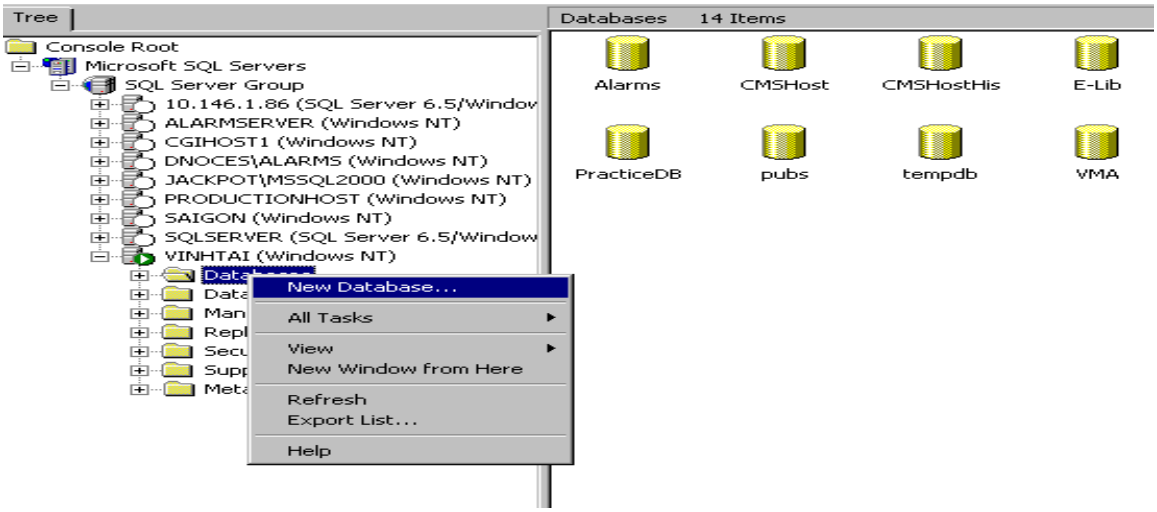
Cách tạo User và Thiết kế Database

Bởi:

Khoa CNTT ĐHSP KT Hưng Yên

Cách tạo một User Database

Chúng ta có thể tạo một database dễ dàng dùng SQL Server Enterprise bằng cách right-click lên trên "database" và chọn "New Database" như hình vẽ sau:



Sau đó chúng ta chỉ việc đánh tên của database và click OK.

Ngoài ra đôi khi chúng ta cũng dùng SQL script để tạo một database. Khi đó ta phải chỉ rõ vị trí của primary data file và transaction log file.

Ví dụ:

USE master

GO

CREATE DATABASE Products

```
ON

( NAME = prods_dat,

FILENAME = 'c:\program files\microsoft SQL server\mssql\data\prods.mdf',

SIZE = 4,

MAXSIZE = 10,

FILEGROWTH = 1

)

GO
```

Trong ví dụ trên ta tạo một database tên là Products với logical file name là prods_dat và physical file name là prods.mdf, kích thước ban đầu là 4 MB và data file sẽ tự động tăng lên mỗi lần 1 MB cho tới tối đa là 10 MB. Nếu ta không chỉ định một transaction log file thì SQL sẽ tự động tạo ra 1 log file với kích thước ban đầu là 1 MB.

Lưu Ý:

Khi tạo ra một database chúng ta cũng phải lưu ý một số điểm sau: Đối với các hệ thống nhỏ mà ở đó vấn đề tốc độ của server không thuộc loại nhạy cảm thì chúng ta thường chọn các giá trị mặc định (default) cho **Initial size**, **Automatically growth file**. Nhưng trên một số production server của các hệ thống lớn kích thước của database phải được người DBA ước lượng trước tùy theo tầm cỡ của business, và thông thường người ta không chọn Autogrowth(tự động tăng trưởng) và Autoshrink(tự động nén). Câu hỏi được đặt ra ở đây là vì sao ta không để SQL Server chọn một giá trị khởi đầu cho datafile và sau đó khi cần thì nó sẽ tự động nới rộng ra mà lại phải ước lượng trước? Nguyên nhân là nếu chọn Autogrowth (hay Autoshrink) thì chúng ta có thể sẽ gặp 2 vấn đề sau:

- **Performance hit:** Ảnh hưởng đáng kể đến khả năng làm việc của SQL Server. Do nó phải thường xuyên kiểm tra xem có đủ khoảng trống cần thiết hay không và nếu không đủ nó sẽ phải mở rộng bằng cách dành thêm khoảng trống từ dĩa cứng và chính quá trình này sẽ làm chậm đi hoạt động của SQL Server.
- **Disk fragmentation** : Việc mở rộng trên cũng sẽ làm cho data không được liên tục mà chứa ở nhiều nơi khác nhau trong dĩa cứng điều này cũng gây ảnh hưởng lên tốc độ làm việc của SQL Server.

Trong các hệ thống lớn người ta có thể dự đoán trước kích thước của database bằng cách tính toán kích thước của các tables, đây cũng chỉ là kích thước ước đoán mà thôi (xin xem "Estimating the size of a database" trong SQL Books Online để biết thêm về cách

tính) và sau đó thường xuyên dùng một số câu lệnh SQL (thường dùng các câu lệnh bắt đầu bằng **DBCC** .Phần này sẽ được bàn qua trong các bài sau) kiểm tra xem có đủ khoảng trống hay không nếu không đủ ta có thể chọn một thời điểm mà SQL server ít bận rộn nhất (như ban đêm hay sau giờ làm việc) để nới rộng data file như thế sẽ không làm ảnh hưởng tới performance của Server.

