# 1 Tổng quan về Cơ sở dữ liệu

**Nguyễn Trong Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com

# Mục tiêu

Trong phần này các bạn sẽ làm quen với

- Khái niệm về cơ sở dữ liệu (CSDL)
- Một số CSDL đặc trưng
- Một số thuật ngữ thông dụng đối với CSDL

# Khái niệm về cơ sở dữ liệu

# Khái niệm về CSDL

- Một CSDL là một kho chứa dành cho các tập hợp dữ liệu hoặc sự kiện có liên quan trong một cấu trúc đặc trưng
- Ví dụ về một CSDL không được vi tính hóa là một danh bạ điện thoại

# Một số cơ sở dữ liệu đặc trưng

# Một số CSDL đặc trưng

- Các CSDL phân cấp (Hierarchical Database)
- Các CSDL mạng (Network Database)
- Các CSDL hướng đối tượng (Object-Oriented Database)
- Các CSDL quan hệ (Relational Database)

# CSDL phân cấp

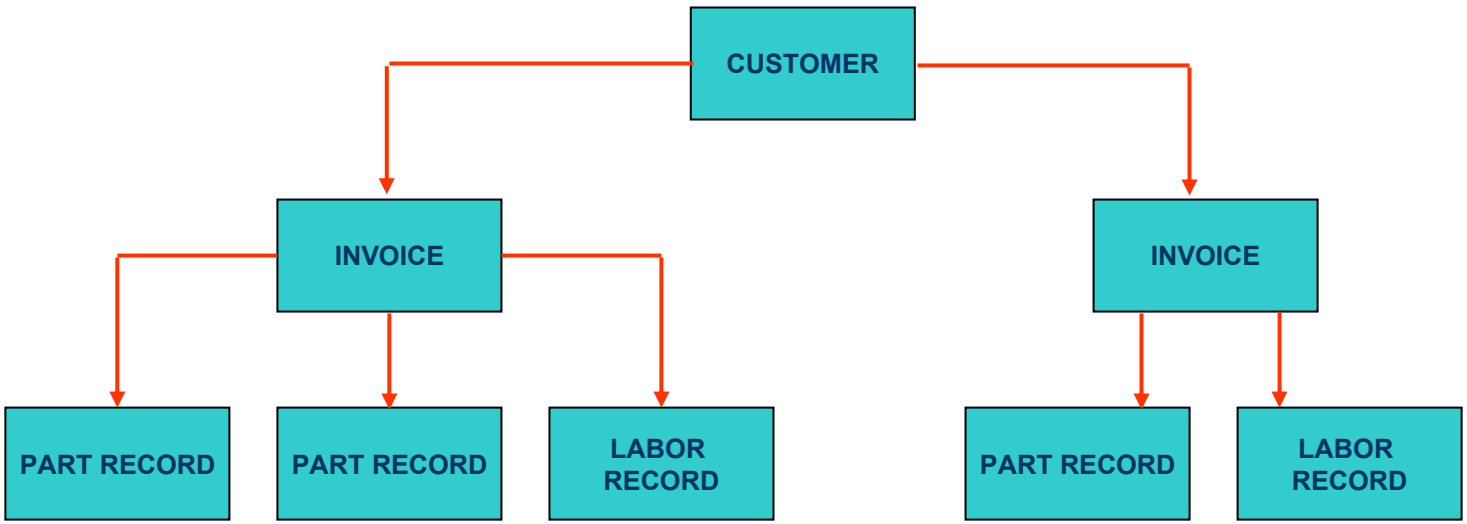
- Các CSDL phân cấp : các bảng được tổ chức thành một cấu trúc dạng cây cố định, mỗi bảng lưu trữ một kiểu dữ liệu. Bảng thân cây (bảng chính) lưu giữ các thông tin tổng quát
- Các bảng của CSDL phân cấp có mối quan hệ một-nhiều

# CSDL phân cấp

- Mối quan hệ cố định nên không đòi hỏi các dữ liệu trùng lặp và có thể nhanh chóng định vị dữ liệu
- Mối quan hệ cố định hạn chế tính linh hoạt của CSDL, một số dạng truy vấn hay báo cáo trở nên khó khăn hoặc không thực hiện được

# CSDL phân cấp

Ví dụ về CSDL phân cấp : các bảng được liên kết trong một CSDL phân cấp với quan hệ một-nhiều

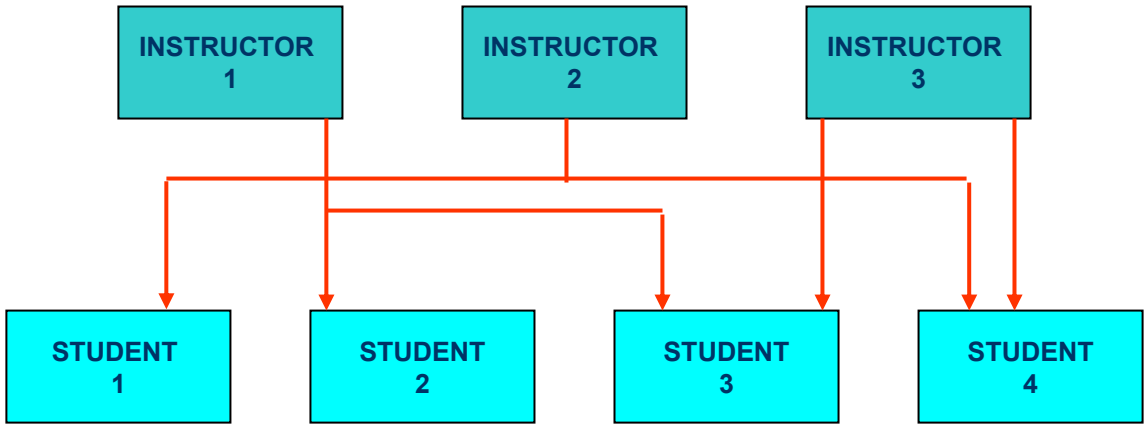


# CSDL mạng

- Các CSDL mạng : tương tự với cấu trúc phân cấp, chỉ khác là một bảng bất kỳ có thể liên hệ tới một số lượng bất kỳ các bảng khác
- Các bảng của CSDL mạng có mối quan hệ nhiều-nhiều
- Thường được dùng cho các ứng dụng với yêu cầu về CSDL là cố định

# CSDL mạng

Ví dụ CSDL mạng : các bảng được liên kết trong một CSDL mạng với quan hệ nhiều-nhiều



# CSDL hướng đối tượng

- Các CSDL hướng đối tượng : nhóm các hạng mục dữ liệu thành các hạng mục phức tạp gọi là các đối tượng
- Một đối tượng được định nghĩa bởi các đặc điểm, các thuộc tính và các thủ tục của nó

# CSDL hướng đối tượng

- Các đặc điểm của một đối tượng có thể là văn bản, âm thanh, video hoặc đồ họa ..
- Các thuộc tính của một đối tượng có thể là màu sắc, kích thước, kiểu dạng ..
- Các thủ tục gắn liền với việc xử lý hoặc giải quyết một công việc của đối tượng

# CSDL quan hệ

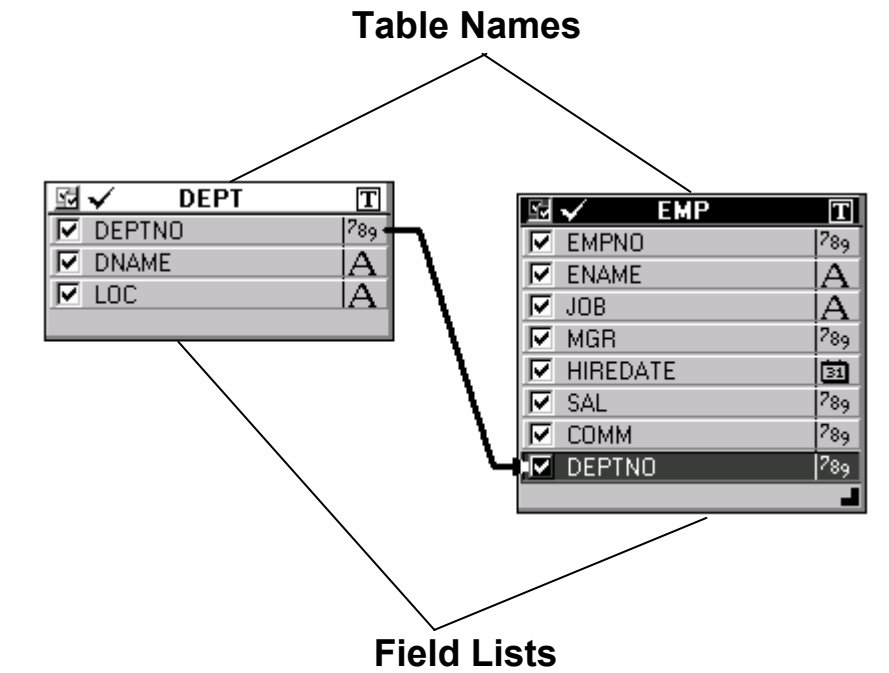
- Các CSDL quan hệ bao gồm một tập các bảng, trong đó một hoặc nhiều trường chung tồn tại trong hai bảng bất kỳ sẽ tạo ra mối quan hệ giữa hai bảng đó
- Cấu trúc CSDL quan hệ đang là CSDL thông dụng nhất hiện nay

# CSDL quan hệ

- Nhiều bảng trong kiểu CSDL quan hệ và các mối quan hệ của nó cho phép giải quyết nhiều tác vụ quản lý dữ liệu khác nhau

# CSDL quan hệ

❖ Ví dụ về CSDL quan hệ : trường chung deptno tồn tại trong hai bảng tạo ra mối quan hệ giữa bảng DEPT và bảng EMP



# Tổng quan về hệ quản trị cơ sở dữ liệu

# Hệ quản trị cơ sở dữ liệu

## *Database management system (DBMS)*

- Là một chương trình hay một bộ các chương trình, cho phép một số lượng người sử dụng bất kỳ truy xuất và chỉnh sửa các dữ liệu trong một CSDL.
- Cung cấp những công cụ cho phép người sử dụng đặt ra các yêu cầu đặc biệt (được gọi là các truy vấn) để dễ dàng nhận được các bản ghi được chọn ra từ CSDL

# Chúng ta sẽ làm quen với

1. **Thao tác đối với cơ sở dữ liệu**
2. **Lịch sử phát triển của SQL server**
3. **Giới thiệu các phiên bản, các thành phần, các công cụ của SQL**
4. **Một số thuật ngữ thông dụng đối với CSDL**

# Thao tác đối với cơ sở dữ liệu

# Hệ quản trị cơ sở dữ liệu

- Một Hệ quản trị CSDL có thể thực hiện các chức năng quản lý dữ liệu như sau :
  - Tạo các bảng
  - Nhập và biên tập dữ liệu
  - Xem các dữ liệu
  - Sắp xếp các bản ghi
  - Truy vấn CSDL
  - Sinh các báo cáo

# Tạo các bảng trong CSDL

- Là bước đầu tiên trong việc xây dựng bất kỳ một CSDL nào, các bảng này sẽ chứa các dữ liệu thô mà DBMS sẽ làm việc trên đó
- Cần định nghĩa kiểu dữ liệu nào sẽ được lưu trữ trong mỗi bảng hay định nghĩa các trường của bảng bằng một tiến trình gồm ba bước :
  - Đặt tên cho trường
  - Chỉ định kiểu trường
  - Chỉ định kích thước trường

# Tạo các bảng trong CSDL

- Thành một giá trị số, giống hệt như các ngày tháng và các thời gian được lưu trữ bên trong như là các con số tuần tự trong các ô của bảng tính.
- Các trường nhị phân lưu trữ các đốI tượng nhị phân, hay các BLOB (Binary large object), có thể là một tập tin hình ảnh đồ họa, có thể là một tập tin âm thanh, đoạn video ngắn hoặc văn bản được định dạng.

# Nhập và biên tập dữ liệu

- Sau khi bảng đã được xây dựng, các dữ liệu có thể được nhập vào.
- Việc nhập dữ liệu chỉ là việc gõ các ký tự trên bàn phím.
- Phần lớn các DBMS đều cho phép bạn tạo ra một biểu mẫu nhập liệu để làm cho công việc nhập dữ liệu trở nên dễ dàng hơn.

# Nhập và biên tập dữ liệu

- Để tạo điều kiện dễ dàng hơn cho người nhập liệu người ta đưa ra một số các kỹ thuật và công cụ có thể sử dụng để kiểm soát việc nhập liệu.
- Các công cụ này đều chấp nhận các ký tự hợp lệ và kiểm soát định dạng hiển thị của khoản mục.

# Xem các bản ghi

- Cách các dữ liệu xuất hiện trên màn hình ảnh hưởng phần nào tới năng suất làm việc của những người sử dụng làm việc với chúng
- Một DBMS cho phép tạo ra các biểu mẫu để xem các bản ghi, các biểu mẫu này được thiết kế giống với các biểu mẫu nhập liệu nhưng chúng được sử dụng để hiển thị các dữ liệu sẵn có

# Xem các bản ghi

- Có những biểu mẫu xem dữ liệu đơn giản như mỗi lần chỉ xem một bản ghi hoặc những biểu mẫu phức tạp hiển thị các thông tin có liên quan được rút ra từ nhiều bảng

# Sắp xếp các bản ghi

- Một trong những tính năng mạnh mẽ nhất của một DBMS là khả năng sắp xếp một bảng dữ liệu
- Tác vụ sắp xếp tùy thuộc vào nội dung của một hay nhiều trường
- Khi sắp xếp các bản ghi cần cân nhắc thứ tự sắp xếp, có thể theo chiều tăng dần hoặc giảm dần

# Truy vấn CSDL

- Một phát biểu do người sử dụng xây dựng nhằm mô tả dữ liệu và đặt ra các tiêu chí để DBMS có thể thu thập các dữ liệu thích hợp và xây dựng các thông tin cụ thể hay một dạng bộ lọc mạnh mẽ hơn, có khả năng thu thập thông tin từ nhiều bảng trong một CSDL quan hệ được gọi là một truy vấn

# Truy vấn CSDL

- Tương tự với việc nhập các điều kiện sắp xếp, bạn có thể nhập các biểu thức và các tiêu chí mà
  - Cho phép DBMS định vị các bản ghi
  - Thiết lập các mối quan hệ hay các liên kết giữa các bảng để cập nhật dữ liệu
  - Liệt kê một tập con các bản ghi
  - Thực hiện các phép toán
  - Xóa các bản ghi lỗi thời
  - Thực hiện các tác vụ quản lý dữ liệu khác

# Sinh các báo cáo

- Không phải tất cả các tác vụ DBMS đều xảy ra trên màn hình
- Một báo cáo được in các thông tin giống như kết quả truy vấn được thu thập bằng cách tập hợp các dữ liệu dựa trên các tiêu chí do người sử dụng cung cấp
- Các bộ sinh báo cáo trong phần lớn các DBMS đều tạo ra các báo cáo từ các truy vấn

# Lịch sử phát triển

- IBM tạo ra ngôn ngữ truy vấn: Thập kỷ 70
- Microsoft Và Sysbase phát triển một sản phẩm cơ sở dữ liệu dựa trên nền của OS/2
- Version 4.2 và version 4.21
- Version 6.0
- Version 6.5
- Version 7.0
- Version Sql server 2000
- Version Sql server 2005 (Bản thử nghiệm)

# Lịch sử phát triển (2)

- Ghi chú:
  - SQL server 2000 chạy được trên Win 9x. Nhưng không chạy được đầy đủ các tính năng của SQL server.
  - Khi cài đặt trên Win 9x. Càng nhiều người dùng hiệu xuất càng giảm.
  - Môi trường tốt nhất cho SQL server 2000 là Windows 2000 advance Server

# Những thông tin liên quan đến SQL server

- Có thể cài đạt môi trường làm viêc trên các loại máy tính. Từ Desktop đến mainframe (máy tính lớn)
- Có nhiều lựa chọn cho những đối tượng sử dụng SQL server.
- Các lựa chọn chính là sự phân loại khách hàng của Microsoft.

# Những thông tin liên quan đến SQL server

1. Ấn bẩn đánh giá - Evaluation Edition (enterprise edition in 120 → 180 days)
2. Ấn bản giành cho cá nhân - Personal Edition (support 1 CPU – under 10 users)
3. Ấn bản chuẩn - Standard Edition (support 4 CPU – RAM 2GB)
4. Ấn bản giành cho những người phát triển ứng dụng Developer Edition
5. Ấn bản giành cho tổ chức kinh doanh, xí nghiệp Enterprise Edition (support 32 CPU – RAM 64GB)
6. MSDE: Phiên bản không đầy đủ của sản phẩm, hạn chế sao chép; không có giao diện người dùng (giống như kho lưu trữ dữ liệu đầu cuối)

# 1 Giới thiệu các thành phần, kiểu dữ liệu và công cụ chính

**Nguyễn Trong Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com

# Một số thành phần

1. Tables
2. Column
3. Stored Procedure
4. User-definitions
5. Triggers
6. Views

# Một số thành phần (2)
# Tables

- Tables: (các bảng) là các đối tượng chứa các kiểu dữ liệu và dữ liệu thô thực sự.
  - Có thể có nhiều kiểu dữ liệu giống nhau trong một bảng.

| | |
|---|---|
| Lop | Varchar(50) |
| Soluong | Int |
| Do | Bit |
| ngayCN | DateTime |
| Ghichu | Varchar(20) |

# Một số thành phần (3)
# Columns

- Columns (các cột) là các phần của bảng đang chứa dữ liệu. Các cột phải được gán một kiểu dữ liệu và tên là duy nhất.
  - Lop: là trường kiểu ký tự có độ dài 50
  - Soluong: là trường là kiểu số nguyên.
  - Do: là trường kiểu logic chỉ có hai giá trị 0 hoặc 1
  - ngayCN: là trường kiểu dữ liệu ngày tháng.
  - Ghichu:

# Một số thành phần (4)

## Stored Procedure

- Stored Procedure: (thủ tục thường trú - thủ tục lưu trữ) giống như các macro, trong đó các đoạn mã lệnh transact SQL có thể viết và lưu trữ mang một tên.
- Ghi chú: chúng ta có thể hiểu Store procedure giống như hàm hay thủ tục trong các ngôn ngữ lập trình.
  - CREATE PROCEDURE [dbo].[Class_Delete]
  - @ClassID int
  - AS
  - Delete Class Where ClassID = @ClassID
  - GO

# Một số thành phần (5)
## User-defined functions

- User-defined functions (các hàm do người dùng định nghĩa) rất giống với các mã lệnh của stored procedure.

```
CREATE FUNCTION dbo.Date2String8
        (
        @dt DateTime
        )
RETURNS CHAR(10)
AS
BEGIN
        RETURN CONVERT ( char(10) , @dt, 103 )
END
```

# Một số thành phần (6)
# Triggers

- Triggers: (các bẩy lỗi) là các thủ tục lưu trữ kích hoạt trước hoặc sau khi bổ sung, sửa chữa hoặc xoá dữ liệu ra khỏi CSDL.
- Trigger: là một dạng đặc biệt của stored procedure dùng phản hồi một sự kiện cụ thể.
- Trigger gồm một đoạn mã được gắn vào bảng dữ liệu.
- Trigger tự động thực hiện khi có sự kiện sảy ra tương ứng với trigger được gán cho sự kiện đó (stored procedure: được gọi khi cần)

# Một số thành phần (7) (Triggers - 2)

- Trigger không sử dụng hai đặc tính của stored procedure là tham số và giá trị trả về.
- Vì là tự động chạy nên chúng ta phải cẩn thận khi dùng. (có thể gây ra phiền toái trong lúc thực thi)

# Một số thành phần (8)
## Views

- Views: (các khung nhìn - bảng ảo) về cơ bản là các truy vấn lưu trữ trong CSDL để có thể tham chiếu tới một hoặc nhiều bảng.
- Không thể hiện cột nào đó trong bảng hoặc liên kết hai hoặc nhiều bảng khác nhau.
- Chúng ta có thể sử dụng Views làm kỹ thuật an toàn dữ liệu.

# Kiểu dữ liệu (Data Type)

- Data Type: (kiểu dữ liệu) là các kiểu lưu trữ cơ bản về dữ liệu. Mỗi kiểu dữ liệu chỉ được gán cho một cột trong bảng.
- Ví dụ:
  - Int: kiểu số nguyên
  - Bit: kiểu logic
  - Varchar: kiểu xâu ký tự

# Một số kiểu dữ liệu

- Các trường văn bản : (trường ký tự hay trường số và chữ cái): chấp nhận bất kỳ chuỗi ký tự số và chữ cái nào không được sử dụng trong các phép toán. Các trường văn bản cũng thường lưu trữ các khoản mục bao gồm các con số

# Một số kiểu dữ liệu

Các trường số : lưu trữ các dữ liệu số thuần túy. Các con số trong một trường số có thể đại diện cho tiền tệ, các phần trăm, các số liệu thống kê, các đạl lượng, hoặc bất kỳ giá trị nào khác vốn có thể được sử dụng (nhưng không nhất thiết) trong các phép toán.

- Các trường ngày hoặc trường thời gian : lưu trữ các khoản mục ngày tháng hoặc thời gian. Kiểu trường này chuyển đổi một khoản mục ngày tháng hoặc thời gian

# Một số công cụ

- Trợ giúp trực tuyến
- Tiện ích mạng client và Server
- Giới thiệu trình điều khiển Enterprise Manager
- Trình soạn thảo Query Analyzer
- Dịch vụ quản lý Server

# Một số công cụ (trợ giúp trực tuyến)

- Trợ giúp trực tuyến:
  - Trong một số chừng mực nào đó, người lập trình không nhớ hết các cú pháp hay các phép toán khác cũng như các thủ tục của SQL.
  - Books online là công cụ trợ giúp trực tuyến tốt nhất

# Một số công cụ (tiện ích mạng)

- Tiện ích mang: để các máy PC khác có thể kết nối với nhau và sử dụng cơ sở dữ liệu SQL server trong hệ thống. Chúng ta cần cấu hình các tiện ích máy trạm giống như cấu hình của Server.
  - Việc cấu hình để nhận biết được các máy trạm.
  - Các máy trạm có thể không dùng một giao thức

# Một số công cụ (Enterprise manager)

- Tiện ích Enterprise manager: Cung cấp cho người quản trị nhiều chức năng để quản lý SQL server bằng giao diện đồ hoạ.
  - Hiện nay Enterprise Manager đã cung cấp hầu hết các tính năng giao diện đồ hoạ để làm việc với SQL server.
  - Ngoài ra còn có một vài chức năng như: đăng ký nhiều SQL server khác, SQL server Profiler, …

# Một số công cụ (Query Analyzer)

- Giúp chúng ta gỡ rối hay phát triển trong SQL server.
- Là trình soạn thảo và thực thi câu lệnh SQL hay Stored Procedure
- Để sử dụng tiện ích này, chúng ta phải kết nối vào SQL server. Khi kích hoạt chúng, ta phải cung cấp tên Server đồng thời cung cấp Username và password

# Các khái niệm

1. Index
2. Primary key
3. Foreign key
4. Constraints

# Indexes

- Indexes (các chỉ mục) giúp tổ chức lại dữ liệu, nên các truy vấn truy vấn chạy nhanh hơn.

# Primary keys

- Primary keys (các khoá chính) bản chất không phải là một đối tượng nhưng là những yếu tố chủ yếu cho các cơ sở dữ liệu quan hệ.
- Các khoá chính làm cho tính duy nhất của dữ liệu muốn lưu trữ có hiệu lực.

# Foreign keys

- Foreign keys (các khoá ngoại) là một hoặc nhiều cột tham chiếu đến các khoá chính
- Là các ràng buộc duy nhất của các bảng khác.
- SQL Sử dụng khoá chính và các khoá ngoại để liên kết các dữ liệu trong các bảng riêng biệt với nhau khi thực hiện truy vấn.

# Constraints

- Constraints (các ràng buộc) là các cơ chế được hệ thống cài đặt dựa trên máy chủ nhằm bảo vệ tính toàn vẹn của dữ liệu.
- Rules: ấn định cho các cột, dữ liệu nhập phải phù hợp với các chuẩn mực mà bạn đã thiết lập.
- Defaults: mặc định có thể thiết lập cho các field (theo cột) mà khi không có dữ liệu nào được nhập vào các field này trong thao tác Insert.

# 2

# Các phát biểu của Transact-SQL (T-SQL)

**Nguyễn Trong Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com

# Lưu ý:

Khi cài đăt xong SQL server 2000

Chúng ta sẽ sử dụng Cơ sở dữ liệu mặc định có sẵn khi chúng ta cài đặt xong SQL server 2000

Northwind

Pubs

# Mục tiêu

Trong phần này các bạn sẽ làm quen với

- Select (chọn lọc bản ghi)
- Insert ( thêm bản ghi)
- Update (Cập nhật)
- Delete (Xoá bản ghi)
- Các hàm và phát biểu hỗ trợ

# Câu lệnh Select

Select <chọn lọc bản ghi>

From <danh sách bảng>

[Where <các điều kiện ràng buộc>]

[Group by <tên cột hay biểu thức sử dụng cột trong select>]

[Having <điều kiện bắt buộc dựa trên Group By>]

[Order By <danh sách cột>]

# Câu lệnh Select với From

Dùng để đọc thông tin nào đó từ cơ sở dữ liệu theo:

- Những trường quy định
- Những biểu thức cho trường đó
- Dấu * cho phép lọc mẩu tin với tất cả các trường trong các bảng.

# Cú pháp đơn giản

Select *

From tenbang

(select * from Categories)

/* Lọc tất cả số liệu của các cột (field) của bảng Categories */

Select cot1,cot2

From tenbang

(select CategoryName, Description from Categories)

/* Lọc tất cả số liệu của cột CategoryName, Description trong bảng Categories */

# Cú pháp đơn giản

```
Select top 2 *
From tenbang
(Select top 2 * from Categories)
/* Lọc 2 bản ghi đầu tiên của các cột (field) trong bảng Categories */
```

```
Select top 2 cot1,cot2
From tenbang
(Select top 2 CategoryName, Description from Categories)
/* Lọc 2 bản ghi đầu tiên của CategoryName, Description cột trong bảng Categories */
```

# Chú ý

Chúng ta có thể cộng hai hay nhiều trường để tạo ra cột mới”

select *, productName + ': ' + QuantityPerUnit as 'Ten va gia'  from products

# Câu lệnh Select có dùng mệnh đề Where

Select *

From tenbang

Where <cac dieukien>

Các điều kiện bao gồm:

- các phép toán so sánh
- Các phép toán logic

Chú ý: *chúng ta có thể sử dụng các phép toán lồng nhau*.

# Câu lệnh Select có dùng mệnh đề Where (các phép toán so sánh)

select * from Products

- > : lớn hơn — where unitPrice > 8
- < : nhỏ hơn — where unitPrice < 8
- >= : lớn hơn hoặc bằng — where unitPrice >= 8
- <= : nhỏ hơn hoặc bằng — where unitPrice <= 8
- = : bằng — where unitPrice = 8
- != : khác — where unitPrice != 8
- <> : khác — where unitPrice <> 8
- !> : Không lớn hơn — where unitPrice !> 8
- !< : không nhỏ hơn — where unitPrice !< 8

# Câu lệnh Select có dùng mệnh đề Where (các phép toán so sánh)

And: Phép toán and

Or: phép toán “or”

Not: phép toán phủ định

Between: nằm trong miền

Like: Phép toán so sánh gần giống, sử dụng dấu % để thực hiện thay thế bằng ký tự đại diện.

In: Phép so sánh trong một tập hợp

Exists: trả về true nếu ít nhất một mẩu tin tồn tại

# Ví dụ câu lệnh Select có dùng mệnh đề Where (các phép toán so sánh)

```
select * from Products where unitPrice < 8

select * from Products where unitPrice > 8

select * from Products where unitPrice <= 8

select * from Products where unitPrice >= 8

select * from Products where unitPrice = 8
```

# Ví dụ câu lệnh Select có dùng mệnh đề Where (các phép toán so sánh)

```
select * from Products where unitPrice =18.00

select * from Products where unitPrice !=18.00

select * from Products where unitPrice <> 18.00

select * from Products where unitPrice !>18.00

select * from Products where unitPrice !<18.00
```

# Ví dụ câu lệnh Select có dùng mệnh đề Where (các phép toán logic)

```
select * from Products where unitPrice < 8 and
CategoryID = 1

select * from Products where unitPrice < 8 Or
CategoryID = 1

select * from Products where CategoryID is Null

select * from Products where CategoryID is  Not
Null
```

# Ví dụ câu lệnh Select có dùng mệnh đề Where (các phép toán logic)

select * from Products where UnitPrice between 8 and 10

select * from Products where ProductName like 'C%‘

select * from Products where ProductName like '%C‘

select * from Products where ProductName like ‘%C%'

# Ví dụ câu lệnh Select có dùng mệnh đề Where (các phép toán logic)

select * from Products where UnitPrice in (6,7,8,9)

select * from Products where UnitPrice not in (6,7,8,9)

select * from Products where UnitPrice is  Null

select * from Products where UnitPrice is  not Null

# Các phép toán tương đương trong Biểu thức so sánh và logíc

select * from Products where UnitPrice in (6,7,8,9

Hai biểu thức tương đương ➔ Hai câu lệnh tương

select * from Products where (UnitPrice=6) or (UnitPrice=7) or (UnitPrice=8) or (UnitPrice=9)

# Cú pháp Select Với Order By

Thông thường Order by được sắp xếp theo trật tự tăng dần.

Cú pháp:

Order by tencot  (tăng dần)

Order by tencot  Desc (giảm dần)

Order by tencot1 + tencot2  Desc (giảm dần)

Order by tencot1 Desc, tencot2  ASC

# Câu lệnh Select Với Order By

select * from products order by supplierID desc

Kết hợp hai trường với nhau

select productID + supplierID,* from products
order by productID + supplierID desc

Kết hợp với Where

select productID + supplierID,* from products
Where productID > 70

order by productID + supplierID desc

# Câu lệnh Select Với Order By

Tương tác với 2 cột

select * from products order by categoryID Desc, supplierID ASC

(Sắp sếp cột CategoryID giảm dần, sau đó sắp supplierID tăng dần)

# Cú pháp Select Với Group By

Khi truy vấn một hay nhiều bảng, thông thường có những nghiệp vụ thuộc trường nào đó có cùng giá trị.

