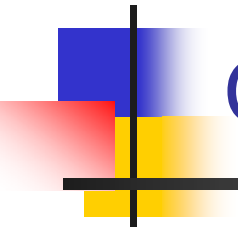
ĐẠI HỌC TÔN ĐỨC THẮNG
PHÒNG TRUNG CẤP CHUYÊN NGHIỆP & DẠY NGHỀ
oOo

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU 2 (SQL Server)

ThS.Lê Văn Hạnh

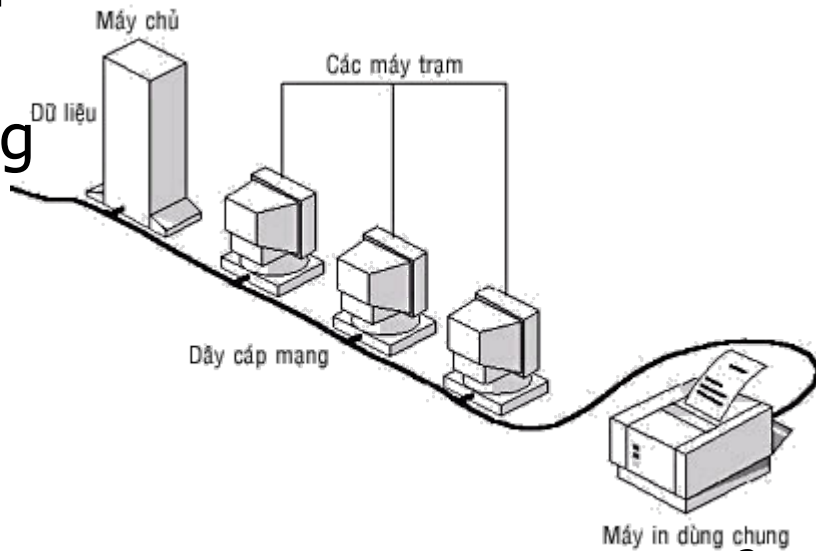


Chương 1

TỔNG QUAN

Khái niệm về cấu trúc vật lý

- Máy chủ (Server):
- Máy trạm (Client): là các máy tính được phép truy xuất các tài nguyên đã được chia sẻ trên mạng.
- Dây cáp mạng (Cable hoặc Media): là một hệ thống dây cáp nối kết vật lý các máy tính, máy in lại với nhau
- Dữ liệu chung (Shared data): là các tập tin, thư mục mà người sử dụng trong hệ thống mạng có thể truy xuất vào máy chủ từ các máy trạm



Khái niệm về các xử lý

- Các xử lý trong một ứng dụng có thể chia làm hai loại xử lý trên máy trạm và xử lý trên máy chủ
- Xử lý trên máy trạm
  - Đọc, cập nhật dữ liệu
  - Tính toán, hiển thị dữ liệu trên màn hình giao diện
  - Có thể sử dụng nhiều loại ngôn ngữ lập trình khác nhau
- Xử lý trên máy chủ Database Server
  - Xử lý các yêu cầu đọc/ghi dữ liệu
  - Quản lý đồng bộ dữ liệu giữa các yêu cầu đọc ghi từ nhiều máy trạm gửi tới
  - Các dịch vụ quản trị dữ liệu tự động theo định kỳ như backup/restore dữ liệu

Vì sao phát triển ứng dụng khách chủ?

- Giảm chi phí
  - Chia sẻ tài nguyên phần cứng/phần mềm
  - Giảm chi phí bản quyền
  - Giảm chi phí nâng cấp, bảo trì, quản lý
- Tốc độ nhanh
  - Các xử lý phức tạp có thể thực hiện tại server
- Tính tương thích cao
  - Nhiều công cụ lập trình được hỗ trợ bởi phần mềm làm việc trên máy chủ

Lịch sử ra đời Microsoft SQL Server

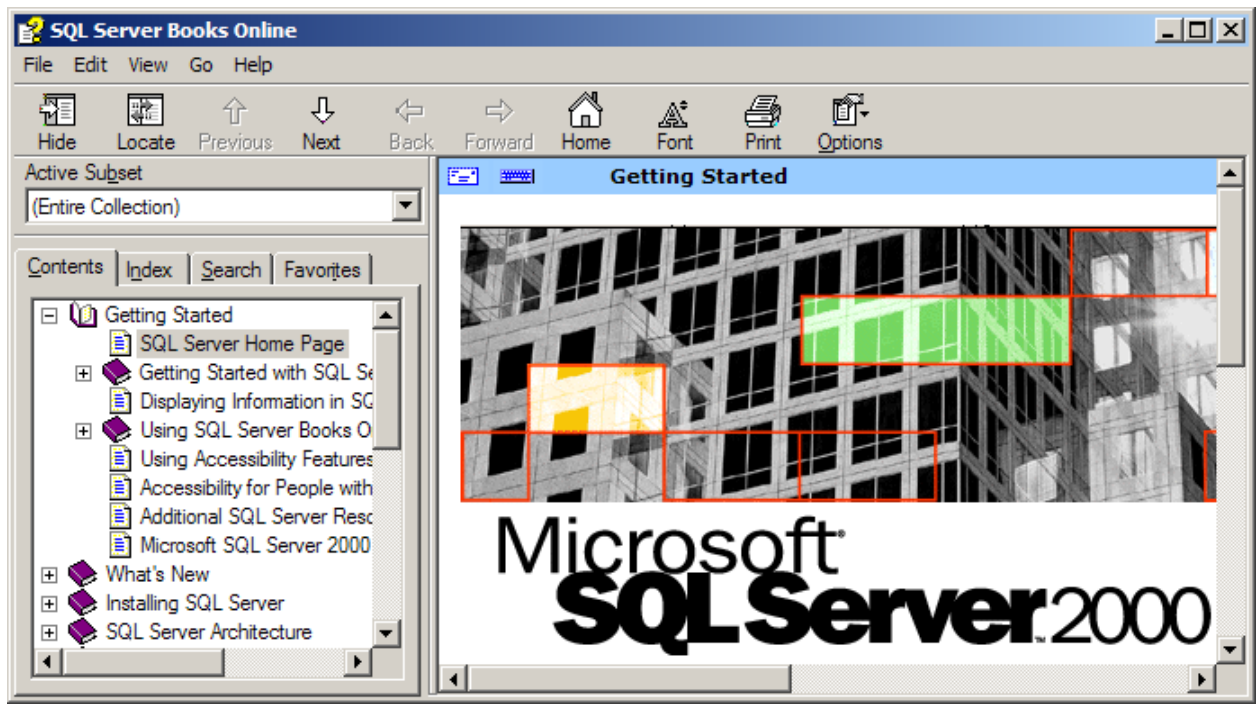
- 1970: IBM giới thiệu ngôn ngữ SEQUEL
- 1987: IBM tích hợp phần mềm quản trị CSDL vào hệ điều hành OS2
- 1988: Hệ quản trị CSDL Ashton-Tate được MS kết hợp với Sybase giới thiệu
- MS bắt đầu phát triển SQL Server trên nền Ashton-Tate và đưa vào WinNT Server sau đó
- Các phiên bản được sử dụng của SQL Server: 4.2, 4.21, 6.0, 6.5, 7.0, 2000, 2005

Cài đặt cơ sở dữ liệu SQL Server Desktop

- Hai phiên bản chính của SQL Server
  - Express
  - Enterprise
- Demo: Cài đặt SQL Server
- Demo: Đăng ký quản trị SQL Server

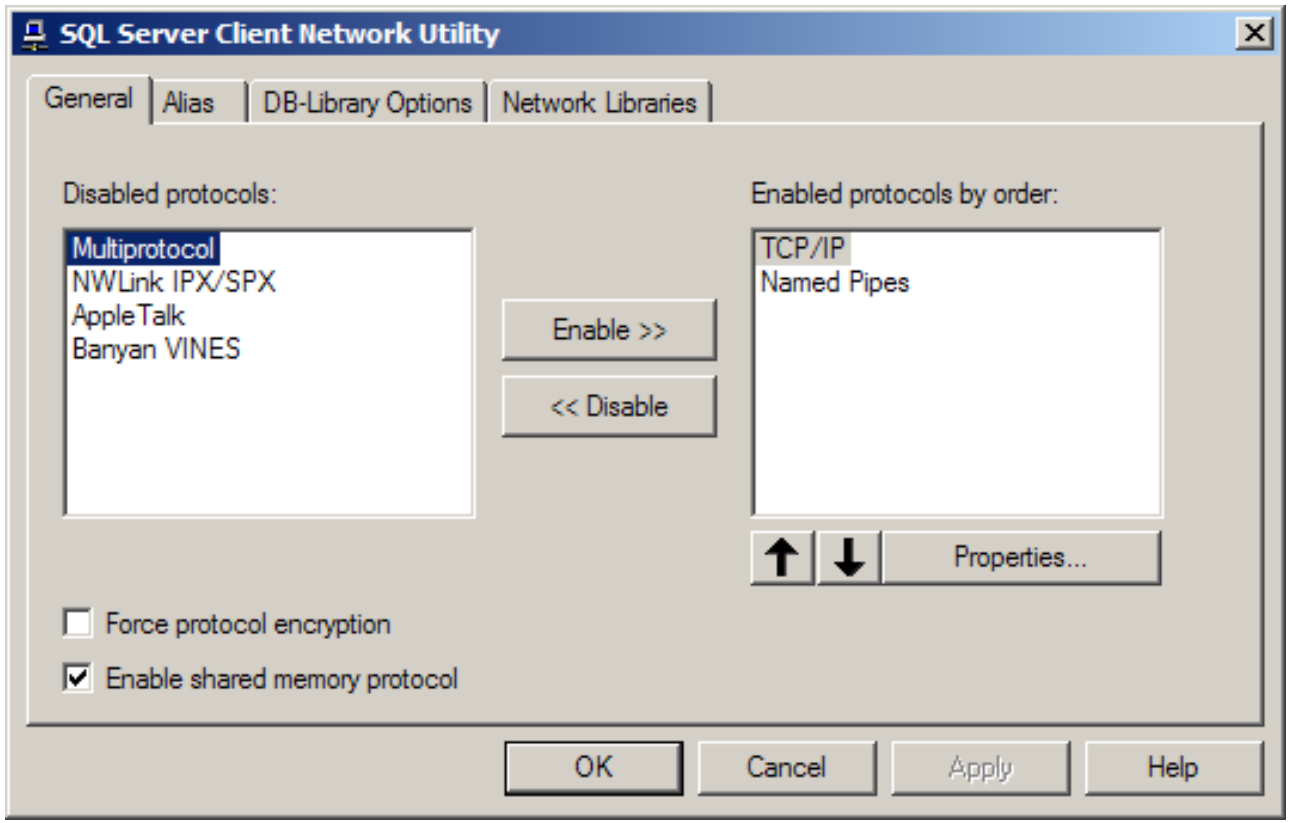
Tiện ích Book Online

- Toàn bộ các tài liệu liên quan đến SQL Server
  - Quản trị SQL Server
  - Cú pháp lệnh
  - Các ví dụ lập trình



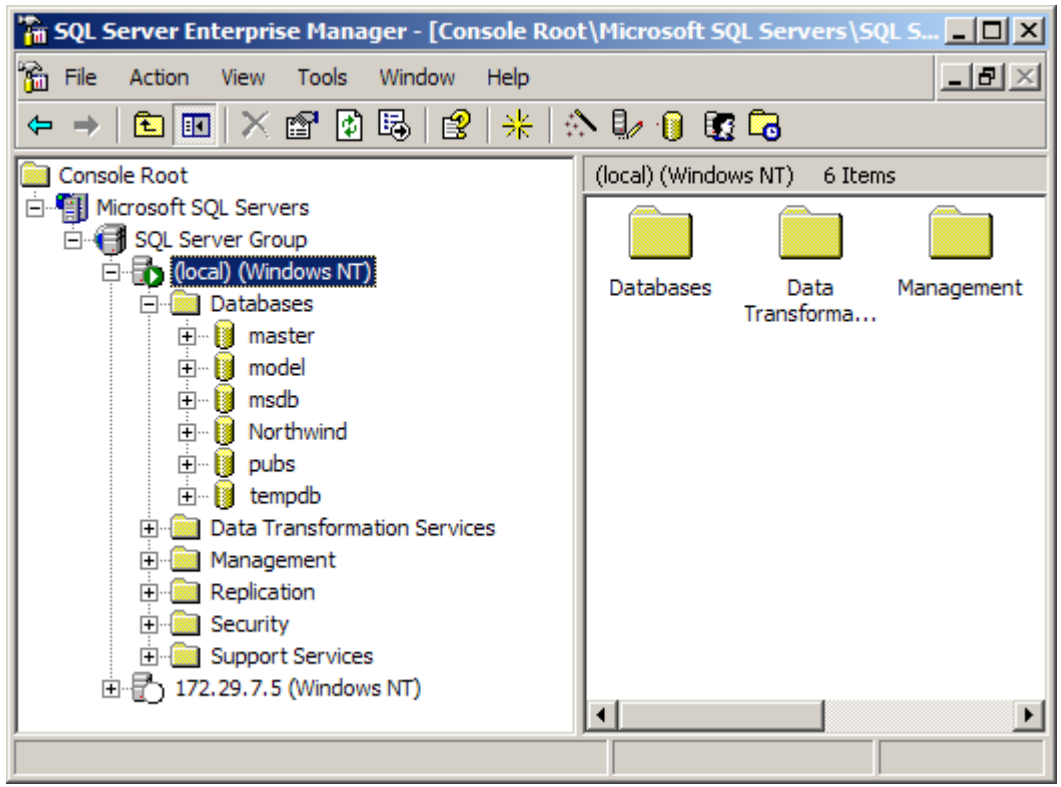
Tiện ích Client network Utility

- Cấu hình các giao thức kết nối mạng mà Client có thể sử dụng



Tiện ích Enterprise Manager

- Công cụ để quản trị SQL Server
  - Database
  - User/Role
  - Replication
  - Security

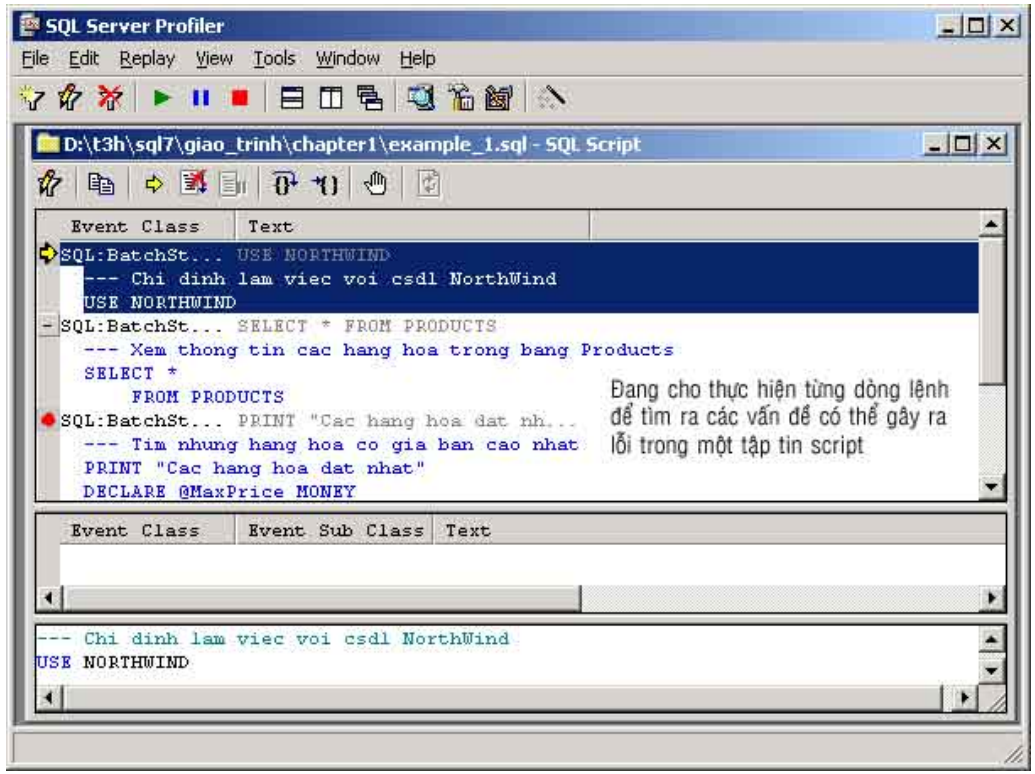


Tiện ích Import and Export Data

- Thực hiện các tính năng nhập dữ liệu/xuất dữ liệu cho các CSDL khác
  - SQL Server
  - Access
  - Excel
  - Oracle
  - Dbase
  - ...