Chú ý giả sử ta dành sẵn 2 GB cho datafile, khi dùng Window Explorer để xem ta sẽ thấy kích thước của file là 2 GB nhưng data thực tế có thể chỉ chiếm vài chục MB mà thôi.

Những Điểm Cần Lưu Ý Khi Thiết Kế Một Database

Trong phạm vi bài này chúng ta không thể nói sâu về lý thuyết thiết kế database mà chỉ đưa ra một vài lời khuyên mà bạn nên tuân theo khi thiết kế.

Trước hết bạn phải nắm vững về các loại **data type**. Ví dụ bạn phải biết rõ sự khác biệt giữa **char(10)**, **nchar(10)varchar(10)**, **nvarchar(10)**. Loại dữ liệu Char là một loại string có kích thước cố định nghĩa là trong ví dụ trên nếu data đưa vào "This is a really long character string" (lớn hơn 10 ký tự) thì SQL Server sẽ tự động cắt phần đuôi và ta chỉ còn "This is a". Tương tự nếu string đưa vào nhỏ hơn 10 thì SQL sẽ thêm khoảng trống vào phía sau cho đủ 10 ký tự. Ngược lại loại varchar sẽ không thêm các khoảng trống phía sau khi string đưa vào ít hơn 10. Còn loại data bắt đầu bằng chữ **n** chứa dữ liệu dạng unicode.

Một lưu ý khác là trong SQL Server ta có các loại Integer như : **tinyint**, **smallint**, **int**, **bigint**. Trong đó kích thước từng loại tương ứng là 1,2,4,8 bytes. Nghĩa là loại **smallint** tương đương với **Integer** và loại **int** tương đương với **Long** trong VB.

Khi thiết kế table nên:

- Có ít nhất một cột thuộc loại **ID** dùng để xác định một record dễ dàng.
- Chỉ chứa data của một entity (một thực thể)

Trong ví dụ sau thông tin về Sách và Nhà Xuất Bản được chứa trong cùng một table

BookID	Title	Publisher	PubState	PubCity	PubCountry
1	Inside SQL Server 2000	Microsoft Press	CA	Berkely	USA
2	Windows 2000 Server	New Riders	MA	Boston	USA
3	Beginning Visual Basic 6.0	Wrox	CA	Berkely	USA

Books

Ta nên tách ra thành table Books và table Publisher như sau:

BookID	Title	PublisherID
1	Inside SQL Server 2000	P1
2	Windows 2000 Server	P2
3	Beginning Visual Basic 6.0	P3

Books

và

PublisherID	Publisher	PubState	PubCity	PubCountry
P1	Microsoft Press	CA	Berkely	USA
P2	New Riders	MA	Boston	USA
P3	Wrox	CA	Berkely	USA

Publishers

- Tránh dùng cột có chứa **NULL** và nên luôn có giá trị **Default** cho các cột
- Tránh lập lại một giá trị hay cột nào đó

Ví dụ một cuốn sách có thể được viết bởi hơn một tác giả và như thế ta có thể dùng một trong 2 cách sau để chứa data:

BookID	Title	Authors
1	Inside SQL Server 2000	John Brown
2	Windows 2000 Server	Matthew Bortniker, Rick Johnson
3	Beginning Visual Basic 6.0	Peter Wright, James Moon, John Brown

Books

hay

BookID	Title	Author1	Author2	Author3
1	Inside SQL Server 2000	John Brown	Null	Null
2	Windows 2000 Server	Matthew Bortniker	Rick Johnson	Null
3	Beginning Visual Basic 6.0	Peter Wright	James Moon	John Brown

Books

Tuy nhiên việc lập đi lập lại cột Author sẽ tạo nhiều vấn đề sau này. Chẳng hạn như nếu cuốn sách có nhiều hơn 3 tác giả thì chúng ta sẽ gặp phiền phức ngay....Trong ví dụ này ta nên chặt ra thành 3 table như sau:

BookID	Title
1	Inside SQL Server 2000
2	Windows 2000 Server
3	Beginning Visual Basic 6.0

Books

AuthID	First Name	Last Name
A1	John	Brown
A2	Matthew	Bortniker
A3	Rick	Johnson
A4	Peter	Wright
A5	James	Moon

Authors

BookID	AuthID
1	A1
2	A2
2	A3
3	A4
3	A5
3	A1

AuthorBook

Ngoài ra một trong những điều quan trọng là phải biết rõ quan hệ (**Relationship**) giữa các table:

- **One-to-One Relationships** : trong mối quan hệ này thì một hàng bên table A không thể liên kết với hơn 1 hàng bên table B và ngược lại.
- **One-to-Many Relationships** : trong mối quan hệ này thì một hàng bên table A có thể liên kết với nhiều hàng bên table B.
- **Many-to-Many Relationships** : trong mối quan hệ này thì một hàng bên table A có thể liên kết với nhiều hàng bên table B và một hàng bên table B cũng có thể liên kết với nhiều hàng bên table A. Như ta thấy trong ví dụ trên một cuốn sách có thể được viết bởi nhiều tác giả và một tác giả cũng có thể viết nhiều cuốn sách. Do đó mối quan hệ giữa Books và Authors là quan hệ Many to Many. Trong trường hợp này người ta thường dùng một table trung gian để giải quyết vấn đề (table AuthorBook).

Để có một database tương đối hoàn hảo nghĩa là thiết kế sao cho data chứa trong database không thừa không thiếu bạn cần biết thêm về các thủ thuật **Normalization**. Tuy nhiên trong phạm vi khóa học này chúng tôi không muốn bàn sâu hơn về đề tài này, bạn có thể xem thêm trong các sách dạy lý thuyết cơ sở dữ liệu.