- Chúng ta muốn đếm số lần xuất hiện của nhóm
- Tính tổng một cột hay nhiều cột của nhóm
- Cú pháp:

Select <chọn lọc bản ghi>

From <danh sách bảng>

[Group by <tên cột hay biểu thức sử dụng cột

trong select>]

# Câu lệnh Select Với Group By

```
USE pubs
select * from titles

SELECT royalty, SUM(advance) as 'total
advance'
   FROM titles
   GROUP BY royalty
```

# Câu lệnh Select Với Having (dựa trên Group By)

Having thường đi với Group By

Having giống biểu thức Where đặc biệt là khi ta không sử dụng group By

Câu lệnh:

SELECT pub_id, AVG(price) FROM titles GROUP BY pub_id HAVING (AVG(price) > 10)

# Câu lệnh select với Distinct

Khi ta có một hay nhiều bảng kết nối với nhau, sẽ xảy ra nhiều trùng lặp nhiều mẩu tin. Nếu chỉ cần lấy một trong những mẩu tin trùng lặp đó ta sử dụng select với Distinct.

Ví dụ:

```
Select customerID, employeeID
from orders

Select distinct customerID, employeeID
from orders
```

# Câu lệnh select với AS

Khi cần thay đổi tên trường trong câu truy vấn ta dùng AS, cho phép ánh xạ tên cũ hay giá trị chưa có thành tên mới (header)

Ví dụ:

```
Select customerID, count(customerID) as 'SO LAN'
from orders
group by customerID
```

# Câu Lệnh INSERT

# Câu lệnh Insert

Thêm dữ liệu (thêm dòng mới) vào bảng.

Chú ý:

- Dữ liệu giống hoặc tương ứng với kiểu dữ liệu đã khai báo của cột đó, nếu không đúng lỗi sẽ phát sinh. Có thể không Insert được.

- Phải có quyền khi Insert (quyền này do người quản trị cung cấp)

# Câu lệnh Insert (2)

Khi Insert dữ liệu vào bảng, có 3 trường hợp sảy ra:

1. Insert dữ liệu vào bảng từ các giá trị cụ thể.

2. Insert vào bảng lấy giá trị từ một bảng hay nhiều bảng khác.

3. Kết hợp của cả hai trường hợp trên.

# Câu lệnh Insert (2.1)
# Insert dữ liệu vào bảng từ các giá trị cụ thể.

Dùng câu lệnh Insert để thêm một bản ghi mới vào bảng.

INSERT INTO

table_name (cot1,cot2...)

VALUES (giatriCot1,giatriCot2,...)

Ví dụ:

INSERT INTO employees(firstName,lastName)

values ('Trong Anh', 'Nguyen')

# Câu lệnh Insert vào bảng từ giá trị cụ thể của bảng khác (2.2)

Ví dụ:

Insert into employees (lastName, firstName)

select companyName, contactName from Customers

where customerID='ALFKI‘

Chú ý:

Ta có thểm chuyển toàn bộ dữ liệu companyName, contactName từ bảng Customers vào bảng Employees Với điều kiện customerID='ALFKI‘

# Kết hợp hai câu lệnh Insert ( 2.1 và 2.2)

Ví dụ:

Insert into employees (lastName, firstName)

select companyName, contactName from Customers

Chú ý:

Với câu lệnh trên ta chuyển toàn bộ dữ liệu companyName, contactName từ bảng Customers vào bảng Employees.

# Câu Lệnh Update

# Câu lệnh Update

Câu lệnh Update dùng để cập nhật dữ liệu đã có trong bảng.

Khi dùng UPDATE để cập nhật dữ liệu cho một bản ghi chỉ định nào đó thường UPDATE sử dụng với mệnh đề Where.

Nếu cần cập nhật tất cả các bản ghi trong bảng, ta có thể bỏ đi mệnh đề WHERE

Chú ý: UPDATE có thể ảnh hưởng đến nhiều bảng

# Câu lệnh UPDATE

Cập nhật cột với giá trị cụ thể

update shippers set companyName = 'Federal Shipping: Test' where shipperID =3

Cập nhật một cột với giá trị từ cột khác của bảng

UPDATE Northwind.dbo.Products SET UnitPrice = UnitPrice * 1.1 WHERE CategoryID = 2

# Câu lệnh UPDATE

Cập nhật dữ liệu với giá trị từ bảng khác

```
UPDATE titles
SET t.ytd_sales = t.ytd_sales + s.qty FROM titles
t, sales s
WHERE t.title_id = s.title_id AND s.ord_date =
(SELECT MAX(sales.ord_date) FROM sales)
```

Ở đây ta sử dụng các điều kiện để cập nhật.

# Câu Lệnh Delete

# Câu lệnh xóa - DELETE

Khi thực hiện xoá bản ghi trong bảng chúng ta cần quan tâm đến tên bảng, và mệnh đề WHERE để lọc các bản ghi nếu có.

Với điều kiện WHERE giống như vất kỳ mệnh đề WHERE nào có trong SELECT, UPDATE, INSERT.

Ghi chú: Không có khái niệm xoá giá trị trong 1 cột, vì xoá 1 cột đồng nghĩa với cập nhật cột đố bằng giá trị rỗng.

# Câu lệnh xóa - DELETE

Xoá bản ghi với giá trị cụ thể

Delete from shippers where shipperID =3

Delete from shippers where shipperID is Null

Xoá bản ghi từ bản ghi con.

```
DELETE [Order Details] FROM Suppliers,
Products WHERE Products.SupplierID =
Suppliers.SupplierID AND
Suppliers.CompanyName = 'Lyngbysild' AND
[Order Details].ProductID = Products.ProductID
```

# Câu lệnh xóa - DELETE

Xoá bản ghi từ bản ghi cha.

DELETE Products FROM Suppliers WHERE Products.SupplierID = Suppliers.SupplierID AND Suppliers.CompanyName = 'Lyngbysild'

Thực hiện DELETE với điều kiện xoá bản ghi mức cao hơn (mức cha)

DELETE Suppliers WHERE CompanyName = 'Lyngbysild'

# Các hàm thông dụng trong SQL 2000 Server

# Các thường được dùng trong Group By

Hàm AVG: trả về giá trị trung bình của cột hay trường trong câu truy vấn.

- SELECT AVG(price) FROM titles WHERE type = 'business'

Hàm Min: Hàm trả về giá trị nhỏ nhất của cột hay trường trong câu truy vấn

- SELECT MIN(au_lname) FROM authors

Hàm Max: Hàm trả về giá trị lớn nhất của cột hay trường trong câu truy vấn

- SELECT MAX(au_lname) FROM authors

# Các thường được dùng trong Group By

Count: Trả về số lượng bản ghi trong câu truy vấn trên bảng.

- SELECT COUNT(*) FROM titles

*Chú ý:*

```
Select customerID, count(employeeID)
from orders
group by customerID
```

Sum: trả về tổng các giá trị của trường, cột trong câu truy vấn.

- select sum(supplierID) from products
- select sum(supplierID) from products where supplierID=1

# Các hàm sử lý chuỗi

Hàm ASCII

Hàm trả về giá trị mã ASCII của ký tự bên trái của chuỗi.

Ví dụ: print ASCII('A') = print ASCII('Aa')

Kết quả trả về: 65

Hàm Char

Chuyển đổi kiểu mã ASCII sang dạng chuỗi

Ví dụ: print char('65')

Kết quả trả về: A

# Các hàm sử lý chuỗi (2)

Hàm Upper

Chuyển đổi sang kiểu chữ hoa

Ví dụ: print upper('sql server')

Kết quả trả về: SQL SERVER

Hàm LOWER

Chuyển đổi sang kiểu chữ thường

Ví dụ: print upper(‘SQL server')

Kết quả trả về: sql server

# Các hàm sử lý chuỗi (3)

Hàm len: trả về độ dài của chuỗi.

Thủ tục LTRIM: loại bỏ các khoảng trắng bên trái

Thủ tục RTRIM: loại bỏ các khoảng trắng bên phải.

# Các hàm sử lý thời gian

Những hàm hay dùng

- getDate(): Trả về ngày tháng năm hệ thống (ngày hiện tại)
- Day(): Trả về ngày
- Month(): Trả về tháng
- Year(): Trả về năm

# Hàm xử lý toán học

Những hàm hay dùng

Square(): Trả về bình phương của một biểu thức

Ví dụ: Print Square(4) ➔ 16

Sqrt(): Trả về căn bậc hai của một biểu thức

Ví dụ: Print Sqrt(16) ➔ 4

Round(): Làm tròn số

Ví dụ: Print Round(10.589,2) ➔ 10.590

# Tóm tắt T-SQL

1.Câu lệnh SELECT (truy vấn dữ liệu)
2.Câu lệnh INSERT (thêm mới dữ liệu)
3.Câu lệnh UPDATE (cập nhật dữ liệu)
4.Câu lệnh DELETE (xoá dữ liệu)

Những câu lệnh trên kết hợp với các biểu thức toán học, các biểu thức logic, các toán tử tạo nên những câu truy vấn dữ liệu linh hoạt

# 4

# Cách tạo và làm việc với Chỉ Mục (INDEX)

**Nguyễn Trong Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com
**Home:** http://www.tronganh.com

# Các đặc điểm của chỉ mục

- Cung cấp một tập các con trỏ logic chỉ tới dữ liệu
- Truy xuất dữ liệu nhanh hơn bình thường.
- Làm tăng tốc độ khi kết nối hai bảng
- Tạo tính duy nhất cho mỗi dòng

# Các đặc điểm của chỉ mục (2)

- Tự động cập nhật chỉ mục khi sửa dữ liệu

- Thời gian và tài nguyên cho việc tạo chỉ mục là nhiều. Nếu không sử dụng thường xuyên ta không nên tạo chỉ mục

# Cấu trúc và hoạt động của chỉ mục

- Chỉ mục được lưu trữ theo cấu trúc của cây nhị phân.
- Mỗi cấp của chỉ mục là một danh sách liên kết đôi

- Khi hiệu chỉnh dữ liệu trong bảng, mỗi chỉ mục của bảng cũng được hiệu chỉnh. SQL đảm bảo tính nhất quán giữa dữ liệu ghi trong các bảng và các chỉ mục giữa chúng.

- Nếu có hai chỉ mục trên một bảng, việc thêm 1 dòng thực hiện ít nhất hai tác vụ I/Os. ➔ chúng ta phải cân nhắc giữa việc cập nhật dữ liệu hay truy vấn nhanh.

# Các tuỳ chọn của chỉ mục

- Chỉ mục clustered
- Chỉ mục Non-Clustered

# Các tuỳ chọn (Clustered)

Sắp xếp lại dữ liệu về mặt vật lý.

Chỉ có duy nhất một clustered trong một bảng

Yêu cầu ít nhất 120% kích thước của bảng có thể sử dụng trong vùng tạm.

Khoảng trống vùng tạm tồn tại trong CSDL tạo chỉ mục mà chúng ta tạo chỉ mục.

# Các tuỳ chọn (Non - Clustered)

- Thứ tự các dòng trong bảng không sắp xếp về mặt vật lý giống Clustered
- Đây là kiểu điển hình để tạo chỉ mục cho cột liên kết các cột khác. Các giá trị có thể được thay đổi thường xuyên.
- SQL server sử dụng mặc định khi tạo chỉ mục là non – clustered
- Chúng ta có thể tạo 249 non – clustered cho mỗi bảng

# Chỉ mục (Index)

Chỉ mục Clustered tạo trước Non - Clustered

Dùng chỉ mục Non – Clustered để tạo Foreign key

Cú pháp:

```
CREATE [UNIQUE]
  [CLUSTERED|NONCLUSTERED] INDEX
  ten_chimuc
  ON ten_bang( ten_cot[,ten_cot]…)
```

# Tạo chỉ mục

Tạo chỉ mục có tên là CallIndex
Tại Bảng TestCalls
tại trường (CallID)

Create nonclustered index CallIndex
on TestCalls(CallID)

# Xem chỉ mục đã được Tạo

Cú pháp:

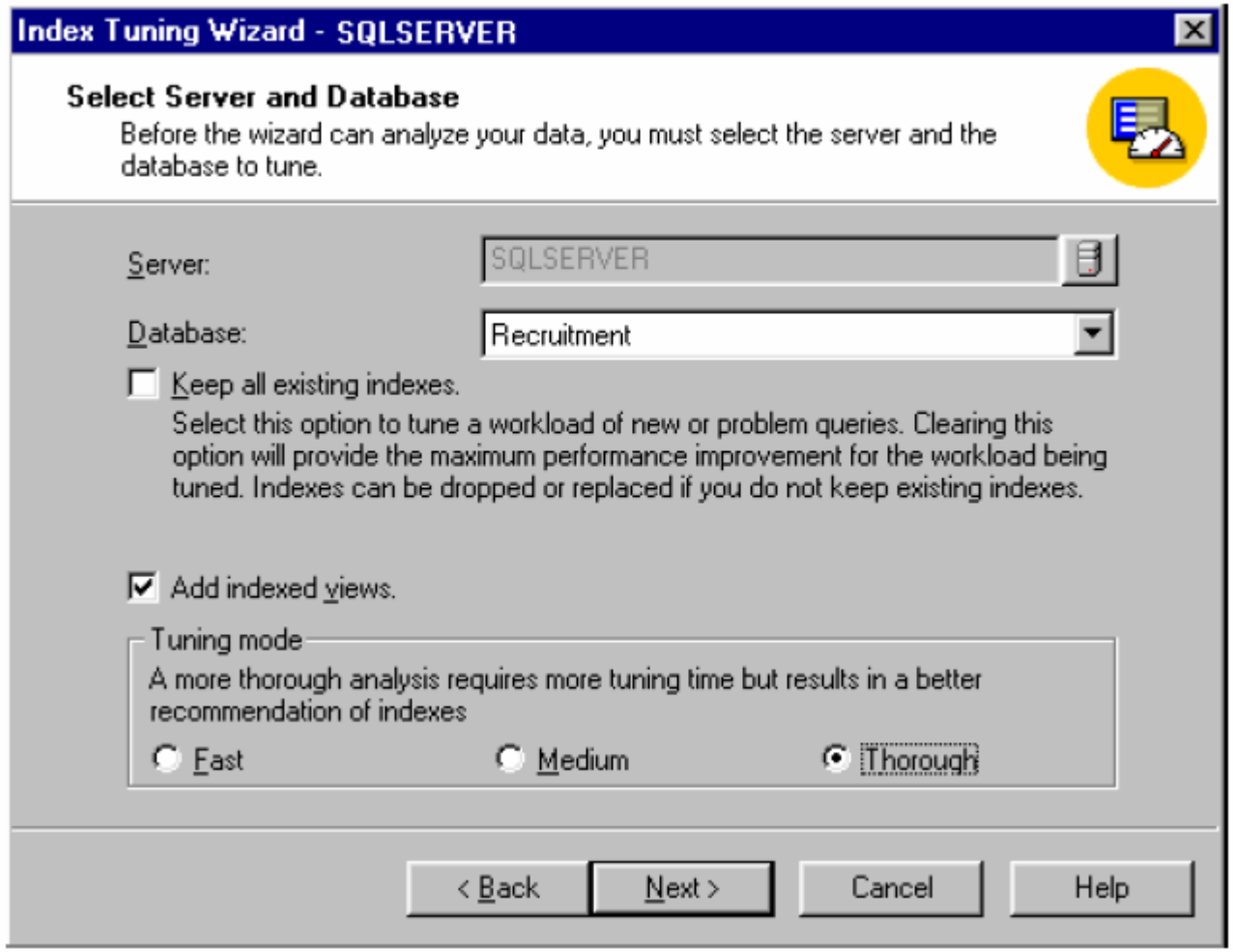
sp_helpindex TenBang

Xem trong bảng TestCalls có bao nhiêu chỉ mục và đó là những chỉ mục nào

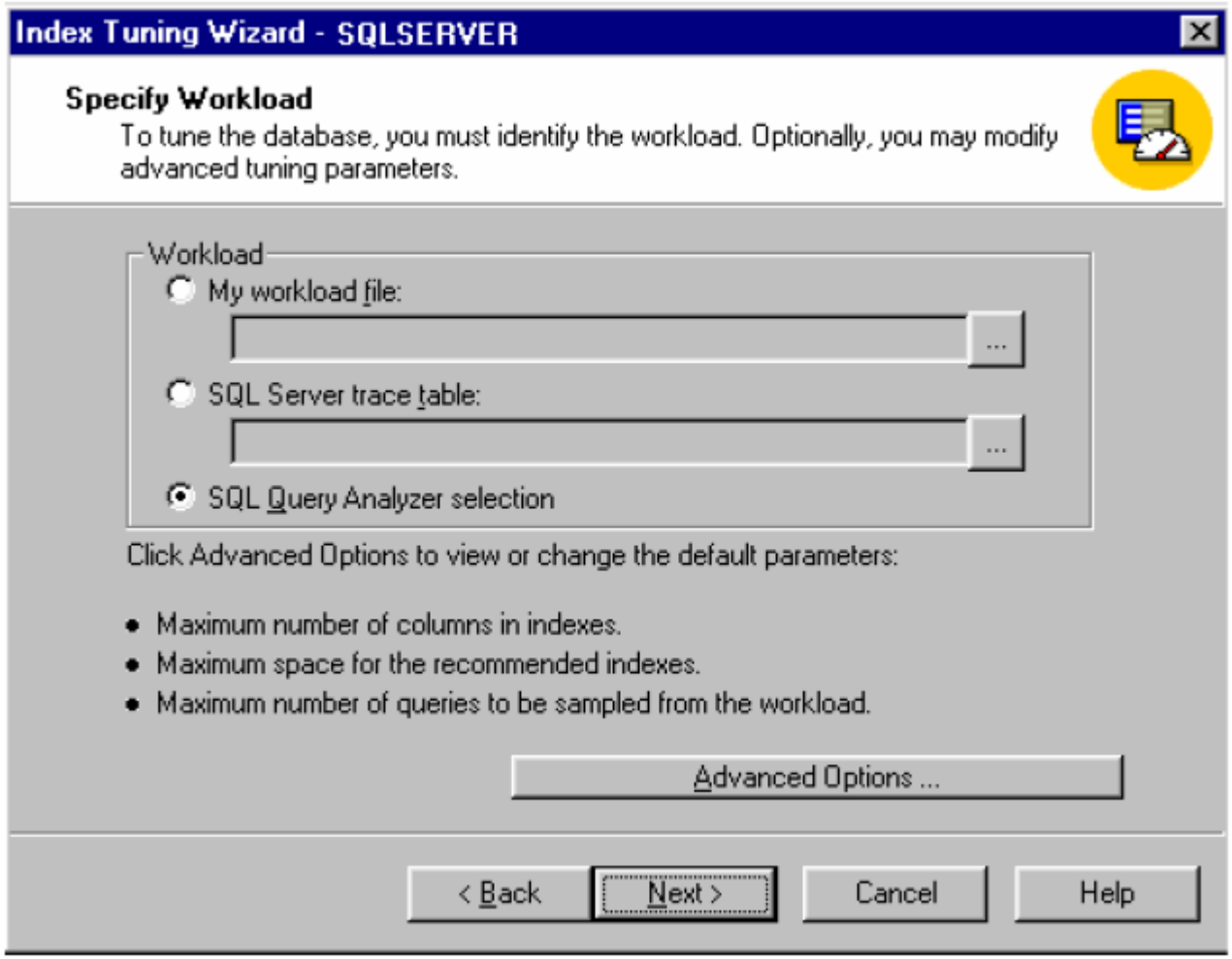
sp_helpindex TestCalls

# Sử dụng Index Tuning Wizard

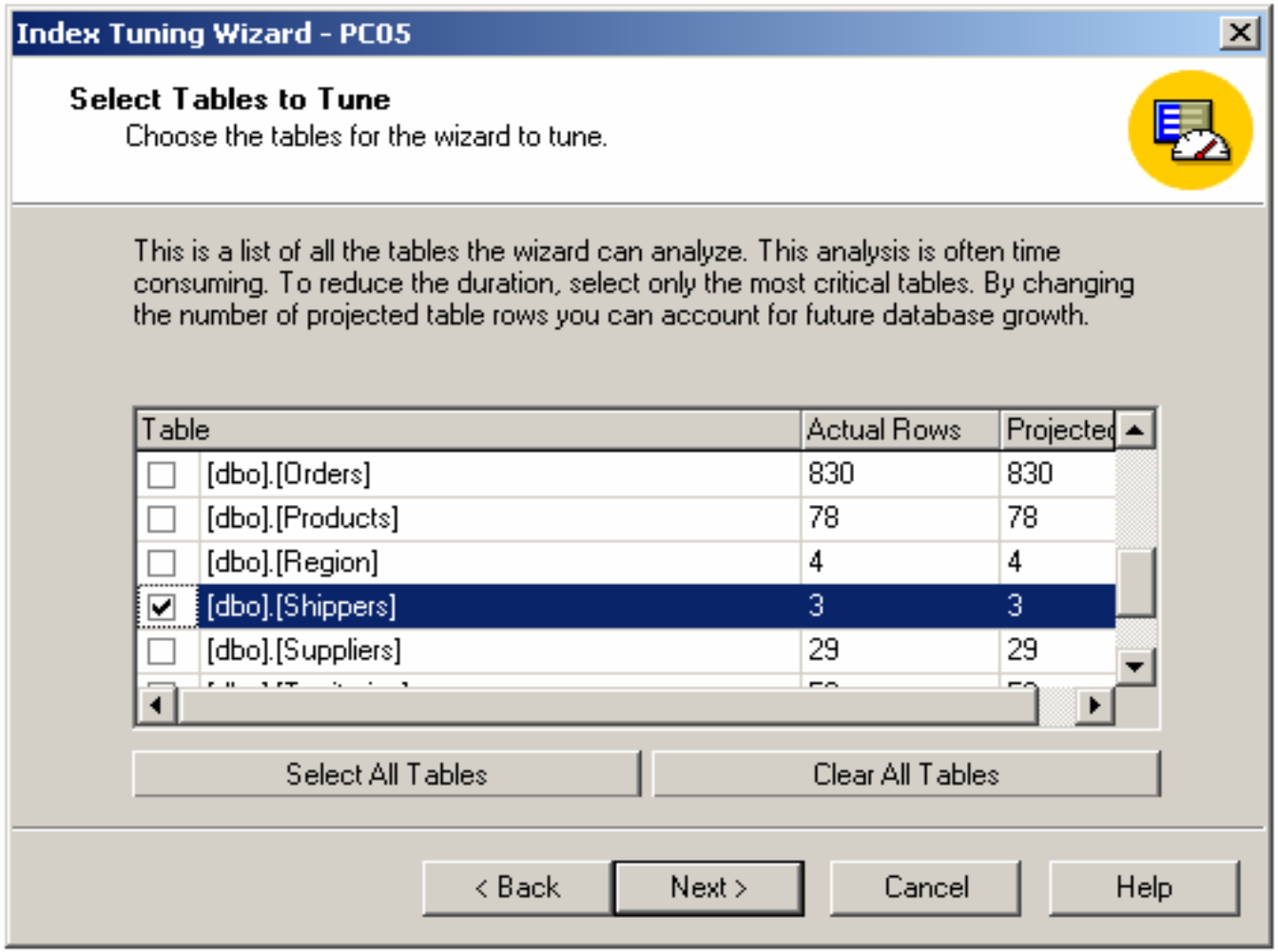
# Sử dụng Index Tuning Wizard (2)



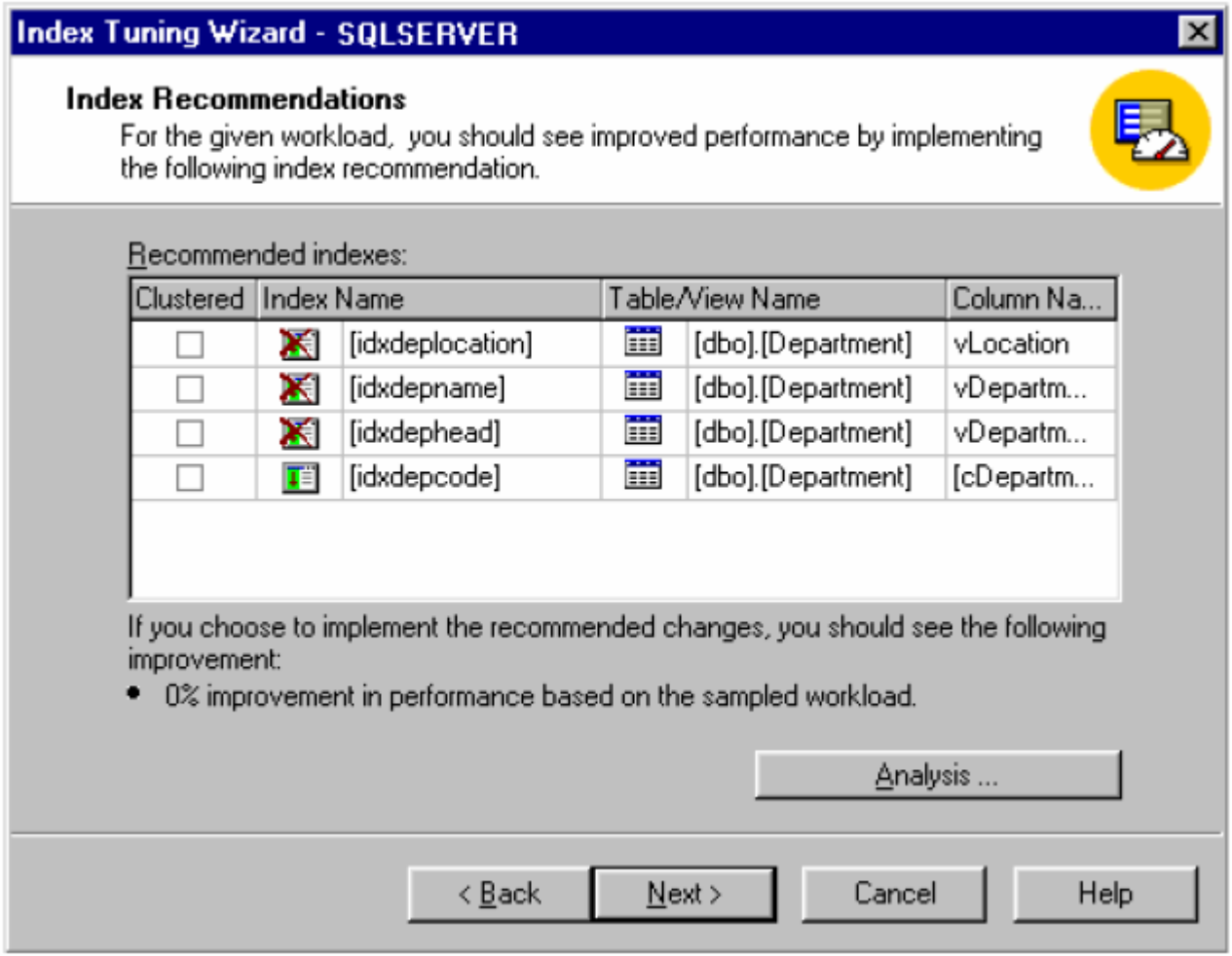
# Sử dụng Index Tuning Wizard (3)



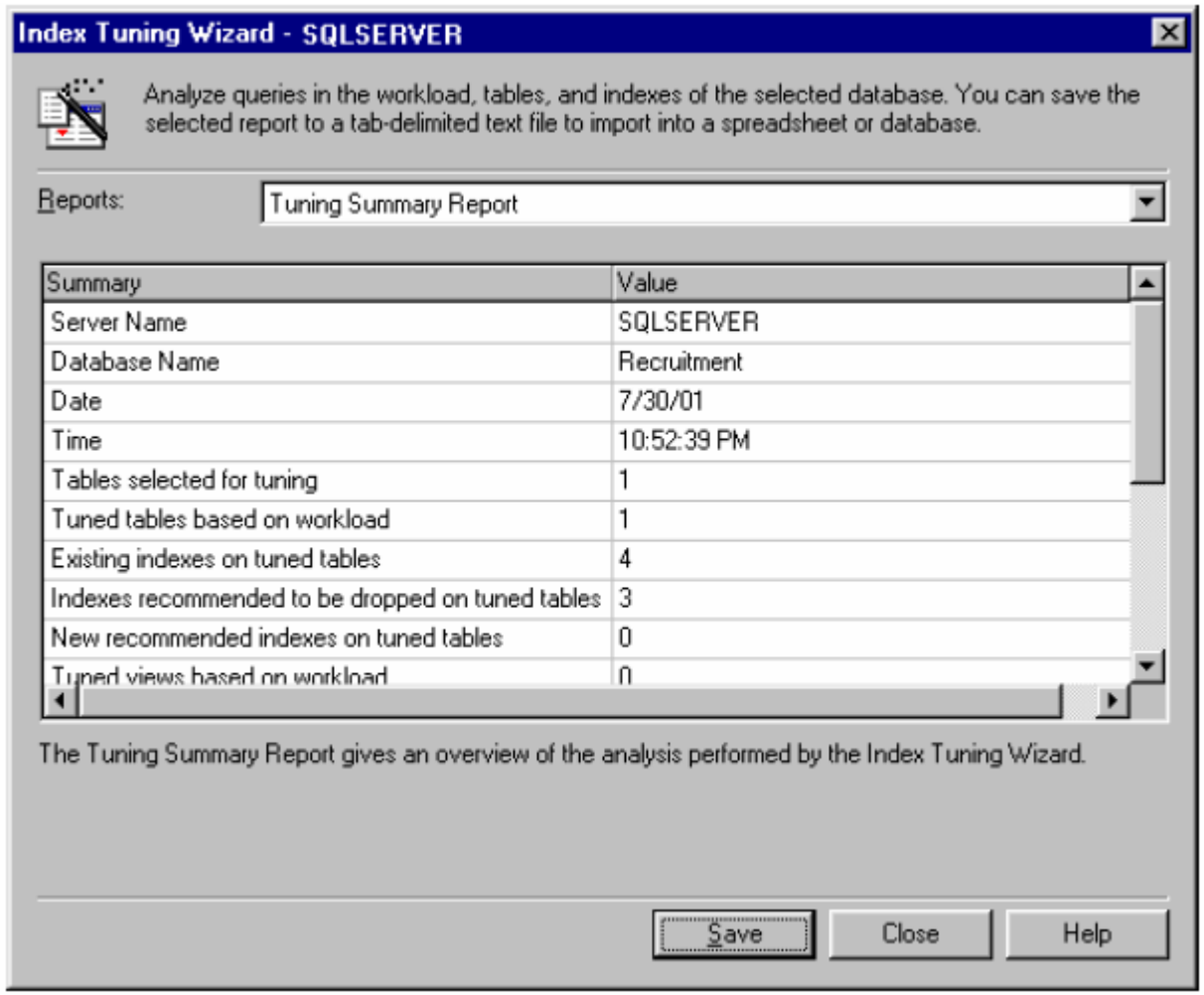
# Sử dụng Index Tuning Wizard (4)