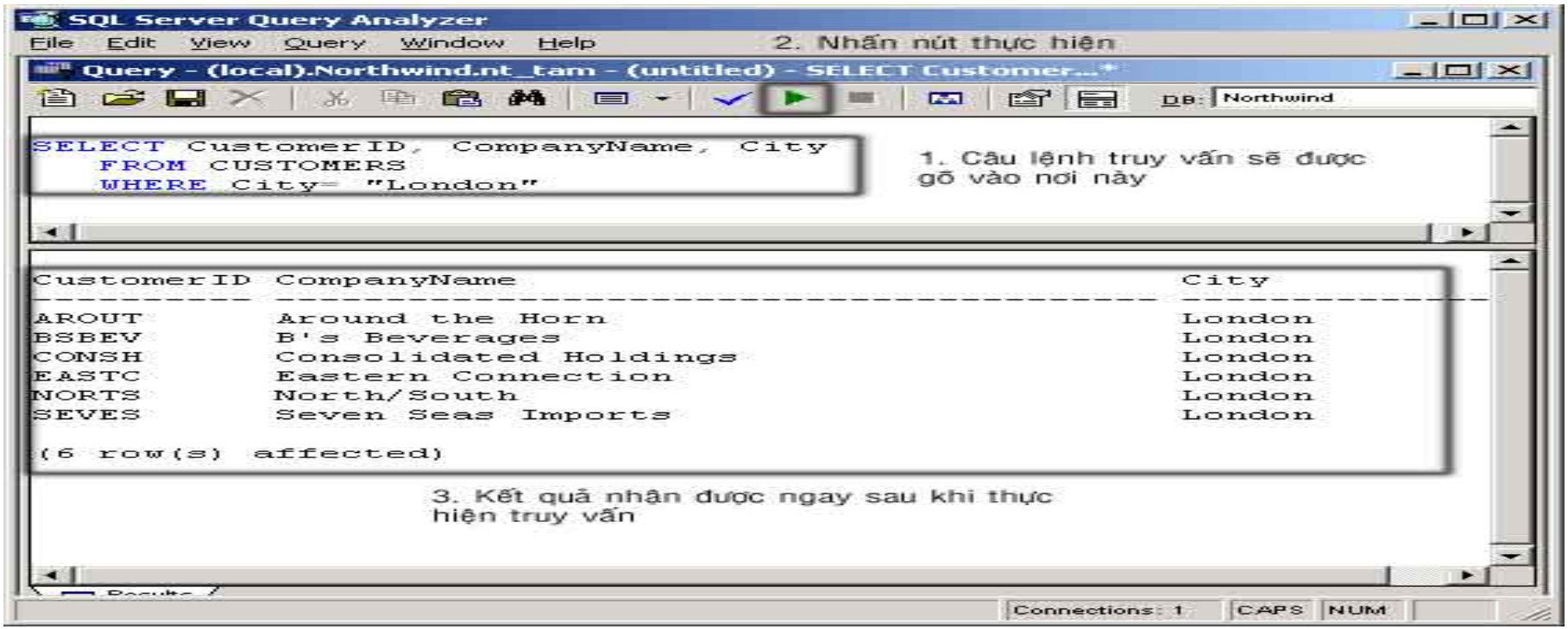
Tiện ích Profiler

- Theo dõi các biến cố xảy ra trên một SQL Server
  - Các biến cố
    - Đăng nhập
    - Truy cập dữ liệu
    - Cập nhật dữ liệu
  - Tạo template profile
  - Kiểm soát thời gian thực hiện/xử lý
  - Kiểm soát locking
  - ...



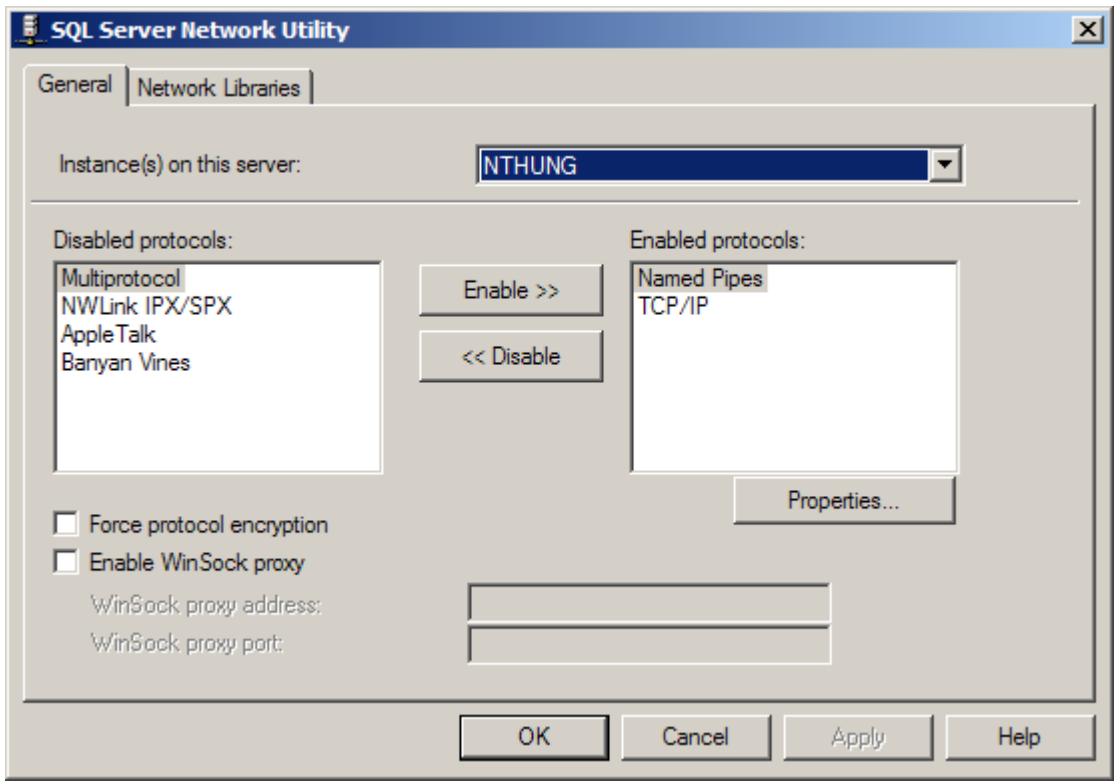
Tiện ích Query Analyzer

- Viết & thực hiện các script
  - Kiểm soát các đối tượng trong CSDL
  - Phát sinh các mẫu câu lệnh script chuẩn
  - Xem kế hoạch thực hiện câu truy vấn, thủ tục nội



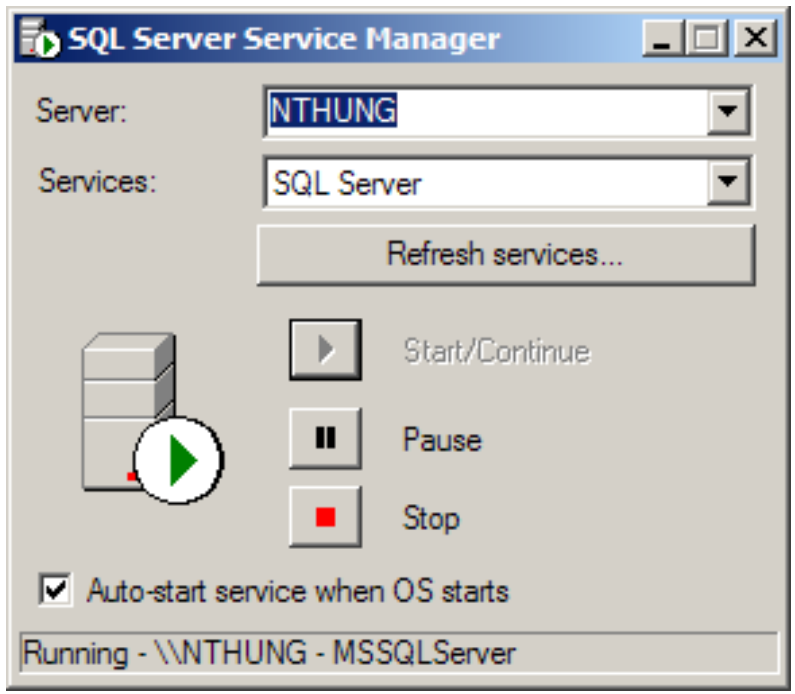
Tiện ích Server Network Utility

- Chỉ định các thư viện mà máy chủ SQL Server dùng khi kết nối với máy trạm
  - Named Pipe
  - TCP/IP
  - Multiprotocol
- Cấu hình mã hoá dữ liệu truyền trên mạng để bảo mật



Tiện ích Service Manager

- Quản lý các dịch vụ của SQL Server
  - SQL Server
  - SQL Server Agent
  - Microsoft Search

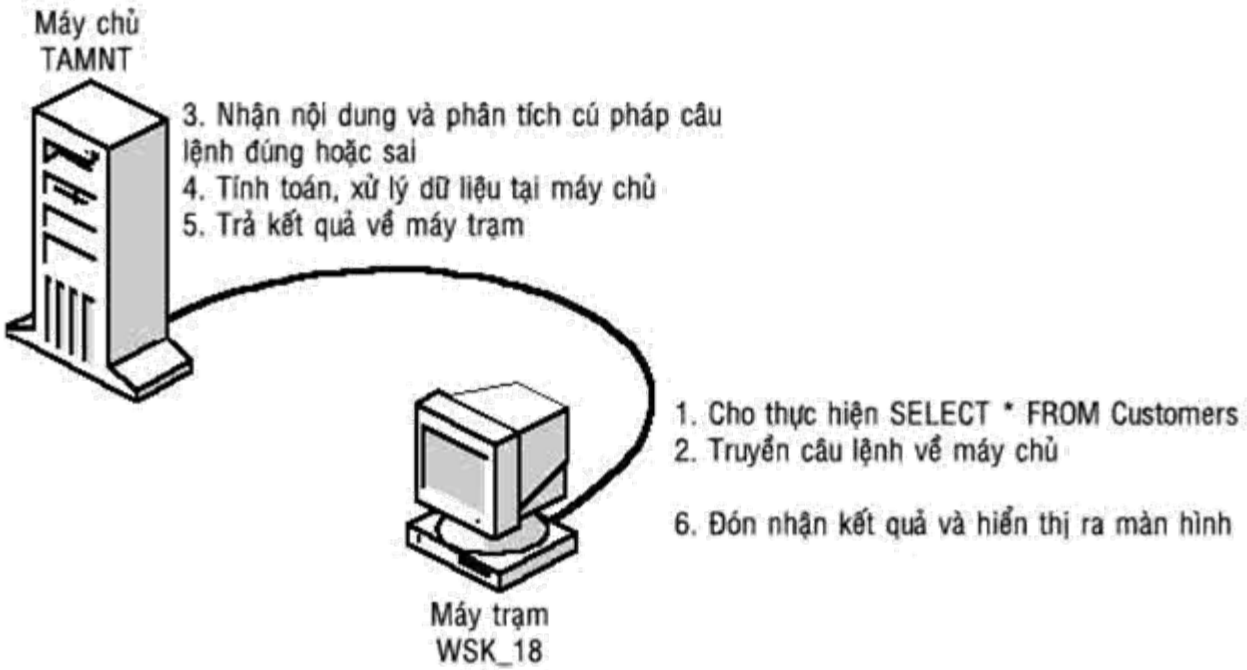


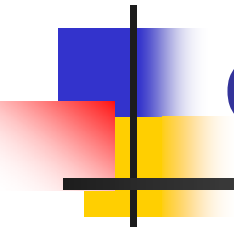
Định nghĩa nối kết vào SQL Server

- Tạo server alias bằng SQL Server Client Network Utilities
- Đăng ký quản trị SQL Server bằng Enterprise Manager
  - Chọn server alias
  - Chọn chế độ kiểm tra khi đăng nhập: Windows Authentication/SQL Server Authentication
  - Chọn chế độ kiểm tra khi kết nối với SQL Server: Login automatically/Prompt

Nối kết từ Query Analyzer vào SQL

- Kết nối với SQL Server tương tự như một client bình thường
  - Cung cấp User name, Password
- Quá trình tương tác giữa SQL Server và Query Analyzer (client) khi thực hiện một lệnh





Chương 2

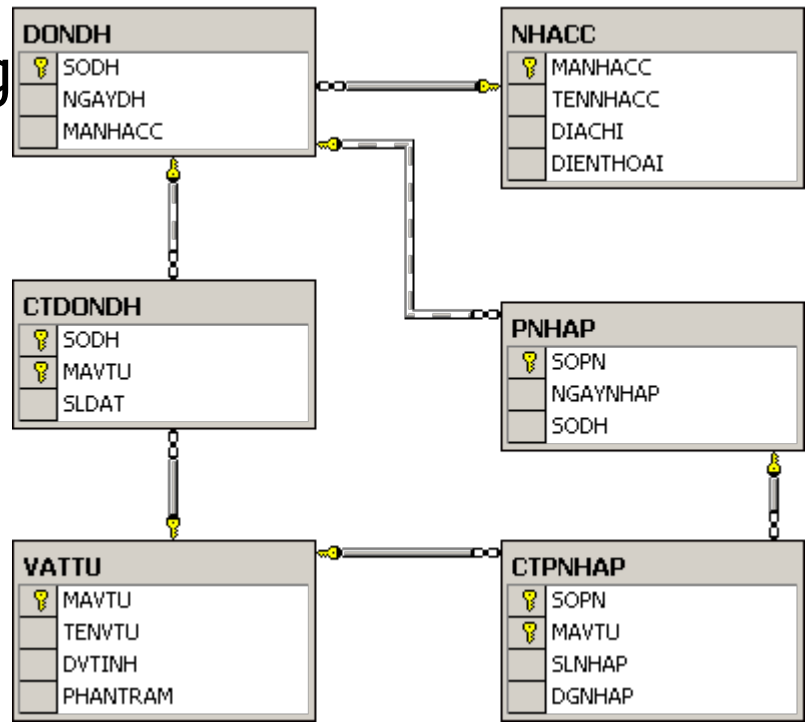
CÁC LỆNH VỀ KiẾN TRÚC CƠ SỞ DỮ LiỆU

Khái niệm về cơ sở dữ liệu

- Database dùng để
  - Chứa các bảng, bảng ảo, thủ tục nội,...
  - Mỗi database có một danh sách các người dùng
    - Người dùng phải có quyền truy cập database
    - Có thể phân nhóm người dùng để cấp quyền
    - SQL Server hỗ trợ Application Role
- Các database hệ thống
  - Master, Model, TempDB, msdb
- Các database ví dụ
  - Northwind, Pubs

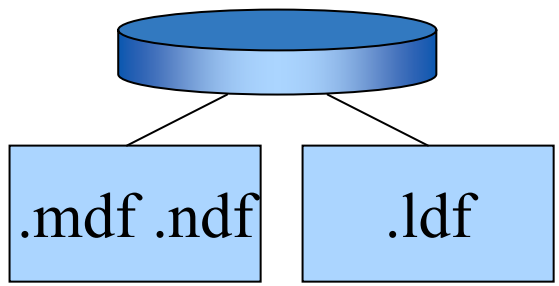
Khái niệm về mô hình quan hệ dữ liệu

- Thể hiện mối quan hệ giữa các bảng trong CSDL
- Có thể sử dụng để
  - Thiết lập mối quan hệ khoá ngoại (FOREIGN KEY)
  - Chỉnh sửa cấu trúc bảng
  - Chỉnh sửa thuộc tính bảng
  - Tạo bảng mới



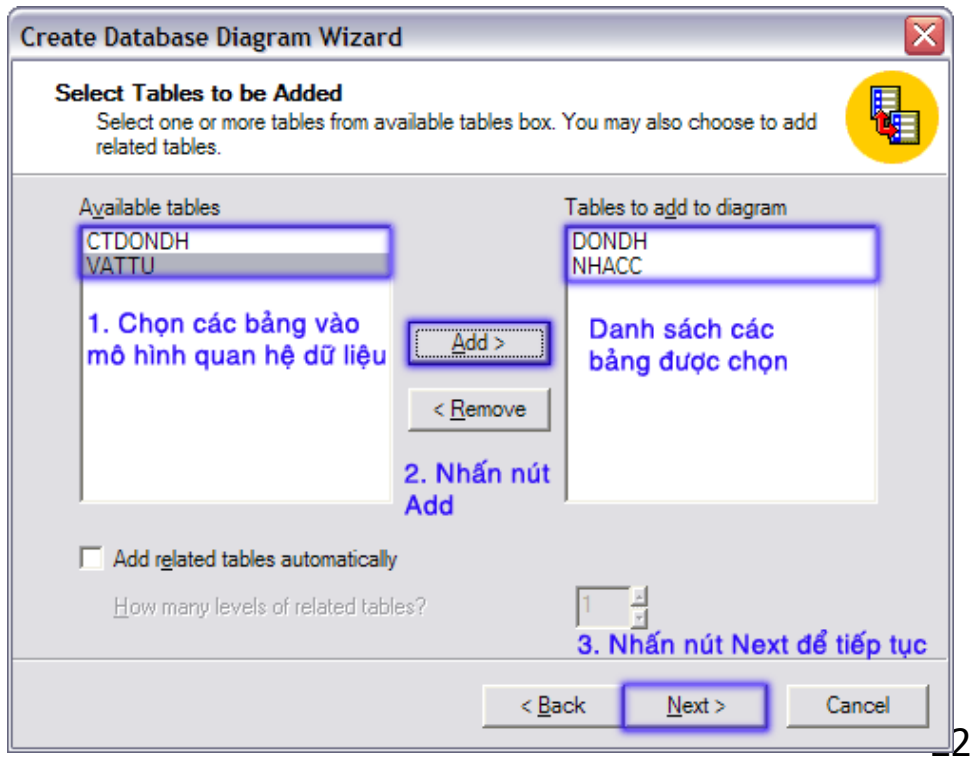
Các tập tin vật lý lưu trữ cơ sở dữ liệu

- Một database bao gồm tối thiểu hai file
  - .mdf: lưu trữ các đối tượng trong database như table, view, …
    - Có thể bổ sung thêm các tập tin lưu trữ khác
    - Tổ chức tốt các tập tin lưu trữ giúp tăng tốc độ xử lý
  - .ldf: lưu trữ quá trình cập nhật/thay đổi dữ liệu
    - Hỗ trợ phục hồi dữ liệu
    - Hỗ trợ backup/restore dữ liệu
- Các thông số về kích thước
  - Initial size
  - File growth
  - Maximum file size



Tạo mới mô hình quan hệ dữ liệu

- Chỉ có thể tạo bằng Enterprise Manager
- Với một CSDL lớn, tạo một hay nhiều mô hình cho các nghiệp vụ thực tế khác nhau

