

UBND TỈNH LÂM ĐỒNG
TRƯỜNG CAO ĐẲNG ĐÀ LẠT

GIÁO TRÌNH
MÔ ĐUN: QUẢN TRỊ CƠ SỞ DỮ LIỆU VỚI SQL SERVER
NGHỀ: CÔNG NGHỆ THÔNG TIN (UĐPM)
TRÌNH ĐỘ: TRUNG CẤP

*(Ban hành kèm theo Quyết định số: /QĐ - ngày tháng năm 2017 của
Trường Cao đẳng Đà Lạt)*



LƯU HÀNH NỘI BỘ

Đà Lạt, năm 2017

TUYÊN BỐ BẢN QUYỀN

Tài liệu này thuộc loại sách giáo trình nên các nguồn thông tin có thể được phép dùng nguyên bản hoặc trích dùng cho các mục đích về đào tạo và tham khảo.

Mọi mục đích khác mang tính lệch lạc hoặc sử dụng với mục đích kinh doanh thiếu lành mạnh sẽ bị nghiêm cấm.

LỜI GIỚI THIỆU

Vài nét về xuất xứ giáo trình:

Giáo trình này được viết theo Kế hoạch số 1241/KH-CDNĐL ngày 30 tháng 12 năm 2016 của Trường Cao đẳng nghề Đà Lạt về việc triển khai xây dựng chương trình đào tạo theo Luật Giáo dục nghề nghiệp để làm tài liệu dạy nghề trình độ cao đẳng.

Quá trình biên soạn:

Trên cơ sở tham khảo các giáo trình, tài liệu về cơ sở dữ liệu và quản trị cơ sở dữ liệu kết hợp với thực tế nghề nghiệp của nghề Công nghệ thông tin (Ứng dụng phần mềm), giáo trình này được biên soạn có sự tham gia tích cực của các giáo viên có kinh nghiệm, cùng với những ý kiến đóng góp quý báu của các chuyên gia về lĩnh vực công nghệ thông tin.

Mối quan hệ của tài liệu với chương trình, mô đun/môn học:

Giáo trình cung cấp kiến thức về cách quản trị cơ sở dữ liệu với SQL, phục vụ cho việc tạo, truy xuất, cập nhật dữ liệu và lưu trữ.

Để học được mô đun này người học phải học qua các môn học cơ sở. Để học hiệu quả hơn, người học cần đến lớp nghe giáo viên giảng về kinh nghiệm, kiến thức logic các thành phần đối với một cơ sở dữ liệu.

Cấu trúc chung của giáo trình Quản trị cơ sở dữ liệu với SQL Server bao gồm 9 bài:

- Bài 1: Tổng quan về SQL Server
- Bài 2: Bộ công cụ Management Studio
- Bài 3: Định nghĩa dữ liệu với T-SQL
- Bài 4: Thao tác dữ liệu với T-SQL
- Bài 5: Bảo mật cơ sở dữ liệu SQL Server.
- Bài 6: Thủ tục lưu trữ.
- Bài 7: Hàm do người dùng định nghĩa.
- Bài 8: Trigger.
- Bài 9: Quản lý giao dịch và khóa

Lời cảm ơn

Giáo trình được biên soạn trên cơ sở các văn bản quy định của Nhà nước và tham khảo nhiều tài liệu liên quan có giá trị. Song chắc hẳn quá trình biên soạn không tránh khỏi những thiếu sót nhất định. Ban biên soạn mong muốn và thực sự cảm ơn những ý kiến nhận xét, đánh giá của các chuyên gia, các thầy cô đóng góp cho việc chỉnh sửa để giáo trình ngày một hoàn thiện hơn.

Đà Lạt, ngày 07 tháng 7 năm 2017

Tham gia biên soạn

1. Chủ biên Trương Thị Thanh Thảo
2. Ngô Thiên Hoàng
3. Phạm Đình Nam
4. Nguyễn Quỳnh Nguyên
5. Phan Ngọc Bảo

MỤC LỤC

Nội dung	Trang
LỜI GIỚI THIỆU	3
BÀI 1: TỔNG QUAN VỀ SQL SERVER.....	12
1. GIỚI THIỆU.....	12
1.1 Hệ quản trị cơ sở dữ liệu ms sql server.....	12
1.2 Mô hình dữ liệu quan hệ	15
2. KIẾN TRÚC TRUY CẬP CƠ SỞ DỮ LIỆU	17
3. GIỚI THIỆU MỘT SỐ DỊCH VỤ CỦA SQL SERVER	18
4. NGÔN NGỮ LẬP TRÌNH TRONG SQL SERVER	19
BÀI 2: BỘ CÔNG CỤ MANAGEMENT STUDIO.....	20
1. CÀI ĐẶT MS SQL SERVER 2008.....	20
1.1 Cài đặt .Net Framework 3.5.....	20
1.2 Cài đặt bản nâng cấp Windows.....	22
1.3 Cài đặt SQL Server 2008	23
1.4 Khởi động SQL server 2008	27
2. MỘT SỐ THAO TÁC VỚI SQL SERVER 2008.....	28
2.1 Đăng nhập SQL Server	29
2.2 Tạo và thực thi các câu truy vấn với trình soạn thảo trên dữ liệu mẫu, kịch bản.29	
3. QUẢN LÝ DATABASE.....	31
3.1 Tạo mới một Database	31
3.2 Sao lưu Database (Backup).....	36
3.3 Xóa Database (Delete)	41
3.4 Phục hồi Database (Restore).....	41
3.5 Gỡ bỏ Database (Detach).....	44
3.6 Đính kèm Database (Attach).....	45
4. CÂU HỎI ÔN TẬP.....	47
BÀI 3: ĐỊNH NGHĨA DỮ LIỆU VỚI T-SQL.....	48
1. NGÔN NGỮ ĐỊNH NGHĨA DỮ LIỆU.....	49
1.1 TẠO VÀ SỬA ĐỔI CẤU TRÚC CƠ SỞ DỮ LIỆU.....	49
1.1.1 Tạo tập tin cơ sở dữ liệu.	49

1.1.2 Khai báo cơ sở dữ liệu đã có vào MS SQL Server.....	49
1.1.3 Đổi tên cơ sở dữ liệu	50
1.1.4 Tăng dung lượng tập tin dữ liệu.....	50
1.1.5 Dời vị trí tập tin .mdf:	50
1.1.6 Cấu hình CSDL chỉ đọc thì dùng SET.....	50
1.1.7 Chỉ định cơ sở dữ liệu được dùng cho các lệnh truy vấn.	50
1.1.8 Xóa cơ sở dữ liệu.....	51
1.1.9 Lưu ý về Quyền tạo CSDL :	51
1.2 TẠO BẢNG TRONG CƠ SỞ DỮ LIỆU.	51
1.2.1 Các ràng buộc toàn vẹn trong tạo bảng:	51
1.2.2 Một số kiểu dữ liệu thường dùng.	52
1.2.3 Lệnh tạo bảng.....	53
1.2.4 Lệnh xóa bảng	55
1.2.5 Lệnh Thêm, Xóa, Sửa cột	55
1.2.6 Lệnh Thêm, Xóa, Sửa, Vô hiệu hóa ràng buộc.	56
2. CÂU HỎI ÔN TẬP.....	57
BÀI 4: THAO TÁC DỮ LIỆU VỚI T-SQL.....	58
1. NGÔN NGỮ THAO TÁC CẬP NHẬT DỮ LIỆU (action query, data modification query).	58
1.1 Thêm dữ liệu	58
1.2 Xóa dữ liệu	59
1.3 Sửa dữ liệu	60
2. NGÔN NGỮ TRUY VẤN DỮ LIỆU	60
2.1 Truy vấn chọn (select query)	60
2.2 Truy vấn nhóm dữ liệu (Select query có group by)	64
2.3 Truy vấn lồng nhau (nested query, sub query)	65
2.4 Truy vấn hợp (union [all]), trừ (Except), giao (Intersect).....	67
3. CẬP NHẬT DỮ LIỆU ĐƯỢC TỔNG HỢP TỪ NHỮNG DỮ LIỆU CÓ SẴN	68
3.1 Thêm một hay nhiều bộ giá trị vào một bảng từ bảng có sẵn.....	68
3.2 Tạo một bảng mới với các bộ giá trị được lấy từ một bảng đã có sẵn.....	69
4. CÂU HỎI ÔN TẬP.....	70
BÀI 5: BẢO MẬT CƠ SỞ DỮ LIỆU TRONG MS SQL SERVER.....	71

1. TẠO VÀ CẤP QUYỀN TÀI KHOẢN ĐĂNG NHẬP SQL SERVER.....	71
1.1 Tạo tài khoản đăng nhập	74
1.1.1 Tạo tài khoản đăng nhập SQL Server.....	74
1.1.2 Tạo tài khoản đăng nhập bằng phát biểu SQL	76
1.1.3 Tạo tài khoản truy cập cơ sở dữ liệu.	78
2. CẤP QUYỀN NGƯỜI SỬ DỤNG TRÊN ĐỐI TƯỢNG CƠ SỞ DỮ LIỆU.81	
2.1 Cấp quyền người sử dụng trên đối tượng cơ sở dữ liệu bằng SQL Server...	81
2.2 Cấp quyền người sử dụng trên đối tượng cơ sở dữ liệu bằng lệnh SQL	82
3. CẤP QUYỀN NGƯỜI SỬ DỤNG TRÊN DỮ LIỆU	83
3.1 Cấp quyền người sử dụng trên dữ liệu.....	83
3.2 Cấp quyền người sử dụng trên dữ liệu bằng ngôn ngữ SQL	85
4. CÂU HỎI ÔN TẬP.....	85
BÀI 6 : THỦ TỤC LƯU TRỮ.....	87
1. CÂU LỆNH ĐIỀU KHIỂN.....	87
1.1 Khởi lệnh BEGIN ... END	89
1.2 Lệnh GOTO và RETURN	89
1.3 Lệnh IF ELSE	90
1.4 Vòng lặp WHILE	90
1.5 Lệnh rẽ nhánh CASE	91
2. TẠO THỦ TỤC LƯU TRỮ (STORED PROCEDURE).....	93
2.1 Khái niệm:.....	93
2.2 Tạo thủ tục lưu trữ.....	94
2.3 Lời gọi thủ tục lưu trữ.....	96
2.4 Định nghĩa biến trong thủ tục lưu trữ	97
2.4.1 Các biến cục bộ và các biến toàn cục.....	97
2.4.2 Sử dụng biến toàn cục.....	97
2.4.3 Khai báo và khởi tạo giá trị cho các biến cục bộ.....	99
2.4.4 Tham số với giá trị mặc định.....	100
2.5 Giá trị trả về trong thủ tục lưu trữ.....	101
2.6 Sửa và xóa thủ tục lưu trữ.....	103
2.6.1 Sửa đổi thủ tục lưu trữ	103
2.6.2 Xóa thủ tục lưu trữ.....	103

3. CÂU HỎI ÔN TẬP THỦ TỤC.....	104
BÀI 7: HÀM DO NGƯỜI DÙNG ĐỊNH NGHĨA.....	105
1. HÀM VÔ HƯỚNG - SCALAR UDF	105
1.1 Sử dụng hàm.....	105
1.1.1 Sử dụng các hàm hệ thống	105
1.2 Hàm vô hướng.....	111
2. HÀM NỘI TUYẾN - INLINE UDF	113
3. HÀM BAO GỒM NHIỀU CÂU LỆNH BÊN TRONG – MULTI STATEMENT UDF	115
4. SỬA VÀ XÓA HÀM	117
4.1 Sửa đổi hàm.....	117
4.2 Xóa hàm	117
5. THỰC THI CÁC HÀM DO NGƯỜI DÙNG ĐỊNH NGHĨA.....	117
6. CÂU HỎI ÔN TẬP HÀM.....	118
BÀI 8: TRIGGER.....	119
1. GIỚI THIỆU VỀ TRIGGER	119
1.1 Các đặc điểm của trigger	119
1.2 Các trường hợp sử dụng trigger.....	119
1.3 Khả năng của trigger	120
2. TẠO, THAY THẾ, LOẠI BỎ TRIGGER	120
2.1 Tạo trigger	120
2.2 Sửa trigger	125
2.3 Xóa trigger	125
3. KÍCH HOẠT TRIGGER DỰA TRÊN SỰ THAY ĐỔI DỮ LIỆU TRÊN CỘT (IF UPDATE).	125
4. SỬ DỤNG TRIGGER VÀ GIAO TÁC (TRANSACTION).....	128
4.1 Giao tác:	128
4.2 Ddl trigger	129
5. VÔ HIỆU HÓA HAY KÍCH HOẠT TRIGGER (DISABLE/ENABLE TRIGGER).....	131
6. CÂU HỎI ÔN TẬP.....	131
BÀI 9: QUẢN LÝ GIAO DỊCH VÀ KHÓA.....	133
1. GIAO DỊCH.....	133

