# TÀI LIỆU THỰC HÀNH MÔN CƠ SỞ DỮ LIỆU NÂNG CAO

## Mục lục

# Index

## Tài liệu tham khảo

- SQL Server 2005 T – SQL Recipes, Apress 2008 (Chapter 5)

## Công cụ

- MS SQL Server 2005 Enterprise
- AdventureWorks Database

## Nội dung

### Cú pháp tạo index

Để tạo index trong MS SQL Server 2005 ta sử dụng câu lệnh sau:

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX index_name
    ON {
    [ database_name. [ schema_name ] . | schema_name. ]
        table_or_view_name}
( column [ ASC | DESC ] [ ,...n ] )
```

Tham số:

- [UNIQUE]: Đảm bảo tính duy nhất trên cột không khóa.
- [CLUSTERED | NONCLUSTERED]: Kiểu của index (Chi tiết tham khảo Indexes Overview – Trang 155 SQL Server 2005 T – SQL Recipes).
- Index_name: Tên của index
- table_or_view_name: Tên bảng hoặc view cần tạo index
- [ASC | DESC]: Chỉ định thứ tự sắp xếp của cột đánh index, ASC – thứ tự tăng dần, DESC – thứ tự giảm dần

### Cú pháp chỉnh sửa index

Để chỉnh sửa một index đã tồn tại ta dùng câu lệnh:

```
ALTER INDEX index_name
ON object_name
...
```

### Ví dụ

Xem xét cơ sở dữ liệu AdventureWorks với bảng Department: http://msdn.microsoft.com/en-us/library/ms124543%28v=sql.100%29.aspx

| Column | Data type | Nullability | Description |
|---|---|---|---|
| **DepartmentID** | smallint | Not null | Primary key for **Department** rows. |
| **Name** | **Name** (user-defined type) nvarchar(50) | Not null | Name of the department. |
| **GroupName** | **Name** (user-defined type) nvarchar(50) | Not null | Name of the group to which the department belongs. |
| **ModifiedDate** | datetime | Not null | Date and time the row was last updated. |

Tạo mới bảng TerminationReason:

```
CREATE TABLE HumanResources.TerminationReason(
   TerminationReasonID smallint IDENTITY(1,1) NOT NULL,
    TerminationReason varchar(50) NOT NULL,
   DepartmentID smallint NOT NULL,
 CONSTRAINT FK_TerminationReason_DepartmentID
   FOREIGN KEY (DepartmentID) REFERENCES
   HumanResources.Department(DepartmentID)
)
GO
```

Khi một khóa chính được tạo ra trên một cột sử dụng câu lệnh CREATE TABLE hoặc ALTER TABLE thì một index cũng được tạo ra. Ví dụ một Clustered Index được tạo ra trên cột TerminationReasonID khi lệnh sau được thực thi:

```
ALTER TABLE HumanResources.TerminationReason
ADD CONSTRAINT PK_TerminationReason PRIMARY KEY CLUSTERED (TerminationReasonID)
```

Để tạo NonClustered Index trên cột DepartmentID ta sử dụng câu lệnh:

```
CREATE NONCLUSTERED INDEX NCI_TerminationReason_DepartmentID ON
HumanResources.TerminationReason (DepartmentID)
```

## Áp đặt tính duy nhất trên cột không khóa

Để áp đặt tính duy nhất trên cột không khóa ta sử dụng từ khóa UNIQUE. Ví dụ sau sẽ tạo index trên cột TerminationReason trong bảng HumanResource.TerminationReason:

```
CREATE UNIQUE NONCLUSTERED INDEX UNI_TerminationReason ON
HumanResources.TerminationReason (TerminationReason)
```

Để kiểm tra kết quả, thêm hai bản ghi sau:

```
INSERT HumanResources.TerminationReason
(DepartmentID, TerminationReason)
VALUES (1, 'Bad Engineering Skills')