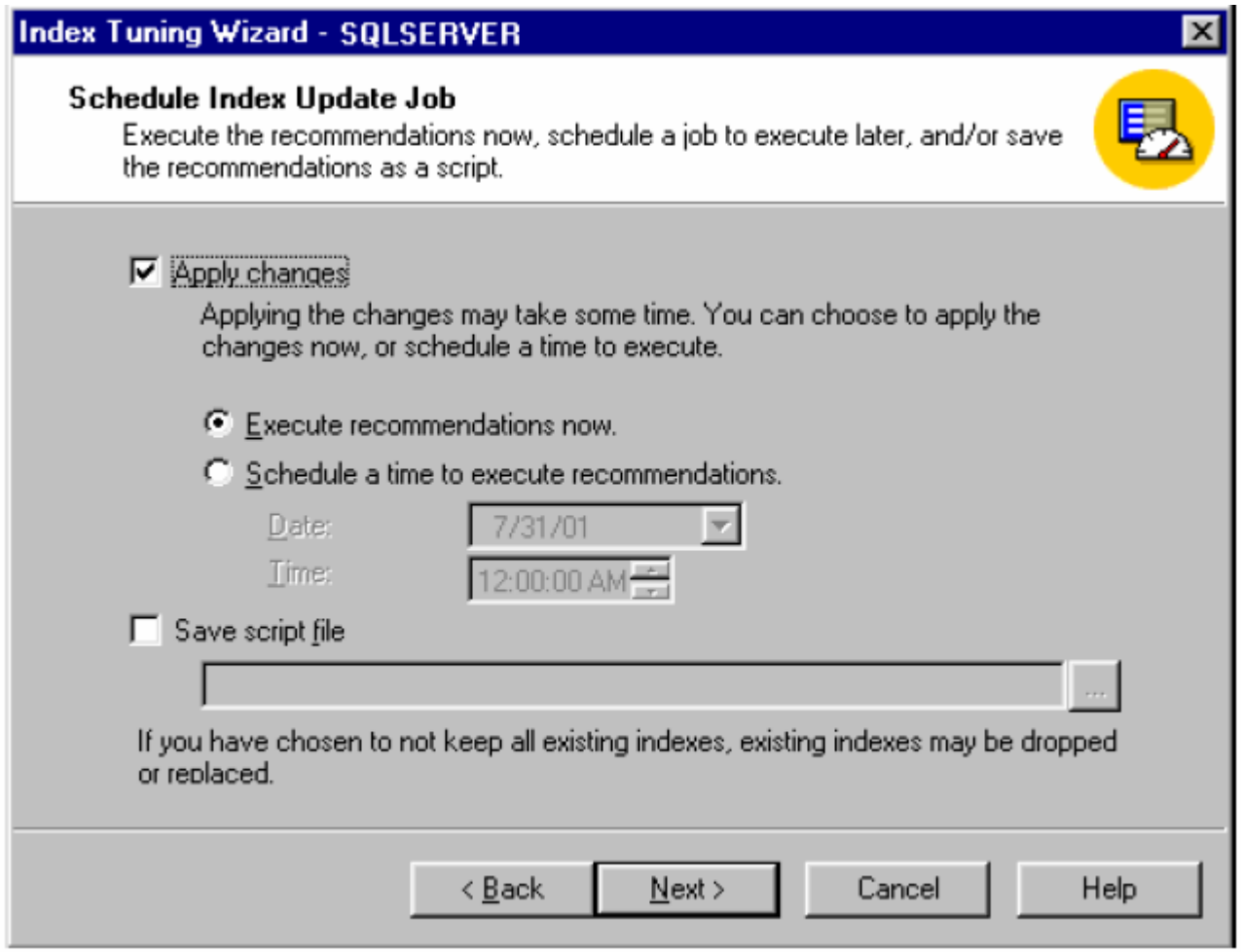
# Sử dụng Index Tuning Wizard (5)



# Sử dụng Index Tuning Wizard (6)



# Sử dụng Index Tuning Wizard (7)



# Sử dụng Index Tuning Wizard (8)

# 5 Khoá và Ràng buộc dữ liệu

**Nguyễn Trong Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com
**Home:** http://www.tronganh.com

# Các vấn đề chính

- Khái niệm cơ bản về ràng buộc
- Các loại ràng buộc
- Ràng buộc dữ liệu nhập
- Khoá chính và khoá ngoại

# Khái niệm cơ bản về ràng buộc

Ràng buộc dùng để kiểm tra khi có sự biến đổi dữ liệu như thêm vào, xoá, cập nhật từ bất kỳ các nguồn khác nhau truy cập đến CSDL.

Nếu dữ liệu thêm vào, xoá, hay cập nhật không thoả mãn các điều kiện hoặc quy luật đã định, tuỳ vào nhóm phân lỗi mà SQL sinh ngoại lệ nhằm thông báo cho người dùng biết. Dữ liệu khi đó sẽ không được phép cập nhật và thay đổi trong CSDL.

# Các loại ràng buộc ở mức cao

Ràng buộc ở mức cao bao gồm:

- Ràng buộc miền – Domains constraints
- Ràng buộc thực thể - Entity constraints
- Ràng buộc toàn vẹn dữ liệu

# Ràng buộc miền - Domains constraints

Liên quan đến một hay nhiều cột.

Ứng với mỗi cột cụ thể có các quy luật hay tiêu chuẩn.

Khi thêm hay cập nhật bản ghi mà không quan tâm đến sự liên quan đến các bản ghi trong bảng.

# Ràng buộc thực thể - Entity constraints

Kiểm tra số liệu xem có đúng chuẩn hay không?

# Ràng buộc toàn vẹn dữ liệu

Kiểm tra giá trị của cột có phù hợp với cột trong bảng khác quan hệ với bảng hiện tại chứa cột ràng buộc hay không

# Các loại ràng buộc ở mức đặc thù

Một số phương thức ràng buộc được thiết lập bao gồm.

- Ràng buộc khoá chính
- Ràng buộc khoá ngoại
- Ràng buộc duy nhất
- Ràng buộc kiểm tra
- Ràng buộc mặc nhiên

# Ràng buộc khoá chính – Primary key constraints

- Giá trị cột phải  duy nhất
- Không lặp lại
- Tuân theo các ràng buộc với những bảng quan hệ
- Khôngng thể NULL

Khi thêm dữ liệu, hay sửa đổi, quá trình kiểm tra ràng buộc sẽ diễn ra.

# Ràng buộc khoá chính – Primary key constraints

```
CREATE TABLE [dbo].[Shippers2] (
    [ShipperID] [int] IDENTITY (1, 1) NOT NULL
Primary key,
    [CompanyName] [nvarchar]  (50),
    [Phone] [nvarchar] (24)
)
```

# Ràng buộc Cập nhật, xoá mẩu tin

Trong trường hợp quan hệ một nhiều (có ràng buộc)

- Không xoá được những bản ghi 1 trước được
- Xoá các bản ghi nhiều (những bản ghi phụ thuộc vào bản ghi 1 trươc)
- Khi xoá dữ liệu, thêm dữ liệu quá trình kiểm tra dữ liệu sẽ diễn ra

# Ràng buộc duy nhất - Unique constraints

Giống ràng buộc khoá chính

Chỉ yêu cầu giá trị duy nhất chứa trong mỗi cột trong bảng dữ liệu

Khi xoá thêm dữ liệu quá trình kiểm tra dữ liệu sẽ diễn ra, nếu trùng với dữ liệu trong cột. Lỗi sẽ phát sinh

# Ràng buộc mặc định – Default constraints

Giống như các ràng buộc khác, nó giúp định nghĩa của bảng có dữ liệu phù hợp ngay cả khi không có người dùng nhập vào

Căn cứ vào giá trị mặc định được khai báo trong cột của bảng, khi mẩu tin thêm vào, nếu bạn cung cấp giá trị ứng với cột đó rỗng, giá trị mặc định sẽ được sử dụng

# Những đặc điểm chính của ràng buộc Default

- Giá trị mặc định chỉ dùng cho trường hợp thêm bản ghi mới
- Không quan tâm đến các hành động cập nhật hay xoá.
- Nếu giá trị đưa vào là khác rỗng, giá trị mặc định sẽ không được sử dụng
- Nếu giá trị đưa và là rỗng, giá trị mặc định sẽ được sử dụng

# Ràng buộc mặc định – Default constraints

Giá trị mặc định (10) sẽ tự động thêm vào nếu là NULL

```
CREATE TABLE [dbo].[Shippers3] (
    [ShipperID] [int] IDENTITY (1, 1) NOT NULL
Primary key,
    [CompanyName] [nvarchar] (50) ,
    [Phone] [nvarchar] (24) Default '10'
)
```

# Ràng buộc kiểm tra - Check Constraints

Không sử dụng dữ liệu không mong muốn

Tránh những bản ghi không phù hợp

Khi cập nhật, thêm dữ liệu quá trình kiểm tra dữ liệu sẽ diễn ra, nếu không thoả mãn với điều kiện đưa ra. Lỗi sẽ phát sinh. ➔ Không thao tác được

# Ràng buộc kiểm tra - Check Constraints

Dữ liệu kiểm tra đối với lương (salary_cap < 100)

```
CREATE TABLE cnst_example22
(id INT NOT NULL,
 name VARCHAR(10) NOT NULL,
 salary MONEY NOT NULL
    CONSTRAINT salary_cap CHECK (salary < 100)
)
```

7

# Thủ tục thường trú

## STORED PROCEDURES

**Nguyễn Trong Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com
**Home:** http://www.tronganh.com

# Thủ tục thường trú

Là tập hợp các T-SQL được lưu trữ trong tên, được thực hiện như một đối tượng riêng biệt

# Ưu điểm

Tăng tốc độ thực hiện

Giảm lưu lượng dao dịch trên mạng

Toàn vẹn dự liệu tăng

Bảo mật tốt hơn

# Dạng của Thủ tục thường trú

- Người dùng định nghĩa (user – define)
- Hệ thống (system)
- Tạm thời (temporary)
- Tách biệt (remote)
- Mở rộng (Extended)

# Thủ tục thường trú
# Stored Procedure

- Khái niệm cơ bản về thủ tục thường trú
- Thay đổi và xoá một thủ tục thường trú
- Tham số và khai báo biến
- Phát biểu có cấu trúc
- Một số thủ tục thường trú cơ bản
- Một số thủ tục thường trú của hệ thống

# Khái niệm

Thủ tục thường trú là một đối tượng xây dựng bởi những phát biểu của SQL server và T-SQL

Thủ tục thường trú được lưu trữ như một phần của cơ sở dữ liệu.

Cấu trúc như là văn bản Text, mỗi khi thực hiện chỉ cần gọi tương tự như thủ tục hoặc hàm trong các ngôn ngữ lập trình

# Cú pháp để tạo thủ tục thường trú

```
CREATE PROCEDURE ten_thutuc
[<Cac tham so>],
[<Cac gia tri mac dinh>]
AS
BEGIN
        caulenh_sql1
        caulenh_sql2
 END
```

# Xác định các thông tin cần thiết để tạo thủ tục thường trú

Nơi tạo thủ tục thường trú: Cơ sở dữ liệu NorthWind

Kiểu của thủ tục thường trú: user-defined

Tên của thủ tục thường trú: sp_Hienthi

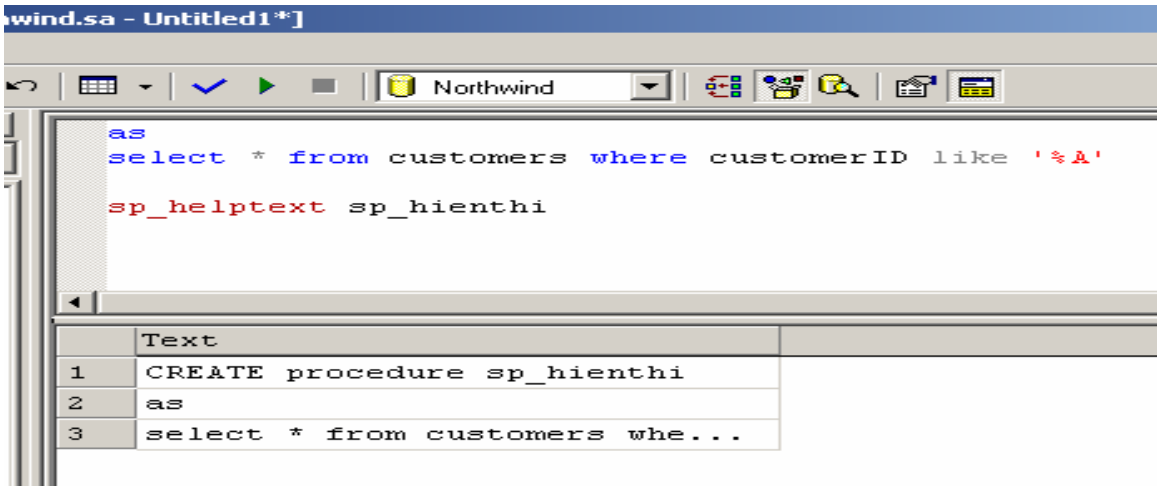
# Nội dung của thủ tục sp_hienthi

```
Create procedure sp_hienthi
As
Begin
    select * from customers where customerID like
'%A‘
End
```

# Xem nội dung của thủ tục thường trú

Sp_helptext sp_hienthi

Nội dung của thủ tục sẽ được hiển thị như hình vẽ dưới đây:



# Gọi thủ tục thường trú

Câu lệnh EXECUTE PROCEDURE thường được dùng để gọi thủ tục thường trú

Cú pháp:

*EXECUTE ten_thutuc*

*hoặc*

*EXEC ten_thutuc*

*hoặc*

*ten_thutuc*

# Tham số trong thủ tục thường trú

Khi thực hiện một thủ tục thường trú, chúng ta có thể truyền tham số để thông báo cho thủ tục thường trú

Có hai loại tham số

- Input parameter
- Output parameter

# Tham số trong thủ tục thường trú

Khi sử dụng tham số trong thủ tục thường trú, chúng ta phải qua. Để khai báo một tham số, chúng ta cần quan tâm đến các yếu tố sau:

- Tên tham số
- Kiểu dữ liệu
- Giá trị mặc nhiên nếu có
- Có hay không chỉ dẫn OUTPUT

# Tham số trong thủ tục thường trú (Khai báo tham số)

Cú pháp:

@ten_thamso [AS] kiểudữliệu

Ví dụ khai báo tham số

@SoDienThoai varchar(20),

@Diachi AS varchar(50)

Khi có nhiều tham số chúng ta sử dụng dấu (,) để phân cách các tham số

# Tham số trong thủ tục thường trú (Khai báo tham số trong thủ tục)

```
Create procedure sp_hienthiTS
@Val Varchar(20)
as
Begin
 select * from customers where customerID like
'%' + @Val
End
```

# Tham số trong thủ tục thường trú (Gọi thủ tục với tham số)

Xem nội dung thủ tục

Sp_helptext sp_hienthi

Gọi thủ tục

Sp_thutuc ‘A’

Ý nghĩa: Hiển thị thông tin của khách hàng. Với điều kiện CustomerID  có ký tự A

# Tham số trong thủ tục thường trú (Khai báo tham số trong thủ tục)

Những tham số được truyền từ bên ngoài vào thủ tục. Các tham số có thể lấy giá trị truyền vào tư ngôn ngữ lập trình. Ví dụ như: Visual Basic, Visual Basic.Net, C# …

Giá trị truyền vào phải đúng thứ tự như đã khai báo. Nếu giá trị truyền vào không tương thích với kiểu dữ liệu đã được khai báo. Lỗi sảy ra ➔ Không chạy được thủ tục

# Xây dựng thủ tục với các giá trị mặc định

Khi khai báo tham số trong thủ tục thường trú, nếu cần chúng ta có thể khởi tạo giá trị mặc nhiên cho tham số.

Khi gọi thủ tục có gán giá trị ngẫu nhiên, nếu người sử dụng không cung cấp giá trị. Nó sẽ lấy giá trị mặc định được định nghĩa trước đó.

# Tạo thủ thục tham số  với giá trị mặc định

```
CREATE procedure sp_InsertShipper
@company Varchar(20) = 'N/A',
@Phone Varchar(20) = 'N/A'
AS
Begin
 Insert Into Shippers(CompanyName,Phone)
 Values(@company, @phone)
End
GO
```

# Tạo thủ thục tham số  với giá trị mặc định

Thủ tục trên thêm mới 1 bản ghi vào bảng Shippers

Nếu có giá trị truyền vào. Thủ tục sẽ lấy các giá trị được truyền vào.

Giá trị truyền vào khi gọi tham số cho companyName và Phone sẽ được lấy mặc định nếu là rỗng

# Kiểm tra kết quả của thủ tục

Chúng ta chạy Thủ tục trên với các tham số dưới đây:

sp_InsertShipper

sp_InsertShipper  ‘Dai Học Thang Long’

sp_InsertShipper ‘DH Thang Long’, ‘858 789 989’

# Thay đổi thủ tục thường trú

Những lưu ý khi thay đổi nội dung của Thủ tục thường trú

- Thủ tục đó phải tồn tại.

- Tuỳ thuộc vào quyền hạn của người dùng đó có thể thay đổi thủ tục thường trú hay không.

- Kiểm tra tất cả các thông tin có liên quan đến các đối tượng khác trong khi bị thay đổi.

# Thủ tục thường trú với tham số (*Tham số Output*)

Lấy giá trị Output:

- Khi chúng ta cần xuất giá trị ra ngoài khi thủ tục thường trú thực thi xong.

- Sử dụng kết quả của thủ tục thường trú làm giá giá trị tham số đầu vào cho một thủ tục thường trú khác.

# Thủ tục thường trú với tham số (*Tham số Output*)

```
CREATE procedure sp_InsertShipper1
    @company Varchar(20) = 'N/A',
    @Phone Varchar(20) = 'N/A',
    @outPhone int OutPut
AS
Begin
   Insert Into Shippers(CompanyName,Phone)
   Values(@company, @phone)
   select @outPhone = @@IDentity
End
GO
```

# Thủ tục thường trú với tham số (*Lấy thông tin từ Tham số Output*)

```
declare @MyID int
exec sp_InsertShipper1 'Test1','Test2', @MyID
OutPut

select @myID as SoMauTin

select * from shippers where ShipperID = @myID
```

# Thủ tục thường trú với tham số (*Nhiều Tham số Output*)

```
CREATE PROCEDURE prcGetInfoShippers
        @ShipperID int,
        @Phone varchar(20) OUTPUT,
       @company varchar(20) OUTPUT
           AS
BEGIN
   SELECT
    @Phone = Phone,
    @company = CompanyName
    FROM Shippers
    WHERE ShipperID = @ShipperID
END
```

# Thủ tục thường trú với tham số (*lấy thông tin từ thủ tục nhiều Tham số Output*)

Khai báo biến

Declare @Phone Varchar(20)

Declare @Company Varchar(20)

Gọi Thủ tục

Exec prcGetInfoShippers 1, @Phone Output, @Company Output

In kết quả

Select @Phone as Phone, @Company as Company

# Những câu lệnh điều khiển

# Những câu lệnh điều khiển

Bất kỳ ngôn ngữ lập trình nào hiện nay đều có những phát biểu điều khiển.

SQL Server 2000 cũng giống như ngôn ngữ lập trình.

Chúng ta tìm hiểu các cú pháp thông dụng như:

- IF … ELSE
- GOTO
- WHILE
- WAITFOR

# Những câu lệnh điều khiển (2)

Bên cạnh đó còn có các câu lệnh điều khiển như:

- CASE …
- WHEN …
- THEN …
- ELSE … END

# Câu lệnh IF ... ELSE

Câu lệnh IF … ELSE được sử dụng nhiều trong các ngôn ngữ lập trình.

Câu lệnh IF … ELSE đóng vai trò rất quan trọng khi lập trình, ngay cả trong Stored Procedure (thủ tục thường trú), hoặc trong trigger (Chúng ta sẽ tìm hiểu phần này sau)

# Câu lệnh IF ... ELSE

Cú pháp:

IF <biểu thức logic>
    Begin <Các câu lệnh SQL> End
ELSE
    Begin <Các câu lệnh SQL> End

<Các câu lệnh SQL>: có thể là 1 hoặc nhiều câu lệnh SQL

# Câu lệnh IF ... ELSE (Minh hoạ)

```
Create PROCEDURE prcGetInfoShippers32
          @ShipperID int,
        @Phone char(20) OUTPUT
           AS
BEGIN
    IF EXISTS(SELECT * FROM Shippers WHERE ShipperID =
           @ShipperID)
    BEGIN
     SELECT
        @Phone = Phone
           FROM Shippers
      WHERE ShipperID = @ShipperID
      END
    ELSE
     Select @Phone = 'Not Found'
END
```

# Câu lệnh IF ... ELSE (Minh hoạ)

Xem kết quả gọi hàm (trường hợp có dữ liệu)

- Declare @Phone varchar(20)
- exec prcGetInfoShippers32 1,@phone OUTPUT
- Select @Phone as SoDienThoai

Xem kết quả gọi hàm (trường hợp không có dữ liệu)

- Declare @Phone varchar(20)
- exec prcGetInfoShippers32 0,@phone OUTPUT
- Select @Phone as SoDienThoai

# Câu lệnh CASE

- Câu lệnh CASE cho phép nhận một giá trị từ nhiều lựa chọn.
- Trường hợp có nhiều câu lệnh IF ELSE lồng nhau, gây cho đoạn chương trình phức tạp, bạn nên sử dụng câu lệnh CASE.

- Để dễ hiểu câu lệnh CASE. Cú pháp Câu lệnh CASE giống như các ngôn ngữ lập trình khác như: C, C++, Java …

# Cú pháp CASE

CASE <giá trị biểu thức Case>

- WHEN <điều kiện 1> THEN <kết quả tương ứng 1>
- WHEN <điều kiện 2> THEN <kết quả tương ứng 2>
- WHEN <điều kiện 3> THEN <kết quả tương ứng 3>
- …
- ELSE <kết quả tương ứng >

END

# Câu lệnh Case
## (*Ví dụ minh hoạ với Select*)

```
Create procedure sp_ConvertStatess
AS
Begin
 Select ShipperID, CompanyName, Phone,
 Case State
  WHEN 'CA' then  'Califorlia'
  WHEN 'KS' then  'Kansas'
  WHEN 'TN' Then 'Tennessee'
  WHEN 'OR' Then 'Oregon'
  WHEN 'MI' Then 'Michigan'
  WHEN 'IN' Then 'India'
  WHEN 'MD' Then 'Marylan'
  WHEN 'UT' Then 'Utah'
  ELSE 'Khong xac dinh'
 End
 From Shippers
End
```

# Câu lệnh Case
## (*Ví dụ minh hoạ với Select*)

Câu lệnh CASE với câu truy vấn SELECT ở trên sẽ có hai trường hợp sảy ra:

- Nếu Không có biểu thức ELSE, kết quả trả về chỉ tương ứng với các điều kiện của các biểu thức WHEN.

- Nếu có biểu thức ELSE, giá trị tương ứng của biểu thức ELSE sẽ được hiển thị khi không có các giá  trị trong các biểu thức WHEN

# Câu lệnh Case

## (*Ví dụ minh hoạ với khi tính toán với biểu thức logic*)

```
Create procedure sp_ConvertStateAndPrice
AS
Begin
 Select ShipperID, CompanyName, Phone,
Case
 WHEN Price < 0.5 Then 0.5
 WHEN Price Between 1.00 and 2.00 Then 2.3
 WHEN Price Between 2.3 and 4 Then 4
 ELSE 5
 END
 From Shippers
End
```

# Ví dụ với 2 CASE

```
Create procedure sp_ConvertStateAndPrice
AS
Begin
 Select ShipperID, CompanyName, Phone,
 Case State
  WHEN 'CA' then  'Califorlia'
  WHEN 'KS' then  'Kansas'
  WHEN 'TN' Then 'Tennessee'
  WHEN 'OR' Then 'Oregon'
  WHEN 'MI' Then 'Michigan'
  WHEN 'IN' Then 'India'
  WHEN 'MD' Then 'Marylan'
  WHEN 'UT' Then 'Utah'
  ELSE 'Khong xac dinh'
 End,
 Case
 WHEN Price < 0.5 Then 0.5
 WHEN Price Between 1.00 and 2.00 Then 2.3
 WHEN Price Between 2.3 and 4 Then 4
 ELSE 5
 END
 From Shippers
End
```

# Câu lệnh While

- Câu lệnh While là câu lệnh điều khiển vòng lặp.
- Vòng lặp sẽ được thực hiện cho đến khi biểu thức trong While sai.
- While được dùng nhiều trong kiểu dữ liệu CURSOR. Thông thường While thường dùng để duyệt từ bản ghi đầu tiên đến bản ghi cuối cùng hoặc ngược lại

# Cú pháp Câu lệnh While

While <biểu thức logic>
  <các câu lệnh SQL>
Begin
 <các khối lệnh bị cấm> (Block statements)
 <Breake>
 <Các câu lệnh SQL>
 [Continue]
End

# Cú pháp Câu lệnh While

BREAKE dùng để thoát khỏi vòng lặp While. Ví dụ khi trong thủ tục thường trú. Nếu gặp Breake nó sẽ thoát khỏi vòng lặp các lệnh phía sau sẽ bị bỏ qua.

Continue ngược lại với BREAKE, nếu gặp câu lệnh này thì quá trình xử lý sẽ quay lại đầu vòng lặp While

# Cú pháp Câu lệnh While (VD)

```
Create procedure sp_UpdateShipper
AS
Begin
 declare @i int
 set @i = 1
 While @i <= 5
 Begin
  update Shippers set price = price + 10 where shipperID =
@i
    set @i = @i+1
 End
End
```

# Câu lệnh RETURN

- Khi cần xác định kết quả đúng hoặc trả về một giá trị nào đó. Chúng ta sử dụng câu kệnh RETUNS

- Nếu gặp câu lệnh RETURN, quá trình sử lý được kết thúc.

- Trong thủ tục đôi khi chúng ta sử dụng RETURN để thủ tục trở thành hàm như các ngôn ngữ lập trình khác,

# Câu lệnh RETURN

Cú pháp:

Return <giá trị nguyên>

Nếu không cung cấp giá trị trả về cho câu lệnh Return, giá trị trả về sẽ là 0

# Câu lệnh Return (minh hoạ) (giá trị trả về là giá trị chỉ định)

Tính tổng số lượng trong bảng Shippers

```
Create procedure sp_GetSumQuantity
AS
Begin
 declare @SumQuantity int
 Select @SumQuantity = Sum(Quantity) from
Shippers
 Return @SumQuantity
End
```

# Tính số ngày của tháng (Return)

```
Create Procedure sp_days
@thangnam varchar(7)
AS
Begin
 Declare @nam int
 Declare @thang int
 Declare @songay int
 set @nam = cast(right(@thangnam,4) as int)
 set @thang = cast(left(@thangnam,2) as int)
 Set @songay =
 Case
   When @thang in (1,3,5,7,8,10,12) then 31
   When @thang in (4,6,9,11) Then 30
   When @thang in (2) and (@nam %4 = 0 and @nam%100 = 0)
then 29
   Else 28
 End
 Return @songay
End
```

# Tính số ngày của tháng (Return)

Chạy thủ tục trên

```
Declare @days int

exec @days = sp_days '10/2001'

Print 'So ngay cua thang la: ' + str(@days)
```

# Tính số ngày của tháng

Đối với thủ tục trên chúng ta đã sử dụng một số cú pháp mà chưa đề cập từ trước:

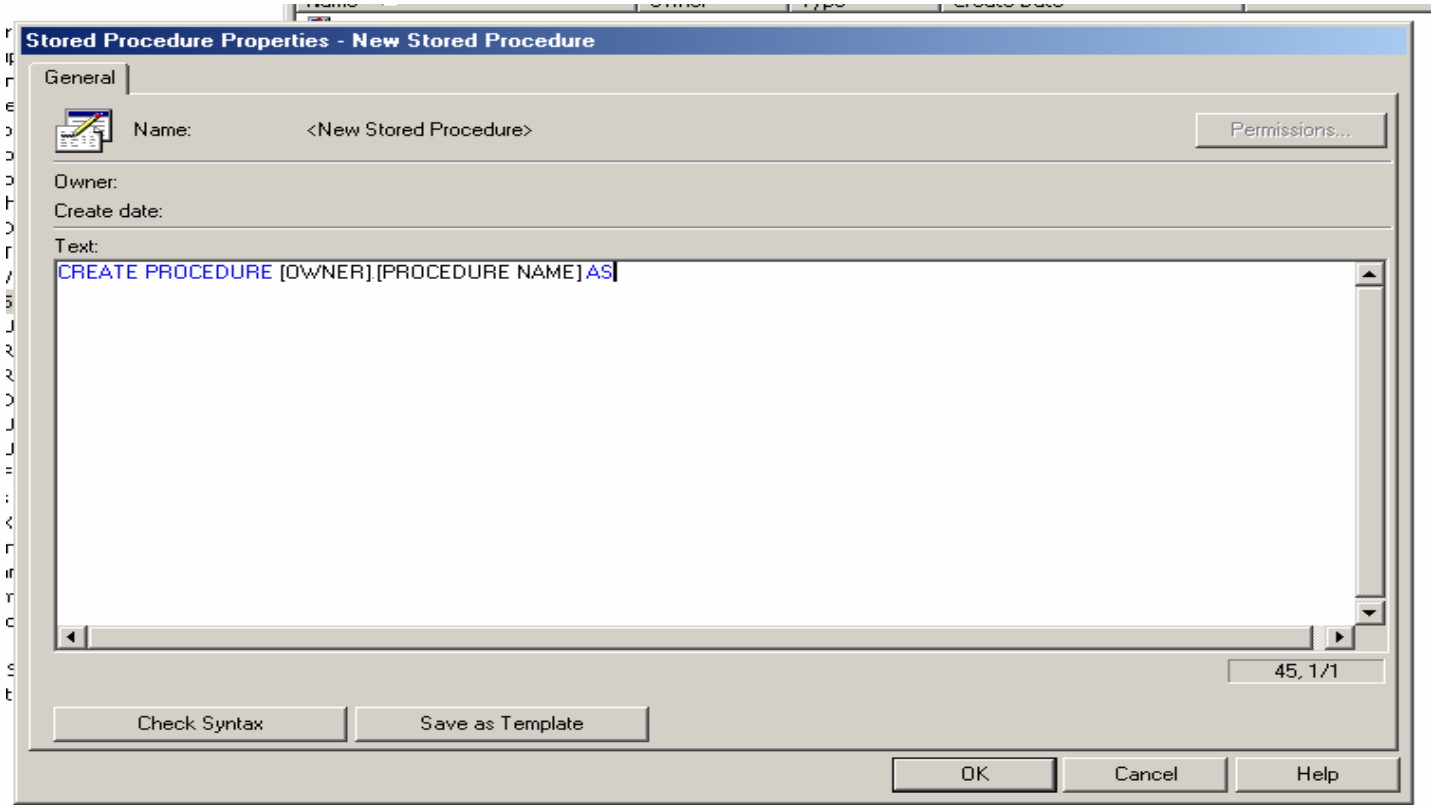
- Câu lệnh In trong CASE
- Hàm Cast chuyển đổi giá trị
- Hàm Left, right lấy chuỗi ký tự con
- Hàm str() chuyển giá trị về dạng ký tự
- Toán tử % lấy giá trị dư của phép chia

# Xây dựng Thủ tục thường trú bằng Enterprise Manager

- Chúng ta có thể xây dựng thủ tục thường trú bằng 2 cách, sử dụng Câu lệnh Create và dùng giao diện Enterprise Manager.
- Dù tạo bằng cách nào, thủ tục thường trú cũng trở thành một đối tượng của CSDL.
- Chúng ta cũng có thể kiểm tra các đối tượng băng Enterprise Manager

# Tạo Thủ Tục Thường Trú Bằng EM

Đưa chuột đến Mục Stored Procedure nhấp chuột phải chọn New Stored Procedure



# Tạo Thủ Tục Thường Trú Bằng EM

Màn hình soạn thảo sẽ như hình trên.

