MỤC LỤC

BÀI 1
CƠ BẢN VỀ LẬP TRÌNH BẰNG TRANSACT- SQL

1.1 Khái niệm cơ bản.

1.1.1 Định danh -IDENTIFIERS.

Tên của tất cả các đối tượng đều được gọi là định danh. Mọi thứ trong SQL Server đều có một định danh, trong đó bao gồm Servers, Databases, và các đối tượng (Object) của CSDL như bảng, Views, cột, chỉ mục, ràng buộc,…. Có những đối tượng bắt buộc phải qui định một định danh, ngược lại có một số đối tượng không cần định danh (SQL Server sẽ tự động định danh)

Ví dụ:

```
CREATE TABLE Table1
(Keycol INT PRIMARY KEY, Description NVARCHAR(30))
```

Qui tắc định danh:

- Tối đa 128 ký tự.
- Bắt đầu là một ký tự thông thường A -> Z
- Bặt đầu là một ký hiệu (@, #) sẽ có một ý nghĩa khác.
- Những định danh nào có khoảng trắng ở giữa thì phải kẹp trong dấu [ ] hoặc “ “
- Đặt các định danh sao cho ngắn gọn, đầu đủ ý nghĩa, phân biệt giữa các đối tượng với nhau, không trùng lắp, không trùng với từ khóa của T-SQL.

1.1.2 Tham chiếu đến các đối tượng trong SQL Server.

- Tên đầy đủ: **Server.database.ower.object**
- Tên ngắn: Nếu là local server thì ta khỏi chỉ Server, nếu ở CSDL hiện hành thì không cần chỉ Database, Owner mặc định là user name trong Database.

Nếu Tham chiếu tường minh thì tăng tốc.

Ví dụ:

```
CREATE TABLE Northwind.dbo.orederhist
CREATE TABLE Northwind..orederhist
```

Database Owner (dbo)

Dbo là một người dùng mà có đầy đủ các quyền thao tác trong CSDL. Bất kỳ một thành viên thuộc nhóm **sysadmin** đóng vai trò người dùng CSDL được gọi là **dbo**. Cũng như bất kỳ một đối tượng nào được tạo bởi bất kỳ thành viên thuộc nhóm **sysadmin** thì mặc nhiên thuộc **dbo.**

Ví dụ: nếu người dùng **Andrew** là thành viên của nhóm **sysadmin** tạo bảng **T1**, thì **T1** thuộc chủ quyền **dbo** và nó thuộc dbo (**dbo.T1)**, không thuộc **Andrew (**không là **Andrew.T1**). Ngược lại, nếu **Andrew** không là thành viên của **sysadmin** nhưng chỉ

là một thành viên của **db_owner** và tạo bảng **T1**, **T1** thuộc về **Andrew** và đủ tư cách gọi là Andrew**.T1**.

User **dbo** không bao giờ bị xóa và nó luôn luôn hiện hữu trong mỗi CSDL. Chỉ có những đối tượng được tạo bởi thành viên của sysadmin (hoặc bởi user dbo) thì thuộc về dbo.

1.1.3 Kiểu dữ liệu (DATA TYPE).

Kiểu dữ liệu là một định nghĩa để xác định loại dữ liệu mà đối tượng có thể chứa đựng. Cột, tham số, biến, giá trị trả về của hàm, thủ tục, tất cả đều phải có kiểu dữ liệu. Kiểu dữ liệu là một đặc tính của một cột (Column). Nó định rõ loại dữ liệu và dạng dữ liệu được nhập vào cột.

Có 2 nhóm:

- **System-Supplied datatype:** Các kiểu dữ liệu cơ bản được hỗ trợ bởi SQL Server.
- **User-defined datatype:** Các kiểu dữ liệu của người dùng tự định nghĩa dựa trên các kiểu dữ liệu cơ bản.

1.1.3.1 System-Supplied Datatype.

System-Supplied Datatype **là** kiểu dữ liệu cơ bản được hỗ trợ bởi SQL Server. Các đối tượng lưu chứa dữ liệu đều có một kiểu dữ liệu để lưu, các đối tượng đó có thể là

- Các cột (Column) trong các bảng.
- Các tham số (parameters) trong strored procedures.
- Các biến (Variables) trong stored procedure, function, script, batch.
- Các hàm T_SQL trả về một hoặc nhiều giá trị thuộc một kiểu dữ liệu nào đó.

Ta có thể dùng các kiểu dữ liệu để tạo các ràng buộc toàn vẹn dữ liệu. Ví dụ cột TENNV thì không thể được định nghĩa với kiểu dữ liệu là Date, vì cột Date chỉ chấp nhận giá trị ngày.

Khi ta gán kiểu dữ liệu cho một đối tượng nào đó thì ta cần quan tâm đến các tính chất sau:

- Loại dữ liệu được chứa đựng bởi đối tượng.
- Chiều dài lưu trữ giá trị hoặc là kích cở của nó.
- Tính đúng của số (đối với các kiểu số).

Các kiểu dữ liệu cơ bản:

Loại	Kiểu dữ liệu cơ sở	Kích cở	Vùng giá trị	Mô tả
Binary	Binary	8 KB	“0”…”9”, “a”..”f”, “A”..”F”	Chứa các bit thông tin

	Varbinary	8 KB	“0”…”9”, “a”..”f”, “A”..”F”	
	Image	2^31 -1 bytes	2^31 –1 bytes	Dữ liệu hình ảnh
Character	Char	255 bytes	1..8000 ký tự	Ký tự hoặc chuỗi
	Varchar	255 bytes	1..8000 ký tự	Ký tự hoặc chuỗi
	Text	2147483647 bytes	2^31-1 ký tự (2147483647)	Ký tự hoặc chuỗi
Date and Time	Datetime	8 bytes	01/01/1753->31/12/9999	Chuỗi biểu diễn ngày giờ
	Smalldatetime	4 bytes	1/1/1900 -> 6/6/2079	Chuỗi biểu diễn ngày giờ
Decimal	Decimal	17 bytes	-10^38-1 -> 10^38-1	Số thực
	Numeric	17 bytes	-10^38-1 -> 10^38-1	Số thực
Foating point	Float	8 bytes	-1.79E+308 -> 1.79E+308	Số thực
	Real	4 bytes	-3.40E+38 ->3.40E+38	Số thực
Integer	Bigint	8 bytes	-2^63 -> 2^63	Số nguyên
	Int	4 bytes	-2^31 -> 2^31-1	Số nguyên
	Smallint	2 bytes	-2^15 -> 2^15-1	Số nguyên
	Tinyint	1 bytes	0..255	Số nguyên
Monetary	Money	8 bytes	-2^63 -> 2^63-1	Dữ liệu tiền tệ
	Smalmoney	4 bytes	-214748.3648 -> 214748.3648	Dữ liệu tiền tệ
Special	Bit	1 bytes	0 hoặc 1	Dữ liệu có một trong hai trạng thái 0 hoặc 1
	Cursor	Kiểu DL cho biến hoặc giá trị trả về của procedure, tham chiếu đến 1 mẫu tin		
	Timestamp	8 bytes	Chuỗi có dạng: 0x000000100000a90	Theo dõi mẫu tin nào bị thay đổi dữ liệu
	Uniqueidentifier	16 bytes	Số thập lục phân	

	SQL_variant	Là kiểu dữ liệu có thể chứa bất kỳ loại dữ tùy ý của SQL Server ngoại trừ **text**, **ntext, image,** and the **timestamp** data type		
	Table			
Unicode	Nchar		4000 ký tự	Ký tự hoặc chuỗi
	Nvarchar		4000 ký tự	Ký tự hoặc chuỗi
	Ntext		2^30-1 ký tự	Ký tự hoặc chuỗi

1.1.3.2 User-defined datatype.

Người sử dụng có thể dựa yêu cầu cần lưu trữ và các kiểu dữ liệu cơ bản để định nghĩa ra một kiểu dữ liệu của người dùng dùng để lưu trữ một dữ liệu đặc biệt nào đó. SQL Sever cho phép bạn cải tiến các kiểu dữ liệu để đảm bảo tính nhất quán khi làm việc trong môi trường dữ liệu đa dạng trong các bảng hay các CSDL khác nhau.

- User-defined data type không cho phép bạn định nghĩa các kiểu dữ liệu phức hoặc có cấu trúc.
- Mỗi một User-defined data type có thể được định nghĩa riêng cho một CSDL hoặc cho toàn bộ các CSDL. Nếu User-defined data type được định nghĩa trong CSDL Master thì nó được dùng chung cho toàn bộ các CSDL.
- Các User-defined data type mà bạn tạo trong CSDL model thì sẽ có trong tất cả các CSDL mới tạo một cách tự động.
- Mỗi user-defined data type được lưu thành một mẫu tin trong bảng **systypes**.
- Bạn có thể tạo và xóa user-defined data type bằng các thủ tục hệ thống. Tên của kiểu dữ liệu phải tuân thủ qui tắt định danh và phải là duy nhất trong mỗi CSDL. Định nghĩa mỗi user-defined data type trong giới hạn của các kiểu dữ liệu cơ bản. Phải chỉ định mặc định là chấp nhận giá trị NULL hay NOT NULL khi đối tượng không có giá trị.

Tạo một User-Defined Data Type

Dùng thủ tục hệ thống ***sp_addtype*** để tạo một user-defined data type.

sp_addtype *type*, *system_data_type* [,'NULL' | 'NOT NULL']

Ví dụ 1: Tạo kiểu dữ liệu tên là **isbn** với kiểu dữ liệu cơ bản là **smallint** và **không chấp nhận giá trị Null**

```
EXEC sp_addtype isbn, 'smallint', NOT NULL
```

Ví dụ 2: Tạo kiểu dữ liệu tên là **zipcode** với kiểu dữ liệu cơ bản là **char**, **độ dài tối đa là 10 và chấp nhận giá trị Null**

```
EXEC sp_addtype zipcode, 'char(10)', NULL
```

Ví dụ 3: Tạo kiểu dữ liệu tên là **longstring** với kiểu dữ liệu cơ bản là **varchar**, **độ dài tối đa là 63** và **chấp nhận giá trị Null**

```
EXEC sp_addtype longstring, 'varchar(63)', NULL
```

Xoá một User-Defined Data Type: dùng thủ tục hệ thống *sp_droptype để xóa một* user-defined data type từ bảng systypes. Một user-defined data type không thể xóa được nếu nó được tham chiếu bởi các bảng và những đối tượng khác.

Sp_droptype *type*

Ví dụ:

```
EXEC sp_droptype isbn
```

Xem các user-defined data types trong CSDL hiện hành: dùng thủ tục **sp_help** hoặc truy vấn trong **information_schema.domains**

Ví dụ:

```
Use SalesDB
Sp_help
  hoặc
   SELECT domain_name, data_type,
   character_maximum_length
   FROM information_schema.domains
   ORDER BY domain_name
```

1.1.4 Batch

Batch là một tập các phát biểu T-SQL nằm liên tiếp và kết thúc bởi phát biểu GO, và được biên dịch đồng thời bởi SQL Server.

Ví dụ:

```
USE pubs
GO /* Signals the end of the batch */

CREATE VIEW auth_titles
AS
SELECT *
FROM authors
GO /* Signals the end of the batch */

SELECT *
FROM auth_titles
GO /* Signals the end of the batch */
```

Lưu ý:

- Các phát biểu trong 1 batch được biên dịch thành một nhóm.
- Nếu một trong phát biểu của batch bị lỗi thì batch cũng xem như lỗi.
- Các phát biểu Create bị ràng buộc trong một batch đơn, tức trong batch đó chỉ có phát biểu Create. Các phát biểu đó là: Create DataBase, Create Table, Create Index,…

1.1.5 Kịch bản - SCRIPT

Một Script là một tập của một hay nhiều bó lệnh được lưu lại thành một tập tin .SQL

1.2 Biến (VARIABLES)

Biến là một đối tượng trong tập lệnh T-SQL mà nó dùng để lưu trữ dữ liệu. Sau khi biến đã được khai báo hoặc định nghĩa, một câu lệnh trong tập lệnh sẽ gán giá trị cho biến và cũng có thể một câu lệnh khác sẽ lấy giá trị của biến ra dùng. Phải được khai báo trước khi dùng.

Biến được dùng để:

- Đếm số lần lặp được thực hiện hoặc dùng để điều khiển vòng lặp
- Dùng lưu giá trị dữ liệu được kiểm tra một số lệnh điều khiển
- Lưu trữ giá trị trả về từ một store Procedure
- ……

Các loại biến: có 2 loại biến là biến cục bộ (local) và biến toàn cục (Global). Biến kiểu Global được SQL Server đưa ra và bạn có thể dùng bất cứ khi nào và cũng không cần khai báo (được xem như là những hàm chuẩn của SQl Server).

Local variable

- Được khai báo trong phần thân của một bó lệnh hoặc một thủ tục.
- Phạm vi hoạt động của biến bắt đầu từ điểm mà nó được khai báo cho đến khi kết thúc một lô (batch) hoặc stored procedure hoặc Function mà nó được khai báo.
- Tên của biến bắt đầu @

Khai báo:

DECLARE @var_name var_type

Gán giá trị cho biến:

Nếu biến vừa khai báo xong thì biến mặc nhiên được gán giá trị là NULL.

Để gán biến ta dùng lệnh SET hoặc dùng câu lệnh SELECT

SET @var_name = *expression*

SELECT { @*var_name* = *expression* } [,...*n*]

Ví dụ 1:

```
USE Northwind
DECLARE @EmpIDVar INT
SET @EmpIDVar=3
SELECT * FROM [Orders]
WHERE
```

Ví dụ 2:

```
DECLARE MyVariable INT
SET @MyVariable = 1
GO – điểm kết thúc một lô batch.
-- @MyVariable đã vượt quá phạm vi và nó đã hết tồn tại
-- Câu lệnh SELECT sau sẽ nhận lỗi sai cú pháp bởi vì nó không tham chiếu được biến @MyVariable.
SELECT * FROM [Orders]
WHERE
```

Ví dụ 3:

```
USE Northwind
GO
-- Khai báo 2 biến.
DECLARE @FirstNameVariable NVARCHAR(20), @RegionVariable NVARCHAR(30)

-- Gán giá trị cho 2 biến.
SET @FirstNameVariable = N'Anne'
SET @RegionVariable = N'WA'

-- Dùng chúng trong mệnh đề WHERE của lệnh SELECT.
SELECT LastName, FirstName, Title
FROM Employees
WHERE FirstName = @FirstNameVariable
   OR Region = @RegionVariable
GO
```

Ví dụ 4:

```
USE Northwind
```

```
GO
-- Khai báo 1 biến
DECLARE @EmpIDVariable INT
-- Gán giá trị biến bằng câu lệnh Select
SELECT @EmpIDVariable = MAX(EmployeeID)
FROM Employees
GO
```

-- Nếu câu Select trả về là một tập giá trị thì biến sẽ nhận giá trị sau cùng.

Ví dụ 5:

```
DECLARE @ProdIDVariable int
SELECT @ProdIDVariable = ProductID
FROM Northwind..Products
```

Ví dụ 6:

```
USE Northwind
GO
DECLARE @EmpIDVariable INT
SELECT @EmpIDVariable = EmployeeID
FROM Employees
ORDER BY EmployeeID DESC
SELECT @EmpIDVariable
GO
```

Global variable trong SQL Server 2000 gọi là System Function: Từ SQL Server 7.0 biến Global được định nghĩa như là hàm hệ thống. Mỗi kết nối sẽ tạo ra một session, SQL Server tạo ra sẵn một số biến có sẵn rất tiện ích trong việc lập trình và quản trị hệ thống. Các biến này không có kiểu, tên bắt đầu @@.

Một hàm hệ thống thường dùng:

@@VERSION	phiên bản của SQL Server và hệ điều hành
	`SELECT @@VERSION`
@@TRANCOUNT	Xem coi có bao nhiêu transaction đang mơ

	`IF (@@TRANCOUNT > 0)`
	`BEGIN`
	`RAISERROR('Task cannot be executed within a transaction.', 10, 1)`
	`RETURN`
	`END`

@@ROWCOUNT	Trả về số dòng bị ảnh hưởng đối với lệnh thực thi gần nhất

	Ví dụ 1:
	`USE Northwind`
	`UPDATE Employees SET LastName = 'Brooke'`
	`  WHERE LastName = 'Brook'`
	`IF (@@ROWCOUNT = 0)`
	`  BEGIN`
	`  PRINT 'Warning: No rows were updated'`
	`  RETURN`
	`  END`
	Ví dụ 2:
	`  UPDATE Customers`
	`  SET Phone =’030’ + Phone`
	`  WHERE Country=’Germany’`
	`  PRINT @@ROWCOUNT`

@@IDENTITY	trả về số Identity phát sinh sau cùng

	`CREATE TABLE TABLE_HD`
	`  (  mahd int Identity Primary Key,`
	`  Ghichu varchar(20) )`
	`CREATE TABLE TABLE_CTHD`
	`  (  Mahd int, Masp char(10), Soluong int)`
	`declare @maso Int`
	`Insert into Table_HD Values ('Mau tin 1')`
	`Insert into Table_HD Values ('Mau tin 2')`
	`set @maso= @@IDENTITY`
	`Insert into Table_CTHD`
	`values (@maso, 'sp001',5)`
	`Insert into Table_CTHD`
	`values (@maso, 'sp002',10)`
	`-- Kiểm tra`
	`SELECT * FROM Table_HD`
	`SELECT * FROM Table_CTHD`

@@ERROR	Trả về lỗi số (STT lỗi) của lệnh sau cùng mà SQL thực thi, là 0 có nghĩa là câu lệnh thực thi hoàn thành.
@@FETCH_STATUS	Trả về trạng thái của lệnh Fetch của biến con trỏ có thành công hay

	không (0: Thành công, -1: bị lỗi hoặc vượt quá phạm vi; -2: Thất bại)

Một số hàm thường dùng:

GetDate()	Lấy ngày, giờ hiện hành của hệ thống
Month(Date); Year(Date)	Lấy tháng, năm của ngày Date
DateAdd(Datepart, Number, Date)	Cộng thêm Date một giá trị số
DATEDIFF (datepart, startdate, enddate)	Khoảng chênh lệch giữa startdate và enddate
DATEPART (*datepart, date)*	Trả về số nguyên biểu diễn datepart nào đó của ngày được chỉ định
CAST (expression AS data_type)	Dùng để chuyển đổi kiểu dữ liệu
CONVERT (*data_type* [(*length)*], *expression* [, *style*])	Dùng để chuyển đổi kiểu dữ liệu
LOWER (*character_expression)*	Chuyển sang chữ thường
UPPER (*character_expression)*	Chuyển sang chữ hoa
REPLACE ('string_expression1', 'string_expression2', 'string_expression3')	Thay thế chuỗi biểu thức
DIFFERENCE (character_expression, character_expression)	So sánh 2 biểu thức

Ví dụ:

```
USE pubs
SELECT 'The price is ' + CAST(price AS
varchar(12))
FROM titles
WHERE price > 10.00
GO

-- Use CONVERT.
USE pubs
SELECT SUBSTRING(title, 1, 30) AS Title,
ytd_sales
FROM titles
WHERE CONVERT(char(20), ytd_sales) LIKE '3%'
GO
```

1.3 Cấu trúc điều khiển.

T-SQL cung cấp một số cấu trúc điều kiển cơ bản để bạn có thể thực thi một khối lệnh dựa trên kết quả của một phép so sánh. Nó cũng tương tự như một số ngôn ngữ lập trình khác.

