tham số trong SQL Server.
- ✓ Viết được các loại hàm trong SQL Server (hàm trả ra kiểu dữ liệu thông thường và trả ra kiểu bảng)
- ✓ Vận dụng viết hàm, thủ tục tùy theo yêu cầu trong các ứng dụng thực tế.
- ✓ Rèn luyện kỹ năng trong việc lựa chọn các cấu trúc điều khiển, cursor, xử lý lỗi trong thủ tục.

12.2. Tóm tắt lý thuyết

12.2.1. Thủ tục lưu trữ

Stored procedure là một tập các câu lệnh T-SQL mà Sql Server biên dịch thành một kế hoạch thực thi đơn. Lần đầu tiên khi sql server thực thi Stored procedure thì nó biên dịch procedure thành kế hoạch và lưu trong bộ nhớ đệm. Mỗi khi gọi procedure này thì nó sử dụng lại kế hoạch này để thực hiện mà không biên dịch lại nữa.

T- SQL Store procedure tương tự như các ngôn ngữ lập trình khác, chúng chấp nhận các tham số nhập, trả về giá trị xuất thông qua tham số hoặc trả về thông điệp cho biết thủ tục thành công hay thất bại.

Các ứng dụng có thể giao tiếp với SQL Server thông qua hai cách:

- + Chương trình ứng dụng gửi các phát biểu T-SQL từ client đến server. Các phát biểu này được gửi qua mạng và được SQL Server biên dịch lại mỗi khi thực thi chúng.
- + Tạo Store procedure, chúng được lưu và biên dịch thành một kế hoạch ở server. Như vậy với cách này, sử dụng Store procedure sẽ giảm được lưu thông mạng, hiệu quả nhanh hơn so với cách gửi các phát biểu T-SQL.

Một Stored procedure được định nghĩa gồm những thành phần chính sau:

- Tên của stored procedure
- Các tham số
- Thân của stored procedure: bao gồm các lệnh của Transact-SQL dùng để thực thi procedure.

- Cú pháp tạo thủ tục:

```
CREATE PROCEDURE tên_thủ_tục [(danh_sách_tham_số)]  
[WITH RECOMPILE|ENCRYPTION|RECOMPILE, ENCRYPTION]  
AS  
BEGIN  
    khai báo các biến cho xử lý  
    {Các câu lệnh transact-sql}  
END
```

- **Cú pháp sửa thủ tục:** Chỉ việc thay CREATE bằng ALTER và biên dịch lại thủ tục.

- **Cú pháp xóa thủ tục:** DROP PROCEDURE *tên_thủ_tục*

- Cách thực thi thủ tục:

[EXEC] tên_thủ_tục [ds các giá trị tương ứng với tham số]

Trong một phiên làm việc, nếu SP được thực hiện, nó sẽ được lưu trữ vào vùng nhớ đệm. Những lần sau nếu SP được gọi thực hiện lại thì nó sẽ được đọc trực tiếp ra từ vùng nhớ đệm → nâng cao hiệu suất chạy truy vấn

12.2.2. Hàm do người dùng định nghĩa

Một hàm - Function - trong SQL Server được định nghĩa là một thủ tục đơn giản bao gồm một nhóm các câu lệnh T- SQL.

Hàm do người dùng định nghĩa chia làm ba loại:

- Scalar UDF (hàm vô hướng) được sử dụng để trả về duy nhất một giá trị dựa trên một các tham số truyền vào: **Varchar, int,**
- Inline table-valued (hàm nội tuyến) trả về một bảng dựa trên một câu lệnh sql duy nhất định nghĩa các dòng và các cột trả về: **Table**
- Multi-statement table-value: hàm bao gồm nhiều câu lệnh sql bên trong, giá trị trả về dạng bảng: biến kiểu bảng

- Cú pháp tạo hàm:

```
CREATE FUNCTION <tên hàm>
([@<tên tham số> <kiểu DL vô hướng> [= <giá trị>] [, . . .]])
RETURNS <kiểu vô hướng>|<table>
[WITH ENCRYPTION]
AS
    BEGIN
        [<các câu lệnh>]
        RETURN <giá trị vô hướng>| (<câu lệnh SELECT>)
```

- Cú pháp sửa hàm: Chỉ việc thay CREATE bằng ALTER và biên dịch lại hàm.

- Cú pháp xóa hàm: DROP FUNCTION DBO.tên_hàm

12.3. Hướng dẫn thực hành

Yêu cầu về thủ tục:

- Viết thủ tục cho phép chèn thông tin vào bảng Lop bằng cả 2 cách trên.
- Tạo thủ tục sửa thông tin sinh viên trong bảng sinh viên
- Tạo thủ tục tìm kiếm sinh viên có tham số truyền vào.

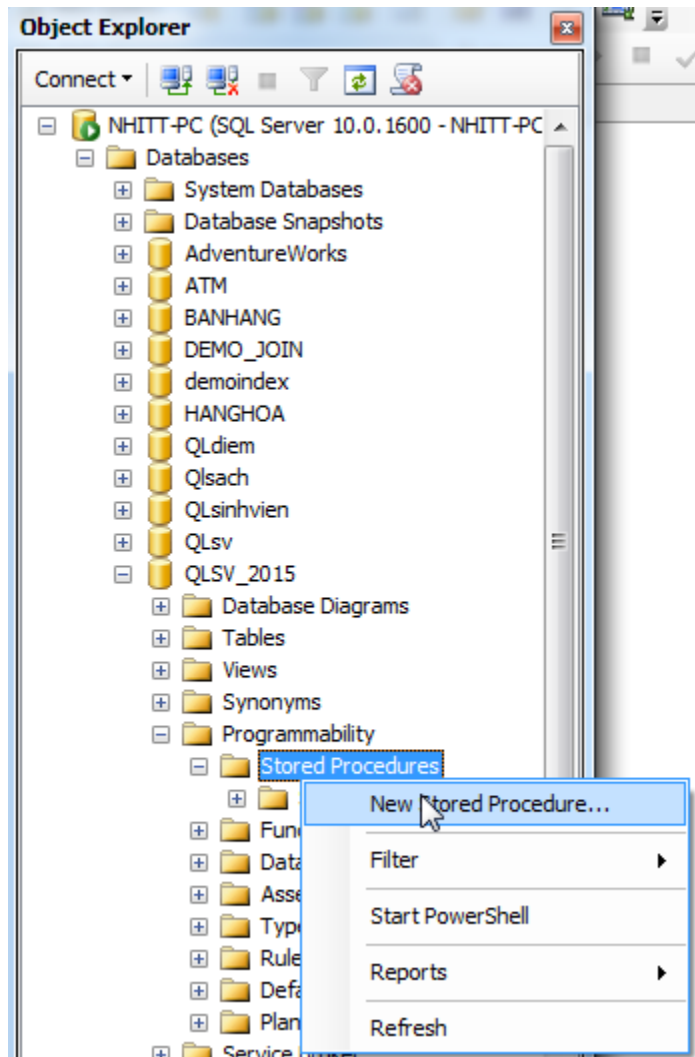
Yêu cầu về hàm:

- Viết hàm có giá trị trả ra kiểu vô hướng, kiểu bảng.
- Viết hàm có tham số và không có tham số.