Nút Check Syntax là công cụ giúp chúng ta kiểm tra cú pháp của các câu lệnh trong thủ tục thường trú.

Sau khi soạn thảo chúng ta chỉ cần check Syntax để kiểm tra cú pháp trước khi lưu

# Thay đổi thủ tục thường trú

Cú pháp:

```
Alter Procedure Ten_thutuc
[<cac tham so>]
[<cac gia tri mac dinh>]
As
Begin
   caulenh_sql1
   caulenh_sql2
End
```

# Thay đổi thủ tục thường trú trong Enterprise Manager

Khi thay đổi trong Enterprise Manager chúng ta chỉ cần nhấp đúp chuột vào thủ tục thường trú muốn thay đổi

Nội dung của thủ tục thường trú được hiển thị

Check Syntax trước khi lưu

# Xoá thủ tục thường trú

Kiểm tra chắc chắn các thông tin của thủ tục trước khi xoá, kể cả về nội dung và ảnh hưởng của thủ tục đó đối với hệ thống.

Cú pháp:

Drop Procedure ten_thutuc

# Xoá thủ tục thường trú trong Enterprise Manager

Khi xoá thủ tục thường trú trong Enterprise Manager ta làm như sau:

Đưa chuột đến thủ tục thường trú rồi nhấp chuột phải tại thủ tục cần xoá.

Chọn Delete trong menu

Thủ tục sẽ bị xoá nếu người dùng có quyền được xoá Thủ tục thường trú

# Một Số
# STORED PROCEDURE
# Của Hệ Thống

# Một số Stored Procedure của hệ thống

- Trong cơ sở dữ liệu SQL server có rất nhiều Stored Procedure hệ thống. Chúng ta có thể gọi những thủ tục thường trú này khi cần thiết.

- Trong một tình huống nào đó, giả sử cần đến các thông tin liên quan đến hệ thống SQL server hay CSDL, những thủ tục này sẽ giúp ích rất nhiều cho mục đích của chúng ta

# Một số Stored Procedure của hệ thống

Sp_who2;

Trả về liệt kê ra tất cả các người dùng truy cập đến SQL server hay cơ sở dữ liệu

Sp_configure;

Muốn biết cấu hình của SQL server 2000 đang dùng.

# Một số Stored Procedure của hệ thống

- Sp_tables: Cung cấp thông tin về bảng
- Sp_password: cho phép thay đổi password của một user
- Sp_helptext: hiển thị nội dung của View và thủ tục thường trú
- Sp_helpdb: hiển thị nội dung của cơ sở dữ liệu.
- Sp_dropuser: Xoá một user
- Sp_columns: hiển thị các cột của bảng
- Sp_adduser: thêm một user vào CSDL
- Sp_addlogin: tạo một user

# Một số bảng chứa các thông tin hệ thống

# Thông tin hệ thống

- Trong SQL server có một bảng chứa các thông tin hệ thống SQL server hay thông tin của các cơ sở dữ liệu thành phần.

- Bên cạnh đó còn có các bảng chứa các thông tin về những đối tượng của CSDL.

# Thông tin hệ thống

Syslogins: chứa danh sách người dùng trong cơ sở dữ liệu SQL Server

Sysobjects: Chứa danh sách các đối tượng trong CSDL SQL Server

Chú ý:

Chúng ta cần xem xét kỹ khi sử dụng thông tin hệ thống. Đặc biệt khi cập nhật dữ liệu vào thông tin hệ thống

# Xử lý lỗi trong thủ tục thường trú

# Tìm hiểu các thủ tục thường trú của hệ thống

# Xử lý lỗi trong thủ tục thường trú và các thủ tục hệ thống

- Khái niệm cơ bản về Lỗi (Error)
- Kiểm soát và thay đổi nội dung lỗi
- Lỗi trong SQL server
- Một số Thủ tục hệ thống

# Khái niệm cơ bản về Lỗi

- Lỗi phát sinh khi bảng, Biến hay tham số không chấp nhận giá trị truyền vào. Một trong những yếu tố không hợp lệ xảy ra đều khiến SQL server phát sinh ra lỗi.
- Lỗi được đưa ra màn hình có nội dung đã được SQL server định nghĩa trước.
- Tuy nhiên, nếu cần chúng ta có thể thay đổi nội dung thông báo sao cho phù hợp với yêu cầu của người sử dụng

# Xử lý lỗi

- Trong khi lập trình trên SQL server, sẽ có lúc chúng ta cần kiểm soát lỗi do các câu lệnh SQL gây ra.
- Một số lỗi do sai kiể dữ liệu, một số lỗi do phân quyền truy cập không hợp lệ…
- Khi phát sinh lỗi, nếu cần chúng ta có thể bắt lỗi và kiểm soát chúng.

# Xử lý lỗi

- ShipperID là cột không cho phép nhập dữ liệu
- Do vậy khi thêm một bản ghi mới và gán dữ liệu cho cột ShipperID thì không thêm được ➔ Sảy ra lỗi
  - Declare @Error int
  - Insert into Shippers(shipperID) Values(1)
  - Select @Error = @@ERROR
  - print 'Gia tri bien loi la :' + convert(varchar,@Error)
  - print 'Gia tri ham loi la :' + convert(varchar,@@Error)

# Sử dụng @ERROR trong thủ tục thường trú

```
Create procedure sp_XulyLoi
AS
Begin
 Declare @Error int
 Insert into Shippers(shipperID) Values(1)
 set @Error = @@ERROR
 if @Error !=0
  Begin
   if @Error=544
    Begin
      print 'Khong them ban ghi moi duoc'
      print 'Cot ShipperID khong thay doi duoc.'
    End
   Else
      print 'Khong phan biet duoc loi'
      print 'Lien lac voi Adminnistrator'
      print 'Loi co so:' + convert(varchar,@Error)
  End
End
```

# Sử dụng @ERROR trong thủ tục thường trú

Thủ tục trên sẽ in ra thông tin khi có lỗi:

In ra thông tin trước ELSE khi lỗi là lúc thêm mới vào bản ghi trong trường hợp xác định là cập nhật dữ liệu của cột ShipperID

In ra thông tin sau ELSE khi lỗi nhưng chưa xác định lỗi là do nguyên nhân tư đâu.

# Kiểm soát lỗi khi sảy ra

Đây là thủ tục lấy mã lỗi thi thêm mới một bản ghi vào cơ sở dữ liệu

```
Create procedure sp_XulyLoiTraVe
AS
Begin
  Declare @Error int
  Insert into Shippers(shipperID) Values(1)
  set @Error = @@ERROR
  Return @Error
End
```

# Kiểm soát lỗi khi sảy ra

```
Create procedure sp_InNoiDungLoi
@vError int
AS
Begin
 if @vError !=0
  Begin
   if @vError=544
    Begin
      print 'Khong them ban ghi moi duoc'
      print 'Cot ShipperID khong thay doi duoc.'
    End
   Else
    Begin
      print 'Khong phan biet duoc loi'
      print 'Lien lac voi Adminnistrator'
      print 'Loi co so:' + convert(varchar,@vError)
    End
 End
End
```

# Kiểm soát lỗi khi sảy ra

Thủ tục trên sẽ chuyển mã lỗi thành ngôn ngữ của người sử dụng.

Giá trị truyền vào là mã lỗi.

Những thông tin được in ra tương ứng với những mã lỗi được truyền vào.

# Kết hợp hay thủ tục trên

Kết hợp hai thủ tục trên ta có kết quả như mong muốn: Thông báo lỗi theo ngôn ngữ của người lập trình (yêu cầu của thiết kế)

- Declare @Error int
- Exec @Error = sp_XulyLoiTraVe
- exec sp_InNoiDungLoi @Error

- Lấy giá trị của thủ tục thứ nhất ra ngoài.
- Truyền vào trong thủ tục 2

# Lệnh RaisError

Trong khi SQL server sử dụng lệnh RaisError để đưa thông báo lỗi ra màn hình.

Tuy nhiên ở mức độ nào đó chúng ta cũng có thể dùng lệnh này để phát sinh lỗi theo nội dung mong muốn.

# Cú pháp Lệnh RaisError

RAISERROR (

<ID thông báo | Nội dung thông báo>, <Chỉ dẫn phát sinh lỗi>, <trạng thái>

[,các tham số]

)

Trạng thái (state): nhằm chỉ rõ lỗi thuộc nhóm nào trong hệ thống của SQL server, trạng thái có giá trị từ: 1 đến 27

# ID thông báo, Nội dung (message ID, Message String)

ID thông báo là số ID của nội dung lỗi trong hệ thống lỗi của SQL Server.

Các thông báo lỗi của hệ thống chứa trong bảng sysmessages.

Chúng ta có thể thêm thông báo lỗi vào bảng hệ thống bằng cách sử dụng Stored Procedure

# Chỉ dẫn phát sinh lỗi (Severity)

- Là code chỉ dẫn phát sinh lỗi.
- Trong SQL Server nếu lỗi phát sinh do dữ liệu thì Severity có giá trị từ 1-18.
- Nếu lỗi xuất phát từ hệ thống thì severity có giá trị từ 20-25

# Ví dụ RAISERROR

```
Create procedure sp_XulyLoiRais
AS
Begin
 Declare @Error int
 Insert into Shippers(shipperID) Values(1)
 set @Error = @@ERROR
 if @Error !=0
  Begin
   Raiserror('Loi them du lieu',1,1)
  End
End
```

# Thêm thông báo lỗi vào danh sách lỗi

Hầu hết các lỗi sảy ra khi phát sinh trong quá trình làm việc trên CSDL SQL Server.

Trong một số trường hợp nào đó chúng ta cần có những thông báo lỗi như nội dung mình mong muốn

# Cú pháp thêm thông báo lỗi

sp_addmessage [ @msgnum = ] *msg_id* ,
[ @severity = ] *severity* ,
[ @msgtext = ] '*msg*'
[ , [ @lang = ] '*language*' ]
[ , [ @with_log = ] '*with_log*' ]
[ , [ @replace = ] '*replace*' ]

Một số tham số đã trình bày trên RAISERROR

# Thủ tục mở rộng
# Extended Stored Procedure

# Thủ tục mở rộng

xp_cmdshell: Thực hiện lệnh dưới dấu nhắc DOS từ SQL server

Cú pháp:

Xp_cmdshell <‘Câu lệnh’> [, no_output]

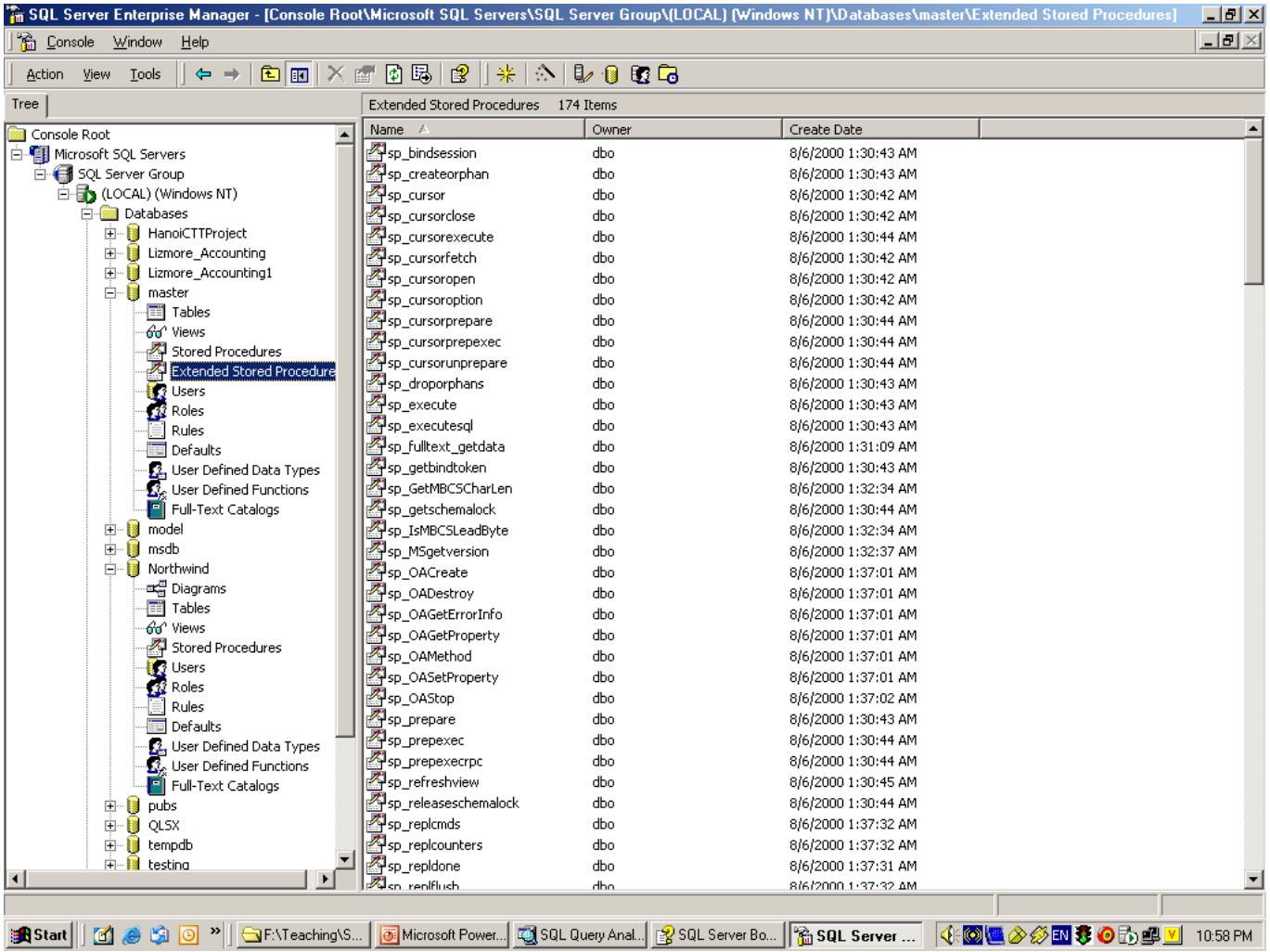
# Thủ tục mở rộng

Xp_msver: trả về thông tin nơi SQL server cài đặt.

Nói chung còn rất nhiều thủ tục mở rộng trong SQL server mà chúng ta chưa tìm hiểu.

Để tìm hiểu các thủ tục mở rộng chúng ta có thể vào Extended Stored Procedure trong Enterprise Manager.

# Các thủ tục khác của hệ thống



# 6 Kịch bản (script) bó (batchs) và OSQL

**Nguyễn Trọng Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com
**Home:** http://www.tronganh.com

# Tóm tắt

- Khái niệm cơ bản về kịch bản (script)
- Khai báo biến trong SQL Server
- Sử dụng hàm IDENTITY
- Sử dụng hàm ROWCOUNT
- Khái niệm cơ bản về OSQL

# Khái niệm về kịch bản (script)

Khi soạn thảo các câu lệnh SQL ở những phần trước, hầu hết những câu lệnh mà ta sử dụng được gọi là kịch bản SQL hay Script

Các câu lệnh Create, Alter, hay Select chúng ta thực hiện dưới mọi hình thức đều được gọi là kịch bản.

Bất kỳ một Script nào được tạo ra đều có mục đích rõ ràng.

# Kịch bản đơn giản

Về mặt kỹ thuật, kịch bản không phải là một kịch bản cho đến khi chúng được ghi thành 1 tập tin.

Khi lưu trữ thành tập tin, nếu cần chúng ta có thể sử dụng mà không cần viết lại toàn bộ các câu lệnh SQL từ đầu.

Lưu dưới dạng File Text

Khi tạo ra script chúng ta có thể sử dụng các hàm của hệ thống, các câu lệnh đã được đề cập trước

# Câu lệnh USE

Dùng để khai báo cơ sở dữ liệu hiện tại.

Chỉ ra đối tượng của CSDL có thể xử lý được.

Nếu không chỉ rõ cơ sở dữ liệu thì các câu lệnh có hiệu lực đối với CSDL hiện hành.

Nên sử dụng khi gọi CSDL khác với CSDL hiện tại đang dùng

# Khai báo biến trong SQL Server

Khi thực hiện các câu lệnh SQL trong bất kỳ ứng dụng nào chúng ta đều cần đến nhu cầu tính toán.

Khi tính toán thường ta cần khai báo biến hoặc sử dụng một số phép toán.

Khi khai báo nhiều biến, ta sử dụng dấu phẩy để ngăn cách các biến.

# Cú pháp khai báo biến

Declare @<tên biến> <kiểu dữ liệu> [,
@<tên biến> <kiểu dữ liệu> ] [,
@<tên biến> <kiểu dữ liệu> ]

Khai báo một, biến có giá trị ban đầu là NULL cho đến khi biến đó được gán giá trị.

Các ứng dụng trong quá trình tính toán, sử dụng phép tính hoặc hàm hệ thống, biến được khai báo để thực hiện mục đích này của người lập trình.

# Ví dụ khai báo biến

Declare @Amount  int

Declare @FullName varchar(20)

Declare @Address,email varchar(200)

Declare @phone,mobile varchar(200), @salary int

Nếu khai báo nhiều biến tắt, các biến đó có cùng kiểu giá trị.

Kết thúc một nhóm biến có cùng kiểu dữ liệu cách nhau dấu phẩy.

# Gán cho biến giá trị cụ thể

Gán với một giá trị trực tiếp

```
SET @Luong = 500
```

Gán giá biến từ một biến và giá trị hằng

```
SET @VATluong = 500 * 0.1
```

Gán giá trị đó  là kết quả của biểu thức SELECT

```
SET @tongluong = (Select sum(Luong) from
BangLuong)
If @Tongluong is NULL
  @Tongluong = 0
```

# Chú ý khi gán giá trị từ câu lệnh SELECT

Khi sử dụng phép gán biến với giá trị từ một biểu thức SELECT cần chú ý:

- Giá trị trả về có cùng kiểu dữ liệu của biến
- Nếu biến không phải là mảng, giá trị phải là duy nhất.
- Phép gán biến không cho phép sử dụng trong các câu lệnh SQL

# Một số hàm hệ thống thông dụng

@@DATEFIRST : trả về ngày đầu tiên trong tuần. Nếu thay đổi ngày hệ thống thì kết quả trả về theo hệ thống thay đổi theo.

@@FETCH_STATUS: Sử dụng chung với FETCH.  Trả về 0 nếu đọc mẩu tin hợp lệ, -1 con trỏ đang ở cuối bảng, -2 bản ghi đó đang bị xoá.

@@IDENTITY: Trả về giá trị nhận dạng cuối cùng của câu lệnh SELECT hay INSERT INTO

@@ROWCOUNT: trả về số bản ghi có ảnh hưởng đến câu lệnh SQL cuối cùng.

# Một số hàm hệ thống thông dụng

@@ERROR: trả về mã lỗi của câu lệnh SQL sau cùng của kết nối đang mở.

@@SERVERNAME: Trả về tên của máy chủ cục bộ mà kịch bản SQL đang thực thi.

@@TRANCOUNT: Trả về số lượng các nghiệp vụ SQL server với các kết nối hiện tại

# Sử dụng @@ROWCOUNT

Trong nhiều trường hợp SELECT, chúng ta muốn biết có bao nhiêu bản ghi trong bảng vừa truy vấn. Sử dụng biến để gán giá trị lấy được từ hàm ROWCOUNT vào biến cục bộ

```
select * from shippers where shipperID < 5
declare @mautin int
select @mautin = @@ROWCOUNT
print 'So mau tin :' + str(@mautin)
```

# BÓ - BATCH

# BÓ - Batch

Bó là một nhóm câu lệnh SQL được nhóm lại thành 1 đơn vị, tất cả câu lệnh SQL trong bó có thể thực hiện như một đoạn lệnh chương trình.

Những câu lệnh này có thể quan hệ với nhau và được thực hiện một cách liên tục

Nếu lỗi phát sinh, thì các câu lệnh đằng sau sẽ bị bỏ qua.

Trong trường hợp có nhiều bó thì cần dùng phát biểu GO để phân chia các bó

# BÓ - Batch

Mỗi bó được gửi đến Server và được thực hiện độc lập.

Khi một bó nào đó có lỗi. Các bó khác vẫn bình thực hiện độc lập

Biến được khai báo và có hiệu lực trong phạm vi của Bó

# Sử dụng Go, trường hợp lỗi

```
declare @loichao varchar(50)
select @loichao = 'Xin chao'
print 'Batch dau tien'
Go

print @loichao
print 'Batch thu 2'
Go

print 'Batch thu 3'
Go
```

# Chú ý

Go không phải là câu lệnh SQL.

Nếu sử dụng Go khi dùng các phương thức ODBC, hay ADO DB … hoặc bất kỳ phương thức truy cập cơ sở dữ liệu nào khác sẽ phát sinh lỗi từ SQL trả về

# OSQL

OSQL là mộ công cụ cho phép thực hiện câu lệnh SQL tại dấu nhắc cửa sổ DOS

osql -Usa -P -d northwind -Q "Select * from shippers"

# OSQL

Chúng cũng có thể chạy kịch bản trong file ở dấu nhắc của DOS.

Lưu nội dung các câu lệnh vào file.sql

Có nội dung:

```
select * from shippers
select * from shippers where shipperid < 5
```

Sau đó gọi lệnh sau:

```
osql -Usa -P -d northwind -i file.sql
```

# SQL động

# SQL động

Trong quá trình phát triển ứng dụng, đôi khi chúng ta cần thực hiện một phát biểu SQL không định nghĩa trước ➔ SQL động

Một câu lệnh SQL mà tên bảng từ bên ngoài hay tên bảng xuất phát từ phép gán nào đó thì chúng ta gọi là SQL động

# SQL động

Truyền tên bảng trong khi chạy các câu lệnh SQL

```
declare @tenbang varchar(50)
set @tenbang = 'shippers'
exec('select * from ' + @tenbang )
```

Ta có thể sử dụng EXEC hoặc EXECUTE để thực hiện câu lệnh SQL

# SQL động

Gán toàn bộ cả câu lệnh SQL trong khi chạy.

Các câu lệnh được xây dựng trong quá trình chạy

```
declare @caulenh varchar(250)
set @caulenh = 'select * From shippers'
set @caulenh = @caulenh + ' where shipperid < 5
'
exec(@caulenh)
```

8

# Hàm người dùng và hàm hệ thống

**Nguyễn Trọng Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com
**Home:** http://www.tronganh.com

# Tóm tắt

- Khái niệm về  hàm do người dùng định nghĩa
- Hàm người dùng trả về giá trị
- Hàm người dùng trả về bảng dữ liệu
- Tạo các hàm hệ thống
- Xoá các hàm do người dùng định nghĩa

# Khái niệm hàm người dùng (user define function)

Hầu hết các ngôn ngữ lập trình, hay cơ sở dữ liệu lớn, luôn luôn có một phần mở rộng cho phép người dùng tự định nghĩa một số quy tắc, hàm hoặc thủ tục.

User define function giống như stored procedure của SQL server. Hàm người dùng cũng có thể truyền tham số nhưng không được mang thuộc tính OUTPUT. Thay vào đó chúng ta dùng câu lệnh RETURN

# Khái niệm hàm người dùng (user define function)

Có hai loại hàm do người dùng định nghĩa:

- Hàm người dùng trả về giá trị
- Hàm người dùng trả về một bảng dữ liệu

# Cú pháp

```
CREATE  FUNCTION [ owner_name. ] ten_ham
    ( [ { @ten_thambien [AS] giatri_trave [ = default
] } [ ,...n ] ] )
RETURNS kieu_dulieu_trave
[ AS ]
BEGIN
    than_ham
    RETURN bieuthuc_trave
END
```

# Chú ý

- Hàm do người dùng định nghĩa không dùng giá trị với kiểu dũ liệu ntext, text, image, cursor, timestamp làm giá trị trả về.
- Có thể cung cấp thông tin về lỗi nếu phát sinh.
- Có thể sử dụng các hàm do người dùng định nghĩa trong các câu lệnh SQL như SELECT

# Ứng dụng hàm người dùng

Câu lệnh SQL thông thường

```
Select top 9
orderID,customerID,convert(varchar(20),OrderDat
e,104),
employeeID,requireddate,shippeddate from
orders
```

# Ứng dụng hàm người dùng

Hàm tự viết;

Chuyển ngày về đạng DDMMYYYY

```
create Function dbo.dngayDDMMYYY(@date
datetime)
returns varchar(20)
AS
Begin
  return convert(varchar(12),@Date,104)
End
```

# Ứng dụng hàm người dùng

Sử dụng hàm người dùng:

Viết lại câu lệnh truy vấn trên

Select top 9 orderID,customerID,dbo.dngayDDMMYYY(Order Date),

employeeID,dbo.dngayDDMMYYY(requireddate), shippeddate from orders

# Hàm người dùng trả về một bảng dữ liệu

Ngoài giá trị trả về là kiểu số nguyên, số thực, kiểu ký tự. Hàm người dùng còn cho phép giá trị trả về là một bảng dữ liệu.

Khi hàm người dùng có giá trị trả về, công việc này rất giống với VIEW.

Tuy nhiên VIEW không thể sử dụng giá trị từ bên ngoài truyền vào. Ngoại trừ chúng ta thay đổi các câu lệnh truy vấn trong VIEW.

# Cú pháp (giá trị trả về của hàm người dùng là bảng dữ liệu)

CREATE FUNCTION [ *owner_name*. ] *ten_ham*
( [ { @*ten_bien* [AS] *kieu_gia_tri_bien*[ = *default* ] } [ ,...*n* ] ] )

RETURNS TABLE

[ WITH < function_option > [ [,] ...*n* ] ]

[ AS ]

RETURN [ ( ] *cau_lenh_select*[ ) ]

# Ứng dụng hàm người dùng

Hàm do lấy dữ liệu trả về bảng

```
create function dbo.laydulieu()
returns Table
AS
 Return (Select top 9 orderID,customerID,
employeeID,requireddate,shippeddate from
orders)
```

# Ứng dụng hàm người dùng

Sau khi tạo thành công. Chúng ta có thể sử dụng hàm như sau:

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

Lúc này gọi hàm, kết quả trả về giông như VIEW, cách sử dụng cũng giống VIEW. Tuy nhiên thì View không truyền được tham số vào bên trong.

# Ứng dụng hàm người dùng trả về bảng dữ liệu (trường hợp tham số truyền vào)

```
Create function dbo.laydulieuTS(
@cusID varchar(20))
returns Table
AS
 Return (Select orderID,customerID,
employeeID,requireddate,shippeddate from
orders where customerID like '%' + @cusID)
```

# Ứng dụng hàm người dùng trả về bảng dữ liệu (trường hợp tham số truyền vào)

Sau khi tạo thành công. Chúng ta có thể sử dụng hàm như sau:

```
select * from dbo.laydulieuTS('A')
```

Lúc này gọi hàm, kết quả trả về giông như VIEW, cách sử dụng cũng giống VIEW tương ứng với giá trị của câu Truy vấn là A.

Mỗi một tham số truyền vào ứng với 1 View

# Xoá hàm người dùng

Khi tạo hàm người dùng, hàm đó sẽ thuộc về đối tượng hàm người dùng trong cơ sở dữ liệu.