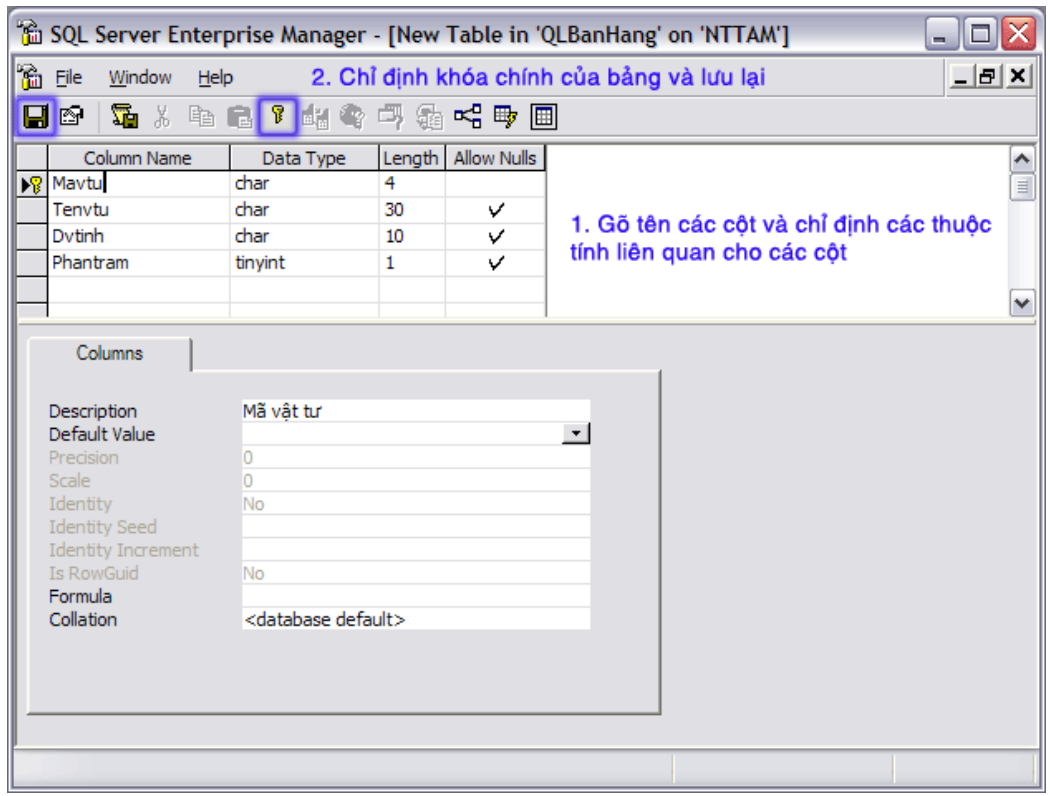


Khái niệm về bảng

- Bảng dùng để lưu trữ các thông tin của một đối tượng trong thực tế
  - Gồm có dòng và cột
  - Bảng trong CSDL thường có khoá chính
  - Các bảng thường liên hệ với nhau bằng các mối quan hệ
- Bảng trong CSDL SQL Server có thể có các ràng buộc, trigger

Các thuộc tính của bảng

- Tên bảng
- Tên cột
- Kiểu dữ liệu
  - Độ dài dữ liệu
  - Số ký số lưu trữ
  - Số số lẻ lưu trữ
- Thuộc tính trên cột
  - Allow null
  - Identity
  - Default value



Tạo cấu trúc bảng đơn giản

```
CREATE TABLE Tên_bảng
(
  Tên_cột1    Kiểu_dữ_liệu [NOT NULL] ,
  Tên_cột2    Kiểu_dữ_liệu [NOT NULL] [, ...]

)
```

- Từ khóa NOT NULL chỉ định không cho phép dữ liệu tại cột bị bỏ trống.

Tạo cấu trúc bảng có cột định danh

```
CREATE TABLE Tên_bảng
(
  Tên_cột1    Kiểu_dữ_liệu_số
  IDENTITY [(Số_bắt_đầu, Chỉ_số_tăng)] ,
  Tên_cột2    Kiểu_dữ_liệu [NOT NULL] [, ...]
)
```

- Kiểu dữ liệu số: dạng số nguyên (int, smallint, tinyint, numeric và decimal)
  - Với numeric và decimal thì phải chỉ định không lấy số lẻ.
- Số bắt đầu: SQL Server sử dụng để cấp phát cho mẩu tin đầu tiên. Mặc định là 1.
- Chỉ số tăng: số cộng lên để cấp phát cho những mẩu tin kế tiếp. Mặc định là 1.

Thay đổi cấu trúc bảng

- Dùng Enterprise Manager
  - Nhanh, đơn giản
  - Dùng giao diện, không dùng lệnh
- Dùng script
  - Phức tạp, phải thuộc cú pháp lệnh
  - Cần thiết khi
    - Sử dụng lại nhiều lần để cập nhật cho CSDL trên máy khác
    - Cập nhật CSDL qua nhiều giai đoạn

Thêm một cột mới trong bảng

```
ALTER TABLE Tên_bảng
  ADD  Tên_cột  Kiểu_dữ_liệu [, ...]
```

- Tên cột: tên của cột mới được thêm vào bảng.
- Kiểu dữ liệu: kiểu dữ liệu tương ứng của cột mới.

```
ALTER TABLE DONDH
ADD   Ngaydknh  DATETIME
```

Hủy bỏ cột hiện có bên trong bảng

```
ALTER TABLE Tên_bảng
  DROP COLUMN Tên_cột [, ...]
```

- Tên cột: tên cột sẽ bị hủy bỏ ra khỏi bảng

```
ALTER TABLE DONDH

DROP COLUMN Ngaydknh
```

Sửa đổi kiểu dữ liệu của cột

```
ALTER TABLE Tên_bảng
  ALTER  COLUMN Tên_cột  Kiểu_dữ_liệu_mới
```

```
ALTER TABLE VATTU
ALTER  COLUMN    Dvtinh  VARCHAR(20)
```

Đổi tên cột, tên bảng dữ liệu

```
EXEC  sp_rename  'Tên_bảng[.Tên_cột]', 'Tên_mới'
[,'COLUMN']
```

- EXEC: dùng để thực thi các thủ tục nội tại của SQL Server
- Tên bảng: tên bảng sẽ đổi tên hoặc chứa tên cột muốn đổi tên
- Tên cột: tên cột muốn đổi tên
- Tên mới: tên mới của cột hoặc bảng sau khi đổi
- COLUMN: sử dụng khi thay đổi tên cột

```
EXEC sp_rename 'NHACC.Tennhacc', 'Hotenncc',
'COLUMN'

EXEC sp_rename  'NHACC',  'NHACCAP'
```

Kiểu dữ liệu do người dùng tự định nghĩa (User Define Type)

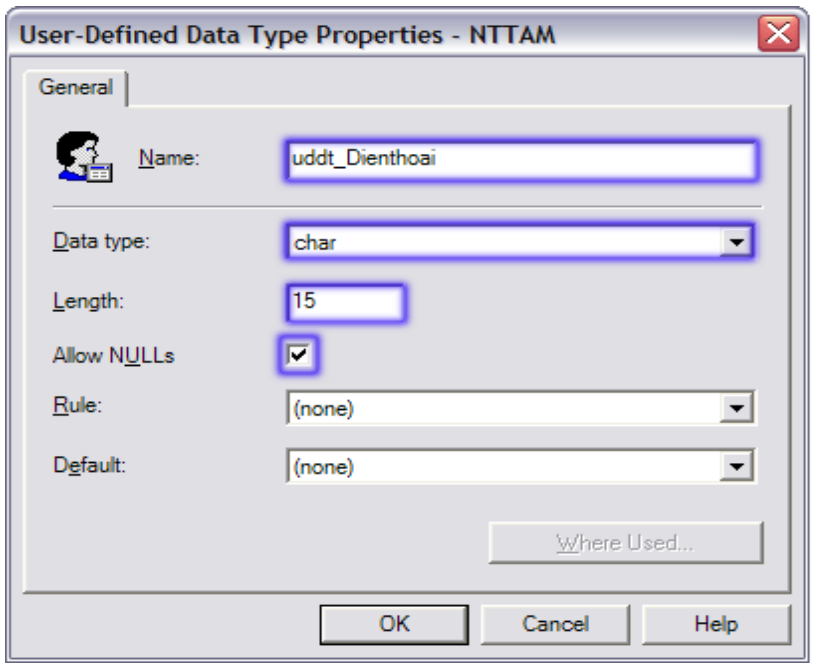
Khái niệm

- Dựa trên kiểu dữ liệu định sẵn
- Bổ sung các thuộc tính
  - Allow null
  - Giá trị mặc định
  - Kiểm tra miền giá trị
- Ưu điểm
  - Giúp thống nhất các cột dữ liệu trong CSDL theo một kiểu
  - Dễ thay đổi, chỉnh sửa

Tạo kiểu dữ liệu người dùng định nghĩa

- Có thể tạo bằng Enterprise Manager
- Cú pháp lệnh

```
EXEC sp_addtype Tên_kiểu_dl_mới, 'Kiểu_dl_cơ_sở'
[,NULL | NOT NULL]
EXEC  sp_addtype uddt_Soluong, 'Decimal(15,2)',
'NOT NULL'
```



Xóa kiểu dữ liệu người dùng định nghĩa

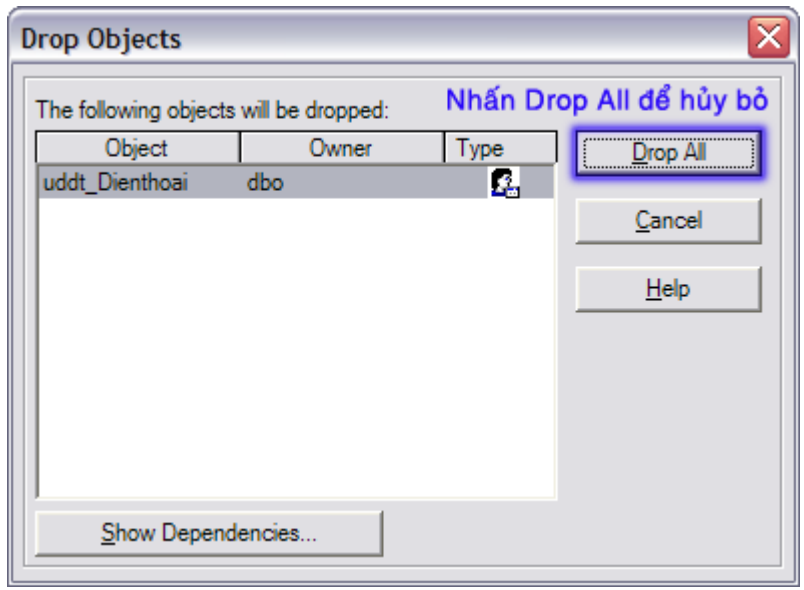
EXEC sp_droptype Tên_kiểu_dl

- Tên kiểu dữ liệu:
  - tên kiểu dữ liệu do người dùng định nghĩa

```
EXEC    sp_droptype    uddt_Soluong
```

Server: Msg 15180, Level 16, State 1, Line 0

Cannot drop. The data type is being used.



Lệnh INSERT INTO

```
INSERT INTO Tên_bảng  [ (Danh_sách_cột) ]
  VALUES (Danh_sách_giá_trị)
```

- Có thể không cần chỉ định ra tên của các cột
  - Số lượng các giá trị bằng số các cột trong bảng
  - Danh sách các giá trị mà chúng ta đưa vào phải theo đúng thứ tự của các cột bên trong bảng

```
INSERT INTO VATTU (MAVTU, TENVTU, DVTINH,
  PHANTRAM)
VALUES ('LO01', 'Loa Panasonic 1000W', 'Bộ', 10)
```

Lệnh INSERT INTO SELECT

```
INSERT [INTO] Tên_bảng [ (Danh_sách_cột) ]
  SELECT Danh_sách_cột
  FROM Tên_bảng_dl_nguồn
  WHERE Điều_kiện_lọc
```

- Danh sách các cột của câu Select phải tương ứng với các cột của mệnh đề Insert

Lệnh DELETE FROM

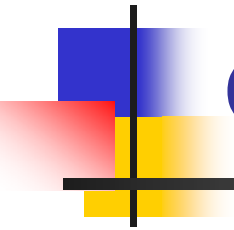
```
DELETE [FROM] Tên_bảng
  [FROM Tên_bảng1
    INNER|LEFT|RIGHT  JOIN Tên_bảng2
    ON Biểu_thức_liên_kết]
  [WHERE Điều_kiện_xóadl]
```

- Tên bảng: bảng có các dòng dữ liệu muốn hủy bỏ
- Tên bảng1, tên bảng2: các bảng có quan hệ dữ liệu, được dùng để kết nối các quan hệ nhằm tra cứu các thông tin trong khi xóa dữ liệu
- Nếu không sử dụng mệnh đề WHERE thì tất cả các dòng dữ liệu sẽ bị hủy

Lệnh UPDATE SET

```
UPDATE Tên_bảng
  SET  Tên_cột =  Biểu_thức [ , ...]
[FROM Tên_bảng1
INNER|LEFT|RIGHT  JOIN Tên_bảng2
ON Biểu_thức_liên_kết]
[WHERE Điều_kiện_sửa_đổi]
```

```
UPDATE PNHAP
SET TGNHAP = ( SELECT  SUM(SLNHAP*DGNHAP)
               FROM CTPNHAP CTPN
               WHERE PN.SOPN=CTPN.SOPN )
FROM PNHAP PN
```



Chương 3

RÀNG BUỘC DỮ LiỆU

Các quy định của công việc trong thực tế

- Trong thực tế mỗi công việc đều có những quy định phải tuân theo
  - Mỗi quy định trở thành một hay nhiều ràng buộc trong CSDL
  - Một số quy định đơn giản, mặc nhiên thấy cũng phải mô tả trong CSDL
- Ví dụ quản lý đơn đặt hàng
  - Số lượng đặt hàng phải lớn hơn 0
  - Các số hoá đơn giao hàng không được trùng nhau
  - Ngày dự kiến nhận hàng phải sau ngày đặt hàng
  - Một đơn đặt hàng phải do một khách hành lập ra
  - Mỗi một mặt hàng phải có nhà cung cấp (mỗi mặt hàng phải có xuất xứ)
  - Số lượng mặt hàng giao cho khách phải nhỏ hơn hay tối đa bằng với số lượng đặt
  - Hai nhà cung cấp có thể trùng tên nhưng là hai nhà cung cấp khác nhau

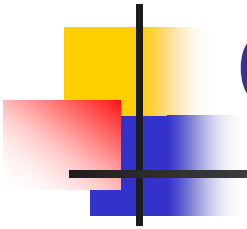
Các ràng buộc toàn vẹn dữ liệu

- SQL Server chia làm hai loại chính
  - Loại đơn giản: sử dụng CONSTRAINT để **mô tả**
  - Loại phức tạp: sử dụng TRIGGER để **thực hiện**
- Các loại ràng buộc đơn giản
  - Kiểm tra duy nhất
    - PRIMARY KEY, UNIQUE
  - Kiểm tra tồn tại
    - FOREIGN KEY
  - Kiểm tra miền giá trị
    - CHECK, DEFAULT
- SQL Server thực hiện việc kiểm tra dữ liệu dựa trên những constraint đã mô tả

Sử dụng constraint để kiểm tra toàn vẹn dữ liệu

- Một constraint luôn gắn với một bảng
  - Tạo constraint ngay khi tạo bảng
    - Thường dùng với PRIMARY KEY, DEFAULT
  - Tạo constraint bằng lệnh ALTER TABLE
    - Thường dùng với CHECK, FOREIGN KEY, UNIQUE

```
CREATE TABLE CTDONDH(
    Sodh CHAR(4) , Mavtu CHAR(4) , SlDat SMALLINT
    PRIMARY KEY (Sodh, Mavtu) ,
    FOREIGN KEY (Sodh)   REFERENCES DONDH (Sodh),
    CHECK (SlDat BETWEEN 10 AND 50))
ALTER TABLE NHACC
    ADD CONSTRAINT UNQ_NHACC_DIACHI UNIQUE (Diachi),
    CONSTRAINT DEF_NHACC_DIENTHOAI
    DEFAULT 'Chưa có' FOR Dienthoai
```



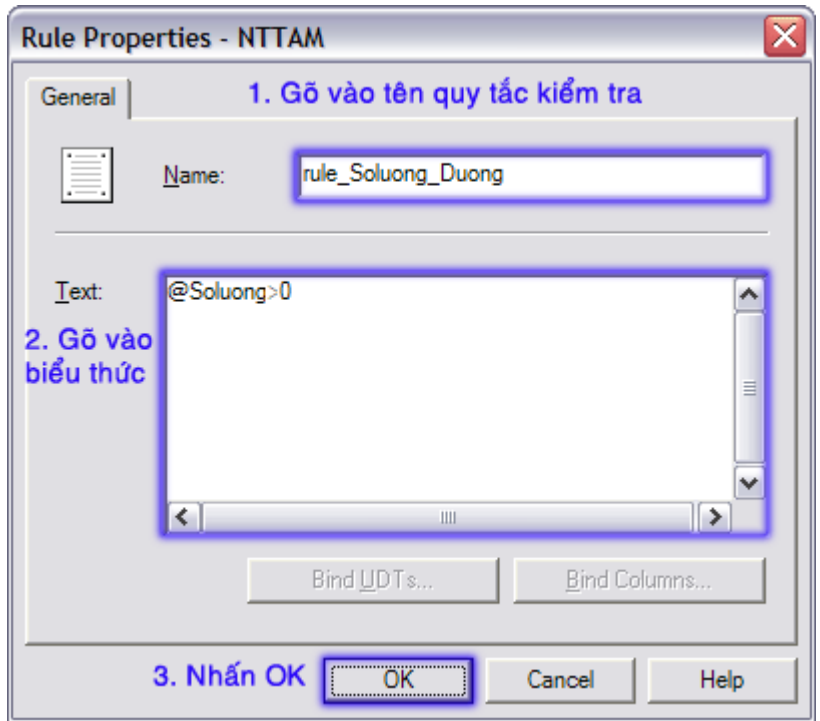
CHECK

- Sử dụng để kiểm tra miền giá trị của dữ liệu
  - Tương tự như CHECK constraint
  - Cùng một đối tượng Rule dùng cho nhiều cột giống nhau trong nhiều bảng
  - Đơn giản hoá việc thay đổi quy tắc kiểm tra khi thực tế thay đổi