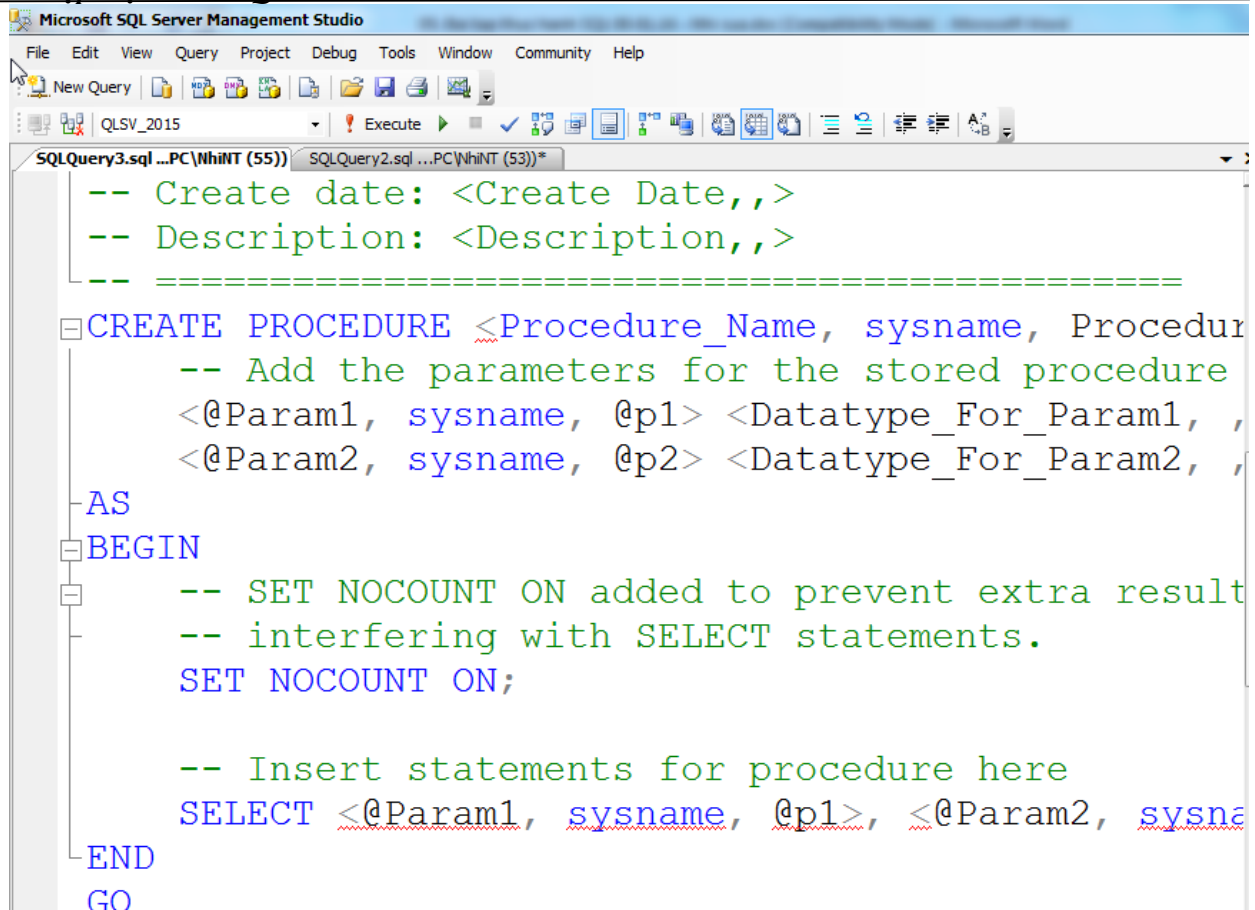
Hướng dẫn tạo thủ tục:

Có thể tiến hành bằng 2 cách: Dùng theo cách sau hoặc mở cửa sổ Query soạn thảo viết tạo thủ tục theo cú pháp.

Bước 1: Khởi động Sql server management studio, sau đó nhấn đúp chuột vào cơ sở dữ liệu QLSV → đến chọn Programmability → stored procedure → click phải chuột vào stored procedure chọn new stored procedure như sau:



Bước 2: Sau khi hộp thoại New stored procedure hiện ra:



Tại cửa sổ truy vấn này, ta chỉ cần sửa lại các tham số của thủ tục cho phù hợp theo yêu cầu và thực hiện viết scrips cho thủ tục theo Template đã có sẵn.

Bước 3: Biên dịch thủ tục (Chọn đoạn scrips viết thủ tục và nhấn nút Excute).
Việc này sẽ tạo ra trong CSDL 1 thủ tục lưu trữ ở thư mục Stored procedure (trong thư mục cha Programmability của CSDL đó)

Bước 4: Thực thi thủ tục.

Bài 1: Mở CSDL QLSV, viết thủ tục lưu tính tổng 10 số nguyên đầu tiên.

Phân tích bài toán:

- Yêu cầu bài không liên quan gì đến các bảng (các đối tượng) của CSDL nên có thể viết thủ tục này trong bất kì CSDL nào.
- Để giải quyết yêu cầu của bài, ta có thể viết thủ tục không tham số hoặc có tham số.
- Trong trường hợp thủ tục viết dưới dạng không tham số, lời gọi không mở rộng được trong trường hợp tính tổng của n số ($n > 10$) => vấn đề này giải quyết bằng cách viết thủ tục có tham số là @n.
- Trong trường hợp thủ tục không tham số ta dùng 2 biến @tong, @i để lưu trữ tổng và biến chạy.
- Sử dụng vòng While để thực hiện tính tổng.
- In kết quả ra màn hình.

Thủ tục minh họa:

+ Cách 1:

```
CREATE PROC SP_tong10
AS
BEGIN
    DECLARE @tong int, @i int
    SET @tong=0
    SET @i=1
    WHILE @i<=10
        BEGIN
            SET @tong=@tong+@i
            SET @i=@i+1
        END
    PRINT N'Tổng của 10 số nguyên đầu tiên là:
'+cast(@tong as char(3))
END
```

Biên dịch và thực thi thủ tục để xem kết quả:

```
EXEC SP_tong10
```

+ Cách 2:

```
CREATE PROC SP_tong10ts(@n int)
AS
BEGIN
    DECLARE @tong int, @i int
    SET @tong=0
    SET @i=1
    WHILE @i<=@n
        BEGIN
            SET @tong=@tong+@i
            SET @i=@i+1
        END
    PRINT N'Tổng của n số nguyên đầu tiên là:
'+cast(@tong as char(3))
END
```

--Lời gọi thủ tục SP_tong10ts có thể thay đổi tham số truyền vào.

```
EXEC SP_tong10ts 12
```

```
EXEC SP_tong10ts 10
```

Bài 2: Mở CSDL QLSV, viết thủ tục hiển thị thông tin về các sinh viên và về các lớp có trong CSDL.

Phân tích bài toán:

- Yêu cầu bài này, ta thấy trong trường hợp này thủ tục không có tham số
- Trong thân thủ tục có 2 câu lệnh truy vấn đơn giản trên 1 bảng.

Thủ tục minh họa:

```
CREATE PROC SP_Select
AS
BEGIN
```

```
SELECT * FROM SINHVIEN
SELECT * FROM LOP
END
GO
```

- Biên dịch và thực thi thủ tục

EXEC SP_Select

Bài 3: Mở CSDL QLSV, viết thủ tục thêm thông tin sinh viên vào bảng sinh viên. Biết rằng, thông tin của sinh viên cần nhập được nhận từ các giá trị thông qua các tham số.

Phân tích bài toán:

- Yêu cầu bài này, ta thấy trong trường hợp này thủ tục phải có tham số. Số lượng tham số đầu vào không trả ra giá trị, có số lượng tham số bằng với số các cột được định nghĩa trong bảng sinhvien và kiểu dữ liệu tương ứng giống với các cột của bảng sinhvien.
- Thủ tục có 6 tham số (vì bảng sinh viên có 6 cột)
- Trong thân thủ tục:
 - + Kiểm tra sự tồn tại của mã sinh viên trong bảng sinh viên, nếu sinh viên này chưa tồn tại thì ok.
 - + Kiểm tra mã lớp đã tồn tại thì ok
 - + Dùng câu lệnh INSERT để thực hiện việc nhập dữ liệu vào bảng sinhvien với thông tin của sinh viên được nhận thông qua các tham số.