1.1	Khái niệm.....	133
1.2	Các thuộc tính của giao dịch.....	134
1.3	Các kiểu giao dịch.....	135
1.3.1	<i>Giao dịch tường minh</i>	135
1.3.2	<i>Giao dịch ngầm định</i>	135
1.3.3	<i>Giao dịch xác nhận tự động</i>	136
1.4	Các cấp độ giao dịch riêng.....	136
1.4.1	<i>ReadUncommitted</i>	136
1.4.2	<i>ReadCommitted</i>	136
1.4.3	<i>RepeatableRead</i>	136
1.4.4	<i>Serializable</i>	137
1.5	Cách sử dụng tệp nhật ký giao dịch (transaction log)	137
1.6	Mô tả giao dịch bằng câu lệnh T-SQL.....	138
1.6.1	<i>BEGIN TRANSACTION</i>	138
1.6.2	<i>COMMITTRANSACTION</i>	139
1.6.3	<i>ROLLBACK TRANSACTION</i>	139
2.	KHÓA.....	140
2.1	Các kiểu của khoá	140
2.1.1	<i>Khoá bi quan (Pessimistic Lock):</i>	140
2.1.2	<i>Khoá tối ưu (Optimistic Lock):</i>	140
2.1.3	<i>Các khoá dùng chung (Shared Locks):</i>	141
2.1.4	<i>Các khoá dành riêng (Exclusive Locks):</i>	141
2.1.5	<i>Các khoá cập nhật (Update Locks):</i>	141
2.2	Khoá chết (Deadlock)	141
2.3	Giải quyết vấn đề Deadlock (Resolving Deadlock)	142
	LUYỆN TẬP 1	145
	LUYỆN TẬP 2	150

GIÁO TRÌNH MÔ ĐUN QUẢN TRỊ CƠ SỞ DỮ LIỆU VỚI SQL SERVER

Tên mô đun: QUẢN TRỊ CƠ SỞ DỮ LIỆU VỚI SQL SERVER

Mã mô đun: MD15

I. Vị trí, tính chất của mô đun:

1. Vị trí: Quản trị cơ sở dữ liệu với SQL Server là mô đun chuyên môn đào tạo trình độ Cao đẳng Công nghệ thông tin (Ứng dụng phần mềm). Mô đun này được bố trí sau các môn học: Cơ sở dữ liệu.
2. Tính chất: Quản trị cơ sở dữ liệu với SQL Server là mô đun chuyên môn bắt buộc.

II. Mục tiêu mô đun:

1. Về kiến thức:

- Trình bày được vai trò của hệ quản trị cơ sở dữ liệu trong việc khai thác và quản trị cơ sở dữ liệu;
- Trình bày được khái niệm về mô hình dữ liệu quan hệ: bảng/quan hệ, bản ghi/dòng dữ liệu, thuộc tính (attributes), miền giá trị (domain), khóa (primary key, foreign key);
- Nêu được các mô hình truy nhập dữ liệu SQL Server;
- Trình bày được kiến trúc bên trong của SQL Server 2008 trở lên: các thành phần và dịch vụ;
- Nêu được các ngôn ngữ lập trình tích hợp trong SQL Server 2008 trở lên;
- Trình bày được cách xây dựng hàm, thủ tục, trigger;
- Giải thích được vai trò của khoá trong các giao dịch;

2. Về kỹ năng:

- Cài đặt và cấu hình được SQL Server 2008 trở lên;
- Thực hiện thành thạo các thao tác quản trị tài khoản người dùng và tài khoản nhóm đối với hệ thống SQL Server 2008 trở lên;
- Bảo vệ tài nguyên dữ liệu trên các hệ thống SQL Server 2008 trở lên;
- Xây dựng chính xác các điều kiện truy vấn;
- Thiết kế được một số truy vấn cơ sở dữ liệu bằng các công cụ;
- Xây dựng được một số hàm, thủ tục và trigger giúp cho việc truy cập, khai thác cơ sở dữ liệu được hiệu quả hơn;
- Quản lý được các giao dịch của hệ thống;
- Sử dụng được một số dịch vụ của SQL Server 2008 trở lên;

3. Về năng lực tự chủ và trách nhiệm:

- Có khả năng tự nghiên cứu, tự học, tham khảo tài liệu liên quan đến môn học để vận dụng vào hoạt động học tập.
- Vận dụng được các kiến thức tự nghiên cứu, học tập và kiến thức, kỹ năng đã được học để hoàn thiện các kỹ năng liên quan đến môn học một cách khoa học, đúng quy định.

III. Nội dung mô đun:

BÀI 1: TỔNG QUAN VỀ SQL SERVER

Mã bài: MĐ15_01

Giới thiệu:

Bài học này giúp sinh viên hiểu được kiến trúc bên trong của hệ quản trị cơ sở dữ liệu SQL Server thấy được vai trò của SQL Server trong các ứng dụng quản trị dữ liệu.

Mục tiêu:

- Trình bày được sự tiến hoá của hệ quản trị cơ sở dữ liệu SQL Server;
- Trình bày vai trò của SQL Server trong các ứng dụng quản trị dữ liệu;
- Nêu được kiến trúc bên trong của hệ quản trị cơ sở dữ liệu SQL Server;
- Rèn luyện thái độ làm việc cẩn thận, khoa học, chính xác.

1. GIỚI THIỆU

1.1 Hệ quản trị cơ sở dữ liệu ms sql server

Hệ quản trị cơ sở dữ liệu (tiếng Anh: Database Management System - DBMS), là phần mềm hay hệ thống được thiết kế để quản trị một cơ sở dữ liệu. Cụ thể, các chương trình thuộc loại này hỗ trợ khả năng lưu trữ, sửa chữa, xóa và tìm kiếm thông tin trong một cơ sở dữ liệu (CSDL). Có rất nhiều loại hệ quản trị CSDL khác nhau: từ phần mềm nhỏ chạy trên máy tính cá nhân cho đến những hệ quản trị phức tạp chạy trên một hoặc nhiều siêu máy tính.

Tuy nhiên, đa số hệ quản trị CSDL trên thị trường đều có một đặc điểm chung là sử dụng ngôn ngữ truy vấn theo cấu trúc mà tiếng Anh gọi là Structured Query Language (SQL). Các hệ quản trị CSDL phổ biến được nhiều người biết đến là MySQL, Oracle, PostgreSQL, SQL Server, DB2, Infomix, v.v. Phần lớn các hệ quản trị CSDL kể trên hoạt động tốt trên nhiều hệ điều hành khác nhau như Linux, Unix và MacOS ngoại trừ SQL Server của Microsoft chỉ chạy trên hệ điều hành Windows.

- Ưu điểm của HQTCSDL:

- + Quản lý được dữ liệu dư thừa.
- + Đảm bảo tính nhất quán cho dữ liệu.
- + Tạo khả năng chia sẻ dữ liệu nhiều hơn.
- + Cải tiến tính toàn vẹn cho dữ liệu ...

- Nhược điểm:

- + HQTCSDDL tốt thì khá phức tạp.
- + HQTCSDDL tốt thường rất lớn, chiếm nhiều dung lượng bộ nhớ.
- + Giá cả khác nhau tùy theo môi trường cài đặt sử dụng và chức năng nâng cao trong HQTCSDDL.
- + HQTCSDDL được viết tổng quát cho nhiều người dùng thì thường chậm.

Microsoft SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (relational database management system – RDBMS) do Microsoft phát triển.

SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ mạng máy tính hoạt động theo mô hình khách chủ cho phép đồng thời cùng lúc có nhiều người dùng truy xuất đến dữ liệu, quản lý việc truy nhập hợp lệ và các quyền hạn của từng người dùng trên mạng.

Ngôn ngữ truy vấn quan trọng của Microsoft SQL server là Transact-SQL

Lần lượt các phiên bản của Microsoft SQL Server đã ra đời, từ 4.2 được nâng cấp thành 4.21, 6.0, 6.5, 7.0. Các hệ thống cũ đang dùng phổ biến hệ quản trị Microsoft SQL Server phiên bản 2000, 2005.

- **SQL Server 2005:** SQL Server 2005, được phát hành vào tháng 11 năm 2005, là phiên bản tiếp theo của SQL Server 2000. Sử dụng trên nền Net Framework 2.0

- **SQL Server 2008:** Đây là phiên bản mới của SQL Server. Ngày 27/02/2008, Sử dụng trên nền Net Framework 3.0 (3.5 đối với phiên bản R2). Hiện nay, MS SQL Server phiên bản 2008 được kết hợp vào sản xuất phần mềm. Mới nhất là phiên bản Microsoft SQL Server 2012.

Những điểm mới trong SQL Server 2008: SQL Server 2008 giới thiệu 4 lĩnh vực chính trong toàn cảnh nền tảng dữ liệu của Microsoft:

- **Nền tảng cho các nhiệm vụ then chốt** - SQL Server 2008 cho phép các tổ chức có thể chạy hầu hết các ứng dụng phức tạp của họ trên một nền tảng an toàn, tin cậy và có khả năng mở rộng, bên cạnh đó còn giảm được sự phức tạp trong việc quản lý cơ sở hạ tầng dữ liệu. SQL Server 2008 cung cấp một nền tảng tin cậy và an toàn bằng cách bảo đảm những thông tin có giá trị trong các ứng dụng đang tồn tại và nâng cao khả năng sẵn có của dữ liệu. SQL Server 2008 giới thiệu một cơ chế quản lý cách tân dựa trên chính

sách, cơ chế này cho phép các chính sách có thể được định nghĩa, quản trị tự động cho các thực thể máy chủ trên một hoặc nhiều máy chủ. Thêm vào đó, SQL Server 2008 cho phép thi hành truy vấn dự báo với một nền tảng tối ưu.

- **Sự phát triển động** - SQL Server 2008 cùng với .NET Framework đã giảm được sự phức tạp trong việc phát triển các ứng dụng mới. ADO.NET Entity Framework cho phép các chuyên gia phát triển phần mềm có thể nâng cao năng suất bằng làm việc với các thực thể dữ liệu logic đáp ứng được các yêu cầu của doanh nghiệp thay vì lập trình trực tiếp với các bảng và cột. Các mở rộng của ngôn ngữ truy vấn tích hợp (LINQ) mới trong .NET Framework đã cách mạng hóa cách các chuyên gia phát triển truy vấn dữ liệu bằng việc mở rộng Visual C#® và Visual Basic® .NET để hỗ trợ cú pháp truy vấn giống SQL vốn đã có. Hỗ trợ cho các hệ thống kết nối cho phép chuyên gia phát triển xây dựng các ứng dụng cho phép người dùng mang dữ liệu cùng với ứng dụng này vào các thiết bị và sau đó đồng bộ dữ liệu của chúng với máy chủ trung tâm.
- **Dữ liệu quan hệ mở rộng** - SQL Server 2008 cho phép các chuyên gia phát triển khai thác triệt để và quản lý bất kỳ kiểu dữ liệu nào từ các kiểu dữ liệu truyền thống đến dữ liệu không gian địa lý mới.
- **Thông tin trong toàn bộ doanh nghiệp** - SQL Server 2008 cung cấp một cơ sở hạ tầng có thể mở rộng, cho phép quản lý các báo cáo, phân tích với bất kỳ kích thước và sự phức tạp nào, bên cạnh đó nó cho phép người dùng dễ dàng hơn trong việc truy cập thông tin thông qua sự tích hợp sâu hơn với Microsoft Office. Điều này cho phép CNTT đưa được thông tin của doanh nghiệp rộng khắp trong tổ chức. SQL Server 2008 tạo những bước đi tuyệt vời trong việc lưu trữ dữ liệu, cho phép người dùng hợp nhất các trung tâm dữ liệu vào một nơi lưu trữ dữ liệu tập trung của toàn doanh nghiệp.



1.2 Mô hình dữ liệu quan hệ

Nền tảng của cấu trúc cơ sở dữ liệu là mô hình dữ liệu. Mô hình dữ liệu được định nghĩa là một sưu tập các ký hiệu, công cụ khái niệm dùng cho việc mô tả dữ liệu, mối quan hệ dữ liệu, các ràng buộc toàn vẹn dữ liệu trong CSDL và một tập hợp các phép toán thao tác trên dữ liệu.

Các hệ quản trị cơ sở dữ liệu quan hệ ngày nay được xây dựng dựa vào lý thuyết của mô hình quan hệ và áp dụng nó để lập mô hình dữ liệu quan hệ có hiệu quả trong lưu trữ và khai thác dữ liệu.

Chẳng hạn với bài toán quản lý điểm thi của sinh viên, nếu lưu trữ dữ liệu theo dạng bảng với các cột MaSV, HoTen, MonHoc, TenKhoa, Diem thì các giá trị của các cột HoTen, MonHoc, TenKhoa sẽ bị trùng lặp. Sự trùng lặp này gây nên một số vấn đề:

- + Ta không thể lưu trữ một sinh viên mới khi sinh viên này chưa có điểm thi
- + Khi cần sửa đổi họ tên sinh viên thì ta phải sửa tất cả các dòng có liên quan đến sinh viên này. Điều này dễ gây ra tình trạng dữ liệu thiếu nhất quán.