1.3.1 Khối BEGIN … END

Nếu bạn cần nhiều phát biểu được thực thi với nhau thì ta đặt các phát biệu trong cặp Begin … End. Nó được hữu dụng trong các cấu trúc điều khiển.

1.3.2 Phát biểu PRINT

Phát biểu PRINT: Dùng để in thông tin ra màn hình kết quả của SQL

PRINT 'any ASCII text' | @local_variable | @@FUNCTION | string_expr

Ví dụ:

```
PRINT 'Hello!'
PRINT N'Chào bạn'
PRINT @@VERSION
```

1.3.3 Cấu trúc điều khiển IF … ELSE

Là một cấu trúc điều khiển, cho phép thực thi một hoặc nhiều phát biểu tùy thuộc vào một điều kiện nào đó. câu lệnh để thực thi một khối các câu lệnh theo một điều kiện nào đó.

Cú pháp:

IF condition

{statements}

[ELSE [Condition 1]

{statements}]

Condition: là một biểu thức logic, có giá trị *True* hoặc *False.*Tùy thuộc vào *condition,* một trong hai khối lệnh sẽ được thực thi.

Ví dụ 1: Kiểm tra xem trong Customers của NorthWind có chứa các khách hàng đến từ Germany không?

```
USE NorthWind
IF (SELECT COUNT(*) FROM Customers
     WHERE Country='Germany') > 0
     BEGIN
       Print ' Có tồn tại các khách hàng từ
Germany ở trong cơ sở dữ liệu.'
```

```
        Print ' statements to process German
customers'
     END
ELSE
     BEGIN
        PRINT ' Không có khách hàng đến tử Germany
trong cơ sở dữ liệu.’
     END
```

Ví dụ 2:

```
USE pubs
GO
DECLARE @msg varchar(255)
IF ( SELECT COUNT(price) FROM titles
WHERE title_id LIKE 'TC%' AND price BETWEEN 10
AND 20) > 0
BEGIN
   SET NOCOUNT ON
   SET @msg = 'Có vài quyển sách có giá từ $10
đến $20. Các sách đó là:'
   PRINT @msg
   SELECT title FROM titles
    WHERE title_id LIKE 'TC%' AND price BETWEEN
10 AND 20
END
ELSE
BEGIN
   SET NOCOUNT ON
   SET @msg = 'Không có quyển sách nào có giá từ
$10 đến $20. Bạn nên tham khảo các quyễn sách có
giá nhỏ hơn $10 sau đây.'
   PRINT @msg
   SELECT title FROM titles
   WHERE title_id LIKE 'TC%' AND price < 10
END
```

Ví dụ 3:

```
USE pubs
```

```
IF (SELECT AVG(price) FROM titles WHERE type =
'mod_cook') < $15
BEGIN
    PRINT 'The following titles are excellent
mod_cook books:'
    PRINT ' '
    SELECT SUBSTRING(title, 1, 35) AS Title
    FROM titles WHERE type = 'mod_cook'
END
ELSE
    IF (SELECT AVG(price)
          FROM titles WHERE type = 'mod_cook')>
    $15
    BEGIN
        PRINT 'The following titles are
    expensive mod_cook books:'
        PRINT ' '
        SELECT SUBSTRING(title, 1, 35) AS Title
        FROM titles WHERE type = 'mod_cook'
    END
```

1.3.4 Biểu thức CASE.

Biểu thức CASE là một biểu thức điều kiện được áp dụng bên trong một phát biểu khác. Case trả về các giá trị khác nhau tuỳ thuộc vào điều kiện hoặc một điều khiển nào đó.

Dạng 1:

```
CASE input_expression
  WHEN when_expression THEN result_expression
    [...n]
  [
    ELSE else_result_expression
  ]
END
```

Dạng 2:

```
CASE
    WHEN Boolean_expression THEN result_expression
    [...n]
  [
    ELSE else_result_expression
```

```
    ]
END
```

Ví dụ 1: Cho 2 số a và b, so sánh 2 số đó, số nào lơn hơn số nào?

```
DECLARE @a As int, @b As int, @ketqua as nvarchar(30)
SET @a=3
SET @b=5
SET @ketqua = CASE
              WHEN @a<@b THEN N'A nhỏ hơn B'
              -- When chỉ dùng trong case
              WHEN @a>@b THEN N'A lớn hơn B'
             ELSE N'A bằng B’
         END  -- END của CASE
PRINT @ketqua
```

Ví dụ 2: Dựa vào price của các title cho biết price đó như thế nào?

```
Use Pub
SELECT title, price,
'classification'=CASE
    WHEN price < 10.00 THEN 'Low Priced'
    WHEN price BETWEEN 10.00 AND 20.00 THEN 'Moderate
ly Priced'
    WHEN price > 20.00 THEN 'Expensive'
    ELSE 'Unknown'
    END
FROM titles
```

Ví dụ 3: Cho biết ý nghĩa của đoạn lệnh sau:

```
USE NorthWind
SELECT ProductID, Quantity, UnitPrice, [discount%]=
     CASE
          WHEN Quantity <=5 THEN 0.05
          WHEN Quantity BETWEEN 6 and 10 THEN 0.07
          WHEN Quantity BETWEEN 11 and 20 THEN 0.09
     ELSE
```

```
        0.1
    END
    FROM [Order Details]
    ORDER BY Quantity, ProductId
```

Ví dụ 4: Ý nghĩa của đoạn lệnh sau?

```
SELECT title, pub_id,
    CASE WHEN price IS NULL THEN (SELECT MIN(price)
                                  FROM titles)
    ELSE price
    END
FROM titles
```

1.3.5 Cấu trúc vòng lập WHILE …

Là phát biểu điều khiển vòng lặp. Vòng lặp sẽ thực hiện cho đến khi biểu thức điều kiện (Boolean_expression) trong While mang giá trị False. Biểu thức điều kiện có thể là một câu SELECT.

```
WHILE Boolean_expression
  { sql_statement | statement_block }
  [ BREAK ]
  { sql_statement | statement_block }
  [ CONTINUE ]
```

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

[CONTINUE] : Lặp lại đầu vòng lặp.

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

Ví dụ 1: Trong khi đơn giá trung bình vẫn còn nhỏ hơn $30 thì cập nhật các đơn giá tăng lên gấp đôi đơn giá cũ.

```
USE pubs
GO
WHILE (SELECT AVG(price) FROM titles) < $30
BEGIN
   UPDATE titles
      SET price = price * 2
   SELECT MAX(price) FROM titles
   IF (SELECT MAX(price) FROM titles) > $50
```

```
            BREAK
        ELSE
            CONTINUE
    END
    PRINT 'Too much for the market to bear'
```

1.3.6 Lệnh RETURN

RETURN [integer_expression]

Nếu gặp phát biểu Return, quá trình xử lý sẽ kết thúc

Đôi khi ta dùng RETURN trong thủ tục để thủ tục có thể trở thành hàm như các ngôn ngữ khác.

1.3.7 Lệnh WAITFOR

Là một chỉ thị cho SQL Server tạm dừng một thời gian trước khi xử lý tiếp các phát biểu sau đó.

WAITFOR { DELAY '*time*' | TIME '*time*' }

'Time': được viết thao dạng hh:mm:ss, tối đa là 24 giờ

DEPLAY 'Time': Hệ thống tạm dừng trong khoảng thời gian 'Time'.

TIME 'Time': Hệ thống được tạm dựng đền thời gian 'Time' chỉ ra.

```
WAITFOR  '02:10'
WAITFOR  '02:10'
```

1.3.8 Lệnh RAISERROR

Phát sinh lỗi của người dùng. Người dùng có thể phát sinh các lỗi từ bảng sysmessage hoặc xây dựng lỗi động tùy thông tin của người dùng. Sau khi một lỗi được định nghĩa thì nó được gửi đến người dùng như là một lỗi hệ thống.

RAISERROR ({ *msg_id* | *msg_str* } {, *severity*, *state* }
[, *argument* [,...*n*]])
[WITH *option* [,...*n*]]

msg_id: là mã thông báo, nó được lưu trong bảng sysmessage. Mã thông báo của người dùng định nghĩa phải được bắt đầu từ trên 50000

msg_str : Nội dung thông báo, tối đa 400 ký tự.

Để truyền tham số vào trong thông báo thì dùng dạng %<Loai ký tự>

Loại ký tự là d, i, o, x, X, hoặc u

Các ký tự	**Mô tả**
d hoặc I	Biểu hiện là số nguyên (integer)
O	Octal không dấu

P	Con trỏ
S	Chuỗi
U	So nguyên không dấu
x or X	Hexadecimal không dấu

Lưu ý: số **float**, double, char không được hỗ trợ

severity: Độ nghiêm khắt của thông báo

Severity Levels: Mức lỗi của một thông báo lỗi cung cấp một sự biểu thị loại vấn đề mà SQL Server gặp phải.

- Mức lỗi **10** là lỗi về thông tin và biểu thị nguyên nhân do thông tin nhập vào.

- Mức lỗi từ **11 đến 16** thì thông thường là do các user.

- Mức từ **17 đến 25** do lỗi phần mềm hoặc phần cứng. Bạn nên báo cho nhà quản trị hệ thống bất cứ khi nào sự cố xảy ra. Nhà quả trị hệ thống phải giải quyết sự cố đó và theo dõi chúng thường xuyên. Khi mức lỗi 17,18,19 xảy ra, bạn có thể tiếp tục làm việc mặc dù bạn không thể thực thi lệnh đặc biệt.

- **Mức lỗi 17:** Những thông báo này cho biết rằng câu lệnh nguyên nhân SQL Server cạn kiệt tài nguyên (Ví dụ như lock hoặc không gian đĩa cho CSDL) hoặc vượt quá tập giới hạn bởi nhà quản trị

- Người quản trị hệ thống nên giám sát tất cả các sự cố mà được phát ra mức trầm trọng từ **17 đến 25** và in ra giải thích lỗi mà bao gồm các thông tin để quay lại từ lỗi.

Nếu sự cố ảnh hưởng đến toàn bộ một CSDL, bạn dùng DBCC CHECKDB (Database) để xác định phạm vi của sự thiệt hại. DBCC có thể xác định một vài đối tượng mà phải bị di chuyển và sẽ tùy ý phục hồi sự thiệt hại. Nếu thiệt hại là lớn, CSDL có thể không phục hồi được. Trong trường hợp đặc biệt, người dùng định nghĩa thông báo lỗi với RAIERROR, dùng mã lỗi trên 50000 và mức lỗi trầm trọng từ 0 đến 8. Chỉ có nhà quản trị hệ thống có thể phát hành lỗi với mức trầm trọng từ 19 đến 25.

- Mức lỗi từ 20 đến 25 chỉ ra sự cố hệ thống. Đó là lỗi không tránh được, mà có nghĩa là tiến trình không còn đang chạy. Tiến trình tê liệt trước khi nó dừng, ghi nhận thông tin về cái gì xảy ra, và sau đó kết thúc.

state: Là một số nguyên tùy ý từ 1 đến 127 mà nó mô tả thông tin diễn giải về trạng thái lỗi.

Argument: Là tham số dùng trong việc thay thế cho biến để định nghĩa thông báo lỗi hoặc thông báo tương ứng với mã lỗi msg_id. Có thể không hoặc có nhiều tham số. Tuy nhiên, không được quá 20. Mỗi tham số thay thế có thể là một biến local hoặc bất kỳ một trong các kiểu dữ liệu int, char, varchar, binary, varbinary. Các kiểu khác không được cung cấp.

Thêm một lỗi mới của người dùng định nghĩa:

Sp_addMessage msg_ID, *severity, 'msg'* [, *'language'*] [, *'with_log'*] [, *'replace'*]

Xóa một lỗi của người dùng

sp_dropmessage Msg-ID

Giải thích

msg_id: là mã số của lỗi mới, là một số int, không được trùng các mã đã có sẵn, bắt đầu là 50001.

severity: là mức lỗi của lỗi, là một số smallint. Mức hợp lệ là từ 1 đến 25. Chỉ có người quản trị CSDL mới có thể phát sinh thêm một thông báo lỗi mới từ 19 đến 25.

'*msg*': là một chuỗi thông báo lỗi, tối đa 255 ký tự.

'*language*': là ngôn ngữ của thông báo lỗi, không chỉ định tì mặc định là ngôn ngữ của phiên kết nối.

'*with_log*': thông báo lỗi có được ghi nhận vào nhật ký của ứng dụng khi nó xảy ra hay không, mặc định là FALSE. Nếu là **true**, thì lỗi luôn luôn được ghi vào nhật ký ứng dụng. Chỉ có những thành viên thuộc **sysadmin** server role mới có thể sử dụng tham số này.

'*replace*': nếu được chỉ định chuỗi **REPLACE**, thì thông báo lỗi đã tồn tại được ghi đè bởi chuỗi thông báo mới và mức lỗi mới. Tham số này phải chỉ định nếu *msg_id* đã có.

Lưu ý: *nếu trả về 0 tức là thêm vào thành công, 1 thất bại*.

BÀI 2:
PROCEDURES, FUNCTIONS

2.1 STORED PROCEDURES.

2.1.1 Giới thiệu Stored proccedures.

Stored procedure (thủ tục) là một tập các lệnh T-SQL và một số cấu trúc điều kiển, được lưu với một tên và được thực thi như là một đơn vị công việc đơn (single unit of work). Trong các ngôn ngữ khác như C, pascal, Basic, một thủ tục thông thường là một tập các câu lệnh với mục đích hoàn tất một mục đích nào đó và có thể được gọi từ một chương trình như là một lệnh đơn.

- Thủ tục trong SQL Server được lưu trữ tại server khi nó được tạo ra. Vì vậy, khi thủ tục được thi hành thì nó được chạy tại Server. Có thể gọi thủ tục chạy bằng một lệnh đơn giản và trong thủ tục có thể chứa rất nhiều lệnh của T-SQL.

- Trước khi thủ tục được tạo, SQL Server sẽ kiểm tra tính đúng đắn của các cú pháp lệnh. Nếu không có lỗi về cú pháp thì thủ tục được tạo, tên của thủ tục được lưu trong bảng hệ thống **SysObjects** và nội dung được lưu trong bảng hệ thống **SysCommanes**.

- Trước khi thủ tục được thực thi, một kế hoạch thực thi (Execution plain) được tạo ra và thủ tục được biên dịch. Từ đó trở về sau tiến trình dịch thủ tục nhanh hơn bởi vì SQL Server sẽ không kiểm tra tính đúng đắng của các câu lệnh nữa, chỉ tạo lại execution plan và biên dịch lại thủ tục.

- Security (an toàn): thủ tục có một đặc tính quan trọng là nó có thể được nâng cao an toàn thông tin thông qua isolation (cô lập) hoặc encryption (mã hóa). Người dùng CSDL có thể được cho quyền thực thi thủ tục nhưng sẽ không có quyền trực tiếp truy xuất các đối tượng của thủ tục. Một thủ tục có thể được mã hóa ngay khi được tạo hoặc chỉnh sửa vì thế người sử dụng không thể đọc được các câu lệnh trong thủ tục.

Phân loại:

- **System sp:** được lưu trữ trong CSDL master và được đặt tên với tiếp đầu ngữ là **sp**. Chúng đóng vai trò khác nhau của các tác vụ được cung cấp trong SQL Server.

  Ví dụ: Sp_help, Sp_helpConstraint, ….

- **Local sp:** được lưu trữ trong các CSDL của người dùng, nó thực thi các tác vụ (Task) trong CSDL chứa nó. Một Local sp có thể được người sử dụng tạo hoặc từ các sp hệ thống.

- **Temporary sp:** giống như là một local sp, nhưng nó chỉ hiện hữu cho đến khi kết nối tạo ra nó bị đóng. Nó được nằm trong CSDL TempDB. Có 3 loại temporary sp: local (private), Global, sp tạo trực tiếp trong TempDB.

- **Extended sp:** Là một thủ tục được tạo từ các ngôn ngữ lập trình khác (không phải SQL Server) và nó được triển khai tính năng của một thủ tục trong SQL Server. Các thủ tục này có tên với tiếp đầu ngữ là xp.

- **Remote sp:** là một thủ tục được gọi thực thi từ một server từ xa.

2.1.2 Tạo, thực thi, hiệu chỉnh, xóa stored procedures.

- Khi thực thi thủ tục, bạn phải cung cấp các giá trị của tham số của các thủ tục nếu có. Một thủ tục có thể được gọi thực thi hoặc tự động thực khi SQL Server khởi động. Gọi thực thi bằng từ khóa EXECUTE.
- Khi cần thêm một thàm số (parameter) hoặc thay đổi một vài phần trong đoạn mã thì ta dùng lệnh ALTER để hiệu chỉnh.
- Xóa một thủ tục dùng lệnh DROP
- Các thủ tục có thể được tạo trước khi các đối tượng mà thủ tục tham chiếu, đặt tính này gọi là tính trì hoãn.