Thủ tục minh họa:

```
CREATE PROC SP_ThemSinhVien
    @mssv      nvarchar(10),
    @hoTen     nvarchar(30),
    @ngaysinh  smalldatetime,
    @gioitinh  bit,
    @noisinh   nvarchar(30),
    @maLop     nvarchar(10)

AS
BEGIN
    IF (EXISTS (SELECT * FROM SinhVien s WHERE s.masv = @mssv))
    BEGIN
        PRINT N'Mã số sinh viên ' + @mssv + N' đã tồn tại'
        RETURN -1
    END
    IF (NOT EXISTS (SELECT * FROM Lop L WHERE L.malop = @maLop))
    BEGIN
        PRINT N'Mã số lớp ' + @maLop + N' chưa tồn tại'
        RETURN -1
    END

    INSERT INTO SinhVien(masv, hoten, ngaySinh,
        gioitinh, noisinh, maLop)
    VALUES (@mssv, @hoTen, @ngaysinh, @gioitinh, @noisinh,
        @maLop)
```

```
RETURN 0 /* procedure tự trả về 0 nếu không RETURN */  
END
```

GO

--Thực thi thủ tục

```
EXEC SP_ThemSinhVien 'sv10', N'Nguyễn Văn A', '1997-09-  
08', 0, N'Thái Bình', 'SEK18.6'
```

Bài 4: Mở CSDL QLSV, viết thủ tục tính tổng 2 số.

Phân tích bài toán:

- Theo yêu cầu của bài ta phải khai báo 2 tham số đại diện cho 2 số input và một tham số OUTPUT để lấy ra giá trị của tổng 2 số.
- Trong thân thủ tục có một câu lệnh gán giá trị của tham số OUTPUT
- Lời gọi thủ tục, ta phải khai báo biến để lưu kết quả trả ra của tham số OUTPUT.

Thủ tục minh họa:

```
CREATE PROCEDURE sp_Conghaiso (  
    @a INT,  
    @b INT,  
    @c INT OUTPUT)
```

AS

```
    SELECT @c=@a+@b
```

Thực hiện lời gọi thủ tục trong một tập các câu lệnh như sau:

```
DECLARE @tong INT
```

```
SELECT @tong=0
```

```
EXECUTE sp_Conghaiso 100, 180, @tong OUTPUT
```

```
SELECT @tong
```

=> câu lệnh “SELECT @tong” sẽ cho kết quả là: 300

✓ Để xóa thủ tục lưu trữ:

DROP PROC sp_Conghaiso

Bài 5: Mở CSDL QLSV, viết thủ tục để tính số lượng sinh viên của mỗi lớp (mặc định là lớp có mã ‘SEK21.2’)

Phân tích bài toán:

- Theo yêu cầu của bài ta phải khai báo 1 tham số đại diện cho tham số input @TSMa_lop và có giá trị mặc định là ‘SEK21.2’ và một tham số OUTPUT để lấy ra giá trị là tổng số lượng sinh viên của biến mã lớp truyền vào.
- Trong thân thủ tục có một câu lệnh truy vấn tính số lượng sinh viên của mã lớp được truyền vào.

- Lời gọi thủ tục, ta phải khai báo biến để lưu kết quả trả ra của tham số OUTPUT.

Thủ tục minh họa:

```
CREATE PROC Soluong_Phong_Default
    @TSMalop nvarchar (10)='SEK21.2',
    @SLSV int OUTPUT
AS
    SELECT @SLSV = COUNT (MASV)
    FROM lop LEFT JOIN sinhvien
    ON lop.malop =sinhvien.malop
    WHERE LOP.malop=@TSMalop
GO
```

- Gọi thủ tục:

```
DECLARE @dem INT;
SELECT @dem =0
--trường hợp nhận giá trị của tham số mặc định:
EXEC Soluong_Phong_Default @SLSV= @dem OUTPUT
--Lời gọi với một giá trị khác của tham số:
EXEC Soluong_Phong_Default 'SEK18.1', @dem OUTPUT
SELECT @dem
```

Bài 6: Viết thủ tục lấy ra danh sách sinh viên theo mã lớp.

Phân tích bài toán:

- Thủ tục này có 1 tham số là mã lớp.
- Trong thân thủ tục: Kiểm tra xem mã lớp cần lấy dssv có tồn tại không trong bảng sinhvien không, nếu tồn tại thì hiện thị dssv với mã lớp tương ứng.

Thủ tục minh họa:

```
CREATE PROC spDanhSachSinhVien
    @maLop varchar(10)
    WITH ENCRYPTION
AS
BEGIN
    IF (NOT EXISTS (SELECT * FROM SINHVIENT S WHERE
    S.malop = @maLop))
    BEGIN
        PRINT N'Mã số lớp ' + @maLop + N' chưa tồn
        tại'
        RETURN -1
    END

    SELECT * FROM sinhvien s WHERE S.malop = @maLop
    /*procedure luôn trả về 0 nếu không RETURN*/
END
GO
```

