Bài 1: DDL(Data Definition Language)

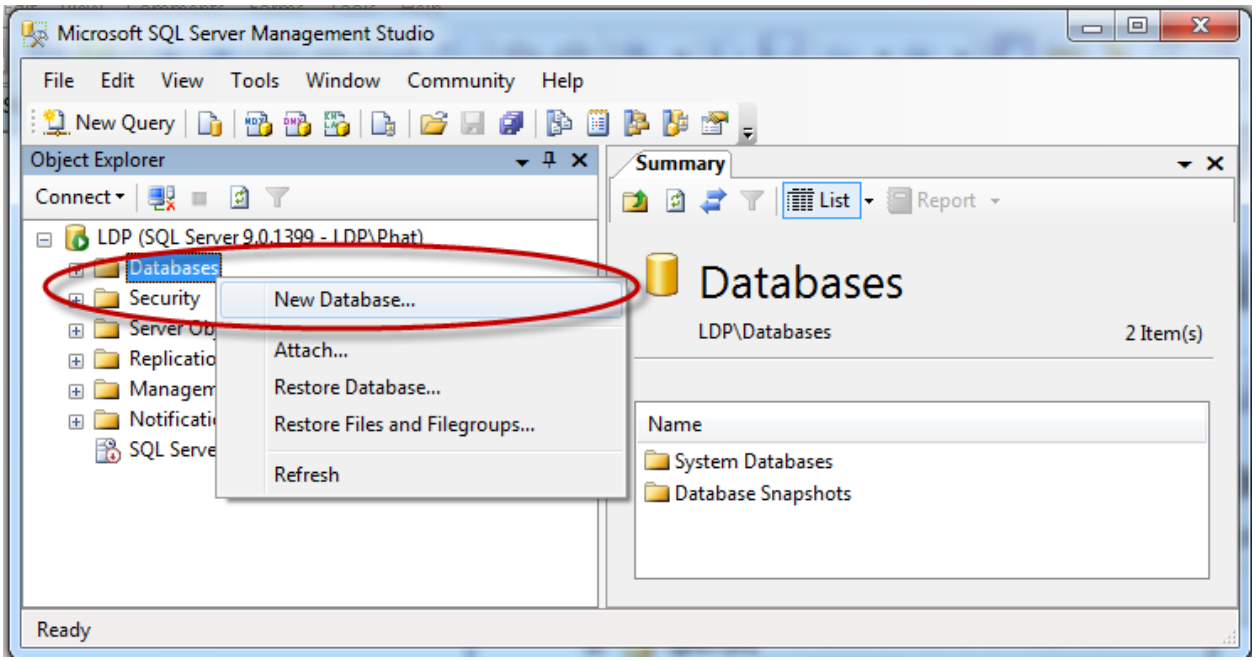
I. MỤC TIÊU:

- Tạo Database
- Các thao tác trên Databse: tạo, thêm, xóa, sửa table
- Các ràng buộc trên Table.

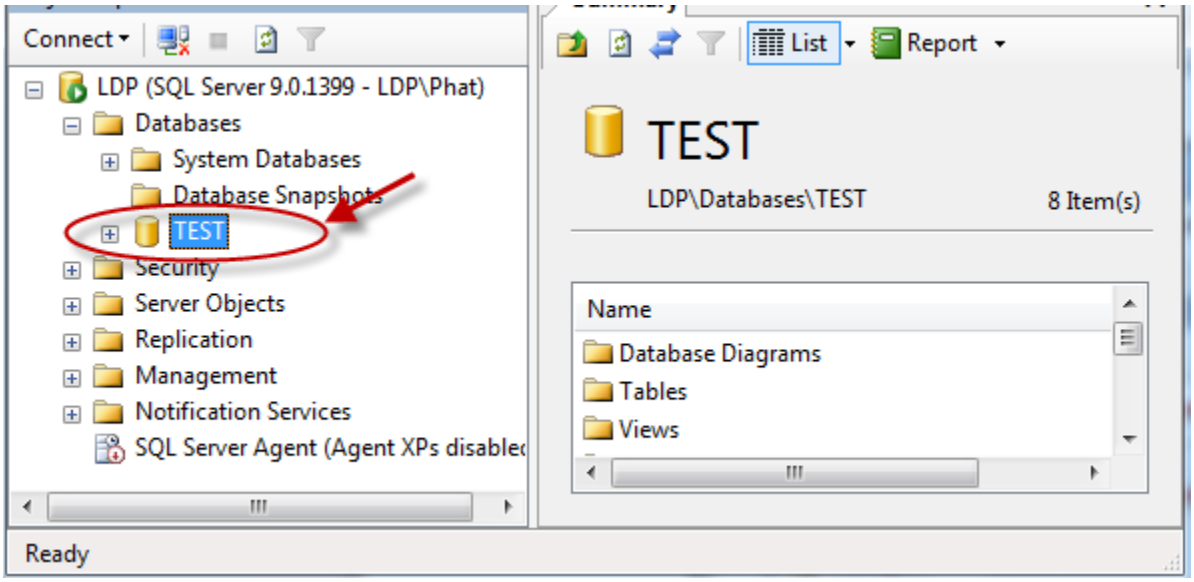
II. TÓM TẮT LÝ THUYẾT:

1. Một số thao tác trên SQL Server 2010 Edition

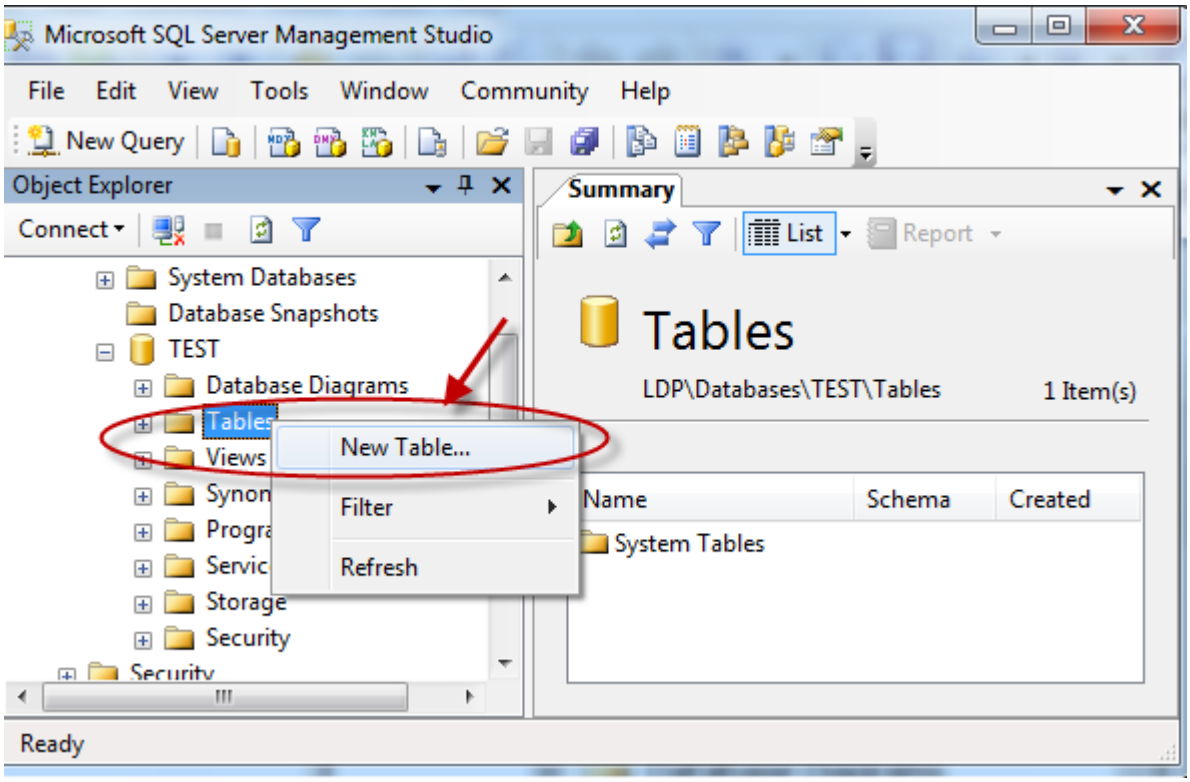
a. Tạo CSDL mới



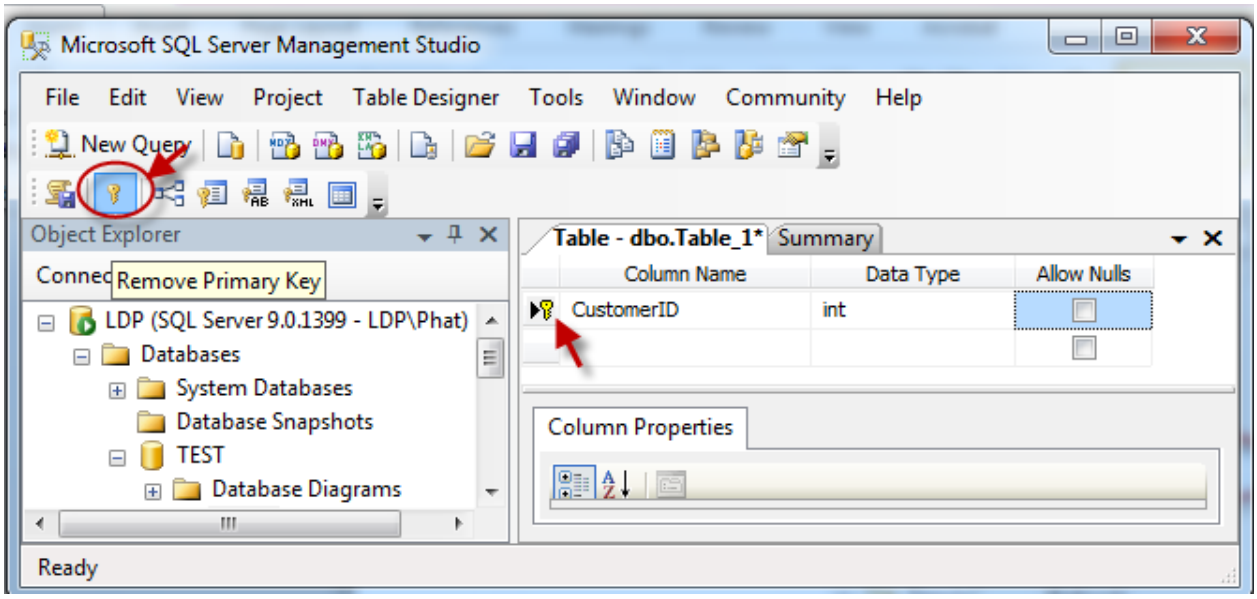
Đặt tên Database trong Textbox Database Name, click OK.



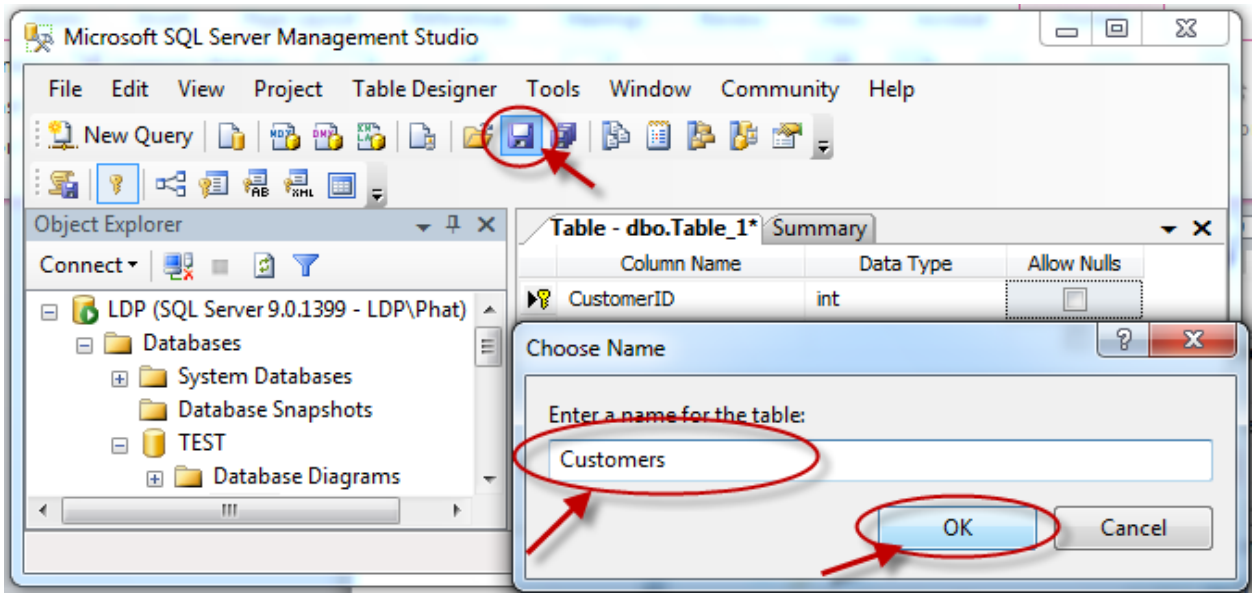
b. Tạo bảng mới

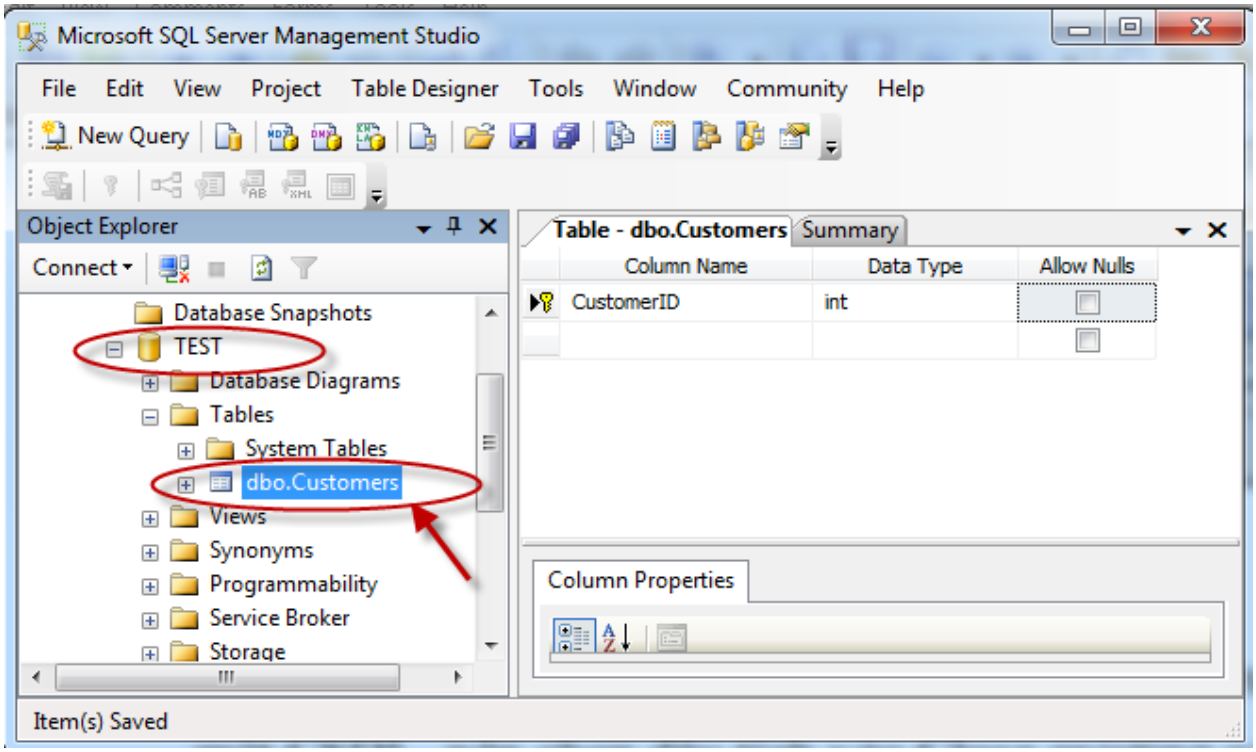


Bảng gồm các các cột. Mỗi cột gồm tên cột (Column Name), kiểu dữ liệu (Data Type) và một giá trị cho biết cột đó có thể chứa giá trị NULL hay không. Trong bảng sẽ có ít nhất một cột làm khóa chính (Primary Key)



Sau khi tạo xong tất cả các cột của bảng, tiến hành Save -> OK

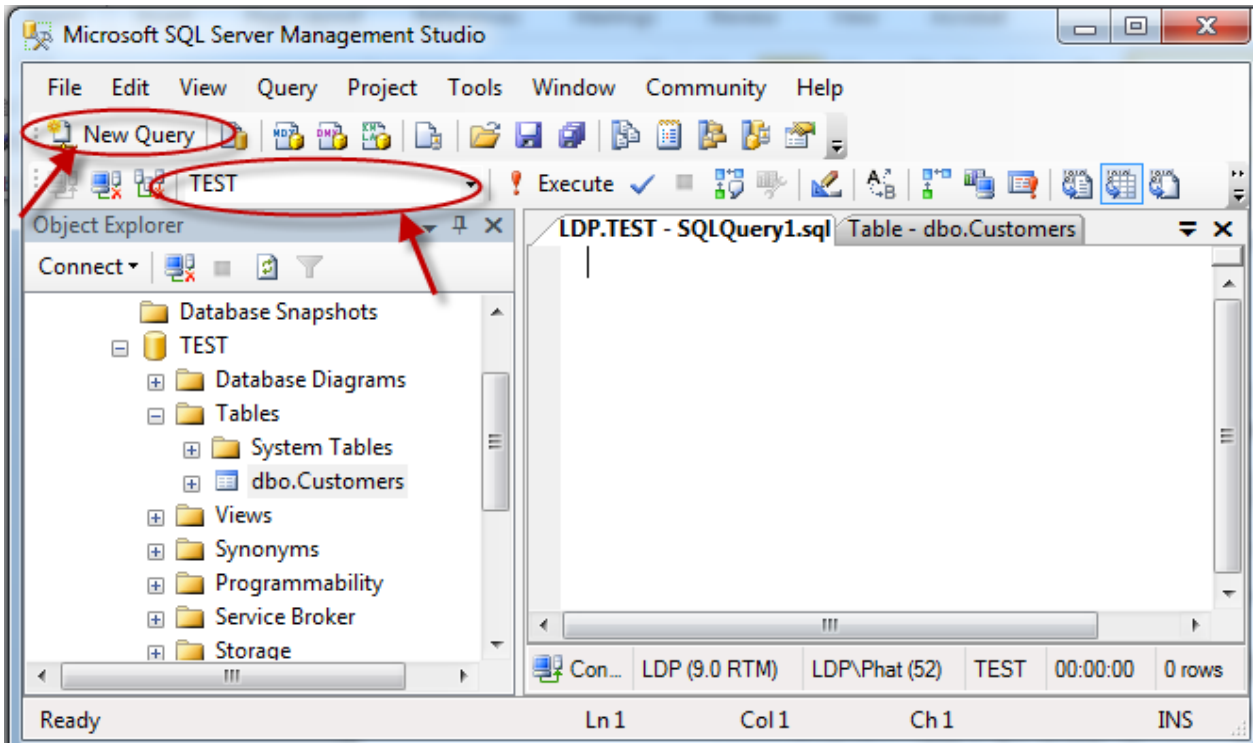




c. Xóa bảng, xóa CSDL

Click chuột phải lên bảng hay CSDL muốn xóa -> Delete - >OK. Trong trường hợp xóa một CSDL, nên chọn dấu tích vào Close existing connections. Khi đó SQL Server 2010 sẽ ngắt tất cả các kết nối vào CSDL này và việc xóa sẽ không gây báo lỗi.

2. Mở một Query Editor để viết câu lệnh SQL



Cần chú ý là câu lệnh SQL sẽ có tác dụng tr ên CSDL đang được chọn trong ComboBox. Do đó cần chú ý lựa chọn đúng CSDL cần tương tác.

3. Ngôn ngữ định nghĩa dữ liệu (Data Definition Language – DDL)

Đây là những lệnh dùng để tạo (create), thay đổi (alter) hay xóa (drop) các đối tượng trong CSDL. Các câu lệnh DDL thường có dạng:

Create object

Alter object

Drop object

Trong đó object có thể là: table, view, storedprocedure, function, trigger…

a. Tạo Database

```
Create Database QLSinhVien
```

Để có thể tạo Database với một số lựa chọn khác có thể tham khảo: SQL Server Books Online (Từ khóa Create Database)

b. *Tạo Table*

- Các kiểu dữ liệu:

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

Char(n)	Kiểu chuỗi với độ dài cố định
Nchar(n)	Kiểu chuỗi với độ dài cố định hỗ trợ UNICODE
Varchar(n)	Kiểu chuỗi với độ dài chính xác
Nvarchar(n)	Kiểu chuỗi với độ dài chính xác hỗ trợ UNICODE
Int	Số nguyên có giá trị từ -2^{31} đến $2^{31} - 1$
Tinyint	Số nguyên có giá trị từ 0 đến 255.
Smallint	Số nguyên có giá trị từ -2^{15} đến $2^{15} - 1$
Bigint	Số nguyên có giá trị từ -263 đến 263-1
Numeric	Kiểu số với độ chính xác cố định.
Decimal	Tương tự kiểu Numeric
Float	Số thực có giá trị từ -1.79E+308 đến 1.79E+308
Real	Số thực có giá trị từ -3.40E + 38 đến 3.40E + 38
Money	Kiểu tiền tệ
Bit	Kiểu bit (có giá trị 0 hoặc 1)
Datetime	Kiểu ngày giờ (chính xác đến phần trăm của giây)
Smalldatetime	Kiểu ngày giờ (chính xác đến phút)
Binary	Dữ liệu nhị phân với độ dài cố định (tối đa 8000 bytes)
Varbinary	Dữ liệu nhị phân với độ dài chính xác (tối đa 8000 bytes)
Image	Dữ liệu nhị phân với độ dài chính xác (tối đa 2,147,483,647 bytes)

- Để tạo Table ta dùng lệnh CREATE TABLE

- Cú pháp:

```
CREATE TABLE <tên bảng>
(   <tên cột 1> <kiểu dữ liệu>,
  …
  <tên cột n> <kiểu dữ liệu>,
  PRIMARY KEY (<tên cột 1>, <tên cột 2>…)
)
```

- Dòng PRIMARY KEY: dùng để tạo khóa chính cho bảng. Lưu ý: khóa chính có thể gồm 1 hoặc nhiều thuộc tính.

CREATE TABLE NHANVIEN

(MANV NVARCHAR(10) NOT NULL,
HOTENNVARCHAR(30) NOT NULL,
GIOITINH BIT,
NGAYSINH SMALLDATETIME,
DIACHI CHAR(50) NULL
DIENTHOAI CHAR(10),
PRIMARY KEY (MANV),
)

Ví dụ: Câu lệnh dưới đây thực hiện việc tạo bảng NHANVIEN bao gồm các cột MaNV, HotenNV, GioiTinh, NgaySinh,DiaChi, DienThoai. Trong đó MaNV là khóa chính

c. Xóa bảng

- Cú pháp: DROP TABLE <tên bảng>

Chú ý: Khi các bảng có ràng buộc khóa ngoại từ các bảng khác tham chiếu đến thì:

+ Phải xóa bảng chứa thuộc tính tham chiếu đến nó trước

+ Hay xóa đi khóa ngoại đó trước khi xóa bảng đó.

d. Thêm cột vào bảng đã có

- Cú pháp:

```
ALTER TABLE <tên bảng> ADD <tên cột> <kiểu dữ liệu>
```

- ***Lưu ý:*** <tên bảng>: bảng đã có trước đó, <tên cột>: cột mới

e. Xóa cột trong bảng đã có

- Cú pháp:

```
ALTER TABLE <tên bảng> DROP COLUMN <tên cột>
```

- ***Lưu ý:*** <tên bảng>: bảng đã có trước đó, <tên cột>: cột muốn xóa

f. Tạo khóa ngoại

- Cú pháp:

```
ALTER TABLE <tên bảng 1>ADD CONSTRAINT <tên khóa ngoại>
FOREIGN KEY (<tên cột 11>, <tên cột 12>,…)  REFERENCES <tên bảng 2> (<tên cột 21>, <tên cột 22>, …)
```

Lưu ý:

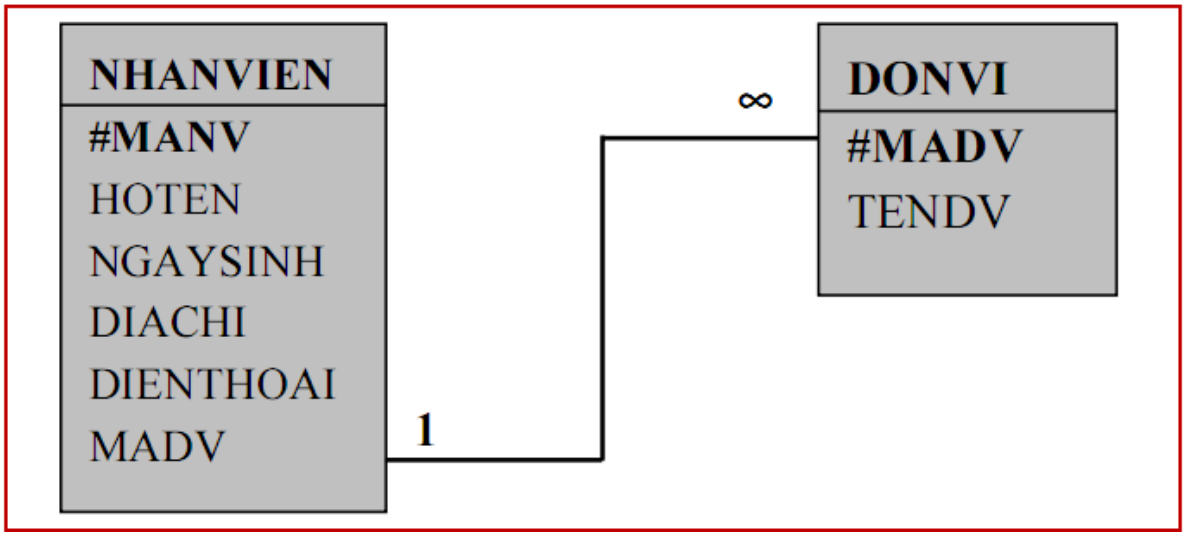
+ <tên bảng 1>: là bảng tham chiếu

+ <tên cột 11>, <tên cột 12>…: các thuộc tính tham chiếu thuộc bảng 1

+ <tên bảng 2>: là bảng được tham chiếu đến

+ <tên cột 21>, <tên cột 22>, …:các thuộc tính được tham chiếu đến thuộc bảng 2

Ví dụ: Tạo 2 bảng NHANVIEN(MANV, HOTEN, NGAYSINH, DIACHI, DIENTHOAI, MADV) và DONVI(MADV, TENDV) theo sơ đồ hình dưới đây:




```
CREATE TABLE donvi
( madv   char(2)  primary key,
   tendv  char(20) not null
)
CREATE TABLE nhanvien
(  manv      char(10) primary key,
  hoten     char(20) not null,
  ngaysinh  datetime null,
  diachi    char(50) ,
  dienthoai char(6) ,
  madv      char(2)  foreign key(madv) references donvi(madv)
)
```

g. Xóa khóa ngoại

- Cú pháp:

```
DROP CONSTRAINT <tên khóa ngoại>
```

h. Tạo ràng buộc kiểm tra:

SQL server hỗ trợ 3 dạng khai báo toàn vẹn dữ liệu:

- Ràng buộc mức cột (Domain Integrity): cho phép chỉ định tập giá trị hợp lệ của cột, cột có được mang giá trị NULL hay không.
- Ràng buộc mức dòng (Entity Integrity): đảm bảo các dòng trong table phải được nhận diện .
- Ràng buộc tham chiếu (Referential Integrity) : cho phép đảm bảo ràng buộc dữ liệu giữa hai bảng có quan hệ : the primary key table và foreign key table.

Các khai báo toàn vẹn dữ liệu này được thực thi qua việc tạo các ràng buộc (constraint) : default, check, referential, primary key, unique, foreign key.