INSERT HumanResources.TerminationReason
(DepartmentID, TerminationReason)
VALUES (2, 'Breaks Expensive Tools')
```

Sau đó thực hiện thêm bản ghi với cột TerminationReason có giá trị là 'Bad Engineering Skills':

```
INSERT HumanResources.TerminationReason
(DepartmentID, TerminationReason)
VALUES (2, 'Bad Engineering Skills')
```

Thông báo lỗi xuất hiện:

```
Cannot insert duplicate key row in object 'HumanResources.TerminationReason'
with unique index 'UNI_TerminationReason'.
The statement has been terminated.
```

Như vậy, một unique index đã được tạo ra đảm bảo không có hai hàng bất kỳ có giá trị trùng nhau trên cột TerminationReason.

### Tạo composite index

Composite index là loại index được thiết lập trên nhiều cột. Ta sử dụng composite index khi nhiều cột thường xuyên xuất hiện trong điều kiện tìm kiếm của các truy vấn. Giả sử TerminationReason và DepartmentID thường xuyên xuất hiện trong mệnh đề WHERE của câu lệnh SELECT. Như vậy ta sẽ tạo ra một NONCLUSTERED INDEX để tăng tốc độ thực hiện các truy vấn đó:

```
CREATE NONCLUSTERED INDEX NI_TerminationReason_TerminationReason_DepartmentID
ON HumanResources.TerminationReason(TerminationReason, DepartmentID)
```

### Thiết lập chỉ mục tăng dần – giảm dần

Mặc định khi tạo ra chỉ mục sẽ là tăng dần. Để thiết lập chỉ mục là tăng dần hay giảm dần ta thêm lựa chọn sau trong cú pháp tạo chỉ mục:

```
( column [ ASC | DESC ] [ ,...n ] )
```

Ví dụ, sử dụng câu lệnh sau để thêm cột ViolationSeverityLevel kiểu smallint:

```
ALTER TABLE HumanResources.TerminationReason
ADD ViolationSeverityLevel smallint
GO
```

Sau đó sử dụng câu lệnh sau tạo chỉ mục giảm dần trên cột ViolationSeverityLevel:

```
CREATE NONCLUSTERED INDEX NI_TerminationReason_ViolationSeverityLevel
ON HumanResources.TerminationReason (ViolationSeverityLevel DESC)
```

### Xem các chỉ mục đã thiết lập

Để xem thông tin về các chỉ mục đã được thiết lập trên một bảng, ta sử dụng thủ tục sp_helpindex với tham số là tên bảng cần xem chỉ mục. Ví dụ muốn xem thông tin về các chỉ mục đã thiết lập trên bảng Employee ta sử dụng cú pháp sau:

```
EXEC sp_helpindex 'HumanResources.Employee'
```

Kết quả như sau:

```
index_name               index_description                                  index_keys
-------------------------------------------------------------------------------------
AK_Employee_LoginID      nonclustered, unique located on PRIMARY            LoginID
...
AK_Employee_rowguid      nonclustered, unique located on PRIMARY            rowguid
IX_Employee_ManagerID    nonclustered located on PRIMARY                    ManagerID
PK_Employee_EmployeeID   clustered, unique, primary key located on PRIMARY  EmployeeID
```

### Vô hiệu hóa một index

Để vô hiệu hóa một index ta sử dụng cú pháp:

```
ALTER INDEX index_name ON
table_or_view_name DISABLE
```

Ví dụ muốn vô hiệu hóa index UNI_TerminationReason trên bảng TerminationReason ta sử dụng câu lệnh sau:

```
ALTER INDEX UNI_TerminationReason ON
HumanResources.TerminationReason DISABLE
```

### Xóa index

Từ khóa DISABLE cho phép vô hiệu hóa index nhưng index vẫn còn trong cơ sở dữ liệu. Để xóa hẳn index khỏi cơ sở dữ liệu ta sử dụng câu lệnh DROP INDEX với cú pháp như sau:

```
DROP INDEX <table_or_view_name>.<index_name> [ ,...n ]
```

Ví dụ muốn xóa index UNI_TerminationReason khỏi cơ sở dữ liệu ta sử dụng câu lệnh sau:

```
DROP INDEX HumanResources.TerminationReason.UNI_TerminationReason
GO
```

### Thay đổi index đã tồn tại với DROP_EXISTING

Lựa chọn DROP_EXISTING cho phép xóa và tạo lại index trong cùng một câu lệnh. Lựa chọn này hữu ích khi muốn thay đổi các cột thiết lập bởi chỉ mục. Câu lệnh ALTER INDEX có thể được sử dụng để thay đổi, dịch lại, tổ chức lại hay vô hiệu hóa chỉ mục nhưng không sử dụng để thêm, xóa các cột thiết lập bởi chỉ mục. Hơn nữa, khi sử dụng DROP_EXISTING để thay đổi một clustered index sẽ không làm các nonclustered index hiện có tự động dịch lại. Ví dụ sau tạo nonclustered index NCI_TerminationReason_DepartmentID:

```
CREATE NONCLUSTERED INDEX NCI_TerminationReason_DepartmentID ON
HumanResources.TerminationReason
(DepartmentID ASC)
WITH (DROP_EXISTING = ON)
GO
```

Sau đó muốn thêm cột ViolationSeverityLevel vào NCI_TerminationReason_DepartmentID ta sử dụng câu lệnh sau:

```
CREATE NONCLUSTERED INDEX NCI_TerminationReason_DepartmentID ON
HumanResources.TerminationReason
(ViolationSeverityLevel, DepartmentID DESC)
WITH (DROP_EXISTING = ON)
GO
```

# Transaction

Trong phần này ta tìm hiểu ba kiểu transaction: autocommit, explicit và implicit.

## Autocommit Transaction

Autocommit transaction là kiểu mặc định trong SQL Server 2005 cho phép mỗi câu lệnh SQL được đặt trong một transation riêng và tự động commit khi câu lệnh kết thúc. Giả sử ta có hai câu lệnh INSERT, câu lệnh thứ nhất bị lỗi, câu lệnh thứ hai thực hiện thành công thì thay đổi do câu lệnh thứ hai tạo ra vẫn được lưu lại.

Ví dụ: Xét bảng NonClusteredTable(ID, Name) đã có một unique clustered với tên index_id2 và dữ liệu hiện có trên bảng như sau:

| | ID | Name |
|---|---|---|
| 1 | 0 | So khong |
| 2 | 1 | So mot |
| 3 | 2 | So hai |
| 4 | 3 | So ba |
| 5 | 4 | So bon |

Như vậy khi thực hiện hai câu lệnh sau cùng lúc ta thấy câu lệnh thứ nhất bị lỗi do ràng buộc unique trên index_id2 nhưng câu lệnh thứ hai vẫn được thực hiện thành công:

```
INSERT INTO NonClusteredTable VALUES('1','So mot')
UPDATE NonClusteredTable SET Name='SO MOT' WHERE ID='1'
```

Kết quả:

```
Msg 2601, Level 14, State 1, Line 1
Cannot insert duplicate key row in object 'dbo.NonClusteredTable' with unique index 'index_id2'.
The statement has been terminated.