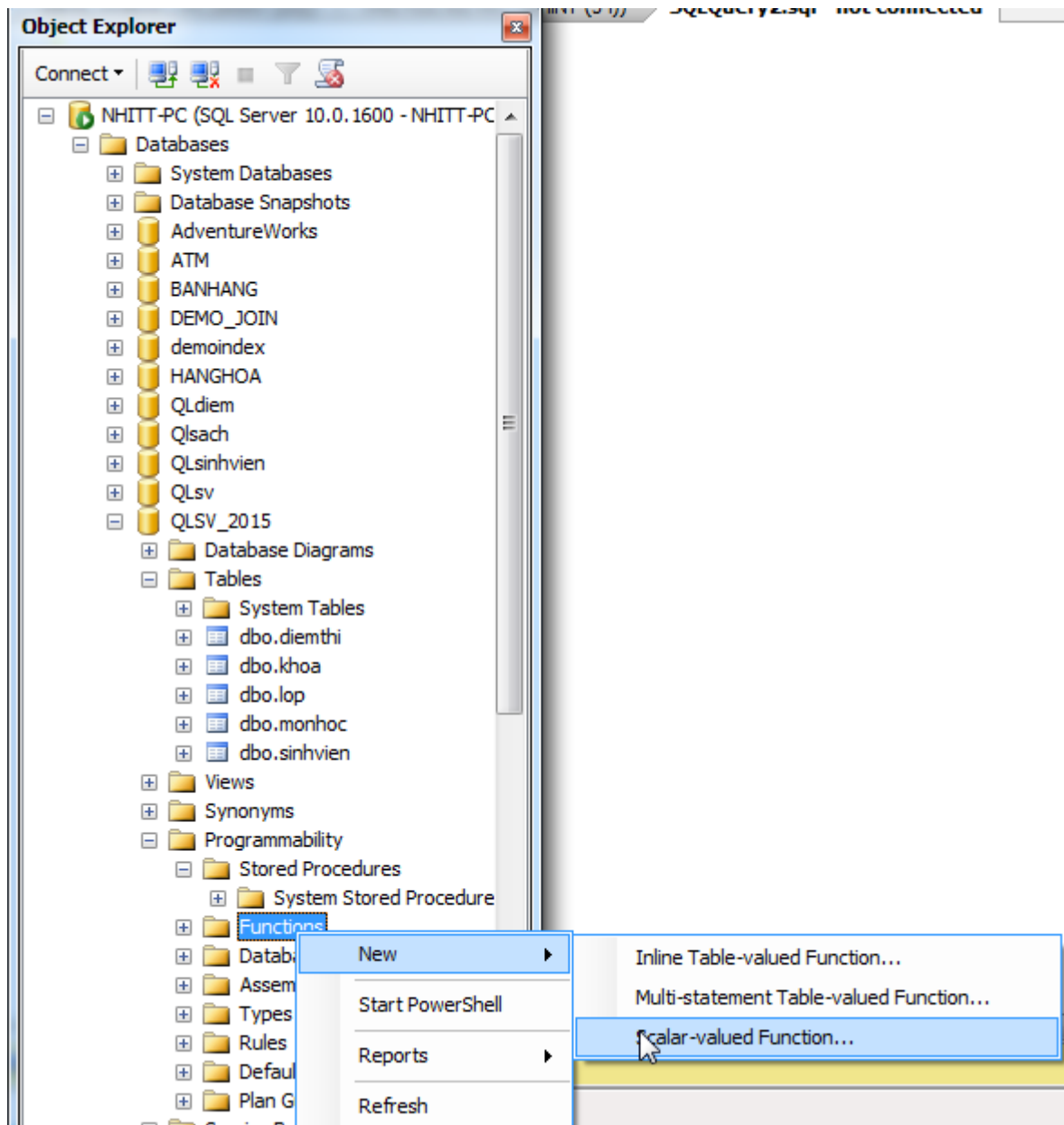

- Sử dụng tùy chọn bảo mật trong thủ tục `sp_DanhSachSinhVien` (Không xem scripts của thủ tục được bằng thủ tục lưu trữ của hệ thống có tên `SP_helptext`)
Hàm:

Hướng dẫn tạo hàm:

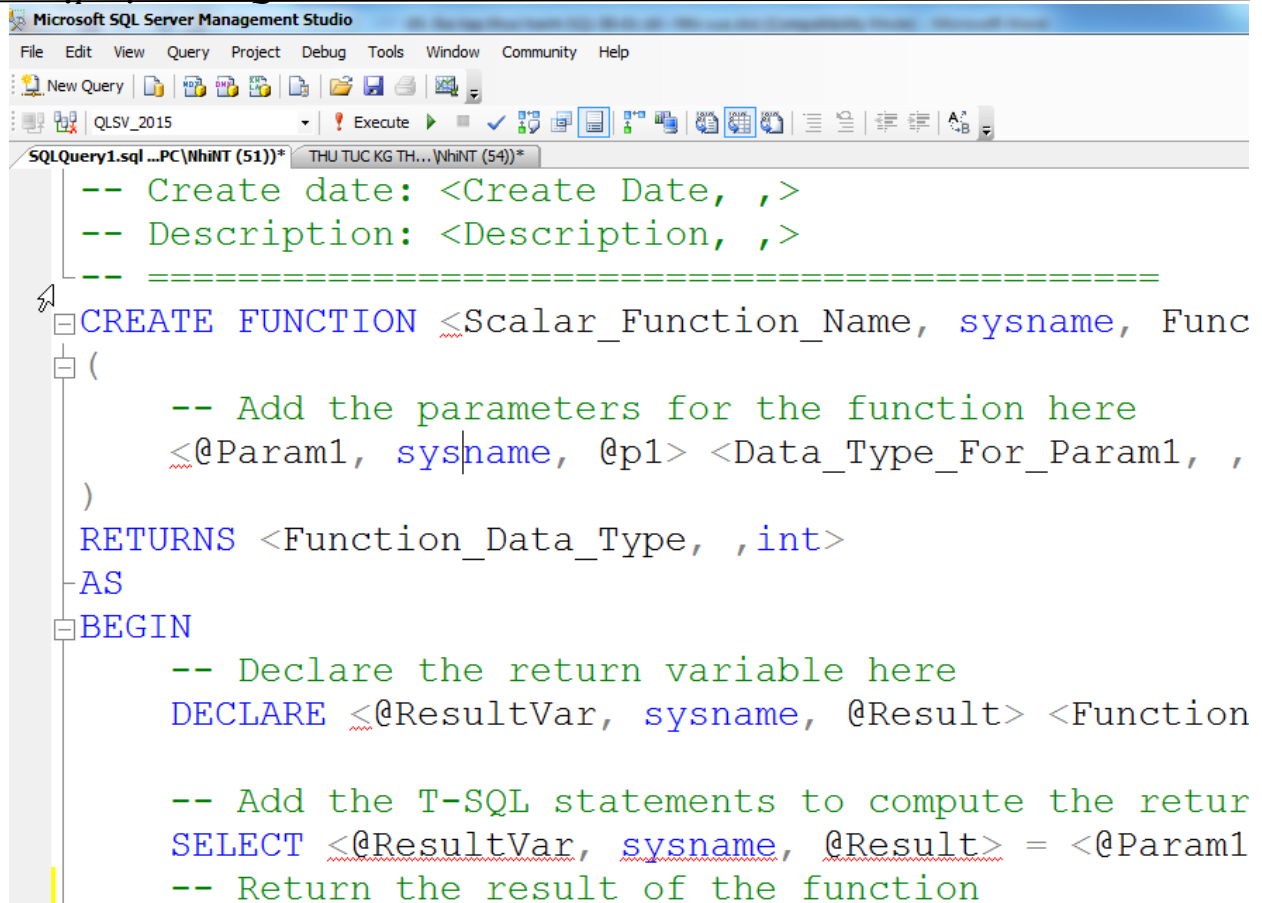
- Để viết hàm trong một cơ sở dữ liệu QLSV ta có 2 cách: Dùng công cụ hoặc viết theo cú pháp tạo hàm trên.
- Để viết hàm bằng công cụ, thực hiện như sau:

Bước 1: Kết nối tới SQL server, sau khi kết nối thành công ta chuyển sang bước 2

Bước 2: kích đúp chuột vào CSDL QLSV, chọn Programmability → Fuction → New fuction → Scalar-value fuction để viết hàm trả về kiểu dữ liệu vô hướng.



Sẽ hiện lên cửa sổ sau:



Bước 3: Sử dụng Template có sẵn sửa chữa các tham số và viết hàm theo yêu cầu.

Bước 4: Biên dịch thủ tục (Chọn đoạn scrips viết hàm và nhấn phím F5). Việc này sẽ tạo ra trong CSDL 1 thủ tục hàm ở thư mục Functions (trong thư mục chương trìnhability của CSDL đó)

Bước 4: Sử dụng hàm:

+ **Hàm trả ra kiểu vô hướng:**

SELECT dbo.tên_hàm (ds các giá trị truyền vào cho hàm theo tham số)

+ **Hàm trả ra kiểu bảng:**

SELECT *

FROM tên_hàm (ds các giá trị truyền vào cho hàm theo tham số)

Bài 7: Viết hàm tính số lượng sinh viên của một lớp. Biết rằng, hàm có tham số truyền vào là mã lớp.

Phân tích bài toán:

- Theo yêu cầu của bài, hàm cần viết có giá trị trả ra là kiểu vô hướng (số lượng sinh viên).
- Hàm có một tham số truyền vào là @tsmalop của lớp cần tính số lượng sinh viên. Tham số này có kiểu dữ liệu giống với kiểu dữ liệu của cột malop trong bang sinhvien.

- Trong thân hàm: Tính số lượng sinh viên của lớp có mã đúng bằng giá trị của tham số mã lớp truyền vào cho hàm và trả ra kết quả của hàm là số lượng sinh viên của lớp có mã đó.