📖 NULL / NOT NULL:

+ Tính chất null của một cột là không bắt buộc phải nhập dl cho cột đó
+ Nếu không xác định Null hay Not Null thì giá trị mặc định sẽ được sử dụng là Null

📖 DEFAULT : thiết lập giá trị mặc định cho cột khi không chỉ định giá trị cho cột

+ Giá trị Default dùng gán giá trị hằng cho cột
+ Các cột Timestamp, indentity không có default constraint vì giá trị tự động xác định
+ Giá trị default có thể là hằng số, hàm hệ thống, biến toàn cục hoặc hàm do người dùng định nghĩa

📖 DEFAULT CONSTRAINT: giá trị default chỉ có ảnh hưởng trong cột của bảng

+ Định nghĩa default constraint khi tạo bảng:

```
CREATE TABLE Table_name
(Column_name Datatype [NULL| NOT NULL]
[CONSTRAINT Constraint_name] DEFAULT  expression […] )
```

+ Định nghĩa default constraint khi bảng đã tồn tại

```
ALTER TABLE Table_name
ADD [CONSTRAINT Constraint_name] DEFAULT  expression FOR column_nam
```

Ví dụ: Tạo bảng events với default constraint

```
CREATE  TABLE events
(EventID int Indentity(1, 1) Not Null,
EventType nvarchar(10) Not Null,
EventTitle nvarchar(100) Null,
EventDate SmallDatetime Null Default Getdate()
)
```

Ví dụ: Thêm các Default constraint cho bảng events:

```
ALTER TABLE events
ADD DEFAULT 'party' for EventType
```

+ Xoá default constraint:

```
ALTER TABLE Table_name
DROP CONSTRAINT Constraintname
```

📖 CHECK:

- Định nghĩa check Constraint:

+ Khi tạo bảng:

```
CREATE TABLE  table_name
(column_name  data_type
[CONSTRAINT  constraint_name]
CHECK (logical expression)
```

+ Khi bảng đã tồn tại

```
ALTER TABLE  table_name
ADD [CONSTRAINT  constraint_name]
```

- CHECK (logical expression): *logical expression* là biểu thức logic kiểm tra giá trị do người sử dụng nhập vào

Ví dụ 1:

```
CREATE TABLE  nhanvien
(manv  smallint     PRIMARY KEY,
tennv varchar(50)     NOT NULL ,
tuoimin  tinyint NOT NULL  CHECK (tuoimin >= 18),   tuoimax tinyint
NOT NULL  CHECK (tuoimax <= 40)
)
```

Ví dụ 2:

```
ALTER TABLE Chucvu
ADD CONSTRAINT HSCV_chk
CHECK (HSPC>=0.1 AND HSPC<=0.5)
```

Ví dụ 3: Tạo một ràng buộc cho bảng DONVI trên cột MADV quy định mã đơn vị phải có dạng 2 chữ số (ví dụ 01,02,…)

```
ALTER TABLE Donvi
ADD  CONSTRAINT check_madv
CHECK (madv LIKE  '[0-9] [0-9]')
```

📖 UNIQUE CONSTRAINT

+ Khái niệm: dùng đảm bảo không có giá trị trùng trong các cột không phải là khóa chính, chấp nhận giá trị null, nhưng chỉ có một giá trị null

+ Mức bảng:

```
CREATE TABLE table_name
(columnname datatype[,…]
[CONSTRAINT constraint_name]
UNIQUE {(column[ASC|DESC][,…n])}
[CLUSTERED | NONCLUSTERED ]
```

+ Khi bảng đã tồn tại:

```
ALTER TABLE  table_name
ADD [CONSTRAINT  constraint_name] UNIQUE
[CLUSTERED | NONCLUSTERED ] (column_name)
```

Ví dụ:

```
CREATE TABLE  nhanvien
(Manhanvien  NUMERIC(6),
Tennv   NVARCHAR(25) NOT NULL,
Email   NVARCHAR(25) UNIQUE,
MucLuong     NUMERIC(8,2)
)
```

Hoặc:

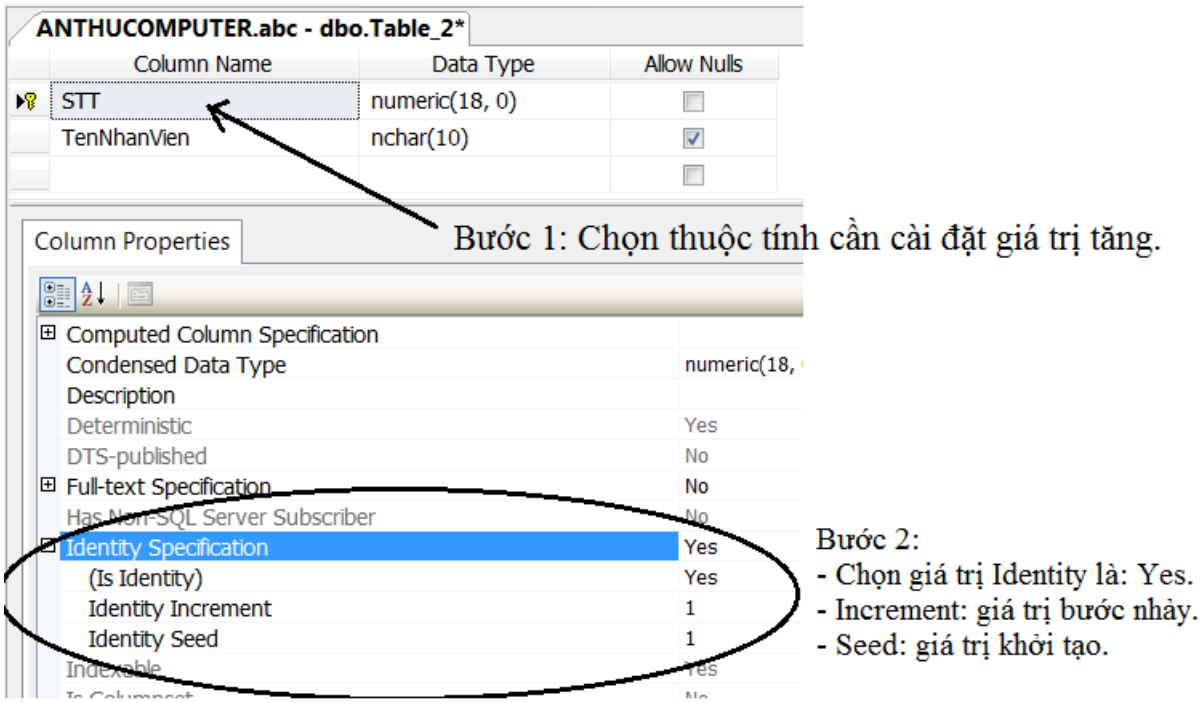
```
CREATE TABLE  nhanvien
(Manhanvien  NUMERIC(6),
Tennv   NVARCHAR(25)  NOT NULL,
Email NVARCHAR(25),
MucLuong NUMERIC(8,2)
CONSTRAINT uni_email  UNIQUE(email)
)
```

5. Tạo giá trị tăng tự động:

nhapxuat : Table

stt	ngay	loai	phieu	hoten	LYDO	makho	mavt	solg	tien	manv
1	5/1/1995	n	a011	lê văn tám	nhập ủy thác	lb	tn01	20	20000	sp01
2	5/2/1995	n	a012	nguyễn thị tuyết	nhập theo HD01	lb	ve03	50	25000	sp02
4	5/6/1995	n	a013	trần ngọc dũng	sữa lkho long bình	ch	ts05	5	50000	sp02
5	5/3/1995	n	n015	lý thùy vân	chuyển hàng cứng	px	ve05	50	50000	sp02
6	5/5/1995	n	x012	trần văn mười	mua gỗ xây căng	ch	ve03	12	24000	sp02
8	5/6/1995	n	x016	trần ngọc dũng	mua xây nhà kho	ch	ts03	10	10000	sp02
9	5/7/1995	n	x020	hoàng ái trân	bán chvlxd nhà	ch	ts05	4	6000	sp01
11	5/3/1995	x	n011							

Các bước thực hiện:



III. NỘI DUNG THỰC HÀNH

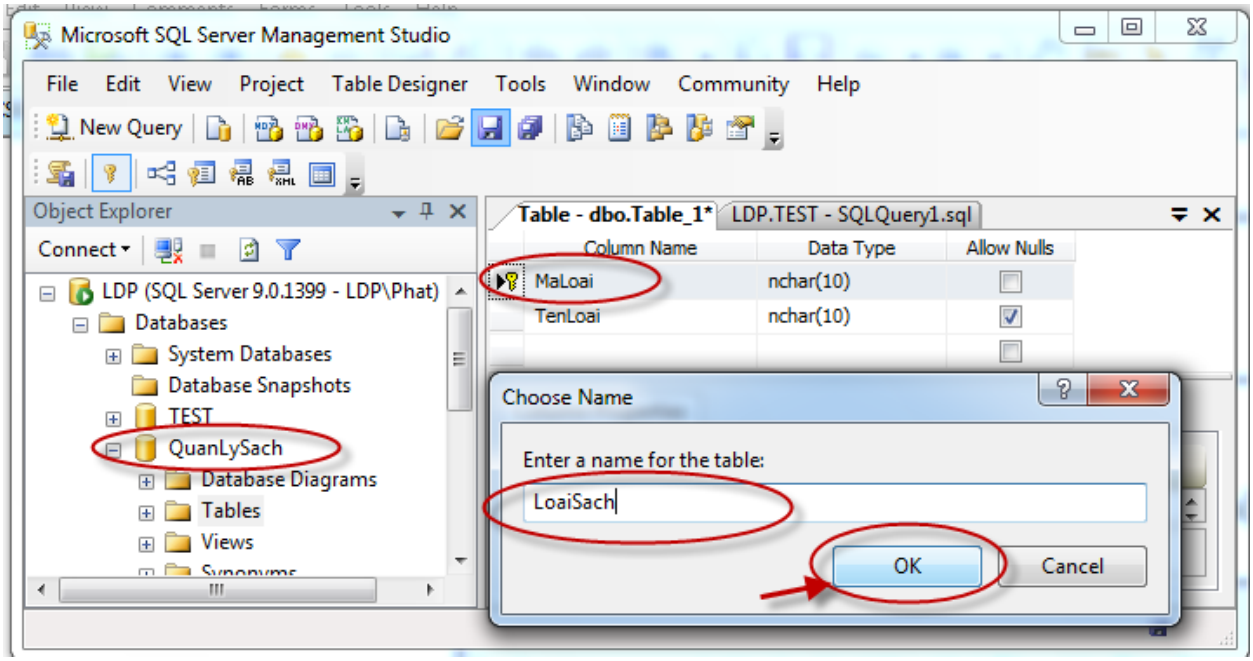
Bài 1. Dùng SQL Server Management studio tạo Database và Table, các ràng buộc trên Table và tạo liên kết bảng.

Cho CSDL"QuanLySach" như sau:

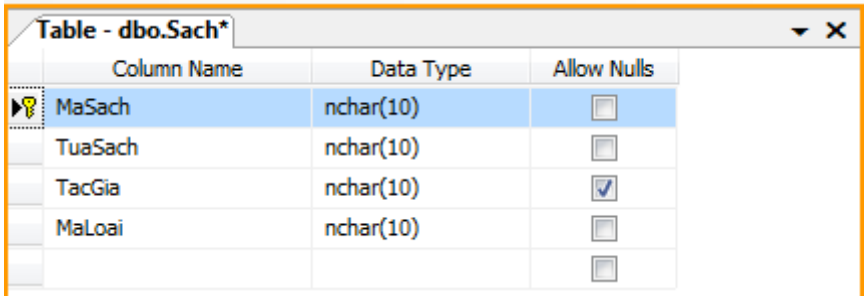
+ LoaiSach (MaLoai, TenLoai)

+ Sach (MaSach, TuaSach, TenTacGia, MaLoai)

- Các khóa ngoại: + Sach.MaLoai -> LoaiSach.MaLoai

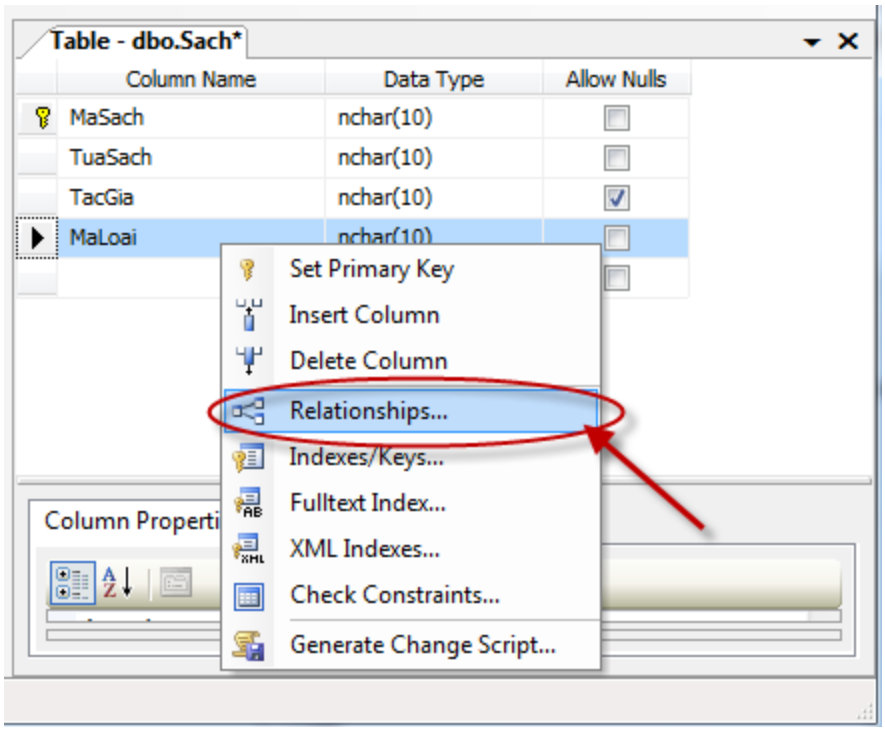


- Tạo các field cho bảng Sach

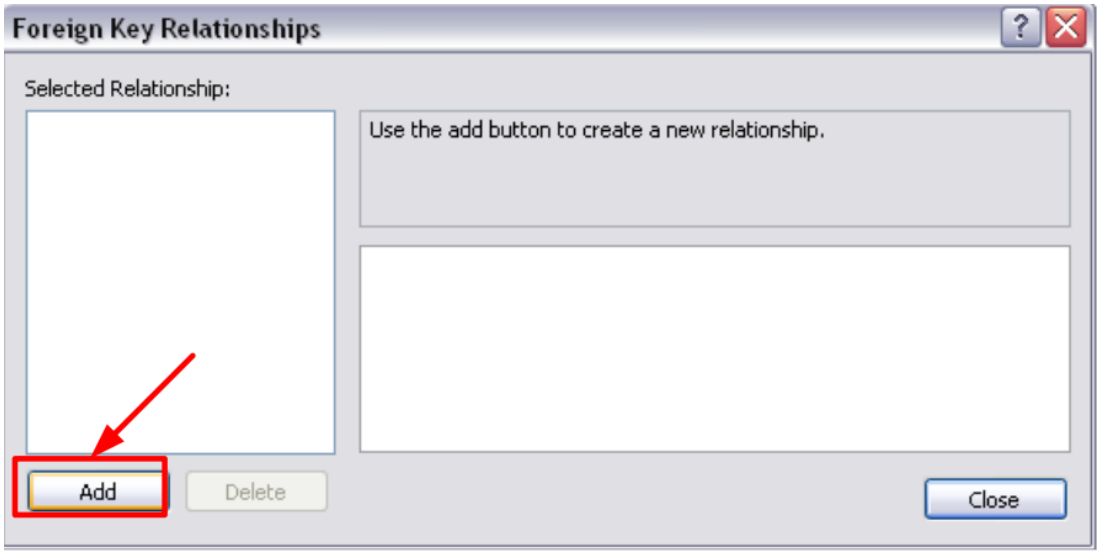


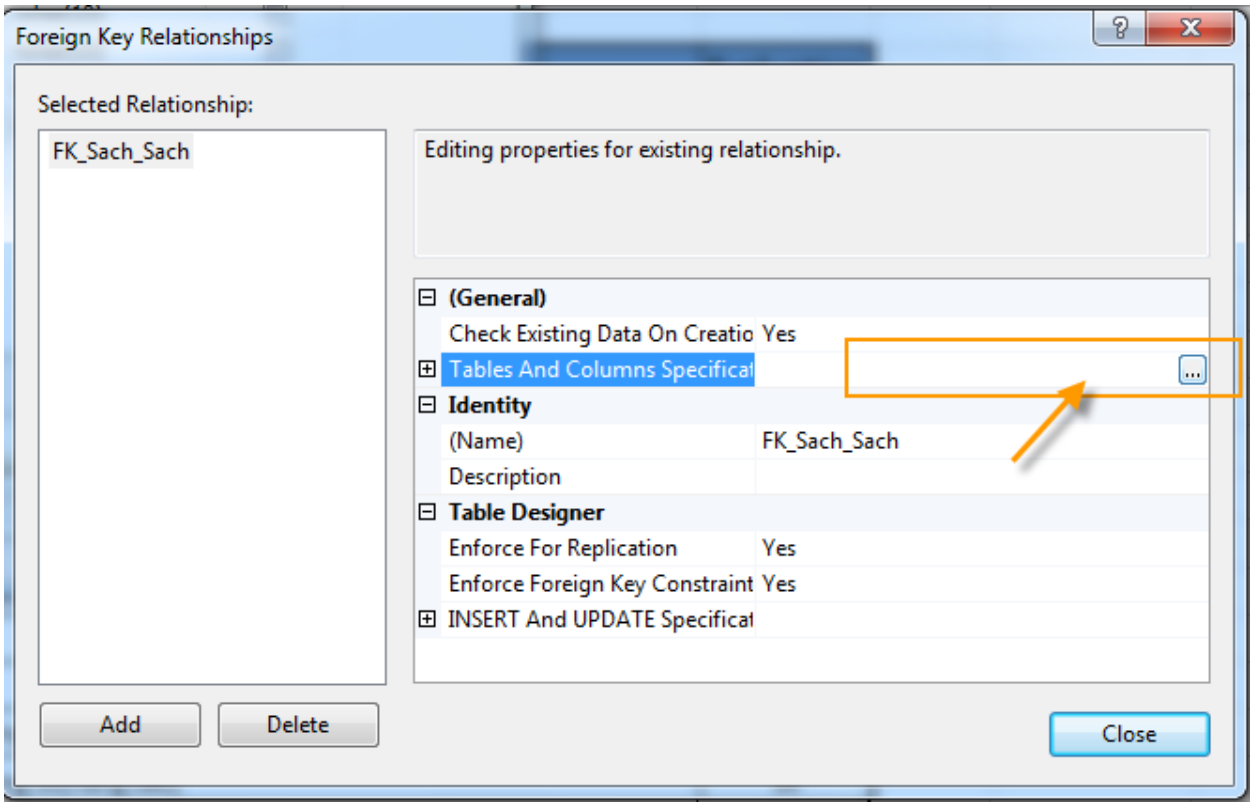
- Tạo khóa ngoại Sach.MaLoai -> LoaiSach.MaLoai:

Chuột phải cột Mã loại của bảng Sach -> Relationships

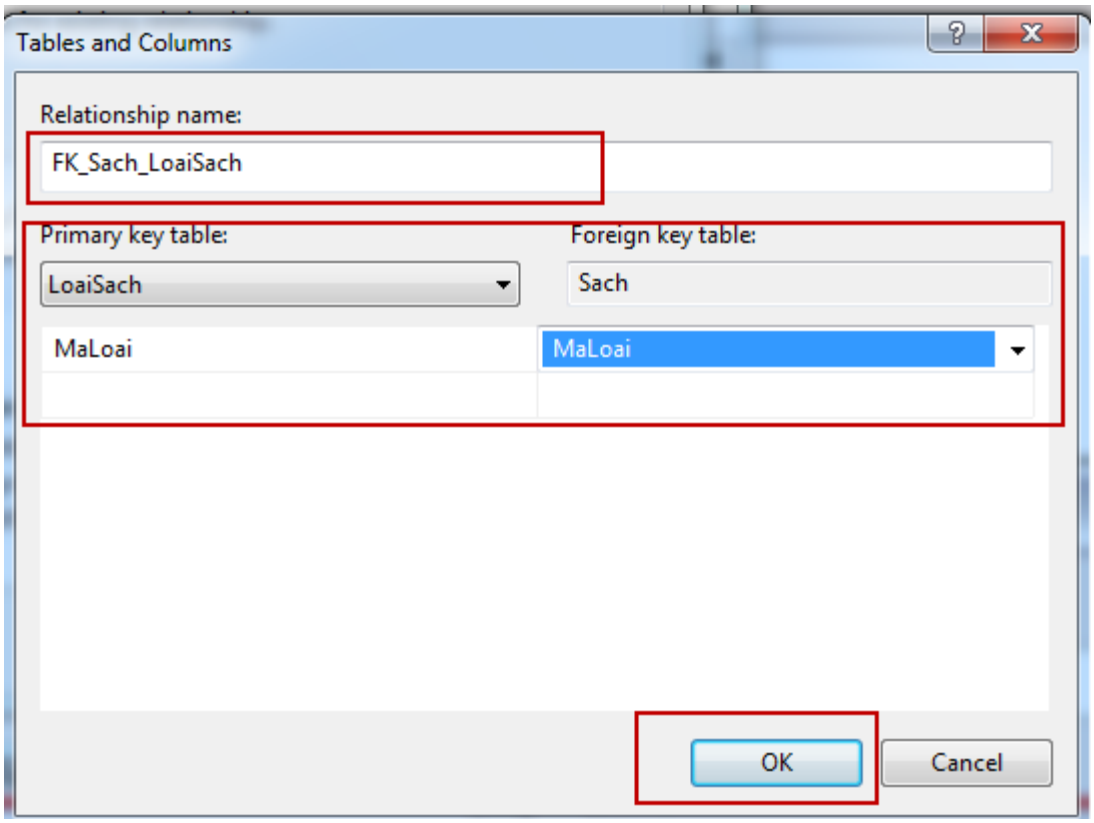


Add khóa ngoại:

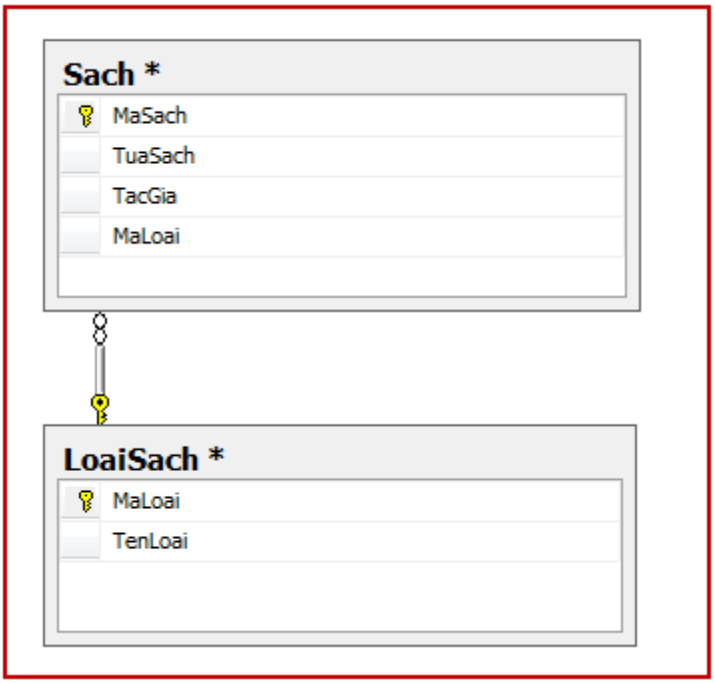




- Sửa lại tên khóa ngoại là: FK_Sach_LoaiSach
- Chọn Primary key table là LoaiSach với cột MaLoai
- Chọn Foreign key table là Sach với cột MaLoai



- Xem lược đồ CSDL cuối cùng:



Bài 2. Chạy file script DDL tạo database QuanLyHoiNghiKhoaHoc, tạo table và các ràng buộc trên table.

- Cho CSDL như sau:

+ BaiBaoCao (MaBaoCao, TuaBaoCao, TacGia)

+ PhienBaoCao (MaPhien, Ngay, GioBatDau, MaChuDe, SoLuongBaoCao)

+ ChuDe (MaChuDe, TenChuDe)

+ DangKy (MaBaoCao, MaChuDe)

+ LichBaoCao (MaBaoCao, MaPhien)

- Khóa ngoại:

+ DangKy.MaBaoCao -> BaiBaoCao.MaBaoCao

+ DangKy.MaChuDe -> ChuDe.MaChuDe

+ LichBaoCao.MaBaoCao -> BaiBaoCao.MaBaoCao

+ LichBaoCao.MaPhien -> PhienBaoCao.MaPhien

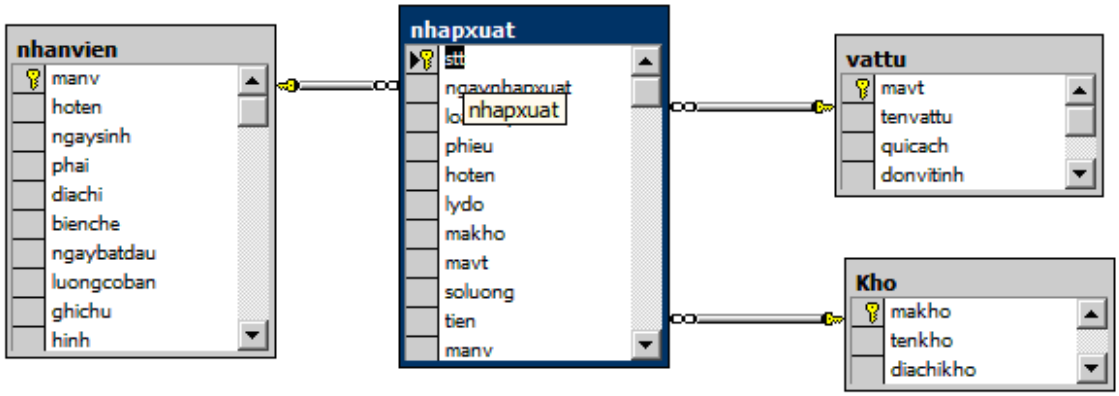
Yêu cầu:

+ Chạy file Script tạo Database QLHoiNghiKhoaHoc

+ Chạy file Scrip tạo table

+ Chạy file tạo khóa ngoại.

Bài 3. Cho mối quan hệ giữa các bảng sau:



Yêu cầu

a. Dùng màn hình Query Analyzer tạo csdl QuanlyKho

Có dữ liệu các bảng như sau:

kho : Table

	makho	tenkho	dckho
▶	ch	chánh hưng	chánh hưng, quận 8
	lb	long bình	xã long bình- đồng nai
	px	phú xuân	123 trần xuân soạn - nhà bè
	TT		
*			

Record: 1 of 4

vattu : Table

	mavt	tenvt	qcach	dvtinh
	tn01	tôn nhựa xanh	1.2mx0.8m	tấm
	tn02	tôn nhựa trắng	1.2mx0.8m	tấm
	ts03	tôn sóng 3 ly	2mx1.6m	tấm
	ts05	tôn sóng 5 ly	2mx1.6m	tấm
	ve03	ván ép con ngựa	3ly 1.5	tấm
▶	ve05	ván ép tân mai	5ly 1.2	tấm
*				

Record: 6 of 6

nhanvien : Table

	manv	hoten	nsinh	phai	dchi	bche	bdau	lcb	ghichu	hinh
▶	sp01	lê văn sơn	16/11/46	Yes	23/28 huỳnh khương an	True	9/10/1993	650	chuyển hàng	
	sp02	bùi thị phương nga	17/07/54	No	184/4b lê văn sỹ-pn	True	9/10/1993	830	quản lý	
	vt01	đào hữu ngư	25/12/77	Yes	123 ly thuong kiet-q10	False	5/15/1994	570	lập trình	
	vt02	trần thị vân	18/06/69	No	302 đường 3 tháng 2- q.3	False	7/15/1994	750	hoa sỹ	
	vt03	lê phước kháng	26/10/70	Yes	747 trần trung trực-q.4	False	10/20/1994	530	kỹ sư	
*										

nhapxuat : Table

stt	ngay	loai	phieu	hoten	LYDO	makho	mavt	solg	tien	manv
1	5/1/1995	n	a011	lê văn tám	nhập ủy thác	lb	tn01	20	20000	sp01
2	5/2/1995	n	a012	nguyễn thị tuyết	nhập theo HD01	lb	ve03	50	25000	sp02
4	5/6/1995	n	a013	trần ngọc dũng	sữa lkho long bình	ch	ts05	5	50000	sp02
5	5/3/1995	n	n015	lý thùy vân	chuyển hàng cứng	px	ve05	50	50000	sp02
6	5/5/1995	n	x012	trần văn mười	mua gỗ xây căng	ch	ve03	12	24000	sp02
8	5/6/1995	n	x016	trần ngọc dũng	mua xây nhà kho	ch	ts03	10	10000	sp02
9	5/7/1995	n	x020	hoàng ái trân	bán chvlxd nhà	ch	ts05	4	6000	sp01
11	5/3/1995	x	n011							
(bel)										

b. Bổ sung ràng buộc thiết lập giá trị mặc định bằng 1 cho cột SOLG và bằng 0 cho cột TIEN trong NHAPXUAT.

c. Bổ sung ràng buộc cho bảng NHANVIEN để đảm bảo rằng một nhân viên chỉ có thể làm việc trong công ty khi đủ 18 tuổi và không quá 60 tuổi.

d. Với các bảng đã tạo được, câu lệnh: DROP TABLE nhanvien

Có thể thực hiện được không? Tại sao?

-Hết-

Bài 2: Lập trình T-SQL cho Store Proceduce(SP)

I. MỤC TIÊU:

- Cách khai báo procedure không truyền tham số
- Cách khai báo Procedure có truyền tham số
- Cách khai báo procedure có sử dụng cấu trúc điều khiển
- Cách khai báo Procedure có sử dụng cấu trúc lặp

II. TÓM TẮT LÝ THUYẾT:

1. STORED PROCEDURE

1.1 Định nghĩa

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

- Tên của Stored Procedure
- Các tham số
- Thân của Stored procedure: bao gồm các lệnh của T-SQL dùng để thực thi procedure.