+ Khi có nhu cầu xóa điểm thi của một sinh viên kéo theo khả năng xóa luôn họ tên sinh viên đó.

MaSV	HoTen	MonHoc	TenKhoa	Diem
99001	TRAN DAN THU	CO SO DU LIEU	CONG NGHE THONG TIN	3.0
99002	NGUYEN HA DA THAO	CO SO DU LIEU	CONG NGHE THONG TIN	8.0
99001	TRAN DAN THU	TIN HOC VAN PHONG	CONG NGHE THONG TIN	6.0
99005	LE THANH TRUNG	TIN HOC VAN PHONG	ANH VAN	5.0

Việc lưu trữ dữ liệu như trên không đúng với mô hình quan hệ. Để lưu trữ đúng với mô hình quan hệ ta phải thay MonHoc bằng MaMH, thay TenKhoa bằng MaKhoa, tách một bảng dữ liệu lớn đó ra thành nhiều bảng con, như mô hình dưới.

MaSV	MaMH	Diem
99001	CSDL	3.0
99002	CSDL	8.0
99001	THVP	6.0
99005	THVP	5.0

MaSV	HoTen	MaKhoa
99001	TRAN DAN THU	CNTT
99002	NGUYEN HA DA THAO	CNTT
99005	LE THANH TRUNG	AV

MaMH	TenMH	SoTiet	MaKhoa	TenKhoa
CSDL	CO SO DU LIEU	90	CNTT	CONG NGHE THONG TIN
THVP	TIN HOC VAN PHONG	90	AV	ANH VAN

2. KIẾN TRÚC TRUY CẬP CƠ SỞ DỮ LIỆU

Dữ liệu được tạo ra trong các hoạt động nghiệp vụ hàng ngày (dữ liệu có thể dạng text, image, video, audio...). Ví dụ dữ liệu của Ngân hàng sẽ là các tài khoản, các giao dịch...dữ liệu sẽ được lưu trong cơ sở dữ liệu được quản lý bởi các hệ quản trị cơ sở dữ liệu, ví dụ như: Microsoft SQL Server, Oracle, MySQL, Access...Hệ quản trị dữ liệu có thể được cài đặt và chạy trên cùng một máy với phần mềm ứng dụng hoặc hệ quản trị cơ sở dữ liệu và phần mềm ứng dụng được cài đặt trên hai hoặc nhiều máy khác nhau. Do đó có 3 kiểu kiến trúc truy cập cơ sở dữ liệu từ một ứng dụng:

- **Kiến trúc một tầng (single – tier architecture):** Kiến trúc này được sử dụng để truy cập một cơ sở dữ liệu cục bộ, nghĩa là phần mềm ứng dụng và hệ quản trị cơ sở dữ liệu cùng chạy trên một máy.
- **Kiến trúc hai tầng(two – tier architecture):** Kiến trúc này còn gọi là kiến trúc client/server, cơ sở dữ liệu sẽ nằm ở một máy, ứng dụng sẽ cài đặt ở một máy, ứng dụng sẽ kết nối đến cơ sở dữ liệu thông qua các trình điều khiển ví dụ như ODBC (Open Database Connectivity).
- **Kiến trúc ba tầng (three tier architecture):** gọi là kiến trúc đa tầng, ứng dụng kết nối đến máy cài đặt application server (ví dụ Glassfish, Jboss,...), máy application server lại kết nối đến máy database server.

Để có thể thực hiện được các kiến trúc truy cập dữ liệu như trên thì một loạt các mô hình truy cập dữ liệu ra đời.

Mô hình	Đặc điểm
DAO (Data Access Object)	là mô hình đầu tiên, dùng để truy cập cơ sở dữ liệu trong các hệ quản trị cơ sở dữ liệu sử dụng công nghệ Microsoft Jet Database Engine ví dụ như Access, kiểu truy cập theo kiến trúc một tầng(tức truy cập đến các cơ sở dữ liệu cục bộ).
RDO	Mô hình này cung cấp các đối tượng cho phép kết nối đến cơ sở dữ liệu thực thi các truy vấn, cập nhật dữ liệu theo kiến trúc 2 tầng
ADO	Cung cấp cách thức làm việc với dữ liệu trong hệ quản trị cơ sở theo kiểu kết nối(connected data access) nghĩa là ADO sẽ duy trì kết nối đến cơ sở dữ liệu cho đến khi ứng dụng được đóng lại. ADO không cho phép truyền dữ liệu qua tường lửa(firewall)
ADO.NET	Là mô hình truy cập dữ liệu mới nhất tính đến thời điểm này. ADO.NET hỗ trợ kiểu truy cập dữ liệu ngắt kết nối(Disconnected Database Accesss), nghĩa là chỉ khi nào ứng dụng gửi yêu cầu kết nối đến cơ sở dữ liệu thì kết nối mới được thiết lập, sau khi yêu cầu được xử lý xong kết nối sẽ bị đóng lại. ADO.NET cho phép truyền dữ liệu qua tường lửa vì dữ liệu được lưu ở định dạng XML.

3. GIỚI THIỆU MỘT SỐ DỊCH VỤ CỦA SQL SERVER

Trình quản lý các dịch vụ trên SQL Server như: MSSQL Server, SQL Server Agent, Microsoft Distributed Transaction Coordinator (MSDTC)

MSSQL Server: là RDBMS, xử lý các phát biểu Transact-SQL và quản lý các file lưu trữ các CSDL.

SQL Server Agent: dùng lập lịch thực hiện tự động các công việc như Backup dữ liệu, Replication, ...

Microsoft Distributed Transaction Coordinator (MSDTC): Hỗ trợ cho các ứng dụng Client làm việc với các dữ liệu được phân bố trên nhiều máy tính.

Combo Server: Chứa danh sách các SQL Server có khả năng truy xuất từ máy của bạn. Tuy nhiên, bạn có thể gõ tên máy chủ cần kết nối không có trong danh sách và click nút mũi tên trên hộp Services, tiện ích Services Manager sẽ cố gắng kết nối tới máy chủ ở xa.

Combo Services: chứa tên các dịch vụ được cung cấp bởi SQL Server đã chọn. Mỗi dịch vụ sẽ ở 1 trong 3 trạng thái. Một số dịch vụ không cung cấp chức năng tạm dừng.

Ví dụ: MSSQL Server vẫn tiếp tục hoạt động khi nó tạm dừng, nhưng các kết nối mới thì không được phép.

4. NGÔN NGỮ LẬP TRÌNH TRONG SQL SERVER

Mỗi hệ quản trị CSDL khác nhau có cách cài đặt các ngôn ngữ này khác nhau. Tuy nhiên nhìn chung chúng đều dựa trên một chuẩn nhất định đó là ngôn ngữ truy vấn có cấu trúc SQL – Structured Query Language. SQL là ngôn ngữ truy vấn dựa trên đại số quan hệ. Câu lệnh của SQL dùng để rút trích dữ liệu của một hay nhiều quan hệ. Kết quả của một câu lệnh SQL (truy vấn) là một quan hệ. Để đơn giản trong cách trình bày, ta xem quan hệ mà câu truy vấn sử dụng để tạo ra quan hệ khác gọi là quan hệ nguồn, quan hệ kết quả của truy vấn là quan hệ đích.

SQL chuẩn quy định cách đặt tên đối tượng trong CSDL như sau :

- + Tên đối tượng dài tối đa 32 ký tự và bắt đầu bằng chữ cái, tên không có khoảng trắng hoặc có dấu tiếng việt.

- + Nếu là tên bảng hoặc khung nhìn (view) thì phải là duy nhất trong CSDL và không được trùng với các từ khóa trong ngôn ngữ của hệ quản trị cơ sở dữ liệu.

- + Nếu là tên cột thì trong cùng một bảng không được đặt tên trùng nhau. Tuy nhiên các cột có thể trùng tên nếu nó nằm trong các bảng khác nhau. Nên đặt tên cho các cột mang cùng ý nghĩa ở các bảng.

Ta quy ước gọi tên quan hệ là bảng, thuộc tính quan hệ là cột, các bộ giá trị là các dòng trong bảng để thực hiện trong môn học này.

BÀI 2: BỘ CÔNG CỤ MANAGEMENT STUDIO

Mã bài: MD15_02

Giới thiệu:

Bài học giúp sinh viên cài đặt và cấu hình được phần mềm SQL Server, xây dựng được mô hình quan hệ của một số cơ sở dữ liệu, thiết kế các câu truy vấn từ đơn giản đến phức tạp và xử lý được các lỗi thường gặp khi xây dựng truy vấn.

Mục tiêu:

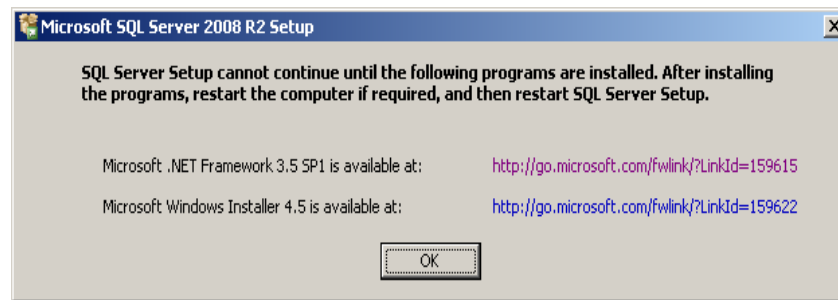
- Trình bày được giao diện đồ họa thao tác đến các đối tượng trong cơ sở dữ liệu SQL Server;
- Trình bày được công cụ thiết kế bảng và thiết kế sơ đồ quan hệ;
- Trình bày được công cụ hỗ trợ thiết kế câu truy vấn;
- Trình bày được công cụ quản lý các script của một dự án;
- Cài đặt và cấu hình được SQL Server;
- Xử lý được các lỗi thường gặp khi xây dựng câu truy vấn bằng công cụ hỗ trợ;
- Xây dựng được mô hình quan hệ của một số cơ sở dữ liệu mẫu;
- Có thái độ làm việc chủ động, tích cực và sáng tạo.

1. CÀI ĐẶT MS SQL SERVER 2008

Đối với các máy tính chưa được cài đặt .Net Framework phiên bản 3.5 trở lên trước khi cài SQL Server 2008 hệ thống yêu cầu cài đặt .Net Framework 3.5 và bản nâng cấp Windows. Trường hợp này thường xảy ra với các máy sử dụng Windows XP hoặc Windows Server từ bản 2003 trở xuống.

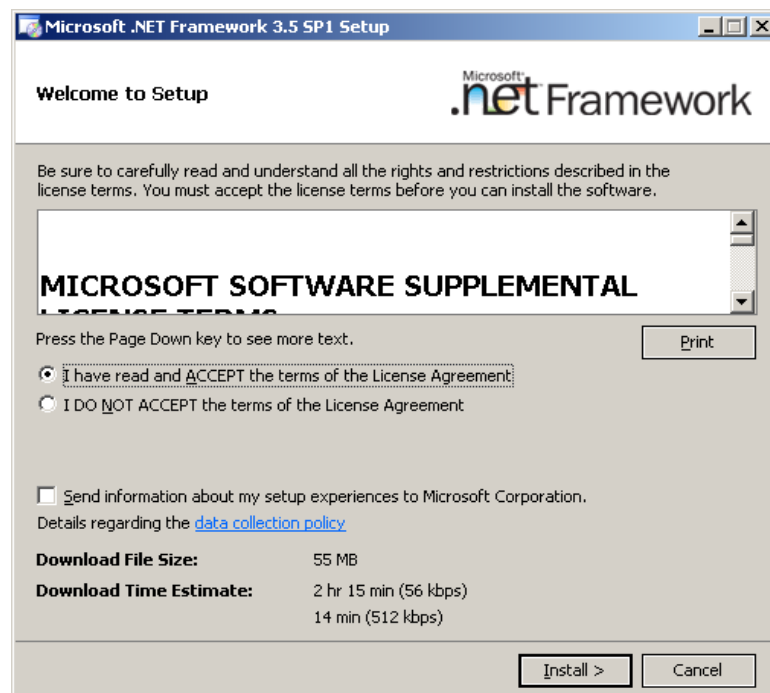
1.1 Cài đặt .Net Framework 3.5

Bước 1: Chạy File cài đặt .Net Framework 3.5 nếu có hoặc tải về từ đường dẫn: <http://go.microsoft.com/fwlink/?LinkId=159615>. Một cách đơn giản hơn bạn hãy nhấp đúp chọn File cài đặt SQL 2008 máy tính sẽ tự động kiểm tra và thông báo như hộp thoại dưới đây:



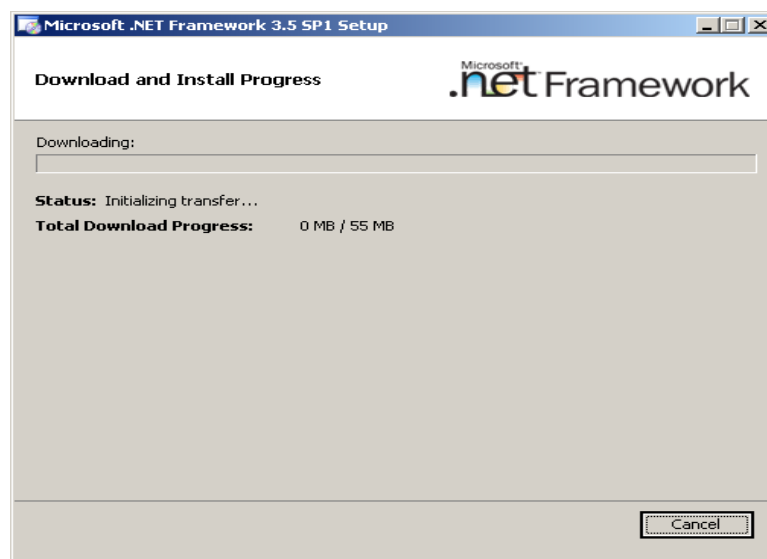
Nhập chọn các đường Link tương ứng để tải File về máy tính

Bước 2: Chờ trong giây lát hộp thoại cài đặt .Net Framework 3.5 xuất hiện như hình dưới đây:

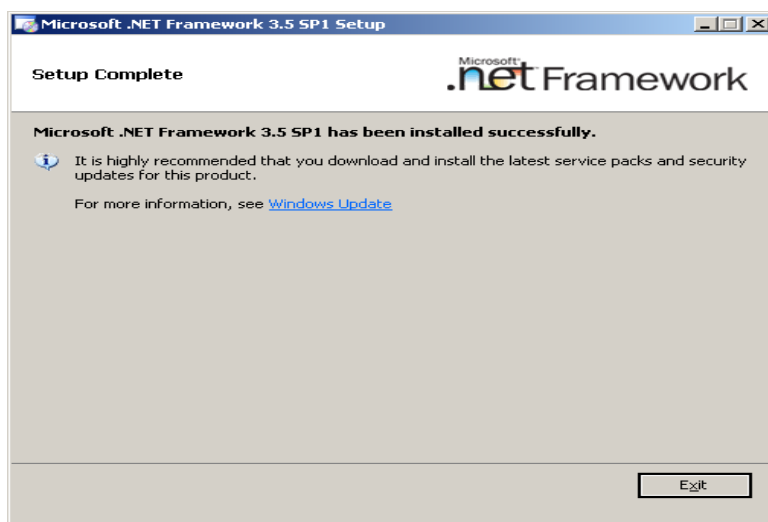


Tích chọn mục “I have read and ACCEPT terms...” sau đó nhấp nút Install.

Bước 3: Hộp thoại tiến trình cài đặt xuất hiện



Bước 4: Chờ máy tính thực hiện tiến trình, cuối cùng nhấp chọn Exit để hoàn tất.

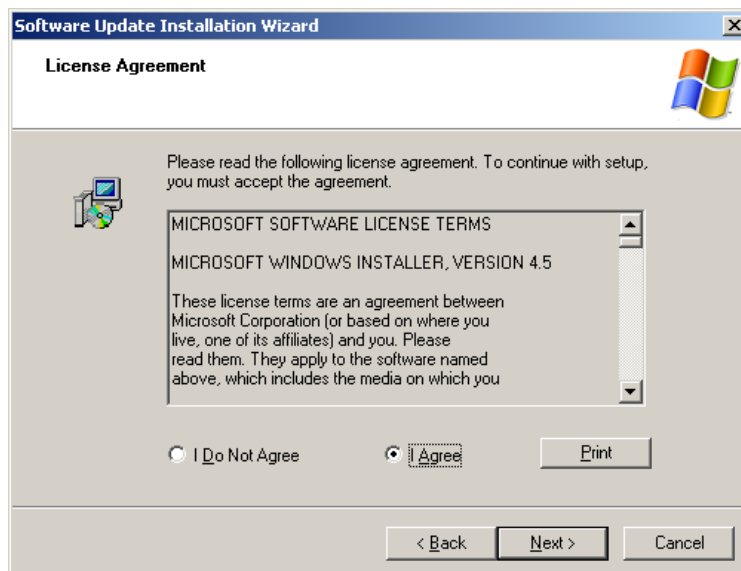


1.2 Cài đặt bản nâng cấp Windows

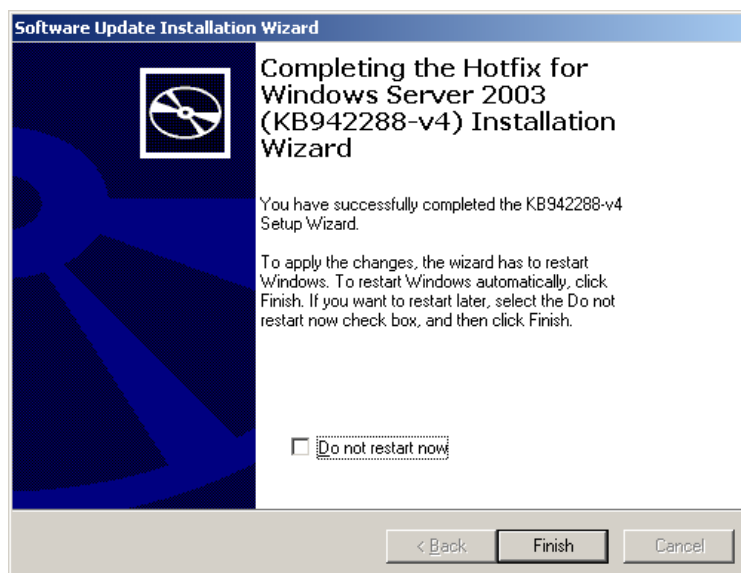
Tương tự như cài đặt .Net Framework 3.5, thực hiện cài bản nâng cấp cho Windows bằng cách nhấp chọn File cài đặt này sau khi đã tải về máy tính. Hộp thoại cài đặt xuất hiện nhấp Next để chuyển sang bước tiếp theo.



Trong hộp thoại thông báo các điều khoản cam kết bản quyền với nhà cung cấp nhấp chọn "I Agree" sau đó nhấp Next để tiến hành cài đặt.



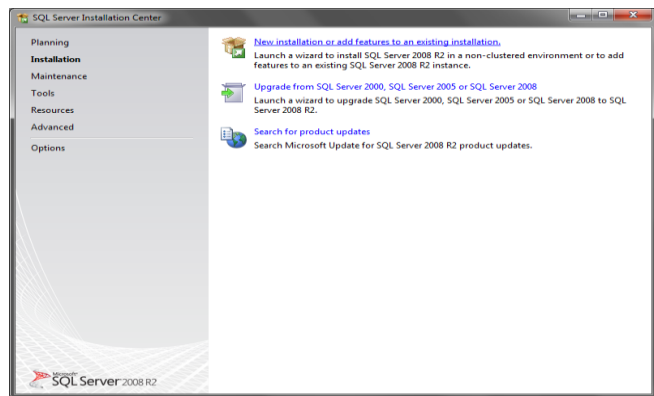
Sau khi cài đặt xong máy tính sẽ xuất hiện thông báo yêu cầu khởi động lại máy tính, hãy nhấp Finish để hoàn tất.



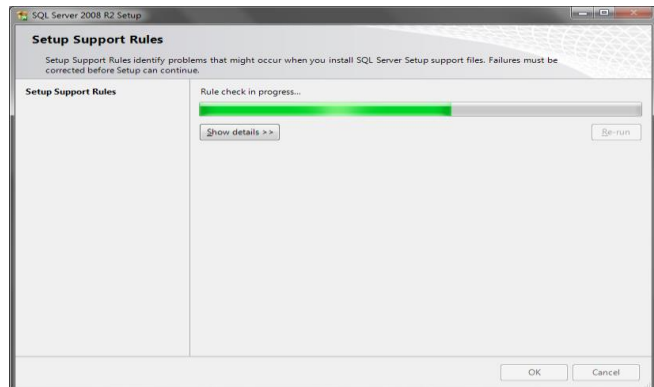
1.3 Cài đặt SQL Server 2008

Sau khi máy tính đã được cập nhật đầy đủ hãy tiến hành cài đặt SQL Server 2008 theo các bước dưới đây:

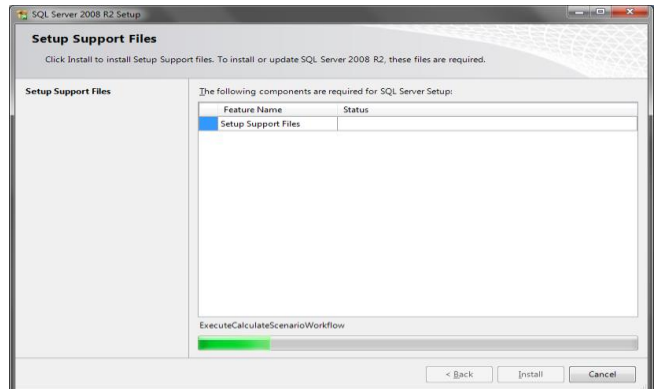
Bước 1: Nhấp đúp chuột vào File setup.exe trong thư mục chứa bộ cài đặt. Hộp thoại SQL Server Installation Center xuất hiện, nhấp chọn New installation or add features to...



Chờ giây lát cho các xử lý hệ thống

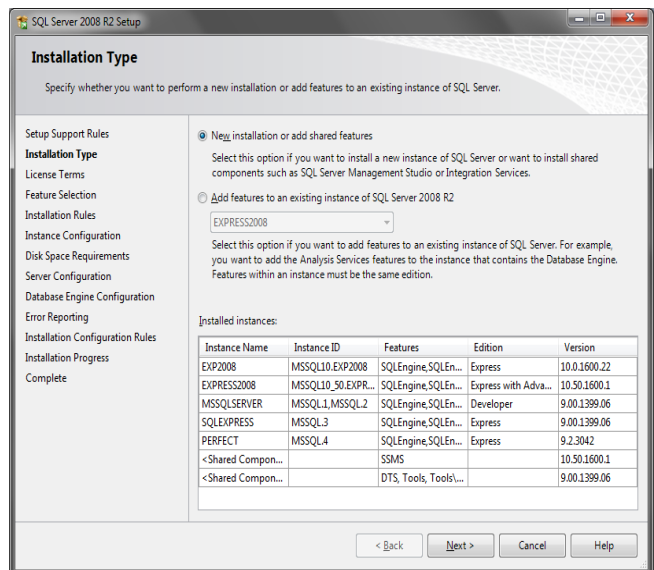


Bước 2: Trong màn hình Setup Support Files nhấp Install để bắt đầu cài đặt



Bước 3: Trong cửa sổ Installation Type:

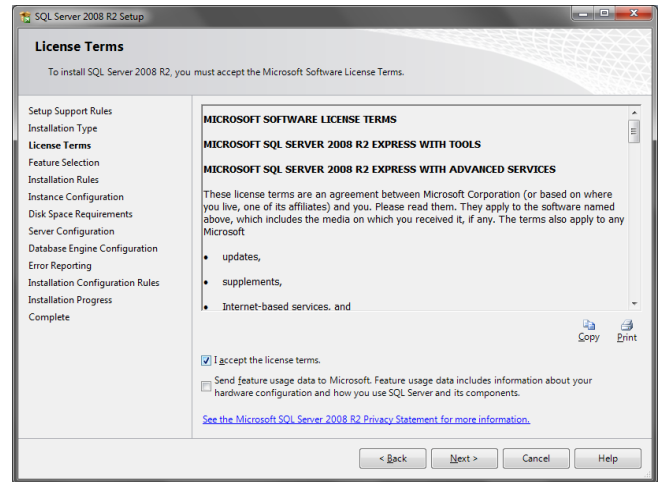
- Tùy chọn New installation or add shared features: Cài đặt SQL trên Instance Name mới (SQL có thể chứa nhiều Instance Name khác nhau trên cùng một hệ thống).
- Tùy chọn Add features to an exiting instance of SQL Server...: Cài đặt SQL trên Instance Name đã



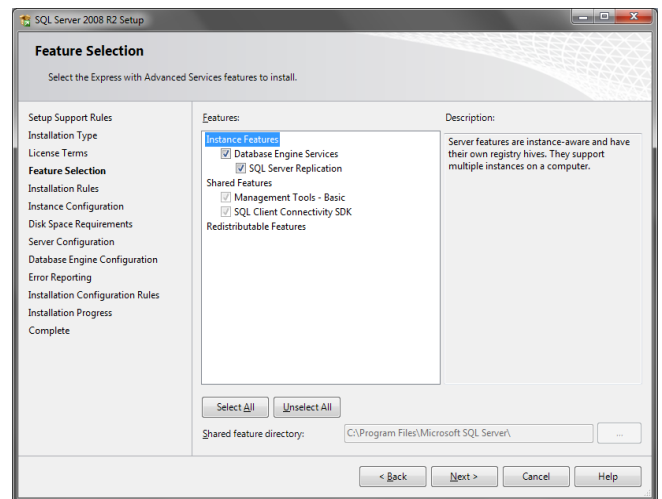
tồn tại.