Hàm minh họa:

```
CREATE FUNCTION Fn_Soluong_SV ( @tsMaLOP NVARCHAR (10)
)
RETURNS int
AS
BEGIN
    DECLARE @SLsv int;
    SELECT @SLsv = COUNT (masv)
    FROM SinhVien
    WHERE Malop=@tsMalop
    RETURN (@SLsv)
END
--Su dung ham (Ham xuat hien trong bieu thuc)
SELECT dbo.Fn_Soluong_sv('SEK21.4');
```

Hoặc sử dụng trong một biểu thức truy vấn:

```
SELECT Malop, count (masv) as slsv
FROM sinhvien
GROUP BY malop
HAVING count(masv) > dbo.Fn_Soluong_sv('SEK21.4')
```

Bài 8: Viết hàm có tham số truyền vào là mã lớp và trả ra thông tin là danh sách sinh viên thuộc lớp đó. Biết rằng, thông tin hiển thị gồm: MaSv, hoten, Ngaysinh.

Phân tích bài toán:

- Theo yêu cầu bài, vì hàm trả ra một danh sách sinh viên nên kiểu trả về của hàm phải là kiểu bảng.
- Hàm có một tham số truyền vào là @tsmalop của lớp có danh sách sinh viên được hiển thị. Tham số này có kiểu dữ liệu giống với kiểu dữ liệu của cột malop trong bang sinhvien.
- Trong thân hàm: Có duy nhất một câu lệnh RETURN trả ra kết quả của hàm là một danh sinh viên của lớp có mã lớp đúng bằng giá trị của tham số truyền vào.

Hàm minh họa:

```
IF object_id(' Fn_DSsv_lop ','FN') IS NOT NULL
    DROP FUNCTION Fn_DSsv_lop
GO
CREATE FUNCTION Fn_DSsv_lop (@tsMaLOP NVARCHAR (10))
RETURNS TABLE
AS

RETURN
(SELECT Masv, HoTen, NgaySinh
```

```
FROM sinhVien
WHERE Malop=@tsMalop )

GO

-- Su dung ham tra ket qua bang nhu la TABLE
SELECT *
FROM Fn_DSsv_lop('SEK21.3')
```

Bài 9: Viết hàm thông kê số lượng sinh viên của mỗi lớp của một khóa. Biết rằng, hàm trả ra các thông tin là danh sách **các lớp** của khóa tương ứng gồm: Malop, tenlop, slsv (đây là cột phải tính toán).

Phân tích bài toán:

- Theo yêu cầu bài, vì hàm trả ra **danh sách các lớp** nên kiểu trả về của hàm phải là kiểu biến bảng @bangtk.
- Hàm có một tham số truyền vào là @khoa của lớp có danh sách sinh viên được hiển thị. Tham số này có kiểu dữ liệu giống với kiểu dữ liệu của cột khoa trong bang lop.
- Trong thân hàm: Nếu không tồn tại khóa thì hiển thị các thông tin về Malop, tenlop, slsv của tất cả các lớp trong bảng lop. Trong trường hợp ngược lại (khóa tồn tại) thì hiển thị thông tin của các lớp trong bảng lop thuộc khóa đó.

Hàm minh họa:

```
CREATE FUNCTION Fn_thongkeSV_LOP (@khoa smallint)
RETURNS @bangtk TABLE
(
    Malop NVarChar(10),
    tenlop NVarChar(30),
    slsv tinyint)
BEGIN
    IF @KHOA =0
        INSERT INTO @bangtk
        SELECT LOP.MALOP, tenlop, COUNT(MASV)
        FROM LOP INNER JOIN sinhVien
        ON LOP.MALOP=SINHVIEN.MALOP
        GROUP BY LOP.MALOP, tenlop
    ELSE
        INSERT INTO @bangtk
        SELECT LOP.MALOP, tenlop, COUNT(MASV)
        FROM LOP INNER JOIN sinhVien
        ON LOP.MALOP=SINHVIEN.MALOP
        WHERE khoa=@khoa
        GROUP BY LOP.MALOP, tenlop

    RETURN
END
```

Lời gọi hàm:

```
SELECT * FROM Fn_thongkeSV_LOP (12)--tồn tại khóa học
```

```
SELECT * FROM Fn_thongkeSV_LOP (0) -- không tồn tại khóa học
```

12.4. Bài tập tự làm

Mở CSDL QLSV trong bài thực hành số 9.

Viết thủ tục thực hiện các công việc sau:

1. Tính số lượng lớp của khoa có mã là 'CNTT'
2. Tính số lượng sinh viên, số lượng lớp theo mã khoa.
3. Tính điểm trung bình lần 1 của học kì 1 của một sinh viên bất kì.
4. Hiển thị thông tin của các sinh viên thuộc một khoa và một khóa bất kì.
5. Hiển thị thông tin về mã sinh viên, họ tên của các sinh viên thuộc khoa có mã khoa là "CNTT" có nhiều hơn 3 môn có điểm dưới trung bình.
6. Hiển thị thông tin của các sinh viên thuộc Khoa nào và Khóa học nào phải học nhiều môn nhất trong toàn trường (sinh viên đó đã có điểm môn học tương ứng).
7. Hiển thị thông tin của các sinh viên của một lớp bất kì đã học nhiều môn nhất. (sinh viên đó đã có điểm lần 1 của các môn học đó).
8. Hiển thị thông tin những môn học thuộc Khoa "CNTT", khoá K21 hiện nay chưa có sinh viên nào học (sinh viên chưa có điểm).
9. Nhập thông tin vào bảng diemthi thỏa mãn điều kiện mã sinh viên và mã môn học phải tồn tại trong bảng sinh viên và môn học.
10. Xóa thông tin của sinh viên có mã bất kì.
11. Cập nhật thông tin về địa điểm của khoa có mã 'CNTT'.
12. Viết thủ tục dùng con trỏ để tạo ra số báo danh tự động của một danh sách thi từ dữ liệu bảng sinh viên gồm các thông tin: Số báo danh, mã sinh viên, họ tên, ngày sinh. Số báo danh tự động dựa vào mã sinh viên lần lượt là 'SBD1', 'SBD2',...
13. Viết thủ tục có tác dụng nhập dữ liệu vào bảng môn học. Biết rằng, chỉ cho phép nhập các môn học có mã chưa tồn tại và giá trị của cột mã môn học phải thỏa mãn: 2 kí đầu là 'MH', còn các kí tự sau là số, ví dụ: MH000001, MH000002,...

Viết hàm thực hiện các công việc sau:

1. Tính giai thừa của một số nguyên bất kì.
2. Tính số ngày của một tháng trong một năm bất kì.
3. Tính số lượng sinh viên của một lớp thuộc một khoa bất kì.
4. Hiển thị thông tin của các sinh viên thuộc khoa có mã 'CNTT' và lớp có mã là 'TK10.4'. Biết rằng, thông tin hiển thị gồm: Mã sinh viên, họ tên, tuổi của sinh viên.
5. Viết hàm tạo ra mã sinh viên tự động tiếp theo của một mã sinh viên đã biết. Ví dụ, mã hiện tại SV001 thì mã tiếp theo sẽ là SV002.
6. Viết hàm trả ra danh sách sinh viên của lớp có mã tùy ý.

BÀI 13: TRIGGER

13.1. Mục tiêu

Sau bài thực hành này, người học có khả năng:

- ✓ Phân biệt được cấu trúc 2 bảng logic dùng trong các kiểu bẫy lỗi (bảng **Inserted** và bảng **Deleted**) và tác dụng của việc sử dụng bẫy lỗi.
- ✓ Viết được bẫy lỗi Insert, Delete, Update và một số các bẫy lỗi khác: Instead Of, dây truyền.
- ✓ Liệt kê các bẫy lỗi được gắn với một bảng, xem thông tin về các bẫy lỗi.
- ✓ Vận dụng tạo ra các bẫy lỗi trong các ứng dụng thực tế.