1.2 Cú pháp

a. Lệnh tạo Procedure

```
CREATE PROCEDURE  procedure_name
        {@parameter data_type  input/output }/*các biến tham số vào  ra*/
AS
Begin
        [khai báo các biến cho xử lý]
        {Các câu lệnh transact-sql}
End
```

Ghi chú:

- Trong SQL Server, có thể ghi tắt một số từ khóa mà tên có chiều dài hơn 4 ký tự. ***Ví dụ:*** có thể thay thế Create Procedure bằng Create Proc
- Tên hàm, tên biến trong SQL không phân biệt chữ hoa thường.

b. Khai báo biến và gán giá trị cho biến, ghi chú

```
/*Khai báo biến*/
DECLARE  @parameter_name data_type

/*Gán giá trị cho biến*/
SET @parameter_name = value
SELECT @parameter_name = value

/*In thông báo ra màn hình*/
print N'Chuỗi thông báo unicode'

--Ghi chú 1, một dòng
/*
  Ghi chú 2
  Nhiều dòng
*/
```

c. Biên dịch và gọi thực thi một Stored-procedured

- Biên dịch: Chọn toàn bộ lệnh tạo Stored-procedure -> Nhấn F5

✓ Gọi thực thi một Stored-Procedure đã được biên dịch bằng lệnh Exec.

```
EXECUTE procedure_name --Stored-proc không tham số
EXEC procedure_name Para1_value, Para2_value, … --Stored-proc có tham số
```

✓ Lệnh cập nhật Procedure

```
ALTER PROCEDURE  procedure_name
        [ {@parameter data_type } ]
AS
Begin
        [khai báo các biến cho xử lý]
        {Các câu lệnh transact-sql}
End
```

✓ Lệnh xóa

```
DROP PROCEDURE  procedure_name
```

1.3 Ví dụ

Tạo Stored-procedure tính tổng của 2 số nguyên

```
--Tạo stored-procedure sp_tong
CREATE PROCEDURE sp_Tong
        @So1  int, @So2   int, @Tong  int out
AS
Begin
        SET @Tong = @So1 + @So2;
End

--Biên dịch stored-procedure → F5

--Kiểm tra
Declare @Sum int
Exec sp_Tong 1, -2, out @Sum
Select @Sum
```

2. CẤU TRÚC ĐIỀU KIỂN

2.1 *Lệnh IF.. ELSE*

- Chức năng: Xét điều kiện để quyết định những lệnh T-SQL nào sẽ được thực hiện.
- Cú pháp:

```
IF Biểu_thức_điều_kiện
    Lệnh | Khối lệnh
[ELSE Lệnh | Khối lệnh]
```

Khối lệnh là một hoặc nhiều lệnh nằm trong cặp từ khóa Begin…end

2.2 *Lệnh WHILE*

- Chức năng: Thực hiện lặp đi lặp lại một đoạn T-SQL khi điều kiện còn đúng.
- Cú pháp:

```
WHILE Biểu_thức_điều_kiện
Lệnh | Khối lệnh
```

- Có thể sử dụng Break và Continue trong khối lệnh của While

- Break: Thoát khỏi vòng lặp While hiện hành
- Continue: Trở lại đầu vòng While, bỏ qua các lệnh sau đó.

2.3 *Lệnh CASE*

- Chức năng: Kiểm tra một dãy các điều kiện và kiểm tra kết quả phù hợp với điều kiện đúng. Lệnh CASE được sử dụng như một hàm trong câu lệnh SELECT.
- Cú pháp: có hai dạng

○ Dạng 1 (Simple Case)

```
CASE Biểu_Thức_Đầu_Vào
WHEN Giá_Trị THEN Kết_Quả
[…n]
[ELSE Kết_Quả_Khác]
END
```

○ Dạng 2 (Searched Case)

```
CASE
WHEN Biểu_thức_điều_Kiện THEN Kết_quả
[…n]
[ELSE kết_quả_khác]
END
```

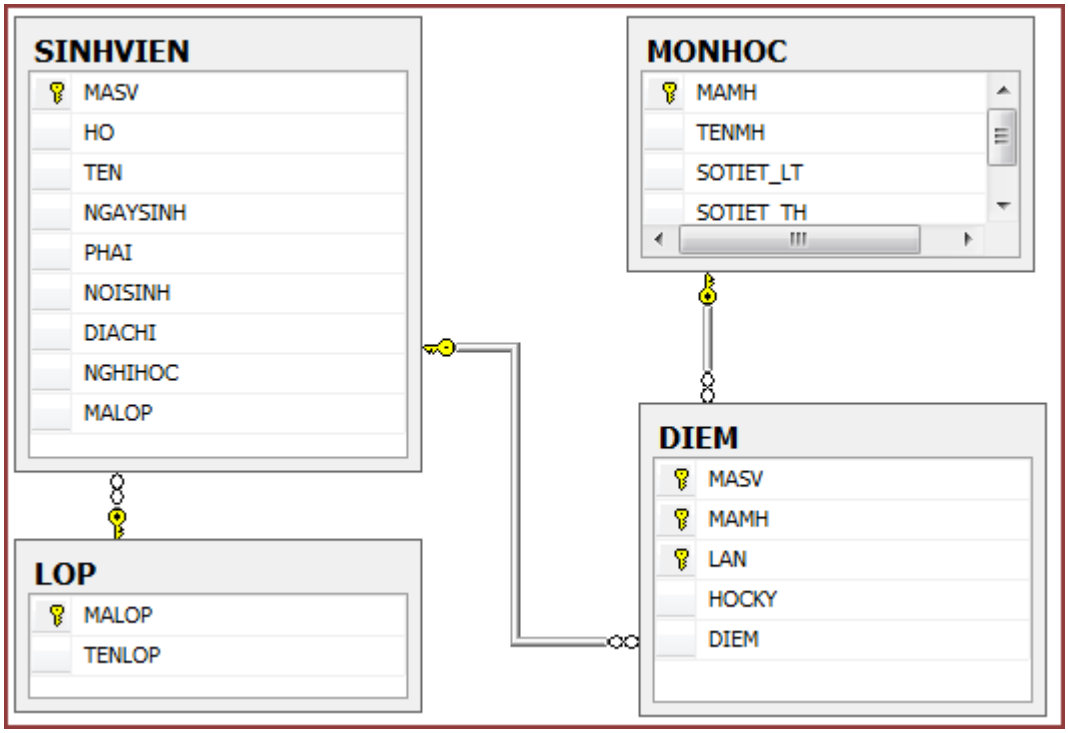
III. NỘI DUNG THỰC HÀNH

Bài 1. Thực thi các Store Procedure đơn giản(không truyền tham số).

a. Chạy Script Store spChuoi, xuất câu: "Đại Học Công Nghệ Sài Gòn"

```
CREATE PROCEDURE  spChuoi
AS
Begin
        print N'Đại Học công nghệ Sài Gòn'
End
Exec spChuoi
```

b. Chạy File Script tạo CSDL về QLSV như sau:



- Chạy file Scrip Store spDSSV: Xuất ra DSSV

```
CREATE PROC spDSSV
AS
Begin
        SELECT * FROM SINHVIEN
End
EXEC spDSSV
```

c. Chạy file Scrip Store spDiemSV: Xuất ra điểm của Sinh viên có mã số là "SV1"

```
CREATE PROC spDIEMSV
AS
Begin
        SELECT * FROM SINHVIEN inner join Diem ON
  SINHVIEN.MASV = DIEM.MASV AND SINHVIEN.MASV = 'SV1'
End
EXEC spDIEMSV
```

Bài 2. Thực thi các Store Procedure đơn giản(có truyền tham số)

a. Chạy file Scrip Store SpChuoi2: Xuất câu "Đại Học Công Nghệ Sài Gòn" + @Khoa (Trong đó @Khoa là tham số)

```
CREATE PROC spCHUOI2 @khoa char(50), @Chuoi char(100) OUTPUT
AS
BEGIN
	SET @Chuoi = N'DHCNSG_' + @khoa
END
DECLARE @Chuoi char(100)
EXEC spChuoi2 'Khoa CNTT',@chuoi output
PRINT @Chuoi
```

b. Chạy file Scrip Store SpTB: Xuất ra trung bình của 3 số @a, @b, @c

```
CREATE PROC spTB @a iNT ,@b int ,@c int, @TB float OUTPUT
AS
BEGIN
	SET @TB = (@a+@b+@c)/3.0
END
DECLARE @TB FLOAT
EXEC spTB 2,8,4, @TB OUTPUT
PRINT @TB
```

c. Chạy file Scrip Store spThemSV, Thêm một sinh viên mới vào bảng SinhVien với thông tin sau:

- Mã SV: Sv20
- Họ: Tran Van
- Tên: Long
- Ngày Sinh: 4/10/1985
- Phái: Nam
- Mã lớp: MMT2

```
CREATE PROC spThemSV
@MASV CHAR(8), @HO VARCHAR(40),@TEN VARCHAR(10),@NGAYSINH SMALLdatetime,@phai char(3),@MALOP CHAR(8)
AS
BEGIN
```

```
        INSERT INTO SINHVIEN(masv,ho,ten,ngaysinh,phai,malop)
        VALUES(@MASV,@HO,@TEN,@NGAYSINH,@PHAI,@MALOP)
END
EXEC SPTHEMSV'SV20','TRAN VAN','LONG','4/10/1985','nam','MMT2'
```

d. Chạy file Script Store spXoaMaLop: Xóa một lớp trong bảng lớp và kiểm tra ràng buộc

```
Create proc spXoaLop
@malop char(8)
as
if exists(select malop from Lop where malop=@malop)
begin
        if exists(select s.malop from sinhvien as s, lop as l where l.malop=s.malop and
s.malop = @malop)
                begin
                        print 'Khong the xoa ma lop vi ton tai rang buoc khoa ngoai'
                        return 0
                end
        else
        delete from lop where malop = @malop
        print 'Da xoa ma lop' + @malop
        end
else
        begin
print 'Khong co ten ma lop ' +@malop
return 0
end
```

Bài 3. Viết các Store Procedure đơn giản sau:

a. Tạo Store SpTongDay: Xuất tổng các số tự nhiên từ 1 đến @n(@n là tham số truyền vào)
b. Tạo Store spNhapDiem: Nhập điểm cho 1 sinh viên
c. Tạo Store spDiemLop: Xuất ra bảng điểm cho một lớp.

Bài 4. Viết store procedure có tên spSuaMaSV: Để sửa mã sinh viên trong bảng sinh viên.

Hướng dẫn:

Bước 1: Chép MaSV cần sửa trong bảng SinhVien ra bảng tạm (temp_SV)

Bước 2: Chép MaSV cần sửa trong bảng Diem ra bảng tạm (temp_Diem)

Bước 3: Xóa MaSV cũ trong Diem

Bước 4: Xóa MaSV cũ trong SinhVien

Bước 5: Cập nhật MaSV mới trong temp_sv

Bước 6: Cập nhật MaSV mới trong temp_Diem

Bước 7: Chép bảng temp_sv sang SinhVien. Sau đó xóa temp_sv

Bước 8: Chép bảng temp_Diem sang Diem. Sau đó xóa temp_Diem

Bài 5. Viết các Store Procedure kết hợp sử dụng cấu trúc điều khiển (IF, CASE) và các hàm đơn giản

a. Viết Store spSoNguyenTo: Xuất ra các số từ 1 đến @n(@n là tham số)
b. Sử dụng CSDL về QLSV ở trên. Viết Store SpTenMon, sử dụng câu lệnh IF dựa vào MaMon xuất ra tên môn học (Ví dụ: Nếu MAMH = 'MAV'. Xuất ra: Anh Van).
c. Viết Store SpTemMon1 giống như trên nhưng sử dụng câu lệnh CASE.

Bài 6. Viết store procedure có tên spSuaMaMH: Để sửa mã môn học trong bảng Môn Học.

-Hết-

I. MỤC TIÊU:

- Nắm vững các tham số truyền vào trong stored procedure.
- Kết hợp stored procedure với các lệnh T-SQL.

II. TÓM TẮT LÝ THUYẾT:

1. Tham số trong stored procedure

Stored Procedure là 1 hàm được lưu trữ sãn trong CSDL. Hàm này có thể có 2 loại tham số chính: Tham số đầu vào và tham số đầu ra.

1.1 Tham số đầu vào

Đây là loại tham số mặc định, cho phép truyền các giá trị vào trong Stored Procedure để hỗ trợ xử lý

Ví dụ:

```
CREATE PROC Cong
      @So1 int, @So2 int
AS
BEGIN
      declare @Kq int
      set @Kq = @So1 + @So2
      print @Kq
END
GO

exec Cong 1,2
```

Kết quả đoạn lệnh trên cho kết quả là "3".

Hình 1: Kết quả thực thi stored procedure cộng 2 số nguyên

1.2 Tham số đầu ra

Tham số dùng để nhận kết quả trả về từ stored procedure. Sử dụng từ khóa OUTPUT (hoặc viết tắt là OUT) để xác định tham số.

Ví dụ:

```
CREATE PROC Tru
      @So1 int,
      @So2 int,
      @Kq int output
AS
BEGIN
      set @Kq = @So1 - @So2
END
GO
--Thuc thi ket qua
DECLARE @test int
EXEC Tru 1, 2, @test output
PRINT @test
```

2. Trả về giá trị trong Stored Procedure

Ngoài cách sử dụng tham số đầu ra để trả về giá trị. Có thể sử dụng RETURN để trả về giá trị từ stored procedure hoặc các câu lệnh SELECT khi truy vấn dữ liệu

2.1 Trả về giá trị từ lệnh RETURN

Lệnh return được sử dụng để trả về giá trị từ stored procedure mà không cần sử dụng tham số đầu ra. Giá trị trả về này có một số đặc điểm:

- Giá trị trả về chỉ có thể là số nguyên. Nếu trả về các loại giá trị khác thì lúc thực thi stored procedure sẽ báo lỗi (ngoại trừ một số kiểu dữ liệu được tự động chuyển đổi sang kiểu số nguyên như: float, double,…)
- Giá trị trả về mặc định là 0, Có thể nhận giá trị trả về này bằng 1 biến
- Sau khi gọi RETURN, stored procedure sẽ trả về giá trị và kết thúc xử lý

Ví dụ:

```
CREATE PROC Test
      @Lenh int
      AS
      BEGIN
            if (@Lenh = 1)
                  return 1
            if (@Lenh = 2)
                  begin
                        declare @float float
                        set @float = 2.6
                        return @float
                  end
            if (@Lenh = 3)
                  begin
                        declare @char varchar(50)
                        set @char = 'hello'
                        return @char
                  end
      END
      GO
      --Thuc thi
      declare @test float
      EXEC @test = Test 1
      print @test
```

Nếu giá trị truyền vào là 1: stored procedure trả về giá trị “1”.

Nếu giá trị truyền vào là 2: stored procedure trả về giá trị “2”.

Nếu giá trị truyền vào là 3: stored procedure báo lỗi không thể chuyển chuỗi „hello“ thành số nguyên.

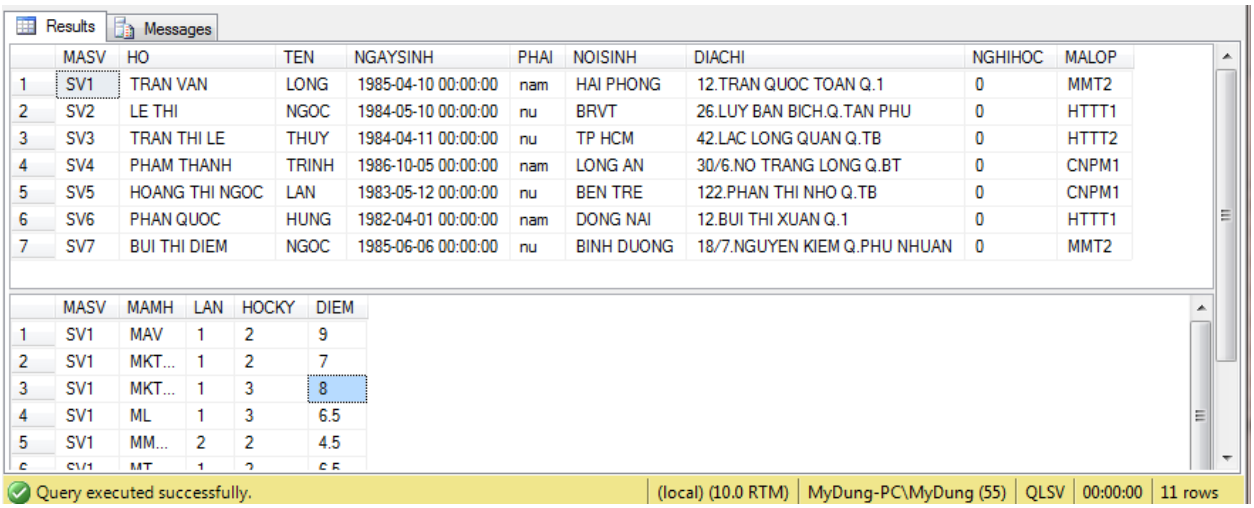
Nếu truyền các giá trị khác: stored procedure trả về giá trị “0”.

2.2 *Trả về dữ liệu từ lệnh SELECT*

Mỗi lệnh SELECT đặt trong stored procedure sẽ trả về 1 bảng

```
CREATE PROC TestSelect
AS
BEGIN
      SELECT * FROM SINHVIEN
      SELECT * FROM DIEM
END
GO
--THUC THI
EXEC TestSelect
```

Kết quả in ra màn hình sẽ là:



Results | Messages

	MASV	HO	TEN	NGAYSINH	PHAI	NOISINH	DIACHI	NGHIHOC	MALOP
1	SV1	TRAN VAN	LONG	1985-04-10 00:00:00	nam	HAI PHONG	12.TRAN QUOC TOAN Q.1	0	MMT2
2	SV2	LE THI	NGOC	1984-05-10 00:00:00	nu	BRVT	26.LUY BAN BICH.Q.TAN PHU	0	HTTT1
3	SV3	TRAN THI LE	THUY	1984-04-11 00:00:00	nu	TP HCM	42.LAC LONG QUAN Q.TB	0	HTTT2
4	SV4	PHAM THANH	TRINH	1986-10-05 00:00:00	nam	LONG AN	30/6.NO TRANG LONG Q.BT	0	CNPM1
5	SV5	HOANG THI NGOC	LAN	1983-05-12 00:00:00	nu	BEN TRE	122.PHAN THI NHO Q.TB	0	CNPM1
6	SV6	PHAN QUOC	HUNG	1982-04-01 00:00:00	nam	DONG NAI	12.BUI THI XUAN Q.1	0	HTTT1
7	SV7	BUI THI DIEM	NGOC	1985-06-06 00:00:00	nu	BINH DUONG	18/7.NGUYEN KIEM Q.PHU NHUAN	0	MMT2

	MASV	MAMH	LAN	HOCKY	DIEM
1	SV1	MAV	1	2	9
2	SV1	MKT...	1	2	7
3	SV1	MKT...	1	3	8
4	SV1	ML	1	3	6.5
5	SV1	MM...	2	2	4.5

Query executed successfully. | (local) (10.0 RTM) | MyDung-PC\MyDung (55) | QLSV | 00:00:00 | 11 rows

Hình 2: Kết quả thực hiện Stored procedure "TestSelect"

3. Kết hợp Stored Procedure với các lệnh T-SQL

Các stored procedure thông thường được tạo ra nhằm thực hiện một số chức năng cần thao tác trong cơ sở dữ liệu. Khi đó, ta cần phải kết hợp nhiều lệnh T-SQL thao tác với dữ liệu như (SELECT, INSERT, UPDATE, DELETE) và các cấu trúc điều khiển (IF, WHILE, CASE,..)

3.1 Ứng dụng thêm sinh viên vào cơ sở dữ liệu

```
CREATE PROC ThemSinhVien
      @mssv char(8),
      @ho varchar(40),
      @ten varchar(10),
      @ngaySinh smalldatetime,
      @phai char(3),
      @diachi varchar(100),
      @maLop char(8)
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, ho,TEN,NGAYSINH,PHAI,DIACHI, maLop)
VALUES(@mssv, @ho,@ten, @ngaySinh, @phai,@diachi, @maLop)
RETURN 0 /* procedure tự trả về 0 nếu không RETURN */
END
GO
```

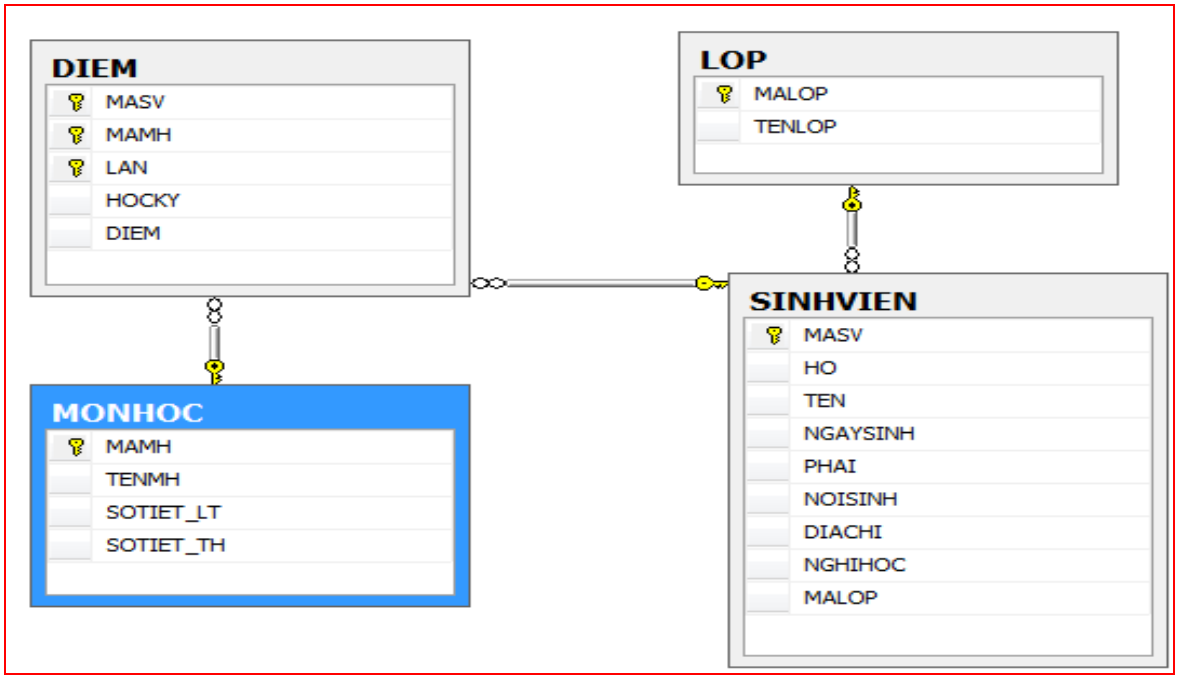
```
--Thực thi
DECLARE @KQ int
EXEC @KQ = THEMSINHVIEN
'SV8','LE','Dung','01/29/95','Nu',N'QUAN 11, TPHCM','TH1'
PRINT @KQ
```

3.2 *Ứng dụng trả về danh sách sinh viên trong lớp*

```
CREATE PROC XuatDSSinhvien
      @malop char(8)
AS
BEGIN
      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
      SELECT * FROM SINHVIEN SV INNER JOIN LOP L ON
SV.MALOP=L.MALOP
            WHERE L.MALOP=@malop
      /*PROCEDURE luôn trả về 0 nếu không có RETURN */
END
--Thực thi
EXEC XuatDSSinhvien 'CNPM1'
```

III. NỘI DUNG THỰC HÀNH

1. Cho cơ sở dữ liệu như sau:



Mô tả cụ thể của các bảng như sau:

a. Bảng Lớp:

Field Name	Type	Constraint
MALOP	Char(8)	Primary Key
TENLOP	Varchar(30)	Unique, Not Null

b. *Bảng sinh viên:*

Field Name	Type	Constraint
MASV	Char(2)	Primary Key
HO	Varchar(40)	Not Null
TEN	Varchar(10)	Not Null
NGAYSINH	SmallDateTime	
PHAI	Char(3)	Default: 'NAM'
NOISINH	Varchar(50)	Default: ' '
DIACHI	Varchar(100)	Default: ' '
NGHIHOC	Bit	Default: 0
MALOP	Char(8)	Foreign Key

c. *Bảng môn học*

Field Name	Type	Constraint
MAMH	Char(8)	Primary Key
TENMH	Varchar(30)	Unique, Not Null
SOTIET_LT	Int	Default: 45 Check: >=0 và <=120
SOTIET_TH	Int	Default: 0 Check: >=0 và <=120

d. *Bảng điểm*

Field Name	Type	Constraint
MASV	Char(8)	Primary Key, Foreign Key
MAMH	Char(8)	Primary Key, Foreign Key
LAN	Int	Primary Key, LAN>=1 và LAN<=2
HOCKY	Int	>=1 và <=8
DIEM	Real	>=0 và <=10 hoặc =-1

(***Bài tập về nhà:*** *Các bạn về nhà viết câu lệnh SQL tạo các bảng trên.)*

2. Yêu cầu thực hành:

Câu 1: Viết Stored procedure để thêm một môn học mới.

Tên thủ tục: sp_ThemMH

Bài 3: Lập trình T-SQL cho Store Proceduce(SP) (tiếp theo)

Câu 2: Viết sp (stored procedure) sửa điểm của sinh viên có các tham số truyền vào như sau: MaSV [input], MaMH [input], DiemSua [input].

Tên thủ tục: sp_SuaDiem

Câu 3: Viết sp cho biết sĩ số của các lớp có trong trường. Thông tin cần xuất ra: MaLop, TenLop, SiSo.

Tên thủ tục: sp_XuatSiSo

Câu 4: Viết sp cho biết danh sách các lớp chưa có sinh viên vào học: Thông tin cần xuất ra: Mã lớp, tên lớp

Tên thủ tục: sp_XuatLop

Câu 5: Viết sp thống kê số lượng sinh viên nam, số sinh viên nữ của các lớp. Kết quả xuất ra bao gồm: Malop, tenLop, So_SVNam, So_SVNu.

-Hết-

Bài 4: Lập trình T-SQL cho Hàm do người dùng định nghĩa (User Defined Functions)

I. MỤC TIÊU:

Thực tập về hàm do người dùng định nghĩa(User Defined Functions):

- ✓ Hàm trả về giá trị là kiểu dữ liệu cơ sở.
- ✓ Hàm trả về một bảng có được từ một câu truy vấn.
- ✓ Hàm trả về một bảng mà dữ liệu có được sau một chuỗi thao tác xử lý và insert.

II. TÓM TẮT LÝ THUYẾT:

❖ **User defined functions:** là các thủ tục chứa đựng các câu lệnh SQL bên trong nó. Giống như store procedure, UDFs cũng có thể được truyền các tham số nhưng UDFs được biên dịch và thực thi tại thời điểm thực thi vì vậy khá chậm hơn so với store procedure.

❖ **Sự hạn chế của User designed functions:**

- UDFs không thể thực hiện các câu lệnh DML như Insertion, Update và Deletion trên bảng.
- UDFs không thể trả về các giá trị không xác định giống như câu lệnh Getdate() ..v.v..
- Stored procedure không thể được gọi từ bên trong một UDFs ngược lại 1 stored procedure có thể gọi một UDFs hoặc một stored procedure bên trong nó.

❖ **Có 3 kiểu dữ liệu trả về của UDFs:**

1. Scalar Functions (returns a single value) : Hàm trả về giá trị là kiểu dữ liệu cơ sở.

Cú pháp:

```
CREATE FUNCTION Tên_hàm([Danh sách tham số])
RETURNS /* Kiểu_trả_về_của hàm */
AS
BEGIN
/* các câu lệnh sql … */
RETURN /* Trị trả về của hàm*/
END
```

Ví dụ:

```
CREATE FUNCTION dbo.fsdt(@MaNhanVien varchar(12))
RETURNS varchar(12)
AS
BEGIN
        DECLARE @sdt varchar(12) – Khai báo biến trả về.
        SELECT @sdt=nhanvien.sdt – Gán giá trị cho biến trả về
```

```
        from nhanvien
        where nhanvien.MaNhanVien=@MaNhanVien
        RETURN @sdt – trả về kết quả hàm
END
go
```

Thực thi: Với hàm được định nghĩa như trên, câu lệnh:

```
select dbo.fsdt('NV01')
```

Sẽ cho kết quả là số điện thoại của nhân viên 'NV03'.

2. Inline Functions (returns a table): Hàm với giá trị trả về là dữ liệu kiểu bảng

Cú pháp:

```
CREATE FUNCTION Tên_hàm([Danh sách tham số])
RETURNS TABLE
AS
RETURN /* Câu lệnh Select */
```

Lưu ý:

- Kiểu trả về của hàm phải được chỉ định bởi mệnh đề **RETURNS TABLE.**
- Trong phần thân của hàm chỉ có duy nhất một câu lệnh RETURN xác định giá trị trả về của hàm thông qua duy nhất một câu lệnh **SELECT**. Ngoài ra, không sử dụng bất kỳ câu lệnh nào khác trong phần thân của hàm.
- Khi một inline function trả về một bảng, ta có thể sử dụng hàm đó trong mệnh đề from của một câu truy cấn khác

Ví dụ:

```
CREATE FUNCTION dbo.Get_nhanvien_ofpb(@MaPhongBan varchar(10))
RETURNS TABLE
AS
RETURN
(SELECT * from nhanvien
where nhanvien.MaPhongBan=@MaPhongBan)
```

Thực thi:

Với hàm được định nghĩa như trên, câu lệnh:

```
select * from dbo.Get_nhanvien_ofpb('PB1')
```

Sẽ trả về danh sách các nhân viên của phòng 'PB1'.