Nhấp Next sang bước tiếp theo

Bước 4: Tích chọn I accept the license terms

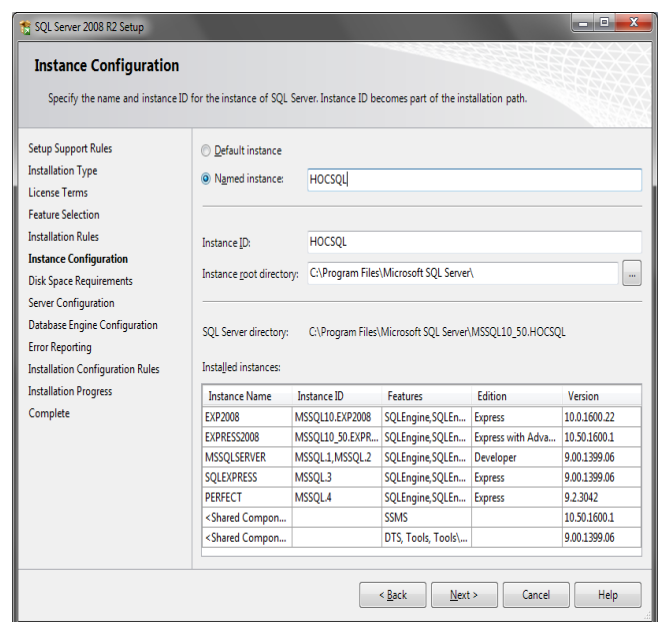


Bước 5: Tích chọn tất cả các mục trong phần Features:



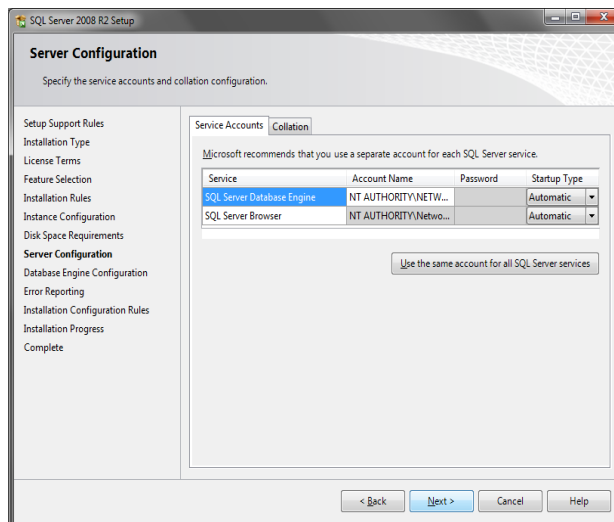
Bước 6: Trong hộp thoại Instance Configuration:

- Default instance: Cài trên Instance Name mặc định.
- Named instance: Chỉ định tên Install Name mới.
- Instance ID: Id của Instance Name (để mặc định)
- Instance root derectory: Thư mục vật lý cài đặt Instance Name (để mặc định)



Thông thường nên chọn mục Named instance và đặt tên cho Instance Name mới như hình ảnh trên. Nhấp Next để chuyển sang bước tiếp theo.

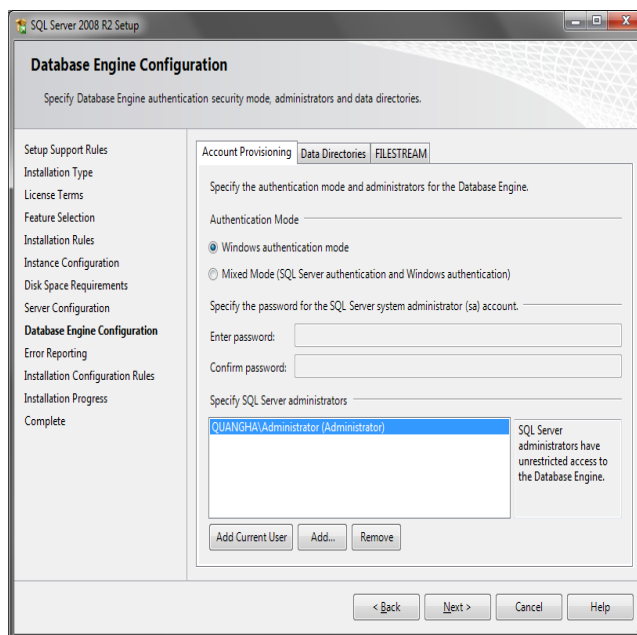
Bước 7: Trong cửa sổ khai báo tài khoản cho các dịch vụ chúng ta để mặc định và nhấp Next để chuyển sang bước tiếp theo.



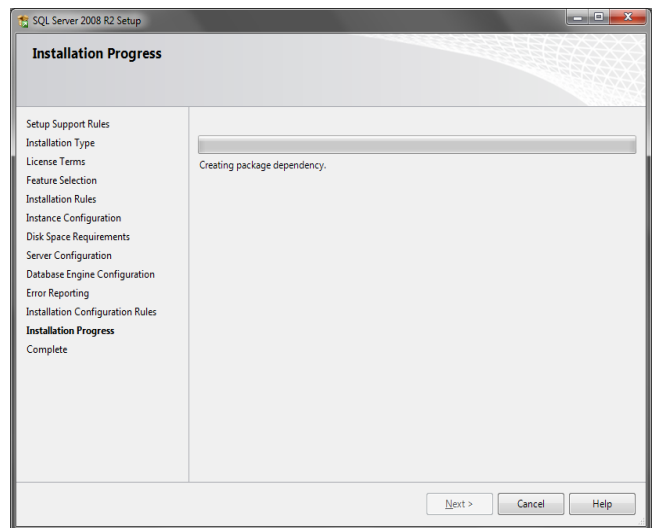
Bước 8: Cửa sổ Database Engine Configuration:

- Windows authentication mode: Truy cập Database với tài khoản của Windows.
- Mixed Mode: Truy cập Database với tài khoản sa của SQL, với tùy chọn này bạn phải đặt tài Password cho tài khoản sa ở mục Enter password và confirm password.

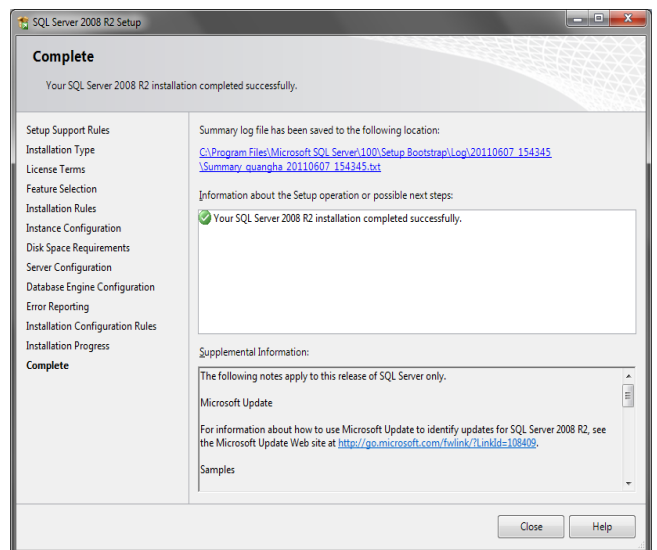
Thông thường chúng ta tích chọn mục Windows authentication Mode, sau đó nhấp Next để chuyển bước tiếp theo.



Bước 9: Nhấp Next để máy tính bắt đầu cài đặt



Bước 10: Chờ máy tính cài đặt trong giây lát, cuối cùng nhấp Close để hoàn tất



1.4 Khởi động SQL server 2008

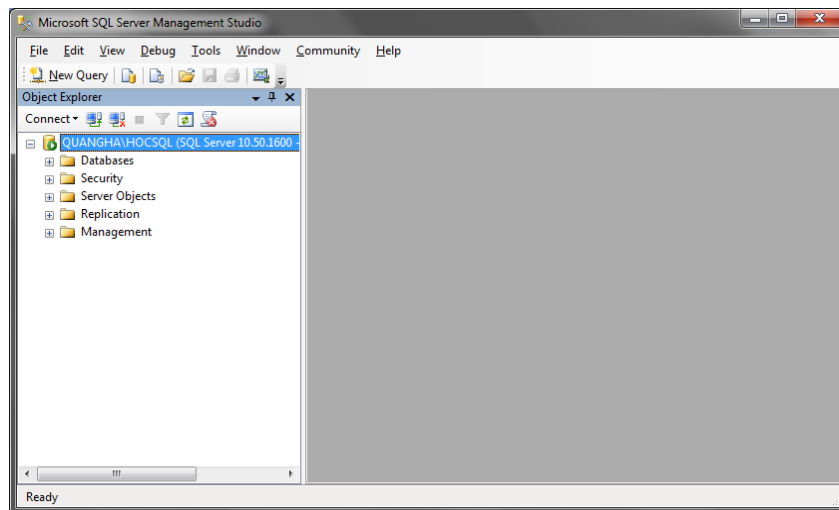
Bước 1: Từ thanh Start nhấp chọn All Program/ Microsoft SQL Server 2008 R2/ SQL Server Management Studio.

Bước 2: Trong mục Server Name nhập đầy đủ tên máy tính và Install Name vừa cài đặt.



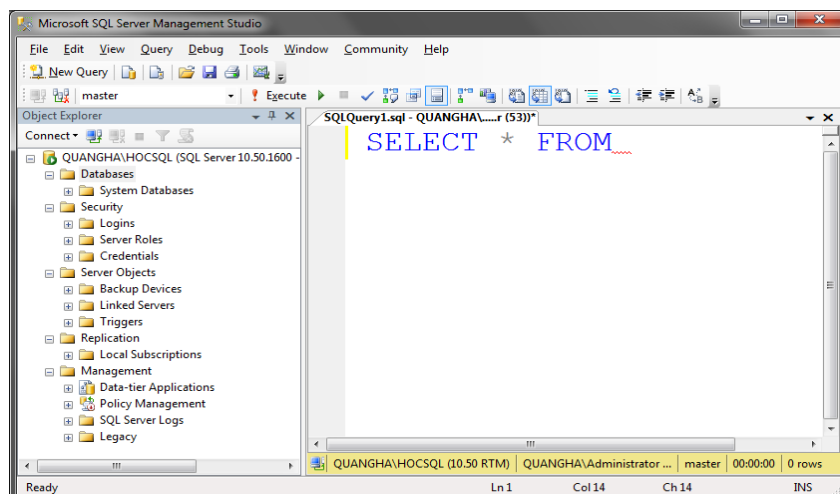
Trong hình ảnh trên chúng ta thấy tên máy tính được ngăn cách với tên Instance Name bằng ký tự “\”.

Bước 3: Tiếp theo nhấp Connect để truy cập vào SQL Server 2008, kết quả như hình dưới đây:



2. MỘT SỐ THAO TÁC VỚI SQL SERVER 2008

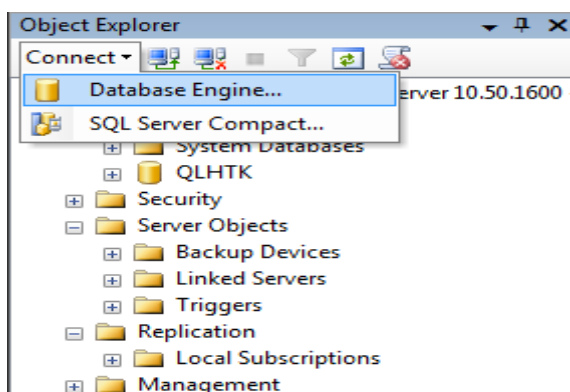
Tìm hiểu giao diện và các thành phần cơ bản



2.1 Đăng nhập SQL Server

Để kết nối tới Server chúng ta thực hiện các bước sau:

Bước 1: Chọn Connect/Database Engine... trong cửa sổ Object Explorer



Bước 2: Hộp thoại Connect to Server xuất hiện trong mục Server name chọn tới Server cần kết nối

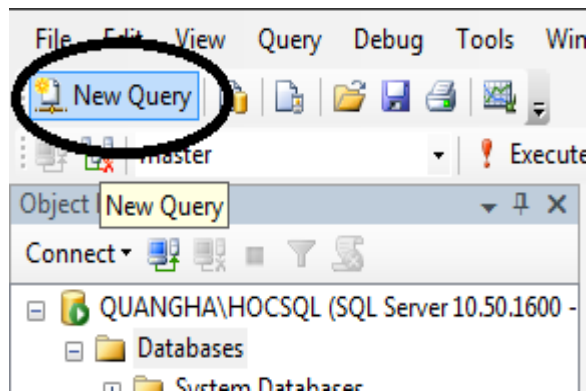


Bước 3: Nhấp Connect để thực hiện kết nối.