Tạo thủ tục

Cách 1: Dùng Enterprice Manager

R-Click tại Store procudure trong CSDL, chọn New Store procudure

Cách 2: Tạo Stored procedure Wizard

Tool → Wizard, click vào DataBase, chọn Create Store Procedure Wizard

Cách 3: Bằng lệnh Create proceduce

CREATE PROC [EDURE] ***procedure_name*** **[;** ***number*** **]**
[{ ***@parameter data_type*** **} [VARYING] [=** ***default*** **] [OUTPUT]**
] [,...*n*] [WITH { RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION }] [FOR REPLICATION]
AS

sql_statement **[...*n*]**

[VARYING]: Chỉ dùng với biến Cursor

Ví dụ 1: Tạo thủ tục để liệt kê các order có ngày giao hàng đã quá hạn theo yêu cầu

```
CREATE PROC dbo.overdueOrders  – quá hạn
AS
    SELECT *
    FROM dbo.orders
    WHERE RequiredDate < GETDATE() and shippeddate
    is null
```

Kiểm tra sự tồn tại của Stored procedures

sp_helptext proc_name

Thực thi một Stored procedures

[[EXEC [UTE]]
{
[***@return_status*** **=]**

{ *procedure_name* [;*number*] | @*procedure_name_var*
}
[[@*parameter* =] { *value* | @*variable* [OUTPUT] | [DEFAULT]]
[,...*n*]
[WITH RECOMPILE]

Đơn giản hơn

EXECUTE ProductName [; *number*][<parameter>[, …n][OUTPUT]]

```
Ví dụ:   EXECUTE  dbo.overdueOrders
```

Thực thi sp ngay khi SQL Server khởi động: Store procedure phải nằm ở CSDL **Master.**

Cách 1: Dùng thủ tục **Sp_procoption** để gán thuộc tính tự động thực thi

USE Master

EXECute Sp_procoption [@ProcName =] '*procedure*'

, [@OptionName =] Startup

, [@OptionValue =] True

Ví dụ:

```
EXECute Sp_procoption dbo.overdueOrders Startup, True
```

Cách 2: Dùng **Enterprice Manager**

R-Click tại tên thủ tục →Properties → Execute whenever SQL Server Start

Hiệu chỉnh một stored procedures

```
USE Northwind
GO
ALTER PROC dbo.overdueOrders
AS
    SELECT CONVERT(CHAR(8), RequiredDate,1)
    RequiredDate, CONVERT(CHAR(8), orderDate,1)
    orderDate, orderId, Customerid, EmployeeID
    FROM dbo.orders
    WHERE RequiredDate<GETDATE()and shippeddate is
    null
    ORDER BY RequiredDate
```

Xóa một stored procedures

DROP ProcedureName

2.1.3 Tham số và biến trong Stored procedures.

Tham số và biến là phần cơ bản để tạo nên sự uyển chuyển của thủ tục.

- **Input parameter**: tham số nhập, đưa giá trị của tham số để thông báo cho thủ tục nên làm gì trong CSDL
- **Output parameter**: tham số xuất chứa giá trị trả về của thủ tục.

Khi dùng tham số phải khai báo tham số (Tên tham số, kiểu dữ liệu, Giá trị mặc nhiên nếu có, có chỉ dẫn tham số OUT PUT không)

@parameter_name [AS] datatype

[=default | NULL] [VARYING] [OUTPUT]

Ví dụ 1: Tạo thủ tục dùng để chèn một mẫu tin vào bảng Customer, với các tham số dạng Input

```
USE SalesDB
GO

CREATE PROC Sp_InsertCust
@No_para VARCHAR(10),
@Name_para NVARCHAR(50),
@Address_para NVARCHAR(50),
@Phone_para VARCHAR(24),
@Fax_para VARCHAR(24),
@Mail_para VARCHAR(50)
AS
INSERT INTO tblCustomer
(CustNo, CustName, Address, Phone, Fdax, Mail)
VALUES (@No_para, @Name_para, @Address_para,
 @Phone_para, @Fax_para,@Mail_para)
```

Thực thi thủ tục có tham số

```
USE SalesDB
GO
Sp_InsertCust ‘CDS’, ‘Trường Tin học ABC’,’12
Nguyễn Văn Bảo’,
‘2352344’,234652’,’cds@yahoo.com’
```

Kiểm tra việc chèn dữ liệu

```
SELECT CustMo, CustName, Address
FROM tblCustomer
```

Ví dụ 2:

```
USE Northwind
GO
IF EXISTS (SELECT name
          FROM  sysobjects
           WHERE  name = 'GetUnitPrice' AND
type = 'P')
         DROP PROCEDURE GetUnitPrice
GO

CREATE PROCEDURE GetUnitPrice @prod_id int, @uni
t_price money OUTPUT
AS
     SELECT @unit_price = UnitPrice
     FROM  Products
     WHERE ProductID = @prod_id
     GO
```

Thực thi: Bạn phải khai báo một biến trước khi gọi thủ tục thực thi

```
DECLARE @price money
EXECUTE GetUnitPrice 77, @unit_price = @price OUTPUT
PRINT CONVERT(varchar(6), @price)
GO
```

Ví dụ 3: Tạo thủ tục InsertRows như sau:

```
USE SalesDB
GO
CREATE TABLE mytable
     ( column1 int, column2 char(10) )
GO
CREATE PROCEDURE InsertRows @start_value int
AS
DECLARE @loop_counter int, @start_val int
SET   @start_val = @start_value - 1
SET   @loop_counter = 0
```

```
WHILE (@loop_counter < 5)
   BEGIN
      INSERT INTO mytable VALUES (@start_val + 1, "new row"
)
      PRINT (@start_val)
      SET @start_val = @start_val + 1
      SET @loop_counter = @loop_counter + 1
   END
GO
```

Thực thi: Hãy thực thi thủ tục với giá trị khởi tạo là 1

```
EXECUTE InsertRows 1
GO
```

Kiểm tra kết quả

```
     SELECT * FROM mytable
```

Ví dụ 4:

```
--- Tạo thủ tục Count_tables có 2 tham số Output
USE Pubs
GO
CREATE PROC count_tables
     @authorcount INT OUTPUT,
     @titlecount INT OUTPUT
AS
SELECT * FROM authors
SET @authorcount=@@rowcount

SELECT * FROM Titles
SET @titlecount=@@rowcount

---- Thực thi thủ tục Count_tables
DECLARE @a_count INT, @t_count INT
EXECUTE count_tables @a_count OUTPUT, @t_count OUTPUT
SELECT authorcount=@a_count, titlecount=@t_count
```

Ví dụ 5: Viết một thủ tục tính giai thừa của một số nằm trong khoảng 0 đến 12

Dùng T-SQL để tính giai thừa bằng đệ quy, của các số từ 0 đến 12. Các tham số có giá trị lớn hơn 12 sẽ không cho phép bởi vì kết quả trả về sẽ vượt quá phạm vi của dữ liệu kiểu int

```
CREATE PROC Giaithua @so1 int
AS
DECLARE @So1Giam1 int, @Ketqua int
IF (@so1 < 0 OR @so1 > 12)
    BEGIN
        -- Giá trị tham số không hợp lệ.
        RETURN -1
    END

IF (@so1=0 or @so1=1)
    SELECT @Ketqua=1
ELSE
    BEGIN
        SET @ So1Giam1=@So1 - 1
        EXEC @Ketqua=Giaithua @ So1Giam1 – Đệ qui gọi
lại chính nó
        IF (@Ketqua= -1)
            BEGIN
                RETURN -1
            END

        SET @Ketqua=@Ketqua * @so1
        IF (@@ERROR <> 0)
            RETURN -1
    END

     RETURN(@Ketqua)
```

Thực thi: Tính giai thừa cho 3

```
DECLARE @kq  INT
EXEC @kq = giaithua 3
PRINT @kq
```

Khi thủ tục Giathua đã tồn tại thì chúng ta có thể dùng bó lệnh sau để hiển thị giai thừa của tất cả các số từ 0 đến 12:

```
DECLARE @Ketqua int, @n int
SET @Ketqua =0
SET @n =0
WHILE (@n <= 12) BEGIN
  EXEC @Ketqua = Giaithua @n
    IF (@Ketqua = -1) BEGIN
        RAISERROR('Error executing factorial procedur
e.', 16, -1)
        RETURN
    END
PRINT CONVERT(varchar, @n) + '! = ' + CONVERT(varchar
(50), @Ketqua)
SET @n=@n + 1
END
```

2.2 FUNCTIONS.

Hàm thực sự tương tự như Stored procedure của SQl Server, nội dung bao gồm các phát biểu T-SQL kết hợp tạo thành hàm, có thể gọi thực thi các hàm như là một đơn vị độc lập.

Hàm được dùng trong:

- Danh sách chọn của một câu lệnh Select để cho ra một giá trị.
- Một điều kiện tìm kiếm của mệnh đề Where trong các câu lệnh T-SQL

Buil-in functions: Những hàm này hoạt động như là một định nghĩa trong T-SQL và không thể hiệu chỉnh. Nó có thể chỉ được tham chiếu trong các câu lệnh T-SQL. Giá trị trả về của hàm có thể là một Rowset (tập các dòng), argergate và scalar (vô hướng).

Nên tìm hiểu và tận dụng tối đa các hàm Buil-in function của SQL Server

User-defined function hay còn gọi là UDFs: Những hàm này do người dùng tự định nghĩa để đáp ứng một mục tiêu nào đó. Một số hạn chế so với thủ tục là các tham số truyền vào không được mang thuộc tính OUTPUT, nghĩa là giá trị của tham số không được truyền ra bên ngoài hàm UDF, thay vào đó ta phải sử dụng giải pháp là trả về giá trị cho hàm bằng phát biểu RETURN. Giá trị trả về của hàm có thể là một giá trị vô hướng (Scalar valued) hoặc bảng (Table-valued)

Scalar Function. Một hàm vô hướng trả về một giá trị đơn và có thể được dùng bất cứ nơi nào biểu thức hay biến có thể được dùng (câu lệnh Select, mệnh đề SET của câu lệnh Update). Một hàm vô hướng có thể được xem như kết quả của vài phép toán hoặc hàm chuỗi.

Table-valued Function. Một hàm có giá trị bảng trả về một tập kết quả và có thể được dùng bất cứ nơi nào bảng hay view được dùng. Hàm giá trị bảng có thể được tham chiếu trong mệnh đề FROM của câu lệnh SELECT. Các hàm người dùng có thể có nhiều phức tạp hơn viêu và có thể có tham số.

Với các ngôn ngữ lập trình khác, nếu tham số không truyền vào thì xem như là hàm sẽ lấy giá trị default của các tham số, nhưng với SQL Server thì phải truyền giá trị Default vào.

2.2.1 Scalar Functions

```
CREATE FUNCTION [ owner_name. ] function_name
   ([ { @parameter_name [AS] scalar_parameter_data_type [ = default ] }
[,...n ] ])
RETURNS scalar_return_data_type
[ WITH < function_option> [ [,] ...n] ]
[ AS ]
BEGIN
   function_body
   RETURN scalar_expression
END
```

Ví dụ:

```
CREATE FUNCTION TotalAmount
(@UnitPrice money, @Quantity smallint, @Discount
real)
RETURNS money
AS
BEGIN
RETURN (@UnitPrice*@Quantity)*(1-@discount)
END
```

Sử dụng:

```
SELECT ProductID,
Total=dbo.TotalAmount(UnitPrice, Quantity,
Discount)
FROM [Order details]
WHERE OrderID=10250
```

2.2.2 Table-valued Functons

Được chia thành hai loại nhỏ: inline table-value và multistatement table-valued.

Một hàm *inline table-valued:* Nó có thể được xem như là một View có tham số. Chúng thực thi một câu lệnh Select như trong một view nhưng có thể bao gồm các tham số, giống như thủ tục.

CREATE FUNCTION [*owner_name.*] *function_name*
([{ *@parameter_name* [AS] *scalar_parameter_data_type* [= *default*] }
[,...*n*]])

RETURNS TABLE

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

[AS]

RETURN [(] *select-stmt* [)]]

Ví dụ 1:

```
CREATE FUNCTION SalesByCategory(@Categoryid Int)
RETURNS TABLE
AS
RETURN
(SELECT c.CategoryName, P. ProductName,
        SUM(Quantity) AS TotalQty
  FROM Categories c
  INNER JOIN Products p ON c.CategoryID= p.
CategoryID
  INNER JOIN [Order Details] od ON p.ProductID =
od.ProductID
  WHERE c.CategoryID= @Categoryid
GROUP BY c. CategoryName,p.ProductName)
```

Hàm Multistatement Table-valuesd là dạng phức tạp nhất. Loại hàm này xây dựng tập kết quả từ một hay nhiều câu lệnh SELECT.

CREATE FUNCTION [owner_name.]function_name

([{@parameter_name [AS] data_type [=default]} [,…n]])

RETURNS @return_variable

TABLE ({column_definition | table_constraint} [,…n])

[WITH { ENCRYPTION | SCHEMABINDING } [[,] ...n]]

[AS]

BEGIN

function_body

```
RETURN
END
```

Ví dụ:

```
CREATE FUNCTION Contacts(@suppliers bit=0)
RETURNS @Contacts TABLE (ContactName
nvarchar(30), Phone nvarchar(24), ContactType
nvarchar(15))
AS
BEGIN
INSERT @Contacts
SELECT ContactName, Phone, 'Customer' FROM
Customers
INSERT @Contacts
SELECT FirstName + ' ' + LastName, HomePhone,
'Employee'
FROM Employees
IF @Suppliers=1
INSERT @Contacts
SELECT ContactName, Phone, ’Supplier‘
FROM Suppliers
RETURN
END
```

Sử dụng

```
SELECT * FROM CONTACTS(1) ORDER BY ContactName
```

BÀI 3:
TRANSACTIONS – LOCK

SQL Server dùng các transaction và các lock để đảm bảo tính vững chắc và toàn vẹn dữ liệu, không chấp nhận các lỗi xảy ra trong hệ thống.

3.1 TRANSACTIONS

Transaction: Một transactin có thể được định nghĩa như là một chuỗi các thao tác thực thi cùng với nhau như là một khối thống nhất đơn của công việc. Một khối thống nhất của công việc phải có bốn đặc điểm được gọi là *ACID* (*Atomicity*, *Consistency*, *Isolation*, và *Durability*).

▪ **Atomicity**: đặc tính này thể hiện rằng hoặc là tất cả các hiệu chỉnh dữ liệu được thực hiện hoặc là tất cả chúng đều không được thực hiện.

▪ **Consistency**: đây là một trạng thái mà tất cả các dữ liệu ở trong tình trạng nhất quán sau khi một transaction được hoàn tất một cách thành công. Tất cả các qui tắc (rules) trong một CSDL quan hệ phải được thoả mãn đối với các hiệu chỉnh trong một transaction nhằm duy trì toàn bộ các toàn vẹn dữ liệu.

▪ **Isolation**: đặc tính này thể hiện rằng bất kỳ sự hiệu chỉnh dữ liệu thực hiện bởi các transaction đồng thời phải độc lập với các hiệu chỉnh khác của các transaction đồng thời khác. Nói một cách đơn giản hơn, một transaction hoặc là truy xuất dữ liệu ở trạng thái mà trước khi transaction đồng thời thực hiện hiệu chỉnh hoặc là truy xuất dữ liệu sau khi transaction thứ hai được hoàn tất.

▪ **Durability**: Đặc tính này thể hiện rằng bất kỳ thay đổi trong dữ liệu bởi một transaction đã hoàn tất giữ nguyên ảnh hưởng trong hệ thống. Vì vậy, bất kỳ sự thay đổi bởi một transaction hoàn tất vẫn còn thậm chí trong sự kiện hệ thống bị fail. Đặc tính này được đảm bảo bởi sự dự phòng và phục hồi transaction log.

Ba loại Transaction:

▪ **Implicip Transactions (Transaction ngầm định):** Khi một connection đang mở trong chế độ implicip, SQL Server bắt đầu một transaction mới một cách tự động sau khi transaction hiện hành hoàn tất hoặc Roll back. Bạn không cần phát họa bắt đầu một transaction; bạn chỉ cần commit hoặc Rollback mỗi transaction. Chế độ Implicit transaction phát sinh một chuỗi các transaction liên tục

Sau khi chế độ transactin implicit đã được bật ON cho một kết nối, SQL Server tự động bắt đầu một transaction khi nó thực thi bất kỳ các lệnh sau: Alter Table, Creat, Delete, Drop, Fetch, Grant, Insert, Open, Revoke, Select, Trucate Table, Update.

▪ **Explicip Transactions: (Transaction tường minh):** Là một transaction mà chúng ta phải định nghĩa bắt đầu một transaction (Begin transaction) và kết thúc một transaction (Commit Transaction)

Bắt đầu một transaction

BEGIN TRAN [SACTION] [transaction_name | @tran_name_variable]

Hoàn Tất Transaction

COMMIT [TRAN [SACTION] [*transaction_name* | *@tran_name_variable*]]

Lưu vị trí Transaction

SAVE TRAN [SACTION] { savepoint_name | @savepoint_variable }

Hũy một Transaction

ROLLBACK [TRAN [SACTION]
[transaction_name | @tran_name_variable
| savepoint_name | @savepoint_variable]]

@@TRANCOUNT: Trả về số thứ tự mà trasaction được mở, tối đa lồng 32 cấp, không nên lồng.

▪ **Distributed Transaction:** là một loại explicip Trangsaction nhưng giao tác của nó liên quan nhiều server. Sự quản lý phải được kết hợp giữa các nhà quản lý tài nguyên của các server và điều này gọi là transaction manager. Các transaction trong một server những tham chiếu từ nhiều database, thực ra cung là một distributed transaction.

Transaction log là một tranction dùng lock để ngăn chặn người dùng hiệu chỉnh dữ liệu ảnh hưởng từ các transaction chưa hoàn tất.

Công dụng transaction log

▪ **Phục hồi các transaction đặc biệt**: Khi một application đưa ra lệnh ROLL BACK hoặc SQL Server nhận ra một lỗi, thì bảng ghi log được dùng để roll back bất kỳ hiệu chỉnh trong suốt quá trình của transaction chưa hoàn tất.