(1 row(s) affected)
```

## Implicit Transaction

Đây là kiểu transaction ẩn, được thiết lập thông qua câu lệnh:

SET IMPLICIT_TRANSACTIONS { ON | OFF }

Khi Implicit_Transactions được thiết lập là ON thì các câu lệnh sau sẽ tự động bắt đầu một transaction:

- SELECT, INSERT, UPDATE, DELETE
- ALTER TABLE
- TRUNCATE TABLE
- OPEN, FETCH
- GRANT, REVOKE

Và để kết thúc transaction thì cuối transaction phải có câu lệnh COMMIT hoặc ROLLBACK, nếu không các thay đổi do transaction tạo ra sẽ không được lưu lại khi ngắt kết nối.

Ví dụ: Xét lại bảng NonClusteredTable(ID, Name) với dữ liệu hiện có trên bảng như sau:

| | ID | Name |
|---|---|---|
| 1 | 0 | So khong |
| 2 | 1 | So mot |
| 3 | 2 | So hai |
| 4 | 3 | So ba |
| 5 | 4 | So bon |

Trước hết ta thiết đặt chế độ Implicit Transaction là ON:

```
SET IMPLICIT_TRANSACTIONS ON
```

Sau đó thực hiện hai câu lệnh cập nhật:

```
UPDATE NonClusteredTable SET Name='SO HAI' WHERE ID='2'
UPDATE NonClusteredTable SET Name='SO BA' WHERE ID='3'
```

Sau đó ta ngắt kết nối và kết nối trở lại thì thấy sự thay đổi do hai câu lệnh trên tạo ra không được lưu lại.

Sử dụng lệnh COMMIT hoặc ROLLBACK để hoàn tất transaction:

```
{COMMIT | ROLLBACK} TRANSACTION Transaction_Name
```

## Explicit Transaction

Đây là kiểu transaction do người dùng tự định nghĩa thông qua các câu lệnh: BEGIN TRANSACTION, ROLLBACK TRANSACTION, COMMIT TRANSACTION, BEGIN DISTRIBUTED TRANSACTION, SAVE TRANSACTION, @@TRANCOUNT

Ví dụ sau sử dụng Explicit Transaction để hoàn tất hoặc hủy bỏ những thay đổi do transaction tạo ra tùy thuộc vào lỗi trả về trong một khối lệnh:

```
-- Before count
SELECT COUNT(*) BeforeCount FROM HumanResources.Department

-- Variable to hold the latest error integer value
DECLARE @Error int

BEGIN TRANSACTION

INSERT HumanResources.Department
(Name, GroupName)
VALUES ('Accounts Payable', 'Accounting')

SET @Error = @@ERROR
IF (@Error<> 0) GOTO Error_Handler

INSERT HumanResources.Department
(Name, GroupName)
VALUES ('Engineering', 'Research and Development')

SET @Error = @@ERROR
IF (@Error <> 0) GOTO Error_Handler

COMMIT TRAN

Error_Handler:
IF @Error <> 0

BEGIN
   ROLLBACK TRANSACTION
END

-- After count
SELECT COUNT(*) AfterCount FROM HumanResources.Department
```

Kết quả như sau:

```
BeforeCount
-----------
18

(1 row(s) affected)


(1 row(s) affected)

Msg 2601, Level 14, State 1, Line 14
Cannot insert duplicate key row in object 'HumanResources.Department'
➥ with unique index 'AK_Department_Name'.
The statement has been terminated.