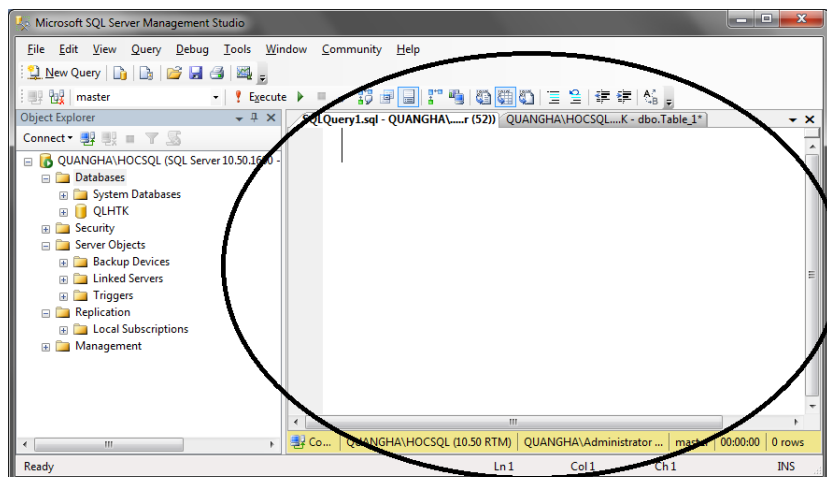
2.2 Tạo và thực thi các câu truy vấn với trình soạn thảo trên dữ liệu mẫu, kịch bản.

Cửa sổ New Query dùng để viết và thực thi các câu lệnh SQL giao tác với các thành phần của Server. Dưới đây là các bước đơn giản để viết và thực thi một câu lệnh SQL đơn giản sử dụng cửa sổ New Query.

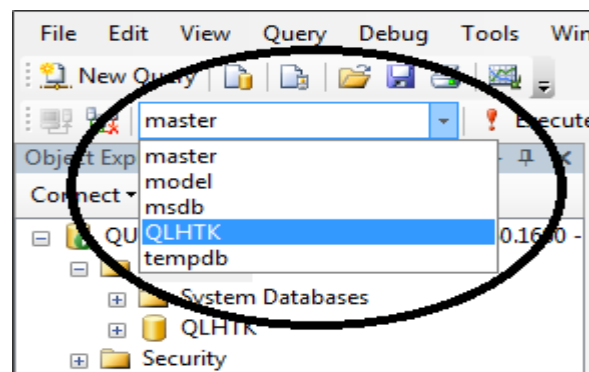
Bước 1: Nhấp chọn biểu tượng New Query trên thanh công cụ



Một màn hình soạn thảo xuất hiện chính giữa cửa sổ giao diện

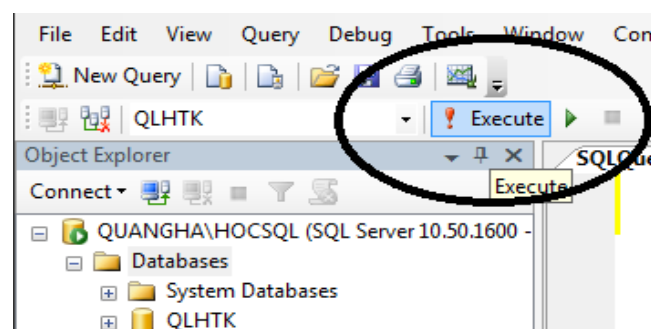


Bước 2: Chọn Database cần giao tác trong hộp chọn Available Database

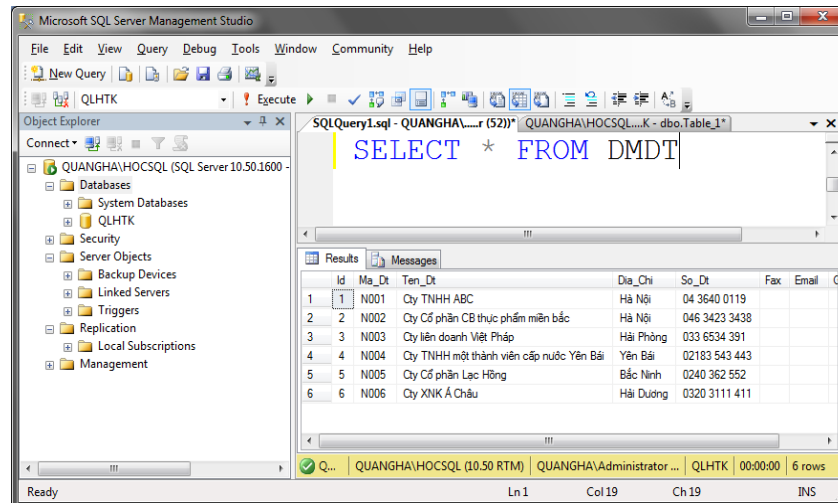


Bước 3: Viết câu lệnh SQL trong màn hình soạn thảo New Query

Bước 4: Nhấn F5 hoặc nhấp chọn biểu tượng Execute để thực thi câu lệnh



Kết quả câu lệnh SQL hiển thị trên màn hình soạn thảo New Query

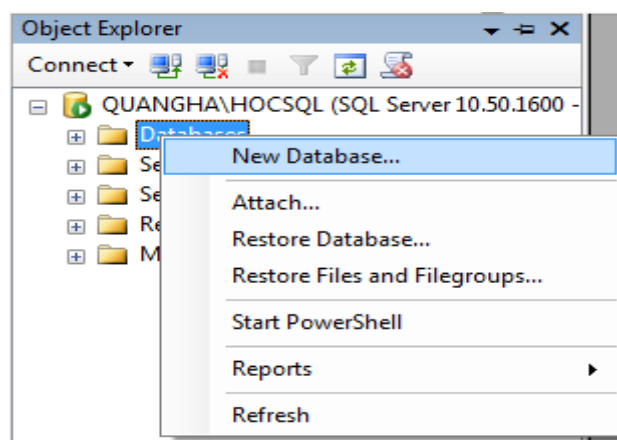


3. QUẢN LÝ DATABASE.

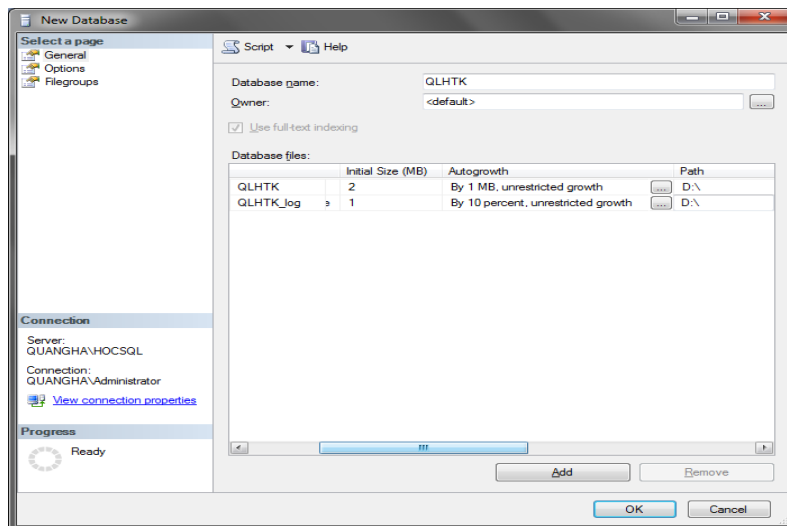
3.1 Tạo mới một Database

Bước 1: Đăng nhập SQL Server 2008

Bước 2: Trong cửa sổ Object Explorer nhấp phải chuột vào mục Database chọn New Database.



Bước 3: Hộp thoại tạo Database mới xuất hiện như hình dưới đây:

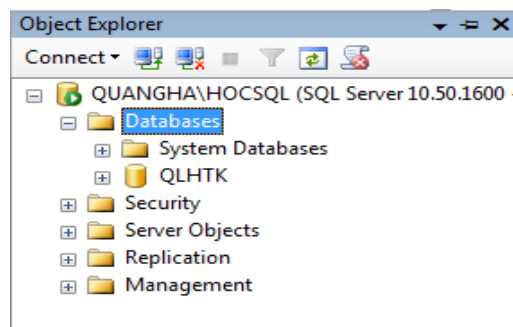


Trong Tab General

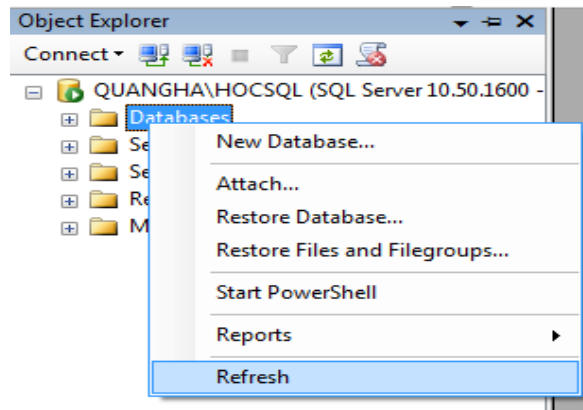
- Database Name: Nhập vào tên Database cần tạo
- Database file: Tại cột Path nhập vào đường dẫn chứa File Database (bao gồm cả đường dẫn cho File Log).

Như ví dụ trên chúng ta đang tạo một Database mới có tên QLHTK và được lưu trữ tại ổ D: của máy tính.

Bước 4: Cuối cùng nhấp Ok để hoàn tất, như vậy trong danh sách Database ở cửa sổ Object Explorer đã xuất hiện một Database mới có tên là QLHTK.



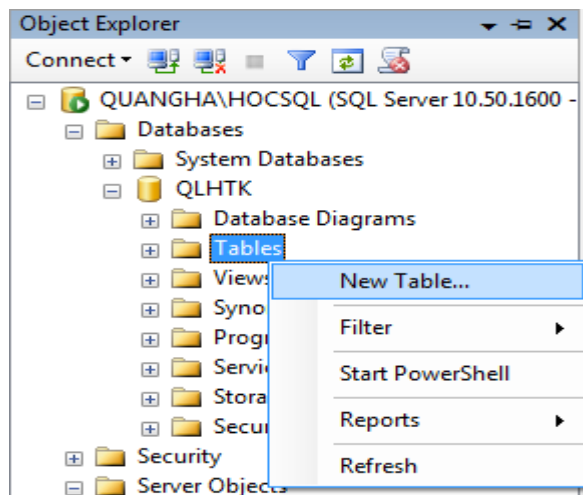
Lưu ý: Sau khi thực hiện xong các bước trên mà vẫn không thấy Database mới xuất hiện trong danh sách, hãy nhấp phải chuột vào mục Database trong cửa sổ Object Explorer chọn Refresh.



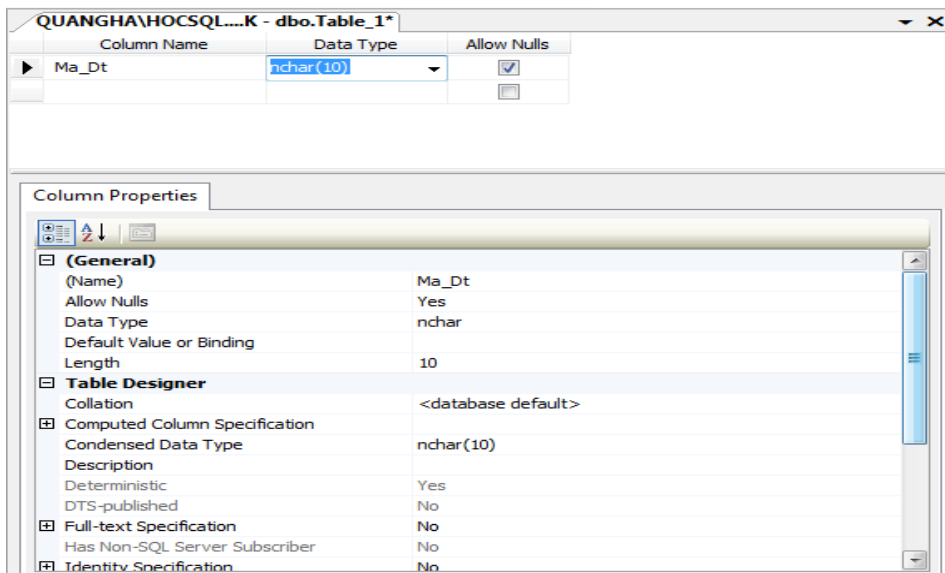
Bước 5: Tạo bảng

Sau khi đã tạo được Database mới để có thể thực hành những ví dụ đơn giản với SQL Server 2008 chúng ta có thể tạo một bảng đơn giản bằng các bước như sau:

Bước 5.1: Trong cửa sổ Object Explorer chọn tới mục Table trong cơ sở dữ liệu vừa tạo, nhấp phải chuột chọn New Table...



Bước 5.2: Hộp thoại tạo bảng xuất hiện như hình dưới đây:



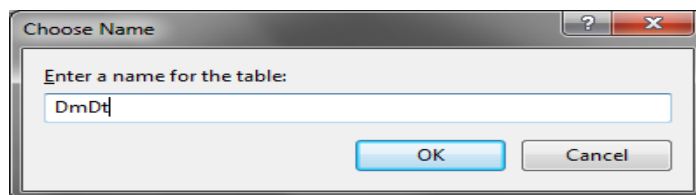
Ở phần phía trên khai báo danh sách cột gồm các thông tin sau:

- Column Name: Tên cột (trường)
- Data Type: Kiểu dữ liệu
- Allow Nulls: Tích chọn nếu cho phép cột nhận giá trị NULL, ngược lại không tích chọn nếu không cho phép trường nhận giá trị NULL.