▪ **Phục hồi tất cả các transaction chưa hoàn tất khi Sql server được bắt đầu:**

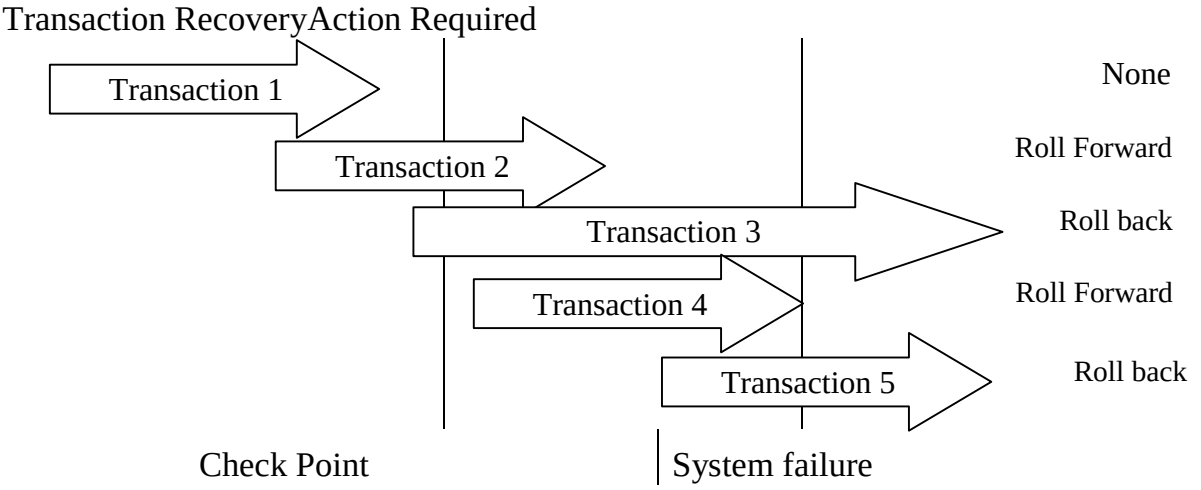
▪ **Hoàn trả lại database lại đến một thời điểm bị lỗi:** Nhằm đảm bảo không phát sinh mâu thuẩn sau khi có sự cố.

Write-ahead Transaction Log: Dùng Write – ahead log đảm bảo rằng không có dữ liệu hiệu chỉnh nào được cất vào đĩa trước khi cất trong log record.

Check point : Là một hành động thực hiện định kỳ trong CSDL, nó sẽ ghi lại tất cả các transaction liên quan đến data lên transaction log, nhằm phục vụ cho việc recovery data.

Check point xãy ra khi:

- Gặp câu lệnh Check point.
- Có sự hiệu chỉnh trên CSDL
- Trước ngay khi SQL Server Shutdow
- Áp định định kỳ.



3.2 LOCK

Lock: là cơ cấu ngăn chặn các xung đột do các user không thể đọc hoặc hiệu chỉnh các dữ liệu mà các dữ liệu này hiện đang trong một tiến trình xử lý khác. Tuy nhiên, bạn vẫn có thể thao tác trên những đối tượng còn phụ thuộc vào chuyển tác mà user khác đang thực hiện. Khi đó hệ thống sẽ kiểm soát tiến trình của bạn có tương thích với quá trình trước đó hay không.

Các vấn đề đồng thời (Concurrency problem)

- **Lost Updates (cập nhật mất dữ liệu):** Lost updates xảy ra khi 2 hoặc nhiều transaction chọn cùng một dữ liệu và sau đó cập nhật dòng dựa trên giá trị cũ. Mỗi transaction không biết những transaction khác. Thao tác cập nhật cuối cùng ghi đè lên những thao tác cập nhật khác mà kết quả dẫn đến mật dữ liệu.

- **Uncommitted Dependency (Dirty Read – đọc dữ liệu sai)**: Uncommitted Dependency xảy ra khi transaction thứ 2 chọn một dòng mà đang sẳn sàng cập nhật bởi một transaction. Transaction thứ 2 đang sẳn sàn đọc dữ liệu mà chưa được hoàn tất và có thể bị thay đổi bởi transaction cập nhật

- **Inconsistent Analysis (Nonrepeatable Real – đọc hai lần mẫu tin)**: Xảy ra khi transaction thứ hai truy xuất cùng một dữ liệu với vài lần và đọc lên những dữ liệu khác nhau ở mỗi lần đọc. Inconsistent Analysis tương tự với Uncommitted Dependency trong trường hợp transaction thứ nhất đang hiệu chỉnh dữ liệu thì một trasaction thứ hai đọc dữ liệu. Tuy nhiên, dữ liệu đọc bởi transaction thứ hai đã được commited bởi transaction update

- **Phantom Reads (đọc các mẫu tin ma)**: Xảy ra khi hành động insert hoặc delete được thi hành trên một dòng dữ liệu mà nó thuộc vùng dự liệu đọc của một transaction khác.

Kiểu locks

- **Share locks:** được dùng cho những thao tác mà không làm thay đổi hay cập nhật dữ liệu (thao tác chỉ đọc), như là một câu select

- **Exclusive locks:** được dùng cho nhữ thao tác hiệu chỉnh dữ liệu, như là Insert, Update, hay Delete. Đảm bảo rằng nhiều cập nhật không thể được thực hiện trong cùng tài nguyên tại cùng một thời điểm.

- **Update locks:** được dùng trên những tài nguyên mà có thể được cập nhật. Ngăn chặn một dạng thông thường của deadlock mà xảy ra khi nhiều session đang độc, đang lock, và có khả năng cập nhật tài kuyên sau này.

- **Intent locks:** Dùng để thiết lập một lock kế thừa. Kiểu intent lock là : Intent shared (IS), Intent exclusive (IX), và share with intent exclusive (SIX)

- **Schema locks**: được dùng khi thao tác thùy thuộc vào giản đồ của table là đang thực thi. Kiểu schema locks là schema modification (Sch-M), schema stability (Sch-S)

- **Bulk Update (BU) locks**: Cho phép chia sẽ cho Bulk-copy thi hành.

Deadlock

Deadlock xảy ra khi có một sự phụ thuộc chu trình giữa hai hay nhiều luồng cho một tập hợp tài nguyên nguyên nào đó. SQL Server sẽ tự giải quyết trường hợp deadlock bằng cách RoolBack một trong các transaction, và ưu tiên rollback những transaction có thời gian ít hơn. Để giảm bớt deadlock, bạn nên:

- ✓ Truy xuất các object theo thứ tự.
- ✓ Tránh sự tướng tác người dùng trong thời gian transaction.
- ✓ Cố giữ transaction càng ngắn càng tốt.
- ✓ Dùng mức cô lập thấp nhất.
- ✓ Dùng giới hạn các connection.

BÀI 4:
SỬ DỤNG CURSORS ĐỂ TRUY XUẤT DỮ LIỆU

4.1 Khái niệm

Cursor là một đối tượng của CSDL mà nó hỗ trợ cho phép truy xuất và thao tác dữ liệu trong một tập kết quả (result set). Sau khi cursor được định vị trên một dòng, các hoạt động có thể thực hiện trên dòng đó hoặc khối các dòng bắt đầu từ vị trí đó.

Dùng Cursor để:

- Định vị một dòng đặc biệt trong tập kết quả.
- Truy xuất một hoặc khối dòng bắt đầu từ vị trí cursor trong tập kết quả.
- Cung cấp thao tác hiệu chỉnh dữ liệu cho các dòng tại vị trí của cursor trong tập kết quả.
- Cung cấp các mức độ khác nhau của tính tường minh trong dự thay đổi được tạo bởi những người dùng khác đến tập kết quả.
- Cung cấp việc truy cập dữ liệu trong tập kết quả cho các câu lệnh T-SQL trong scripts, Stored procedure, và triggers.

Các thao tác cần thực hiện trong khi sử dụng cursor trong SQL Server:

- Cursor cần phải khai báo và các thuộc tính của nó cũng cần được xác định.
- Mở cursor.
- Phải lấy (fetch) các dòng cần thiết từ cursor.
- Dữ liệu trong dòng hiện hành có thể được hiệu chỉnh nếu cần thiết.
- Tạm thời không dùng cursor thì phải đóng cursor lại.
- Cursor cần phải được giải phóng (deallocate) khi không cần dùng nữa.

Cursor:

- **T-SQL Server cursors**: Cursor này được dựa trên câu lệnh khái báo cursor, nó được dùng chủ yếu trong các script, strore procedure, triggers. Nó được thi hành trên server và được quản lý bởi các câu lệnh T0SQL gửi từ client đến server. Khi làm việc với cursor thì phải khai báo, mở, truy xuất, xử lý, đóng, giải phóng.
- **API Server cursors (API-Application Program interface)**: Nó được thực thi trên server và được quản lý bởi một hàm cursor API. API Server cursors được cung cấp bởi hàm cursor API trong OLE DB, ODBC, và DB-Library. Mỗi lần một ứng dụng của client gọi một hàm cursor API, SQL server OLE DB provider, ODBC driver, hoặc DB-Library DLL gửi các yêu cầu đến server cho hành động ứng với các hàm cursor API. Nó chỉ khác với T-SQL cursor ở syntax, còn về bản chất tương tự.
- **Client Cursors**: SQL server ODBC driver, DB-Library DLL, và ADO API DLL giúp thi hành cursor client một cách nội tại. cursor client được thi hành bằng cách nắm giữ tập kết quả dữ liệu của clients. Mỗi lần Application của client gọi hàm cursor API, thì SQL server OLE DB provider, ODBC driver, hoặc DB-Library DLL thực thi tính toán ngay trên tập kết quả dữ liệu được giữ trên client. Ta chỉ dùng client cursor để làm giảm bớt sự giới hạn mà server consor không được cung cấp các câu lênh T-SQL. Nếu con trỏ static hoặc Scroll thì ta có thể dùng client cursor.

Loại cursor

Static

- Tập kết quả của contrỏ loại Static được xây dựng trong tempdb khi con trỏ được mở.
- Một static con trỏ luôn luôn hiện tập kết quả giống như tập kết quả có được ngay sau khi con trỏ mở.
- Con trỏ không phản ánh được bất kỳ sự thay đổi nào trong database ngay cả khi những dòng dữ liệu có thay đổi, các dòng mới được insert bởi các transaction khác cũng vẫn không hiện lên mặc dù chúng thỏa điều kiện lọc dữ liệu.
- Thao tác Insert, Update, Delete đều không có tác dụng khi dùng static cursor

Keyset-driven:

- Thành viên và thứ tự của các dòng trong một keyset-driven cursor là cố định khi cursor được mở. Con trỏ được điều khiển bới một tập giá trị nhận dạng goi là Keyset. Keyset được xây dựng từ một tập các cột mà dùng để nhận dạng các dòng trong tập kết quả. Keyset được xây dựng trong Tempdb khi con trỏ được mở
- Cho phép hiệu chỉnh (update, delete) dữ liệu trên cột không là keyset (bởi chủ cursor hoặc từ user khác) khi user duyệt thông qua con trỏ.
- Có thể thêm (insert) vào bảng nếu như cursor thể chèn dữ liệu vào bảng.

Dynamic: Trái ngược với static cursor.

- Dynamic cursors phản ánh được toàn bộ sự thay đổi của các dòng dữ liệu trong tập kết quả khi duyệt con trỏ.
- Giá trị dữ liệu, thứ tự, và thành viên của các dòng trong tập kết quả có thể thay đổi ứng với mỗi lần duyệt con trỏ.
- Tất cả các lệnh Insert, Update, Delete của các user đều hữu hiệu thông qua con trỏ.
- Sự Update hữu hiệu ngay tức thời nếu chúng được update thông qua qua con trỏ ứng với tại mẫu tin hiện thời, còn nếu update bên ngoài con trỏ thì nó không hữu hiệu cho đến khi nó hoàn tất.

Fast Forward only: Tương tự như Dynamic cursor nhưng nó chỉ có thể duyệt con trỏ thào một chiiều từ First đến Last

Cusrsor type	**Membership**	**Order**	**Values**
Forward-only	Dynamic	Dynamic	Dynamic
Static	Fixed	Fixed	Fixed
Dynamic	Dynamic	Dynamic	Dynamic
Keyset-driven	Fixed	Fixed	Dynamic

4.2 Làm việc với T-SQL server cursors

Khai báo cursor

Bằng câu lệnh declare ở sau từ khóa AS trong stored procedures hoặc functions

DECLARE *cursor_name* CURSOR
[LOCAL | GLOBAL]
[FORWARD_ONLY | SCROLL]
[STATIC | KEYSET | DYNAMIC | FAST_FORWARD]
[READ_ONLY | SCROLL_LOCKS | OPTIMISTIC]
[TYPE_WARNING]
FOR *select_statement*
[FOR UPDATE [OF *column_name* [,...*n*]]]

- *select_statement:* là câu lệnh truy vấn để định nghĩa tập kết quả của cursor. Từ khóa COMPUTE, COMPUTE BY, FOR BROWSE, and INTO Không cho phép trong *select_statement* này.
- READ ONLY Không cho phép Update trong cursor này.
- UPDATE [OF column_name [,...n]]: Quy định cột cho phép được update khi dùng cursor. Nếu OF column_name [,...n] được chỉ định rõ ràng thì chỉ có các cột được chỉ định mới được cho phép hiệu chỉnh, nếu không có column list, all columns có thể update.

Mở cursor

Mở cursor trước khi dùng (Cursor phải được khai báo rồi)

OPEN {cursor_name }

Lấy mẫu tin hoặc điều hướng cursor (FETCH)

Truy xuất từng dòng dữ liệu. Kiểm tra phạm vi con trỏ bằng **@@Fetch_status**

FETCH [[NEXT | PRIOR | FIRST | LAST | ABSOLUTE n | RELATIVE n]]

FROM cursor_name[INTO @variable_name [,...n]]

@@Fetch_status: Trả về giá trị 0 hoặc 1. Trả về **1** có nghĩa là con trỏ đã được dời đến quá cuối tập kết quả. Trả về **0** vẫn còn trong phạm vi của tập kết quả. Khi đó dùng vòng lặp While để duyệt cả tập kết quả của Cursor.

Xử lý dữ liệu:

Có thể dùng các câu lệnh Upadte hoặc Delete để hiệu chỉnh dữ liệu

- Sử dụng dữ liệu của mẫu tin hiện hành: dữ liệu được lấy lên và gán cho các biến tương tứng trong câu lệnh Fetch

- Cập nhật dữ liệu thông qua cursor: thực chất là dữ liệu được hiệu chỉnh trực tiếp vào trong bảng

Cập nhật giá trị cho cột

UPDATE <Ten Table>

SET <TenColumn>= <Value> [,..n]

WHERE CURRENT OF <Cur_Name>

Xoá dữ liệu thông qua cursor: Thực sự là dữ liệu xóa trên bảng

DELETE <Ten Table>

WHERE CURRENT OF <Cur_Name>

Đóng Cursor

Kết thúc hành động của cursor cho lần mở (open), nó vẫn hiện hữu cho đến khi gặp một lệnh Open khác hoặc gặp lệnh Close cursor.

CLOSE cursor_name

Giải phóng Cursor

Giải phóng cursor, huy bỏ tham chiếu đến con trỏ từ session hiện hành. Tiến trình này làm cho tài nguyên trở về trạng thái sẵn sàng truy xuất.

DEALLOCATE cursor_name

4.3 Ví dụ.

Giả sử người quản lý cần một bảng báo cáo lịch sử khách hàng theo dạng sau:

Customer:ALFKI - Alfreds Futterkiste

Order:10643 (Aug 25 1997)

Order:10692 (Oct 3 1999)

Order:10702 (Oct 13 1999)

Customer:ANATR - Ana Trujillo Emparedados y helados

Order:10625 (Aug 8 1997)

Order:10759 (Nov 28 1999)

Customer:ANTON - Antonio Moreno Taquería

Order:10507 (Apr 15 1997)

Order:10535 (May 13 1999)

Order:10573 (Jun 19 1999)

Order:10677 (Sep 22 1999)

Order:10682 (Sep 25 1999)

Đoạn Batch thực hiện báo cáo như sau:

```
--- Khai báo contrỏ
DECLARE rpt CURSOR FOR
SELECT c.CustomerID, c.CompanyName, o.OrderID,
o.OrderDate
FROM Customers c, Orders o
WHERE c.CustomerID = o.CustomerID AND c.CustomerID
LIKE 'A%' AND DatePart( year, o.OrderDate) = 1997

DECLARE @cid char( 8), @cname char( 40),
@ordid char( 8), @orddt datetime, @old char( 8)
--- Mở contrỏ
OPEN rpt
--- Lấy dữ liệu của mẫu tin đầu tiên vào các biến
FETCH NEXT FROM rpt INTO @cid, @cname, @ordid, @orddt
SELECT @old = ’ ‘

WHILE @@fetch_status = 0
BEGIN
     IF @old = @cid
          BEGIN
          PRINT ' Order:' + rtrim( @ordid) + ' (' +
     cast(@orddt as CHAR( 10)) + ')'
         END
     ELSE
         BEGIN
          PRINT 'Customer:' + rtrim( @cid) + ' - ' +
     rtrim(@cname)
          PRINT ' Order:' + rtrim( @ordid) + ' (' +
     cast(@orddt as CHAR( 11)) + ')'
          SELECT @old = @cid
         END
     FETCH NEXT FROM rpt INTO @cid, @cname, @ordid,
@orddt
END
--- Đóng con trỏ
```

```
CLOSE rpt
--- Giải phóng con trỏ
DEALLOCATE rpt
```

Ví dụ 2:

```
DECLARE MyCursor CURSOR FOR
SELECT c.CustomerID,c.Companyname,c.contactname,
     o.OrderID,o.OrderDate
FROM Customers c, Orders o WHERE c.CustomerID =
o.CustomerID
FOR UPDATE
OPEN MyCursor
DECLARE @cid VARCHAR( 8), @c VARCHAR( 80), @o INT,
     @od DATETIME, @cn VARCHAR( 80)
FETCH NEXT FROM MyCursor INTO @cid, @c, @cn, @o, @od
SELECT @cid
BEGIN TRANSACTION
UPDATE Customers SET CompanyName = 'q'
WHERE CURRENT OF Mycursor
DEALLOCATE MyCursor
SELECT *  FROM Customers
ROLLBACK TRANSACTION
```

BÀI 5:
BẪY LỖI - TRIGGER

5.1 Giới thiệu về trigger

Chất lượng của một CSDL được đánh giá một phần bởi tính nhất quán và độ chính xác của dữ liệu trong CSDL. Để đảm bảo tính toàn vẹn dữ liệu ta có nhiều phương pháp, trigger là một phương pháp hữu hiệu.