13.2. Tóm tắt lý thuyết

- ✓ Bẫy lỗi là một thủ tục lưu đặc biệt, Triger tự động được chạy mỗi khi có một hành động liên quan đến nó xảy ra
- ✓ Triger không có tham số, không trả ra giá trị như SP bình thường.
- ✓ Các kiểu trigger cơ bản: Insert trigger, Delete trigger, Update trigger, Trigger Instead of, trigger tổng hợp.
- ✓ Khi gắn bẫy lỗi Instead Of vào một bảng, nó sẽ bỏ qua các thao tác mặc định của câu lệnh bị bẫy lỗi (Insert, Update, Delete). Điều đó có nghĩa là đoạn mã của bẫy lỗi Instead Of được thực thi thay vì thực thi các câu lệnh bẫy lỗi.

Nói cách khác, bất cứ khi nào các lệnh INSERT, UPDATE, DELETE được sử dụng trên một đối tượng (TABLE, VIEW) thì bẫy lỗi tương ứng với câu lệnh đó sẽ được thực thi.

- Cú pháp tạo Trigger:

```
CREATE TRIGGER <tên_trigger> ON <tên 1 bảng>|<tên 1 view>
[WITH ENCRYPTION]
{
{FOR | AFTER| INSTEAD OF}
<INSERT [, UPDATE] [, DELETE]>
}
AS
[IF UPDATE (column) | [[AND | OR] UPDATE (column)]
<câu lệnh SQL>
```

- **Cú pháp sửa Trigger:** Chỉ việc thay CREATE bằng ALTER.
- **Cú pháp xóa Trigger:** DROP TRIGGER tên_trigger
- **Xem lại định nghĩa của trigger:** sp_helptext
- **Xem danh sách các trigger trên 1 bảng:** sp_helptrigger
- **Chú ý:**

- ✓ Một bảng có nhiều trigger
- ✓ Mỗi một trigger có tên duy nhất
- ✓ Trong trigger thường dùng mệnh đề IF EXISTS

- Vai trò của 2 bảng logic Inserted và Deleted trong các kiểu bắt lỗi:

Kiểu bắt lỗi	Bảng <i>Inserted</i>	Bảng <i>Deleted</i>
UPDATE	Chứa một bản sao của các bản ghi đã được cập nhật sau khi câu lệnh được thực thi xong.	Chứa một bản sao nguyên bản của các bản ghi đã được cập nhật sau khi câu lệnh được thực thi xong..
DELETE	Không sử dụng.	Chứa các bản ghi đã được xóa sau khi câu lệnh thực thi xong.
INSERT	Chứa một bản sao của các bản ghi đã được chèn thêm sau khi câu lệnh được thực thi xong	Không sử dụng.

- ✓ **Để cấm tất cả trigger của 1 bảng:**

ALTER TABLE *table_name* DISABLE TRIGGER ALL | *trigger_name*

- ✓ **Để cho phép tất cả các trigger của 1 bảng:**

ALTER TABLE *table_name* ENABLE TRIGGER ALL | *trigger_name*

- ✓ Trong trường hợp bảng tồn tại các ràng buộc thiết lập giống với ràng buộc trong trigger thì tạm thời có thể vô hiệu hóa các ràng buộc để kiểm tra sự hoạt động của trigger được chính xác:

ALTER TABLE *tên_bảng* NOCHECK CONSTRAINT ALL

- ✓ Và ta có thể kích hoạt lại việc kiểm tra lại các ràng:

ALTER TABLE *tên_bảng* CHECK CONSTRAINT ALL

13.3. Hướng dẫn thực hành

Yêu cầu:

- Tạo bắt lỗi Insert, Update, Delete trên 1 bảng nào đó trong CSDL QLSV.
- Sử dụng các tùy chọn trong bắt lỗi.
- Liệt kê các bắt lỗi trong 1 bảng, xem thông tin các bắt lỗi.

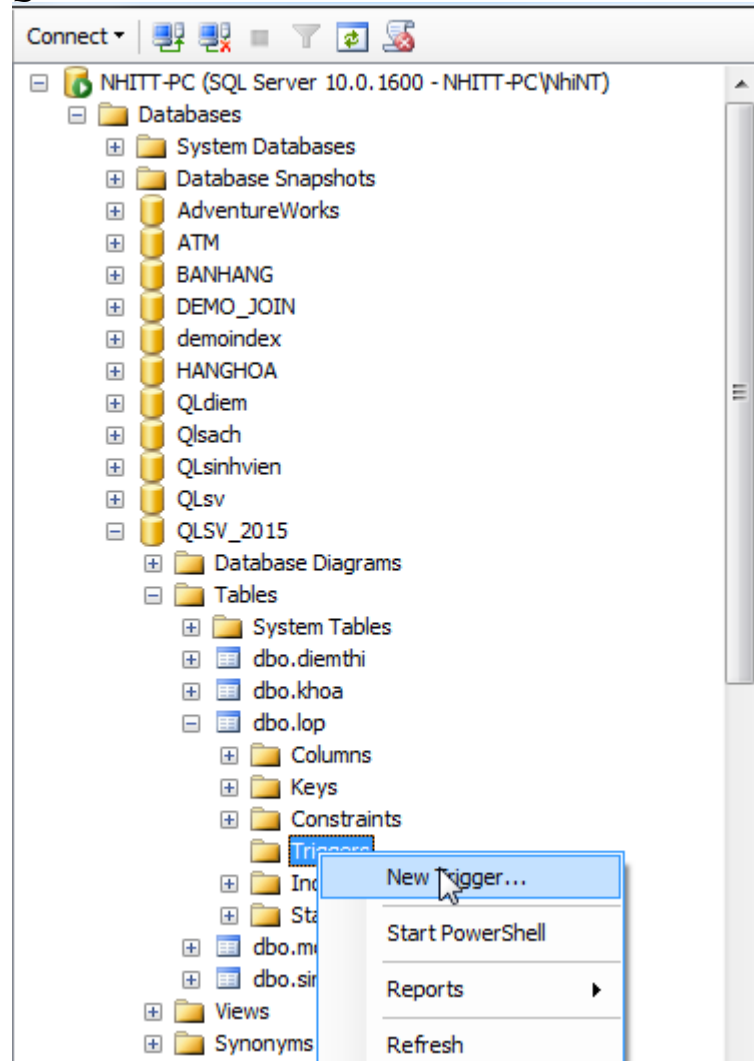
Hướng dẫn tạo bắt lỗi:

Có thể tiến hành bằng 2 cách: Dùng theo cách sau hoặc mở cửa sổ Query soạn thảo viết Trigger theo cú pháp đã định nghĩa.

Muốn tạo ra một Trigger trên bảng LOP của CSDL QLSV, ta làm như sau:

Bước 1:

Kết nối đến SQL Server, chọn CSDL QLSV, mở thư mục Tables, mở thư mục LOP, click chuột phải vào thư mục Triggers, chọn **New trigger**:



Xuất hiện cửa sổ sau:

A screenshot of a SQL query editor window. The title bar shows the file name 'SQLQuery6.sql ...PC\NhINT (55)*'. The editor contains a SQL script for creating a trigger. The script starts with a comment '-- Description: <Description,,>' followed by a separator line of dashes. The main script is: 'CREATE TRIGGER <Schema_Name, sysname, Schema_Name> ON <Schema_Name, sysname, Schema_Name>.<Table_Name, sysname, Schema_Name> AFTER <Data_Modification_Statements, , INSERT, UPDATE, DELETE> AS BEGIN -- SET NOCOUNT ON added to prevent extra : -- interfering with SELECT statements. SET NOCOUNT ON; -- Insert statements for trigger here END GO'. The text is color-coded: green for comments, blue for SQL keywords, and magenta for identifiers. The 'GO' statement is at the bottom of the script.

Bước 2:

Thực hiện việc sửa và viết trigger theo Template đã có sẵn.