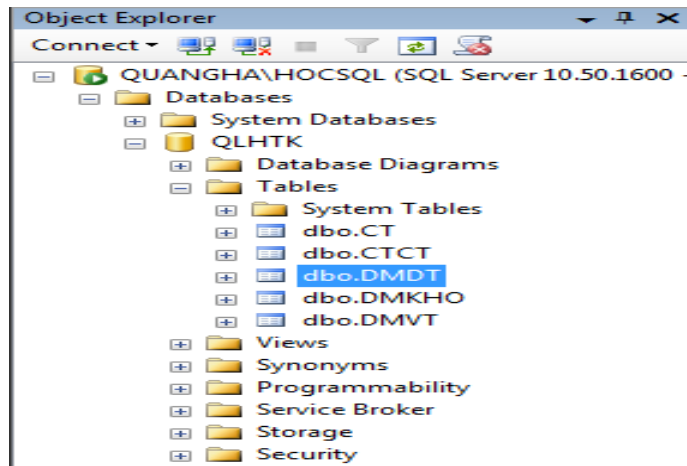
Ở phần phía dưới Column Properties cho phép khai báo các thuộc tính cho từng cột (trường) phần này sẽ được trình bày chi tiết trong các bài tiếp theo.

Nhấp vào dòng dưới cùng trên phần khai báo danh sách cột để thêm một cột mới

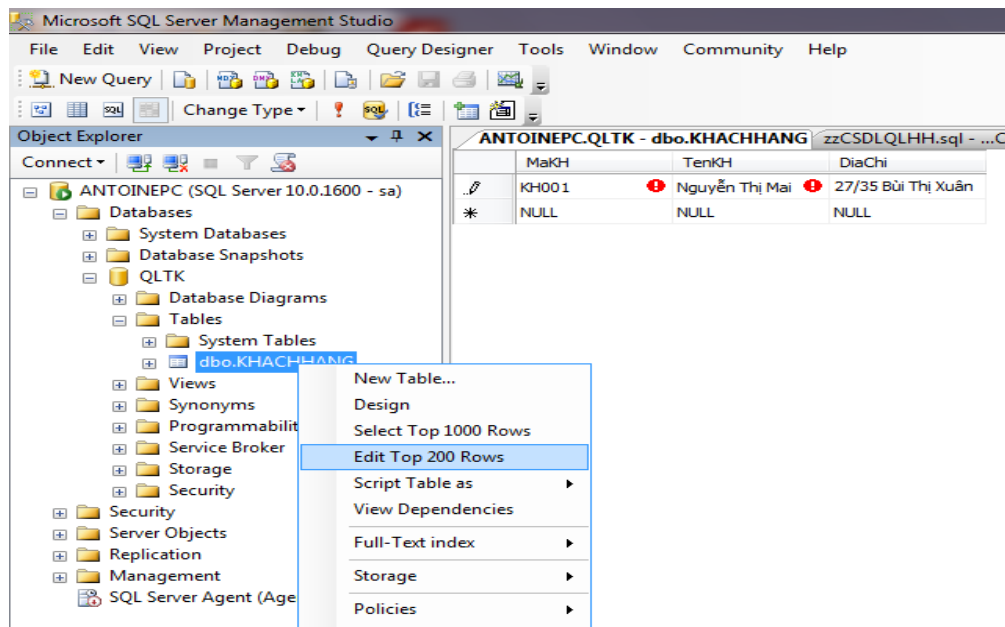
Bước 5.3: Sau khi đã nhập đầy đủ các cột (trường) cho bảng, hãy vào File\Save để lưu lại bảng, hộp thoại Choose Name xuất hiện, nhập tên cho bảng cần thêm mới sau đó nhấp Ok để hoàn tất.



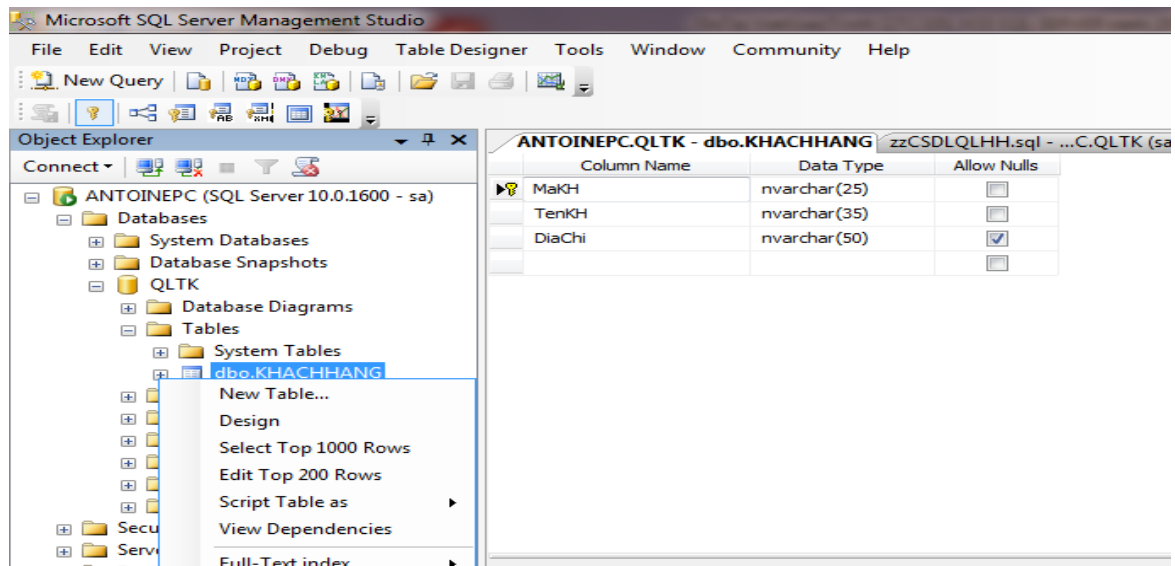
Bước 5.4: Thử tìm xem bảng vừa tạo đang ở đâu? Trong cửa sổ Object Explorer chọn tới Database vừa tạo mở rộng mục Table trong phần này sẽ chứa bảng mà chúng ta vừa tạo.



Bước 5.5: Mở bảng và nhập dữ liệu: Click phải chuột vào bảng vừa tạo\ chọn Edit Top 200 Rows (mặc định) sẽ xuất hiện bảng cho ta nhập liệu vào.



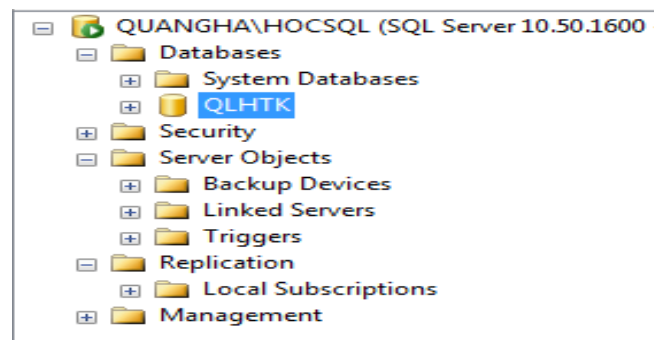
Bước 5.6: Thay đổi thiết kế bảng: Click phải chuột vào bảng vừa tạo\ chọn Design sẽ xuất hiện bảng khai báo cấu trúc bảng cho ta chỉnh sửa.



3.2 Sao lưu Database (Backup)

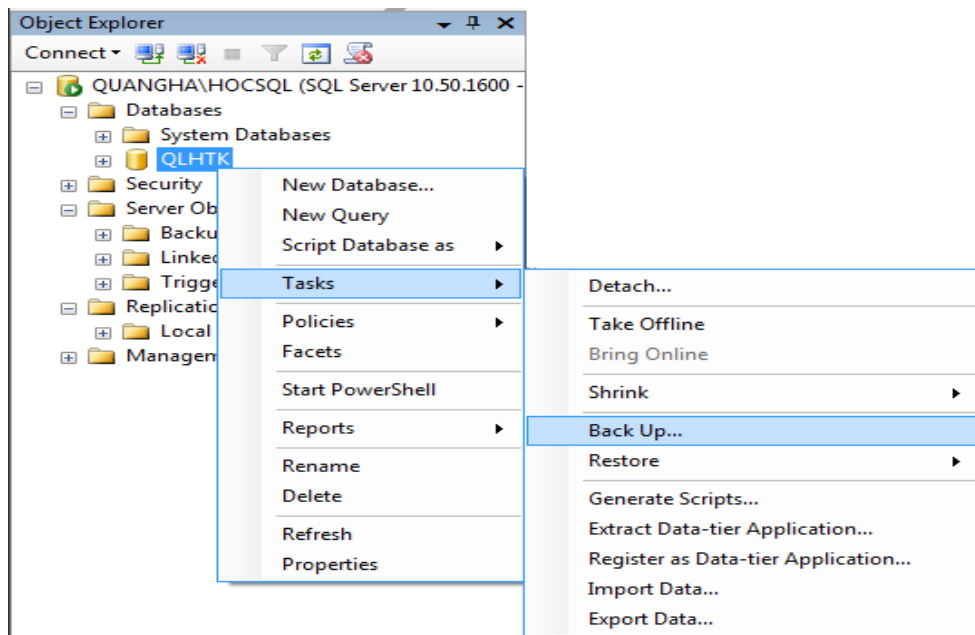
Khi cần cất giữ Database chúng ta sử dụng chức năng Backup để nén dữ liệu và cất giữ ở dạng File vật lý. Các bước thực hiện như sau:

Bước 1: Chọn Database cần cất trữ trong cửa sổ Object Explorer.

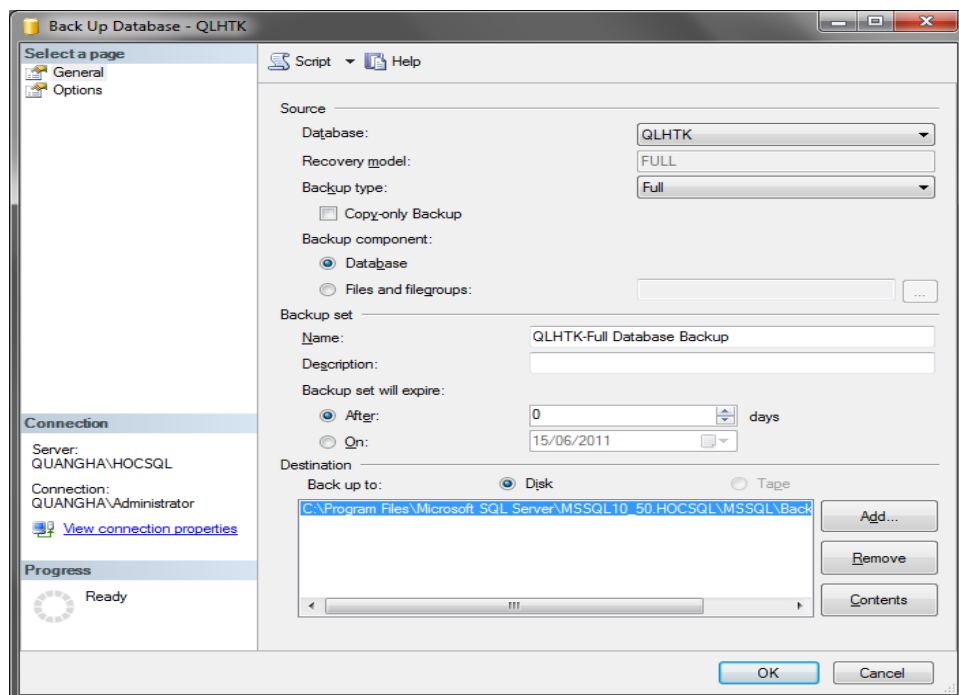


Giả sử như chúng ta chọn Database vừa tạo

Bước 2: Nhấp phải chuột vào Database này chọn Tasks/Back Up...