- Trigger là một loại stored procedure đặc biệt, nó được định nghĩa để tự động thực thi khi có một câu lệnh Update, Insert, hoặc Delete được phát ra trên bảng hoặc View.
- Trigger là một công cụ mạnh mà nó có thể dùng để ràng buộc các qui tắc quản lý một các tự động khi dữ liệu bị hiệu chỉnh.
- Trigger cũng có thể nới rộng các tính toàn vẹn kiểm soát logic của SQL Server.
- Trigger tự động thực thi, không thể gọi một trigger thi hành một cách trực tiếp.

Dùng trigger khi:

- Ràng buộc toàn vẹn dữ liệu cho phù hợp với mô hình quan hệ CSDL.
- Kiểm soát dữ liệu hiện tại khi có thay đổi đến giá trị trong mẫu tin trong bảng.
- Kiểm tra dữ liệu nhập vào phù hợp với mối quan hệ dữ liệu giữa các bảng.
- Định nghĩa thông báo lỗi của người dùng.
- So sánh trạng thái của dữ liệu trước và sau hiệu chỉnh

Đặc điểm và giới hạn của trigger

- Trigger (After trigger) là Reactive; constraints và instead of trigger là proactive. (Reactive: khi delete/insert một dòng vào table, thì sau khi insert thì trigger mới được tự động thực thi thì gọi là reactive, proactive là kiểm tra trước khi Insert/Delete)
- Các constraint được kiểm tra trước, sau đó mới tới trigger.
- Các bảng có thể có nhiều trigger cho bất kỳ hành động nào. Tuy nhiên không nên dùng quá nhiều trigger trong cùng một bảng. SQl Server chỉ cho phép chỉ định trigger nào thi hành đầu tiên, thi hành cuối cùng, còn các trigger khác thứ tự thi hành không xác định. Vì vậy, nếu có quá nhiều trigger trên 1 đối tượng có thể sẽ gặp nhiều rắt rối khi có nhiều trigger không xác định thứ tự.
- Không thể tạo trigger trên các đối tượng ở temporary.
- Nên thiết kế trigger không trả về tập kết quả nhằm đảm bảo tính chất chuyển tác giữa các user và lập trình.
- Các trigger có thể xử lý hành động trên nhiều dòng.
- Trigger không ngăn ngừa thay đổi cấu trúc, trigger chỉ quan tâm đến sự thay đổi dữ liệu trong bảng. Khi bạn xóa đối tượng trong CSDL với các bước hợp lý, SQL Server sẽ cho phép xóa đối tượng đó và trigger không thể kiểm soát được.
- **Trigger Events:** Có 3 biến cố mà trigger sẽ tự động thực thi khi biến cố xảy ra, đó là **Insert, Update, Delete**. Trigger không thể được gọi một cách trực tiếp.

Cơ cấu thực thi trigger:

Khi insert hoặc update dữ liệu của bảng bật trigger, trigger sẽ lưu trữ dòng dữ liệu mới hoặc dòng dữ liệu đã hiệu chỉnh vào một bảng có tên là **Inserted** trong bộ nhớ cash, khi xóa dữ liệu của bảng bật trigger lên, trigger sẽ lưu trữ dòng dữ liệu bị xóa vào bảng có tên là **Deleted** trong bộ nhớ cash. Các bảng tồn tại trong bộ nhớ và được truy vấn bởi các lệnh T-SQL trong các trigger. Bạn có thể sử dụng thông tin trong bảng inserted và Deleted để so sánh, lưu trữ, rollback,… nếu cần, khi đó bạn không cần tạo ra các biến để lưu trữ thông tin và tốc độ truy xuất nhanh.

Hai loại trigger trong SQL Server 2000: INSTEAD OF và AFTER (nếu chỉ nói trigger có nghĩa là nói đến AFTER Trigger)

- **FOR triggers và AFTER triggers:** các trigger này chỉ được thực thi khi tất cả các thao tác Insert, Update hay Delete thực hiện xong. Tất cả các hành động tham chiếu và kiểm tra constraint cũng phải được thực hiện xong trước khi trigger thi hành. Loại trigger này chỉ cài đặt được trên bảng, không cài đặt được trên View.

Khi tạo trigger và không chỉ định rõ thì mặc định là AFTER, FOR chỉ là từ khóa tương thích ngược với các phiên bản trước của SQL Server.

- **INSTEAD OF triggers:** Trigger này chỉ có trong SQL Server 2000. Trigger này sẽ thi hành thay cho các câu lệnh Insert, Delete, Update. Như vậy khi tạo trigger kiểu này bạn phải viết lại các lệnh insert, Delete, Update đối với dữ liệu. Có thể áp dụng cho cả bảng và View, tuy nhiên nó không cho phép áp dụng với các view có lựa chọn WITH CHECK OPTION.

Nested trigger: có nghĩa là bảng Table1 có trigger, Table2 có trigger khác. Nếu ta thao tác trên Table1 thì trigger của nó sẽ thực thi, nếu thao tác này có liên quan đến Table2 thì trigger2 ở bảng Table2 thực thi. Gọi là lồng các trigger, bạn có thể lồng tối đa là 32 cấp.

5.2 Tạo và quản lý các trigger

Một trigger có thể tạo và quản lý bằng cách sử dụng Query Analyzer hoặc Enterprice Manager. Tạo một trigger trên đối tượng thì phải có quyền Owner đối với đối tượng.

5.2.1 Tạo trigger

```
CREATE TRIGGER trigger_name
ON { table | view }
[ WITH ENCRYPTION ]
{
  { { FOR | AFTER | INSTEAD OF } { [DELETE] [,] [ INSERT ] [, ] [ UPDATE ]
}
    [ WITH APPEND ]
    [ NOT FOR REPLICATION ]
    AS
    sql_statement [ ...n ]
  }
}
```

Giải thích:

- *trigger_name* : Tên của trigger. Nếu trong 1 bảng có nhiều trigger thì tên của các trigger phải là duy nhất.

- ON { *table* | *view* }: Chỉ định table/View được áp dụng trigger, chỉ có instead of được áp dụng cho cả view và table.

- [WITH ENCRYPTION]: Trigger được mã hóa. Thông tin mã hóa năm trong bảng syscomment

- **{** FOR | AFTER | INSTEAD OF **}:** Xác định loại trigger cho thao tác DELETE, INSERT, UPDATE.

Delete trigger: trigger sẽ được thực thi khi có mẫu tin bị xóa khỏi bảng, SQL Server tạo ra bảng mang tên DELETED để cất mẫu tin bị xóa, trong trigger ta có thể tham khảo đến mẫu tin này.

Insert trigger: trigger sẽ được thực thi khi có mẫu tin chèn vào bảng, SQL server tạo ra bảng mang tên INSERTED để cất mẫu tin chèn, trong trigger ta có thể tham khảo đến mẫu tin này.

Update trigger: Mỗi khi có mẫu tin nào đó được cập nhật, giá trị những cột có liên quan đến trigger sẽ được kiểm tra trước khi cập nhật. Mẫu tin bị cập nhật sẽ được sao lưu trong bảng Inserted (chứa giá trị mới) và Deleted (chứa giá trị cũ).

- [NOT FOR REPLICATION]: Trigger sẽ không thực hiện khi bảng có liên quan đến kỹ thuật sao chép nhân bản (relication)

Lưu ý:

- Một hành động (Insert hoặc Delete hoặc Update) có thể kích hoạt cùng lúc nhiều trigger khác nhau.
- Trigger sẽ thi hành cho dù thao tác của người sử dụng có tác động thực sự trên các mẫu tin hay không, do đó dùng @@ROWCOUNT để kiểm tra trước khi cho các hành động trong trigger thi hành.
- Các phát biểu sau không thể có trong trigger: các phát biểu Create, Drop, Alter Table, Alter DataBase, Truncate Table, Grant/Revoke, Reconfigure, Load DataBase, Transaction, Update statistics, Select Into, Disk.

5.2.2 Quản lý trigger

Alter Trigger [1] Hiệu chỉnh trigger

Drop Trigger : Xóa trigger

Sp_rename : đổi tên.

Sp_helptrigger, Sp_heltext: Xem code trigger

DISABLE TRIGGER/ ENABLE TRIGGER trong câu lệnh Alter Table

5.3 Vài ví dụ về trigger.

[1] Tham thảo cú pháp lệnh trong Books-Online

Ví dụ 1: Viết một trigger cho thao tác Insert, để kiểm tra ngày lập hoá đơn thì luôn luôn lớn hơn ngày giao.

```
CREATE TRIGGER Trg_NgayLap_NgayGiaoHD
ON Hoadon AFTER INSERT
AS
DECLARE @NgayLapHD DateTime, @NgayGiao DateTime
SELECT
@NgayLapHD=hd.NGayLapHD,NgayGiao=hd.NgayGiaoNhan
FROM HoaDon hd INNER JOIN Inserted i ON
hd.MaHD=i.Mahd
If @NgayGiao<@NgayLapHD
     BEGIN
          RAISERROR(500103,10,1)
          ROLLBACK TRANSACTION
     END
--------------------------------
INSERT HoaDon VALUES (1003,'1/1/2004','N','TP.
HCM’,111,‘12/24/2003‘)
```

Ví dụ 2: Tạo một trigger có tên là Msg_InstUpd_Kh, trigger này thực hiện chi 2 thao tác Insert, Update của bảng KhachHang. Trigger sẽ thông báo “Có n dong da duoc hieu chinh”.

```
Use SalesDB
GO
CREATE TRIGGER Msg_InstUpd_Kh
On KhachHang
FOR INSERT, UPDATE
AS
RAISEORROR(‘”Có %d dòng đã được hiệu chỉnh””,
0,1,@@Rowcount)
RETURN
```

Kiểm chứng: bằng cách chèn hoặc cập nhật 1 hoặc nhiều mẫu tin vào bảng KhachHang.

```
INSERT INTO KhachHang …..
```

Ví dụ 3: Hai bảng HoaDon và CT_HoaDon có quan hệ 1-n, tức là khi thêm một chi tiết hóa đơn trong bảng chi tiết hoá đơn thì hóa đơn này phải đã được phát sinh trong bảng hóa đơn. Vì thế ta phải viết một trigger Insert cho bảng CT_HoaDon. Nếu ta có trigger này thì khi ta vi phạm SQL Server cũng sẽ báo lỗi vì vi phạm ràng buộc toàn vẹn khóa ngoại. Tuy nhiên ta hãy viết một trigger để thực hiện thông bao khi có lỗi này xuất hiện.

```
Use SalesDB
GO
CREATE TRIGGER Trigger_Ins_CT_HD
ON HoaDon
FOR INSERT
/* Insert trigger cho bảng hóa đơn*/
AS
IF NOT EXISTS
(Select * From Inserted I inner join HoaDon hd ON
i.Mahd = hd.Mahd)
/* Nếu Mahd được chèn vào bảng CT_HoaDon không tồn
tại trong bảng hóa đơn thì không chèn được */
BEGIN
RAISERROR(60000,16,1,'MaHd', 'CT_HoaDon','Mahd','Hoa
Don')
     ROLLBACK
END
```

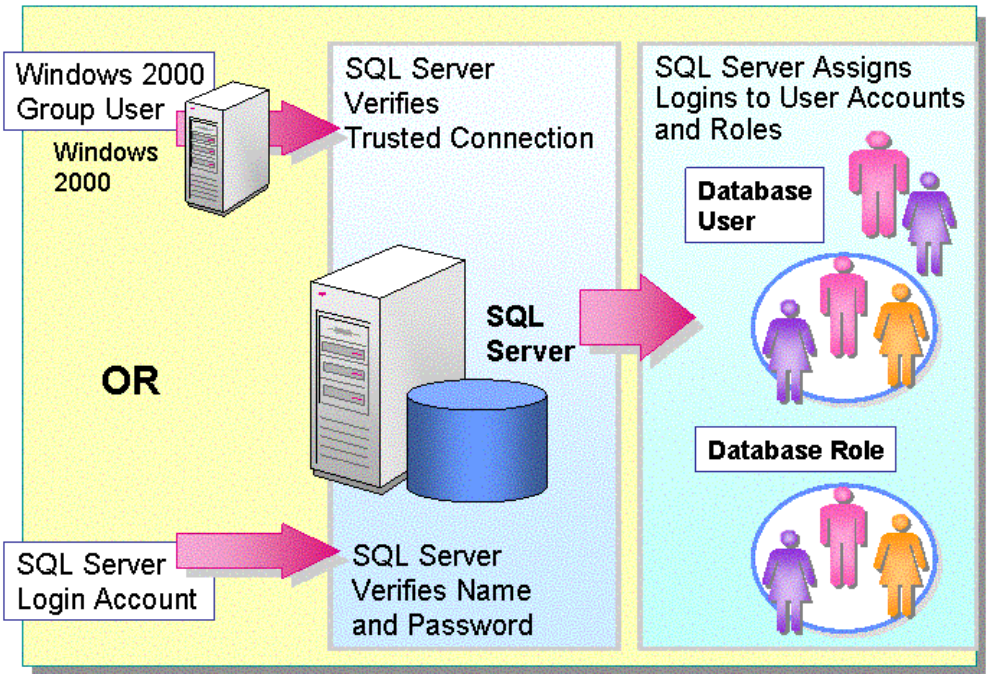
Ví dụ 4: Price trong CT_HoaDon luôn luôn lớn hơn hay bằng GiaGoc trong bảng sản phẩm. Để bắt ràng buộc này bạn thường dùng Check Constraints, tuy nhiên bạn cũng có thể bẩy lỗi bằng trigger

```
Create Trigger MGT_Gia
ON CT_HoaDon
FOR UPDATE
AS
IF EXISTS (Select * From INSERTED I where i.price <
tblItem.OriginalPrice)
     Begin
     RAISERROR(‘Khong thể cập nhật vì giá không hợp
lệ’,16,1)
     ROLLBACK TRAN
     End
```

BÀI 6:
BẢO MẬT TRONG SQL SERVER

6.1 Khái niệm về bảo mật.

6.1.1 Mô hình truy cập bảo mật của SQL Server.



Hình 1: Các lớp kiểm tra bảo mật chứng thực của SQL Server

Việc kết nối đến SQL Server 2000 thật đơn giản. Sự bảo mật được kiểm tra ở ba nơi khác nhau: có thể bị kiểm hợp lệ bởi Windows 2000, bản thân SQL Server, mức CSDL riêng lẽ. Ngay sau khi bạn kết nối vào SQL Server bạn chưa thật sự truy cập được bất kỳ một đối tượng CSDL nào, bạn cần phải được cấp quyền (permissions) truy cập đến đối tượng.

6.1.2 Các chế độ bảo mật.

SQL Server 2000 cung cấp hai chế độ bảo mật:

<u>Hai loại chứng thực</u>

Windows Authentication

SQL Server kiểm tra nhận dạng của user và sau đó cho phép hay từ chối đăng nhập truy xuất dựa trên cơ sở tên của User mà không cần tên đăng nhập và password riêng biệt. Điều này gọi là kết nối tin tưởng. Khi bạn kết nối đến SQL Server theo cách này thì có nghĩa là bạn trình bày với SQL Server một ủy nhiệm bảo mật của Windows (như là một thẻ bài truy cập của bạn). Bạn xây dựng các ủy nhiệm này trong quá trình đăng nhập vào mạng windows 2000. Các ủy nhiệm bảo mật này được truyền âm thầm cho bạn, vì thế bạn không cần làm bất cứ điều gì đặc biệt để vượt qua việc kiểm tra bảo mật.

SQL Server Authentication

Người quản trị CSDL có thể tạo ra các tài khoản và password đăng nhập SQL Serevr. Các tài khoản này hoàn toàn không tuỳ thuộc vào các tài khoản hay nhóm người dùng hệ điều hành. Nếu có một kết nối chỉ định chứng thực SQL Server thì SQL Server 2000 thực thi chứng thực chính nó bằng cách kiểm tra xem tài khoản đang nhập có tồn tại hay không và mật khẩu chỉ định có khớp với một ghi nhận trước đây trong SQL Server 2000 không.

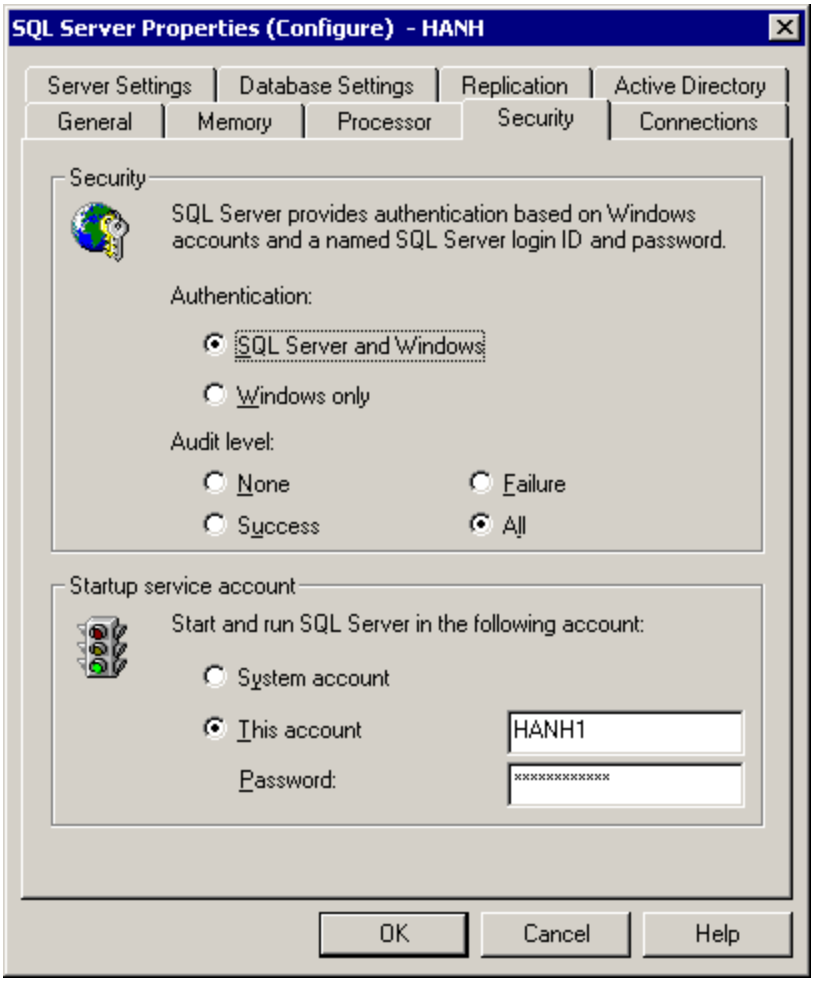
Chế độ chứng thực:

Windows Authentication mode: Người sử dụng chỉ có thể kết nối với SQL Server 200 bằng Windows Authentication (Kết nối tin tưởng)

Mixed mode: Người dùng có thể kết nối với SQL Server 200 bằng cách dùng cả Windows Authentication và SQL Server Authentication

Chuyển đổi chế độ chứng thực:

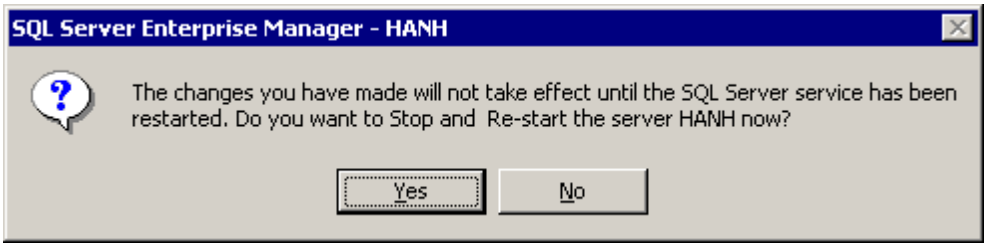
Sau khi cài đặt, bạn có thể sử dụng Enterprise Manager để chuyển đổi qua lại giữa các chế độ. Tại cửa sổ Enterprise Manager, nhắp nút phải chuột tại instance và chọn Properties



Hình 2: cấu hình Security cho SQL Server

Ở trang Security click chọn hoặc SQL Server And Windows hoặc Windows Only để đổi chế độ chứng thực.