RULE

- Sử dụng một biến đại diện cho cột sẽ kiểm tra
- Mô tả điều kiện kiểm tra dữ liệu dựa trên biến
- Hạn chế
  - Không thể mô tả ràng buộc trên hai cột

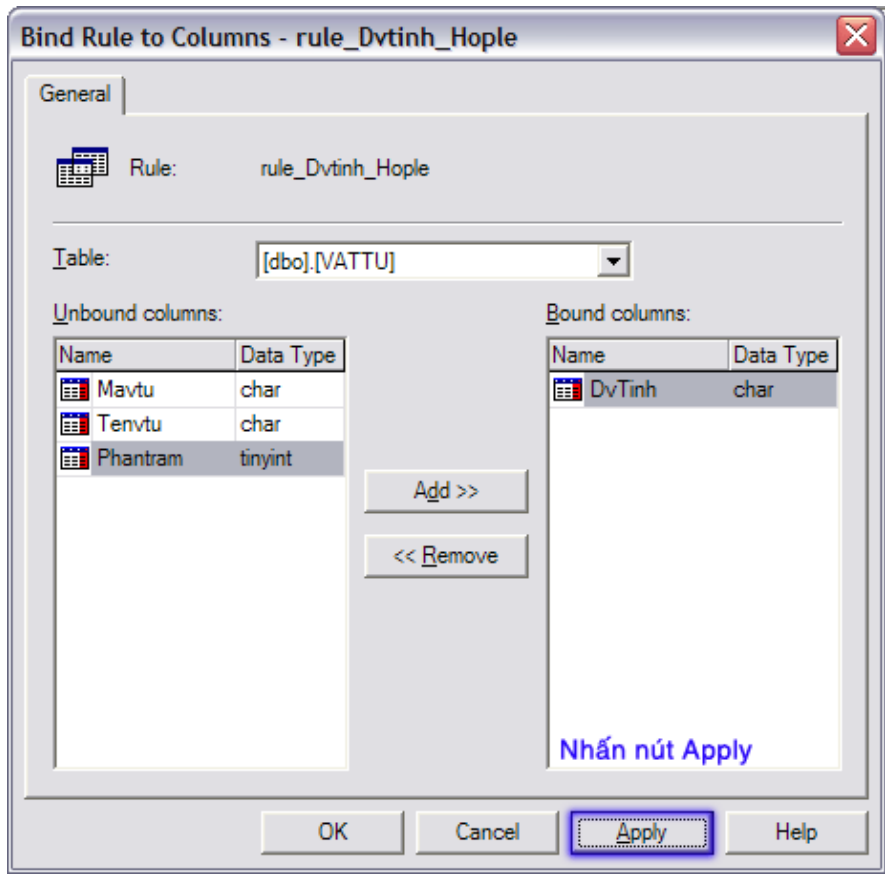
```
CREATE  RULE  Tên_qui_tắc
AS Biểu_thức

CREATE RULE rule_Soluong_Duong
AS
  @Soluong>0
```



Áp dụng quy tắc kiểm tra miền giá trị

- Rule sau khi tạo mới phải được kết nối với cột trong bảng

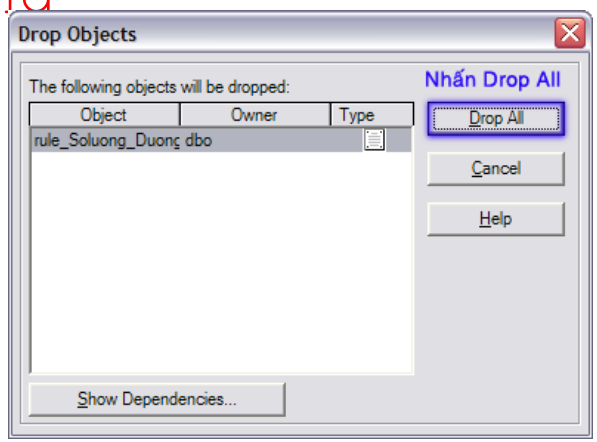
```
EXEC  sp_bindrule
Tên_qui_tắc, Tên_đối_tượng

EXEC sp_bindrule
rule_Dvtinh_Hople ,
'VATTU.Dvtinh'
```

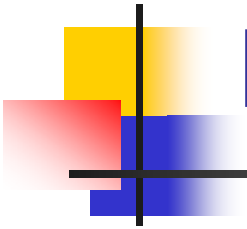


Xóa quy tắc kiểm tra miền giá trị

- Gỡ bỏ quy tắc kiểm tra khỏi bảng
  - Sử dụng Enterprise Manager
  - EXEC sp_unbindrule Tên_đối_tượng
- Xoá quy tắc kiểm tra
  - Phải gỡ bỏ quy tắc kiểm tra ra khỏi tất cả các bảng trước khi xoá

```
DROP RULE  Tên_qui_tắc
DROP  RULE  rule_Soluong_Duong
```





DEFAULT

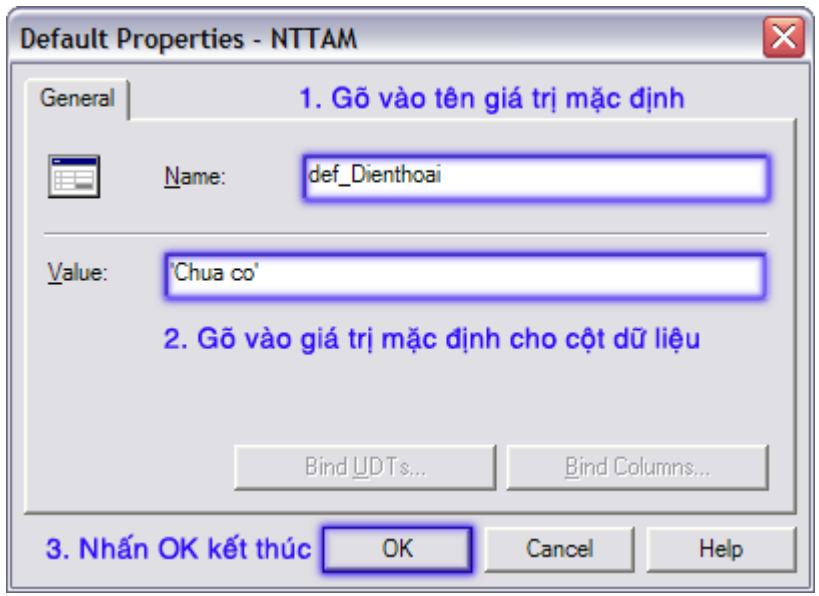
- Tạo ra một giá trị mặc định để có thể gán vào một cột hay một kiểu dữ liệu
  - Tương tự như DEFAULT constraint
  - Giúp tạo một giá trị như một hằng số, thống nhất giữa tất cả các cột trong các bảng khác nhau
  - Dễ quản lý, dễ thay đổi

Tạo mới giá trị mặc định

- Tạo mới bằng Enterprise Manager
- Tạo mới bằng script

```
CREATE   DEFAULT  Tên_giá_trị_mặc_định
AS
  Biểu_thức
```

```
CREATE DEFAULT
Def_Dienthoai
AS
  'Chưa có'
```

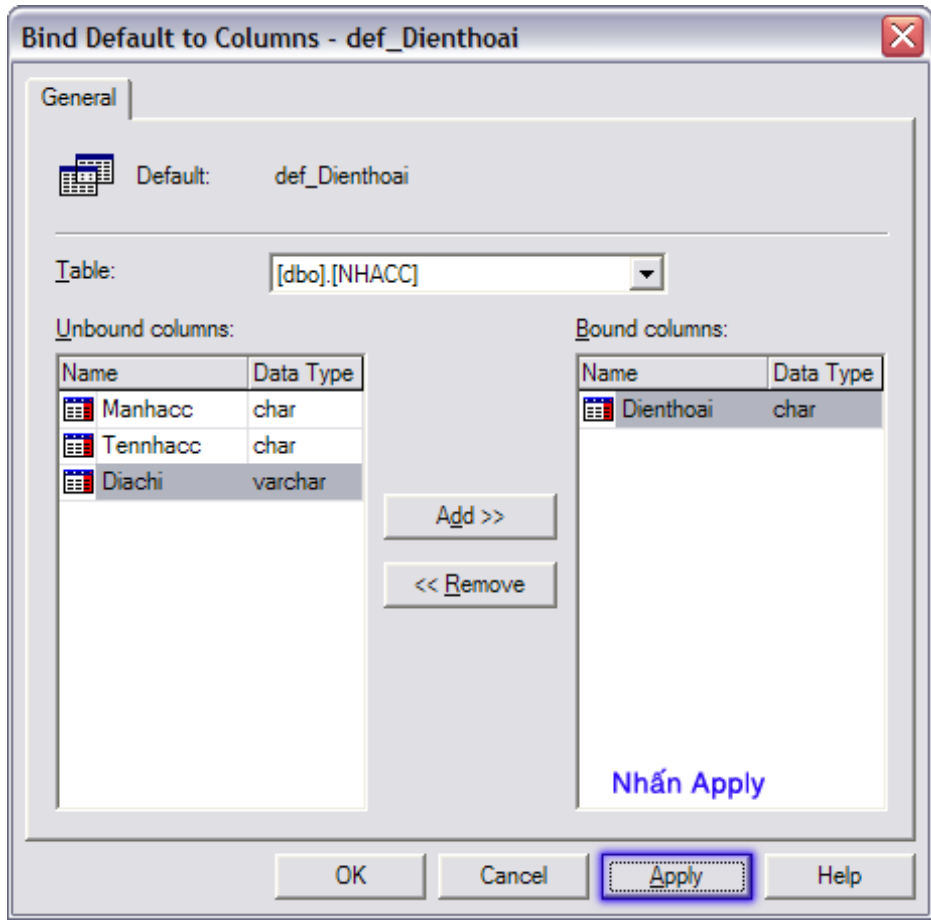


Liên kết giá trị mặc định vào cột dữ liệu

- Tương tự như Rule, giá trị mặc định sau khi tạo ra phải được liên kết với một cột hay kiểu dữ liệu

```
EXEC  sp_bindefault
Tên_mặc_định,
Tên_đối_tượng
```

```
EXEC sp_bindefault
def_Dienthoai ,
  'NHACC.Dienthoai'
```



Xóa giá trị mặc định

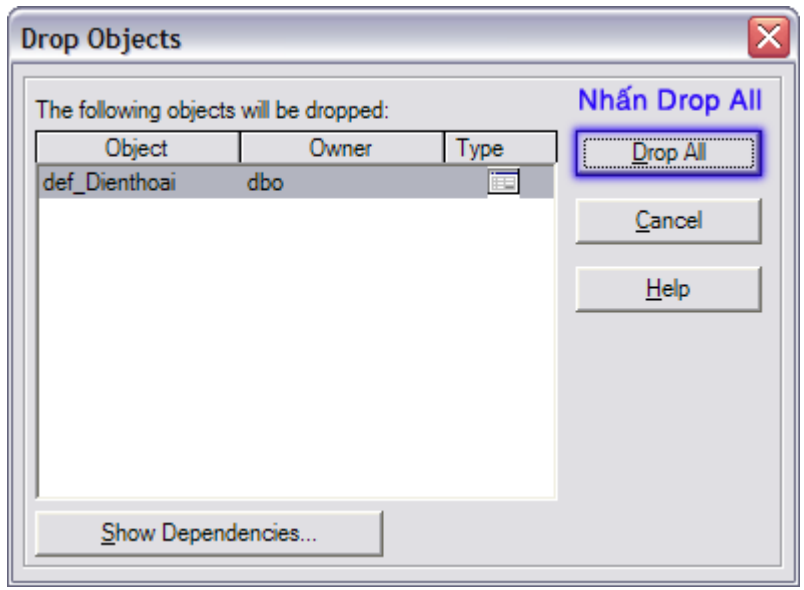
- Gỡ bỏ giá trị mặc định

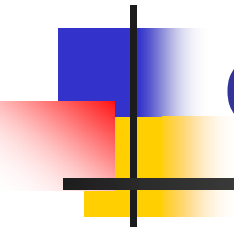
```
EXEC  sp_unbindefault  Tên_đối_tượng
```

- Xoá giá trị mặc định
  - Phải gỡ bỏ giá trị mặc định khỏi tất cả các cột trước khi xoá

```
DROP DEFAULT
Tên_gt_mặc_định  [, ...]
```

`DROP DEFAULT def_Dienthoai`





Chương 4

MỘT SỐ HÀM THƯỜNG DÙNG

Các hàm chuyển đổi kiểu dữ liệu

- Một hàm của SQL Server có thể sử dụng ở bất cứ đâu thay cho một giá trị cụ thể

- Đổi kiểu dữ liệu
  - `CAST (Biểu_thức AS Kiểu_dữ_liệu)`
- Đổi kiểu dữ liệu và định dạng
  - `CONVERT (Kiểu_dữ_liệu, Biểu_thức [, Định_dạng])`
- Đổi một số thành chuỗi
  - `STR (Số_thực, Số_ký_tự [, Số_lẻ])`

Các hàm ngày giờ

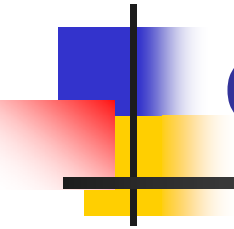
- Cộng ngày
  - **DATEADD (Đơn_vị, Con_số, Ngày_chỉ_định)**
- So sánh hai biến ngày
  - **DATEDIFF (Đơn_vị, Ngày1, Ngày2)**
- Lấy tên ngày, tháng, năm
  - **DATENAME (Đơn_vị, Ngày)**
- Thời điểm hiện hành
  - **GETDATE()**
- Lấy một thành phần ngày, giờ trong biến ngày
  - **DATEPART (Đơn_vị, Ngày)**
- Lấy ngày, tháng, năm của biến ngày
  - **DAY (Ngày)**
  - **MONTH (Ngày)**
  - **YEAR (Ngày)**

Các hàm toán học

- Lấy trị tuyệt đối
  - `ABS (Biểu_thức_số)`
- Hằng số Pi
  - `PI()`
- Luỹ thừa
  - `POWER (Biểu_thức_số, Số_mũ)`
- Lấy số ngẫu nhiên
  - `RAND ([Số_nguồn])`
- Làm tròn số
  - `ROUND (Biểu_thức_số, Vtrí_làm_tròn)`
- Dấu của kết quả biểu thức
  - `SIGN (Biểu_thức_số)`
- Lấy căn bậc 2
  - `SQRT (Biểu_thức_số)`

Các hàm xử lý chuỗi

- Hàm viết hoa, thường
  - **UPPER (Chuỗi), LOWER (Chuỗi)**
- Hàm cắt chuỗi
  - **LEFT (Chuỗi nguồn, Số_ktự), RIGHT (Chuỗi nguồn, Số_ktự)**
  - **SUBSTRING (Chuỗi nguồn,Vị_trí,Số_ktự)**
- Hàm cắt khoảng trắng, tạo chuỗi khoảng trắng
  - **LTRIM (Chuỗi), RTRIM (Chuỗi), SPACE (N)**
- Hàm tạo chuỗi lặp
  - **REPLICATE (Chuỗi_lặp, N)**
- Chiều dài chuỗi
  - **LEN (Chuỗi)**
- Đảo chuỗi
  - **REVERSE (Chuỗi)**
- Tìm và thay thế chuỗi
  - **REPLACE (Chuỗi nguồn, Chuỗi_tìm, Chuỗi_thay_thế)**
- Đổi từ số thành ký tự và ngược lại
  - **CHAR (Số) , ASCII(Ký_tự)**



Chương 5

LỆNH TRUY VẤN DỮ LiỆU

Toán tử số học

Ký hiệu	Ý nghĩa
+	Thực hiện phép cộng hai số.
-	Thực hiện phép trừ hai số.
*	Thực hiện phép nhân hai số.
/	Thực hiện phép chia hai số.
%	Thực hiện phép chia lấy phần dư.

Toán tử nối chuỗi

- Sử dụng dấu + làm toán tử nối chuỗi

```
SELECT 'Hello' + ' ' + 'The World!'

SELECT 'Ngày đặt hàng D007 là: '
      + CAST(NGAYDH AS CHAR(11))
  FROM DONDH
  WHERE SODH='D007'
```

Toán tử so sánh

Ký hiệu	Ý nghĩa
=	Thực hiện phép so sánh bằng.
>	Thực hiện phép so sánh lớn hơn.
<	Thực hiện phép so sánh nhỏ hơn.
>=	Thực hiện phép so sánh lớn hơn hoặc bằng.
<=	Thực hiện phép so sánh nhỏ hơn hoặc bằng.
<>	Thực hiện phép so sánh khác.
!=	Thực hiện phép so sánh khác.
!>	Thực hiện phép so sánh không lớn hơn.
!<	Thực hiện phép so sánh không nhỏ hơn.