3. Table valued Functions (multiple operations, complex logic just like Stored procedures): Hàm trả về một bảng mà dữ liệu sau một chuỗi thao tác xử lý và insert.
Cú pháp:

```
CREATE FUNCTION Tên_hàm([Danh sách tham số])
RETURNS @bien_bang TABLE định_nghĩa_bảng
AS
BEGIN
RETURN /* Các xử l ý Insert dữ liệu vào @bien_bang…*/
END
```

Ví dụ:

```
CREATE FUNCTION dbo.DS_phongban()
RETURNS @Danhsach TABLE (MaPhongBan varchar(10),sonv int)
AS
BEGIN
        Insert into @Danhsach
        Select nhanvien.MaPhongBan,COUNT(nhanvien.MaNhanVien)
        from nhanvien
        group by nhanvien.MaPhongBan
        RETURN
END
```

Thực thi : Với hàm được định nghĩa như trên, câu lệnh:

```
select * from dbo.DS_phongban()
```

Cho kết quả thống kê tổng số nhân viên của các phòng ban.

Lưu ý: Khi một table-valued function trả về một bảng, ta có thể sử dụng bảng đó trong mệnh đề from của một câu truy vấn khác.

4. Tạo hàm User Defined Functions:

- Trong Server Explorer, mở cơ sở dữ liệu cần tạo hàm → programability → Right_click vào function folder.
- Chọn New Inline Function, New Table-valu ed Function, hoặc New Scalar-valued Function trên shortcut menu.

  Lưu ý: N ếu bạn chọn tạo hàm ban đầu là **Inline function** thì bạn không thể sửa câu lệnh SQL thành hàm Scalar-Valued Function, vì sẽ xảy ra lỗi khi lưu.
- Tên hàm là duy nhất.

➢ Viết các lệnh SQL cho hàm.

5. Xóa hàm user **defined function:**

```
DROP <Tên_hàm_cần_xóa>
```

III. NỘI DUNG THỰC HÀNH:

Thực thi file script "Lab4567.sql". Sử dụng CSDL này để thực hiện các yêu cầu sau:

Bài 1: Thực thi các câu lệnh bên dưới, nhận diện từng loại hàm và cho biết kết quả nhận được sau khi thực thi từng câu lệnh.

a. Cho biết hàm bên dưới trả về kiểu dữ liệu gì? Loại hàm gì? Kết quả?

```
--Xây dựng hàm
CREATE FUNCTION sfunc1(@a int, @b int, @c int)
RETURNS int
AS
BEGIN
        DECLARE @max int – Khai báo biến trả về ở đây.
        SET @max=@a – Lệnh SET dùng gán giá trị cho biến
        if @max<@b set @max=@b
        if @max<@c set @max=@c
        RETURN @max – trả về kết quả của hàm
END
GO
select dbo.sfunc1(5,3,8) –Thực thi
```

b. Cho biết hàm bên dưới trả về kiểu gì? Kết quả? Loại hàm gì?

```
--Xây dựng hàm
CREATE FUNCTION ifunc2(@thang int)
RETURNS TABLE
AS
RETURN
(SELECT * from BangLuong where month(BangLuong.NgayThang)=@thang)
GO
select * from dbo.ifunc2(1)
 --Thực thi hàm ifunc2
```

c. Cho biết hàm bên dưới trả về kiểu gì? Kết quả? Loại hàm gì?

```
--Xây dựng hàm
CREATE FUNCTION dbo.m_func3()
RETURNS @Danhsach TABLE (MaPhongBan varchar(10),sonv int) -- @Danh sách là một biến kiểu bảng.
AS
BEGIN
        Insert into @Danhsach – Lệnh gán giá trị cho một biến kiểu bảng.
        select nhanvien.MaPhongBan, COUNT(nhanvien.MaNhanVien)
        from nhanvien group by nhanvien.MaPhongBan
        RETURN
END
select * from dbo.m_func3();--Thực thi
```

Bài 2: Dùng loại hàm Scalar-valued function để xây dựng hàm theo các yêu cầu bên dưới:

a. Xây dựng hàm tính tổng của 2 số nguyên.

b. Xây dựng hàm giải phương trình bậc 1 ax+b=0

c. Xây dựng hàm tính tuổi với tham số là ngày sinh.

d. Sử dụng hàm tính tuổi ở câu c để tính tuổi cho các nhân viên trong bảng nhân viên.

Bài 3: Dùng loại hàm InlineTable-valued function để xây dựng hàm theo các yêu cầu bên dưới:

a. Xây dựng hàm tính tuổi của các nhân viên trong bảng nhân viên (hàm không đối số).

b. Xây dựng hàm tính tổng tiền tạm ứng của từng nhân viên(hiển thị MaNhanVien, hotennv, ThucLanh).(hàm không đối số)

c. Xây dựng hàm giống câu b có sử dụng đối số: @thang int, để thống kê tạm ứng theo đối số tháng của mỗi nhân viên.

Bài 4: Dùng loại hàm Multi statement Table-valued function để xây dựng hàm theo các yêu cầu bên dưới:

a. Xây dựng hàm tính tổng ngày công của nhân viên(hàm không đối số).

b. Xây dựng hàm trả về danh sách lương các nhân viên theo tháng.

Bài 5: Dùng loại hàm Scalar-valued function để xây dựng các hàm tính tiền tạm ứng còn lai, thực lãnh. Dùng loại hàm Multi statement Table-valued function để xây dựng hàm tính lương nhân viên theo tháng trong đó có sử dụng các hàm Scalar-valued.

-Hết-

I. MỤC TIÊU:

Thực tập về cursor.Thực thi các lệnh khai báo cursor, mở cursor, lọc dữ liệu từ trong cursor. Thực thi các lệnh kiểm tra, đóng cursor. Sử dụng cursor để duyệt dữ liệu.

II. TÓM TẮT LÝ THUYẾT:

1. Giới thiệu:

✓ Một cursor là một vùng làm việc tạm thời được tạo ra trong bộ nhớ hệ thống khi một câu lệnh SQL được thực thi.

✓ Một cursor chứa thông tin của câu lệnh select và các hàng dữ liệu mà nó truy cập.

✓ Vùng làm việc tạm thời được sử dụng để lưu trữ dữ liệu được lấy từ cơ sở dữ liệu và các thao tác trên dữ liệu đó.

✓ Một cursor có thể chứa nhiều hơn một hàng, nhưng chỉ có thể xử lý một hàng tại một thời điểm.

2. Khai báo Cursor

```
DECLARE cursor_name
INSENSITIVE | SCROLL | CURSOR  FOR
<select_statement >
FOR  READ ONLY | UPDATE  [OF column_name [ ,...n ]]
--Transact-SQL Extended Syntax
DECLARE <cursor_name> CURSOR
[ LOCAL | GLOBAL ]
[ FORWARD_ONLY | SCROLL ]
[ STATIC | KEYSET | DYNAMIC | FAST_FORWARD ]
[ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
[ TYPE_WARNING ]
FOR
<select_statement >
[ FOR UPDATE [ OF column_name [ ,...n ] ] ]
```

Lưu ý: Tên Cursor trong các cách khai báo không bắt đầu bằng ký tự @

Ý nghĩa các tham số:

- ⇨ **Insensitive/Static**: Nội dung của cursor không thay đổi trong suốt thời gian tồn tại, trong trường hợp này cursor chỉ là ReadOnly.
- ⇨ **Dynamic**: trong thời gian tồn tại, nội dung của cursor có thể thay đổi nếu dữ liệu trong các bảng liên quan có thay đổi.
- ⇨ **Local**: cursor cục bộ, chỉ có thể sử dụng trong phạm vi một khối (query batch) hoặc một thủ tục/hàm.
- ⇨ **Global**: cursor toàn **cục**, có thể sử dụng trong một thủ tục/hàm hay một query batch bất kỳ hoặc đến khi bị hủy một cách tường minh.
- ⇨ **Forward_only**: cursor **chỉ** có thể duyệt một chiều từ đầu đến cuối.
- ⇨ **Scroll**: có thể duyệt lên xuống cursor tùy ý (duyệt theo đa chiều)
- ⇨ **Read_only**: chỉ có thể đọc từ cursor, không thể sử dụng cursor để update dữ liệu trong các bảng liên quan(ngược lại với for update…)

Mặc định khi khai báo cursor nếu không chỉ ra các tùy chọn thì cursor có các tính chất:

- Global
- Forward_only
- **Read_only** hay "**for update**" tùy thuộc vào câu truy vấn
- Dynamic

3. Duyệt cursor

- ⇨ **D**ùng lệnh **Fetch** để duyệt tuần tự qua cursor theo cú pháp:

```
FETCH
NEXT|PRIOR|FIRST|LAST|ABSOLUTE <n|@nvar>|RELATIVE <n|@nvar>
FROM
GLOBAL <cursor_name> | <@cursor_variable_name>
< INTO @variable_name [ ,...n ] >
```

- ⇨ **Mặc đinh**: **Fetch next**
- ⇨ **Đối với Cursor** dạng **Forward_Only**, chỉ có thể **Fetch Next**
- ⇨ Biến hệ thống @@Fetch_Status cho biết lệnh fetch vừa thực hiện có thành công hay không, giá trị của biến này là cơ sở để biết đã duyệt đến cuối cursor hay chưa.

4. Qui trình sử dụng cursor

- ⇨ Khai báo cursor.
- ⇨ "Mở" cursor bằng lệnh Open: Open tên_cursor
- ⇨ Khai báo các biến tạm để chứa phần tử hiện hành(đang được xử lý) của cursor:

+ Các biến tạm phải cùng kiểu dữ liệu với các trường tương ứng của phần tử trong cursor

+ Có n trường trong cursor thì phải có đủ n biến tạm tương ứng

⇨ Fetch (next,…) cursor để chuyển đến vị trí phù hợp.

+ Có thể đưa các giá trị của dòng hiện hành vào các biến thông qua mệnh đề into của lệnh fetch.

+ nếu không có mệnh đề into, các giá trị của dòng hiện hành sẽ được hiển thị ra cửa sổ kết quả (result pane) sau lệnh fetch.

+ Có thể sử dụng vị trí hiện tại như là điều kiện cho mệnh đề where của câu delete/update (nếu cursor không là read_only).

⇨ Lặp lại việc duyệt và sử dụng cursor, có thể sử dụng biến @@fetch_status để biết đã duyệt qua hết cursor hay chưa. @@fetch_status=0: lấy dữ liệu thành công. @@fetch_ status<0: không lấy được dữ liệu.

⇨ Đóng cursor bằng lệnh Close: **Close Tên_cursor**

Lưu ý: Sau khi đóng vẫn có thể mở lại nếu cursor chưa bị hủy

⇨ Hủy cursor bằng lệnh DEALLOCATE: **Deallocate Tên_cursor**

❖ Ví dụ 1: Tập hợp kết quả được tạo ra khi mở cursor bao gồm tất cả các hàng và các cột trong bảng.Thực hiện lấy dòng đầu tiên bằng lệnh Fetch next …

```
declare curnhanvien cursor --khai báo cursor
for select * from nhanvien
open curnhanvien --mở cursor
fetch next from curnhanvien --Duyệt dòng hiện hành trong cursor
close curnhanvien --đóng cursor
deallocate curnhanvien --xóa cursor
```

❖ Ví dụ 2: Với tùy chọn SCROLL trong dòng lệnh khai báo cursor thì tất cả các tùy chọn trong lệnh Fetch(duyệt cursor) được hỗ trợ.

```
declare curnhanvien scroll cursor --Khai báo cursor với tùy chọn scroll
for select * from nhanvien
 open curnhanvien
fetch last from curnhanvien --Lấy dòng cuối cùng trong cursor và set nó làm dòng hiện hành
fetch prior from curnhanvien --Lấy dòng ngay trước dòng hiện hành trong cursor và set nó làm dòng hiện hành
fetch absolute 7 from curnhanvien --Lấy dòng thứ 7 trong cursor và set nó làm dòng hiện hành
```

```
fetch absolute 10 from curnhanvien --Lấy dòng thứ 10 trong cursor và set nó làm dòng hiện hành
fetch relative -3 from curnhanvien --Lấy dòng đứng trước dòng hiện hành 3 dòng trong cursor và set nó làm dòng hiện hành
close curnhanvien
deallocate curnhanvien
```

⇨ Ví dụ 3: Sử dụng biến @@fetch_status điều khiển hoạt động của cursor trong vòng lặp while.

```
declare curnhanvien cursor
for select * from nhanvien
open curnhanvien
fetch next from curnhanvien  --Thực hiện duyệt dòng đầu tiên trong cursor
while @@FETCH_STATUS=0     --Kiểm tra biến @@Fetch_status khi có nhiều hơn 1 hàng để duyệt
begin
fetch next from curnhanvien  --Duyệt dòng kế tiếp
end
close curnhanvien
deallocate curnhanvien
```

⇨ Ví dụ 4: Sử dụng lệnh Fetch (duyệt cursor) để lưu trữ các giá trị vào trong các biến.

```
declare curnhanvien cursor
for select hotennv, sdt from nhanvien
open curnhanvien
declare @hoten varchar(50), @sdt varchar(12) --Khai báo các biến để lưu các giá trị được nhận từ lệnh Fetch
--Thực hiện duyệt lần đầu tiên, lưu các giá trị vào các biến
--Lưu ý: Thứ tự các biến phải giống thứ tự các cột trong câu lệnh select
fetch next from curnhanvien into @hoten , @sdt
while @@FETCH_STATUS=0 --Kiểm tra biến @@Fetch_Status
begin
print @hoten+':'+@sdt --Hiển thị giá trị hiện tại trong biến
```

```
fetch next from curnhanvien into @hoten, @sdt --Duyệt dòng kế tiếp
end
close curnhanvien
deallocate curnhanvien
```

⇨ Ví dụ 5: Sử dụng lệnh Fetch (duyệt cursor) để lưu trữ các giá trị vào trong các biến (tiếp theo). Duyệt cursor và cập nhật MaNhanVien mới cho tất cả các nhân viên trong bảng nhân viên:

```
declare curlistPB cursor
for select MaPhongBan, TenPhongBan from phongban
open curlistPB
declare @MaPhongBan nchar(5), @TenPhongBan nchar(10)
fetch next from curlistPB into @MaPhongBan, @TenPhongBan
while @@FETCH_STATUS=0
begin
        update nhanvien set MaNhanVien=@MaPhongBan + MaNhanVien
        where MaPhongBan = @MaPhongBan
        fetch next from curlistPB into @MaPhongBan, @TenPhongBan
end
close curlistPB
deallocate curlistPB
```

III. NỘI DUNG THỰC HÀNH:

Thực thi file script "Lab4567.sql". Sử dụng CSDL này để thực hiện các yêu cầu sau:

Bài 1: Thực thi các lệnh ở câu a và b. Ghi nhận lại kết quả và cho biết:

Câu a: Cho ý nghĩa từng dòng lệnh.

```
1. declare curnhanvien cursor
   for select * from nhanvien
2. open curnhanvien
3. fetch next from curnhanvien
4. close curnhanvien
5. deallocate curnhanvien
```

Câu b: Cho biết ý nghĩa từng dòng lệnh. Tùy chọn **scroll** đóng vai trò gì trong phần khai báo?

1. declare curnhanvien scroll cursor	6. fetch absolute 4 from curnhanvien
2. for select * from nhanvien	7. fetch absolute 5 from curnhanvien
3. open curnhanvien	8. fetch relative -2 from curnhanvien
4. fetch last from curnhanvien	9. close curnhanvien
5. fetch prior from curnhanvien	10. deallocate curnhanvien

Câu c: Hãy sửa lại câu b theo các yêu cầu như sau:

- Đổi vị trí 2 dòng lệnh 4&5 cho nhau.
- Dòng lệnh số 6: thay số 4 bằng số -2
- Dòng lệnh số 7: thay số 5 bằng số -5
- Dòng lệnh số 8: thay số -2 bằng số 2

Thực thi Câu c, hãy cho biết kết quả thay đổi như thế nào? Tại sao?

Bài 2: Tham khảo các ví dụ ở phần tóm tắt lý thuyết để làm bài này. (ví dụ 3 để làm câu a, ví dụ 4 để làm câu b)

Câu a: Hãy khai báo một cursor cho câu lệnh select lọc ra các nhân viên thuộc phòng Đào tạo. Dùng câu lệnh Fetch next… để duyệt qua các dòng trong cursor.

Câu b: Hãy khai báo một cursor cho câu lệnh select lọc họtên nhânviên và lương của tất cả nhân viên . Dùng câu lệnh Fetch next… để duyệt qua các dòng trong cursor, và giá trị của từng dòng được lưu vào các biến. Dùng lệnh print để xuất giá trị các biến.

Bài 3: Tham khảo ví dụ 5 ở phần tóm tắt lý thuyết để làm bài này.

Câu a: Hãy khai báo một cursor cho câu Select lọc MaPhongBan và SDT của tất cả các phòng ban. Dùng câu lệnh Fetch next… để duyệt qua các dòng trong cursor, và cập nhật sdt mới cho từng phòng ban với thông tin SDT mới = '083' + SDT cũ.

Câu b: Hãy khai báo một cursor cho câu select lọc 'Luong' của tất cả các record từ bảng NhanVien. Dùng câu lệnh Fetch next… để duyệt qua các dòng trong cursor, và cập nhật 'Luong' mới cho từng record biết rằng:

+ Nếu là bậc 'CD' thì ThucLanh= Luong * 1.2
+ Nêu là bậc 'DH' thì ThucLanh= Luong * 1.5
+ Nếu là bậc 'ThS' thì ThucLanh=Luong * 1.8
+ Nếu là bậc 'TS' thì ThucLanh=Luong * 2.0

Bài 4: Con trỏ lồng nhau.

Tham khảo ví dụ về cursor lồng nhau trên trang:
http://msdn.microsoft.com/en-us/library/ms180169.aspx.
Hãy sử dụng cursor lồng nhau để in ra danh sách nhân viên của từng phòng ban.

Bài 5: Hãy sử dụng cursor lồng nhau để in ra danh sách các nhân viên có số ngày nghỉ nhiều nhất trong từng tháng của từng phòng ban.

I. MỤC TIÊU:

- Dùng trigger để thay thế cho các trường hợp muốn kiểm tra ràng buộc toàn vẹn kèm theo các thông báo cho người dùng.
- Cú pháp các câu lệnh DML dùng để khai báo các loại trigger.

II. TÓM TẮT LÝ THUYẾT:

1. Trigger là gì?

Trigger được xem là một thủ tục và có thêm một số đặc điểm chuyên biệt như sau:

- Tự động thực hiện khi có thao tác Insert, Delete hoặc Update trên dữ liệu.
- Thường dùng để kiểm tra các ràng buộc toàn vẹn của CSDL hoặc các qui tắc nghiệp vụ.
- Một trigger được định nghĩa trên một bảng, nhưng các xử lý trong trigger có thể sử dụng nhiều bảng khác nhau.

Trong Trigger, các bảng dữ liệu tạm Inserted và Deleted có ý nghĩa như sau:

- Bảng Inserted: chứa các dòng dữ liệu mới từ bên ngoài được đưa vào cơ sở dữ liệu thông qua việc thực hiện lệnh: Insert/Update/Delete.
- Bảng Deleted: chứa các dòng dữ liệu cũ vừa mới bị xóa ra khỏi bảng trong cơ sở dữ liệu thông qua các lệnh: Update/ Delete.
- Inserted và Deleted là các bảng trong bộ nhớ chính:
  - Cục bộ cho mỗi Trigger
  - Có cấu trúc giống như Table mà trigger định nghĩa trên nó.
  - Chỉ tồn tại trong thời gian trigger đang xử lý.
- Nếu thao tác insert/ update/ delete thực hiện trên nhiều dòng, trigger cũng chỉ được gọi một lần → Bảng Inserted/Deleted có thể chứa nhiều dòng.

2. Khai báo Trigger với câu lệnh T-SQL:

❖ **Lệnh Create Trigger**

```
CREATE TRIGGER <tên triger>
ON <tên bảng>|<tên view>
<FOR | AFTER| INSTEAD OF>  <INSERT , UPDATE] , DELETE>
AS
<câu lệnh SQL>
```

❖ **Insert Trigger (Trigger chèn):**

+ Tự động được thực hiện mỗi khi record mới được chèn vào bảng gắng với nó
+ Một bảng tạm Inserted sẽ được sinh ra.
+ Record cần chèn sẽ được ghi vào bảng Inserted và bảng cơ sở.

Ví dụ:

```
CREATE TRIGGER trInsNV
ON NHANVIEN
FOR INSERT
AS
print ('đã thêm mới 1 nhân viên')
```

+ Mỗi khi thêm mới một record vào bảng NhanVien trigger này sẽ tự động thực hiện xuất ra câu thông báo 'đã thêm mới 1 nhân viên'.

❖ **Delete Trigger (Trigger xóa)**

Ví dụ:

```
CREATE TRIGGER trDelNV
ON NHANVIEN
FOR DELETE
AS
        print ('đã xóa 1 hàng')
```

+ Mỗi khi xóa một record ra khỏi bảng NhanVien trigger này sẽ tự động thực hiện xuất ra câu thông báo 'đã xóa 1 hàng'.

❖ **Update Trigger (Trigger cập nhật)**

Ví dụ:

```
CREATE TRIGGER trUpNV
ON NHANVIEN
FOR UPDATE
AS
        print ('đã sửa 1 hàng')
```

+ Mỗi khi sửa một record trong bảng NhanVien trigger này sẽ tự động thực hiện xuất ra câu thông báo 'đã sửa 1 hàng'.

❖ **Xoá một trigger:**

```
DROP TRIGGER <tên trigger>
```

❖ **Sửa nội dung bên trong của trigger:**

```
ALTER TRIGGER <tên trigger>
<nội dung mới của trigger>
```

❖ **Làm cho Trigger mất hiệu lực:**

```
ALTER TABLE <tên table>
DISABLE TRIGGER <tên trigger>
```

III. NỘI DUNG THỰC HÀNH

Thực thi file script "Lab4567.sql". Sử dụng CSDL này để thực hiện các yêu cầu sau:

Bài 1: Thực thi các câu lệnh bên dưới, nhận diện từng loại Trigger và cách thức để Trigger thực thi.

a. Tạo Insert Trigger trên bảng Nhân Viên

```
CREATE TRIGGER trInsNV
ON NHANVIEN
FOR INSERT
AS
        print ('đã thêm mới 1 nhân viên')
```

+ Thực thi Trigger (1 lần)
+ Viết câu lệnh thao tác dữ liệu (Insert, Delete, Update) tương ứng với từng Trigger để nhận diện Trigger được gọi mỗi lần thao tác dữ liệu

```
insert into NhanVien values ('NV11', N'Lý Văn Tài', '02/28/1980', 2500000, 'PB2','CD','344433666');
```

Kết quả

```
đã thêm mới 1 nhân viên
(1 row(s) affected)
```

Yêu cầu:

b. Tạo Delete Trigger trên bảng Nhân Viên, khi xóa một record thì xuất ra câu thông báo 'đã xóa 1 hàng'. Thực thi Trigger. Viết câu lệnh Delete một record trong bảng Nhân Viên. Cho biết kết quả?

c. Tạo Update Trigger trên bảng Nhân Viên, khi sửa một record thì xuất ra câu thông báo 'đã sửa 1 hàng'. Thực thi Trigger. Viết câu lệnh Update một record trên bảng Nhân Viên. Cho biết kết quả.

Bài 2: Viết Trigger cho bảng nhan viên, thực hiện sửa mã phòng ban của một hoặc nhiều nhân viên thì in ra:

a. Danh sách các mã nhân viên vừa được sửa thông tin.
b. Danh sách các mã nhân viên vừa được sửa và tên phòng ban mới.
c. Danh sách các mã nhân viên vừa được sửa và tên phòng ban cũ.

d. Danh sách các mã nhân viên vừa được sửa cùng với tên phòng ban mới và tên phòng ban cũ.

Bài 3: Thực hiện công việc sau:

Bước 1: Tạo thêm một cột soluongNV(số lượng nhân viên) trong bảng Phòng Ban.

Bước 2: Viết Trigger trên bảng Nhân Viên, để tự động cập nhật soluongNV trên bảng Phòng Ban mỗi khi thêm hoặc xóa một record trên bảng Nhân Viên.

Bài 4: Khi thực hiện xóa thông tin của một phòng ban bất kỳ, viết Trigger thực hiện xóa các thông tin liên quan đến các nhân viên của phòng ban này.

Bài 5: Viết Trigger khi thêm mới một phòng ban thì kiểm tra xem đã có tên phòng ban trùng với tên phòng ban vừa được thêm hay không? Xử lý 2 trường hợp:

+ Trường hợp 1: Thông báo và vẫn cho Insert.
+ Trường hợp 2: Thông báo và không cho Insert.

-Hết-

Bài 7: Sử dụng T_SQL lập trình cho Trigger(tiếp theo).

I. MỤC TIÊU:

Sử dụng câu lệnh T-SQL lập trình cho TRIGGER:

+ Trên TABLE và VIEW.
+ Trigger lồng.
+ Một số các hàm và lệnh hệ thống được dùng lập trình cho trigger.

II. TÓM TẮT LÝ THUYẾT :

1/ **Cú pháp khai báo của trigger:**

CREATE TRIGGER <***tên trigger***>
ON <***tên bảng/ tên view***>
WITH <***Cài đặt tùy chọn cho trigger***>
FOR | AFTER | INSTEAD OF
<INSERT | UPDATE | DELETE | LOGON… >
AS
< câu lệnh T-SQL >

Chú ý: một TRIGGER chỉ áp dụng cho **một đối tượng duy nhất.**

Diễn giải:

- Mệnh đề **WITH**: Thường dùng để đính kèm một số chức năng cho trigger, ví dụ như lệnh:
  - **With Encryption**: áp dụng để mã hóa trigger, không cho phép người khác xâm nhập và chỉnh sửa nội dung của trigger
  - **With Execute As**: giúp ta linh động trong việc kiểm soát quyền hạn.
- Mệnh đề **FOR | AFTER | INSTEAD OF** có ý nghĩa như sau:
  - **FOR**: mệnh đề FOR có ý nghĩa tương đương với mệnh đề AFTER.
  - **AFTER**: Mệnh đề AFTER chỉ áp dụng trên một table.
  - **INSTEAD OF**: Mệnh đề AFTER chỉ áp dụng trên **một table** hoặc **một view.** Mệnh đề FOR được thay thế với INSTEAD OF.
- Mệnh đề **INSERT | UPDATE | DELETE**: tương ứng với các sự kiện xảy ra cho table, view mà trigger được kích hoạt.
- Mệnh đề **LOGON**: được sử dụng cho các đối tượng khác trong hệ thống như: user…

2/ Một số lệnh hay thường được sử dụng trong lập trình cho trigger :

- Thông báo lỗi trong Trigger : để thông báo lỗi trong trigger ta dùng hàm : ***Raiserror('Chuỗi thông báo lỗi',16,1)***
- Không cho thay đổi dữ liệu (trả về trạng thái ban đầu khi chưa thựa hiện kích hoạt trigger): ***Rollback Tran***
- **@@ Rowcount**: Trả về số dòng bị ảnh hưởng bởi câu lệnh T-SQL ngay trước đó trong Trigger.
- Hàm **Return:** Là lệnh được sử dụng tại nhiều vị trí trong trigger mà tại đó bạn muốn chấm dứt việc thi hành các câu lệnh khác còn lại.