Sau khi bạn chuyển chế độ, bạn phải stop và sau đó restart dịch vụ của SQL Server service để sự thay đổi này có tác dụng. Enterprise Manager sẽ hỏi ý kiến bạn.



Hình 3: Hộp thoại hỏi ý kiến người dùng có muốn Stop và Restart Server

6.1.3 Tìm hiểu các Server-Wide Permission.

SQL Server 2000 hỗ trợ một số server role được định nghĩa trước được kết hợp với các quyền quản trị. Các server role này cấp các quyền server-wide để thực hiện các tác vụ khác nhau và bao gồm cac quyền mà bạn có thể cấp cho những người dùng thông qua việc sử dụng các server role này. Bạn không thể xoá các server role và không thể thay đổi các quyền của chúng. Để cấp các quyền cho một người dùng, thì bạn thêm đăng nhập của chúng server role. Với Transact-SQL, bạn có thể thêm các người dùng hay nhóm các người dùng vào một server role

Lưu ý: Các Server role về cơ bản giống như các group trong Windows 2000. Trong SQL Server 2000 có 8 server role

Server Role	Thành viên của Server Role có thể …
sysadmin	Thực thi bất cứ thao tác nào trong một thể hiện SQL Server 2000 và trong bất kỳ CSDL. Mặc nhiên, tất cả các thành viên của nhóm Windows built-in Administrators, tài khoản người dùng sa thuộc vào server role này.
serveradmin	Cấu hình SQL Server 2000 bằng cách dùng thủ tục hệ thống sp_configure và có thể kết thúc các sevice. Các thành viên của nhóm điều hành viên built-in của Windows là rất tốt để nhận server role này.
setupadmin	Cài đặt và cấu hình linked server, remote server, và replication. Có thể chỉ định một stored procedure được thực thi lúc khởi động (startup), như là sp_serveroption. Các thành viên của nhóm điều hành viên built-in của windows là rất tốt nhận server role này.
securityadmin	Thực hiện tất cả các thao tác liên quan đến security trong SQL Server 2000, kể cả quản lý các quyền câu lệnh CREATE DATABASE, điều khiển server logins, và đọc error log. Giúp đỡ ở các nhân viên văn phòng. Thành viên của nhóm điều hành viên built-in của windows là rất tốt nhận server role này.
processadmin	Quản lý các tiến trình chạy các instance của SQL Server. Có thể ngắt (kill) tiến trình của các user, các truy vấn.
dbcreator	Có thể tạo, hiệu chỉnh, và xóa các CSDL. Những nhà quản trị CSDL lâu năm đảm trách server role này tốt.

Server Role	Thành viên của Server Role có thể …
Diskadmin	Có thể quản trị các tập và các thiết bị dự phòng. Nói chung Role này dùng để tương thích ngược với SQL Server 6.x.
bulkadmin	Có thể thực hiện các câu lệnh BULK INSERT. Cho phép các thành viên của sysadmin server role làm đại diện các tác vụ BULK INSERTmà không cần gán các quyền sysadmin. Hãy cẩn thận bởi vì các thành viên cũng phải truy xuất đọc đến bất kỳ dữ liệu được chèn và quyền INSERT trên bất kỳ bảng mà dữ liệu sẽ được chèn.

Lưu ý: Một thành viên của bất kỳ server role nào đều có thể thêm các user khác có server role đó.

6.1.4 Tìm hiểu các quyền (Permission) chỉ định trên cở sở dữ liệu.

Khi Truy xuất đến SQL Server 2000 thì bạn chưa có quyền truy xuất đến các CSDL. Ngoại trừ các thành viên trong sysadmin role, thành viên trong một server role có sẵn quyền truy xuất CSDL. Các quyền truy xuất CSDL phải được cấp một cách rõ ràng bởi một system administrator hoặc thành viên của administrator role trong CSDL. Các quyền có thể được cấp (grant), từ chói (deny), cởi bỏ (revoke) và bao gồm các quyền tạo đối tượng, quản trị CSDL, thực thi các câu lệnh T-SQL, chèn dữ liệu vào bảng, xem dữ liệu bằng view. SQL Server 2000 có một số cơ chế để cấp các quyền cụ thể cho các user trong một CSDL.

Các quyền cụ thể trên CSDL

Quyền	Mô tả
Database owner	User có thể được chỉ định như là chủ (owner) của CSDL và có thể thực hiện bất kỳ hành động liên quan đến CSDL.
DBO role	Tất cả các thành viên của sysadmin server role thì tự động là thành viên của dbo role trong mỗi CSDL, và có thể thực hiện bất kỳ thao tác liên quan đến CSDL.
User	Các user và group có thể được cấp user truy xuất đến CSDL theo tài khoản bảo mật của Windows 2000 hoặc SQL Server 2000. Sau đó một giấy phép người dùng CSDL được cấp các quyền trong CSDL thông qua database role, role chung, và chỉ định cấp các quyền câu lệnh và đối tượng.
Guest user	Một user mà có thể truy xuất đến 1 instance của SQL Server 2000 (nhưng người này không có tài khoản truy người dùng để truy xuất đến CSDL cụ thể) có thể được cho phép truy xuất đến CSDL như là một người khách (guest user). Tài khoản guest có thể được cấp các quyền cụ thể trong CSDL (đề đọc dữ liệu). Theo mặc nhiên, một CSDL không có tài khoản guest user.
Public role	Tất cả các user được phép truy xuất đến CSDL trở thành thành viên của public role trong mỗi CSDL. Public role có thể được cấp các quyền cụ thể (Tổng quát các quyền cần thiết cho tất cả các user của CSDL).

Quyền	Mô tả
Fixed database role	Cho phép các user có thể được thêm vào các fixed database role trong một CSDL. Các Fixed database role chứa các quyền định trước trong CSDL để thi hành các hoạt động của database-wide.
User-defined database role	Cho phép các user có thể thêm vào user-defined database role trong một CSDL. Các role này có thể được tạo bở administrator và cấp một cách cụ thể các quyền đưa ra hoặc các quyền trong CSDL.
Statement permissions	Quyền thực thi các câu lệnh quản trị (như CREATE PROCEDURE) có thể được cấp, huỹ bỏ, từ chói các users, groups, và roles.
Object permissions	Quyền truy xuất đến các đối tượng CSDL (như là bảng hay view) có thể được cấp, huỹ bỏ, từ chối các users, groups, và roles.
Application role	Quyền thực thi các hành động trong một CSDL có thể được cấp cho một application, thậm chí cấp cho user. Một application kết nối đến một CSDL và kích hoạt application role. Các User truy xuất đến một CSDL thông qua sự kết nối này để dành lấy các quyền kết hớp với application role trong suốt quá trình kết nối. Các quyền phát hành đến một user cụ thể thì không liên quan khi user đang truy xuất CSDL thông qua application role.

6.1.5 Fixed Database Roles.

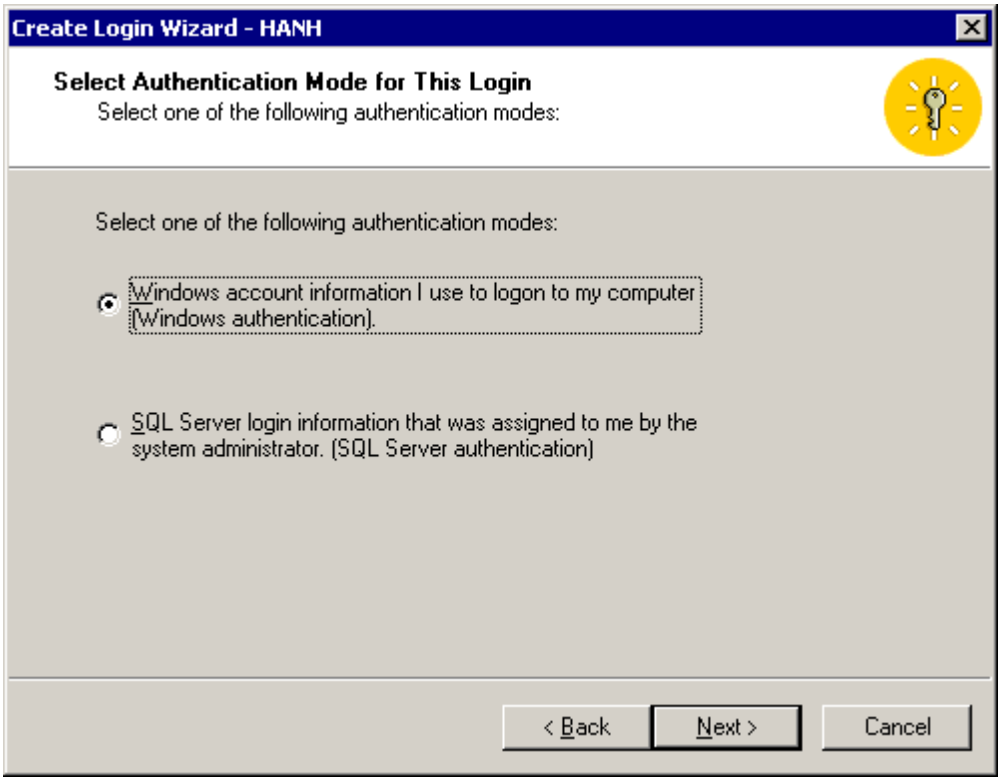
Mỗi CSDL có 9 role về CSDL được định nghĩa trước với các quyền kết hợp với database-wide để thực hiện các tác vụ khác nhau. Bạn không thể xóa những database role này và cũng không thể thay đổi các quyền của chúng. Để gán một người dùng các quyền này trong một CSDL, bạn thêm tài khoản của người dùng vào database role. Nếu những fixed database role này không gán tổng hợp các quyền mà bạn cần thì bạn có thể tạo các role với các quyền người dùng (thông thường nhiều quyền hạn chế).

9 Fixed database role có thể có trong SQL Server 2000.

Database Role	Thành viên của CSDL này có thể …
db_owner	Thực hiện bất kỳ tác vụ trong CSDL của SQL Server 2000. Các thành viên của role này có cùng quyền như là chủ của CSDL là các thành viên của dbo role.
db_accessadmin	Thêm hay xóa các user và group của Windows 2000 hoặc Win NT4.0 và các user trong một CSDL (dùng thủ tục hệ thống sp_grantdbaccess).
db_securityadmin	Quản lý tất cả các permission, role, role membership, và chuyển ower (ownership) trong một CSDL (sử dụng lệnh GRANT, REVOKE, và DENY).
db_ddladmin	Thêm, hiệu chỉnh, xóa các đối tượng trong CSDL (sử dụng lệnh CREATE, ALTER, và DROP).
db_backupoperator	Chạy các lệnh DBCC, phát hành checkpoint, và dự phòng CSDL (sử dụng các câu lệnh T-SQL: DBCC, CHECKPOINT, và BACKUP).
db_datareader	Đọc dữ liệu từ bất kỳ các bảng hoặc view của người dùng trong CSDL (bạn có quyền SELECT đối với tất cả table và view).
Db_datawriter	Hiệu chỉnh hoặc xóa dữ liệu từ các bảng hay view của người dùng trong CSDL (bạn phải có quyền INSERT, UPDATE, và DELETE đối với tất cả các table và view).
Db_denydatareader	Không đọc dữ liệu từ bất kỳ bảng trong CSDL (bạn không có quyền SELECT đối với bất kỳ đối tượng). Có thể permission on any objects). Có thể được sử dụng với role db_ddladmin để cho phép tạo các đối tượng làm chủ bằng dbo role, nhung không có thể đọc nhạy cảm chứa trong các đối tượng đó.
Db_denydatawriter	Không hiệu chỉnh hay xóa dữ liệu từ các bảng của người dùng trong CSDL (bạn không có quyền INSERT, UPDATE, và DELETE đối với các đối tượng)

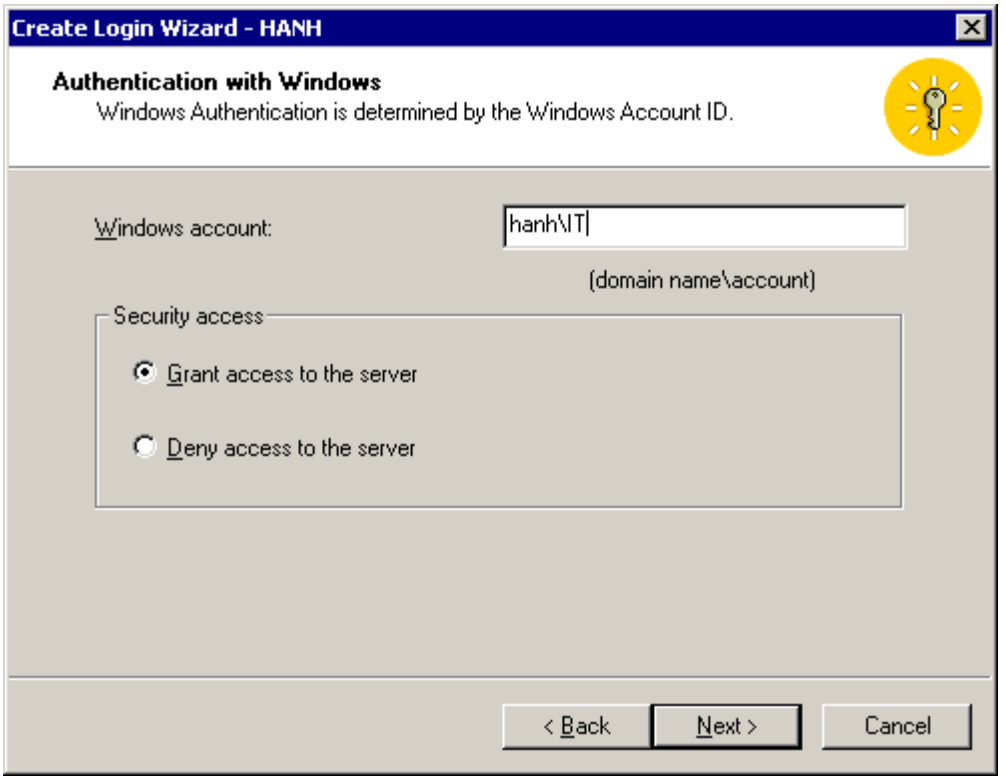
6.2 Tạo tài khoản đăng nhập (Login).