Toán tử luận lý

- Sử dụng các toán tự thông thường AND, OR, NOT vẫn dùng trong các câu SQL

```
SELECT * FROM VATTU
WHERE (DVTINH='Bộ'  AND  PHANTRAM>10)
        OR (DVTINH='Cái'  AND  PHANTRAM>20)
```

Lệnh SELECT FROM

```
SELECT  Danh_sách_các_cột | Hàm_thống_kê  AS  Bí_danh
FROM Tên_bảng
[ WHERE Điều_kiện_lọc ]
GROUP BY Danh_sách_cột_nhómdl
HAVING Điều_kiện_lọc_nhóm
[ ORDER BY Tên_cột [DESC] [, ...] ]
```

- Khi lấy dữ liệu từ nhiều bảng
  - Các cách kết hợp bảng: INNER|LEFT|RIGHT|FULL JOIN
- Khi sử dụng các hàm tính toán thống kê
  - Phải dùng GROUP BY
  - Điều kiện dựa trên kết quả thống kê phải đặt trong HAVING
- Sử dụng UNION để kết hợp nhiều câu SELECT

Một số mệnh đề khác trong SELECT

- Select Into
  - Chép dữ liệu ra bảng mới
  - Chỉ chạy được 1 lần, ở lần sau bảng đã tồn tại thì sẽ gây ra lỗi
- Mệnh đề tổng hợp dữ liệu cuối nhóm
  - Compute
  - Compute By
  - Sử dụng chung với các hàm thống kê SUM, COUNT, MAX, MIN, AVG
  - Với Compute By, phải có mệnh đề ORDER BY đi kèm

- SELECT MSSV, MAKHOAHOC
- FROM KETQUA
- **ORDER BY MSSV**
- **COMPUTE COUNT(MAKHOAHOC) BY MSSV**

Truy vấn con

- Tạo ra một tập hợp dữ liệu để sử dụng trong các mệnh đề khác của câu truy vấn, thường là WHERE
  - Nằm trong ngoặc ( )
  - Chỉ được phép dùng một cột hoặc một biểu thức sẽ trả về giá trị trong mệnh đề SELECT
  - Có thể trả về là một giá trị đơn lẻ hoặc một danh sách các giá trị
  - Cấp độ lồng nhau không giới hạn
- Các từ khoá điều kiện thường dùng
  - IN, ALL, ANY

```
SELECT TENNHACC, DIENTHOAI
FROM NHACC
WHERE MANHACC IN
  (SELECT MANHACC FROM DONDH
     WHERE CONVERT(CHAR(7), NGAYDH, 21)='2002-01')
```

Biểu thức CASE dạng đơn giản

```
CASE Biểu_thức
  WHEN  Giá_trị_1  THEN   Biểu_thức_kết_quả_1
  [WHEN  Giá_trị_2  THEN   Biểu_thức_kết_quả_2
      ...]
  [ ELSE Biểu_thức_kết_quả_N]
END
```

- Giá trị 1, giá trị 2
  - Các giá trị cụ thể để so sánh bằng (=) với biểu thức
- Biểu thức kết quả 1, biểu thức kết quả 2
  - Biểu thức sẽ được trả về khi việc so sánh của biểu thức bằng với các giá trị so sánh tương ứng

Ví dụ

```
SELECT LOAI=
      CASE LEFT(MAVTU, 2)
         WHEN 'DD' THEN 'Đầu DVD'
         WHEN 'VD' THEN 'Đầu VCD'
         WHEN 'TV' THEN 'Tivi'
         WHEN 'TL' THEN 'Tủ lạnh'
         WHEN 'BI' THEN 'Bia lon'
         WHEN 'LO' THEN 'Loa thùng'
         ELSE 'Chưa phân loại'
      END,
      MAVTU, TENVTU, DVTINH
FROM VATTU
ORDER BY LEFT(MAVTU, 2)
COMPUTE COUNT(MAVTU) BY LEFT(MAVTU, 2)
```

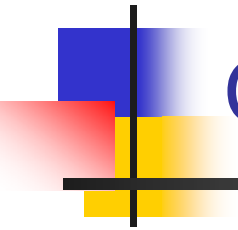
Biểu thức CASE dạng tìm kiếm

```
CASE
  WHEN  Bt_logic_1   THEN   Biểu_thức_kết_quả_1
  [WHEN  Bt_logic_2   THEN   Biểu_thức_kết_quả_2
       ...]
  [ ELSE Biểu_thức_kết_quả_N]
END
```

- Biểu thức logic1, biểu thức logic2
  - Các biểu thức luận lý dùng để thực hiện các phép so sánh trong biểu thức CASE.
- Biểu thức kết quả 1, biểu thức kết quả 2
  - Biểu thức sẽ được trả về khi một trong các biểu thức luận lý so sánh có kết quả là đúng.

Ví dụ

```
SELECT GHICHU=
    CASE
        WHEN PHANTRAM <20 THEN 'Lời ít'
        WHEN PHANTRAM BETWEEN 20 AND 40 THEN 'Lời
  nhiều'
        ELSE 'Rất lời'
    END,
    TENVTU, DVTINH, PHANTRAM
FROM VATTU
ORDER BY PHANTRAM
```



Chương 6

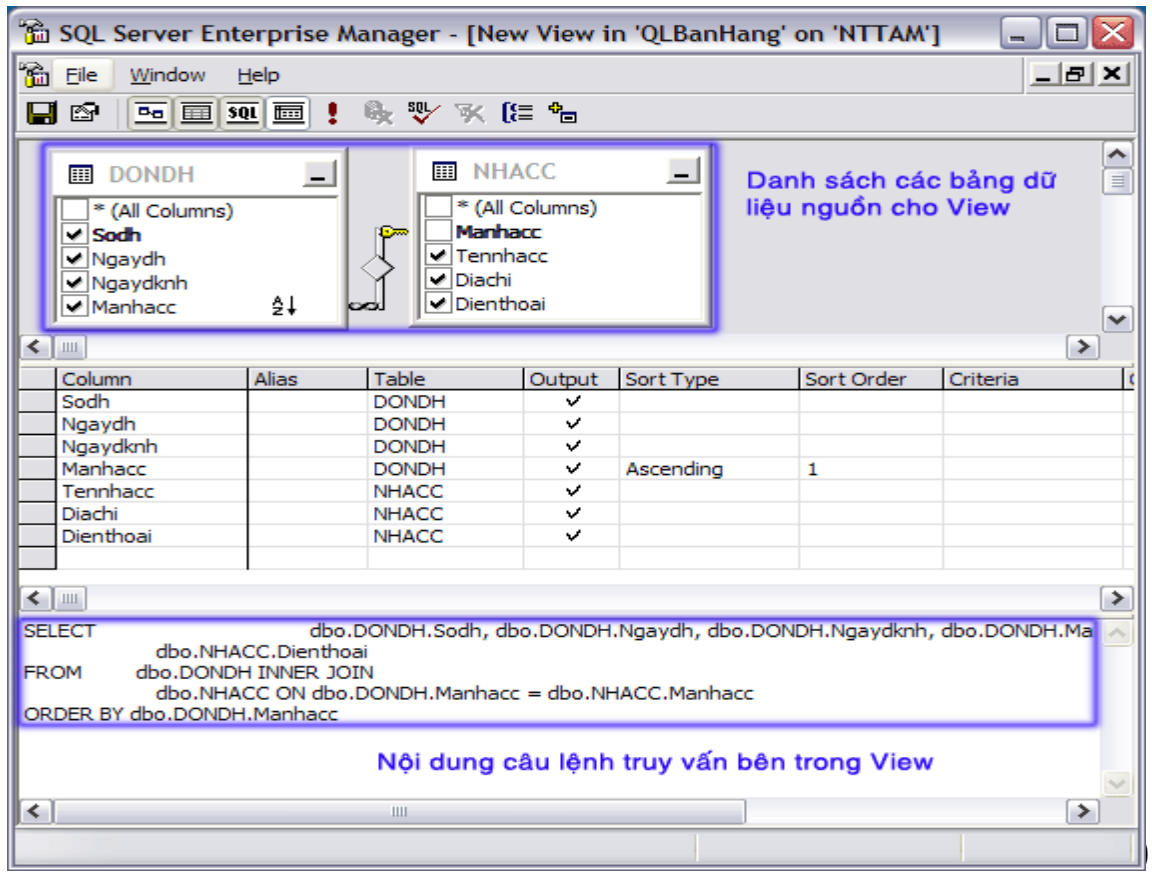
VIEW

Khái niệm

- Được xây dựng từ câu truy vấn SELECT để hiển thị dữ liệu từ một hay nhiều bảng
  - Tập hợp dữ liệu, thể hiển cùng một dữ liệu theo nhiều cách khác nhau
  - Làm việc tương tự như một bảng nhưng không lưu trữ dữ liệu
  - Cho phép thêm/xoá/sửa
  - Bảo mật dữ liệu, bảo mật nội dung câu truy vấn dữ liệu
- Một số hạn chế trong câu lệnh SELECT
  - Order By
  - Computed, Computed By

Tạo bảng ảo bằng Enterprise Manager

- Đơn giản, công cụ tự động phát sinh câu lệnh
- Có thể chuyển về dạng viết lệnh SQL



Xem và cập nhật dữ liệu trên VIEW

- Làm việc như một bảng thông thường
  - Sử dụng câu SELECT để xem dữ liệu

  Select * From **vw_DonDH**
- Sử dụng INSERT/UPDATE để cập nhật dữ liệu
  - Chỉ có thể cập nhật vào một bảng
  - Để INSERT dữ liệu vào bảng, view phải thỏa mãn các yêu cầu về khóa, ràng buộc khóa ngoại, các cột NOT NULL, các cột tính toán, order by, group by, distinct
- Sử dụng Delete để xoá dữ liệu
  - View tạo từ hai hay nhiều bảng không thể xoá
- Có thể xây dựng các trigger trên bảng ảo

Cập nhật dữ liệu qua VIEW

- View có nhiều hạn chế khi thực hiện cập nhật dữ liệu
  - Group By, Order By, Distinct
  - Thiếu cột khoá
  - Ràng buộc toàn vẹn
- SQL Server phiên bản 2000 cung cấp loại trigger INSTEAD OF
  - Cơ chế tương tự như trigger thông thường
  - Mở rộng khả năng cập nhật, tính toán dữ liệu, đặc biệt với bảng ảo
  - Xem thêm phần TRIGGER

Tạo mới view bằng CREATE VIEW

```
CREATE VIEW Tên_bảng_ảo
[(Tên_các_cột)]
[WITH ENCRYPTION]
AS Câu_lệnh_SELECT
[WITH CHECK OPTION]
```

- Tên các cột: sử dụng trong bảng ảo khi tham chiếu đến các cột
- WITH ENCRYPTION: mã hóa nội dung câu lệnh SELECT
- WITH CHECK OPTION: không cho cập nhật dữ liệu không thoả điều kiện của mệnh đề WHERE trong câu lệnh SELECT

```
CREATE VIEW vw_DONDH_NHACC
AS
SELECT DONDH.*, NHACC.Diachi AS Diachi, NHACC.Tennhacc
  AS Hoten    FROM DONDH INNER JOIN NHACC ON
  DONDH.Manhacc = NHACC.Manhacc
```

Sửa đổi nội dung View

```
ALTER  VIEW Tên_bảng_ảo
   [(Tên_các_cột)]
   [WITH ENCRYPTION]
AS
   Câu_lệnh_SELECT_mới
   [WITH CHECK OPTION]
```

- Tương tự như xoá bảng rồi tạo lại

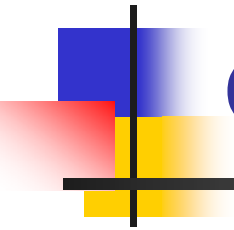
```
DROP VIEW Tên_bảng_ảo
Go
CREATE  VIEW Tên_bảng_ảo
   [(Tên_các_cột)]
   [WITH ENCRYPTION]
AS
   Câu_lệnh_SELECT_mới
   [WITH CHECK OPTION]
```

VD: thêm dữ liệu vào view

- Với view được tạo như sau:

```
CREATE VIEW vSinhVien AS
SELECT      MASV, TENSV, PHAI, DIACHI,
            SV.MALOP AS LOP, L.MALOP, TENLOP
FROM SINHVIEN SV INNER JOIN LOP L ON
      SV.MALOP=L.MALOP
```

- Lệnh
  INSERT INTO vSinhVien (MASV, TENSV, PHAI, DIACHI, LOP) VALUES (1,'AAA','NU','123','A1')
  sẽ thêm record mới vào table SinhVien



Chương 7

LẬP TRÌNH TRONG SQL SERVER

Khai báo biến cục bộ

- Dùng để lưu trữ các giá trị tạm thời trong quá trình tính toán
  - Biến phải có kiểu dữ liệu
  - Biến muốn sử dụng trong một batch phải khai báo trước

```
DECLARE @Tên_biến  Kiểu_dữ_liệu [, ...]
```

```
DECLARE  @Tongsldat  INT,  @Hotenncc  CHAR(50)
DECLARE  @Ngayxh  DATETIME
```

Gán giá trị cho biến

- Sử dụng lệnh SET hoặc SELECT

```
SET @Biến = Giá_trị
```

```
SET @a = 5
```

```
Select @Biến = Tên_Cột From Tên_Bảng
```

```
Select @TRPHG=TRPHG FROM PHONGBAN WHERE
  MAPB='P1'
```

Xem giá trị hiện hành của biến

- Lệnh Print

```
Print @Biến
Print @A
```

- Khi có kết hợp với chuỗi, phải đổi kiểu dữ liệu sang kiểu chuỗi bằng hàm CAST hay CONVERT

```
Print 'Giá trị của @A ' + cast(@A as char(4))
```

Phạm vi hoạt động của biến

- Một biến chỉ có phạm vi hoạt động cục bộ
  - Trong một Batch
  - Trong một Stored Procedure hay Trigger

```
DECLARE @Ngaydhgn DATETIME
SELECT @Ngaydhgn=MAX(NGAYDH)
FROM DONDH
GO
PRINT 'Ngày đặt hàng gần nhất: ' +
  CONVERT(CHAR(12),@Ngaydhgn)--CAST(@NGAYDHGN AS
  CHAR(12))
GO
```

Ý nghĩa sử dụng

- Cung cấp các thông tin hệ thống như
  - Phiên bản SQL Server
  - Số dòng dữ liệu vừa được xử lý bởi câu lệnh
  - Mã lỗi
  - Số lượng kết nối
  - Tình trạng cursor
  - …
- Không cần khai báo
  - Biến do SQL Server định sẵn
  - Tên bắt đầu bởi @@

Một vài biến hệ thống thường dùng

- RowCount
  - Tổng số mẩu tin được tác động của câu lệnh truy vấn gần nhất.
- Error
  - Số mã lỗi của câu lệnh thực hiện gần nhất
  - Khi một câu lệnh thực hiện thành công thì giá trị là 0.
- Fetch_Status
  - Trạng thái của việc đọc dữ liệu trong bảng theo cơ chế từng mẩu tin (cursor).
  - Khi đọc dữ liệu của mẩu tin thành công thì giá trị là 0.

Cấu trúc rẽ nhánh IF...ELSE

```
IF Biểu_thức_luận_lý
  Câu_lệnh1 | Khối_lệnh1
[ ELSE
  Câu_lệnh2 | Khối_lệnh2 ]
```

```
IF (SELECT COUNT(*) FROM CTPXUAT
      WHERE SLXUAT>4) > 0
BEGIN
   PRINT 'Danh sách các hàng hóa bán với số lượng > 4'
   SELECT CTPX.MAVTU, TENVTU, SLXUAT
      FROM CTPXUAT CTPX INNER JOIN VATTU VT
      ON VT.MAVTU=CTPX.MAVTU
      WHERE SLXUAT>4
END
ELSE
   PRINT 'Chưa bán hàng hóa nào với số lượng >4'
```

Cú pháp If Exists

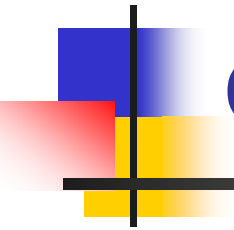
```
IF EXISTS (Câu_lệnh_SELECT)
  Câu_lệnh1 | Khối_lệnh1
[ ELSE
  Câu_lệnh2 | Khối_lệnh2 ]
```

```
IF EXISTS (SELECT * FROM CTPXUAT WHERE SLXUAT>4)
BEGIN
      PRINT 'Danh sách các hàng hóa bán với số lượng
  > 4'
      SELECT CTPX.MAVTU, TENVTU, SLXUAT
      FROM CTPXUAT CTPX INNER JOIN VATTU VT ON
  VT.MAVTU=CTPX.MAVTU
      WHERE SLXUAT>4
END
ELSE
      PRINT 'Chưa bán hàng hóa nào với số lượng >4'
```

Cấu trúc lặp WHILE

```
WHILE  Biểu_thức_luận_lý
BEGIN
  Các_lệnh_lặp
END
```

```
DECLARE @Songuyen INT
SET @Songuyen=100
WHILE (@Songuyen<110)
BEGIN
   PRINT 'Số nguyên : ' + CONVERT(CHAR(3),
  @Songuyen)
   SET @Songuyen = @Songuyen + 1
END
```



Chương 8

CURSOR

Khái niệm về cursor

- Các lệnh của SQL Server làm việc trên một nhóm nhiều mẩu tin
- Cursor là cấu trúc giúp làm việc với từng mẩu tin tại một thời điểm
  - Khai báo cursor như một câu lệnh SELECT
  - Có thể di chuyển giữa các mẩu tin trong cursor để làm việc
  - Có thể dùng cursor để cập nhật dữ liệu (Update, Delete)