Để xoá  hàm chỉ cần chọn tên hàm rồi delete ( trường hợp trong Enteprise Manager) Hoặc sử dụng DROP FUNCTION cùng tên hàm mà chúng ta muốn xoá

# Xoá hàm người dùng

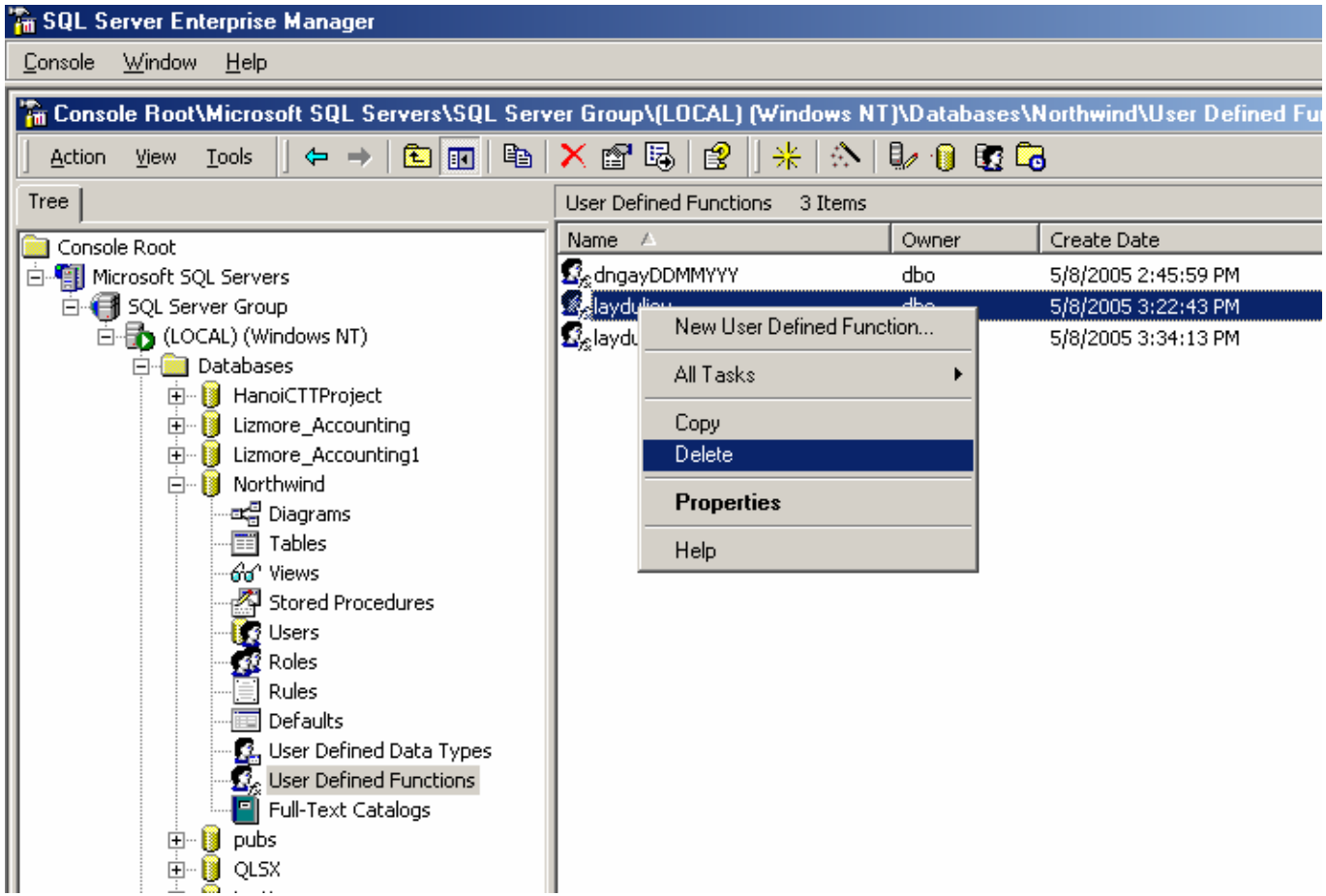
Cú pháp:

DROP FUNCTION ten_ham

Ví dụ: Xoá hàm laydulieu

DROP FUNCTION laydulieu

# Xoá hàm trong Enterprise Manager



# Tạo hàm hệ thống

Tất cả các hàm người dùng trên đều được tạo ra trong cơ sở dữ liệu hiện hành.

Nếu hàm có tính phổ biến cho việc sử dụng chung trong nhiều cơ sở dữ liệu chúng ta nên tạo hàm đó làm hàm hệ thống.

Khi là hàm hệ thống chúng ta có thê gọi như các hàm của SQL server thông thường. Như các hàm getdate(), convert()

# Tạo hàm hệ thống

Để tạo hàm hệ thống chúng ta nên tuân thủ 3 nguyên tắc sau:

- Tạo hàm trong cơ sở dữ liệu MASTER
- Dùng từ bắt đầu cho tên hàm là fn_
- Thay đổi chủ nhân của hàm bằng cách sử dụng thủ tục thường trú sp_changeobjectowner

# Tạo hàm hệ thống

Tạo hàm hệ thống tăng giá trị của biến truyền vào nên 1:

```
create function fn_tang1(
@bien int)
returns int
AS
Begin
 Set @bien = @bien + 1
 return @bien
END
```

# Tạo hàm hệ thống

Thay đổi chủ nhân của hàm vừa tạo

exec  sp_changeobjectowner 'fn_tang1','system_function_schema‘

Gọi hàm vừa tạo trong cơ sở dữ liệu NORTHWIND

select fn_tang1(shipperID), Phone from shippers

# Xoá một hàm hệ thống

Để xoá một hàm hệ thống chúng ta cần phải thông qua một thuộc tính cho phép trong cơ sở dữ liệu MASTER, đây chính là phần bảo mật của SQL server

Trước khi xoá một hàm hệ thống chúng ta phải sử dụng sp_configure thiết lập ‘allow update’ sang trạng thái 1, sau khi xoá xong cập nhật lại trạng thái 0

# Xoá một hàm hệ thống

```
-- thay đổi trạng thái ‘allow update’ sang 1
exec sp_configure 'allow update',1

-- cập nhật hệ thống
reconfigure with override

-- xoá hàm hệ thống
drop function system_function_schema.fn_tang1

exec sp_configure 'allow update',0
reconfigure with override
```

9

# Giao Dịch và Khoá

## Transactions - Lock

**Nguyễn Trọng Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com
**Home:** http://www.tronganh.com

# Tóm tắt

- Khái niệm cơ bản về giao dịch
- Các loại giao dịch trong SQL Server 2000
- Làm việc với SQL Server Log và Checkpoints
- Lock và Unlock

# Khái niệm cơ bản về giao dịch

Transactions hay còn gọi là giao dịch là một khái niệm giống đơn vị.

Xuất phát từ lập trường cơ sở dữ liệu bao gồm 1 hay nhiều nhóm nhỏ của các câu lệnh. Các nhóm dữ liệu sẽ được thực hiện toàn bộ hoặc không làm gì cả.

Các câu lệnh SELECT, DELETE, UPDATE đều có thể là một phần của giao dịch

# Các giao dịch

Tập các câu lệnh trong giao dịch có thể thực hiện hoặc không được xem như một câu lệnh duy nhất.

Giao dịch có các câu lệnh sau:

- BEGIN : bắt đầu một giao dịch
- COMMIT: xác định giao dịch hoàn thành
- ROLLBACK: quay ngược giao dịch
- SAVE: định nghĩa điểm đánh dấu cho phép quay ngược ROLLBACK chỉ một phần giao dịch

# Giao dịch BEGIN TRAN

Là giao dịch đơn giản nhất của quá trình xử lý giao dịch, chỉ ra rằng đây là điểm bắt đầu của một khối giao dịch.

Cú pháp:

Begin tran[saction]

[ten_giaodich | <@bien_giaodich>]

# Giao dịch COMMIT TRAN

COMMIT xác định kết thúc hay hoàn thành giao dịch. Tại thời điểm COMMIT được gọi, giao dịch được xem như là đã thực hiện thành công.

Cú pháp:

Commit tran[saction]

[ten_giaodich | <@bien_giaodich>]

# Giao dịch ROLLBACK TRAN

Khi gặp giao dịch này tất cả những phát biểu được thực hiện từ khi gặp giao dịch BEGIN sẽ bị huỷ bỏ. Ngoại trừ chúng ta định nghĩa thêm điểm dừng cho thao tác Rollback

Cú pháp:

ROLLBACK TRAN[SACTION]

[<transaction name | <save point name> |

<@transaction variable> | <@savepoint variable> ]

# Giao dịch SAVE TRAN

Để lưu lại vị trí của một giao dịch, chúng ta dùng kỹ thuật đánh dấu.

Chúng ta cũng có thể đánh dấu để khi tham chiếu ROLLBACK. Một giao dịch có thể có nhiều vị trí đánh dấu.

Cú pháp:

SAVE TRAN[SACTION]

[<save point name> | <@savepoint variable>]

# Ví dụ về Giao dịch

```
Begin Transaction MyTran
-- update
update shippers set companyName = 'DTT corp' where shipperID =
8
update shippers set companyName = 'HanoiCTT' where shipperID =
7
-- Kiem tra
select * from shippers

SAVE Transaction CNChanged
update shippers set companyName = 'VIDA' where shipperID = 6
-- Kiem tra
select * from shippers

Rollback Transaction CNChanged

Commit transaction
```

# Giải thích

Các câu lệnh thực hiện, giá trị cập nhật được giữ nguyên đối với 2 câu lệnh Update trước.

Câu lệnh Update thứ 3 không được cập nhật vì gặp Rollback. Do vậy nó bỏ qua các câu lệnh bắt đầu với SAVE TRANSACTION

# SQL SERVER LOCK

# Khoá (Lock)

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.

Khi làm việc trên cơ sở dữ liệu đa người dùng, xung đột giữa nhiều người sử dụng cùng thực hiện là thường xuyên sảy ra. Xử lý đụng độ hay tranh chấp trên đối tượng, chúng ta phải biết khi nào nên khoá (lock) khi nào không thể khoá, và những loại lock nào đang có.

# CÁC VẤN ĐỀ CÓ THỂ NGĂN NGỪA BẰNG LOCK

Lock hướng giải quyết 4 vấn đề sau:

- Dirty reads (đọc dữ liệu sai)
- Unrepeatable reads (đọc hai lần bản ghi)
- Phantoms (Đọc các bản ghi nháp, không có)
- Lost updates (cập nhật, mất dữ liệu)

# Đọc dữ liệu sai (Dirty reads)

Đọc dữ liệu sai sảy ra khi giao dịch đọc một bản ghi mà một phần giao dịch khác chưa hoàn thành.

Nếu giao dịch trước đó hoàn thành thì sẽ không sảy ra các vấn đề này, nhưng nếu giao dịch trước đó chưa hoàn thành hay đang thực hiện chế độ Rollback chúng ta sẽ phải đọc dữ liệu cũ, dữ liệu sai.

# Đọc bản ghi hai lần (Unrepeatable reads)

Khi đọc mẩu tin hai lần trong một giao dịch trong khi giao dịch khác chỉ thông báo về tình trạng dữ liệu trong một khoảng thời gian quy định

# Phantoms (đọc các bản ghi không có)

Nghĩa là chúng ta đọc được những bản ghi không có. Vì những bản ghi đó xuất hiện không bị tác động bởi các lệnh UPDATE hoặc DELETE.

Khác với hai vấn đề trên Phantoms là các vấn đề liên quan đến hệ điều hành, nó không yêu cầu bất kỳ sơ đồ nào để mô tả

# Lost Update (cập nhật mất DL)

Xảy ra khi một giao dịch cập nhật dữ liệu vào cơ sở dữ liệu thành công, nhưng lại ghi đè lên dữ liệu của giao dịch khác.

Sảy ra khi: 2 giao dịch đang đọc mẩu tin dữ liệu, sau đó giao dịch 1 ghi dữ liệu của bản ghi, giao dịch 2 cũng ghi kết quả chỉ có giao dịch 2 được cập nhật

# 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

# Chế độ Lock

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

# Chế độ Lock

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

10

# Bẫy Lỗi (Triggers)

**Nguyễn Trọng Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com
**Home:** http://www.tronganh.com

# Tóm tắt

- Khái niệm về  Trigger
- Sử dụng Trigger để ràng buộc trọn vẹn dữ liệu
- Sử dụng Trigger cho tham chiếu toàn vẹn động
- Tạo ra các quy tắc toàn vẹn dữ liệu động
- Sử dụng INSTEAD OF Trigger để tạo nên VIEWs

# Tóm tắt

- Những Trigger thông thường
- Kiểm soát Trigger
- Thực hiện Trigger

# Khái niệm Trigger

- Trigger là một dạng đặc biệt của thủ tục thường trú dùng để phản hồi một sự kiện cụ thể.
- Trigger là một đoạn mã được gắn vào bảng dữ liệu. Chúng được thực hiện khi có một sự kiện tương ứng với Trigger được gán cho sự kiện đấy
- Trigger không sử dụng hai đặc tính của thủ tục thường trú là tham số và giá trị trả về.
- Cần cân nhắc trước khi dùng Trigger

# Một số ứng dụng thường dùng

Ràng buộc toàn vẹn dữ liệu cho phù hợp với mô hình quan hệ cơ sở dữ liệu.

Kiểm soát dữ liệu hiện tại khi có thay đổi đến giá trị trong mẩu tin của bảng.

Kiểm tra dữ liệu nhập vào, phù hợp với mối liên hệ dữ liệu giữa các bảng với nhau.

Kiểm chứng khi xoá bản ghi trong bảng

# Phân loại chính

Dựa vào ứng dụng của Trigger trên một bảng dữ liệu, Trigger có ba loại như sau:

- INSERT Triggers
- UPDATE Triggers
- DELETE Triggers
- Tập hợp của ba loại trên

# Cú pháp

Cú pháp để tạo một Trigger giống như tạo thủ tục thường trú. Tuy nhiên, Trigger được tạo ra cho bảng dữ liệu cụ thể.

```
Create Trigger <Tên Trigger>
On <Tên bảng | Tên Views>
[WITH ENCRYPTION]
{
   { For | After }
     < [Delete] [, ] [Insert] [, ] [Update] > | INSTEAD OF
}
AS < Câu lệnh SQL>
```

# Chú ý

ON : chỉ ra rằng Trigger được viết cho bảng hoặc tên bảng ảo. Trigger với từ khoá AFTER không hỗ trợ VIEW.

With Encryption:

Giống như trong Thủ tục thường trú hoặc bảng ảo cho phép ngăn ngừa việc sửa đổi nội dung Trigger. Sử dụng ALTER Trigger thì with Encryption không hỗ trợ.

# Chú ý

FOR | AFTER

Mệnh đề FOR (AFTER) chỉ ra rằng Trigger sẽ áp dụng cho hành động nào trong ba hành động sau: INSERT, DELETE, UPDATE. Mệnh đề có dạng như sau:

- FOR INSERT
- FOR DELETE
- FOR UPDATE
- FOR INSERT, UPDATE,  DELETE

# Insert Trigger

Sử dụng mệnh đề FOR INSERT sẽ thực hiện khi có mẩu tin được thêm vào bảng. Với mỗi bản ghi được Insert, SQL sẽ tạo ra một bảng sao của bản ghi và lưu bảng sao của bản ghi này vào trong bảng mang tên INSERTED.

Bảng này chỉ tồn tại trong quá trình.

# DELETE TRIGGER

Giống như Trigger Insert, mỗi khi có bản ghi được xoá khỏi bảng thì Trigger này thực hiện việc kiểm tra dữ liệu.

Nếu thoả mãn các điều kiện thì bản ghi này được xoá khỏi bảng. Nếu không thì bản ghi trả lại giá trị bình thường, hành động Delete được huỷ bỏ.

SQL server cũng tạo ra một bảng sao các bản ghi bị xoá đưa vào bảng DELETED.

# Update Trigger

Mỗi khi có một bản ghi nào đó được cập nhật, giá trị của những cột có liên quan trigger sẽ được kiểm tra trước khi cập nhật

# NOT FOR REPLICATION

Nếu thêm câu lệnh này vào trong Trigger, thì Trigger sẽ không được thực hiện trừ khi có liên quan đến kỹ thuật sao chép nhân bản

# Sử dụng Trigger để ràng buộc trọn vẹn

Trong trường hợp này Trigger có thể được thực hiện các chức năng sau:

- Quan hệ 1-1
- Kiểm tra tính duy nhất của lớp loại.
- Kiểm tra dữ liệu khi cần thiết với điều kiện ràng buộc dữ liệu

# Trigger for Insert

```
Create trigger trgIns
on shippers
for insert
AS
if not exists(Select 'True' From Inserted I where
i.Price > 2)
Begin
 raiserror('Khong them ban ghi moi voi Price <
2',16,1)
 rollback tran
END
```

# Trigger For Delete

```
Create Trigger trgDel
on Shippers
For Delete
AS
if not exists(Select 'True' From Deleted I where
i.Price > 2)
Begin
 raiserror('Khong Xoa ban ghi moi voi Price <
2',16,1)
 rollback tran
END
```

# Trigger for Update

```
Create Trigger trgUpdate
on Shippers
For update
AS
if not exists(Select 'True' From updated I where
i.Quantity > 2)
Begin
 raiserror('Khong Cap Nhat ban ghi moi voi Price
< 2',16,1)
 rollback tran
END
```

# Trigger để kiểm tra quy tắc ràng buộc

Ứng dụng của Trigger thông thường sử dụng để:

- Tham chiếu dữ liệu trong bảng rời rạc
- Kiểm tra sự khác nhau trước và sau khi cập nhật dữ liệu
- Kiểm tra lỗi

# Trigger với giá trị từ bảng khác

```
CREATE Trigger trgSelect
on TestView
For Insert, Update
AS
if not exists(Select 'True',T.Price From Inserted I,
Shippers T where T.Price > I.Price and I.TEstID =
T.ShipperID)
Begin
 raiserror('Kiem tra lai gia tri cua View',16,1)
 rollback tran
END
```

# SỬ DỤNG IF UPDATE

Trong Update trigger, chúng ta có thể giới hạn cột nào được phép cập nhật.

Để thực hiện công việc trên chúng ta sử dụng hai hàm:

- IF UPDATE

# IF UPDATE

Chỉ có hiệu lực trong Trigger.

Sử dụng hàm này với mục đích chia ra những đoạn mã cho mỗi giá trị cập nhật hoặc chỉ có thể cho phép với cột được kiểm tra bằng hàm UPDATE

# Trigger for Update có sử dụng hàm Update()

```
Create trigger trgUTestView
on TestView
for update
As
 IF Update(Price)
  Begin
    If Exists(Select 'true' from Inserted I, shippers where
I.Price < shippers.ShipperID)
    Begin
     Raiserror('Khong cap nhat hop le',16,1)
     Rollback Tran
    End
 End
```

# Trigger for Update có sử dụng hàm Update()

Trigger trên được ứng dụng cho việc Update

Nếu tồn tại Price cập nhật nhỏ hơn ShipperID nhỏ nhất.

11

# CÁC GIAO DỊCH VÀ TRUY VẤN PHÂN TÁN

## (Distributed Queries Transactions)

**Nguyễn Trọng Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com
**Home:** http://www.tronganh.com

# Tóm tắt

- Khái niệm về  truy vấn phân tán
- Kết nối nhiều SQL Server
- Tạo cơ sở dữ liệu và các đối tượng trên Server

# KHÁI NIỆM

Khi xây dựng cơ sở dữ liệu cho ứng dụng, nếu cơ sở dữ liệu có số lượng bản ghi không vượt quá khả năng sử lý của Server, không cần phân tán.

Nếu quá lớn, chúng ta lên phân tán nhiều phần nhỏ. Từng phần CSDL liên quan với nhau sẽ được đặt trên Server khác nhau.

Chúng ta cần có giải pháp hợp lý cho các câu truy vấn phân tán và các giao dịch phân tán.

# GIAO DỊCH PHÂN TÁN

Các thứ tự BEGIN, ROLLBACK, COMMIT tương tự như trên một Server nhưng phải tạo nhiều kết nối và hoạt động phức tạp hơn.

Thực hiện các giao dịch phân tán gồm 2 giai đoạn:

- Chuẩn bị - Prepare
- Kết thúc – Commit

# GIAI ĐOẠN CHUẨN BỊ
## (Prepare phase)

Server nguồn gửi một yêu cầu là lệnh của giao dịch chuyển đến Server liên quan.

Tại thời điểm Server nhận yêu cầu, Server này phải thực hiện việc tuần tự cho đến khi kết thúc các giao dịch đã yêu cầu

# GIAI ĐOẠN KẾT THÚC
## (Commit phase)

Giả sử rằng tất cả những Server nhận yêu cầu từ Server nguồn đều thực hiện các giao dịch đó thành công.

Server nguồn sẽ gửi một thông tin như dấu hiệu đi trước và tiếp theo là giao dịch kết thúc Commit

# SO SÁNH HAI LOẠI GIAO DỊCH

| Lệnh SQL Server | Lệnh Distributed |
|---|---|
| BEGIN TRAN | BEGIN DISTRIBUTED TRAN |
| SAVE TRAN | Không hỗ trợ |
| ROLLBACK TRAN | ROLLBACK TRAN |
| COMMIT TRAN | COMMIT TRAN |

**Chú ý:**
**Các giao dịch phân tán không phải là giao dịch ngầm định**
**Giao thức xác nhận hai pha phải được thực thi trên SQL Server**
**Server nhận yêu cầu phải hỗ trợ DTC**

# TRUY VẤN PHÂN TÁN

Tạo một liên kết đến 1 Server khác, nghĩa là chỉ thị cho Server hiện tại biết rằng kết nối đến Server khác trên mạng, tạo ra một số chuẩn mực nhất định để thực hiện việc trao đổi hai hay nhiều Server khác nhau.

- Cung cấp tên Server, thông tin liên kết cần kết nối.
- Cung cấp thông tin đăng nhập cần kết nối.

# Sp_addlinkedserver

[@server = ] ‘server’
[,[@srvproduct=] ‘<product_name]’]
[,[@provider=] ‘<provider_name]’]
[,[@datasrc=] ‘<data_source path]’]
[,[@location=] ‘<location]’]
[,[@provstr=] ‘<connect string]’]
[,[@catalog=] ‘<Database]’]

# Sp_addlinkedserver

Sp_addlinkedserver
@server = '192.168.0.1',
@srvproduct = 'SQLServer OLEDB Provider',
@provider='SQLOLEDDB',
@datasrc= 'northwin‘

Trước khi muốn thao tác chúng ta phải tạo kết nối với Server mà chúng ta muốn thao tác

# KIỂM TRA CÁC KẾT NỐI VỚI SERVER

Để kiểm tra xem những server nào chúng ta kết nối thành công với các server khác.

Exec sp_linkedservers

# XOÁ KẾT NỐI SERVER

Để xoá kết nối Server ta gọi thực thi của thủ tục thường trú với tên SQL server chỉ định.

Sp_dropserver <tên server>

Ví dụ:

exec sp_dropserver '192.168.0.1

# ĐĂNG NHẬP SERVER ĐƯỢC KẾT NỐI

Sau khi kết nối đến Server khác phải thực hiện đăng nhập để thao tác trên cơ sở dữ liệu

exec sp_addlinkedsrvlogin
@rmtsrvname = ‘<tên server>',
@rmtuser = ‘<userseft>',
@rmtpassword = ''

# ĐĂNG NHẬP SERVER ĐƯỢC KẾT NỐI

@rmtsrvname : tên server cần truy vấn giống như tên trong phát biểu sp_addlinkedserver

@rmtuser = tên user để login vào server

@rmtpassword = mã đăng nhập

Chú ý;

Các thông tin đăng nhập giống như các thông tin khi đăng nhập tại SQL analyzer.

# ĐĂNG NHẬP SERVER ĐƯỢC KẾT NỐI

Ví dụ:

```
exec sp_addlinkedsrvlogin
@rmtsrvname = '192.168.1.34',
@rmtuser = 'sa',
@rmtpassword = ''
```

# TRUY VẤN DỮ LIỆU (Select)

Sau khi kết nối và đăng nhập vào CSDL mà chúng ta truy vấn dữ liệu như trên máy hiện tại mà chúng ta đang làm việc.

select * from

pc05.northwind.dbo.shippers

select * from

<Tên máy chủ>.<tên cơ sở dữ liệu>.<tên owner>.<tên bảng>

# TRUY VẤN DỮ LIỆU (Update)

Kiểm tra dữ liệu trước khi cập nhật:

```
select * from
pc05.northwind.dbo.shippers

update pc05.Northwind.dbo.shippers
set phone = '0912' where shipperID = 5
```

# OPENQUERY

Để sau khi kết nối và đăng nhập vào cơ sở dữ liệu của Server khác thành công, Ngoài việc thực hiện các phát biểu như đã trình bày ở trên.

Chúng ta có thể truy cập dữ liệu bằng cách sử dụng hàm OPENQUERY

# OPENQUERY

```
select Phone from
OpenQuery(TenServer,'select phone
from Northwind.dbo.shippers')
```

Tên server có thể là tên khác máy hiện tại hoặc máy khác (trong trường hợp các máy có kết nối với nhau và có quyền truy cập với nhau)

12

# TÌM HIỂU BCP và DTS

## (Bulk Copy Program and Data Trasaction Services)

**Nguyễn Trọng Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com
**Home:** http://www.tronganh.com

# BCP và DTS

SQL server hỗ trợ hai công cụ giúp di chuyển những khối dữ liệu lớn hiệu quả và nhanh chóng

Trong phần này ta làm việc với BCP. Sau đó chúng ta sẽ làm việc với DTS ở phần tiếp theo

# Các vấn đề chính của DTS

- Tiện ích BCP
- BCP import
- BCP Export

# TIỆN ÍCH BCP

BCP được sử dụng để chuyển dữ liệu text và dữ liệu theo khuôn dạng của SQL Server qua lại giữa các bảng với nhau.

Phân biệt chữ hoa và chữ thường đối với các tham số trong câu lệnh BCP

# CÚ PHÁP BCP

bcp {[[*tên CSDL*.][*owner*].]{*tên bảng* | tên *view*} }
{in | out | queryout | format}
*data_file*
[-c] [-w] [-t] [-S *server_name*[*instance_name*]] [-U *login_id*] [-P *password*]

Đây là những tham số chính được sử dụng để thao tác với CSDL.

# CÁC THAM SỐ VÀ Ý NGHĨA

[-c] : Kiểu dữ liệu là kiểu ký tự

[-w]: Kiểu dữ liệu là kiểu UNICODE

[-t]: Text file

[-S *server_name*[*instance_name*]]

[-U *login_id*]: Người dùng truy cập vào trong CSDL

[-P *password*]: Mật khẩu đăng nhập

# Nhập dữ liệu (BCP - Import)

Dữ liệu phải đúng với cấu trúc của bảng về số cột.

Các giá trị truyền vào phải phù hợp với kiểu dữ liệu của từng cột.

Thao tác với dữ liệu phụ thuộc chính vào các tham số

In: Nhập dữ liệu từ datafile

Out: Xuất dữ liệu từ table hoặc view vào datafile

Queryout: kết xuất vào file đích sử dụng câu truy vấn để lấy dữ liệu

Format: tạo định dạng file

# Xuất dữ liệu ra text file (Export)

Sử dụng dấu phẩy làm ngăn cách giữa các cột

bcp northwind.dbo.shippers in c:\ship.txt -c -t, -Usa –P

Kết quả: chuyển dữ liệu từ file ship.txt vào bảng shippers (chú ý về kiểu dữ liệu và số cột trong file text cho tương ứng với kiểu dữ liệu trong bảng)

# Nhập dữ liệu vào từ text file

bcp northwind.dbo.shippers out c:\ship.txt -c -t, -Usa –P

Kết quả:

Xuất dữ liệu ra text file, dữ liệu của mỗi cột được ngăn cách bằng dấu ‘,’ (trong trường hợp cột không có dữ liệu, dấu phẩy vẫn tồn tại)

# DTS
# (Data Transaformation Services)

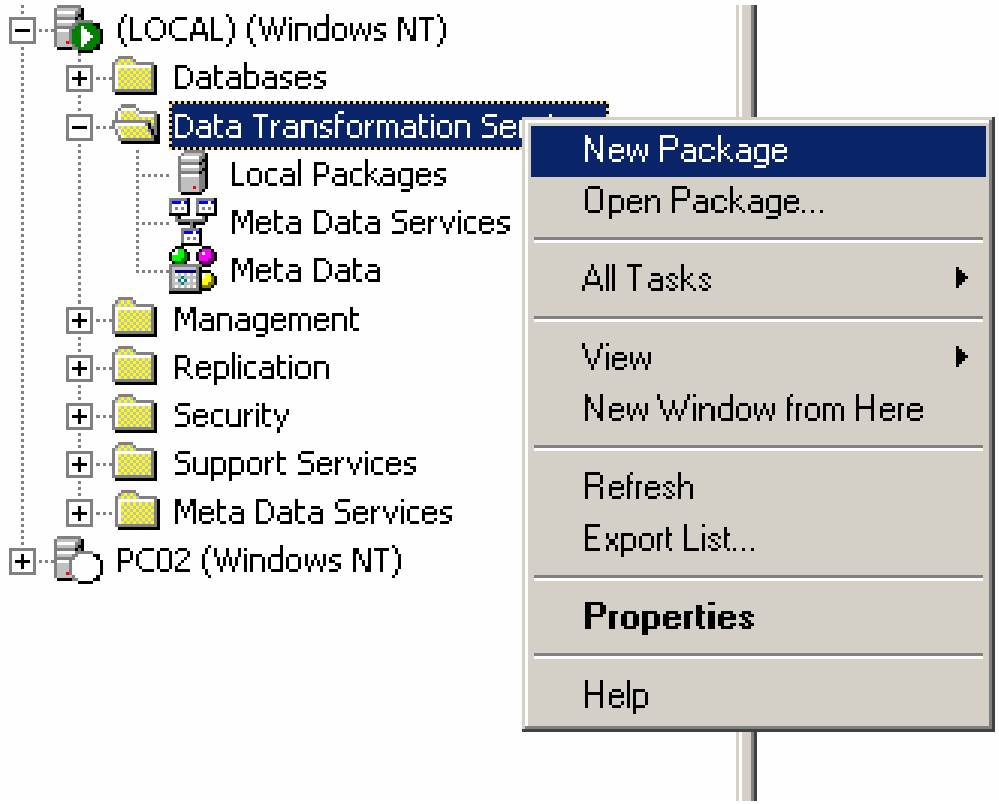
# Dịch vụ chuyển dịch dữ liệu

DTS được thiết kế ở mức tổng quát hoá với giao diện đồ hoạ thân thiện.