6.2.1 Dùng Create Login Wizard.



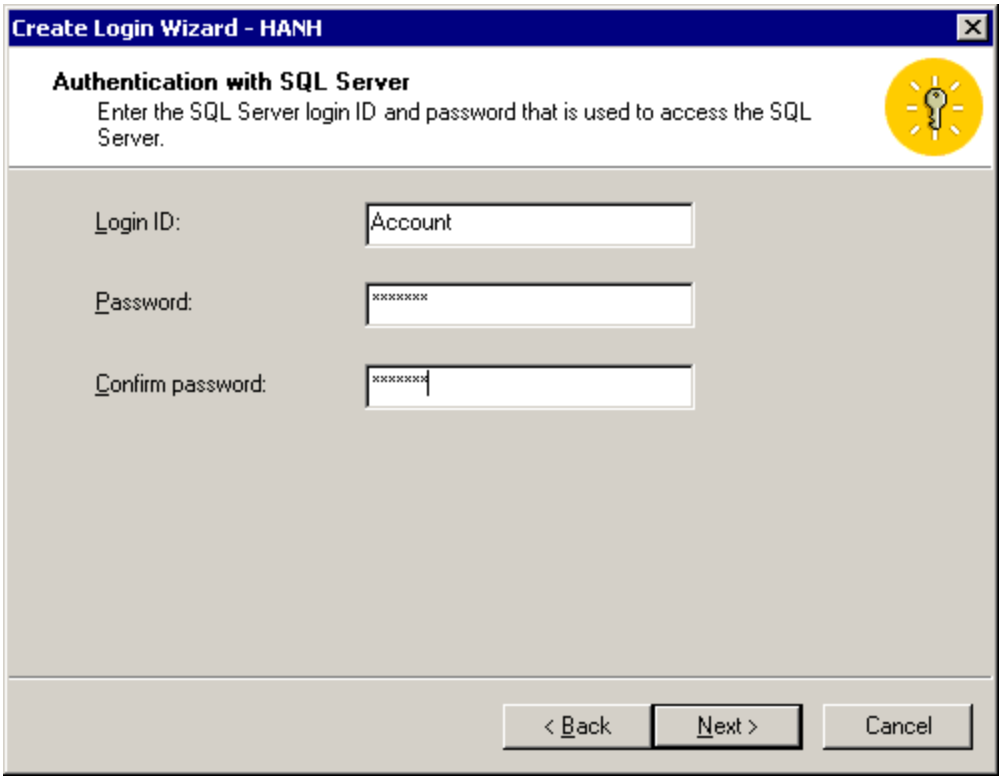
Hình 4: Chọn chế độ đăng nhập cho login đang tạo

Nếu bạn chọn Windows authentication thì bạn phải liên kết login ID này với một user hay group của Windows 2000 có sẵn. Khai báo như trong hình sau



Hình 5: Xác định 1 tài khoản của Windows 2000 và xác định cho phép hoặc từ chối login mới truy xuất đến Server

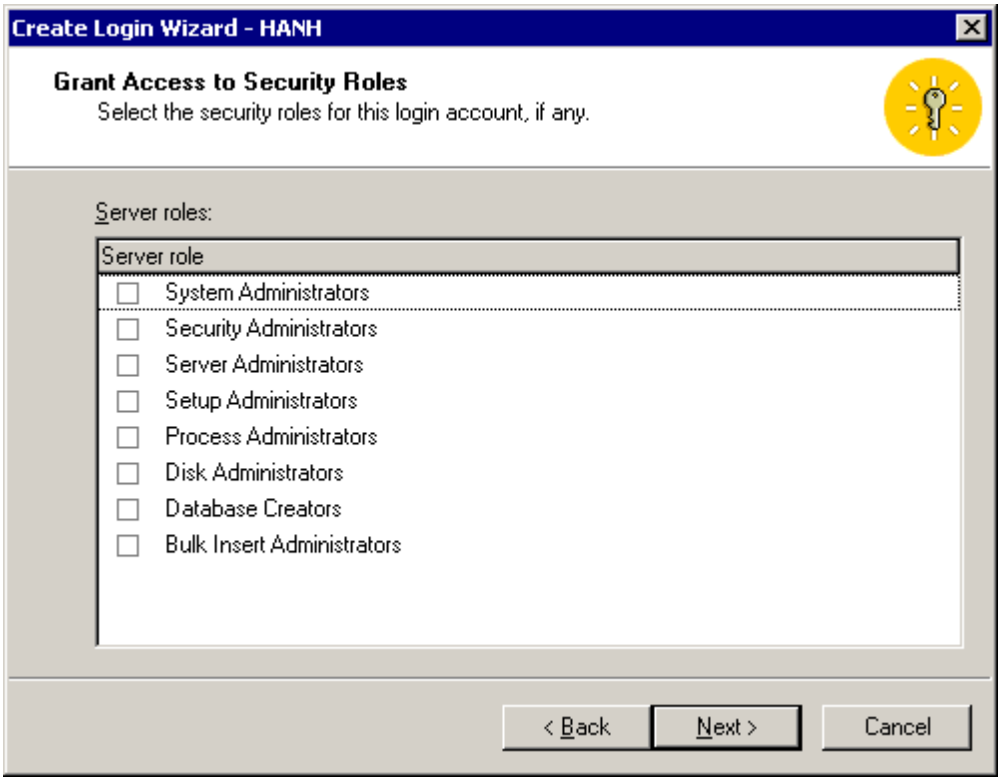
Nếu bạn chọn SQL Server authentication, bạn sẽ tạo một tài khoản bảo mật của SQL Server 2000 như trong hình kế.



Hình 6: Khai báo LogiID và Password

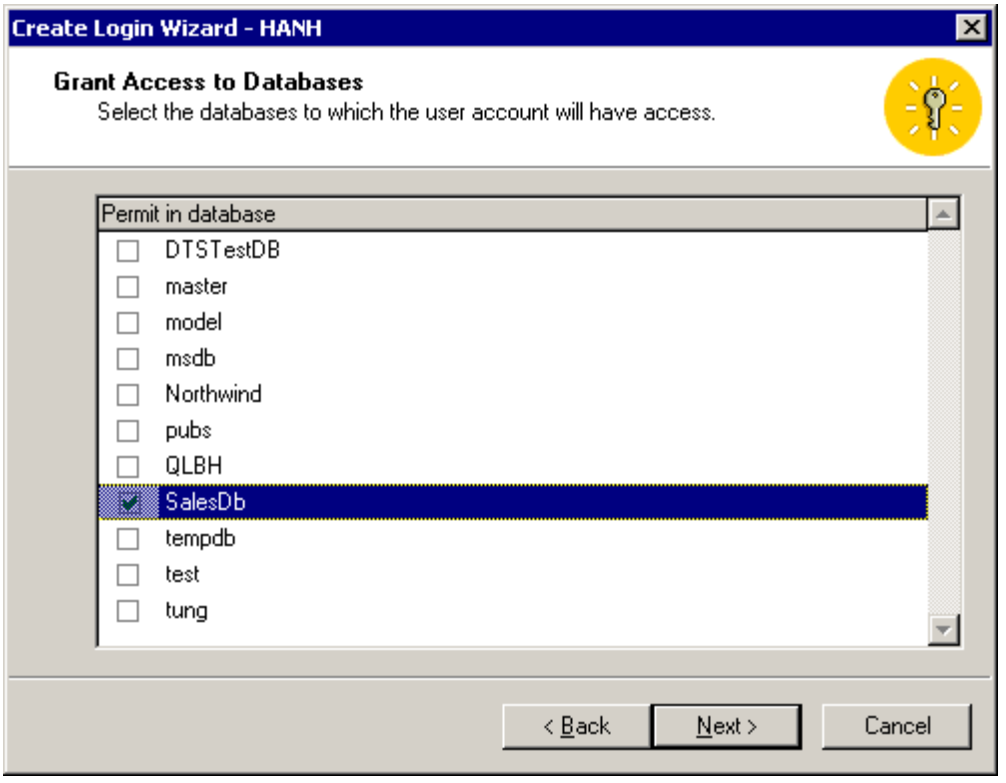
Phải chỉ định rõ tên login và Password. Để ngăn chặn (deny) một login, đơn giản bạn chỉ cần xóa bỏ login từ nơi chứa login trong Enterprise Manager (hoặc từ bảng sysxlogins trong CSDL master).

Sau khi bạn xác định kiểu đăng nhập và liên kết hay tạo tài khoản bảo mật, bạn chỉ định server role (nếu cần) cho login này trong hộp thoại kế. Nếu user sẽ không là một server-wide administrator, thì không cần chọn các server role.



Hình 7: Chỉ định server role cho login đang được tạo

Kế tiếp bạn chỉ định CSDL (nếu cần) để cho user này có thể truy xuất trong hộp thoại kế tiếp. Nhớ rằng hầu hết các server role không cung cấp CSDL truy xuất



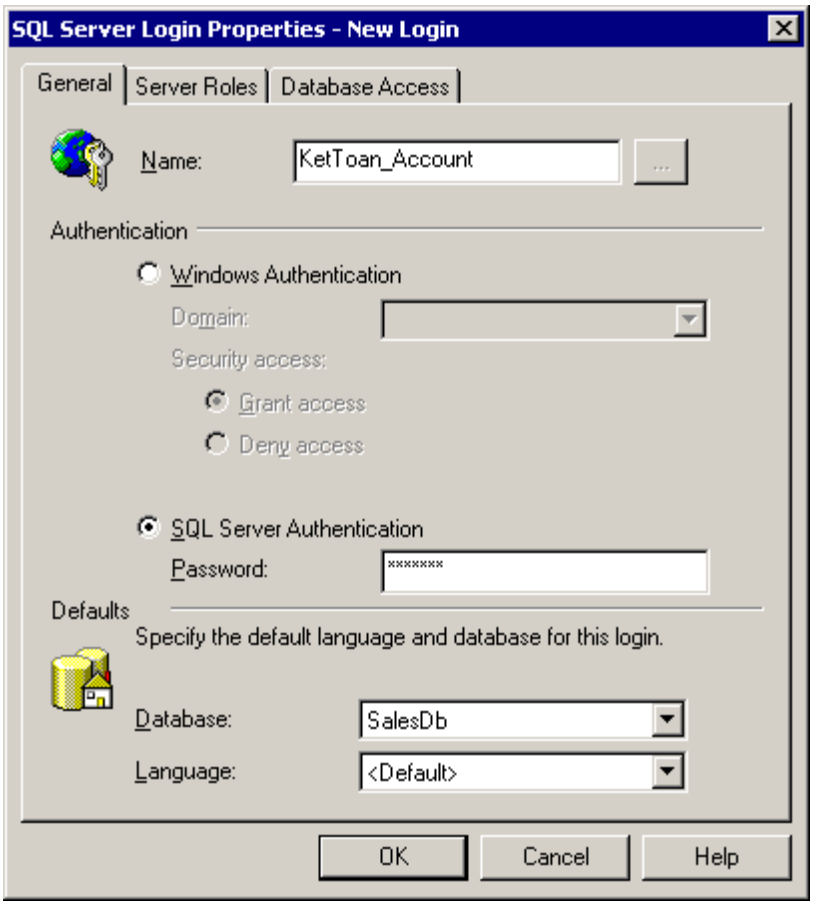
Hình 8: Chỉ định 1 hay nhiều CSDL để login này được truy xuất

Cuối cùng, bạn được SQL Server cho xem trước các lựa chọn mà bạn đã thực hiện trước khi login mới được tạo thực sự. Click vào nút Finish để tạo login.

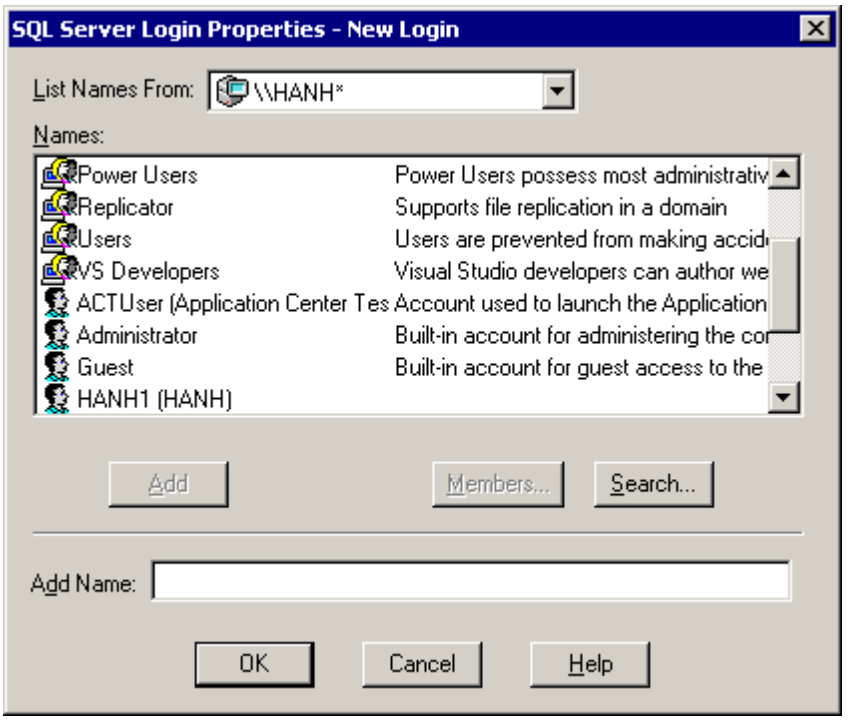
Hình 9: Chọn nút Finish để hoàn tất

6.2.2 Dùng Enterprise Manager để tạo một Login.

Để tạo một login bằng Enterprise Manager, nhấp nút phải chuột tại Security/Login của 1 instance, và chọn New Login. Thông thường, các trang Server Role, và Database Access trong hợp thoại Propertie của login



Hình 10: Chỉ định Name, chế độ chứng thực, chọn CSDL mặc định



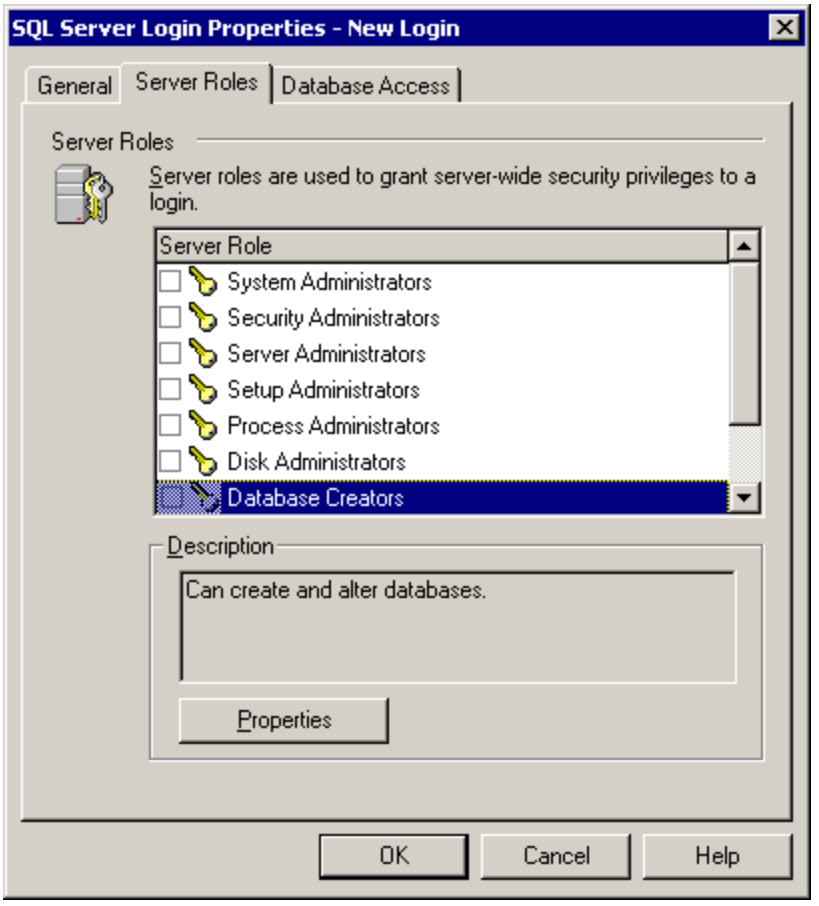
Hình 11: Chọn tài khoản người dùng của Windows

Bạn cũng chọn một CSDL mặc định và ngôn ngữ trong trang General. CSDL mặc định sẽ là CSDL hiện hành khi mà User đăng nhập vào. Mặc định không chọn là

CSDL Master nhưng bạn nên chọn 1 CSDL khác làm mặc định. Ngôn ngữ mặt định là ngôn ngữ mặc định của instance hiện hành.

Lưu ý:

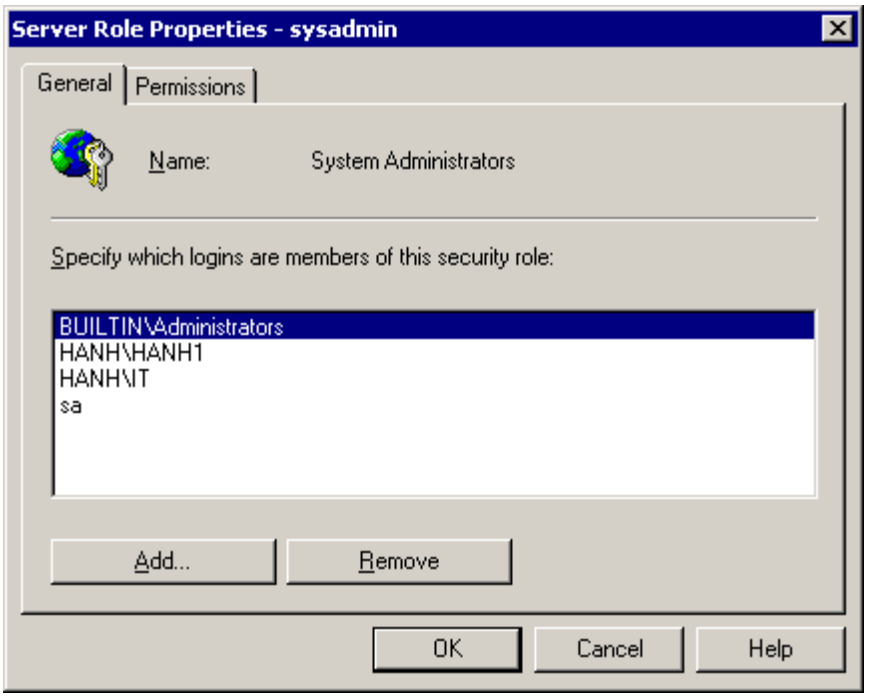
Bạn cũng có thể từ chối sự truy xuất của một User hay group của Windows. Điều này được áp đặt đến việc truy xuất khác của user hay group (kể cả thành viên của một nhóm khác mà có login khác).



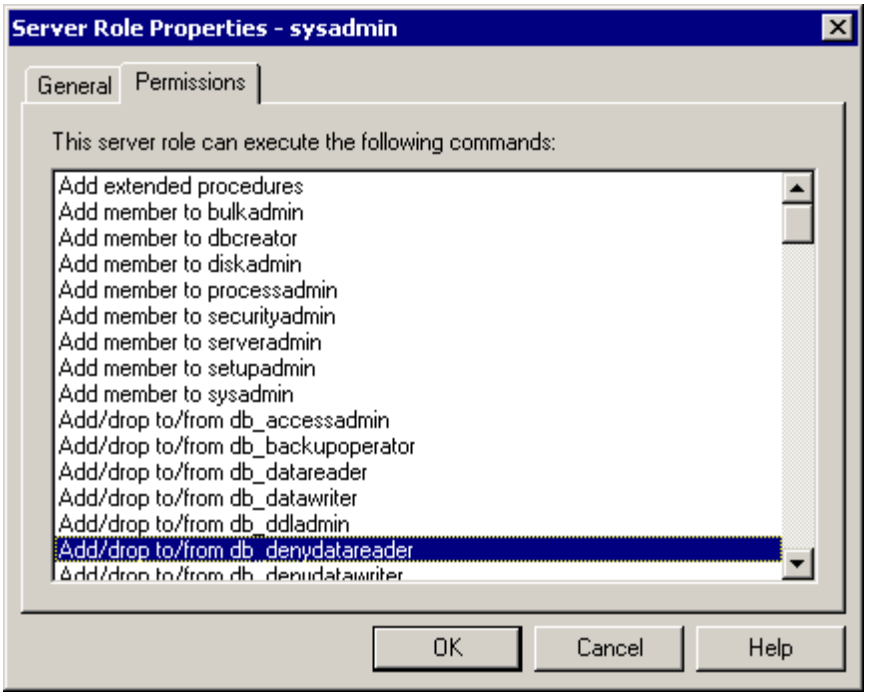
Hình 12: Gán login thuộc server role nào đó nếu cần

Lưu ý rằng nếu System Administrators server role được chọn, thì bạn có thể xem nhóm built-in Administrator group, và login sa của the SQL Server là những thành viên của sysadmin server role.