Các bước sử dụng kiểu dữ liệu cursor

- Định nghĩa biến kiểu cursor bằng lệnh DECLARE
  - Có hai loại cursor: Local, Global
  - Cách di chuyển mẩu tin trong cursor: Forward only, scroll
  - Cách quản lý dữ liệu của cursor: static, dynamic, keyset
- Sử dụng lệnh OPEN để mở ra cursor đã định nghĩa trước đó
- Đọc và xử lý trên từng dòng dữ liệu bên trong cursor
  - Sử dụng biến @@Fetch_status
  - Các lệnh Fetch và cấu trúc while
- Đóng cursor lại bằng lệnh CLOSE và DEALLOCATE
  - Sau khi close, có thể mở lại
  - Deallocate: hủy cursor khỏi bộ nhớ

Cú pháp Declare

```
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]]
```

```
DECLARE cur_Vattu CURSOR
DYNAMIC
FOR
    SELECT * FROM VATTU
```

Cú pháp Open

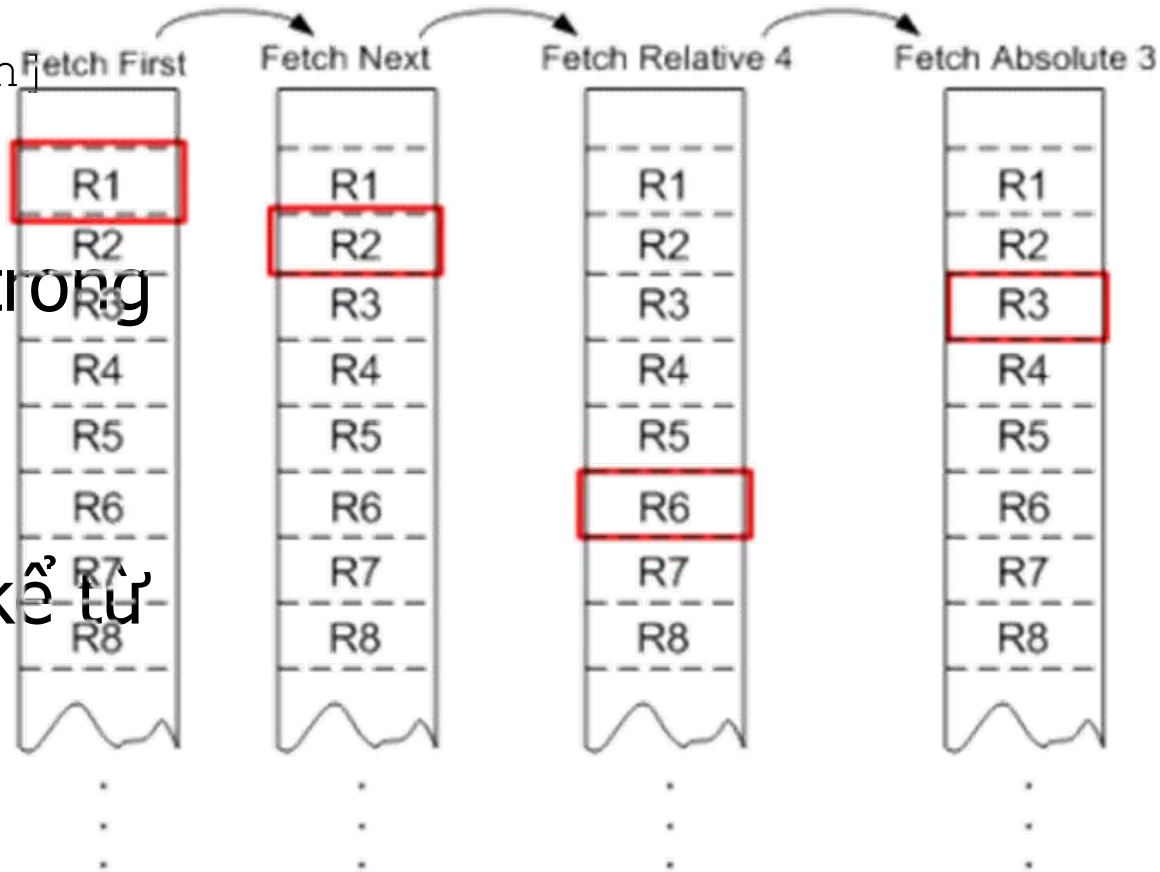
```
OPEN Tên_cursor
```

```
DECLARE cur_Vattu CURSOR
DYNAMIC
FOR
    SELECT * FROM VATTU

OPEN cur_Vattu
```

Cú pháp FETCH

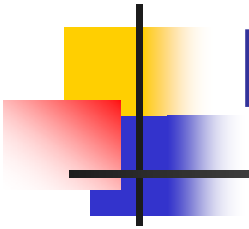
```
FETCH  [NEXT | PRIOR | FIRST | LAST
| ABSOLUTE n   | RELATIVE n]
  FROM Tên_cursor
  [INTO Danh_sách_biến]
```

- Absolute n:
  - Đọc dòng thứ n trong cursor
- Relative n:
  - Đọc dòng thứ n kể từ vị trí hiện hành



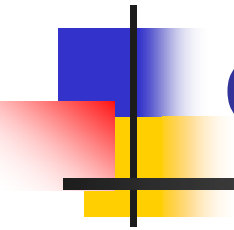
Ví dụ hoàn chỉnh

```
--1. Khai báo biến cursor
DECLARE cur_Vattu CURSOR KEYSET
FOR
      SELECT * FROM VATTU
      WHERE MAVTU LIKE 'TV%'
      ORDER BY MAVTU
--2. Mở cursor
OPEN cur_Vattu
--3. Đọc dữ liệu
FETCH NEXT FROM cur_Vattu
WHILE @@FETCH_STATUS = 0
BEGIN
      -- Xử lý dòng mới vừa đọc được
      -- Thực hiện đọc tiếp các dòng kế
      FETCH NEXT FROM cur_Vattu
END
--4. Đóng cursor
CLOSE cur_Vattu
DEALLOCATE cur_Vattu
```



Bài tập

- Sử dụng CURSOR thực hiện các yêu cầu sau:
  - Đếm số lượng sinh viên
  - Đếm số lượng môn học có trên 2 sinh viên
  - Liệt kê và đếm số lượng sinh viên có ĐTB >= 5
  - Cập nhật giá trị cho thuộc tính DTB trên table SinhVien



Chương 9

STORED PROCEDURE

(THỦ TỤC NỘI TẠI)

Thủ tục nội tại là gì?

- Là một tập hợp các
  - Dòng lệnh, biến
  - Cấu trúc điều khiển
  - Các tham số đầu vào, ra
- Lưu trữ trong database tại server
  - Dùng để thực hiện các xử lý cần nhiều thao tác với dữ liệu
  - Có thể được gọi thực hiện từ client, trả kết quả về thông qua
    - Các tham số đầu ra
    - Một cursor gồm nhiều dòng dữ liệu (Recordset)
- Tính hiệu quả
  - Dễ dàng thao tác với dữ liệu, sử dụng T-SQL để lập trình
  - Tăng tốc độ ứng dụng: thực thi tại server, biên dịch 1 lần,…

Các thủ tục nội tại hệ thống

- SQL Server bao gồm rất nhiều các thủ tục nội tại có sẵn gọi là thủ tục nội tại của hệ thống
  - Tên bắt đầu bằng sp_...
  - Thực hiện các chức năng quản trị database và server tương tự các thao tác trên Enterprise Manager
  - Thông thường được lưu trữ tại database **Master**

```
EXEC sp_addlogin 'TTTH', 'T3HNVC', 'NorthWind'
```

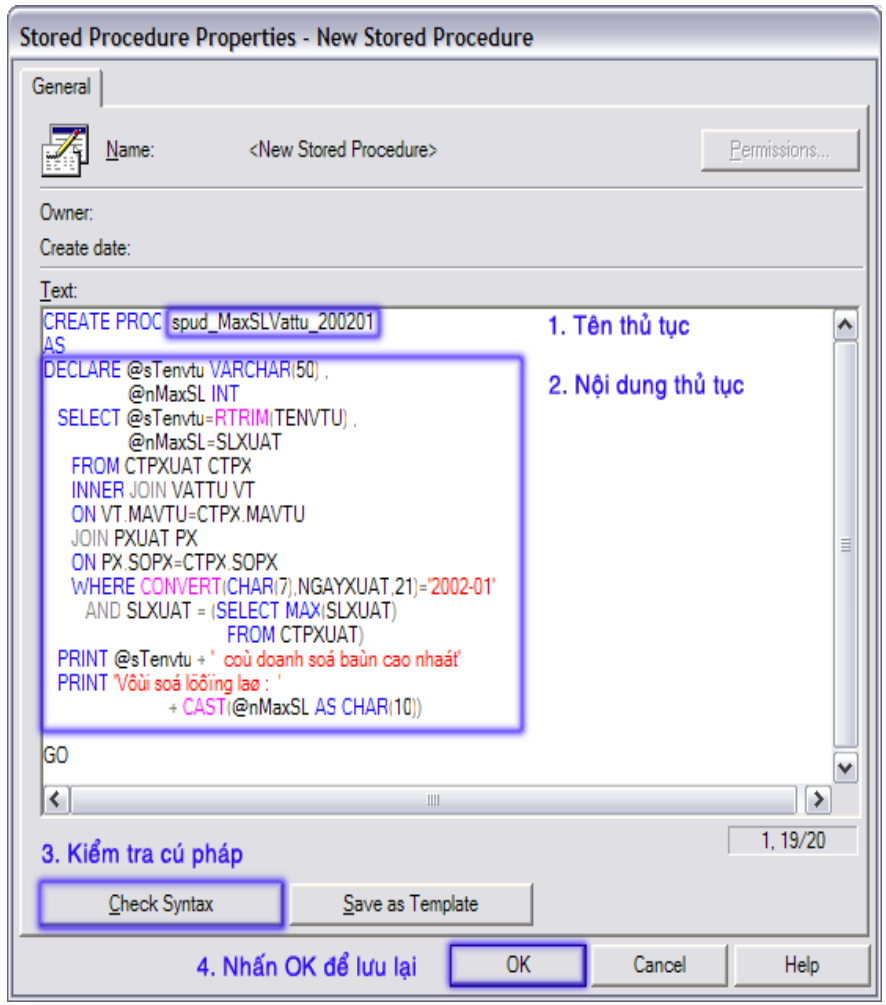
Các lợi ích khi sử dụng thủ tục nội tại

- Tốc độ xử lý
  - Nhanh vì nội dung của thủ tục được lưu trữ và thực hiện ngay tại máy chủ
  - Dữ liệu cũng được lưu trữ trên máy chủ nên không mất thời gian truyền dữ liệu qua hệ thống mạng
- Tổ chức, phân chia các xử lý tại máy chủ hoặc tại máy trạm
  - Giảm thời gian xây dựng ứng dụng
  - Tăng khả năng tái sử dụng
- Bảo mật
  - Sử dụng thủ tục để hạn chế quyền hạn của người dùng nhưng vẫn đảm bảo cung cấp đầy đủ các dữ liệu và khả năng cập nhật dữ liệu

Tạo mới một thủ tục nội tại

- Tạo bằng Enterprise Manager
- Tạo bằng Script

```
CREATE PROC[EDURE] Tên_TT
AS
[DECLARE   Biến_cục_bộ]
  Các_lệnh
Go
```



Gọi thực hiện thủ tục nội tại

- Coi như một lệnh T-SQL, chỉ gọi thực hiện được bằng script

```
EXEC[UTE]   Tên_Thủ_Tục [Tham_số]

EXE   spud_MaxSLVattu_200201
```

- Chú ý
  - Nếu lệnh gọi thủ tục là dòng đầu tiên của batch thì không cần từ khoá EXEC
  - Giá trị của các tham số có thể là các biến
  - Biến nhận giá trị trả về phải có từ khoá **output**

Thay đổi nội dung của thủ tục nội tại

- Dùng Enterprise Manager
- Dùng Script

```
ALTER   PROC[EDURE] Tên_thủ_tục
AS
[DECLARE   Biến_cục_bộ]
Các_lệnh
```

- Lợi điểm của script
  - Tái sử dụng lại để cập nhật vào các database khác
  - Quản lý được quá trình thay đổi của thủ tục

Hủy bỏ thủ tục nội tại

- Sử dụng Enterprise Manager
- Sử dụng script

```
DROP PROC[EDURE]    Tên_thủ_tục
```

```
DROP PROC spud_MaxSLVattu_200201
```

Tham số đầu vào

- Cho phép truyền vào các thông tin cần cho những xử lý bên trong một thủ tục

```
CREATE   PROC[EDURE] Tên_thủ_tục
@Tên_tham_số  Kiểu_dữ_liệu   [=Giá_trị]  [, ...]
AS
…
```

- Khi gọi thực hiện, tham số đầu vào có thể truyền qua
  - Giá trị cụ thể
  - Biến

Ví dụ

```
CREATE PROC spud_TongTGXuat
@sSopx  CHAR(4)
AS
DECLARE @nTongTG MONEY
   SELECT @nTongTG=SUM(SLXUAT*DGXUAT)
      FROM CTPXUAT
      WHERE @sSopx=SOPX
   PRINT 'Trị giá phiếu xuất '
      + CAST(@sSopx AS CHAR(4))
   PRINT 'Là : '
      + CAST(@nTongTG AS VARCHAR(15))
GO

EXEC spud_TongTGXuat 'X001'
EXEC spud_TongTGXuat @sSopx='X001'
```

Tham số đầu ra

- Giúp nhận các kết quả xử lý mà thủ tục trả về

```
CREATE   PROC[EDURE] Tên_thủ_tục
@Tên_tham_số  Kiểu_dữ_liệu   OUTPUT  [, ...]
AS
…
```

- Khi gọi thực hiện, tham số đầu ra
  - Là biến đã được khai báo trước
  - Phải có từ khoá **output** đi kèm

Ví dụ

```
ALTER PROC spud_TinhSLDat
@sSodh  CHAR(4), @sMavtu  CHAR(4), @nSldat  INT  OUTPUT
AS
    IF NOT EXISTS(SELECT SODH FROM CTDONDH WHERE SODH=@sSodh AND
  MAVTU=@sMavtu)
    BEGIN
        PRINT 'Xin xem lại số đặt hàng, mã vật tư!'
        RETURN
    END
    SELECT @nSldat=SLDAT FROM CTDONDH WHERE SODH=@sSodh AND
  MAVTU=@sMavtu
GO
```

Gọi thực hiện thủ tục

```
DECLARE @nSLdathang INT
EXEC spud_TinhSLDat
@sSodh='D001',@sMavtu='DD02',
                @nSLdathang=@nSlDat
OUTPUT
PRINT 'Đơn đặt hàng D001 với vật tư DD02'
PRINT 'Có số lượng đặt là: ' +
CAST(@nSLdathang AS VARCHAR(10))
```

Kết quả trả về

Đơn đặt hàng D001 với vật tư DD02

Có số lượng đặt là: 15

Mã hóa nội dung thủ tục

```
CREATE|ALTER  PROC[EDURE] Tên_thủ_tục
[@Tên_tham_số  Kiểu_dữ_liệu   [OUTPUT] [, ...] ]
WITH  ENCRYPTION
…
```

- Bảo mật nội dung xử lý của thủ tục
- Sau khi mã hoá, không thể xem lại nội dung của thủ tục bằng Enterprise Manager hay bất cứ lệnh nào

Biên dịch thủ tục

```
CREATE|ALTER  PROC[EDURE] Tên_thủ_tục
[@Tên_tham_số  Kiểu_dữ_liệu   OUTPUT  [, ...] ]
WITH  RECOMPILE  [ ENCRYPTION ]
AS
…
```

- Thông thường: thủ tục chỉ được dịch và lập kế hoạch thực hiện ở lần gọi đầu tiên
- With Recompile
  - Yêu cầu SQL Server dịch và lập kế hoạch thực hiện lại trước khi chạy
  - Chỉ nên dùng khi thủ tục làm việc với dữ liệu của bảng bị cập nhật nhiều

Thủ tục lồng nhau

- Trong một thủ tục có thể gọi thực hiện thủ tục khác
- Cấp độ lồng nhau tối đa là 32 cấp
  - Biến @@NestLevel cho biết cấp độ lồng hiện hành

Sử dụng lệnh RETURN trong thủ tục

- Là một cách để trả về kết quả xử lý mà không dùng tham số đầu ra
  - Chỉ có thể trả về kết quả kiểu số nguyên
  - Kết quả trả về được nhận bởi 1 biến

    EXEC **@Biến=**Tên_thủ_tục [Các_tham_số]
  - Giá trị mặc định trả về là 0
- Sau khi gọi return, thủ tục sẽ chấm dứt xử lý

Ví dụ

```
CREATE PROC spud_TinhTongSLdat
@sManhacc CHAR(3) , @sMavtu
  CHAR(4) , @nTongSLdat INT
  OUTPUT
AS
    IF NOT EXISTS(SELECT * FROM
  VATTU WHERE MAVTU=@sMavtu)
        RETURN 1
    IF NOT EXISTS(SELECT * FROM
  NHACC WHERE MANHACC=@sManhacc)
        RETURN 2
    SELECT @nTongSLdat=SUM(SLDAT)
  FROM DONDH DH INNER JOIN
      CTDONDH CTDH ON
  DH.SODH=CTDH.SODH WHERE
      MANHACC=@sManhacc AND
  MAVTU=@sMavtu
    IF @nTongSLdat IS NULL
        SET @nTongSLdat=0
    RETURN
```

```
--Gọi thực hiện thủ tục
DECLARE @nTongSLdat INT, @nKetqua
  INT
EXEC @nKetqua=spud_TinhTongSLdat
  'C02', 'TV14', @nTongSLdat OUTPUT
IF @nKetqua=1
    PRINT 'Mã vật tư không đúng'
ELSE
  IF @nKetqua=2
  PRINT 'Mã nhà cung cấp không
  đúng'
  ELSE
  PRINT 'Tổng số lượng đặt là: ' +
  CAST(@nTongSLdat AS VARCHAR(10))
```