Bước 3:

Kích hoạt trigger (bằng cách thực thi câu lệnh INSERT, UPDATE, DELETE ứng với Trigger kiểu INSERT, kiểu UPDATE, kiểu DELETE)

Bài 1: Viết trigger trên bảng sinhvien sao cho khi thêm 1 sinh viên vào một lớp thì số sinh viên của lớp tương ứng sẽ tăng lên 1.

Phân tích bài toán:

- Theo yêu cầu bài, trigger được viết trên bảng SINHVIEN.
- Đây là trigger kiểu Insert.
- Trong thân trigger có câu lệnh cập nhật số sinh viên trên bảng lớp sao cho mã lớp của lớp này bằng đúng với mã lớp của sinh viên được bổ sung (malop trong bảng INSERTED).

Trigger minh họa:

```
CREATE TRIGGER trInsSV_LOP
ON SINHVIEN
FOR INSERT
AS
UPDATE LOP
SET SISO = SISO+1
WHERE LOP.MALOP = (SELECT MALOP FROM INSERTED)
```

Kích hoạt trigger:

Thực thi câu lệnh bổ sung 1 sinh viên vào bảng để xem sự kích hoạt của trigger:
INSERT INTO SINHVIEN (MASV, hoten, ngaysinh, gioitinh, Noisinh, Malop)
VALUES ('Sv12', N'Nguyễn Thị Hoa', '1798-12-08', 0, N'Hưng yên', 'SEK21.3')

Bài 2: Viết Trigger trên bảng lớp sao cho khi xóa thông tin của lớp thì thông tin liên quan cũng bị xóa theo.

Phân tích bài toán:

- Theo yêu cầu bài, trigger được viết trên bảng LOP.
- Đây là trigger kiểu DELETE dạng dây chuyền.
- Trong thân trigger sử dụng cấu trúc IF để kiểm tra có lớp cần xóa không, nếu không tồn tại lớp đó thì hủy hành động xóa. Ngược lại, sẽ xóa thông tin bảng liên quan là bảng SINHVIEN.

Trigger minh họa:

```
CREATE TRIGGER TG_DELETELOP ON LOP
WITH ENCRYPTION
FOR DELETE
AS
BEGIN
IF(@@ROWCOUNT <=0)
```

```
PRINT 'Không có lớp nào trong bảng!'
ROLLBACK TRANSACTION

ELSE
BEGIN
    DELETE FROM SINHVIEN AS S, DELETED AS D
    WHERE S.MALOP=D.MALOP
END
END
```

- Thực thi câu lệnh Delete để kích hoạt trigger xem trigger hoạt động có chính xác không.

```
DELETE
FROM LOP
WHERE Malop='SEK21.4'
```

- Sử dụng tùy chọn bảo mật trong trigger TG_DELETELOP (Không xem scrips của trigger được bằng thủ tục lưu trữ của hệ thống có tên SP_helptext)

Bài 3: Viết trigger trên bảng môn học sao cho không cho phép sửa dữ liệu tại cột Hoten của sinh viên.

Phân tích bài toán:

- Theo yêu cầu bài, trigger được viết trên bảng SINHVIEN.
- Đây là trigger kiểu UPDATE và UPDATE mức cột là cột Hoten.
- Trong thân trigger sử dụng cấu trúc IF để kiểm tra nếu cập nhật dữ liệu trên cột Hoten thì hủy hành động Update.

Trigger minh họa:

```
CREATE TRIGGER Tg_NoUpdateHT
ON Sinhvien
FOR UPDATE AS
IF UPDATE (Hoten)
BEGIN
    PRINT 'Không được sửa dữ liệu trên cột Hoten'
    ROLLBACK TRANSACTION
END
```

Thực thi câu lệnh Update trên cột Hoten để kích hoạt trigger xem trigger hoạt động có chính xác không.

```
UPDATE Sinhvien
SET Hoten='Trần Thị Lý'
WHERE Masv='SV01'
```

Bài 4: Viết trigger trên bảng lớp sao cho chỉ xóa được những lớp chưa có sinh viên, ngược lại sẽ thông báo lỗi.

Phân tích bài toán:

- Theo yêu cầu bài, trigger được viết trên bảng LOP.
- Đây là trigger kiểu INSTEAD OF DELETE, loại trigger nếu ta chỉ viết trigger cho sự kiện DELETE đơn thuần thì trigger kích hoạt sẽ không chính xác.
- Trong thân trigger sử dụng cấu trúc IF để kiểm tra nếu tồn tại sinh viên của lớp cần xóa thì hủy hành động xóa. Ngược lại sẽ chứa 1 câu lệnh xóa thông tin về lớp cần xóa. (Điều này khác với trigger thông thường).

Trigger minh họa:

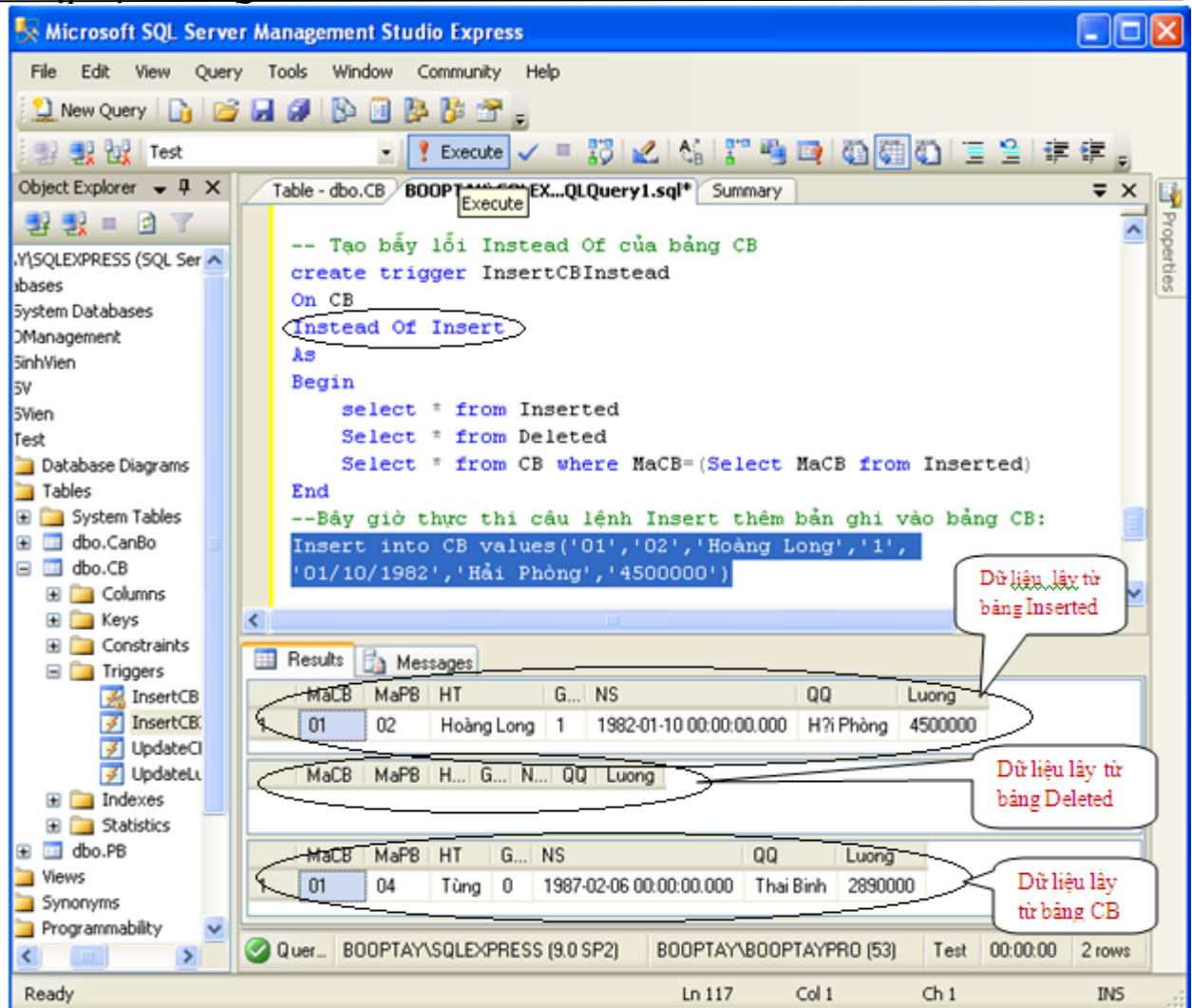
```
CREATE TRIGGER Tg_DeleteLopNoSV ON lop
INSTEAD OF DELETE
AS
BEGIN
    If (Select count(*) from sinhvien sv, Deleted d where
    sv.malop=d.malop)>0
    BEGIN
        RAISERROR ('không xóa được', 14, 1)
        ROLLBACK TRAN
    END
    ELSE
    BEGIN
        PRINT 'Xóa thành công'
        DELETE FROM lop WHERE malop IN(SELECT malop FROM
        DELETED)
    END
END
GO
```