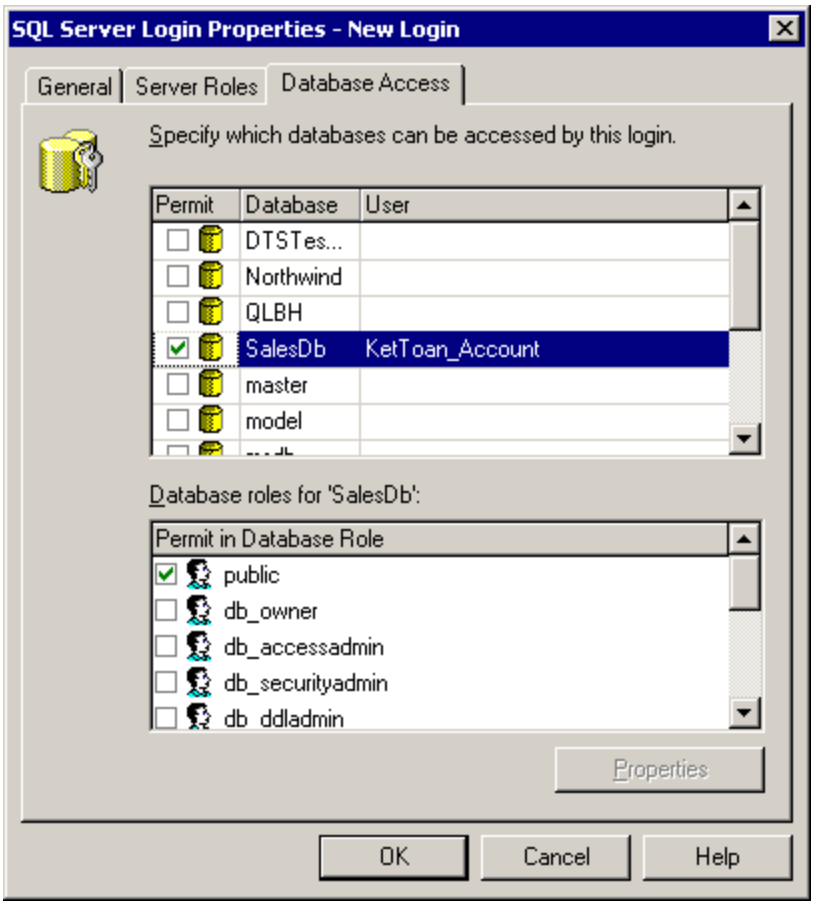
Chọn nút properties để xem một số propererties của Server role



Hình 13:Thành viên của server role



Hình 14: Các quyền hạn cụ thể của Server role



Hình 15: Chỉ định CSDL được truy xuất tới và các quyền cụ thể cho login

Khi bạn chọn 1 CSDL thì login là thành viên của public role trong mỗi CSDL. Bạn không thể xóa (remove) một login ra khỏi public role.

6.2.3 Tạo Login bằng T-SQL.

Bạn cũng có thể tạo các Login bằng T-SQL

Windows Logins

Các thủ tục hệ thống dùng để cấp, hũy, từ chối, hiệu chỉnh một login cho một user hay group của Windows. Chỉ có những thành viên của sysadmin hoặc securityadmin server roles mới có thể thực thi các thủ tục hệ thống này

Thủ tục hệ thống	Mô tả
Sp_grantlogin *'login'*	Tạo một login cho một user hay group của Windows 2000.
Sp_revokelogin *'login'*	Hũy login từ SQL Server đối với các user hay group của Windows 2000 (hoặc Windows NT 4.0). Điều này không rõ ràng lắm vì nó vẫn có thể được cấp quyền truy xuất đến SQL Server 2000.
Sp_denylogin *'login'*	Ngăn chặn một user hay thành viên của Windows 2000 (hoặc Windows NT 4.0) kết nối đến SQL Server 2000. Ngăn chặn các user hay group của thôgn qua một login khác kết với user hay group của Windows.
Sp_defaultdb *'login'*,	Thay đổi CSDL mặc định cho một login.

Thủ tục hệ thống	Mô tả
'database'	
Sp_defaultlanguage *'login', 'language'*	Thay đổi ngôn ngữ mặc định của một login.

Ví dụ:

```
Sp_grantlogin 'CDCN4\Bill'
```

SQL Server Logins

Các thủ tục hệ thống sau cho phép cấp, hũy, từ chối, hiệu chỉnh một login kết với một tài khoản người dùng SQL Server. Chỉ có các thành viên của sysadmin hoặc securityadmin server roles mới có thể thực thi các thủ tục hệ thống này

Thủ tục hệ thống	Mô Tả
Sp_addlogin *'login',* ['password', 'database', 'language', 'sid', encryption_option']	Tạo một login SQL Server mới. Password là NULL nếu không chỉ định. CSDL mặc định là master nếu không chỉ định. Ngôn ngữ mặc định là ngôn ngữ của server hiện hành nếu không chỉ định. Mặc định, password ở trong csdl master.
Sp_droplogin *'login'*	Xóa một SQL Server login.
Sp_password 'old_password', 'new_password', 'login'	Thêm hoặc thay đổi password cho SQL Server login.
Sp_defaultdb 'login', 'database'	Thay đổi CSDL mặc định
Sp_defaultlanguage *'login', 'language'*	Thay đổi ngôn ngữ mặc định.

Ví dụ:

```
Sp_addlogin 'Joe', 'Joe123', 'Northwind'
```

Tạo một SQL Server login mới có tên là **Joe**, với password là **Joe123** và CSDL mặc định Northwind.

Server Roles

Các thủ tục hệ thống sau được dùng để thêm hay xóa một login vào một server role. Chỉ có thành viên của sysadmin server role mới có thể thêm các login vào bất kỳ server role. Các thành viên của server role cũng có thể thêm các login vào server role đó.

Thủ tục hệ thống	Mô tả
Sp_addsrvrolemember *'login', 'role'*	Thêm login như là thành viên của server role.
Sp_dropsrvrolemember *'login', 'role'*	Xoá login không là thành viên của một server role.

Ví dụ:

```
Sp_addsrvrolemember 'Joe', 'securityadmin'
```

Thêm login có tên là Joe vào server role Security Administrator.

Database Access

Các thủ tục sau đây được dùng để thêm hay xóa một login (Windows hoặc SQL Server) hiện hữu được quyền truy xuất trong CSDL hiện hành. Không giống như SQL Server Enterprise Manager, bạn có thể cấp một nhóm của Windows 2000 (hoặc Windows NT 4.0) group truy xuất đến CSDL mà không cần tạo login trước một cách tường minh trong bảng sysxlogins. Chỉ có các thành viên của sysadmin server role, và db_accessadmin và db_owner fixed database role mới có thể thực thi các thủ tục hệ thống này.

Thủ tục hệ thống	Mô tả
Sp_grantdbaccess 'login', 'name_in_db'	Thêm một login như là một user trong CSDL hiện hành. Mặc dù tên user name trong CSDL có thể khác với tên login, điều này không khuyến cáo.
Sp_revokedbaccess *'name'*	Bỏ một login như là một user trong CSDL hiện hành.

Ví dụ:

```
USE Northwind
EXEC Sp_grantdbaccess 'Joe'
```

Cho phép login tên là Joe truy xuất đến CSDL hiện hành, dùng user name là Joe trong CSDL Northwind.

Database Roles

Các thủ tục hệ thống sau đây được dùng để thay đổi database owner, thêm hoặc xóa một tài khoản bảo mật vào một database role có sẵn, hoặc tạo hoặc xóa một user-defined database role.

```
Use Northwind
EXEC Sp_addrolemember 'db_securityadmin', ' CDCN4\KeToan'
```

Thêm tài khoản CDCN4\KeToanvào db_securityadmin database role trogn CSDL Northwind.

Thủ tục hệ thống	Mô tả
Sp_changedbowner 'login', remap_alias_flag	Thay đổi owner của một CSDL người dùng. Chỉ có những thành viên của sysadmin server role hoặc owner database hiện hành mới có thể thau đổi owner của CSDL.
Sp_addrolemember 'role', 'security_account'	Thêm một tài khoản vào một database role trong CSDL hiện hành. Bạn có thể thêm một user-defined database role vào fixed hoặc user-defined database role. Chỉ có những thành viên của sysadmin server role và db_owner and db_security fixed database roles mới có thể thêm thành viên vào database role. Thành viên của database role có thể thành viên vào cho database role đó.

Thủ tục hệ thống	Mô tả
Sp_droprolemember 'role', 'security_account'	Xóa một tài khoản từ một CSDL vào CSDL hiện hành. Chỉ có những thành viên của sysadmin server role và db_owner và db_security fixed dababase roles mới có thể xóa các thành viên ra khỏi database role. Các thành viên của database role mới có thể xóa các thành viên ra khỏi database role.
Sp_addrole 'role', 'owner'	Thêm một user-defined database role mới trong CSDL hiện hành. Mặc dù bạn có thể chỉ định một owner của role, sử dụng mặc định là dbo là không được khuyến cáo. Các thành viên của sysadmin server role và db_securityadmin và db_owner fixed database roles mới có thể tạo user-defined database roles.
Sp_droprole *'role'*	Xóa một user-defined database role ở CSDL hiện hành. Các tàhnh viên của sysadmin server role, db_securityadmin và db_owner fixed database roles mới có thể xóa user-defined database roles.

Để xem các thông tin về login, dùng các thủ tục hệ thống sau:

Thủ tục hệ thống	Mô tả
Sp_helplogins [*'login'*]	Trả về các thông tin của tất cả các login hoặc một login được chỉ định, kể cả các CSDL mà login có truy xuất đến và các database roles mà login là thành viên.
Sp_helpsrvrolemember [*'role'*]	Trả về thông tin về tất cả các server roles và những thành viên của chúng hoặc tất cả các thành viên trong một server role chỉ định.
Sp_helpuser ['security_account']	Trả về thông tin về tất cả các user hoặc user chỉ định trong CSDL hiện hành, kể cả tất cả các hội thành viên của database role.
Sp_helprolemember [*'role'*]	Trả về các thông tin của tất cả các database role hoặc tất cả các thành viên trong database role chỉ định trong CSDL hiện hành.
Sp_helpntgroup [*'name'*]	Trả về các thông tin về các nhóm hoặc 1 nhóm chỉ định của Windows 2000 (hoặc Windows NT 4.0) trong CSDL hiện hành.

Gán các quyền Database

Các view và các stored procedure có thể được tạo trên các table. Khi một user cố gắng lấy thông tin thông qua view hoặc procedure, thì SQL Server 2000 phải kiểm tra user có được phép lấy dữ liệu hay không. Nếu view hoặc procedure được làm chủ

bởi một user và của các bảng cơ sở lại là của một user khác thì SQL Server 2000 phải kiểm tra các quyền trên mỗi object trong dây chuyền đó. Khi một chuỗi các ownership kéo dài thì điều này sẽ ảnh hưởng đến việc thực hiện. Nhưng có lẽ quan trọng hơn là nó có thể bị gắt gối cho nhà quản trị vạch ra và gỡ lỗi các từ việc đưa ra bảo mật

Các quyền về lệnh

Câu lệnh Transact-SQL	Quyền để thực thi câu lệnh Transact-SQL
CREATE DATABASE	Thừa kế bởi thành viên của sysadmin và dbcreator server roles. Mặc dù sysadmin và securityadmin server roles có thể cấp quyền một cách trực tiếp cho các tài khoản để thực hiện câu lệnh này, Tóm lại các tài khoản bảo mật sử dụng dbcreator server role nếu system administrator đại diện quyền. Quyền này chỉ tồn tại trong CSDL master.
BACKUP DATABASE BACKUP LOG	Kế thừa bởi các thành viên của sysadmin server role và db_owner và db_backupoperator fixed database roles. Mặc dù bạn có thể cấp quyền để chạy những câu lệnh này một cách trực tiếp đến các tài khoản bảo mật, một cách tổng quát bạn sẽ sử dụng db_backupoperator fixed database role.
CREATE TABLE CREATE VIEW CREATE PROCEDURE CREATE DEFAULT CREATE RULE CREATE FUNCTION	Kế thừa các thành viên của sysadmin server role và db_owner và db_ddladmin fixed database roles. Quyền cho phép tạo những đối tượng thì được cấp trực tiếp đến nhà lập trình trong suốt thời gian triển khai. Theo mặc định các object được làm chủ bởi người tạo ra đối tượng (mặc dù các đối tượng tạo bởi các thành viên của sysadmin server role thì chủ sẽ là dbo role). Các thành viên của db_owner hoặc db_ddladmin fixed database roles có thể được chỉ định dbo role như là owner của đối tượng đựơc tạo. Ngoài ra, các thành viên của sysadmin server role hoặc db_owner hoặc db_ddladmin fixed database role có thể chỉ định bất kỳ user như là chủ của object mà chúng tạo ra. Tuy nhiên, các user mà không là thành viên của một trong các role này thì không thể chỉ định user khác hoặc dbo role làm chủ của object chúng tạo ra.
CREATE TRIGGER	Kế thừa bởi chủ của table mà của sysadmin server role, và db_owner and db_ddladmin fixed database roles. Những thành viên này không thể cấp quyền để chạy câu lệnh này cho những tài khoản bảo mật khác..

Ví dụ:

```
CREATE TABLE Northwind.dbo.CustomerTable
 (CustID nchar (5), CustomerName nvarchar (40))
```

Cho phép tạo một bảng, cấp ownership cho dbo role. Chỉ có những thành viên sysadmin server role và db_owner hoặc db_ddladmin fixed database roles có thể thực hiện một cách thành công câu lệnh này.

Thay đổi Ownership của Object

Một thành viên của db_owner, db_ddladmin, hoặc db_securityadmin fixed database role, hoặc a member of the sysadmin server role có thể thay đổi ownership của bất kỳ object trong bằng cách chạy thủ tục sp_changeobjectowner.

```
sp_changeobjectowner 'CDCN4\KeToan.Customer', 'dbo'
```

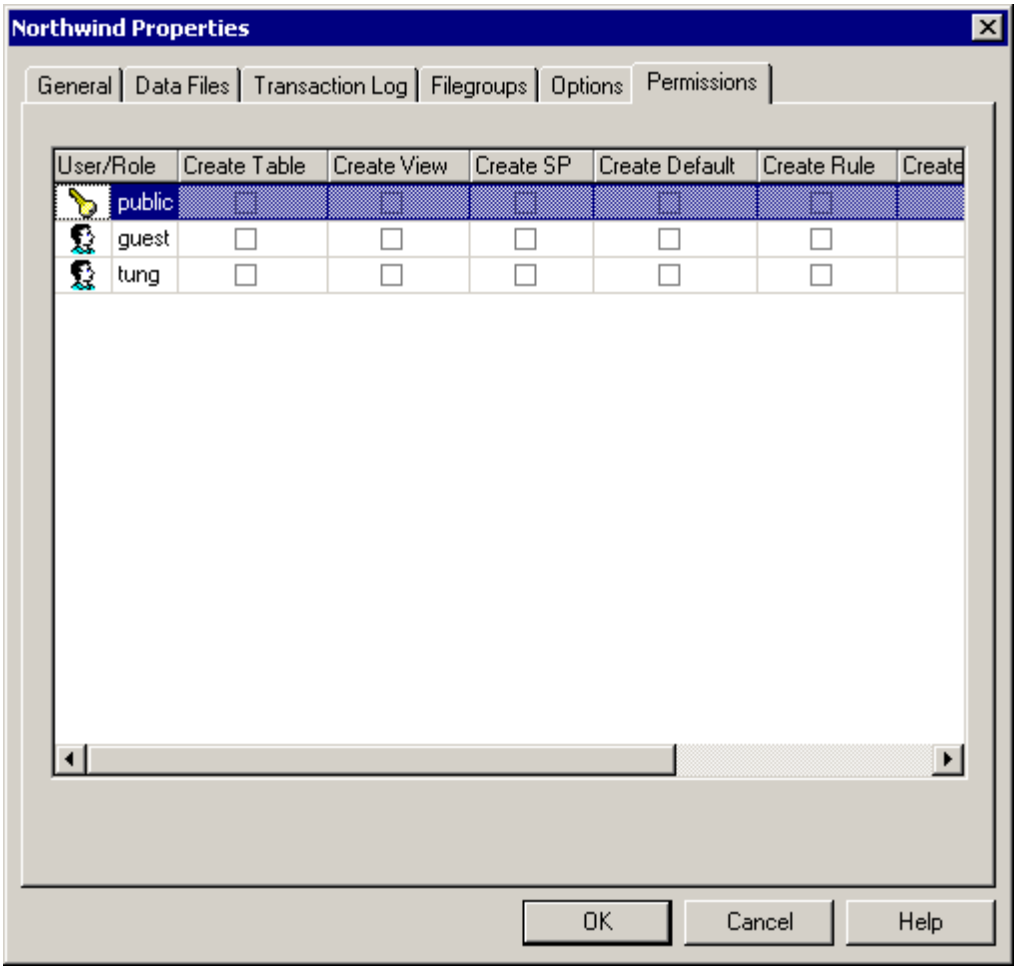
Đổi ownership của table Customer từ CDCN4\KeToanBill cho dbo role.

Lưu ý

Thay đổi owner của một object thì sẽ xó tất cả các quyền hiện có trên đối tượng. Nếu bạn cần giữ lại các quyền thì nên tạo Scrip trước khi đổi owner.

Dùng Enterprise Manager đề Grant, Deny, or Revoke quyền lệnh

Nhắp nút phải chuột tại CSDL cần thực hiện cấp/xoá/từ chối/xem các quyền



Hình 16: Trang Permission của hộp thoại thuộc tính của CSDL