Sử dụng bảng tạm trong thủ tục

- Không thể tạo bảng ảo (create view) trong thủ tục
- Khi việc xử lý trở nên phức tạp, cần dùng bảng tạm để thực hiện
  - Tạo bảng tạm bằng lệnh SELECT INTO #Tên_Bảng
- Bảng tạm
  - Được lưu trong database TempDB
  - Gồm hai loại: cục bộ (#) và toàn cục (##)
  - Bảng tạm cục bộ chỉ tồn tại trong thủ tục, bị hủy sau khi thủ tục kết thúc xử lý

Tham số kiểu cursor bên trong thủ tục

- Thủ tục có thể tính toán để trả về một cursor
- Sử dụng từ khoá VARYING OUTPUT để khai báo
- Các bước thực hiện
  - Trong thủ tục
    - Khai báo cursor
    - Mở cursor
  - Sau khi gọi thủ tục
    - Đọc dữ liệu từ cursor để xử lý
    - Đóng cursor

Ví dụ

```
CREATE PROC spud_TinhDsoban
    @sNamThang CHAR(6), @cur_Dsvtu CURSOR
    VARYING OUTPUT
AS
    --Tạo bảng tạm tính ra tổng số lượng bán
    SELECT CTX.MAVTU, SUM(SLXUAT) AS
    TONGSLBAN INTO #tab_TongSLBan
    FROM CTPXUAT CTX INNER JOIN VATTU VT ON
    VT.MAVTU = CTX.MAVTU INNER JOIN PXUAT PX
    ON PX.SOPX = CTX.SOPX WHERE
    CONVERT(CHAR(6), NGAYXUAT, 112)=
    @sNamThang
    GROUP BY CTX.MAVTU
    --Kiểm tra dữ liệu có phát sinh?
    IF EXISTS (SELECT MAVTU FROM
    #tab_TongSLBan)
    BEGIN
--B1: Khởi tạo giá trị biến CURSOR
        SET @cur_Dsvtu = CURSOR
  FORWARD_ONLY
        FOR
            SELECT MAVTU, TONGSLBAN
            FROM #tab_TongSLBan
            WHERE  TONGSLBAN = (
  SELECT MAX(TONGSLBAN) FROM
  #tab_TongSLBan)
        --B2: Mở cursor ra
        OPEN @cur_Dsvtu
        DROP TABLE #tab_TongSLBan
        RETURN 0
  END
  ELSE
        -- Khi không có dữ liệu phát sinh
        DROP TABLE #tab_TongSLBan
    RETURN 1
GO
```

Ví dụ

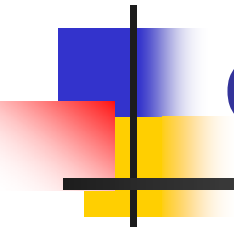
```
DECLARE @cur_Dsvt CURSOR, @nGttv INT, @sMavtu CHAR(4),  @nTongslBan INT
--Gọi thực hiện thủ tục
EXEC  @nGttv  = spud_TinhDsoban '200203', @cur_Dsvt OUTPUT
--Xử lý tiếp sau đó
IF @nGttv =0
BEGIN
    PRINT 'Danh sách các vật tư'
    WHILE (0=0)
    BEGIN
        FETCH NEXT FROM @cur_Dsvt
        INTO @sMavtu, @nTongslBan
        IF @@FETCH_STATUS<>0
        BREAK
        PRINT 'Mã vật tư: ' + @sMavtu
        PRINT 'Tổng số lượng : ' +  CAST(@nTongslBan AS VARCHAR(10))
        PRINT REPLICATE('-', 50)
    END
END
ELSE
    PRINT 'Không có bán hàng trong năm tháng chỉ định'
```

Thủ tục cập nhật bảng dữ liệu

- Nhận các giá trị dữ liệu để cập nhật vào bảng
  - Đảm bảo được việc kiểm tra các ràng buộc trước khi cập nhật
  - Có thể cập nhật vào nhiều bảng, thực hiện các tính toán
  - Có thể gọi thực hiện từ các ngôn ngữ lập trình khác nhau (client) để xử lý các cập nhật dữ liệu phức tạp
  - Bảo mật được dữ liệu trong các bảng

Thủ tục hiển thị dữ liệu

- Sử dụng câu SELECT để trả về tập hợp các dòng dữ liệu
- Bổ sung khả năng cho bảng ảo
  - Truyền tham số
  - Có thể thực hiện các bước tính toán trước câu SELECT
- Hạn chế
  - Không thể sử dụng chung với SELECT như bảng ảo
- Sử dụng
  - Trả về bộ dữ liệu chỉ đọc, dùng để hiển thị trên màn hình hay báo biểu



Chương 10

TRANSACTION

(GIAO TÁC)

Khái niệm về giao tác

- Giao tác dùng để chỉ một công việc
  - Gồm nhiều bước
  - Các bước được thi hành lần lượt
  - Cả công việc sẽ thất bại nếu một trong các bước thực hiện bị thất bại
- Tính ACID
  - **A**tomic: Tất cả các bước được gói trong giao tác như một hành động duy nhất
  - **C**onsistency: Dữ liệu được đảm bảo toàn vẹn cho dù transaction có thành công hay không
  - **I**solation: Khi hai transaction thực hiện đồng thời, chúng được giữ độc lập để các kết quả không ảnh hưởng lẫn nhau
  - **D**urability: Sau khi transaction thực hiện thành công, dữ liệu được đảm bảo kể cả khi hệ thống bị sự cố

Giao tác không tường minh

- SQL Server chia các câu lệnh thực hiện làm hai loại
  - Giao tác tường minh
  - Giao tác không tường minh
- Mặc định, SQL Server thực hiện các lệnh ở chế độ giao tác không tường minh
  - Mỗi câu lệnh coi như một transaction: INSERT, UPDATE, DELETE,…
  - Sau khi thực hiện lệnh, các thay đổi dữ liệu sẽ được cập nhật ngay vào CSDL

Giao tác tường minh

- Giao tác tường minh là giao tác phải khai báo trước
  - Sử dụng từ khoá **Begin Tran**
  - Các lệnh theo sau thuộc vào giao tác đã khai báo
  - Tác dụng thay đổi dữ liệu được cập nhật ngay vào CSDL nhưng
    - Các giao tác khác không thấy được thay đổi này
    - Sẽ bị hủy bỏ khi phiên làm việc giữa client – server chấm dứt
- Kết thúc giao tác
  - Quá trình thực hiện lệnh, nếu bị lỗi có thể hủy bỏ giao tác bằng lệnh **Rollback Tran**
    - **Toàn bộ dữ liệu thay đổi đều bị hủy bỏ**
  - Khi các lệnh đã hoàn tất, kết thúc giao tác bằng **Commit Tran**
    - **Dữ liệu thay đổi được lưu lại**

Ví dụ

```
SET ANSI_WARNINGS OFF
GO
SELECT COUNT(*) AS 'Tổng vật tư trước khi thêm'  FROM VATTU
BEGIN TRAN
INSERT INTO VATTU (Mavtu, Tenvtu, Dvtinh, Phantram) VALUES ('BU01',
  'Bàn ủi PhiLip', 'Cái', 17)
SELECT COUNT(*) AS 'Tổng vật tư sau khi thêm trong giao tác' FROM
  VATTU
ROLLBACK TRAN
SELECT COUNT(*) AS 'Tổng vật tư hiện tại' FROM VATTU
SET ANSI_WARNINGS ON
```

Kết quả trả về

Tổng vật tư trước khi thêm

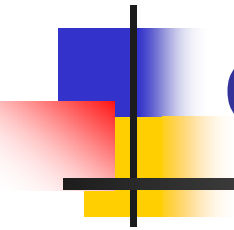
11

Tổng vật tư sau khi thêm trong giao tác

12

Tổng vật tư hiện tại

11



Chương 11

USER DEFINE FUNCTION

(HÀM DO NGƯỜI DÙNG TỰ ĐỊNH NGHĨA)

Làm việc với UDF

- Là đối tượng mới trong CSDL của SQL Server
- Mang đầy đủ tính chất của một hàm
  - Tham số
  - Giá trị trả về
  - Cách gọi thực hiện
- Hai nhóm hàm trong SQL Server
  - Hàm xác định
  - Hàm không xác định
    - Trả về giá trị thay đổi trong những lần gọi khác nhau dù giá trị tham số truyền vào giống nhau
    - VD: getdate()
  - Các hàm của SQL Server có sẵn là những hàm đơn trị

Các loại hàm do người dùng định nghĩa

- Chia làm hai loại chính
  - Hàm đơn trị (Scalar UDF)
  - Hàm trả về giá trị dạng bảng
    - Hàm đọc bảng (In-line table UDF): sử dụng một câu SELECT để đọc giá trị từ các bảng. Kết quả trả về là kết quả của câu SELECT
    - Hàm tạo bảng (Multi-statement table UDF): định nghĩa cấu trúc bảng kết quả, sử dụng lệnh INSERT để đưa dữ liệu vào bảng
- Ý nghĩa sử dụng
  - Hàm đơn trị
    - Dùng như các hàm thông thường sẵn có trong SQL Server
  - Hàm trả về giá trị bảng
    - Dùng thay cho bảng ảo, mở rộng khả năng truyền tham số khi đọc dữ liệu

Tạo mới UDF

```
CREATE FUNCTION [Tên_FUNCTION] Khai báo các tham số )
RETURNS Kiểu_dữ Liệu_trả_về
AS
BEGIN
  --Các câu lệnh bên trong FUNCTION--
  RETURN
END
```

- Tùy theo loại hàm mà cú pháp có thay đổi

Hàm đơn trị

```
CREATE FUNCTION F_NamThang(@d DateTime)
Returns char(6)
Begin
   Declare @strD char(6)
   Set @strD = Convert(char(6),@d,112)
   Return @strD
End
```

Hàm đọc bảng

```
CREATE FUNCTION F_DSHangHoa(@LoaiHH varchar(50))
Returns Table AS
  Return (Select * From DM_HANG_HOA where
  MaLoai_HH=@LoaiHH)

Go
```

Chú ý:

- Không có BEGIN, END
- Kết quả trả về là kiểu TABLE
- Câu lệnh SELECT có các hạn chế giống như với bảng ảo

Hàm tạo bảng

```
CREATE FUNCTION F_DSHangHoa(@LoaiHH varchar(50),
@PhanTram numeric)
RETURNS @DSHangHoa Table(Ma_HH varchar(50),Ten_HH
  varchar(50),
  DonGiaKhuyenMai numeric)
As
Begin
  Insert Into @DSHangHoa(Ma_HH,Ten_HH,DonGiaKhuyenMai)
  Select ID_HH,Ten_HH,DonGiaHienHanh
  From DM_HANG_HOA where IDLoai_HH=@LoaiHH

  Update @DSHangHoa
  Set DonGiaKhuyenMai=DonGiaKhuyenMai -
  (DonGiaKhuyenMai*@PhanTram)/100
  Return
End
```

Các giới hạn khi xây dựng UDF

- Không thể gọi một stored procedure
- Không thể sử dụng các hàm loại không xác định được xây dựng sẵn trong SQL Server
  - Getdate, Rand, …
- Việc sử dụng RAISERROR và @@ERROR là hoàn toàn không hợp lệ.
- UDF không thể được sử dụng để sửa đổi cấu trúc các đối tượng trong CSDL như table, view, stored proc,…

Quản lý UDF

- Sử dụng các lệnh script tương tự như với thủ tục nội tại để quản lý các UDF
- Đổi tên

```
sp_rename[@objname=]'tên_đối_tượng',
                    [@newname=]'tên_mới'
                    [ , [ @objtype = ]
  'loại_đối_tượng']
```

- Thay đổi nội dung hàm
  - **ALTER FUNCTION**
- Xoá hàm
  - **DROP FUNCTION**

Gọi thực hiện các UDF loại đơn trị

- Sử dụng ở bất cứ nơi nào có thể thay bằng một giá trị đơn, trong mệnh đề
  - SELECT
  - WHERE, HAVING
  - GROUP BY
  - SET của câu UPDATE
  - VALUES của câu INSERT
  - Mô tả CHECK constraint
  - Mô tả giá trị DEFAULT
  - Mô tả công thức cột tính toán
  - Trong CASE

Ví dụ

- Giả sử đã xây dựng hàm ChuanChuoi để cắt khoảng trắng và viết hoa các từ

```
SELECT ID_KHG,dbo.ChuanChuoi(Ten_KHG) as HoTen FROM
  DM_KHACH_HANG

SELECT * FROM DM_KHACH_HANG

WHERE dbo.ChuanChuoi(Ten_KHG)='Tran Toan'

UPDATE DM_KHACH_HANG
SET Ten_KHG=dbo.ChuanChuoi(Ten_KHG)

Declare @HoTen varchar(50)
SET @HoTen='  tran   tHi   mai'
INSERT DM_KHACH_HANG
VALUES('KHG05',dbo.ChuanChuoi(@HoTen))
```

Ví dụ

- Hàm lấy đơn giá hiện hành của hàng hoá

```
CREATE FUNCTION DonGiaHienHanh(@MaHH
  varchar(50))
RETURNS numeric AS
BEGIN
   Return(Select  DonGiaHienHanh
      From DM_HANG_HOA Where ID_HH=@MaHH)
END
```

- Kiểm tra đơn giá phiếu xuất không lớn hơn 10% đơn giá hiện hành

```
ALTER TABLE CT_PHIEU_XUAT
ADD CONSTRAINT LayDonGia
CHECK
  (DonGia<=dbo.DonGiaHienHanh(MaHH)*110/100)
```

Sử dụng các UDF thuộc loại đọc bảng

- Coi hàm như một bảng ảo, sử dụng trong các câu SQL độc lập hay kết hợp với các bảng khác

```
CREATE FUNCTION dbo.LocDSKhachHang(@KyTuDau
  char(1))
RETURNS TABLE
AS
  RETURN SELECT *
                    FROM DM_KHACH_HANG
              WHERE LEFT(Ten_KHG, 1) =  @KyTuDau
```

- Việc sử dụng UDF đã xây dựng ở trên có thể gọi thực hiện như sau:

```
SELECT * FROM dbo.LocDSKhachHang('T')
```

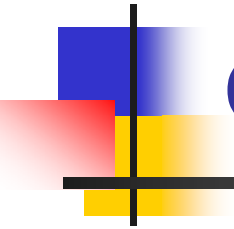
Sử dụng các UDF thuộc loại tạo bảng

- Sử dụng tương tự như hàm đọc bảng, tuy nhiên hàm tạo bảng giúp trả về những bộ giá trị phức tạp cần nhiều tính toán

```
CREATE FUNCTION F_DSHangHoa
(@LoaiHH varchar(50), @PhanTram numeric)
RETURNS @DSHangHoa Table(Ma_HH varchar(50),Ten_HH
  varchar(50),DonGiaKhuyenMai numeric)
Begin
  Insert Into @DSHangHoa(Ma_HH,Ten_HH,DonGiaKhuyenMai)
  Select ID_HH,Ten_HH,DonGiaHienHanh
  From DM_HANG_HOA where IDLoai_HH=@LoaiHH

  Update @DSHangHoa Set DonGiaKhuyenMai=DonGiaKhuyenMai-
                        (DonGiaKhuyenMai*@PhanTram)/100
  Return
End
```

- *Để sử dụng UDF đã xây dựng ở trên, bạn gõ vào các câu lệnh sau:*

```
SELECT * FROM dbo.F_DSHangHoa('TiVi',10)
```



Chương 12

TRIGGER

Trigger là gì?