Chúng ta vừa có thể chuyển dịch dữ liệu từ nơi này đến nơi khác lại vừa có thể lập trình được trên nó

# DTS PACKAGE EDITOR

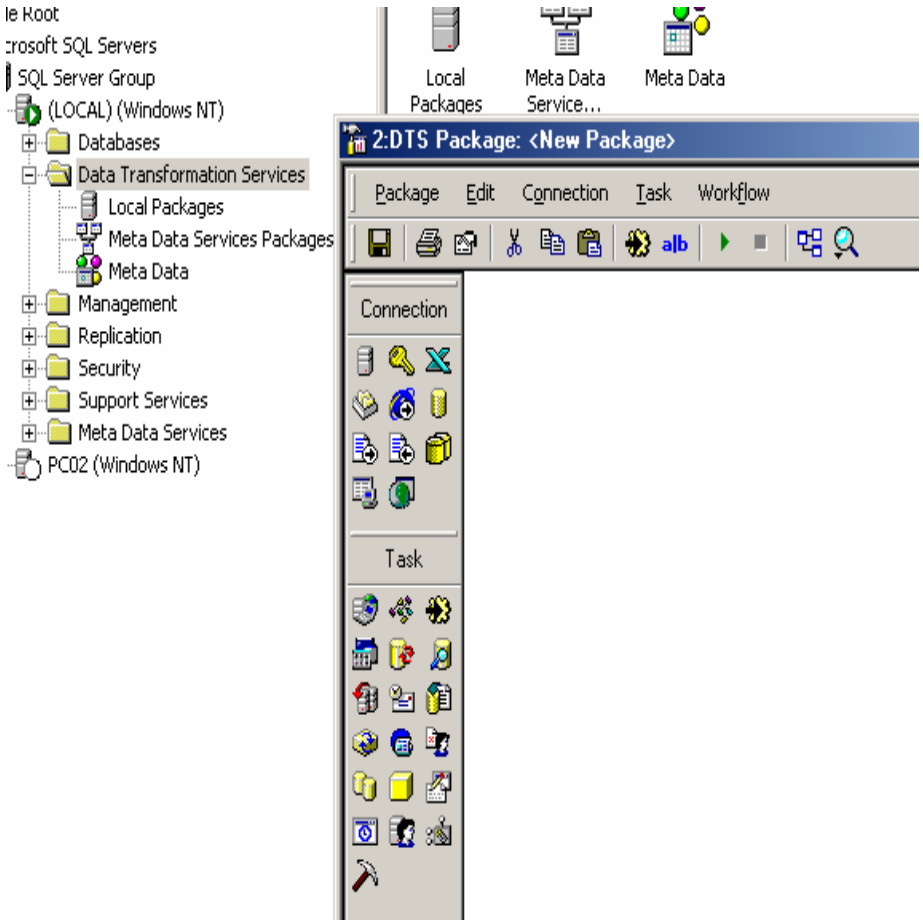
**Chọn mục như hình vẽ Chúng ta sẽ chuyển đến PACKAGE EDITOR trong Enterprise Manager**



# DTS PACKAGE EDITOR

Sau khi chọn New package chúng ta sẽ có hai thanh công cụ chính:

- Connection
- Task



# Connection


Một connection là một đường ống truyền dữ liệu kết nối dữ liệu nguồn và dữ liệu đích.

Có tất cả 11 lựa chọn connection đi cùng với SQL server.

| | | |
|---|---|---|
| SQL server | Access | Excel |
| dbase5 | Paradox | Text file (data out ) |
| Text (data in) | Oracle | OLEDB |
| OLEDB For ODBC | HTML file | |

# Connection

Thông thường chúng ta sử dụng SQL Server như là dữ liệu nguồn hoặc dữ liệu đích cho connection.

Tuy nhiên, DTS có thể cho chúng ta di chuyển dữ liệu ở những nguồn và đích khác nhau không nhất thiết phải là SQL Server, ví dụ như:

DB2 sang Oracle

# TÁC VỤ (Task)

Là một đơn vị công việc mà bạn muốn DTS thực hiện. Có rất nhiều dạng tác vụ được xây dựng sẵn trong DTS.

Chúng ta cũng có thể tự tạo ra những task riêng cho mình.

Những task mà DTS cung cấp có thể đáp ứng hầu hết những công việc chuyển dữ liệu mà chúng ta cần.

# TÁC VỤ (Task)


FTP: cho phép tải về hoặc chuyển file lên server bằng giao thức FTP. Tiện dụng khi làm việc thường xuyên với các máy chủ ở xa.

ActiveX Script: cho phép sử dụng các ngôn ngữ kịch bản như Vbscript hay Jscript…

Transform Data (chuyển dịch dữ liệu): bao gồm các chức năng chuyển đổi và lưu trữ dữ liệu. No cho phép cả việc thay đổi khi chuyển dữ liệu.

# TÁC VỤ (Task)

Ngoài ra còn có các Task như

| | | |
|---|---|---|
| Execute Process | Execute SQL | Data Driven Query |
| Copy SQL Server Obj | Send mail | Bulk Insert |
| Execute package | Message Queu | Transfer Err Msg |
| Transfer Database | Analysis Services | Transfer Master Pro |
| Transfer Job and Logins | Dynamic Property | Dynamic mining Prediction |

# SỬ DỤNG IM/EX WIZARD

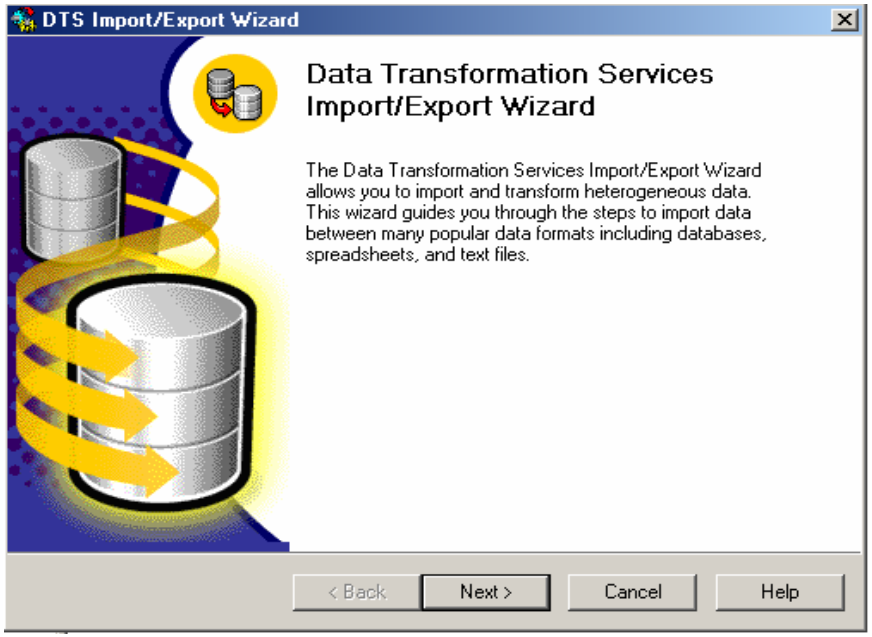
Công cụ này trong Enterprise Manager và được gọi theo nhiều cách khác nhau.

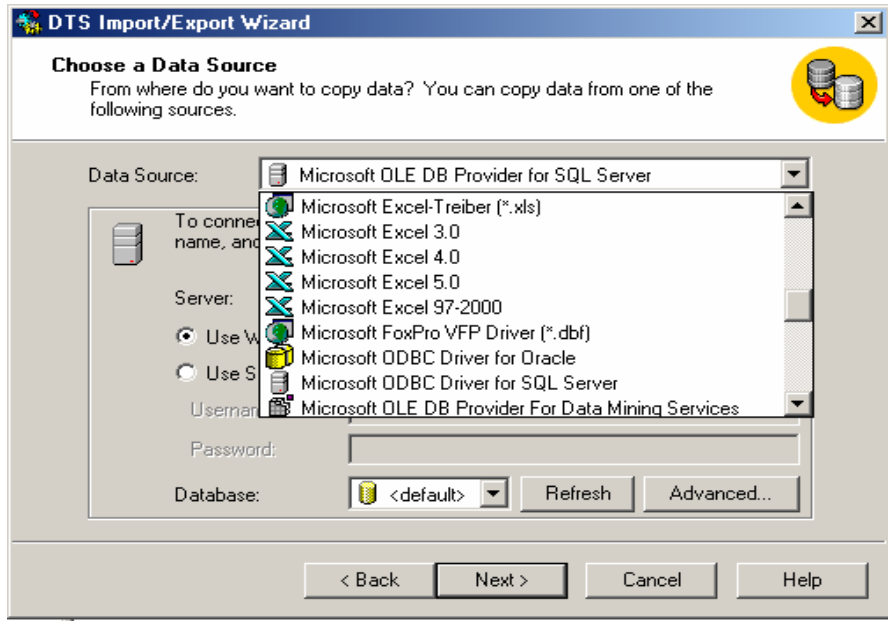
Mặc định nguồn dữ liệu là SQL Server nhưng có thể chọn từ danh sách Data source từ rất nhiều nguồn dữ liệu để kết nối khác.

Có nhiều nguồn cung cấp cho Connection. Mỗi nguồn dữ liệu có cách kết nối đặc trưng riêng.

# SỬ DỤNG IMPORT WIZARD

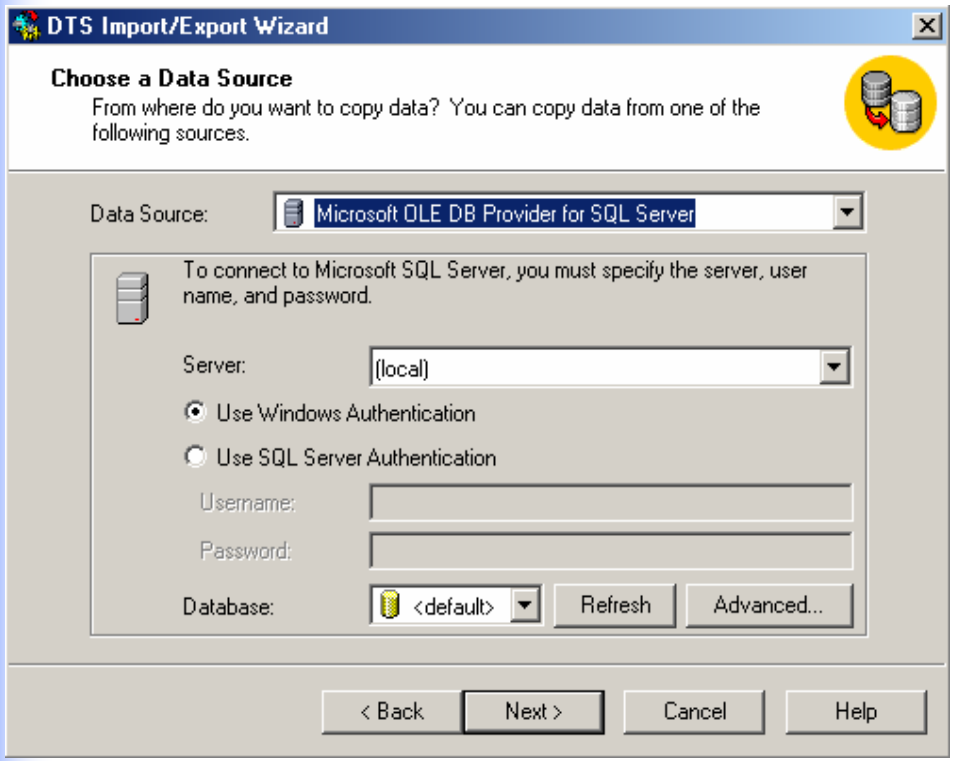
**Chọn CSDL ➔ Chọn All Task khi nhấp chuột phải tại database Cần Import ➔ Chọn Import Database**

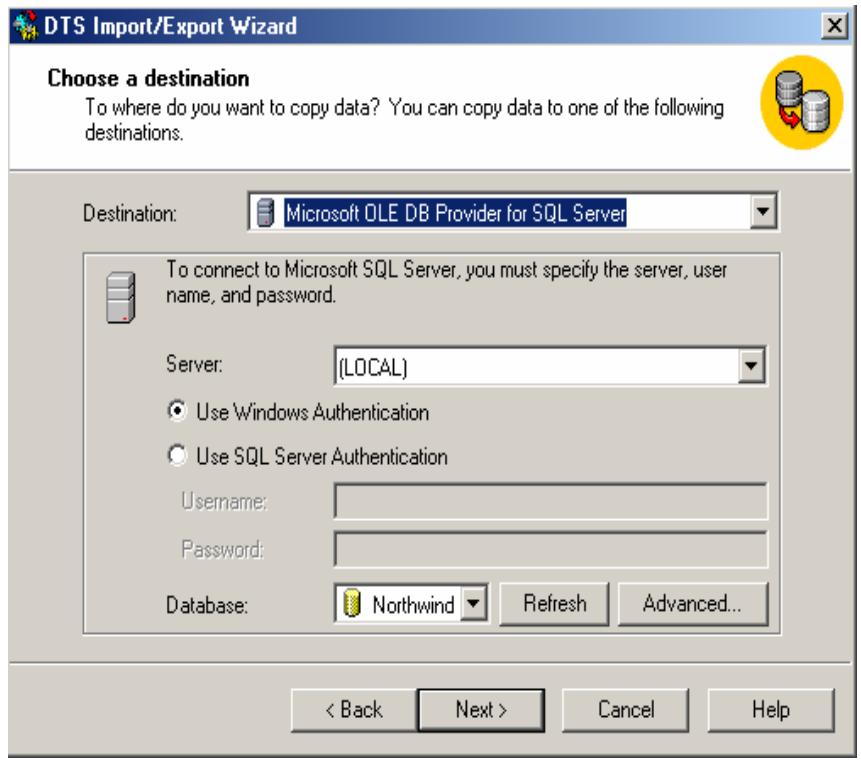


# SỬ DỤNG IMPORT WIZARD

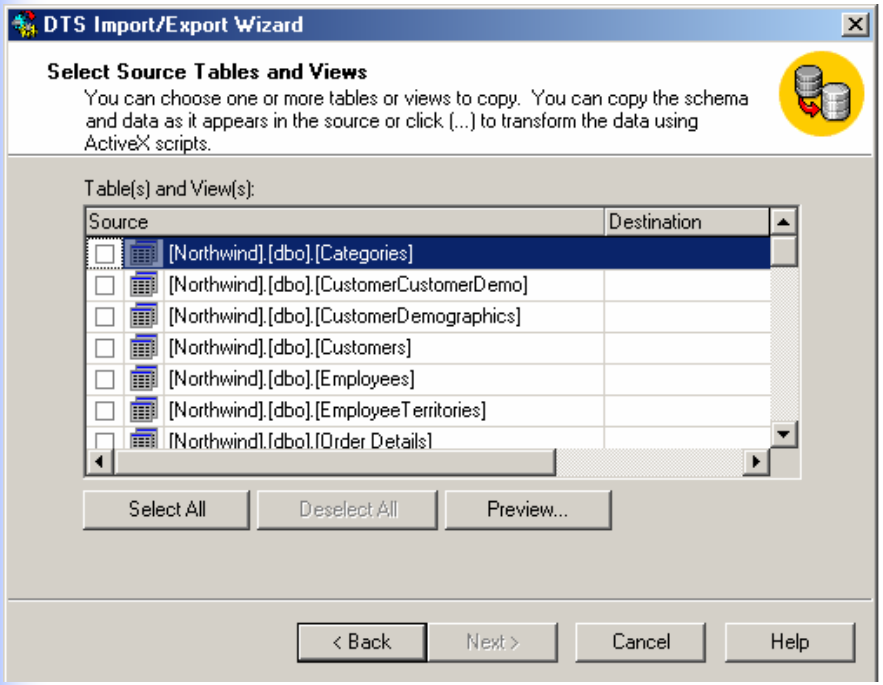
Chọn data source và destination.  Hai hộp thoại này rất giống nhau

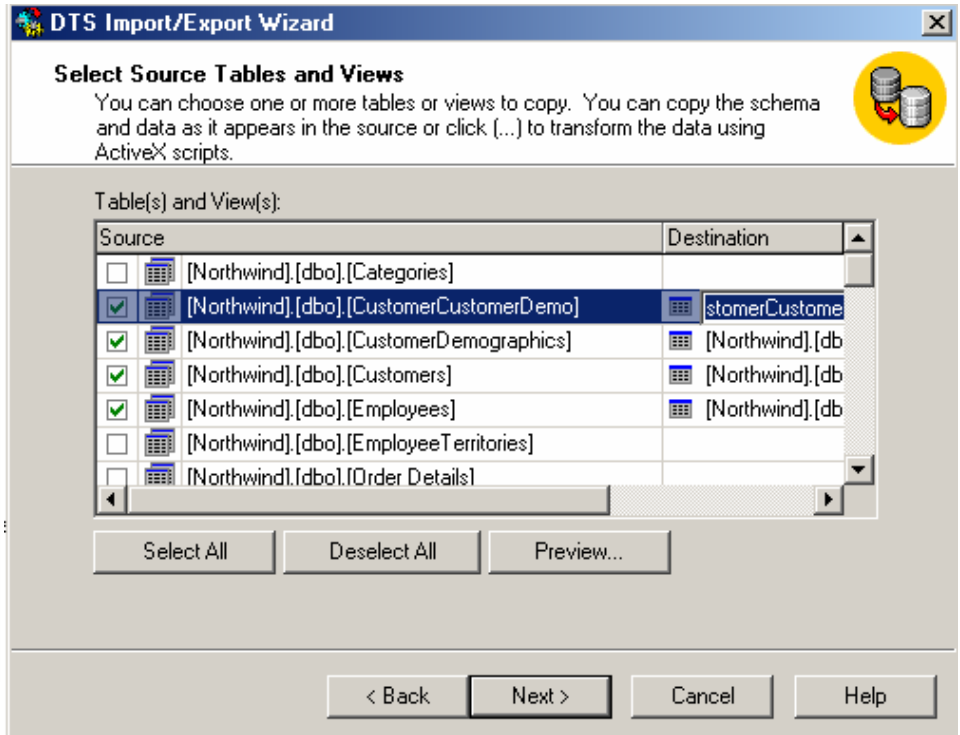


# SỬ DỤNG IMPORT WIZARD

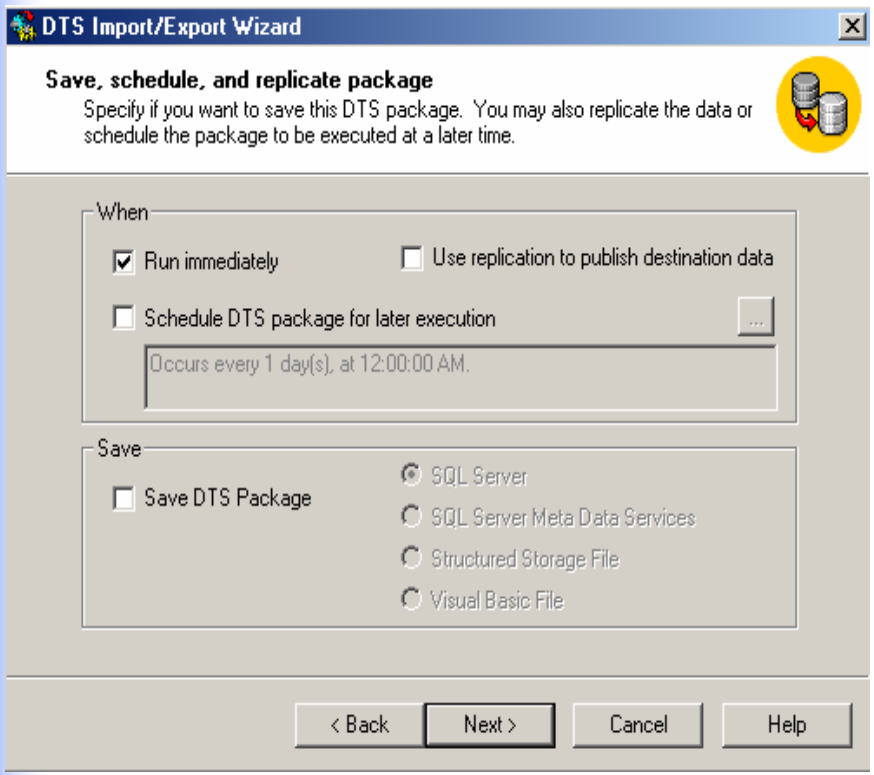
**Chọn bảng cần Import thì ta đánh dấu sau đó chọn Next**

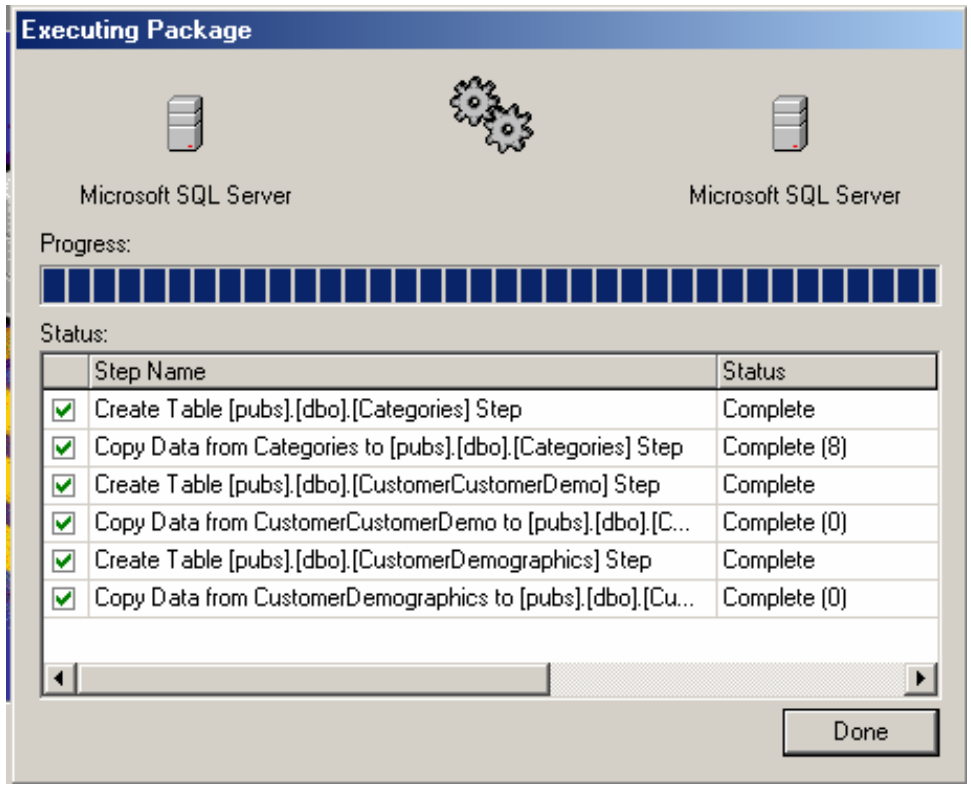


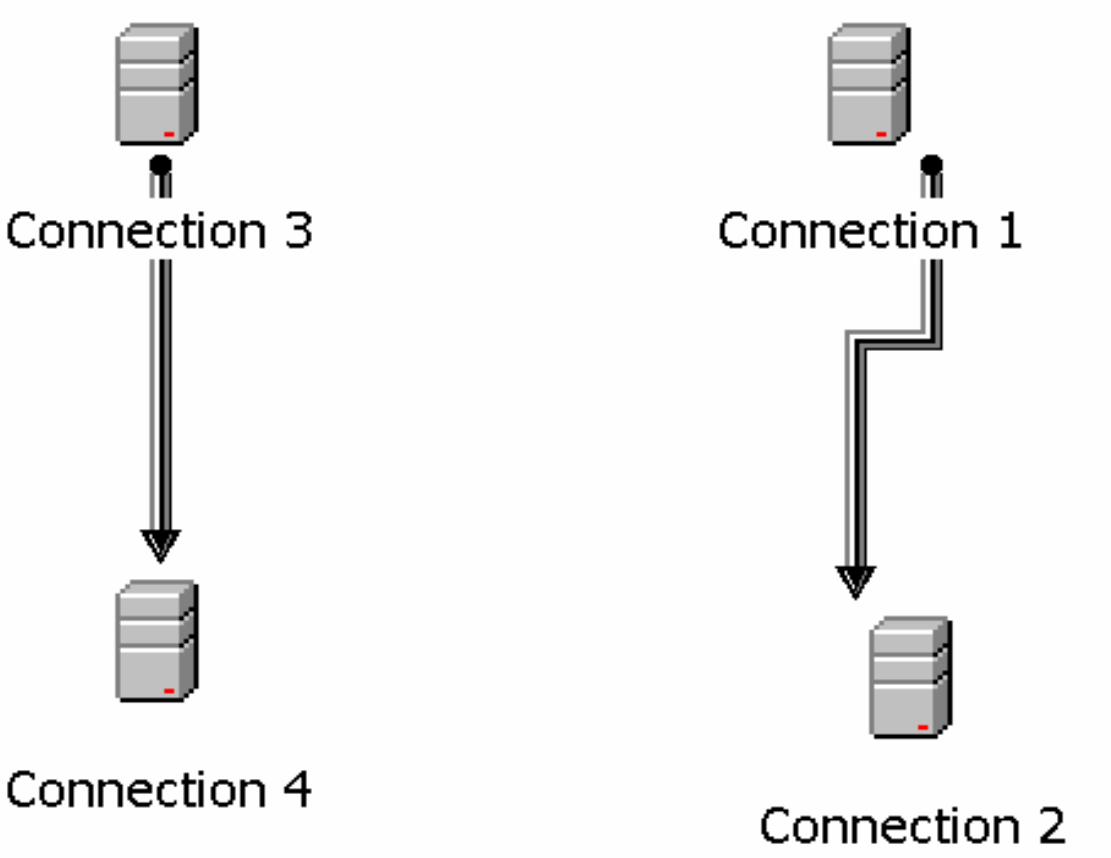
# SỬ DỤNG IMPORT WIZARD

**Sau khi thực hiện thành công,
dữ liệu được copy đến CSDL đích**



# Xem nội dung của Package



# Xem nội dung của Package

Nhấp chuột đúp vào local Package. Ta thấy Package Editor như hình vẽ trên.

Chúng ta sẽ thấy một số thao tác logic xử lý bằng giao diện đồ hoạ. Các thao tác xử lý được nhóm lại thành từng nhóm task

13

# KỸ THUẬT TÌM KIẾM

## (Full – Tex Search)

**Nguyễn Trọng Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com
**Home:** http://www.tronganh.com

# Full Text Search

- Khái niệm cơ bản về tìm kiếm
- Cấu trúc của Full – Text Search
- Cấu hình chỉ mục của Full – Text Search
- Tìm kiếm trong bảng dữ liệu
- Truy vấn dữ liệu băng Full – Text Search
- Các dạng tìm kiếm

# Khái niệm cơ bản tìm kiếm

Hầu như các ứng dụng với dữ liệu lớn luôn có các chức năng tìm kiếm theo một tiêu chuẩn nào đó của người sử dụng.

Sử dụng mệnh đề LIKE, với mệnh đề này cho phép tìm kiếm dữ liệu trùng khớp hay tương tự với hai ký tự “%” hay “_”, nếu chúng ta không cung cấp tiêu chuẩn thì SQL không thể tìm kiếm thoả mãn điều kiện

# CẤU TRÚC CỦA FULL TEXT SEARCH

FTS không phải là một phần của SQL Server.

FTS là một phần của Microsoft Index Server (vị trí của tập tin) và Site Server (kết hợp với vị trí của tập tin và trang URLs), được thiết lập trong dịch vụ MMSearch.

MMSearch là một ký thuật tối ưu trong quá trình tìm kiếm từ trong một văn bản hay chuỗi

# CẤU HÌNH CHỈ MỤC VÀ DANH MỤC CHO FT

SQL server không cho phép tạo chỉ mục hay danh sách chỉ mục, trừ khi cơ sở dữ liệu đó được thiết lập FULL TEXT SEARCH.

Cú pháp:

Exec sp_fulltext_database [@action =] ‘{enable | disable }’

Enable: cho phép tạo chỉ mục cho CSDL hiện hành

Disable: Huỷ bỏ hay không cho phép chỉ mục cho CSDL hiện hành

# TẠO VÀ HUỶ CHỈ MỤC CHO CSDL MẶC ĐỊNH

Tạo chỉ mục:

Exec sp_fulltext_database @action = ‘enable’

Huỷ bỏ chỉ mục:

Exec sp_fulltext_database @action = ‘disable’

Việc tạo chỉ mục và huỷ dựa trên một stored procedure nhưng xác định tham số.

# TẠO, HUỶ DANH MỤC FULL-TEXT

Sau khi cấu hình chỉ mục cho cơ sở dữ liệu hiện hành, chúng ta cần phải tạo danh mục chỉ mục để lưu trữ các chỉ mục của các bảng dữ liệu.

Cú pháp:

EXEC sp_fulltext_catalog [@ftcat=] ‘<tên catalog>’

[@action = ]

{‘create | drop | start_increamental | start_full’ | stop | rebuild}

# TẠO VÀ HUỶ DANH MỤC

Tạo chỉ mục cho danh mục hiện hành

exec sp_fulltext_catalog @ftcat = 'ftsaccount',@action ='Create‘

Xoá danh mục, chỉ mục cho cơ sở dữ liệu hiện hành:

exec sp_fulltext_catalog @ftcat = 'ftsaccount',@action =‘Drop‘

# CHO PHÉP BẢNG DỮ LIỆU SỬ DỤNG FULL – TEXT SEARCH

Sau khi thực hiện hai bước thành công, tiếp theo chúng ta chọn ra bảng dữ liệu thuộc nhóm dữ liệu cho phép tìm kiếm bằng kỹ thuật FTS

Cú pháp:

exec sp_fulltext_table 'Shippers', 'create', 'ftsaccount','PK_shippers‘

Sử dụng

Sp_helpindex để xem các index đã tồn tại

# TẠO CHỈ MỤC CHO CỘT

```
sp_fulltext_database @action='Enable'
sp_fulltext_catalog 'ftsaccount','create'
sp_fulltext_catalog 'ftsaccount','start_full'
select * from customers
sp_helpindex customers
exec sp_fulltext_table
'customers','create','ftsaccount','PK_customers'
exec sp_fulltext_column @tabname ='customers',
@colname= 'customerID',@action ='add'
```

# TRUY VẤN DỮ LIỆU VỚI FULL-TEXT SEARCH

Sau khi làm các thao tác trên. Chúng ta truy vấn dữ liệu giống như các điều kiện của biểu thức WHERE:

```
select CustomerID,companyName,contactName
from Customers
where contains(CustomerID,'"*B*"')
order by CustomerID
```

# 17

# BẢN SAO DỮ LIỆU

## (Replication)

**Nguyễn Trọng Anh**
**E-Mail**: anh@tronganh.com
tronganh@gmail.com
**Home:** http://www.tronganh.com

# BẢN SAO DỮ LIỆU

- Được sử dụng khi muốn lưu lại một khối lượng thông tin của cơ sở dữ liệu một cách tự động vào một cơ sở dữ liệu khác.