- Thực thi câu lệnh Delete 1 lớp để kích hoạt trigger xem trigger hoạt động có chính xác không.

```
DELETE
FROM LOP
WHERE Malop='SEK21.2'
```

- Ta thử sửa trigger này thành trigger kiểu Delete thông thường để kiểm tra sự kích hoạt của trigger.

Viết và thực thi ví dụ sau để hiểu rõ hơn về trigger dạng instead of:
Tạo bẫy lỗi Instead Of của bảng CB cho sự kiện Insert.



Quan sát kết quả nhận được ở dưới bảng kết quả, ta thấy rằng, **bản ghi mới cần thêm không được thêm vào bảng CB như bình thường**, mà thay vào đó là việc thực thi các câu lệnh của bẫy lỗi Instead Of để lấy dữ liệu từ các bảng trung gian và bảng chính thỏa mãn điều kiện của câu truy vấn. Hay nói cách khác, **các câu lệnh trong trigger được thực thi mà không quan tâm đến sự kiện bẫy lỗi của trigger**.

13.4. Bài tập tự làm

Mở CSDL QLSV, viết các trigger thực hiện các yêu cầu sau:

1. Thỏa mãn ràng buộc chỉ cho phép nhập dữ liệu vào bảng sinh viên nếu sinh viên có mã lớp đã tồn tại trong bảng lớp và giới tính của sinh viên phải có giá trị là 0 hoặc là 1.
2. Thỏa mãn không cho phép nhập trùng dữ liệu tại cột mã lớp trong bảng lop.
3. Thỏa mãn một lớp có tối đa 10 sinh viên (dùng masv để kiểm tra số lượng sinh viên của từng lớp trên bảng sinhvien).
4. Thỏa mãn 1 sinh viên trong 1 kỳ học không học quá 6 môn học (giả sử các sinh viên phải có điểm của môn học đó).
5. Thỏa mãn khi xóa dữ liệu của bảng khoa thì thông tin liên quan cũng bị xóa theo.

6. Thỏa mãn 1 sinh viên học môn có tên ‘Cơ sở dữ liệu’ thì sinh viên đó phải là sinh viên khoa công nghệ thông tin (mã khoa là ‘CNTT’). Giả sử rằng, sinh viên học môn đó phải có điểm môn học đó và mã của môn học đó là ‘CSDL’.
7. Thỏa mãn khi thay đổi mã lớp thì thông tin liên quan cũng bị thay đổi theo.
8. Thỏa mãn nếu mã sinh viên chưa tồn tại trong bảng điểm thi thì nhập điểm cho sinh viên ứng với môn học đó (nếu diemlan1>5 thì diemlan2=NULL). Nếu mã sinh viên có trong bảng điểm thi thì kiểm tra nếu diemlan1<5 thì tiến hành cập nhật điểm cho cột diemlan2 ứng với sinh viên thi môn học đó (ngược lại diemlan2=NULL). (Hướng dẫn trigger tổng hợp cho sự kiện Insert và Update)
9. Thỏa mãn khi nhập thông tin vào bảng môn học thì mã môn học có 2 ký tự đầu bắt buộc là ‘MH’.
10. Thỏa mãn khi nhập thông tin vào bảng môn học thì mã môn học có 2 ký tự đầu bắt buộc là ‘MH’ và kiểm tra sự trùng mã.

Bài tập về nhà

Cho CSDL QLDTKH với các bảng sau:

SinhVien (MaSV, HoTen, DiaChi, Lop)

DeTai (MaDT, TENDT)

SinhVien_DeTai (MaSV, MaDT)

HocVi (MaHV, TenHV)

GiaoVien (MaGV, HoTen, DiaChi, MaHV)

GiaoVien_DeTai (MaGV, MaDT)

KetQua (MaSV, MaDT, Diem)

Ý nghĩa của các bảng:

SinhVien: Lưu danh sách các sinh viên

DeTai: Lưu danh sách các đề tài

KetQua: Lưu kết quả bảo vệ của các đề tài

GiaoVien: Lưu thông tin các giáo viên

SinhVien_DeTai: Lưu các đề tài mà sinh viên thực hiện

GiaoVien_DeTai: Lưu các đề tài mà giáo viên hướng dẫn

HocVi: Lưu danh mục các học vị.

Hãy tạo các bảng trên với các ràng buộc theo các quan hệ trong bảng sau:

Bảng Cha	Bảng Con	Kiểu quan hệ
SinhVien	SinhVien_DeTai	1-n
DeTai	SinhVien_DeTai	1-n
GiaoVien	GiaoVien_DeTai	1-n
SinhVien_DeTai	KetQua	1-1
DeTai	GiaoVien_DeTai	1-n
HocVi	GiaoVien	1-n

Nhập dữ liệu vào các bảng và thực hiện các yêu cầu sau:

1. Viết Trigger để không cho phép xóa học vị khi vẫn đang có giáo viên có học vị đó
2. Viết Trigger để khi xóa một sinh viên thì xóa tất cả các đề tài mà sinh viên đó đang nghiên cứu
3. Viết trigger để thiết lập ràng buộc sau:
 - Nếu giáo viên có học vị “ Tiến sĩ ” thì có thể hướng dẫn tối đa 4 đề tài
 - Nếu giáo viên có học vị “ Thạc sĩ ” thì có thể hướng dẫn tối đa 3 đề tài
 - Nếu giáo viên có học vị “ Kỹ sư ” thì có thể hướng dẫn tối đa 2 đề tài
 - Nếu giáo viên có học vị khác các học vị trên thì không được hướng dẫn nghiên cứu đề tài nào.
4. Viết Trigger để khi thay đổi mã giáo viên thì thay đổi các thông tin mã giáo viên trong các bảng liên quan.
5. Viết trigger để một đề tài có không quá 3 sinh viên tham gia nghiên cứu.
6. Viết trigger để khi xóa một sinh viên thì xóa kết quả bảo vệ của sinh viên đó.

BÀI 14: THỰC HÀNH: KIỂM TRA TỔNG HỢP

14.1. Mục tiêu

Sau bài thực hành này, người học có khả năng:

- ✓ Thành thạo truy vấn, truy vấn nâng cao.
- ✓ Phân quyền người dùng hợp lý.
- ✓ Thiết kế các chỉ mục, view theo yêu cầu.
- ✓ Lập trình viết các thủ tục, hàm và trigger trong một CSDL của ứng dụng cụ thể.
- ✓ Rèn luyện tính cẩn thận và tư duy trong lập trình T-SQL.

14.2. Đề bài tập (Giáo viên phát trực tiếp)