AfterCount
-----------
18

(1 row(s) affected)
```

# Sao lưu và phục hồi dữ liệu

## Các loại backup

Microsoft SQL Server 2005 có 3 lựa chọn để backup dữ liệu: full, transaction log và differential backup.

Full backup cho phép tạo một bản sao lưu toàn bộ cơ sở dữ liệu. Khi thực hiện full backup không cần phải offline cơ sở dữ liệu nhưng lại tốn thời gian và tài nguyên của hệ thống đặc biệt khi kích thước cơ sở dữ liệu lớn.

Differential backup là một lựa chọn khác cho phép giảm thời gian và không gian lưu trữ khi tiến hành backup. Differential backup chỉ sao lưu những thay đổi dữ liệu từ lần full backup gần nhất. Do vậy lựa chọn này thường sử dụng đi kèm với full backup.

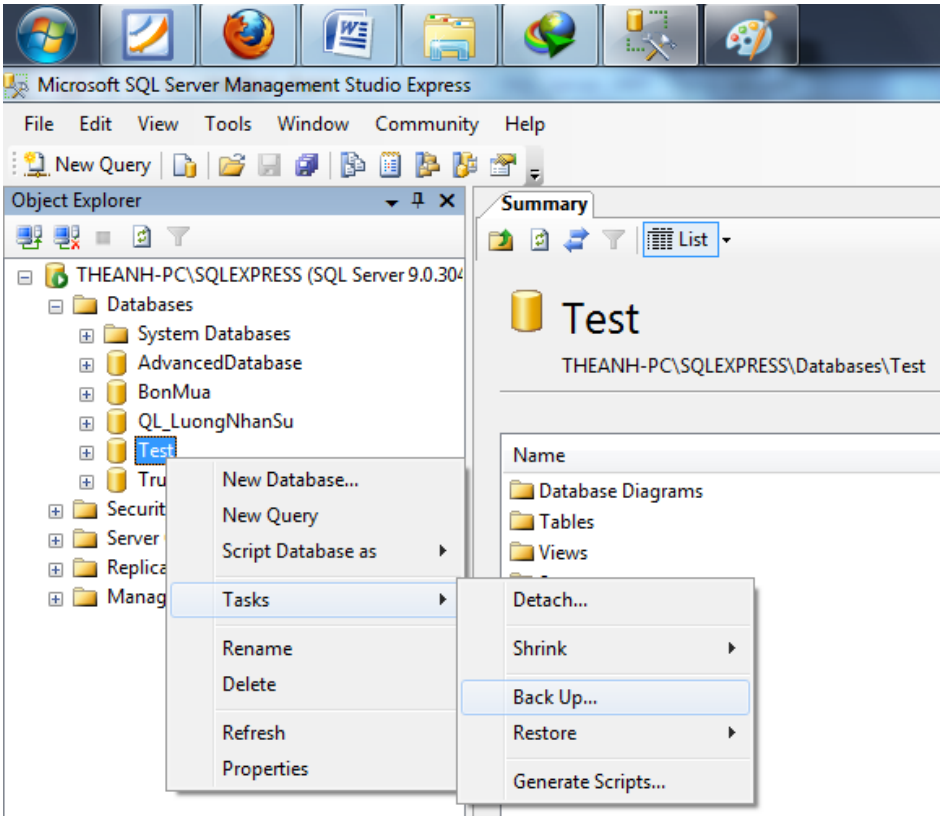
Trong quá trình hoạt động, mọi thay đổi đều được SQL Server lưu trong transaction log. Transaction log backup cho phép sao lưu các transaction trong file log vào thiết bị lưu trữ, khi đó SQL Server sẽ xóa các transaction đó khỏi file log.

Ví dụ về một kế hoạch sao lưu dữ liệu: Một DBA thực hiện full backup vào buii tối thứ 6 và different backup vào các buổi tối từ thứ 2 đến thứ 5 và transaction log backup mỗi giờ một lần. Giả sử sự cố xảy ra vào 9h:05 ngày thứ 4, DBA sẽ tiến hành khôi phục dữ liệu như sau: Dùng bản full backup để khôi phục dữ liệu về tối ngày thứ 6. Sau đó dùng bản differential backup để đưa cơ sở dữ liệu về trạng thái tối thứ 3. Cuối cùng sử dụng transaction log backup để đưa cơ sở dữ liệu về thời điểm 9h sáng thứ 4.

## Thực hiện sao lưu cơ sở dữ liệu

Để thực hiện sao lưu cơ sở dữ liệu ta thực hiện theo các bước sau:

- Bật MS SQL Server
- Click chuột phải vào cơ sở dữ liệu cần sao lưu
- Chọn phần Task
- Chọn Backup




- Chọn chế độ backup: Full, Differential hoặc Transaction log