- Nội dung của bản sao chính là nội dung của bản chính nhưng phụ thuộc vào thời gian đồng bộ hoá dữ liệu của bản sao và bản chính.

# NHÀ XUẤT BẢN (publisher)

Dữ liệu nguồn được quản lý và xem như là một nhà xuất bản. Publisher sẽ định ra các mục dữ liệu để đưa đi phát hành/

Những dữ liệu này sẽ được nhà phân phối (Distributor) thu thập và gửi đi đến người sủ dụng có nhu cầu đăng ký sử dụng (Subcriber)

# NHÀ PHÂN PHỐI (Distributor)

Thu thập dữ liệu từ publisher

Có thể hoạt động trên của server của publisher hoặc có thể hoạt động trên server tách biệt nhau.

# CÀI ĐẶT CỤ THỂ

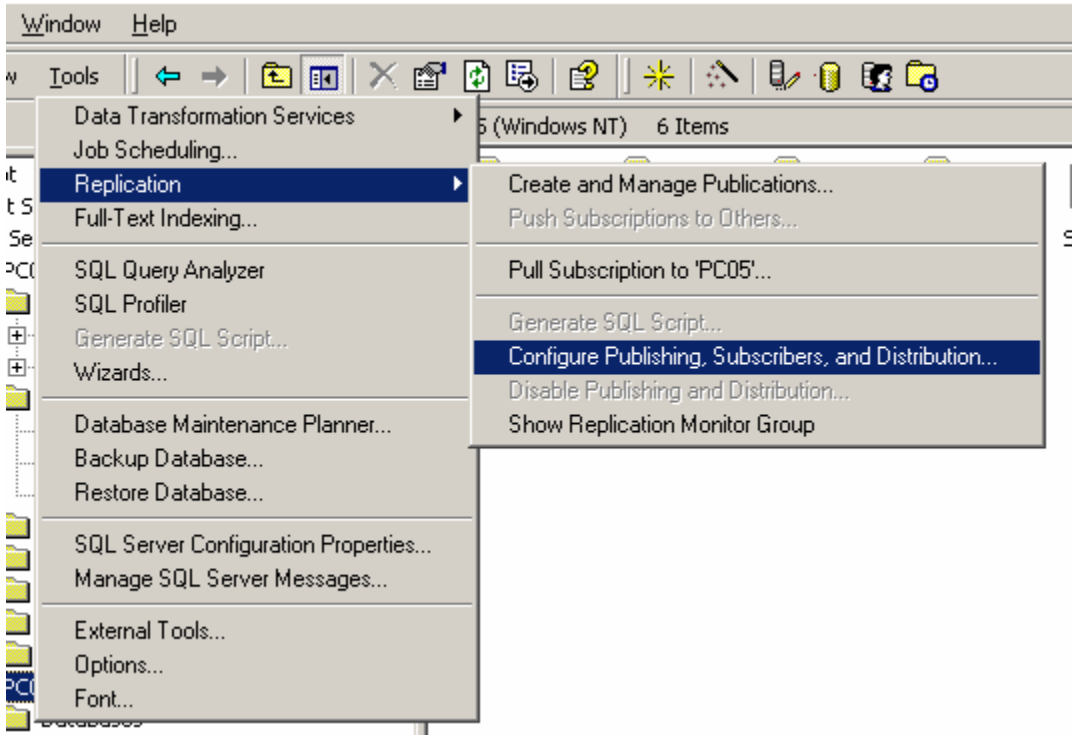
Cấu hình Distributor

Tạo và quản lý Publications

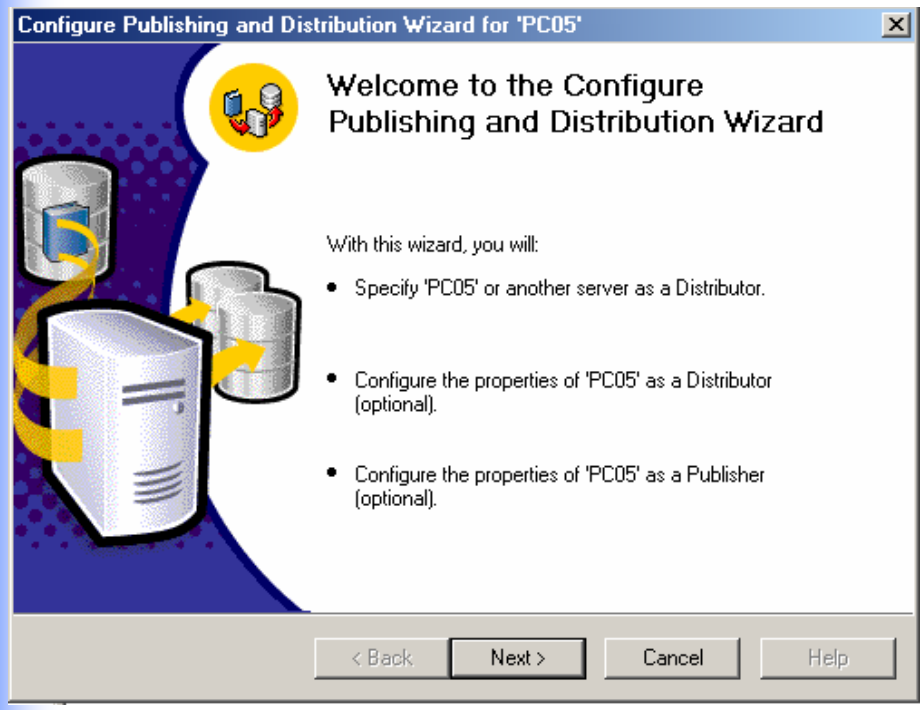
Tạo Push Subscription

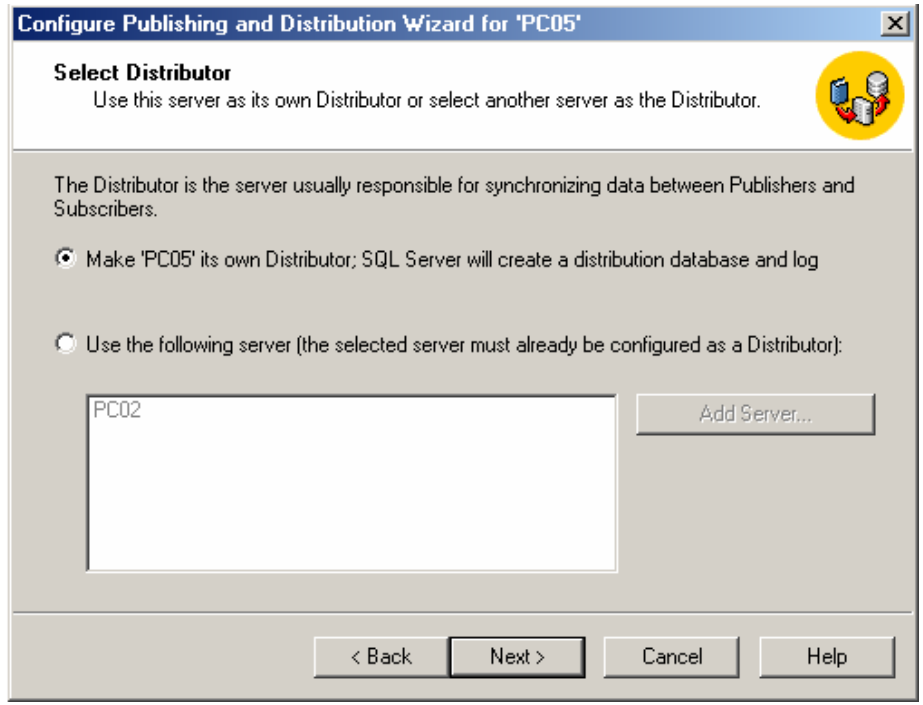
# CẤU HÌNH DISTRIBUTOR

Menu Tools == >
Replication == >
Config Publishing
and Distribution

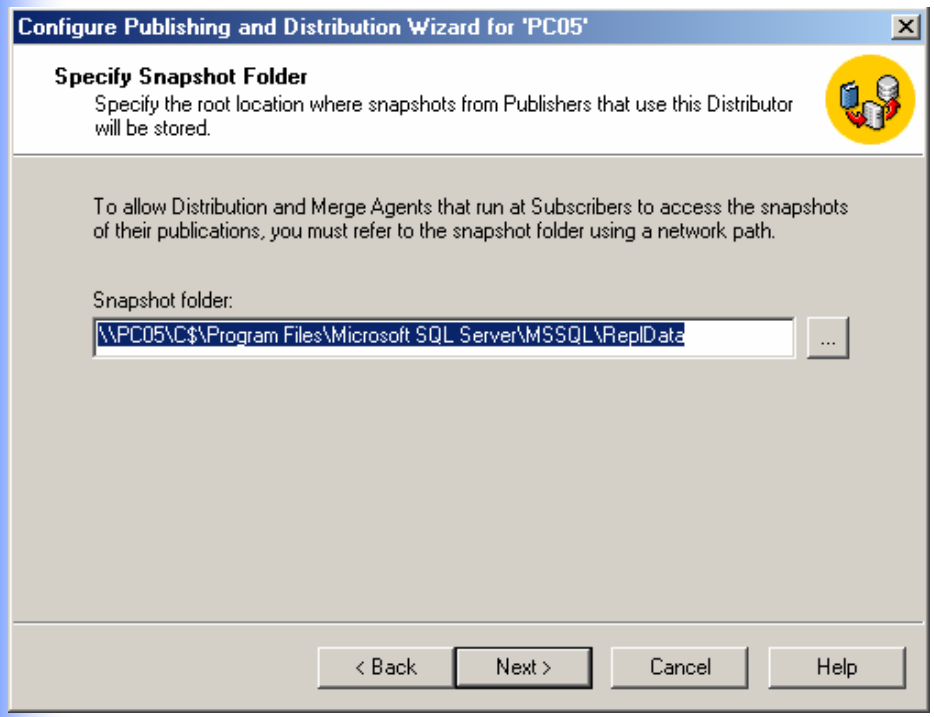


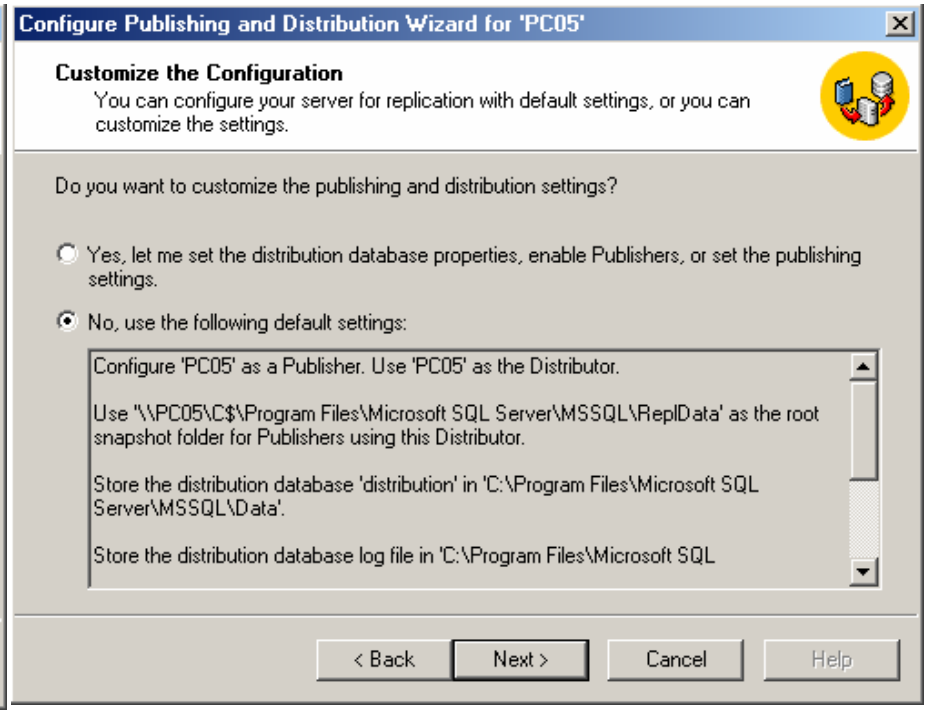
# CẤU HÌNH DISTRIBUTOR



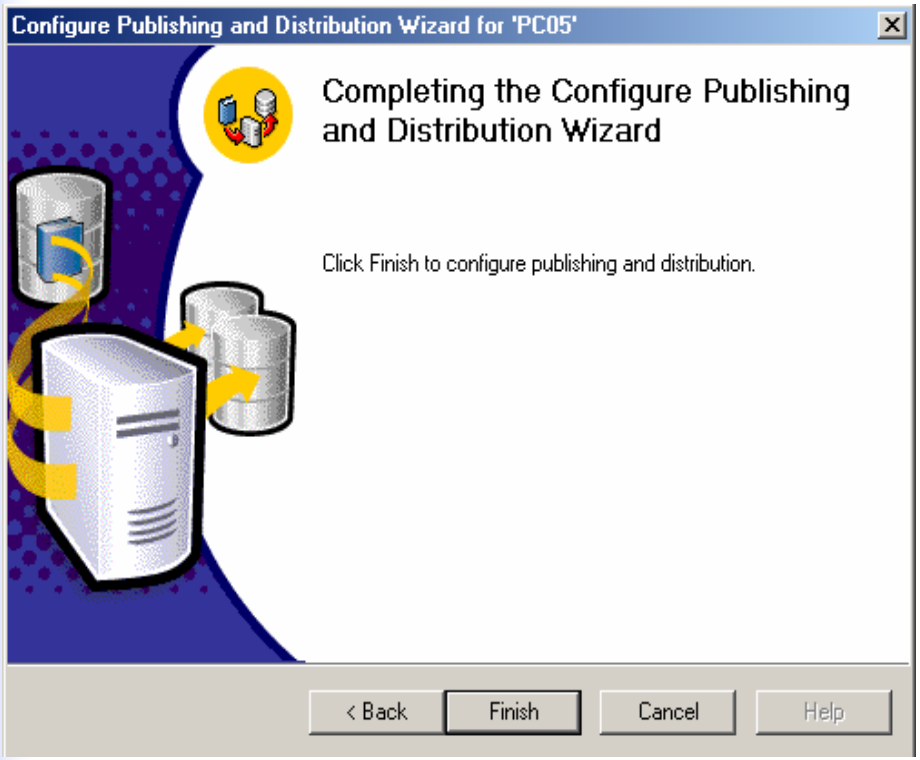



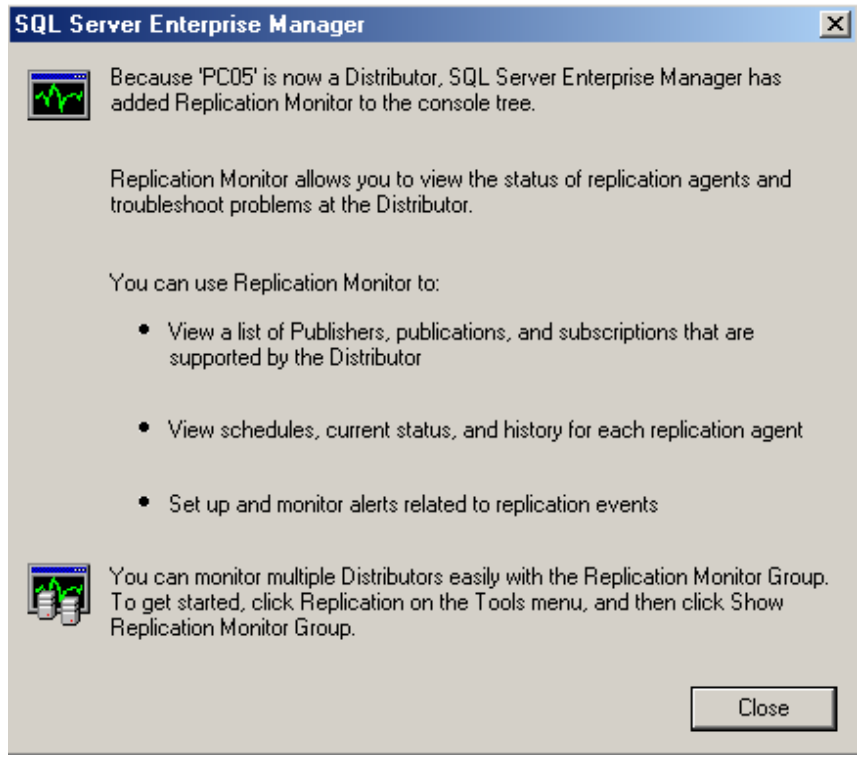
# CẤU HÌNH DISTRIBUTOR



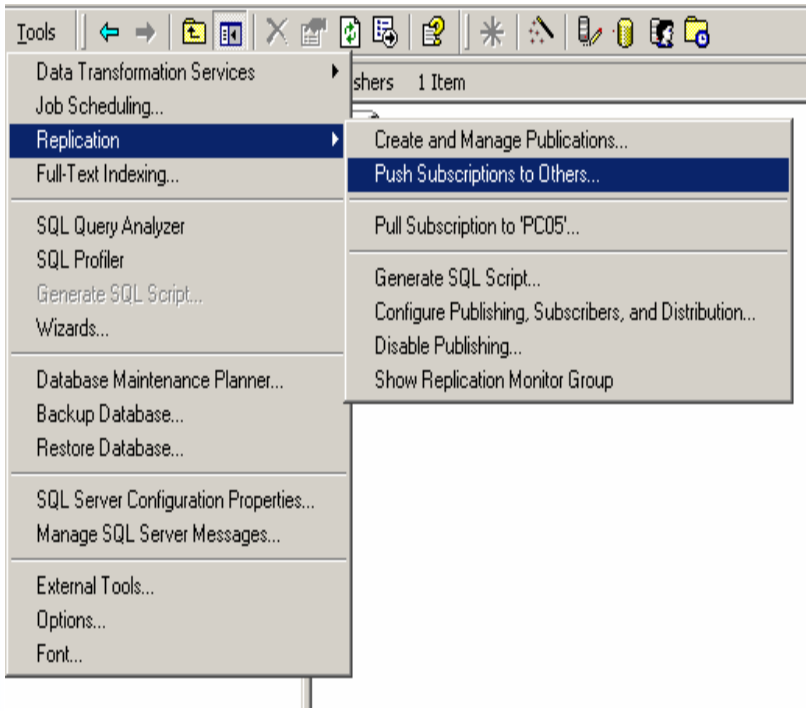



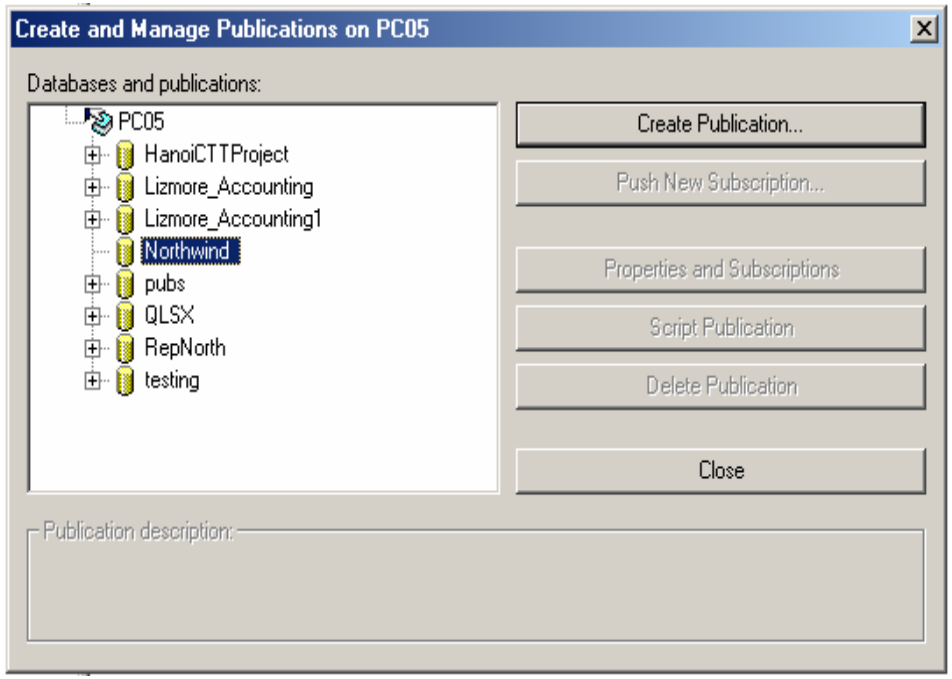
# CẤU HÌNH DISTRIBUTOR



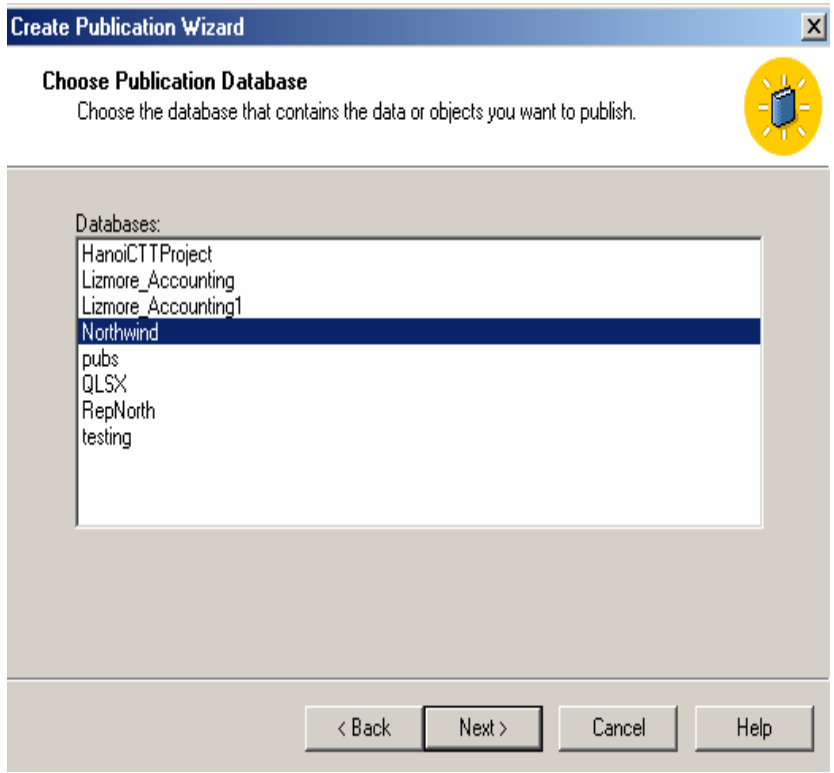



# Tạo và quản lý Publications

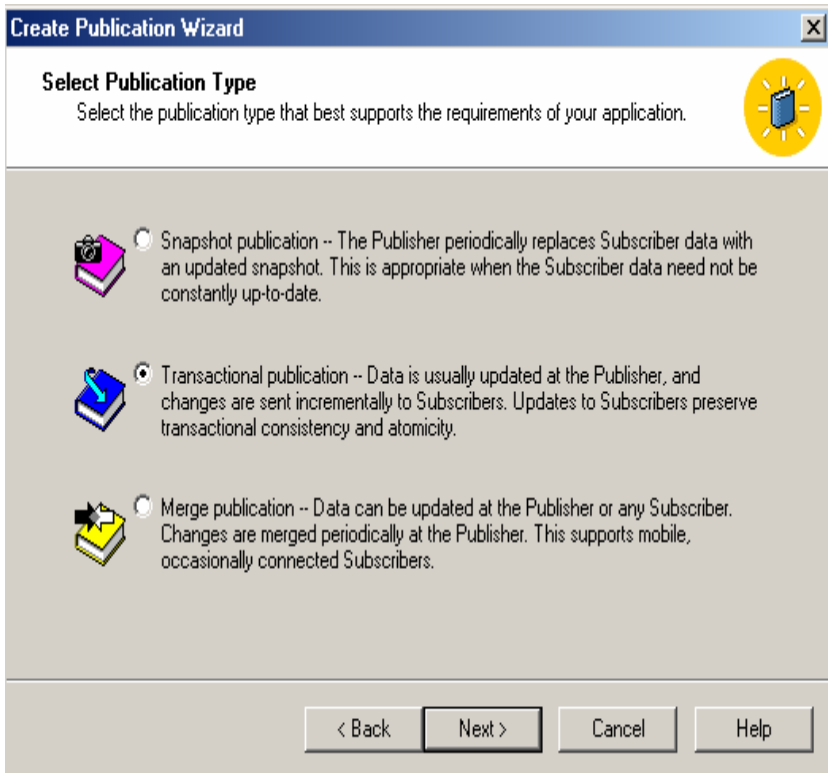


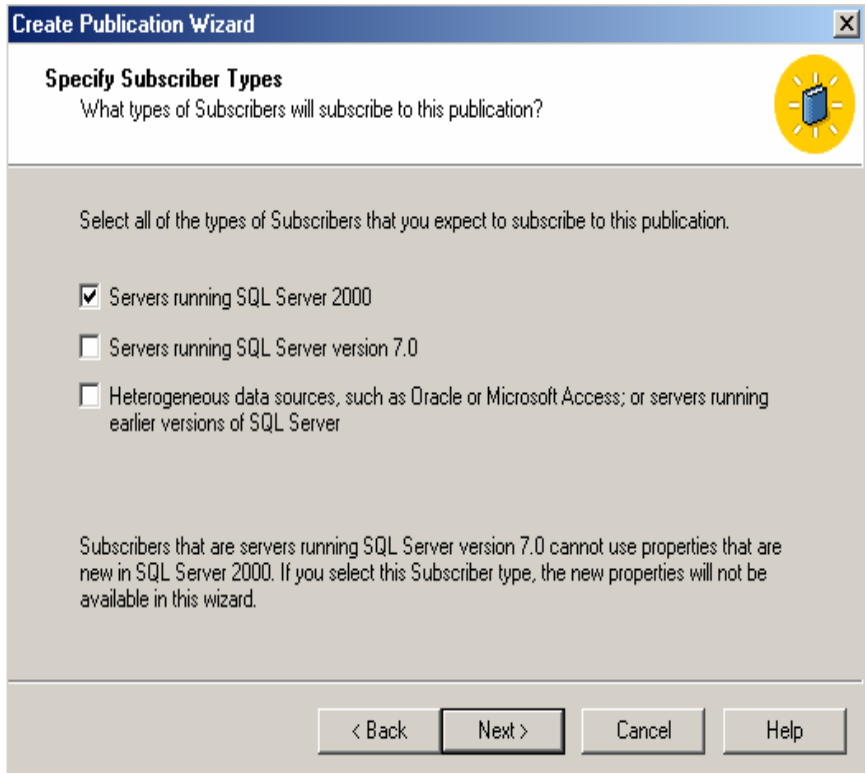



# Tạo và quản lý Publications

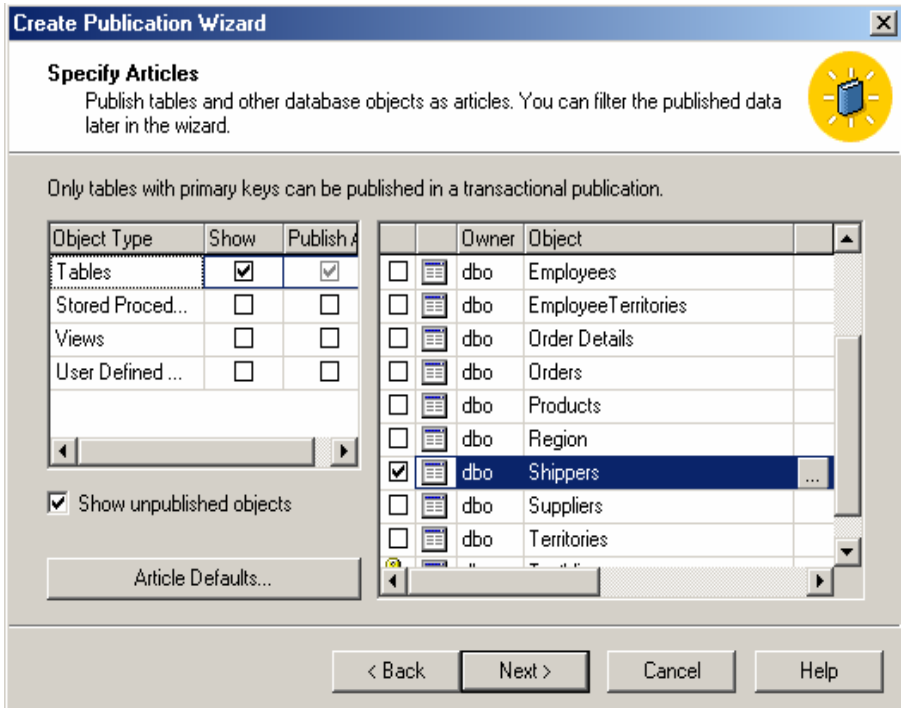
# Tạo và quản lý Publications

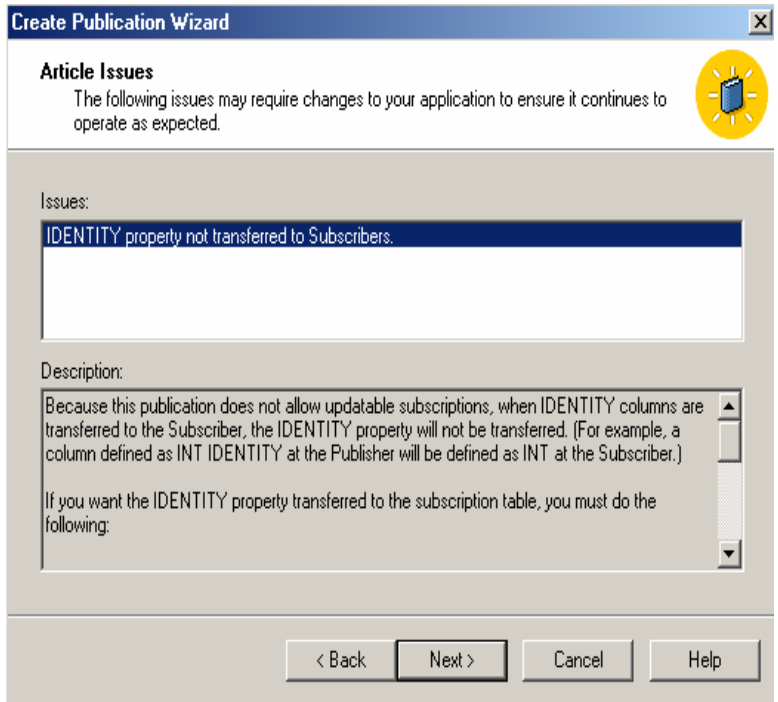


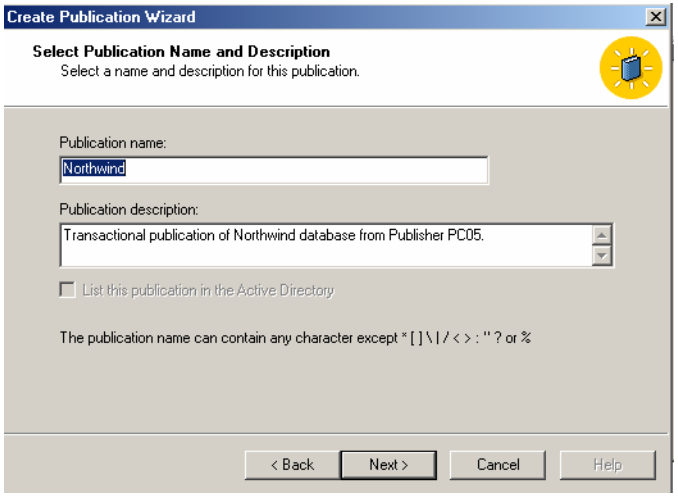
# Tạo và quản lý Publications

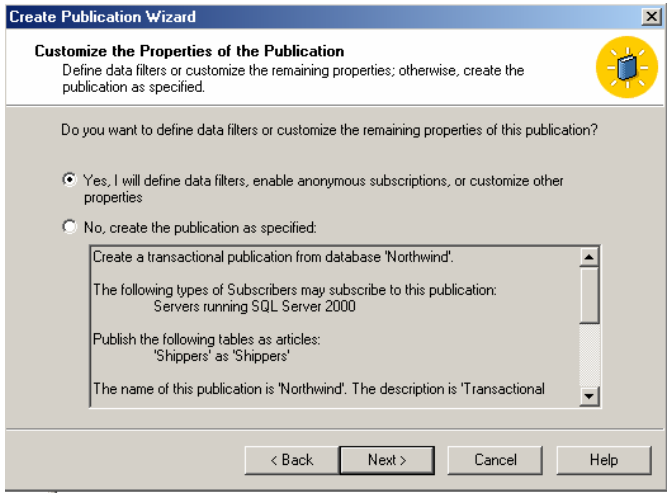
Chọn bảng cần copy (có thể là một hay nhiều bảng cần copy)