III. NỘI DUNG THỰC HÀNH:

Chạy file script : QuanLyLuong.sql

Bài 1: Chạy đoạn script sau và cho biết chức năng của chúng :

```
use QuanLyLuong
go
create trigger ThemSua_NV
On NhanVien
After Insert, Update
As
Declare @MaNV char(5)
Declare @MaTD char(5), @Luong Float
If not exists (Select * From Deleted)
Begin
        If (select count(*) from Inserted)>1 –- Thay thế: if @@rowcount>1
        Begin
                Raiserror ('Chi them mot mau tin',16,1)
                Rollback Tran
                Return
        End
        Else
        Begin
                Select @MaNV= MaNhanVien, @MaTD = MaTD From Inserted
                If not exists (Select * from TrinhDo Where MaTD= @maTD)
                Begin
                        Raiserror ('Khong co Trinh Do nay',16,1)
                        Rollback tran
```

```
                    Return
                End
                Else
                Begin
                    select @luong = LuongCB From TrinhDo
                    where MaTD = @MaTD
                    Update nhanvien Set luong= @luong
                    where manhanvien=@maNV and MaTD = @MaTD
                    Return
                End
        End
End
Else
        BEGIN
        If Update(MaNhanVien)
        Begin
                raiserror ('Khong cap nhat ma nhan vien',16,1)
                rollback tran
                return
        End
        If Update(MaTD)
        Begin
                Select @MaNV= MaNhanVien, @MaTD = MaTD From Inserted
                Select @luong = LuongCB From TrinhDo Where MaTD = @MaTD
                Update nhanvien Set luong= @luong where manhanvien=@maNV
                return
        End
END
```

Bài 2: Chạy đoạn script sau và cho biết chức năng của chúng :

```
Create TRIGGER KiemTraLuong ON  nhanvien
after insert, update
AS
declare @Luong float
```

```
declare @ma char(5)
BEGIN
If not exists (Select * From Deleted)
Begin
	If (select count(*) from Inserted)>1
		-- hoặc: if @@rowcount>1
	Begin
		Raiserror ('Chi them mot mau tin',16,1)
		Rollback Tran
		Return
	End
end
Else
	select @ma=manhanvien ,@Luong = Luong from Inserted
	if cast( cast(@luong/1000 as int)*1000 as float) < @Luong
		begin
			raiserror('du lieu sai' ,16,1)
			rollback tran
			return
		end
	else
	begin
		update NhanVien
		set Luong= @luong
		where MaNhanVien =@ma
		return
	end
END
GO
```

Bài 3: Chạy đoạn script sau và cho biết chức năng của chúng :

```
-- Đoạn code này sử dụng cho phiên ban từ SQL Server 2008 trở đi
Use master;
Go
Create login Thudb with password = '123456' must_change,
   check_expiration = on
Go
Grant view server state to Thudb;
Go
Create trigger connection_limit
On all server with execute as 'Thudb'
For logon
As
Begin
If ORIGINAL_LOGIN()= 'Thudb' AND
   (Select Count(*) From sys.dm_exec_sessions
         Where is_user_process = 1 AND
            original_login_name = 'Thudb') > 3
   Rollback
End
```

Bài 4: Viết trigger thực hiện công việc kiểm tra có các trường hợp sau:

+ Nếu người dùng cập nhật lại số ngày công của các tháng trước tháng hiện tại(tính theo giờ hệ thống) thì báo lỗi "Không cho phép cập nhật ngày công tháng cũ đã tính lương".
+ Nếu tháng cập nhật là tháng hiện tại nhưng số ngày công < 0, thì xuất ra câu thông báo "Số ngày công <0, không hợp lệ".
+ Ngược lại, nếu người dùng cập nhật lại số ngày công theo đúng tháng hiện tại và >0 thì thực hiện tính lại bảng lương cho nhân viên có số ngày bị cập nhật mới.

Bài 5: Viết Trigger thực hiện tính lại số tiền đã mượn tạm ứng trong một bảng lương khi xảy ra cập nhật số tiền hoàn trả của một nhân viên và trước khi cập nhật hãy kiểm tra các trường hợp sau:

+ Nếu số tiền hoàn trả <=0 thì trả về thông báo "Số tiền không hợp lệ"
+ Nếu số tiền >0 nhưng có số có phần lẽ <1000đ thì thông báo "Nhập sai kết quả".

+ Ngược lại, nếu số tiền hoàn trả hợp lệ hoàn toàn thì: thực hiện cập nhật lại số tiền đã mượn tạm ứng.
+ Công thức tính: TamUng(mới) = TamUng(cũ) – HoanTra

Bài 6: Khi muốn xóa một nhân viên, người dùng thường nhập vào mã nhân viên. Hãy viết trigger cho việc xóa nhân viên này, Nhưng trước khi xóa hãy kiểm tra các trường hợp sau:

+ Nếu mã nhân viên cần xóa không có trong danh sách thì báo cho người dùng biết.
+ Nếu mã nhân viên cần xóa có trong bảng nhân viên thì hãy kiểm tra xem nhân viên này có bảng lương hay chưa, nếu chưa thì xóa, còn ngược lại đã có bảng lương rồi thì không được xóa nữa và thông báo anh này đã có bảng lương.

Bài 7: Khi thực hiện cập nhật lương mới cho nhân viên, thì vẫn thực hiên cập nhật bình thường. Tuy nhiên thực hiện thêm việc kiểm tra hai việc sau:

+ Nếu lương mới có thấp hơn so với lương cũ hay không nếu có thì thông báo cho người dùng rõ.
+ Thực hiện tính lại bảng lương tháng hiện tại cho nhân viên được cập nhật lương mới.

IV. BÀI TẬP LÀM THÊM:

Bài 1: Viết trigger mà khi người ta thực hiện xóa tên một phòng ban thì phải thực hiện kiểm tra và hoàn tất các công việc sau:

+ Kiểm tra nếu phòng ban bị xóa không có nhân viên nào thì thực hiện xoá bình thường.
+ Nếu Phòng ban bị xóa có tên nhân viên thì phải thực hiện xóa các nhân viên trong phòng ban này xong, rồi thực hiện tiếp việc xóa phòng ban(kiểm tra ràng buộc toàn vẹn liên kết khóa ngoại). Khi xóa nhân viên thì kiểm tra hai việc sau:
  - Nếu các nhân viên này đã có bảng chấm công (hoặc đã có bảng lương) thì không được xóa nhân viên và thông báo đã có nhân viên đang làm việc, và cũng không xóa phòng ban.
  - Nếu các nhân viên này chưa có bảng chấm công thì thực hiện xóa nhân viên một cách bình thường.

Bài 2: Viết trigger mà khi người dùng thực hiện sửa lương cơ bản ở bảng trình độ thì tất cả các bảng lương ở tháng hiện tại phải được tính lại với lương cơ bản mới vừa được cập nhật mới. Nhưng trước khi cập nhật và tính lại lương hãy kiểm tra các trường hợp sau:

+ Nếu lương cơ bản cập nhật âm thì không cho phép cập nhật và thông báo cho người dùng.
+ Nếu lương cơ bản cập nhật >0 nhưng có số có phần lẽ <1000đ thì thông báo "Nhập sai kết quả".
+ Ngược lại, nếu số tiền hoàn trả hợp lệ hoàn toàn thì: thực hiện tính lại bảng lương..

I. MỤC TIÊU:

✓ Thực hiện và kiểm chứng khi thực thi một số giao tác căn bản minh họa tính chất ACID khi hệ thống bị lỗi.

✓ Phương Thức LOCKS and ISOLATION LEVEL: Shared Locks , Exclusive Locks (for [insert| update| delete]) , Update Locks.

✓ DEADLOCK và PRIORITY (Bế tắc và Xử lý ưu tiên).

II. TÓM TẮT LÝ THUYẾT :

1. Giao tác: là một tập lệnh có truy xuất đến cơ sở dữ liệu và có đặc điểm sau :

+ Làm cho dữ liệu từ trạng thái nhất quán này đến trạng thái nhất quán khác.
+ Ban đầu dữ liệu đúng thì sau khi thực hiện giao tác thì dữ liệu vẫn đúng.
+ Để đảm bảo tính toàn vẹn của dữ liệu, khi một giao tác(transations) được thực hiện thì phải đảm bảo tính chất **ACID** bao gồm bốn tính chất :

- Atomicity - Nguyên tử: Đảm bảo hoặc toàn bộ các hoạt động trong giao tác thành công hoặc thất bại
- Consistency - Nhất quán: Khi transaction hoàn thành, dữ liệu phải ở trạng thái nhất quán.
- Isolation - Cô lập: Cho dù có nhiều giao tác có thể thực hiện đồng thời, hệ thống phải đảm bảo rằng sự thực hiện đồng thời đó dẫn đến một trạng thái hệ thống tương đương khi các giao tác thực hiện tuần tự theo một thứ tự nào đó.
- Durability - Bền vững: Sau khi giao tác thành công, giả sử xảy ra sự cố thì tất cả các dữ liệu được thay đổi trong giao tác vẫn được hồi phục lại theo yêu cầu giao dịch:

2. Cú pháp :

```
CREATE PROCEDURE <Tên Thủ tục>
<biến 1, biến 2, … , biến n-1 output, biến n output>
AS
        BEGIN TRAN
        SET TRANSACTION ISOLATION LEVEL <Tên mức cô lập>
        -- Xem thêm đặc điểm và ưu điểm mức cô lập bên dưới.
                <Các lệnh giao tác>
        COMMIT TRAN
```

📖 Một số từ khóa khi viết lệnh T-SQl dùng quản lý giao tác :

➢ **BEGIN TRAN**: Bắt đầu một giao tác. Khi thực hiện runtime thì phải đảm bảo một trong hai cặp: [BEGIN … COMMIT] hoặc [BEGIN … ROLLBACK] được thực hiện.

➢ **SAVE TRAN:** Đánh dấu một vị trí trong giao tác (gọi là điểm đánh dấu).

➢ **ROLLBACK:** Quay lui trở lại đầu giao tác hoặc một điểm đánh dấu trước đó trong giao dịch và không có tác dụng như **RETURN**.

➢ **COMMIT:** Đánh dấu điểm kết thúc một giao dịch. Khi câu lệnh này thực thi cũng có nghĩa là giao tác đã thực hiện thành công.

➢ **RETURN:** Mang ý nghĩa hoàn tất thủ tục.

🕮 **Xác định lỗi:**

➢ Lỗi do hệ thống: Lỗi phát sinh khi thực hiện các câu lệnh Insert, Update, Delete. Hệ thống cung cấp biến **@@error** để xác định lỗi. Biến **@@error** có giá trị là :

- 0 : thực hiện giao tác thành công.
- != 0 : giao tác bị lỗi

➢ Lỗi do người dùng: Lỗi này là do người dùng viết các câu lệnh thực thi bị lỗi, không đem lại kết quả như mong đợi.

3. Một số lỗi khi xảy ra truy xuất đồng thời :

+ **Lost update** - Mất dữ liệu cập nhật: Tình trạng này xảy ra khi có nhiều hơn 1 thao tác cùng thực hiện cập nhật trên 1 đơn vị dữ liệu. Khi đó, tác dụng tác dụng của giao tác cập nhật thực hiện sau sẽ đè lên tác dụng của giao tác thực hiện cập nhật trước.

+ **Uncommit data, Dirty read** - Đọc dữ liệu chưa commit: Xảy ra khi một giao tác thực hiện đọc trên 1 đơn vị dữ liệu mà đơn vị dữ liệu này đang bị cập nhật bởi 1 giao tác khác nhưng việc cập nhật chưa được xác nhận

+ **Unrepeatable data** - Dữ liệu không thể lặp lại: Tình trạng này xảy ra khi một giao tác đang thực hiện đọc trên một đơn vị dữ liệu nhưng giao tác khác lại được cho phép thay đổi(ghi) trên đơn vị dữ liệu này. Điều này làm cho lần đọc sau đó của bản giao tác đầu tiên không còn nhìn thấy dữ liệu ban đầu nữa.

+ **Phantom** - Bóng ma: Là tình trạng mà một giao tác đang thao tác trên một tập dữ liệu nhưng giao tác khác lại chèn thêm hoặc xóa bớt các dòng dữ liệu mà giao tác kia quan tâm.

4. Xử lý tranh chấp đồng thời:

🕮 **Locks** : là cơ cấu cho phép ngăn ngừa các hành động trên đối tượng có thể gây ra xung đột với những gì đã thực hiện và hoàn thành trên đối tượng trước đó. Chế độ **Lock**:

➢ **Share locks (khoá chia sẻ):** Đây là loại căn bản nhất, lock chia sẻ tài nguyên cho phép đọc dữ liệu, không cho phép thay đổi bất kỳ thuộc tính nào của tài nguyên.

➢ **Exclusive locks (khoá độc quyền):** Không tương thích với các loại khoá khác. Khoá này ngăn ngừa 2 người sử dụng cùng cập nhật, xoá, thêm dữ liệu.

+ **Update locks (khoá cập nhật):**•Kết hợp giữa share lock và exclusive lock. Với câu lệnh UPDATE chỉ ra bản ghi bằng mệnh đề WHERE, trong khi chưa cần cập nhật thì sẽ là trạng thái share lock. Khi câu lệnh UPDATE thực hiện ở chế độ Exclusive lock

+ **Intent locks** Dùng giải quyết phân cấp đối tượng. Trong SQL server 2000, intent locks chỉ giải quyết đến bảng chứ không quan tâm đến từng bản ghi trong bảng

+ **Schema locks xuất phát từ hai loại sau:**

- Schema modification locks (Sch-M): giản đồ thay đổi cách tạo đối tượng, không yêu cầu các phát biểu CREATE, ALTER, hay DROP.

- Schema stability lock (Sch-S): tương tự như Share lock, lock này ngăn ngừa các yêu cầu của các phát biểu CREATE, ALTER, DROP khi đã thiết lập Schema Modification Lock.

📖 **Deadlock là gì?**

+ Một hệ thống ở trạnh thái deadlock nếu tồn tại một tập hợp các giao tác sao cho mỗi giao tác trong tập hợp đang chờ một giao tác khác trong tập hợp.

+ Có hai phương pháp chính giải quyết vấn đề deadlock: Ngăn ngừa deadlock, phát hiện deadlock và khôi phục. Giao thức ngăn ngừa deadlock đảm bảo rằng hệ thống sẽ không bao giờ đi vào trạng thái deadlock.

❖ **Các mức độ cô lập:**

➢ ***Read Uncommitted:***

• Đặc điểm:

+ Không thiết lập Share Lock trên những dữ liệu cần đọc

+ Không bị ảnh hưởng bởi những lock của các giao tác khác trên những dữ liệu cần đọc

+ Không phải chờ khi đọc dữ liệu (kể cả khi dữ liệu đang bị lock bởi giao tác khác)

• Ưu điểm:

+ Tốc độ xử lý nhanh

+ Không cản trở những giao tác khác thực hiện việc cập nhật dữ liệu.

+ Khuyết điểm:

+ Có khả năng xảy ra mọi vấn đề trong việc xử lý đồng thời, đặc biệt là vấn đề Dirty Reads

➢ ***Read Committed***

• Đặc điểm:

+ Đây là mức độ cô lập mặc định của SQL Server

+ Tạo Shared Lock trên dữ liệu được đọc, Shared Lock được giải phóng ngay sau khi đọc xong dữ liệu

- Tạo Exclusive Lock trên đơn vị dữ liệu được ghi, Exclusive Lock được giữ cho đến hết thao tác.

- Ưu điểm:

+ Giải quyết được vấn đề Dirty Reads

+ Share Lock được giải phóng ngay, không giữ đến hết thao tác nên không ngăn cản được các cập nhật của các giao tác khác

- Khuyết điểm:

+ Chưa giải quyết được vấn đề Unrepeatable Reads, Phantoms, Lost Update

+ Phải chờ khi chưa thể được đáp ứng trên yêu cầu lock trên dữ liệu đang bị giữ lock bởi thao tác khác

➢ ***Repeatable Read***

- Đặc điểm:

+ = Read Committed + giải quyết Unrepeatable Reads

+ Tạo Share Lock trên dữ liệu được đọc, Share Lock được giữ cho đến hết thao tác. Không cho phép các giao tác khác cập nhật thay đổi giá trị trên dữ liệu này.

+ Tạo Exclusive Lock trên dữ liệu được ghi, Exclusive Lock được giữ cho đến hết giao tác.

- Ưu điểm: Giải quyết được Dirty Reads, Unrepeatable Reads, Lost Update.

- Khuyết điểm:

+ Chưa giải quyết được vấn đề Phantoms.

+ Phải chờ khi chưa thể đáp ứng yêu cầu lock trên dữ liệu dang bị giữ lock bởi thao tác khác.

+ Các giao tác khác không được phép cập nhật trên những giữ liệu đang bị shared Lock.

+ Vẫn cho phép Insert những dòng dữ liệu thỏa mãn điều kiện thiết lập những Shared Lock Phantoms.

➢ ***Serializable:***

- Đặc điểm:

+ Serializable = Repeatable Read + giải quyết Phantoms.

+ Tạo Shared Lock trên dữ liệu được đọc, Shared Lock được giữ cho đến hết giao tác. Không cho phép các giao tác khác được cập nhật, thay đổi giá trị trên dữ liệu này.

+ Không cho Insert dữ liệu thõa mãn điều kiện thiết lập Shared Lock (sử dụng khóa trên một miền giá trị -Key Range Lock)

+ Tạo Exclusive Lock trên đơn vị dữ liệu được ghi, Exclusive Lock được giữ cho đến hết giao tác.

- Ưu điểm:

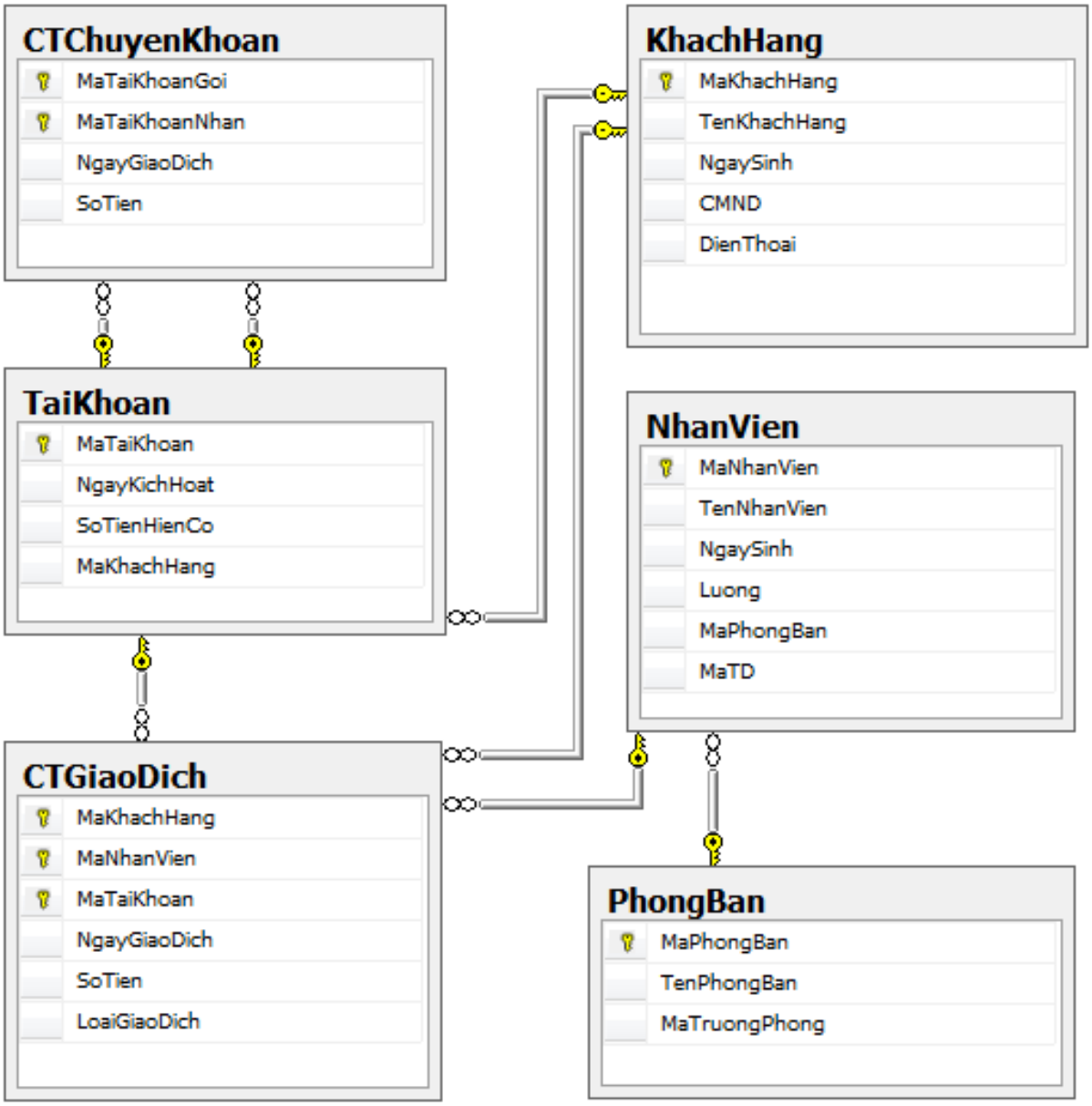
+ Giải quyết được vấn đề Dirty Reads, Unrepeatable Reads, Phantoms.

- Khuyết điểm:

+ Phải chờ khi chưa thể đáp ứng yêu cầu lock trên đơn vị dữ liệu đang bị giữ lock giao tác khác

III. NỘI DUNG THỰC HÀNH:

+ Chạy file script : QuanLyATM.sql
+ Relationship :



Bài 8: Quản lý Transactions

Bài 1: **Thực thi thủ tục. Cho tình huống như sau :**

Thực hiện việc chuyển tiền từ @TK1(mã tài khoản 1) sang @TK2(mã tài khoản 2) một số tiền @SoTien thì ta thực hiện các bước như sau :

B1 : Đọc số tiền hiện có của tài khoản 1 và đưa vào @SoDu1

B2 : Cập nhật số tiền hiện có của tài khoản 1 = @SoDu1 - @SoTien

B3 : Đọc số tiền hiện có của tài khoản 2 và đưa vào @SoDu2, Cập nhật số tiền hiện có của tài khoản 1 = @SoDu2 + @SoTien

B4 : Thông báo thành công.

+ Nhận xét tiến trình trên : Nếu Bước 2 thành công mà bước 3 thất bại thì @TK1 và @TK2 có số tiền không đồng nhất => bị lỗi.

+ Mong muốn : Bước 2 và bước 3 phải được thực hiện đồng bộ nếu không thì không có bước nào được thực hiện.

Thủ tục thực hiện chuyển tiền như sau :

```
Alter procedure ChuyenTien
@TK1 char(20),
@TK2 char(20),
@SoTien money
as
        declare @SoTien1 money,@SoTien2 money
        BEGIN TRAN
        SET TRANSACTION ISOLATION LEVEL SERIALIZABLE
        if (@TK1  in(select MaTaiKhoan from TaiKhoan where MaTaiKhoan= @TK1))
and
(@TK2  in(select MaTaiKhoan from TaiKhoan where MaTaiKhoan= @TK2))
        begin
                waitfor delay '00:00:05'
                set @SoTien1 = (select SoTienHienCo
From TaiKhoan
where MaTaiKhoan=@TK1)
                set @SoTien2 = (select SoTienHienCo
From TaiKhoan
where MaTaiKhoan=@TK2)
                if (@SoTien <=@SoTien1 )
```

```
            begin
                  update TaiKhoan
                  set SoTienHienCo= @SoTien1 - @SoTien
                  where MaTaiKhoan= @TK1
                  update TaiKhoan
                  set SoTienHienCo= @SoTien2 + @SoTien
                  where MaTaiKhoan= @TK2
                  print 'Thực hiện thành công'
            end
            else
            begin
                  print 'Số tiền chuyển quá lớn so với thực tế!'
            end
      end
      else
      begin
            print 'thong tin khong hop le!'
      end
      COMMIT TRAN
```

```
exec  ChuyenTien '12345678910','12345678911', 1190000000
Select *From TaiKhoan
```

Trả lời câu hỏi :

Sinh viên hãy thực thi thủ tục trên và kiểm tra các trường hợp có thể xảy ra và trả lời câu hỏi sau : thủ tục trên có đáp ứng được tình huống đưa ra không? Sử dụng mức cô lập SERIALIZABLE có đúng hay không?

Bài 2: **Viết các thủ tục xử lý các đụng độ sau :**

STT	T1 – CTGiaoDich rút 900.000.000	T2 – CTGiaoDich rút 1.000.000.000
	@MaTaiKhoan = '12345678902' **@SoTienRut = 900.000.000**	**@MaTaiKhoan = '12345678902'** **@SoTienRut = 1.000.000.000**
1	--B1 : Đọc số tiền hiện có vào biến @SoDu Waitfor delay '0:0:15'	
2		--B1 : Đọc số tiền hiện có vào biến @SoDu
3	--B2 : Nếu @SoDu>@SoTienRut thì tiến hành cập nhật tài khoản và thông báo thành công	
4		--B2 : Nếu @SoDu>=SoTienRut thì tiến hành cập nhật tài khoản và thông báo thành công
5	--B3 : Ngược Lại Nếu : @Sodu<@Sotienrut Thì Thông Báo Không Thành Công.	
6		--B3 : Ngược lại nếu : @SoDu< @SoTienRut thì thông báo không thành công.
Yêu cầu : Giả sử thực hiện các lệnh theo thứ tự : 1->2->3->4->5->6. Hoặc thực hiện lại lệnh với thứ tự : 1->2->4->3->6->5. Viết hàm xử lý đụng độ trên. ***Gợi ý :*** Giao tác thực hiện ghi sau ghi đè lên dữ liệu của giao tác thực hiện việc ghi trước, làm mất dữ liệu cập nhật		

Bài 3: **Cho bối cảnh đụng độ sau :**

STT	T1 – CTGiaoDich tra cứu số dư	T2 – CTGiaoDich Nạp tiền 1.000.000.000
	@MaTaiKhoan = '12345678902'	**@MaTaiKhoan = '12345678902'** **@SoTienGoi = 1.000.000.000**
1	--B1 : Đọc số tiền hiện có vào biến @SoDu Waitfor delay '0:0:15'	
2		--B1 : Đọc số tiền hiện có vào biến @SoDu
3		--B2 : Thực hiện lệnh cập nhật lại SoTienHienCo = @SoDu + @SoTienGoi
4	--B2 : Đọc số tiền hiện có vào biến @SoDu Waitfor delay '0:0:15'	
5		--B3 : Đọc số tiền hiện có vào biến @SoDu Waitfor delay '0:0:15'
Yêu cầu : Giả sử thực hiện các lệnh theo thứ tự : 1->2->3->4->5->6. Viết hàm xử lý đụng độ trên. ***Gợi ý :*** 2 lần đọc A của T1 có kết quả khác nhau,dẫn đến không đọc lại được dữ liệu cập nhật.		

-Hết-

I. MỤC TIÊU:

Nhập, xuất dữ liệu. Sao lưu, phục hồi dữ liệu, đính kèm dữ liệu, và biên dịch file script cho một cơ sở dữ liệu.

II. TÓM TẮT LÝ THUYẾT :

1/ Cú pháp thực hiện Back up Database và Backup Transaction Log:

```
USE <Tên Database>
GO

BACKUP DATABASE <Tên Database> -- Back up Database
TO DISK = 'Đường dân đến file.bak'
WITH
DESCRIPTION= 'Thông báo'
GO

BACKUP LOG <Tên Database> -- Backup Transaction Log
TO DISK = 'Đường dân đến file .TRN'
WITH
DESCRIPTION= 'Thông báo'
GO
```

2/ Cú pháp thực hiện Restore database :

```
RESTORE DATABASE <Tên Database> -- RESTORE Database
  FROM DISK = 'Đường dân đến file.bak'
  WITH
        RECOVERY -- hoặc NORECOVERY
GO
RESTORE LOG QuanLyNhanVien -- RESTORE Transaction Log
  FROM DISK = 'Đường dân đến file.TRN'
  WITH
       RECOVERY -- hoặc NORECOVERY
GO
```

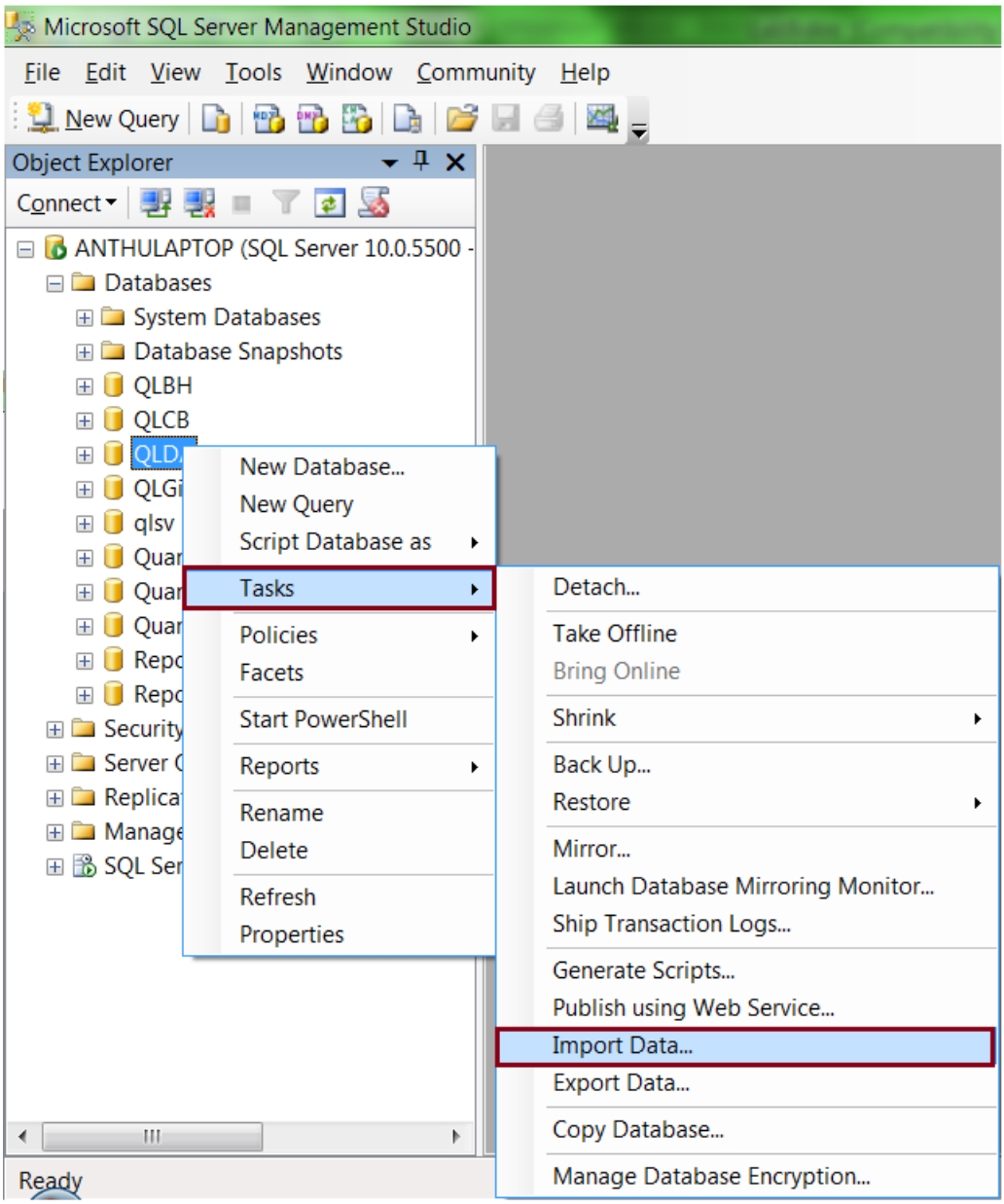
Xem ý nghĩa thông số Recovery hoặc NoRecovery chi tiết tại phần sử dụng công cụ phục hồi dữ liệu.