- Cấu trúc gần giống như một thủ tục nội tại nhưng
  - Không có tham số đầu vào và đầu ra
  - Phải được liên kết với một bảng/ bảng ảo trong CSDL
- Không thể gọi mà được thực hiện tự động. Sử dụng trong việc:
  - Tính toán, cập nhật giá trị tự động
  - Kiểm tra dữ liệu nhập
- Khai báo sử dụng
  - Kết hợp với các hành động INSERT/UPDATE/DELETE trên bảng hay bảng ảo
  - Khi tạo ra, tham gia vào transaction khởi tạo bởi câu lệnh cập nhật dữ liệu tương ứng

Các xử lý bên trong trigger

- Kiểm tra các ràng buộc dữ liệu phức tạp
  - Các ràng buộc mô tả phức tạp, không thể dùng constraint
  - Gọi hành động Rollback Tran để hủy thao tác cập nhật khi vi phạm ràng buộc
  - Bảo đảm dữ liệu luôn được toàn vẹn
  - Bảo đảm việc kiểm thử ứng dụng không làm hư dữ liệu có sẵn
- Tính toán, tự động cập nhật giá trị
  - Bổ sung các hành động cập nhật dữ liệu để đảm bảo tính toàn vẹn dữ liệu
  - Đơn giản hoá việc xây dựng ứng dụng
- Chỉ định các bẫy lỗi dễ hiểu
  - Tăng tính thân thiện của ứng dụng
  - Dễ dàng nhận ra các lỗi khi lập trình

Các hạn chế trên trigger

- Không được tạo và tham chiếu bảng tạm
- Không tạo hay thay đổi, xoá cấu trúc các đối tượng sẵn có trong CSDL
  - CREATE/ALTER/DROP
- Không gán, cấp quyền cho người dùng
  - GRANT/REVOKE

Các loại trigger

- SQL Server có hai loại trigger
  - Trigger thông thường: AFTER (FOR) trigger
    - Chạy sau các hành động kiểm tra dữ liệu của các Rule, Constraint
    - Dữ liệu đã bị tạm thời thay đổi trong bảng
  - INSTEAD OF trigger
    - Chạy trước các hành động kiểm tra dữ liệu
    - Dữ liệu chưa hề bị thay đổi
    - Có thể thay thế hành động cập nhật dữ liệu bằng các hành động khác

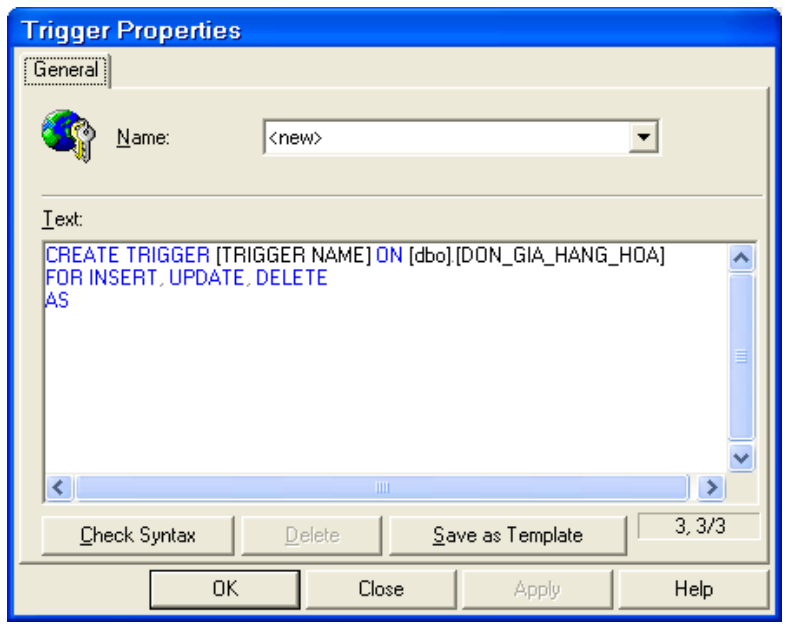
Các bảng trung gian Inserted và Deleted

- Inserted
  - Chứa dữ liệu được thêm mới trong hành động INSERT/UPDATE
  - Có ở cả hai loại trigger
  - Cấu trúc bảng giống với bảng thực sự được cập nhật dữ liệu
- Deleted
  - Chứa dữ liệu bị xoá trong hành động DELETE/UPDATE
  - Có ở cả hai loại trigger
  - Cấu trúc bảng giống với bảng thực sự được cập nhật dữ liệu
- Hành động update trong SQL Server
  - Xoá dòng dữ liệu cũ
  - Thêm vào dòng dữ liệu mới với thông tin đã cập nhật

Tạo mới trigger

- Tạo mới bằng Enterprise Manager
- Tạo mới bằng script

```
CREATE   TRIGGER Tên_Trigger ON Tên_bảng
{ [ INSTEAD OF ] | [ FOR | AFTER ] }
{ [ INSERT [, UPDATE [,DELETE ] ] ] }
AS
[DECLARE   Biến_cục_bộ]
    Các_lệnh
```



Mô tả

- Tên bảng
  - Tên bảng mà trigger tạo mới sẽ liên kết
- INSTEAD OF: chỉ định đây là trigger loại instead of trigger
  - Mỗi bảng chỉ có quyền tạo một instead of trigger cho một hành động cập nhật
- FOR hoặc AFTER
  - Nếu tạo trigger thông thường
- INSERT, UPDATE, DELETE
  - Hành động cập nhật dữ liệu tác động vào bảng để kích hoạt trigger.

Xóa trigger

- Xoá trigger bằng Enterprise Manager
- Xoá bằng script

```
DROP TRIGGER   Tên_trigger
```

  - Áp dụng cho database trên server khác khi cần
  - Quản lý được quá trình thay đổi của các đối tượng liên kết với bảng

Sửa nội dung trigger

- Sửa nội dung bằng Enterprise Manager
- Sửa nội dung bằng script

```
ALTER   TRIGGER Tên_Trigger ON Tên_bảng
FOR INSERT [, UPDATE [,DELETE ]]
AS
[DECLARE   Biến_cục_bộ]
    Các_lệnh
```

  - Áp dụng được cho các CSDL trên các server khác
  - Quản lý được quá trình thay đổi của các trigger gắn với bảng

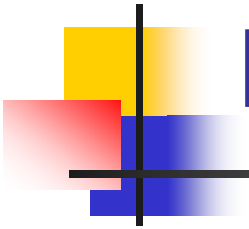
Trigger lồng nhau

- Trigger có thể lồng nhau
  - Hành động cập nhật → Trigger → Cập nhật bảng khác → Trigger trên bảng tương ứng
  - Instead Of trigger không phát sinh lại trên chính bảng mà nó liên kết
    - Cập nhật → Instead of Trigger → Gọi câu lệnh cập nhật xuống bảng → Instead of trigger
- Số cấp lồng tối đa
  - 32 cấp
  - Sử dụng biến `@@NestedLevel`
- Cấu hình cho phép trigger lồng nhau
  - `EXEC sp_configure 'nested triggers', [0 | 1]`

Khi thêm mới mẫu tin

- Thường dùng để kiểm tra
  - Khóa ngoại, Miền giá trị, Liên thuộc tính trong cùng một bảng
  - Liên thuộc tính của nhiều bảng khác nhau
- 3 loại đầu tiên, chỉ dùng trigger nếu muốn cung cấp các báo lỗi cụ thể bằng tiếng Việt
  - Nếu đã khai báo các ràng buộc này bằng constraint
- Các cấu trúc lệnh thường dùng khi kiểm tra
  - If Else
  - If Exists
  - Raiserror
  - Rollback Tran

Khi hủy bỏ mẫu tin



- Tương tự, kiểm tra các ràng buộc như trigger INSERT
- Nên kiểm tra ràng buộc khoá ngoại
  - Thông thường ràng buộc này dẫn đến việc phải cập nhật một số dữ liệu trên bảng khác
  - Chú ý: SQL Server có thuộc tính CASCADE DELETE

Khi sửa đổi mẫu tin

- Tương tự, kiểm tra các ràng buộc như trigger INSERT
  - Ràng buộc khoá ngoại có thể sử dụng CASCADE UPDATE để thực hiện tự động
- Xác định cột đang được cập nhật

```
If Update(Tên_cột)
  Xử lý
```

Trigger cập nhật giá trị tự động

- Sau khi kiểm tra ràng buộc trigger có thể
  - Rollback nếu dữ liệu không hợp lệ
  - Thực hiện tiếp các hành động cập nhật trên bảng khác để đảm bảo toàn vẹn dữ liệu: Cập nhật giá trị tự động
  - Vd: Insert → CTGiaoHang → Cập nhật bảng TONKHO
- Các hành động cập nhật thường thực hiện
  - Hủy bỏ dữ liệu do quan hệ khoá ngoại
  - Tính lại các cột 'tính toán' trong các bảng liên quan
- Vị trí thực hiện
  - Trong cùng trigger kiểm tra ràng buộc đã định nghĩa
  - Sau khi kiểm tra dữ liệu đã hợp lệ (thoả mãn các ràng buộc)

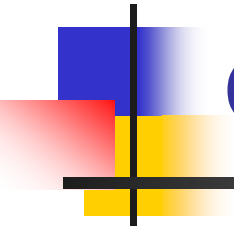
Instead of trigger

- Thông thường có thể được cập nhật View nhưng có nhiều giới hạn
  - Group By, Order By, Distinct
  - Ràng buộc khoá ngoại
  - Thiếu các cột NOT NULL trong bảng
- Trigger Instead of
  - Xảy ra trước khi SQL Server kiểm tra ràng buộc
  - Thay đổi hành động cập nhật vào bảng ảo bằng hành động thích hợp trên bảng gốc

Ví dụ

- Tạo bảng ảo sau

```
Select D.SoDH, NgayDH, MaNhaCC, V.MaVTu, TenVTu, SoLuong,
DonGia From CTDONDH CT, DONDH D, VATTU V Where CT.SoDH =
D.SoDH And CT.MaVTu = V.MaVTu


CREATE TRIGGER tg_vw_CTDONDH_BI
     INSTEAD OF INSERT ON vw_CTDONDH
AS
-- Nếu chưa có đơn đặt hàng, thêm đơn đặt hàng vào DONDH
Insert Into DONDH Select SoDH, NgayDH, MaNhaCC From
Inserted   Where SoDH Not In (Select SoDH From DonDH)
-- Nếu chưa có vật tư, thêm vật tư vào bảng VATTU
Insert Into VATTU(MaVTu, TenVTu) Select MaVTu, TenVTu From
Inserted Where MaVTu Not In (Select MaVTu From VATTU)
-- Thêm các chi tiết đặt hàng vào CTDONDH
Insert Into CTDONDH Select SoDH, MaVTu, SoLuong, DonGia
From Inserted
```



Chương 13

LOGIN & USER

Khái niệm về login và user

- Login: tên hệ thống (duy nhất) được SQL Server cấp phép truy cập hệ thống
- User: tên (duy nhất trong database) gắn với một login name cụ thể được SQL Server cấp phép truy xuất một database xác định
- Vậy,
  - Một loginname trong SQL Server có thể có nhiều user gắn kết
  - Một user được tạo trong database nào chỉ được phép truy xuất database đó
- Để tạo nhiều user truy xuất nhiều database gắn kết với một login name, sau khi tạo xong 1 login name ta phải mở các database tương ứng và lần lượt tạo từng user

Tạo login name

sp_addlogin ‘ketoanvien’, ‘123456’,’qlbhtbmt’

Declare @tc int Declare @tc int
Exec @tc = sp_addlogin ‘kt’, ‘abc’, ‘qlbhtbmt’
If @tc=1 ...

sp_grantlogin ‘LVHanh’

- Cú pháp
  - sp_addlogin
    [@loginame =] 'login’
    [[@passwd =] 'password']
    [, [@defdb =] 'database']
    [, [@deflanguage =] 'language']
    [, [@sid =] sid]
    [, [@encryptopt=] 'encryption_option
  - Trả về 0 (thành công) hoặc 1 (thất bại)
  - sp_grantlogin [@loginame=] 'login‘: Cấp phép cho user sẵn có của Windows truy xuất hệ thống SQL Server

Tạo login name

- CREATE LOGIN login_name
  { WITH <tùy chọn 1> | FROM <sources> }
  - <tùy chọn 1> : PASSWORD ='password' [HASHED] [MUST_CHANGE]
    [, <tùy chọn 2> [,...]]
    - <tùy chọn 2> : DEFAULT_DATABASE = database | DEFAULT_LANGUAGE = language | CHECK_EXPIRATION = { ON | OFF}
  - <sources> :=WINDOWS [WITH [DEFAULT_DATABASE = database |

DEFAULT_LANGUAGE = language]]

- Ví dụ:
  - CREATE LOGIN ketoan WITH PASSWORD='abc', DEFAULT_DATABASE=qlbhtbmt
  - CREATE LOGIN LVHanh FROM WINDOWS WITH DEFAULT_DATABASE=qlbhtbmt

Tạo user

- Cú pháp
  - sp_adduser [@loginame =] 'login' [, [@name_in_db =] 'user'] [, [@grpname =] 'role']
- Lưu ý:
  - Trước khi thực hiện tạo user phải mở kết nối đến CSDL dự định cho phép truy xuất
  - User sau khi được tạo ra được phép truy xuất CSDL đang mở và có quyền hạn tuân theo role đã chỉ định. Trường hợp không chỉ định role, user không có bất kỳ quyền hạn nào trên các đối tượng: table, view, ...
- Ví dụ:

use QLBHTBMT

GO

sp_adduser 'ketoan', 'kt'

Tạo user

- CREATE USER username [FOR LOGIN loginname]
- Ví dụ:
  - CREATE USER KT FOR LOGIN ketoan
  - CREATE USER Ktoan
    - Lỗi do chưa có login tên Ktoan trong hệ thống
- Xóa user: DROP USER username
- Xóa login: DROP LOGIN loginname

Các nhóm user định sẵn trong CSDL

- Database Role

sp_droprolemember

Role	Quyền hạn
db_accessadmin	Cấp / hủy quyền truy xuất của các login
db_backupoperator	Tạo sao lưu cho CSDL
db_datareader	Đọc dữ liệu từ tất cả table, view
db_datawriter	Cập nhật dữ liệu trên tất cả table
db_ddladmin	Thực hiện các lệnh DDL trên CSDL
db_denydatareader	Cấm user khác đọc dữ liệu
db_denydatawriter	Cấm user khác cập nhật dữ liệu
db_owner	Toàn quyền
db_securityadmin	Thêm, xóa các thành viên trong các role

- sp_addrolemember [@rolename =] 'role',
  [@membername =] 'security_account'
- sp_addrolemember 'db_datareader', 'KT'

Cấp quyền cho user

- Cú pháp

```
GRANT <ALL | PRIVILEGES>
[ON  OBJECT [(column [ ,...n ])] ]
TO <user> [ ,...n ]
[WITH GRANT OPTION ]
```

- PRIVILEGES

Scalar function permissions	EXECUTE, REFERENCES.
Table-valued function permissions	DELETE, INSERT, REFERENCES, SELECT, UPDATE.
Stored procedure permissions	EXECUTE, SYNONYM, DELETE, INSERT, SELECT, UPDATE.
Table permissions	DELETE, INSERT, REFERENCES, SELECT, UPDATE.
View permissions	DELETE, INSERT, REFERENCES, SELECT, UPDATE.

Thu hồi / cấm quyền của user

- REVOKE [GRANT OPTION FOR]
  <permission> [,...n]
  ON [OBJECT [(column [,...n])]
  FROM <user> [,...n]

- DENY <ALL | PRIVILEGES>
  [ON OBJECT [(column [,...n])]]
  TO <user> [,...n]

Ví dụ minh họa

- Tạo login và user abc với mật khẩu abc cho phép truy xuất CSDL QLDA với các quyền hạn như sau:
  - Cho phép đọc dữ liệu NhanVien trên các thuộc tính: MANV, HONV, TENLOT, TENNV, NGSINH, PHAI, DCHI, MAPB
  - Cho phép đọc dữ liệu trên PhongBan và DiaDiemPBan
  - Cho phép đọc, thêm dữ liệu trên PhanCong
  - Chỉ được sửa dữ liệu trên cột ThoiGian của PhanCong
  - Cho phép tạo VIEW
  - Được thi hành các procedure: spThongKeLuongTheoPB, spTimDeAnTheoTen

Ví dụ minh họa

```
CREATE LOGIN  abc WITH PASSWORD='abc',
  DEFAULT_DATABASE=QLDA
GO
use QLDA
GO
CREATE USER abc FOR LOGIN abc
GO
```

Ví dụ minh họa

GRANT SELECT ON NHAN VIEN (MANV, HONV, TENLOT, TENNV, NGSINH, PHAI,DCHI,MAPB) TO ABC

GRANT SELECT ON PHONGBAN TO ABC

GRANT SELECT ON DIADIEMPBAN TO ABC

GRANT SELECT, INSERT ON PHANCONG TO ABC

GRANT UPDATE ON PHANCONG(ThoiGian) TO ABC

GRANT CREATE VIEW TO ABC

GRANT EXECUTE ON spThongKeLuongTheoPB TO ABC

GRANT EXECUTE ON spTimDeAnTheoTen TO ABC

GO

Ví dụ minh họa

```
CREATE PROC spThongKeLuongTheoPB
AS
BEGIN
  SELECT pb.MaPB, TenPB,
  'TongLuong'=SUM(Luong)
  FROM PhongBan pb, NhanVien nv
  WHERE pb.MaPB=nv.MaPB
  GROUP BY pb.MaPB, TenPB
END
```

Ví dụ minh họa

```
CREATE PROC spTimDeAnTheoTen
   @tenda nvarchar(50)
AS
BEGIN
   DECLARE @dk nvarchar(52)
   SET@dk = '%' + @tenda + '%'

   SELECT *
   FROM DeAn
   WHERE TenDA LIKE @dk
END
```

Ví dụ minh họa

- Kiểm tra bằng cách:

1. Chọn FILE / CONNECT
2. Trong CONNECT USING, chọn SQL SERVER AUTHENTICATION và gõ:
   - LOGIN NAME: abc
   - PASSWORD: abc
3. Thử SELECT, IN SERT, UPDATE, DELETE, EXECUTE, ... trên các table, procedure chưa cho phép và đã cho phép để xem kết quả.

Bài tập (QLBH TBMT)

- Viết procedure nhận 3 tham số @sohd, @mahh, @slg
  - Kiểm tra sự hợp lệ của các giá trị trong 3 tham số
  - Nếu hàng hóa có mã là @mahh là mặt hàng đã được đặt trong đơn đặt hàng của hóa đơn giao hàng này thì :
    - Thực hiện thêm bộ mới với giá trị trong 3 tham số này vào table ChiTietHD
    - Cập nhật giá trị cho cột TongTien trong table HoaDon
- Tạo procedure nhận tham số là @tenkh, thực hiện tìm tất cả các khách hàng có tên chứa các ký tự trong @tenkh
- Tạo procedure nhận tham số là @ngayhd. Cho biết danh sách các đơn đặt hàng được lập trong ngày này cùng tổng số lượng hàng hóa, tổng tiền của từng đơn đặt hàng này

Bài tập (QLBH TBMT)

- Tạo login và user nvbh với password nvbh có khả năng truy xuất csdl QLBHTBMT
- Cấp quyền hạn cụ thể cho user nvbh như sau:

TABLE/CHỨC NĂNG	QUYỀN
NHANVIEN	SELECT(MANV, HOTEN)
KHACHHANG	SELECT
DDH, HOADON	SELECT, INSERT
CTDATHANG, CHITIETHD	SELECT, INSERT, UPDATE, DELETE
HANGHOA	SELECT, INSERT
	TẠO VIEW
Các procedure	EXECUTE