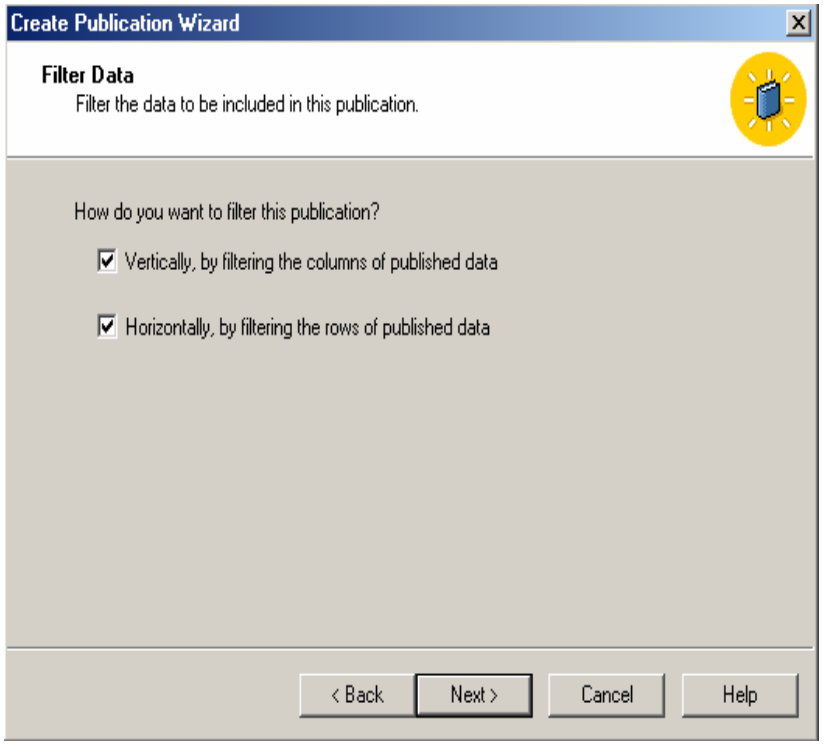



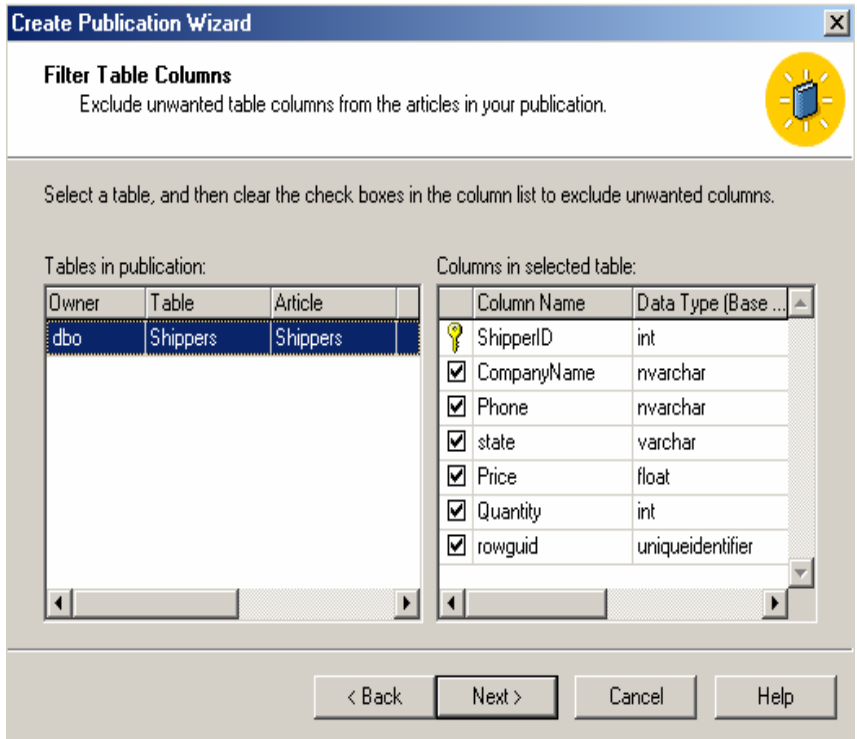
# Tạo và quản lý Publications



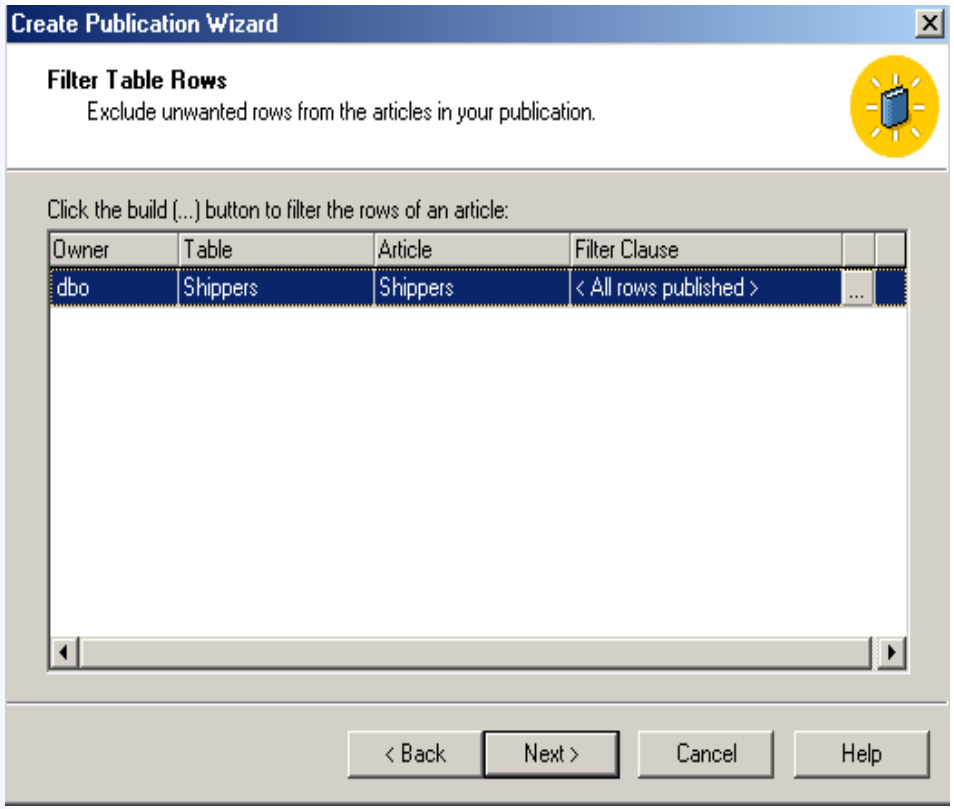



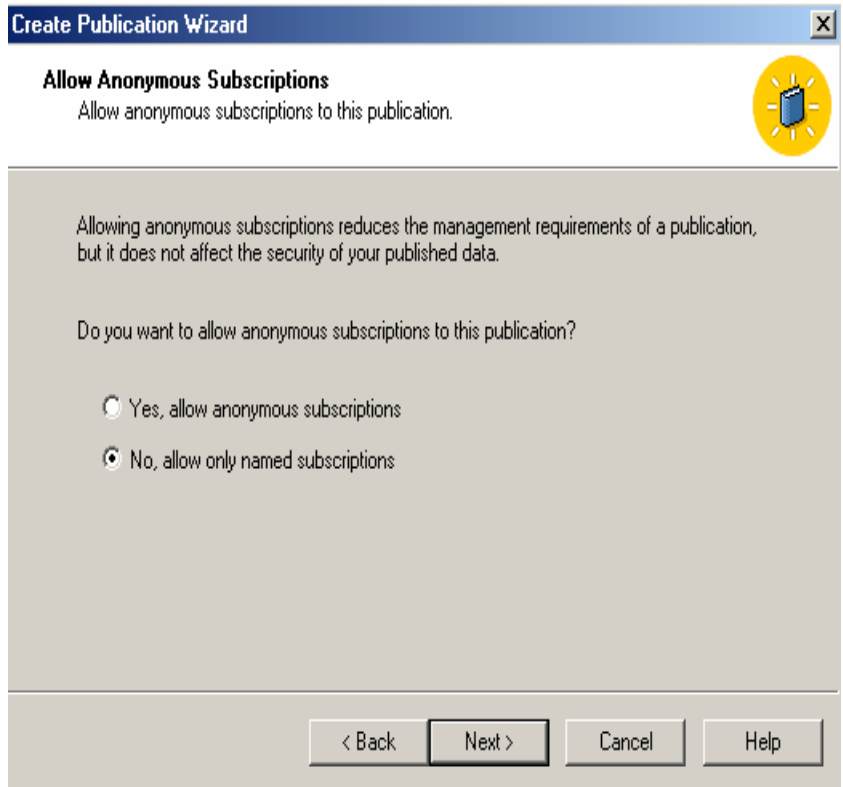
# Tạo và quản lý Publications



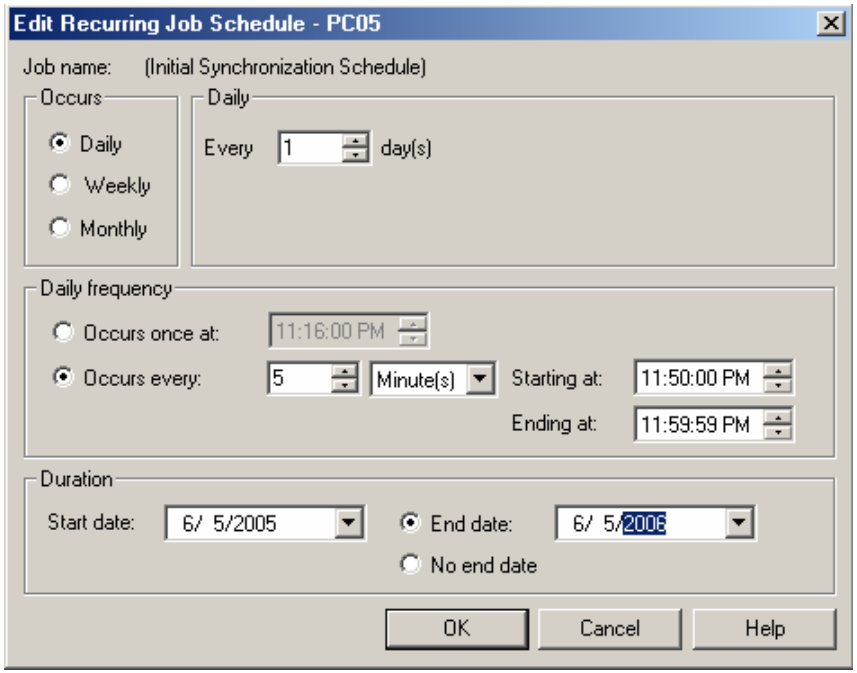
# Tạo và quản lý Publications

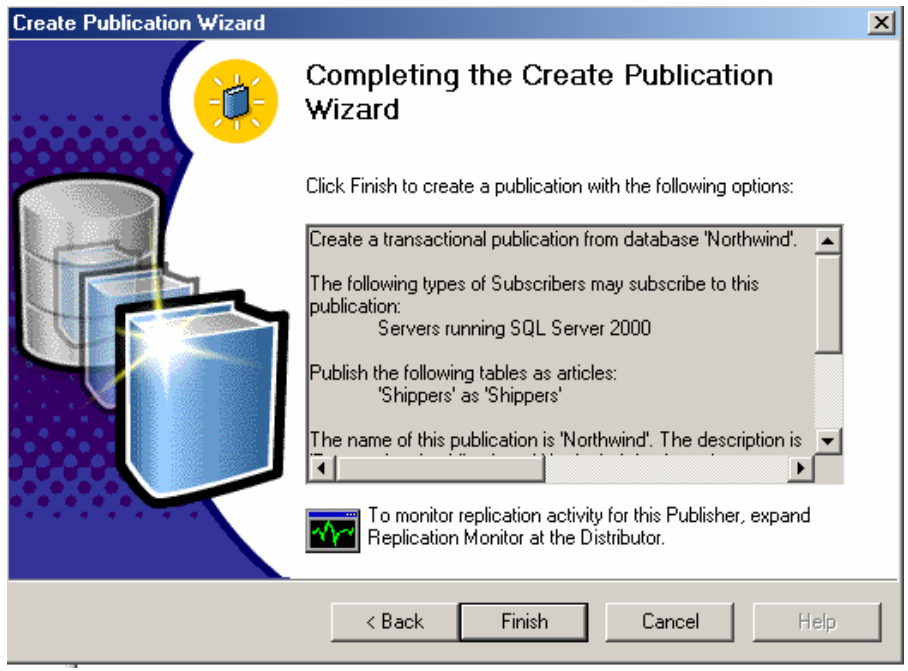


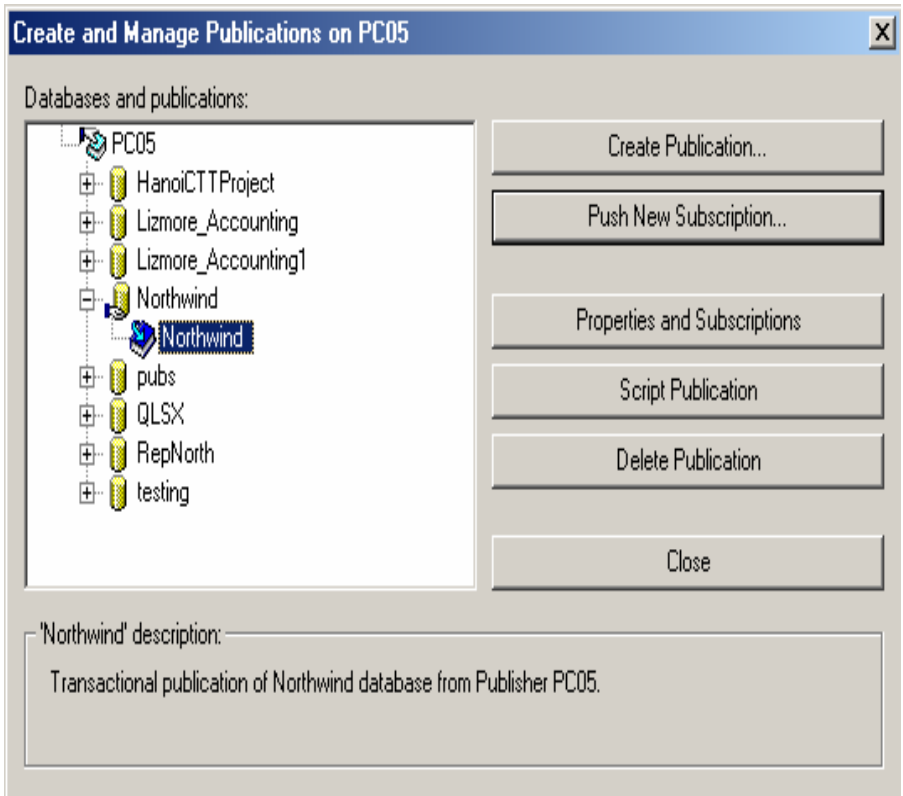
# Tạo và quản lý Publications

**Khi chọn thời gian đặt chế độ tự động copy dữ liệu.**
**Chú ý: Đặt thời gian bắt đầu ít nhất phải lớn hơn thời Gian hiện tại.**

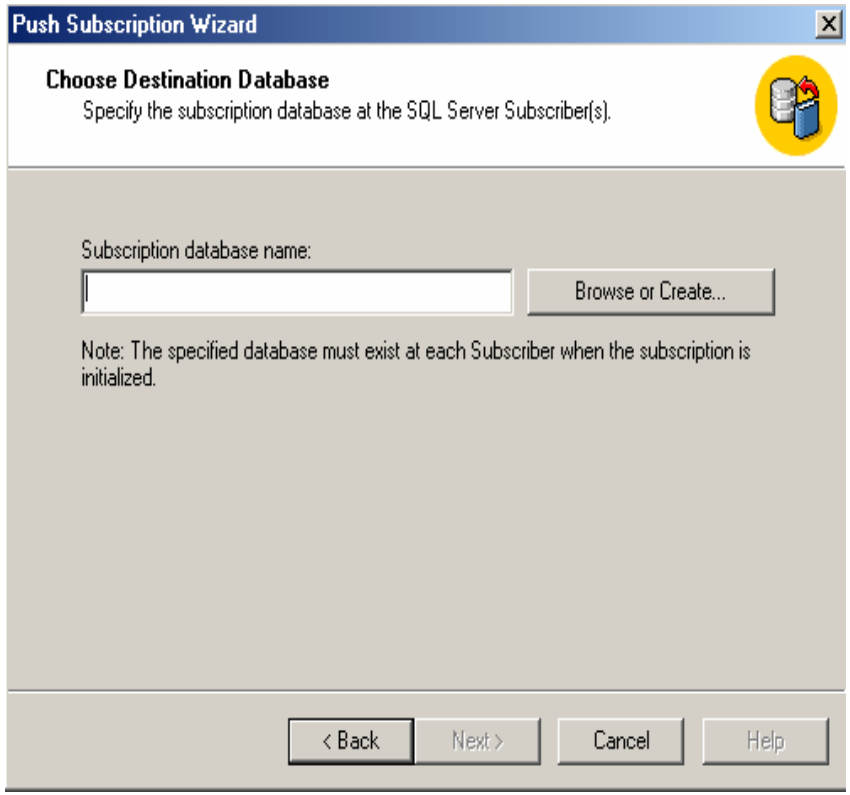


# Tạo Push Subscription

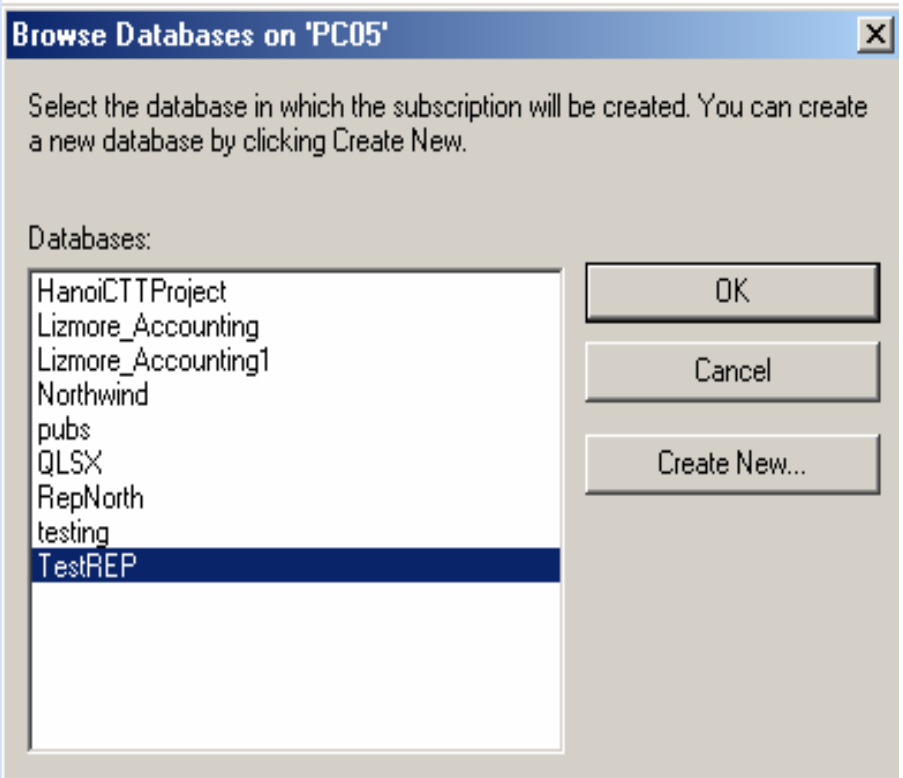


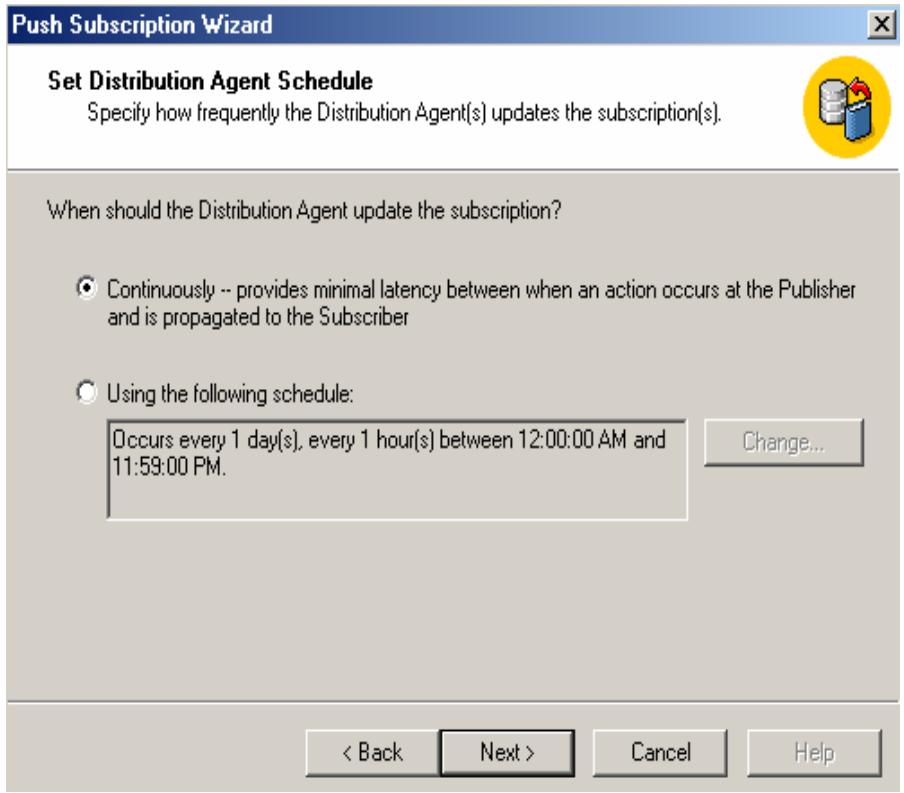

# Tạo Push Subscription

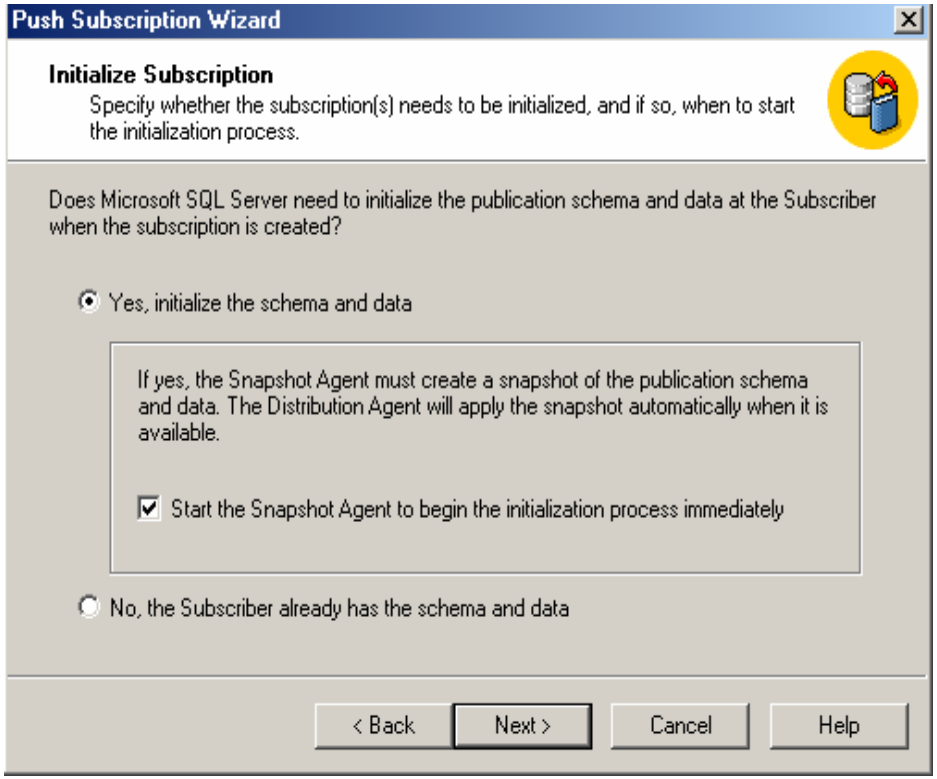

**Chọn Browse ỏ Create New để tạo cái mới nếu cần thiết**

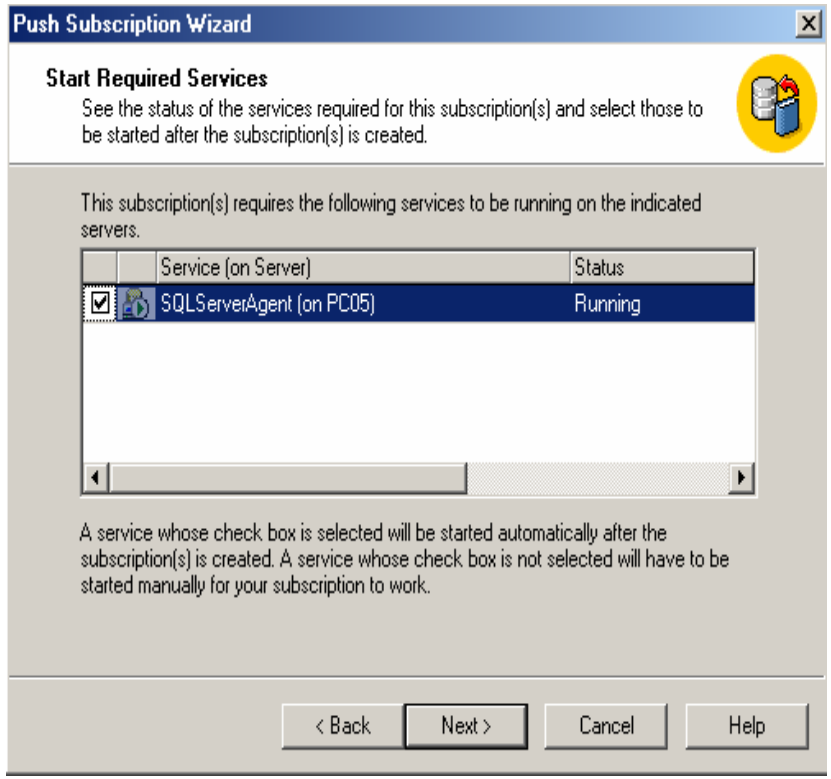
# Tạo Push Subscription



# Tạo Push Subscription



# Tạo Push Subscription

# Chú ý

Việc thiết lập Sao lưu tự động, chúng ta nên đặt database file ở thư mục khác thư mục mặc định.

Tốt nhất đặt khác Ổ ĐĨA hoặc dữ liệu sao lưu nên đặt ở máy khác. Như vậy sẽ hạn chế được rủi ro với máy làm việc hiện taịi

# TÀI LIỆU THAM KHẢO

Nội dung những phần đã học

http://www.tronganh.com/ref/sqlserver/