III. NỘI DUNG THỰC HÀNH:

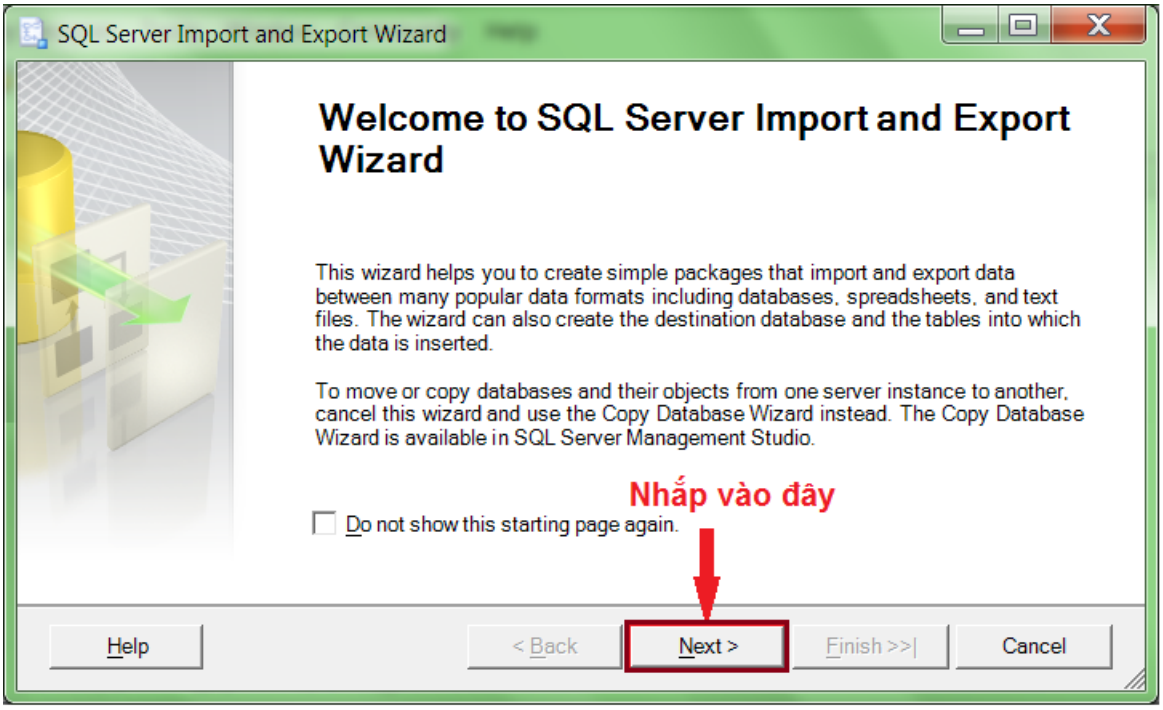
Bài 1: Sử dụng công cụ thực hiện các chức năng sau đây.

1/ Sinh viên thực hiên các bước sau để Import/Export Data vào SQL Server:

Bước 1:

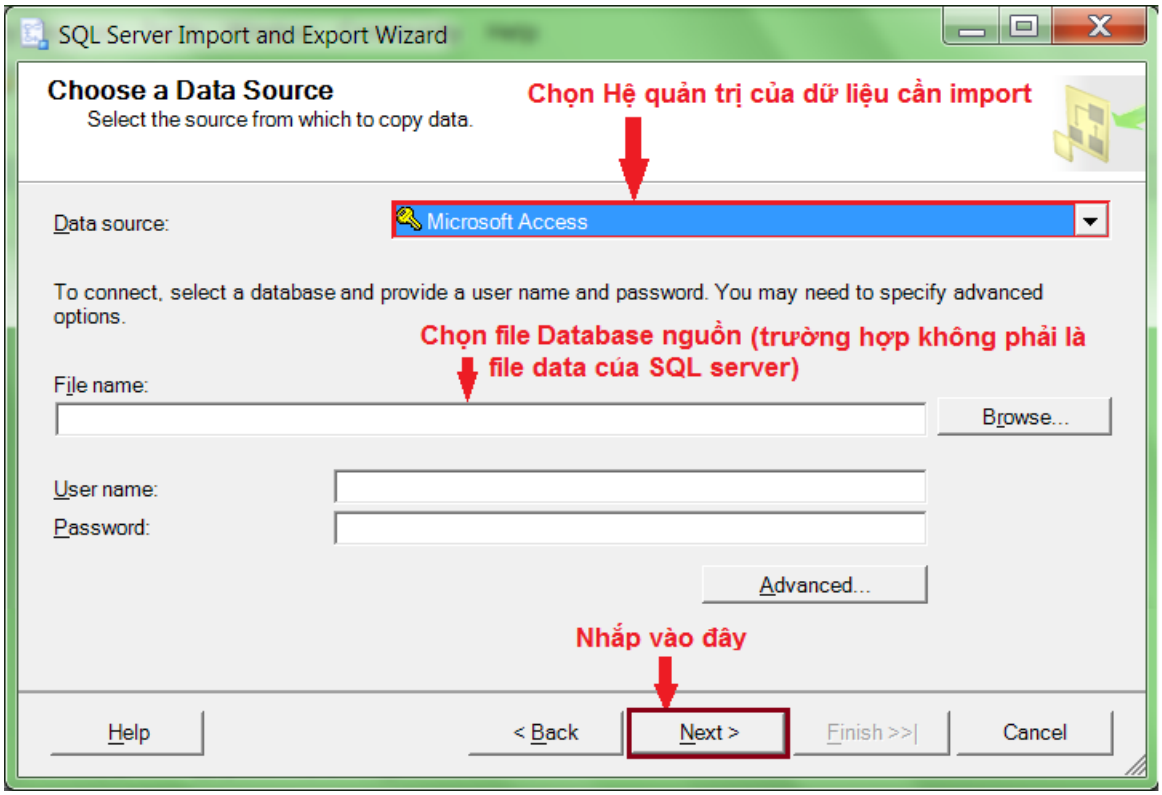


Bước 2:

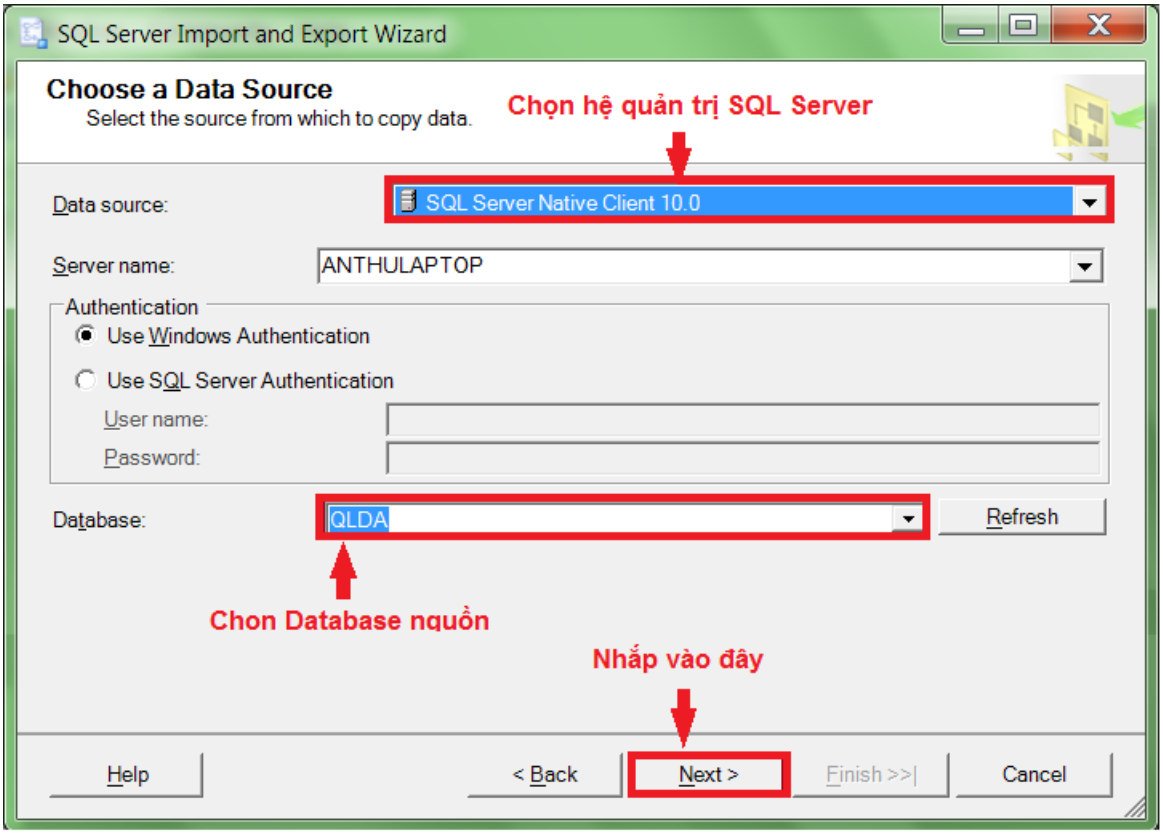


Bước 3:

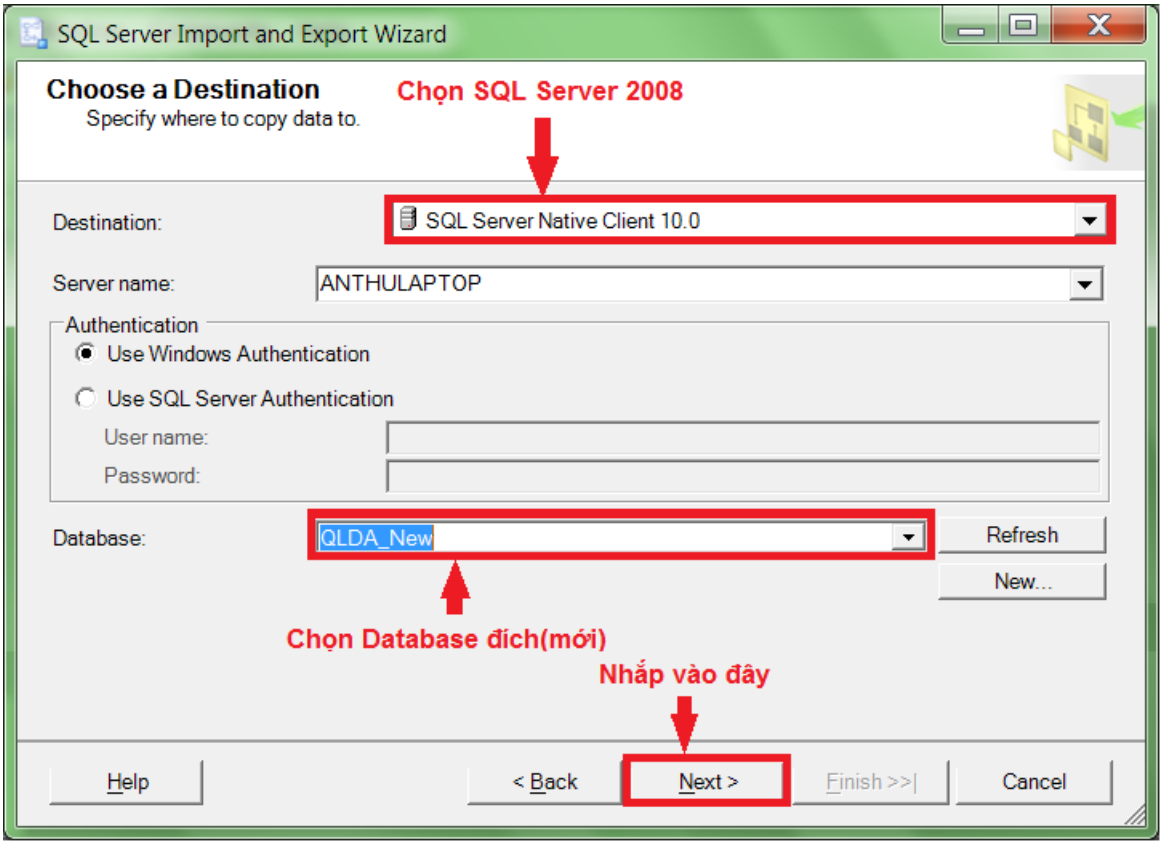
Hộp thoại dành cho trường hợp file dữ liệu gốc **không phải** của SQL Server.



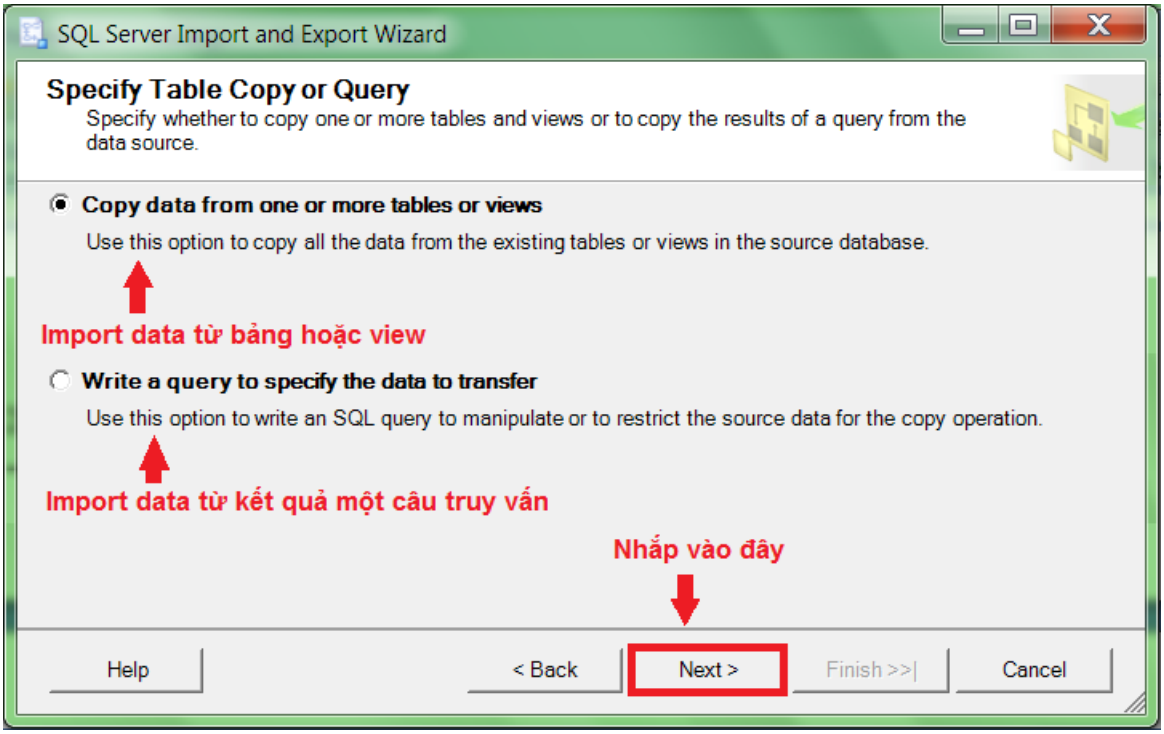
Hộp thoại dành cho trường hợp file dữ liệu gốc là của SQL Server.



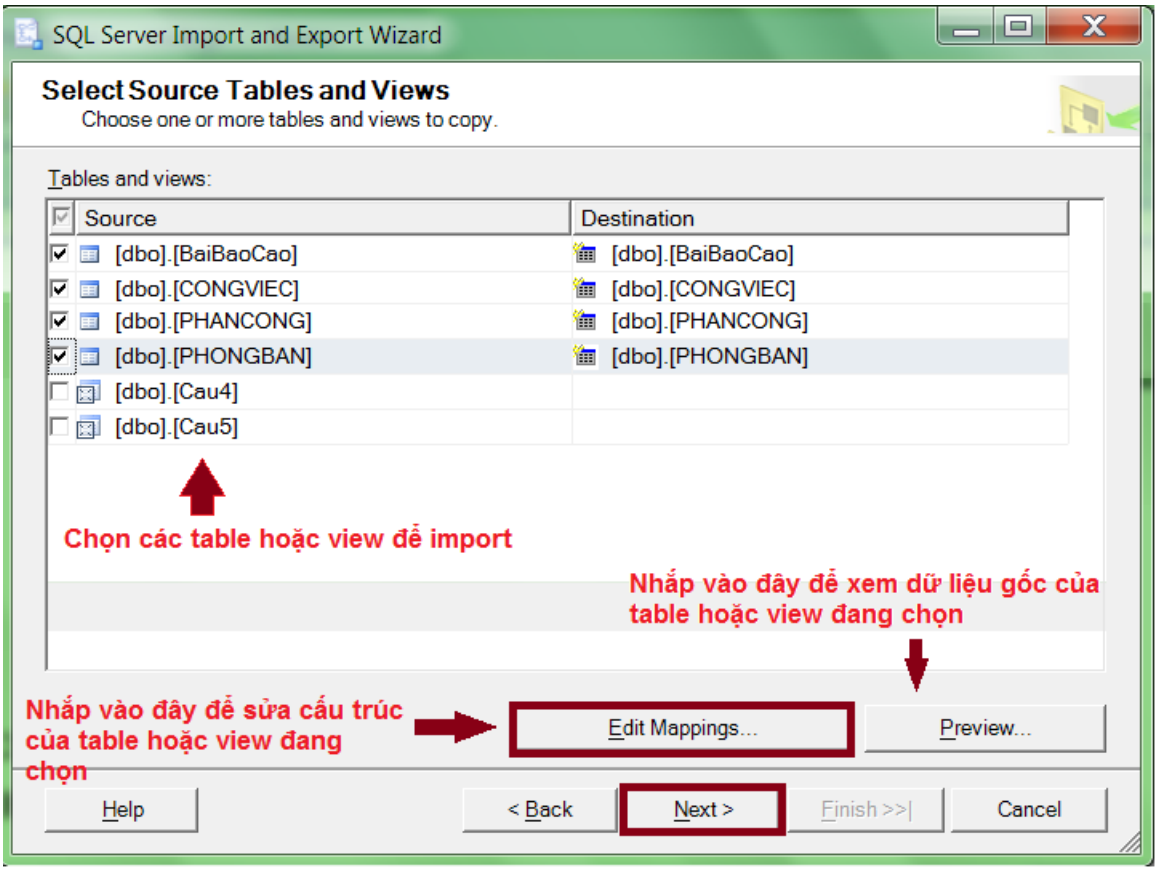
Bước 4:



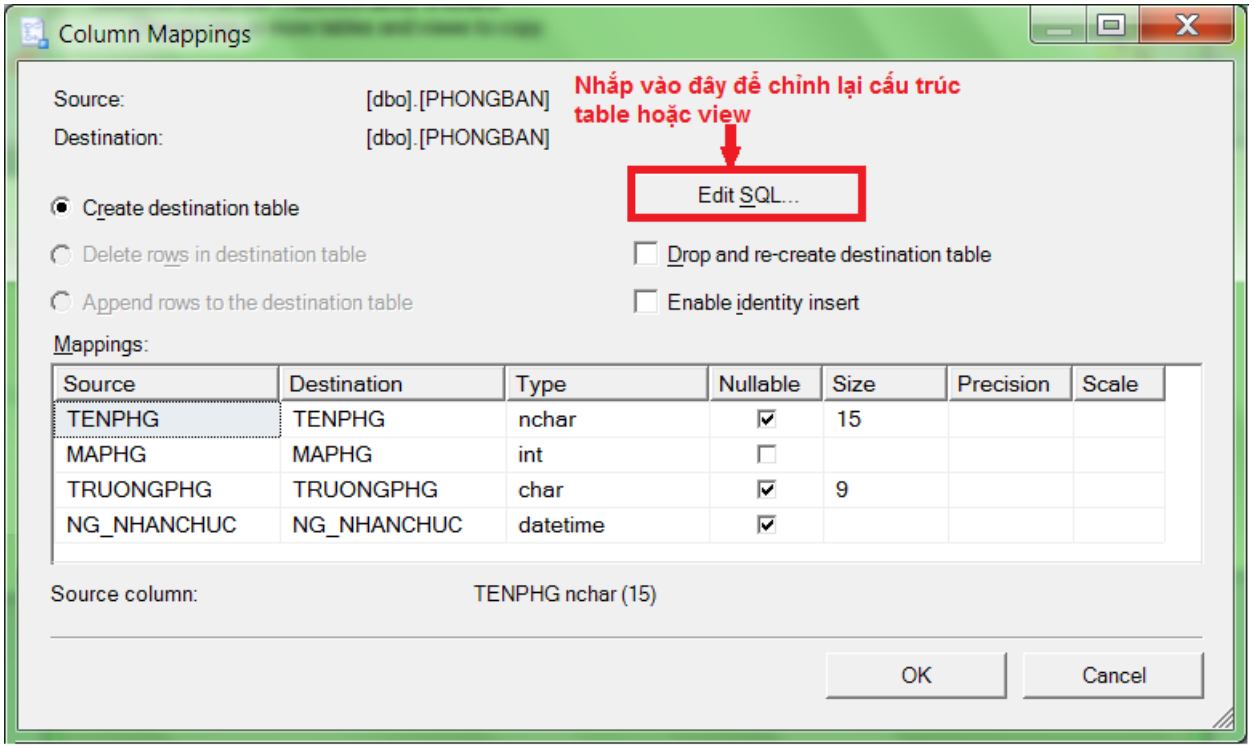
Bước 5:



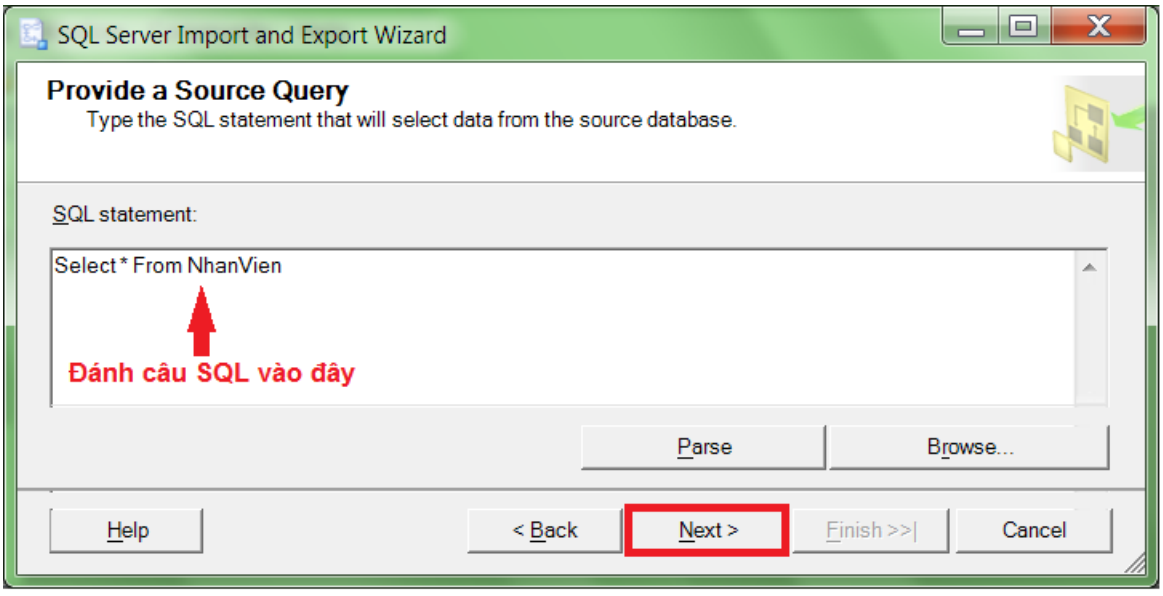
Bước 6: Nếu chọn từ table hoặc view có sẵng trong Database:



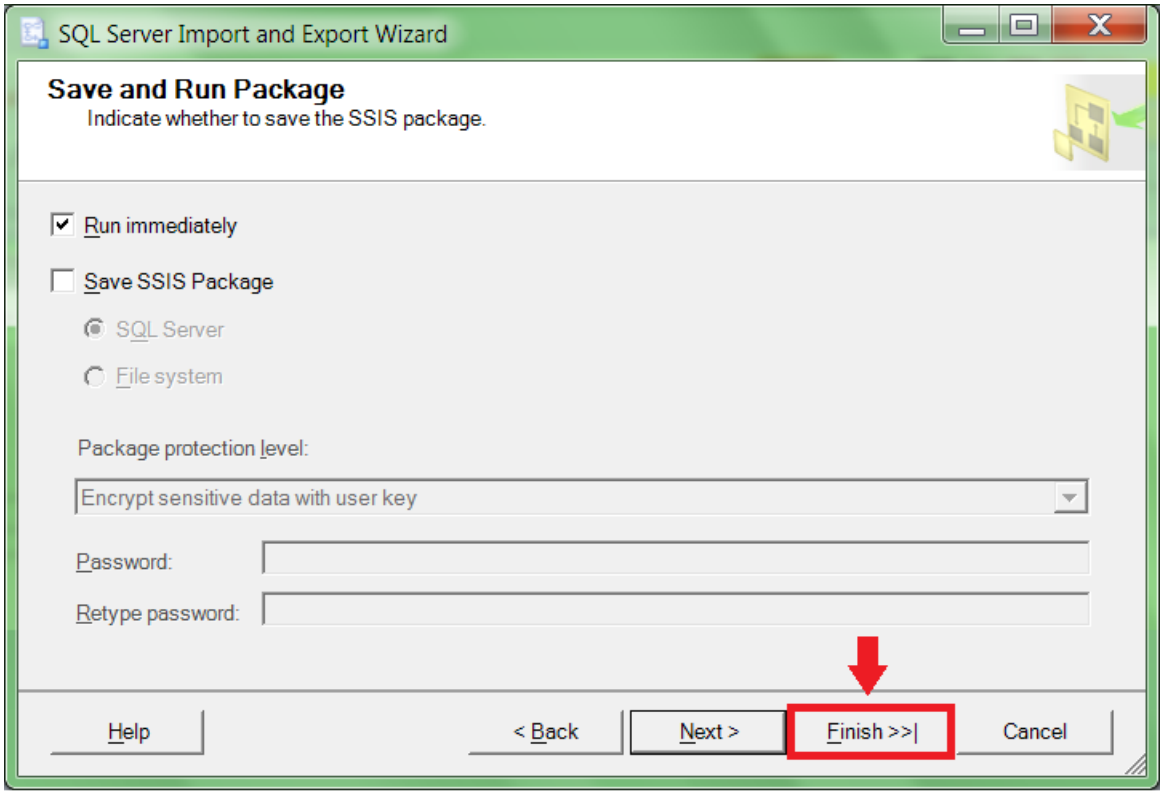
Sau khi chọn Edit Mappings, chúng ta có cửa sổ như sau:



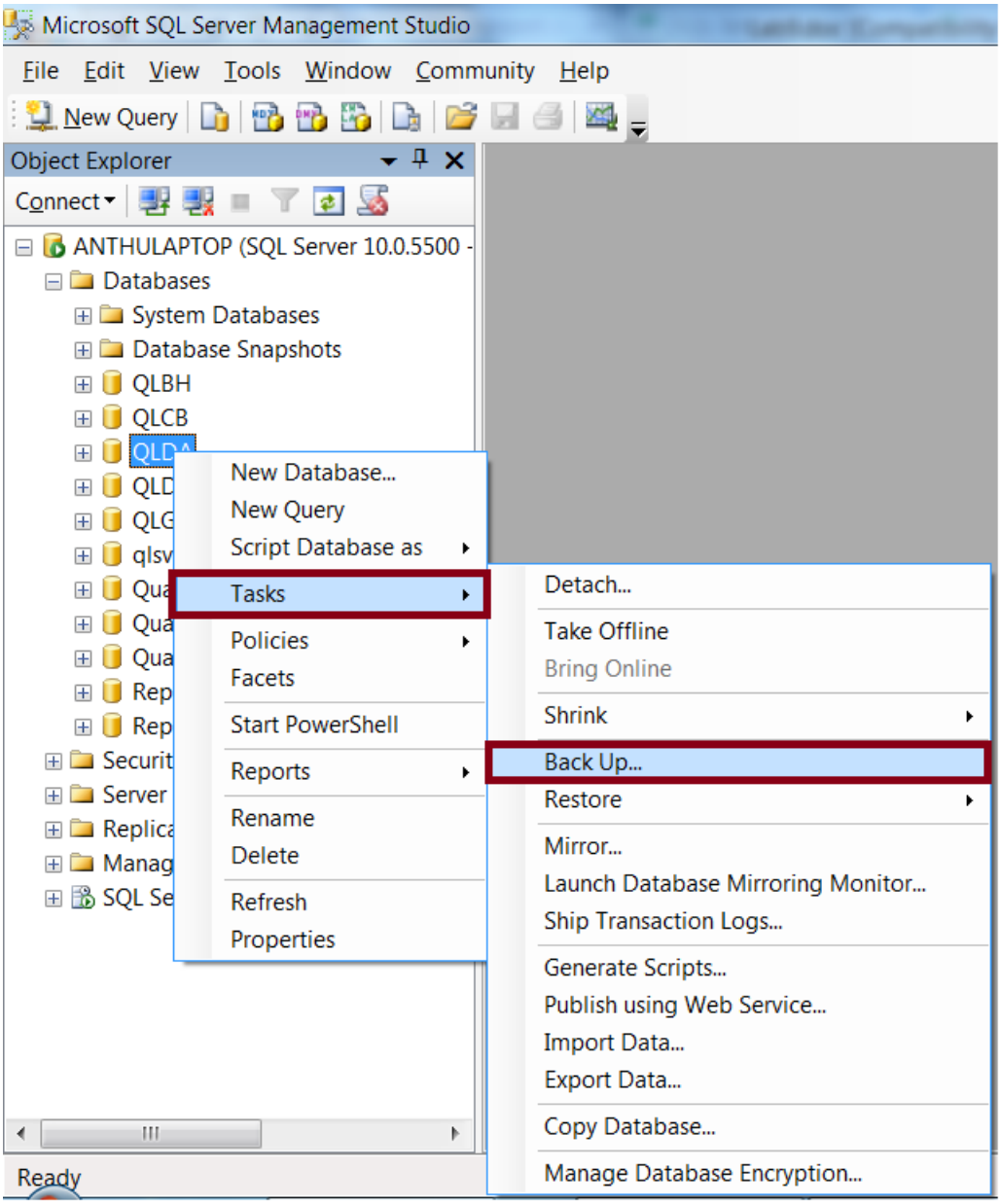
Nếu chọn kết quả từ một câu truy vấn SQL:



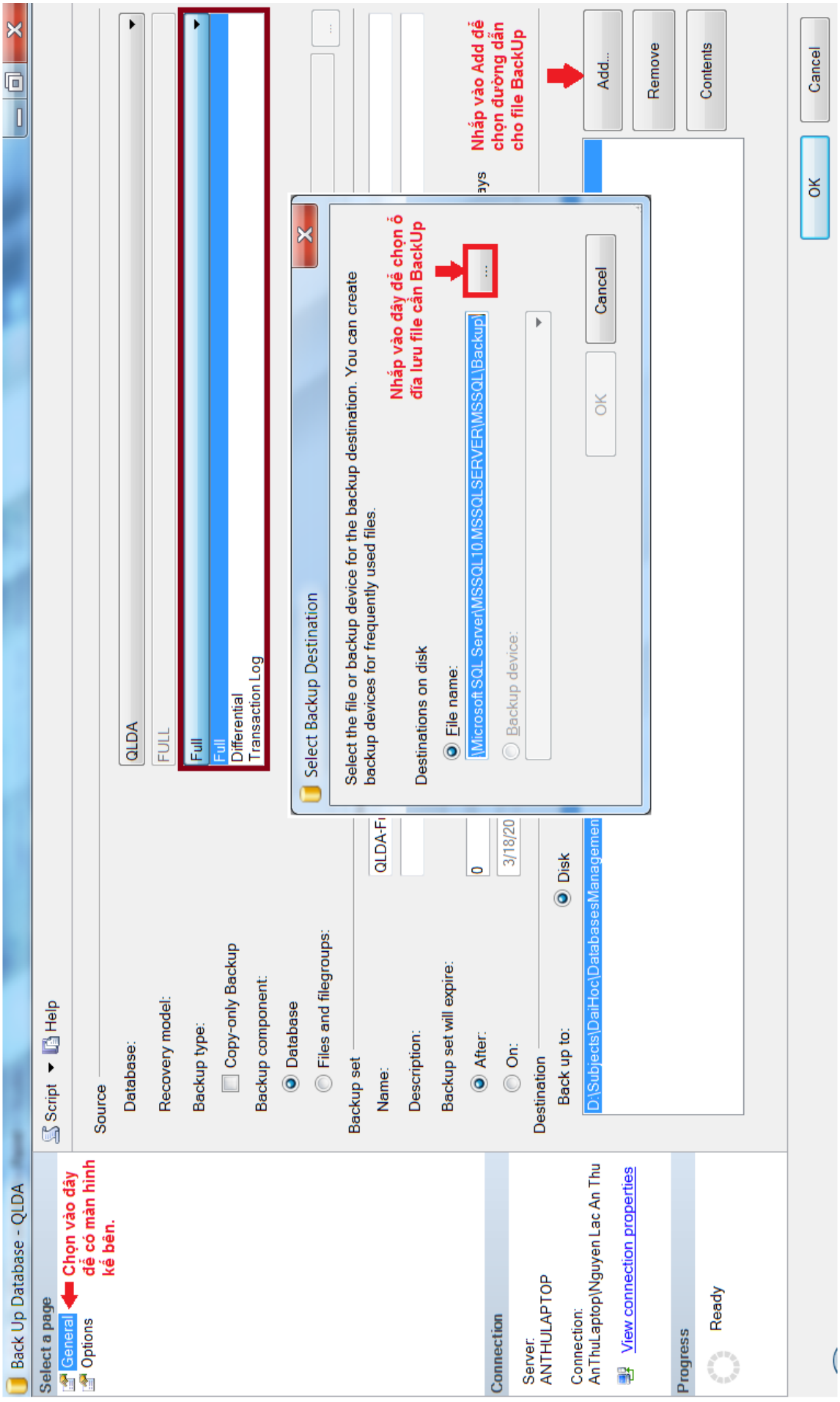
Bước 7: Kết thúc Import/ Export Data…



Bài 2: Back up và Restore Data (Sao lưu và phục hồi dữ liệu):



- Chọn **Back Up** để mở cửa sổ thực hiện các thao tác sao lưu dữ liệu của một Database.


Back Up Database - QLDA
Select a page
General
Options
Chọn vào đây để có màn hình kế bên.
Script
Help
Source
Database:
QLDA
Recovery model:
FULL
Backup type:
Full
Full
Differential
Transaction Log
Copy-only Backup
Backup component:
Database
Files and filegroups:
Backup set
Name:
QLDA-Fi
Description:
Backup set will expire:
After:
0
On:
3/18/20
Destination
Back up to:
Disk
D:\Subjects\DaiHoc\DatabasesManagemen
Add...
Remove
Contents
Connection
Server:
ANTHULAPTOP
Connection:
AnThuLaptop\Nguyen Lac An Thu
View connection properties
Progress
Ready
OK
Cancel
Select Backup Destination
Select the file or backup device for the backup destination. You can create backup devices for frequently used files.
Destinations on disk
File name:
\Microsoft SQL Server\MSSQL10.MSSQLSERVER\MSSQL\Backup\
Backup device:
Nhấp vào đây để chọn ổ đĩa lưu file cần BackUp
Nhấp vào Add để chọn đường dẫn cho file BackUp
OK
Cancel

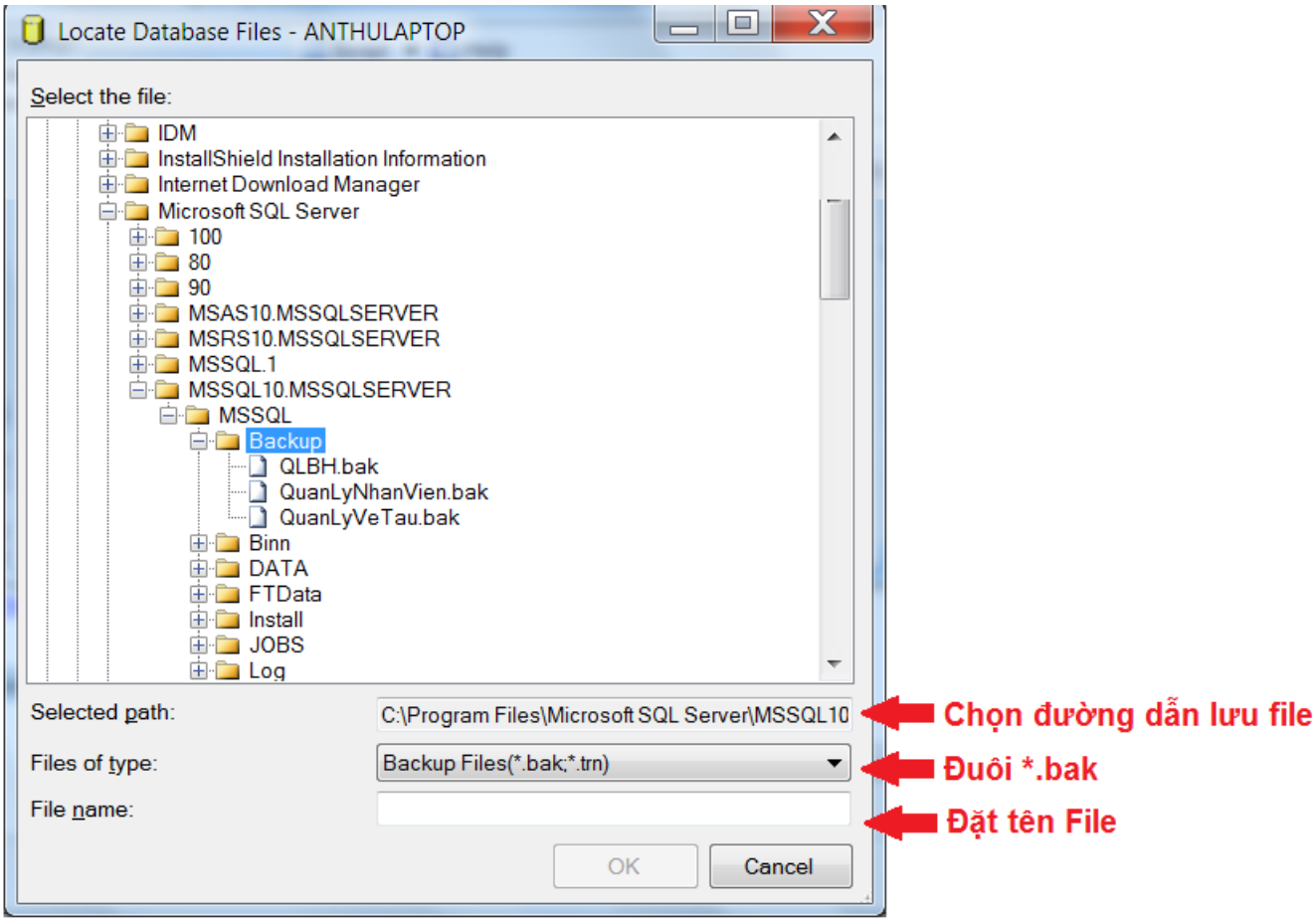
- **Backup Type** có ba trường hợp chính là: Full, Differential, Transaction Log. Ý nghĩa của ba trường hợp này như sau:

+ **Full**: Copy tất cả data files, user data và database objects như system tables, indexes, user-defined tables trong một database.

+ **Defferential**: Copy những thay đổi trong tất cả data files kể từ lần full backup gần nhất.

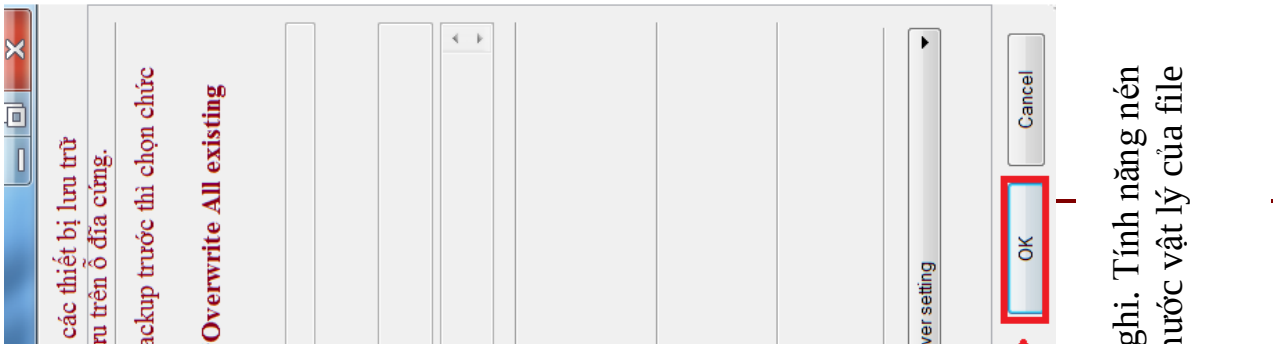
+ **Transaction Log**: Ghi nhận một cách thứ tự tất cả các **transactions** chứa trong **Transaction Log File** kể từ lần Transaction Log Backup gần nhất. Loại Backup này cho phép ta phục hồi dữ liệu trở ngược lại vào một thời điểm nào đó trong quá khứ mà vẫn đảm bảo tính nhất quán.

- Chọn ổ đĩa để lưu file backup các file backup có phần mở rộng là *.bak



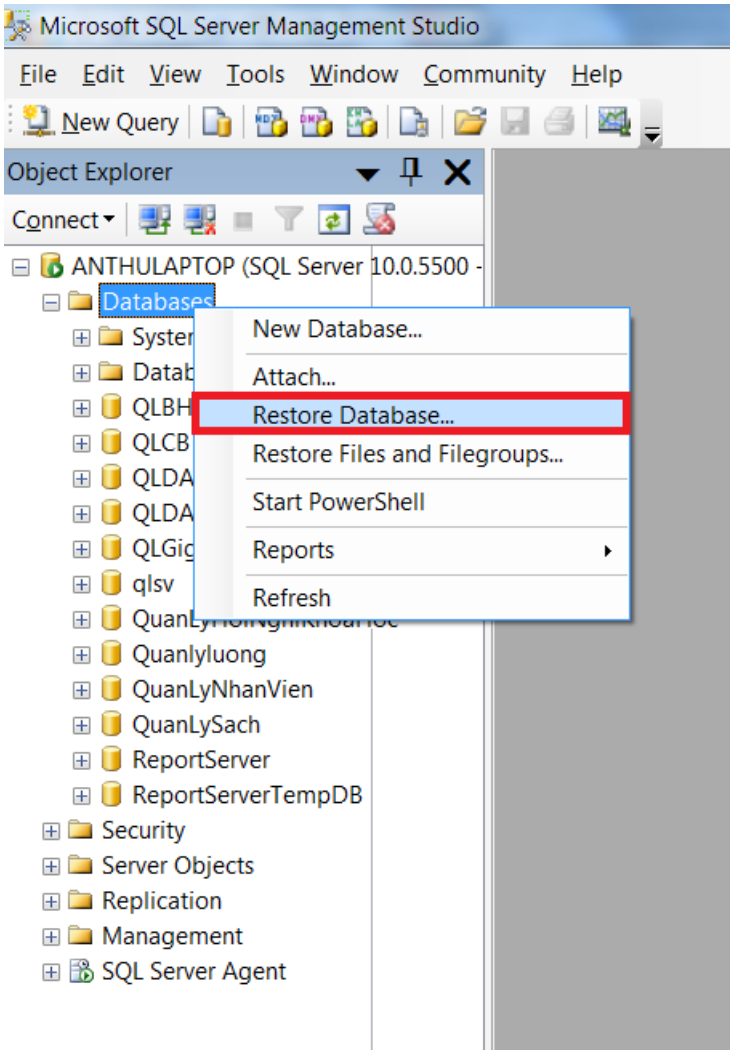
- Thông báo Back up thành công.

- Trong quá trình thao tác sao lưu dữ liệu **trên ổ đĩa cứng** bạn có thể bỏ qua bước này. Tuy nhiên, nếu bạn sao lưu dữ liệu sang các thiết bị ngoại vi khác: băng, đĩa… hoặc cài đặt một số tính năng hỗ trợ cho việc sao lưu thì phải mở trang "**Option**" như hình vẽ kế tiếp:

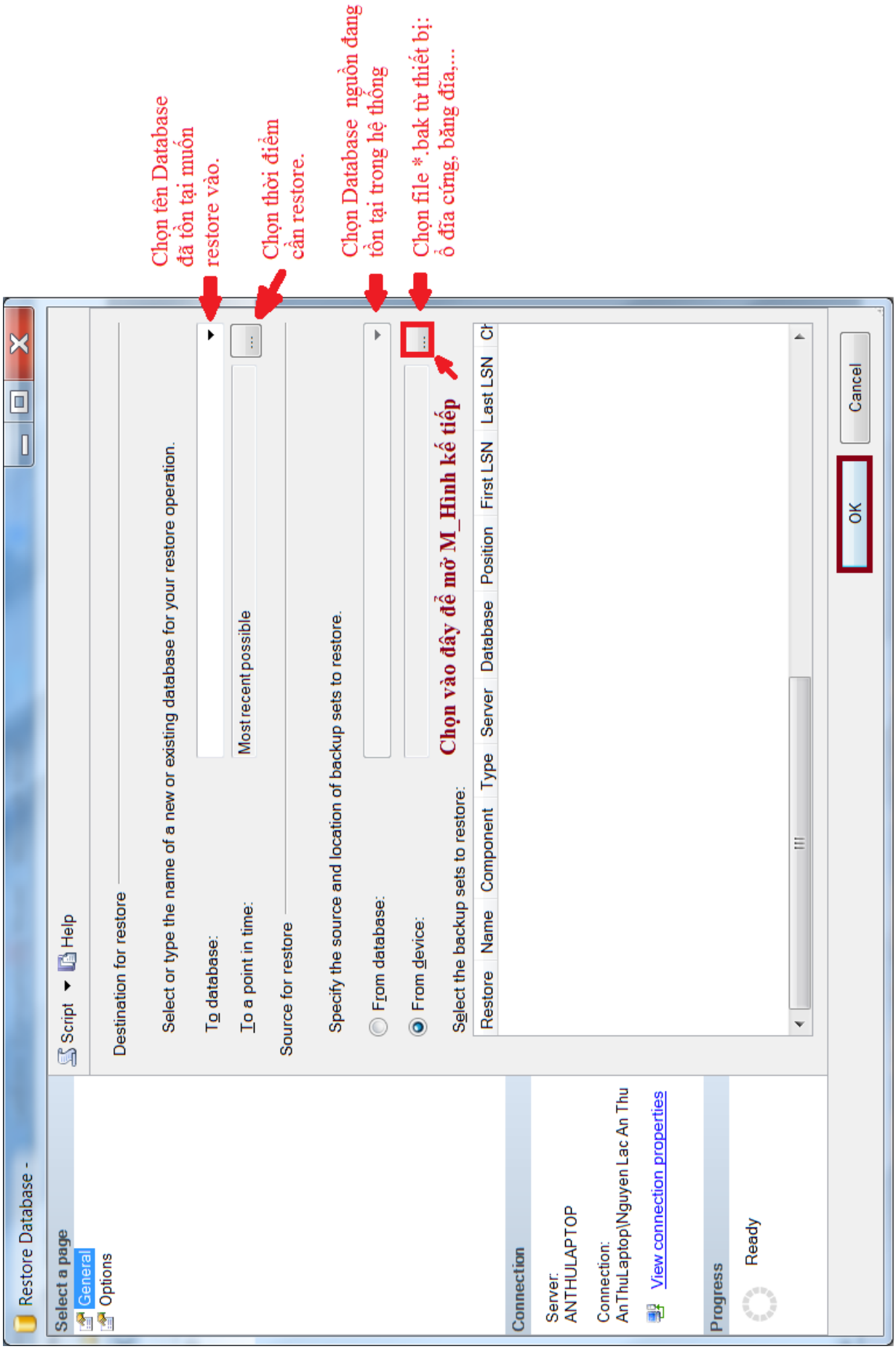


Bài 3: Restore Database:

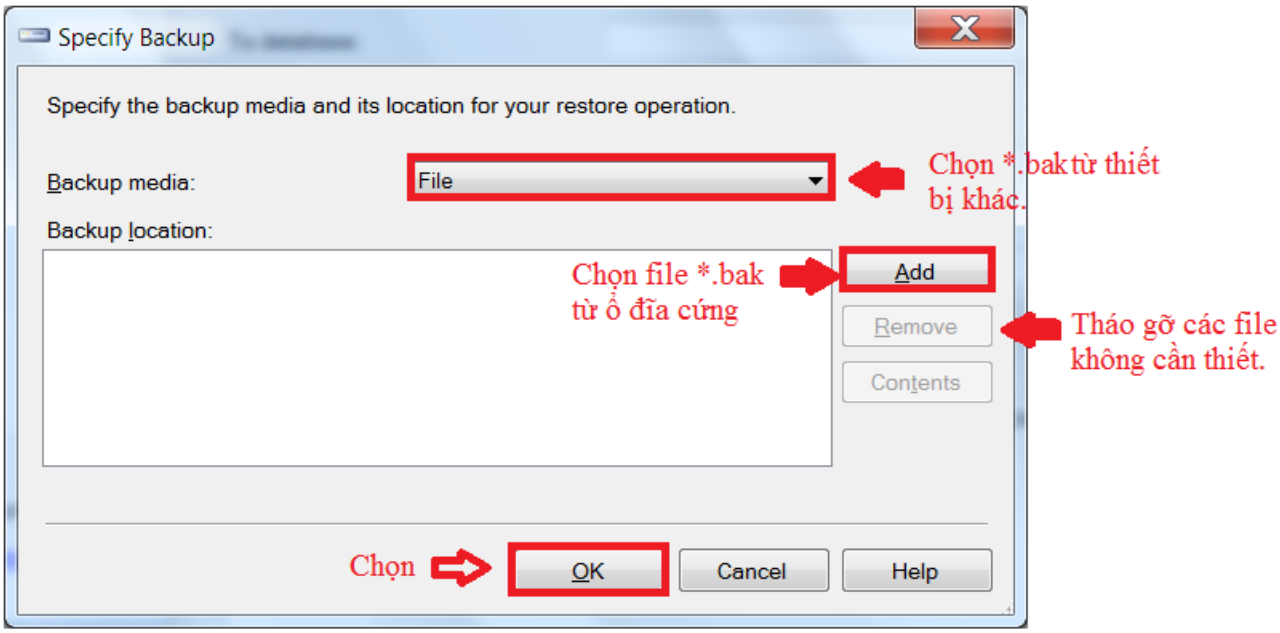
- Nhắp phải vào thư mục Database -> chọn Restore Database



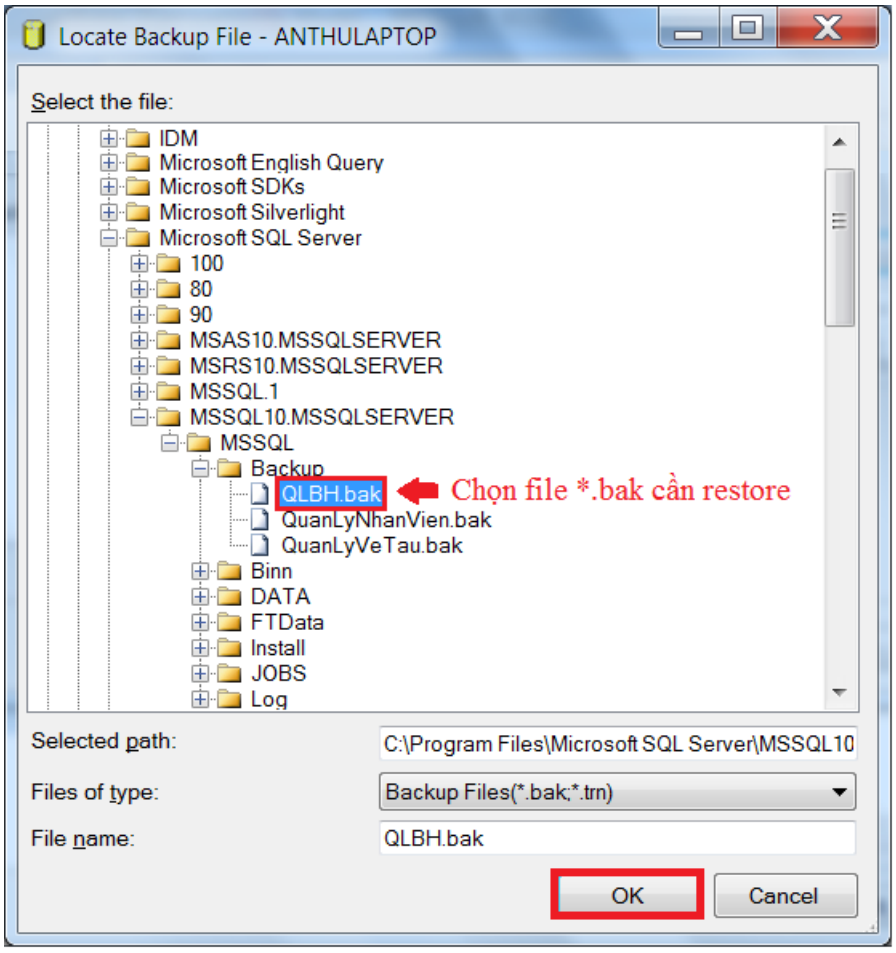
- Chọn tập tin nguồn dạng file *.bak, đã được Back up vào ổ cứng


Chọn tên Database đã tồn tại muốn restore vào.
Chọn thời điểm cần restore.
Chọn Database nguồn đang tồn tại trong hệ thống
Chọn file *.bak từ thiết bị: ổ đĩa cứng, băng đĩa,...
Chọn vào đây để mở M_Hình kế tiếp
Restore Database -
Select a page
General
Options
Script
Help
Destination for restore
Select or type the name of a new or existing database for your restore operation.
To database:
To a point in time:
Most recent possible
Source for restore
Specify the source and location of backup sets to restore.
From database:
From device:
Select the backup sets to restore:
Restore
Name
Component
Type
Server
Database
Position
First LSN
Last LSN
Connection
Server:
ANTHULAPTOP
Connection:
AnThuLaptop\Nguyen Lac An Thu
View connection properties
Progress
Ready
OK
Cancel

- Mở hộp thoại chọn file *.bak từ ổ cứng như sau:

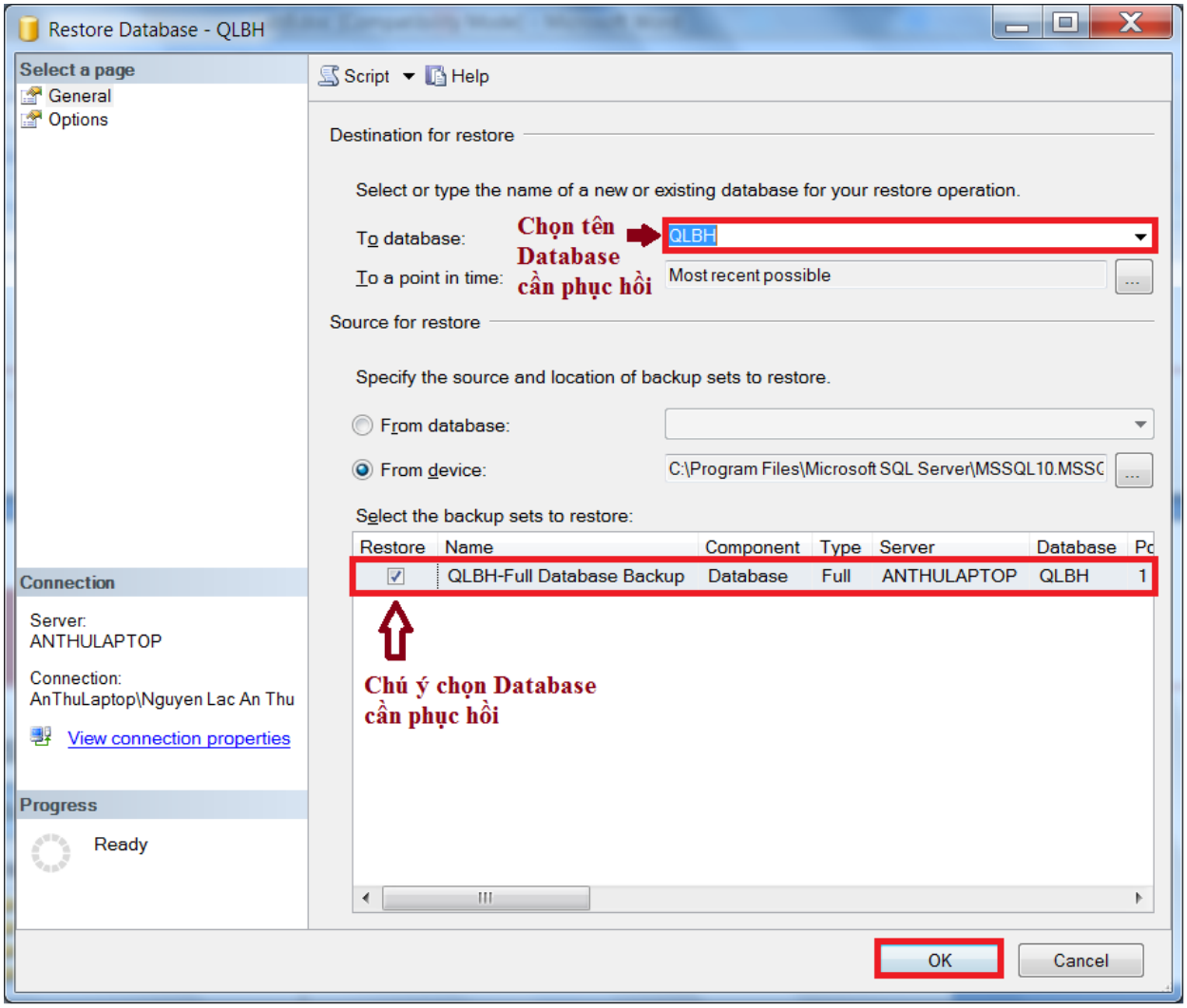


- Chọn file cần Restore được lưu trong ổ đĩa cứng:

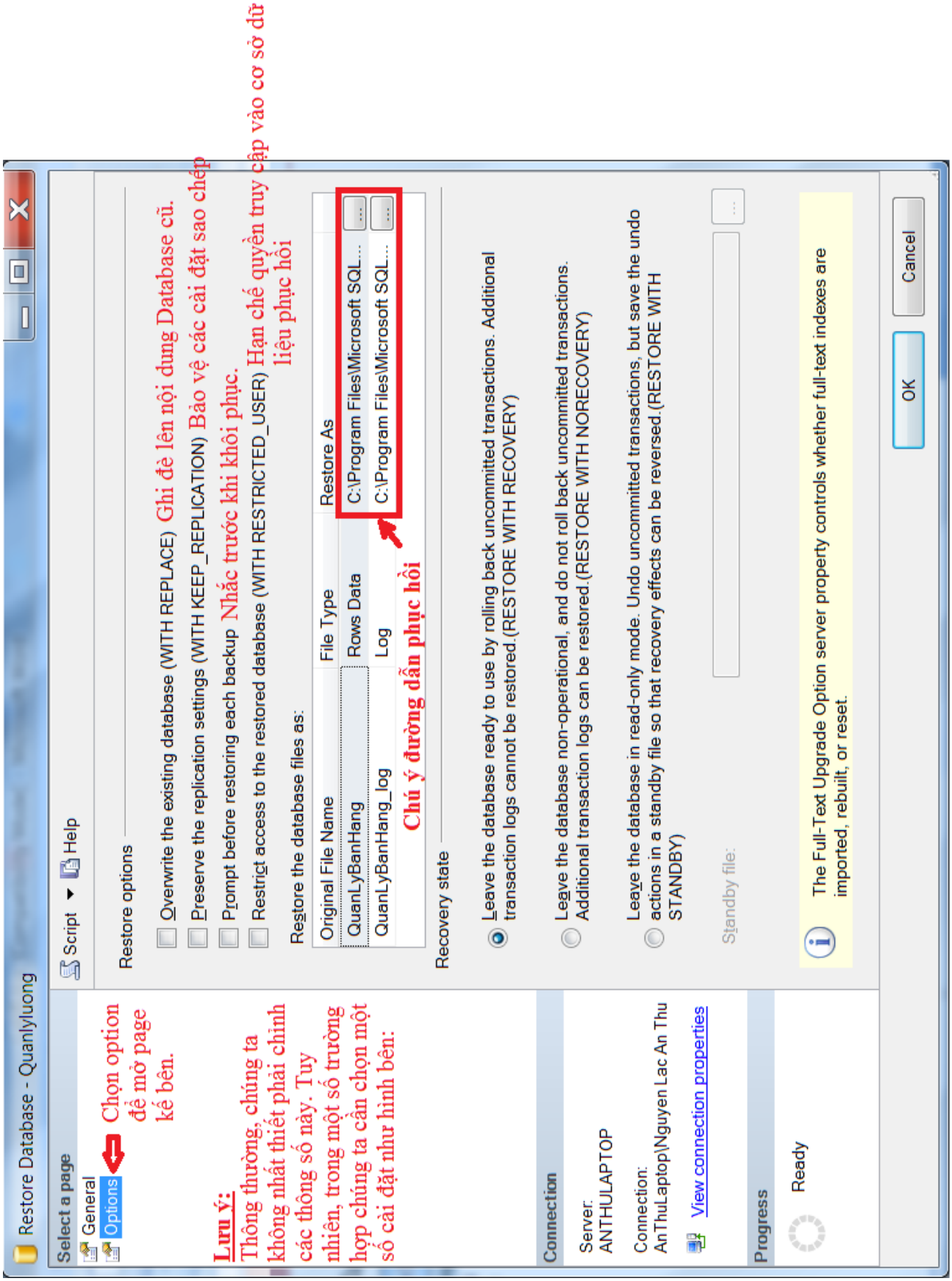


-

- Xem lại kết quả sau khi chọn:



- Nếu muốn cài đặt khác bạn nhấp vào page Option như hình bên dưới đây:



- Khi phục hồi dữ liệu, hệ thống sẽ làm hai công việc là: copy data, log, index... từ ở đĩa sao lưu vào Data Files và sau đó sẽ lần lượt thực thi các Transaction trong Transaction Log. Do đó, chúng ta cần chú ý một số tùy chọn của các chức năng Recovery stale trong quá trình phục hồi như sau:

❖ Leave the database ready to use by rolling back the uncommitted transactions. Additional transaction logs cannot be restored. (RESTORE WITH RECOVERY).

⇨ Ý nghĩa của tùy chọn này là: các giao dịch(TRANSACTION) chưa hoàn tất(INCOMPLETE TRANSACTION) sẽ được phục hồi về trạng thái trươc khi xảy ra giao dịnh(ROLL BACK) và Database ở trạng thái hợp lệ (CONSISTENT) nhưng ta không thể nào phục hồi các Transaction Log của File Backup được nữa.

❖ Leave the database non-operational, and do not roll back the uncommitted transactions. Additional transaction logs can be restored. (RESTORE WITH NORECOVERY)

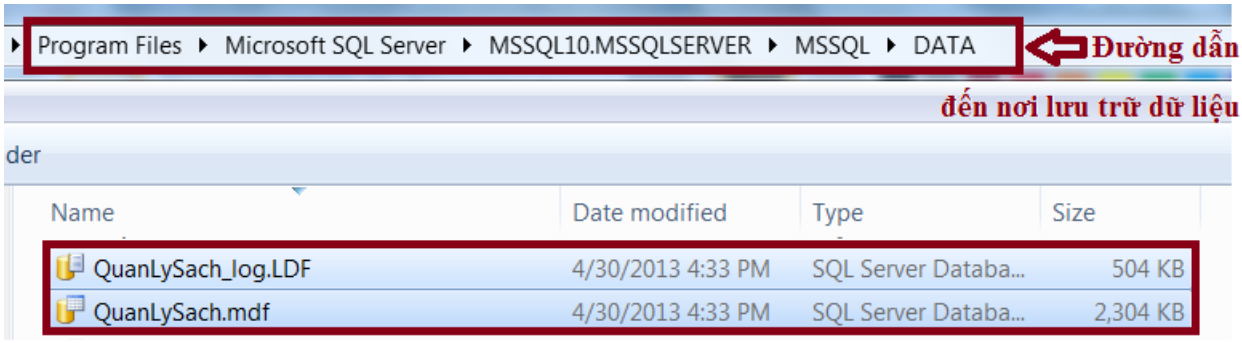
⇨ Ý nghĩa của tùy chọn này là: Nghĩa là các Transaction chưa hoàn tất sẽ không được Roll Back. Như vậy Database lúc này sẽ ở trong tình trạng chưa hợp lệ và không thể dùng được.

❖ Leave the database in read-only mode. Undo uncommitted transactions, but save the undo actions in a standby file so that recovery effects can be reverted. (RESTORE WITH STANDBY)

⇨ Ý nghĩa của tùy chọn này là: để dung hòa hai sự chọn lựa trên bạn có thể chọn chức năng này, chức năng này cho phép các các Transaction chưa hoàn tất sẽ được Roll Back để đảm bảo Database hợp lệ và có thể sử dụng được nhưng chỉ dưới dạng Read-only mà thôi, đồng thời sau đó bạn có thể tiếp tục phục hồi các File Backup còn lại.

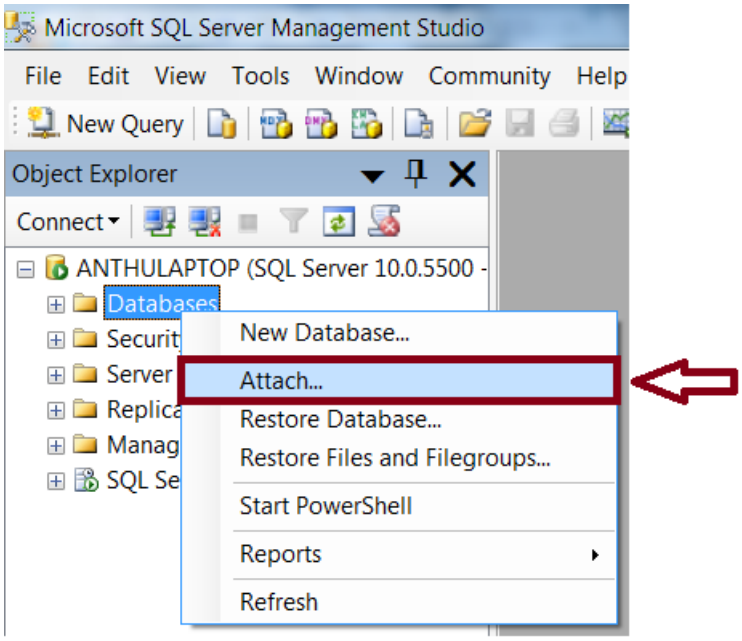
Bài 4: **Attach Database:** Database trong SQL được lưu trữ bao gồm hai file: ***.mdf(Primary Database File)** và ***.ldf(Log Data File)**. Chính vì vậy, ngoài việc chúng ta Back Up hoặc Restore dữ liệu như ở trên, chúng ta còn có thể lưu trữ hai file này và khi cần sử dụng lại chúng ta chỉ cần sử dụng chức năng **Attach Database** để sử dụng dữ liệu.

❖ Hai file trên thường được lưu trữ trong thư mục như hình vẽ bên dưới:

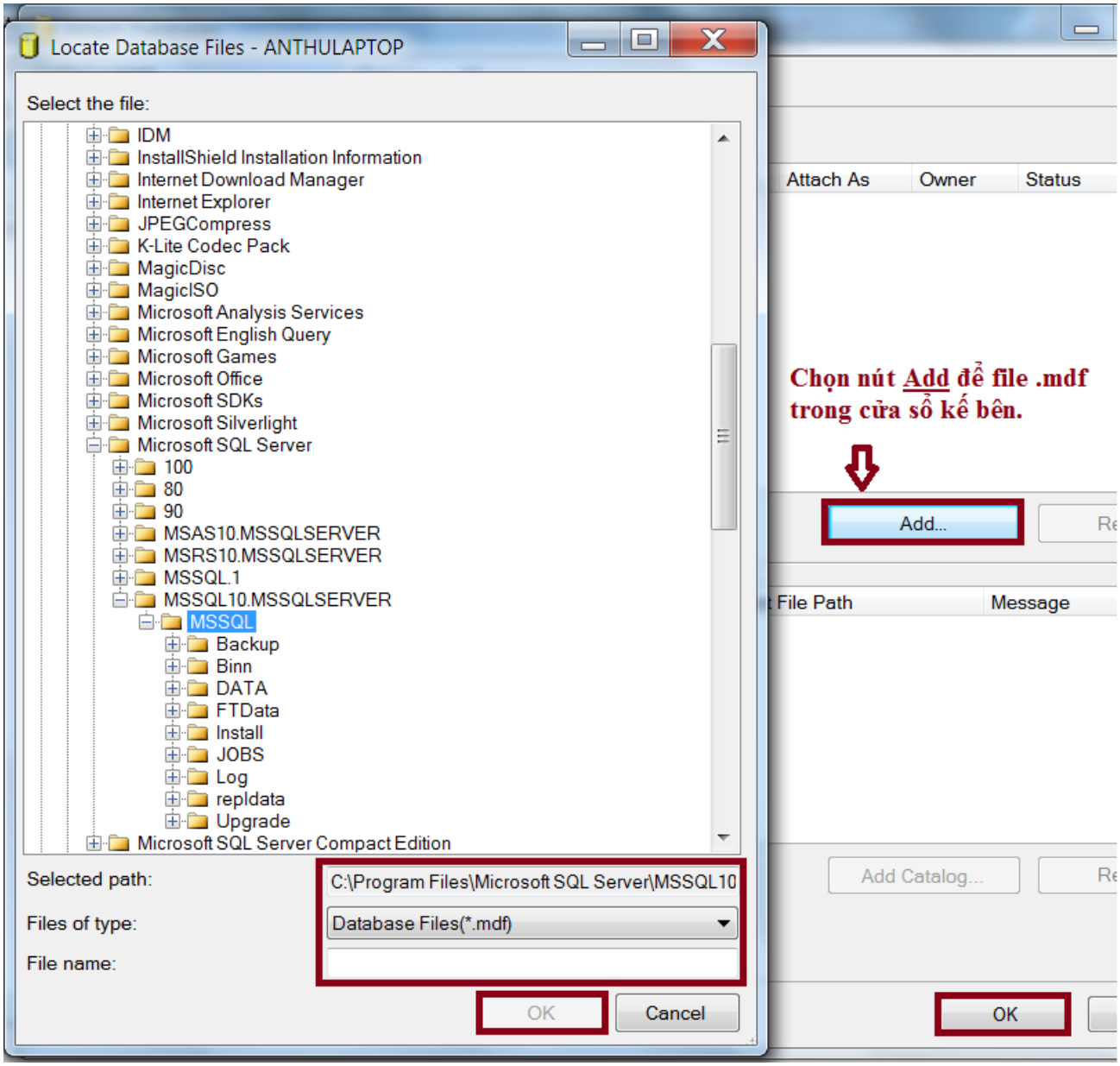


❖ Các bước thực hiện Attach Database:

- Nhắp phải vào Database -> Attach như hình bên dưới:



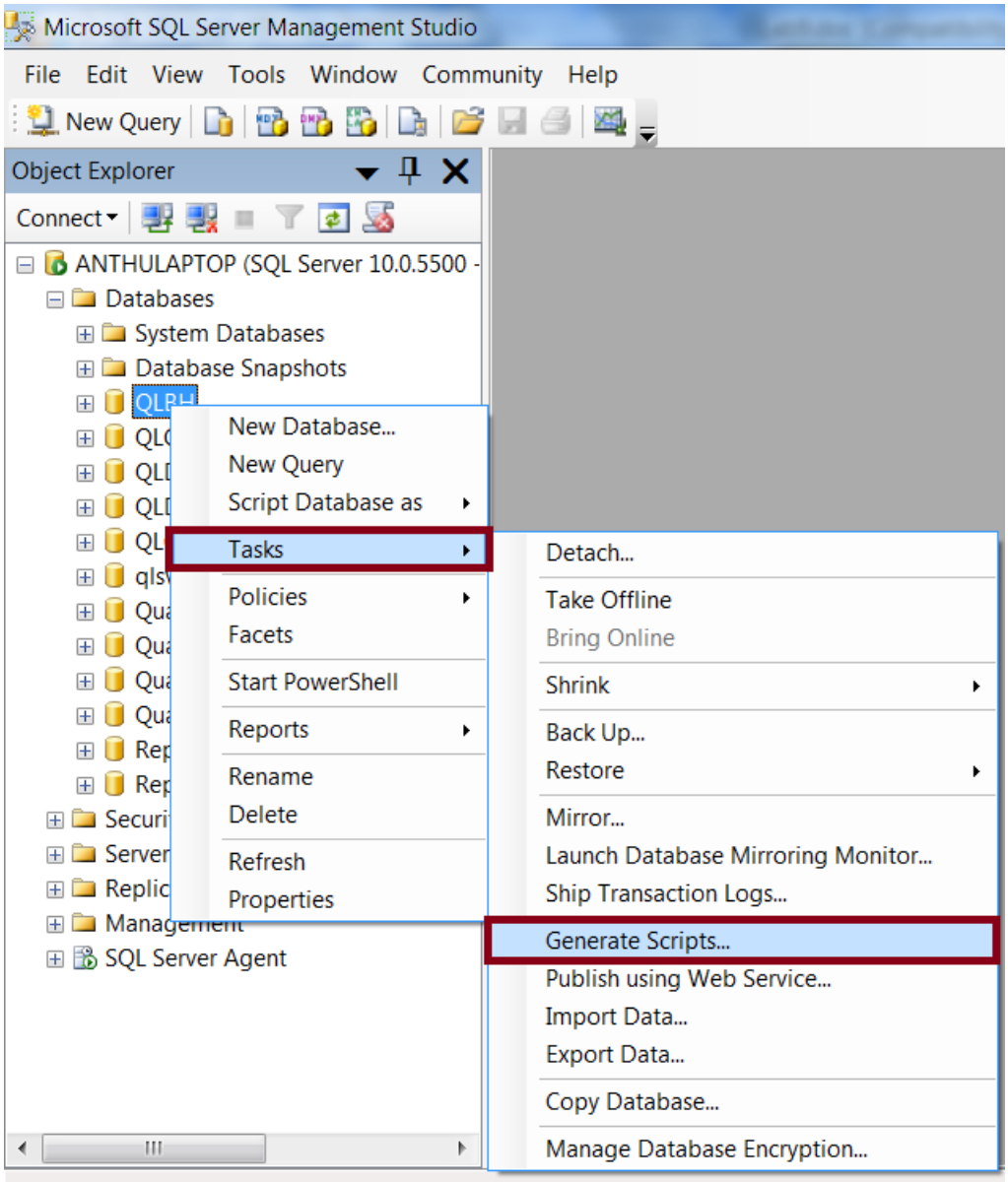
- Chọn file .mdf theo hình vẽ bên dưới:



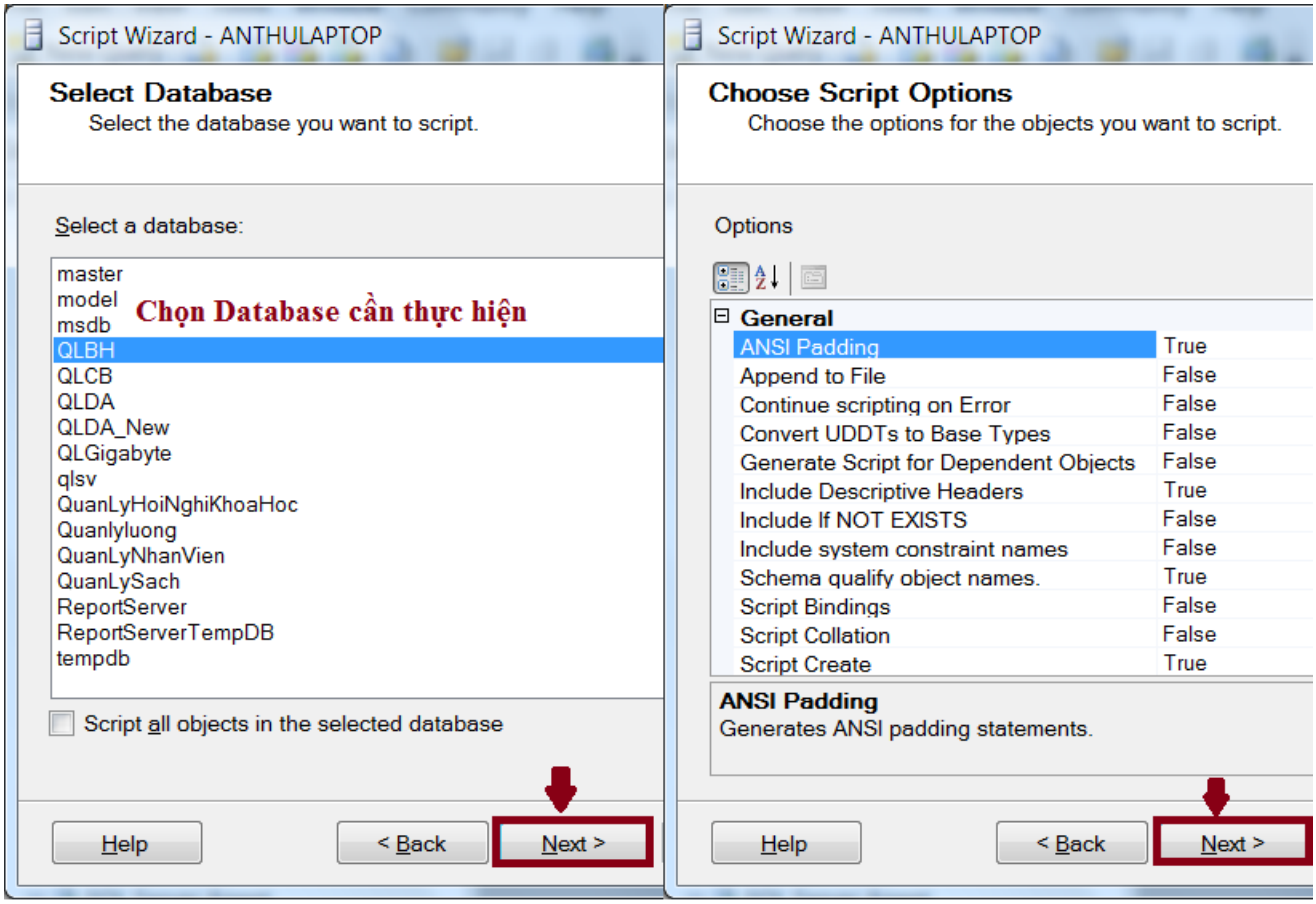
- Hoàn tất công việc Attach file.

Bài 5: **Generate Scripts**: là một chức năng dùng để tạo file "***.sql**" cho cấu trúc bảng, views, proceduces, triggers… đã tạo trong Database với mục đích lưu trữ. Thực hiện các bước sau đây:

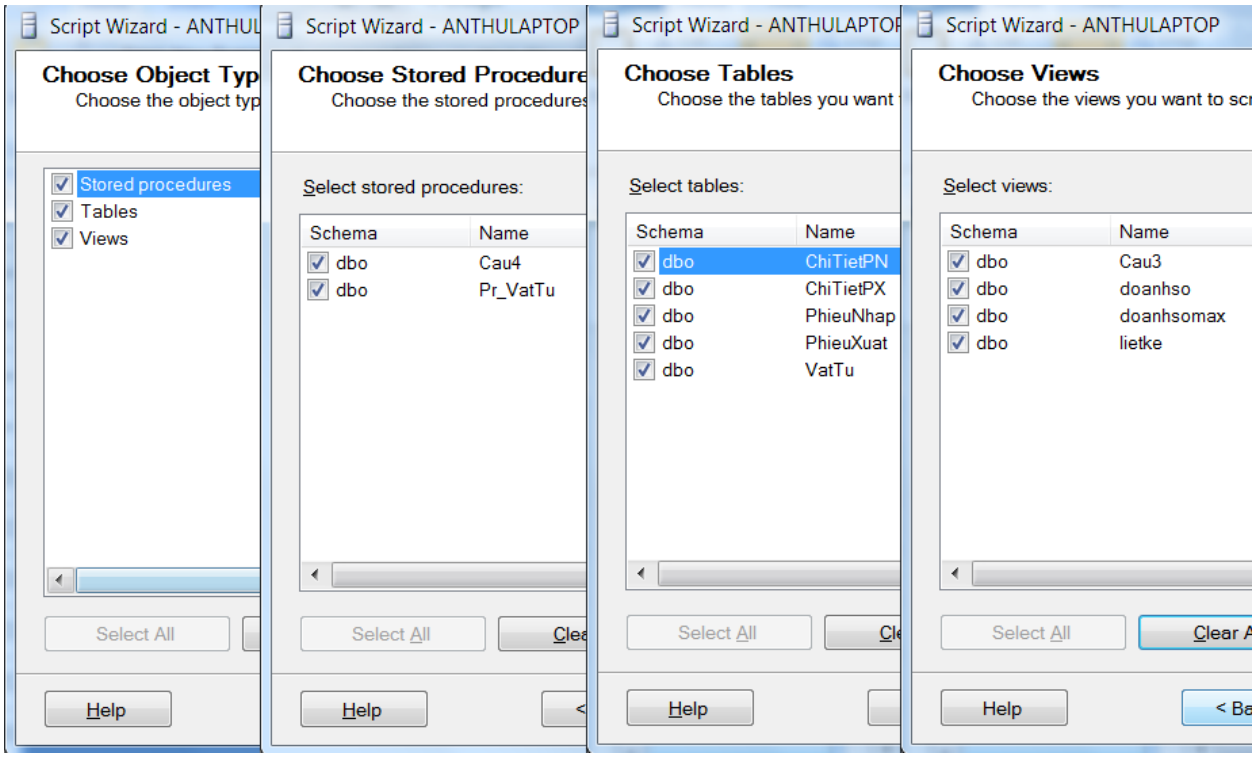
- Bước 1:



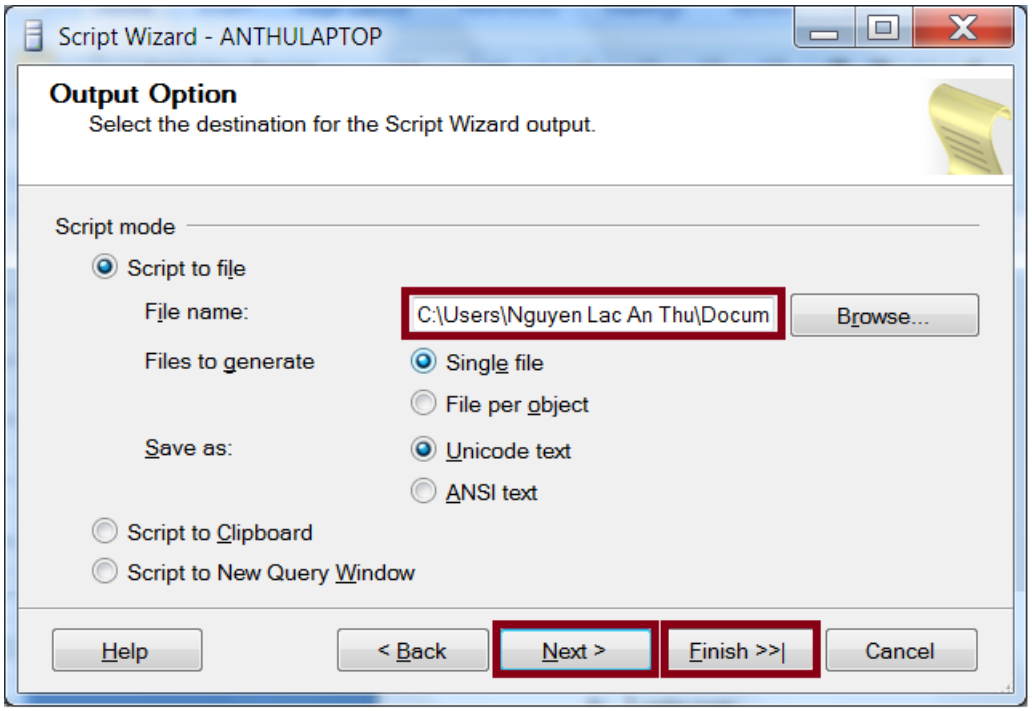
- Bước 2:



- Bước 3: Bấm nút NEXT để mở hộp thoại và chọn các tables, views, proceduces, triggers… cần tạo scripts.



- Bước 4: có ba tùy chọn cho chúng ta chọn lựa: lưu file, lưu trên clipboard, hoặc là mở file script ra một cửa sổ mới.



- Hoàn tất chức năng tạo file script.

Bài 6: Chạy các file script bên dưới, và cho biết chức năng của chúng.

a. Back Up dữ liệu:

```
USE QuanLyNhanVien
GO

BACKUP DATABASE QuanLyNhanVien
TO DISK = 'D:\QuanLyNhanVien.bak'
WITH
DESCRIPTION= 'Backup log Bang Luong  vao o dia D'
GO

BACKUP LOG QuanLyNhanVien
TO DISK = 'D:\QuanLyNhanVien.TRN'
WITH
DESCRIPTION= 'Backup log Bang Luong  vao o dia D'
GO
```

b. Restore dữ liệu:

```
RESTORE DATABASE QuanLyNhanVien
  FROM DISK = 'D:\QuanLyNhanVien.bak'
  WITH
        RECOVERY
GO

RESTORE LOG QuanLyNhanVien
  FROM DISK = 'D:\QuanLyNhanVien.TRN'
  WITH
       RECOVERY
GO
```

Bài 7: Viết lệnh thực hiện sao lưu cơ sở dữ liệu và Transaction Log cho một Database có sẵn.

Bài 8: Viết lệnh thực hiện phục hồi cơ sở dữ liệu và Transaction Log từ file *.BAK và *.TRN đã thực hiện ở bài 3.

-Hết-