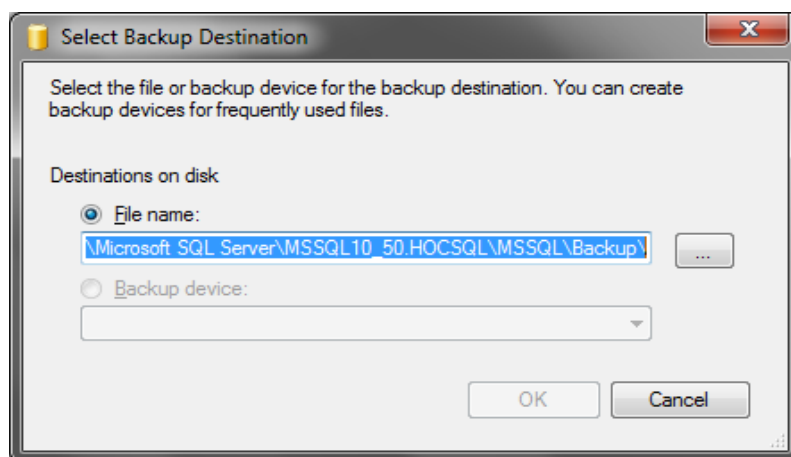


Bước 3: Cửa sổ Back up.. Database xuất hiện như hình dưới đây:

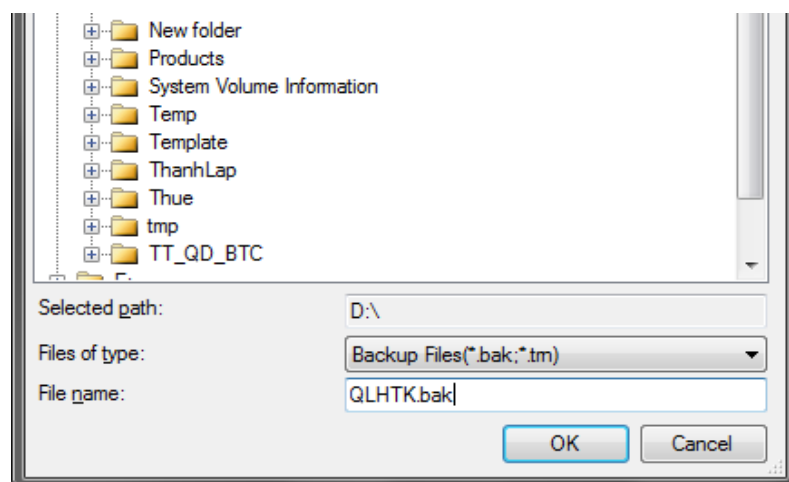


Trong Tab General, tại mục Destination thông thường SQL Server 2008 chọn sẵn thư mục lưu File Backup tại ổ C:\ của máy tính, tuy nhiên chúng ta nên lưu tại một đường dẫn nào đó an toàn hơn bằng cách nhấp chọn Remove, sau đó nhấp nút Add.

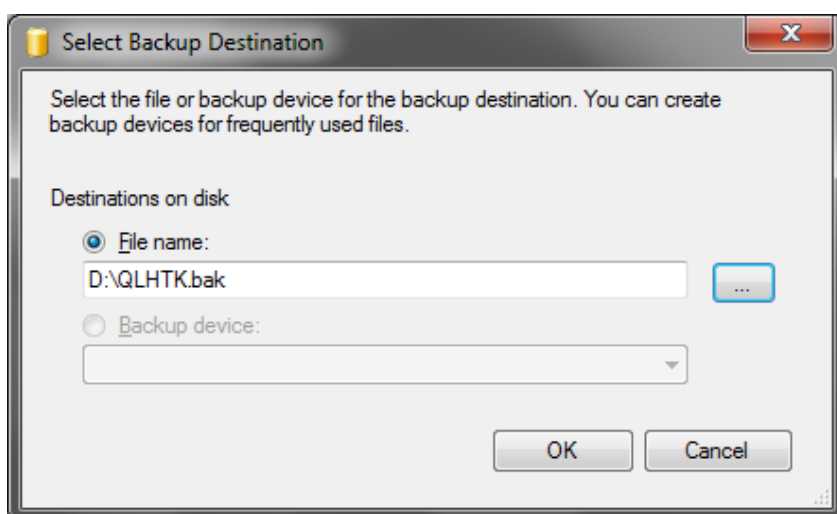
Bước 4: Hộp thoại Select Backup Destination xuất hiện như hình dưới đây:



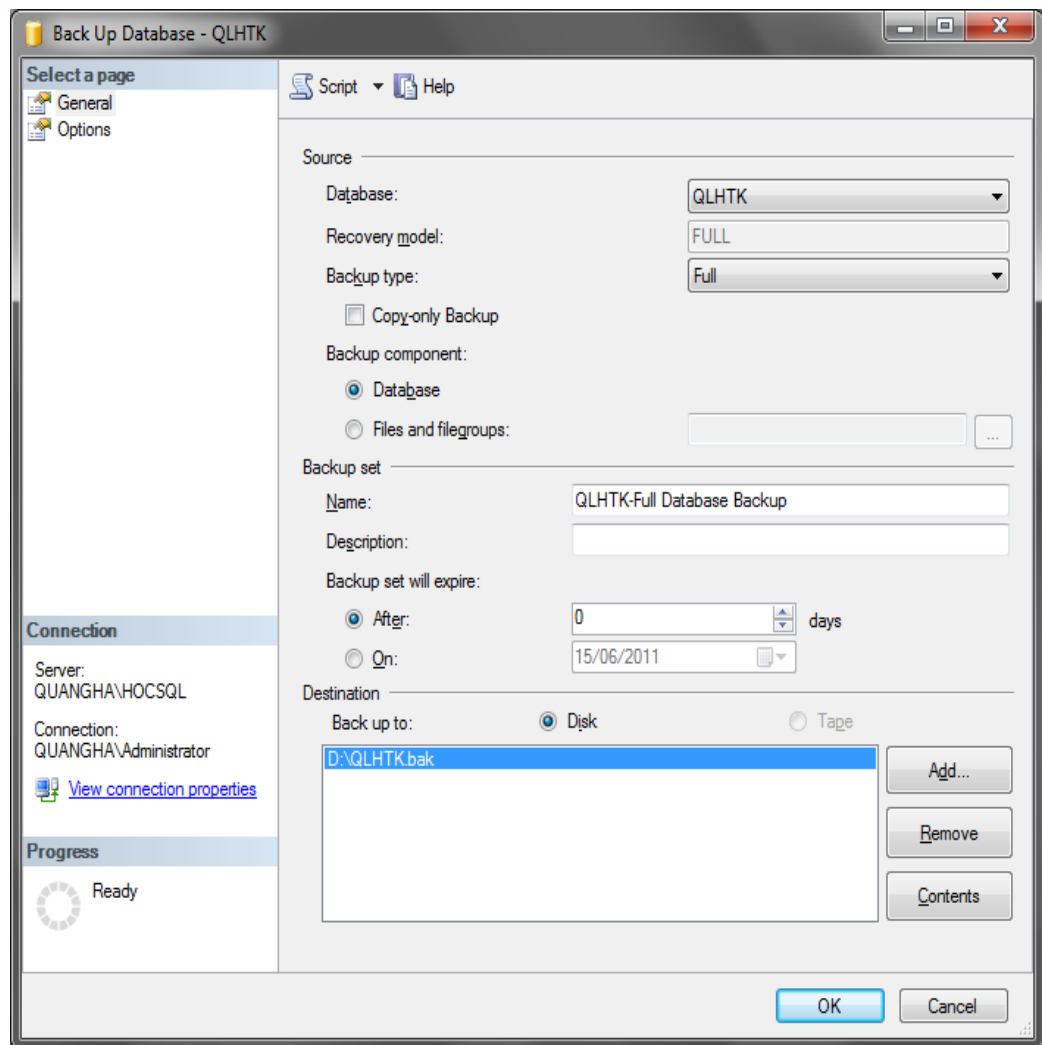
Nhấp chuột vào nút (...) để chọn đường dẫn mới



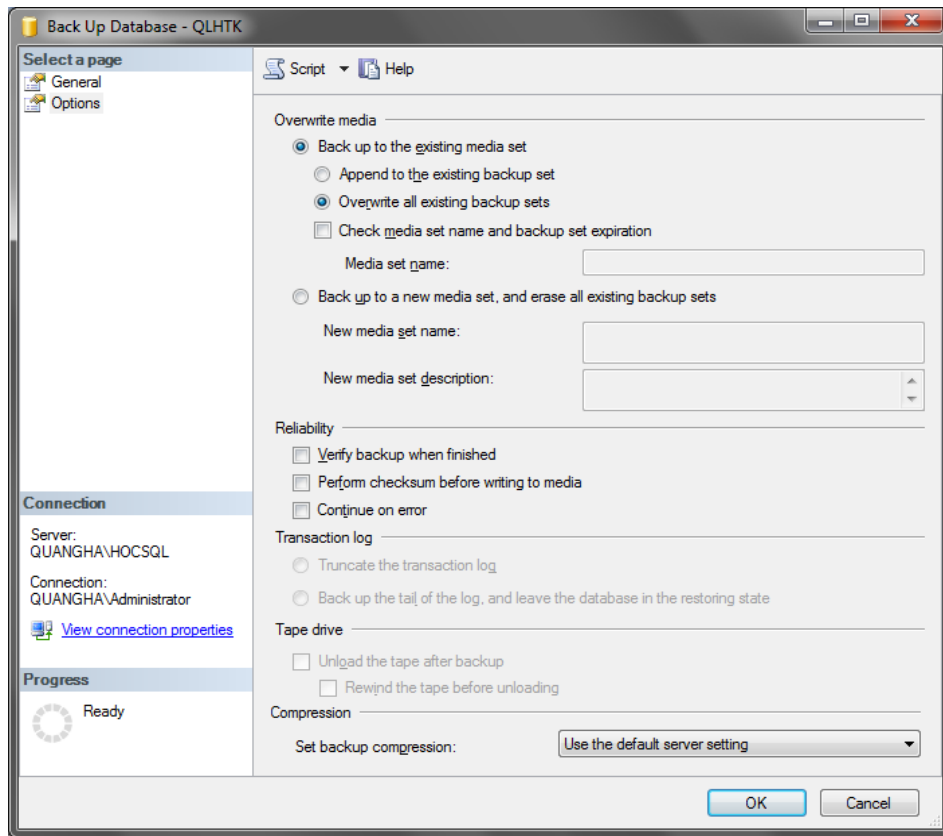
Như hình ảnh trên chúng ta lưu File Backup tại ổ đĩa D:\ của máy tính, bạn đừng quên đặt tên cho File Backup trong mục File name với phần mở rộng là .bak, như ví dụ trên File Backup sẽ được lưu tại ổ D:\ và có tên là QLHTK.bak. Nhấp Ok để chọn đường dẫn và xác nhận tên File Backup, tiếp tục nhấp Ok ở màn hình Select Backup Destination với đường dẫn mới.



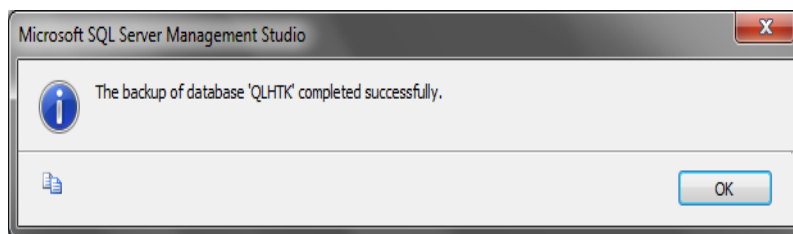
Bước 5: Sau bước trên chúng ta được kết quả màn hình Backup Database như hình dưới đây:



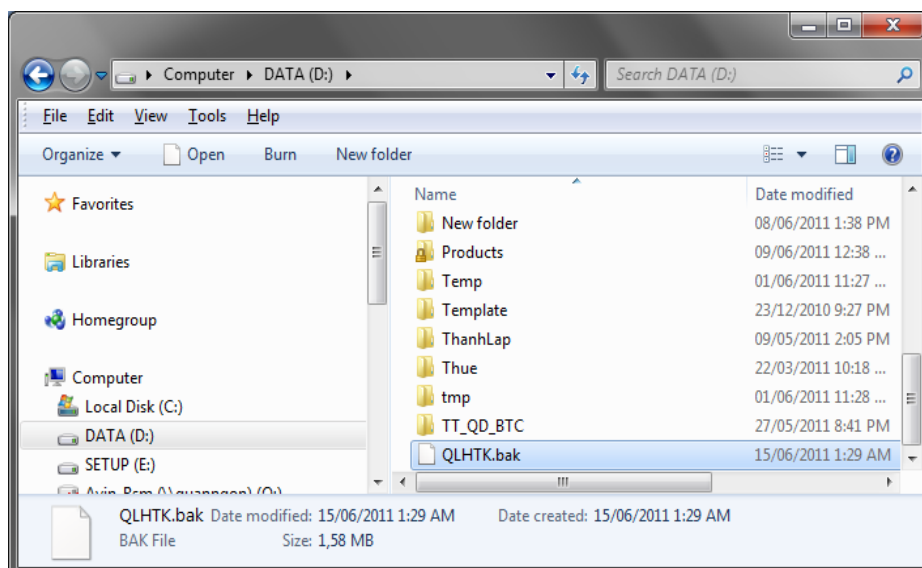
Bạn tiếp tục chọn tới Tab Options trong mục Select a page



Trên Tab này nhấp chọn mục Overwrite all existing backup set để ghi đè lên dữ liệu cũ đã tồn tại nếu có. Cuối cùng nhấp Ok để máy tính thực hiện Backup, chờ trong giây lát chương trình sẽ thông báo hoàn tất thao tác như hình dưới đây:



Bước 6: Bây giờ hãy kiểm tra lại bằng cách truy cập vào ổ D:\ của máy tính



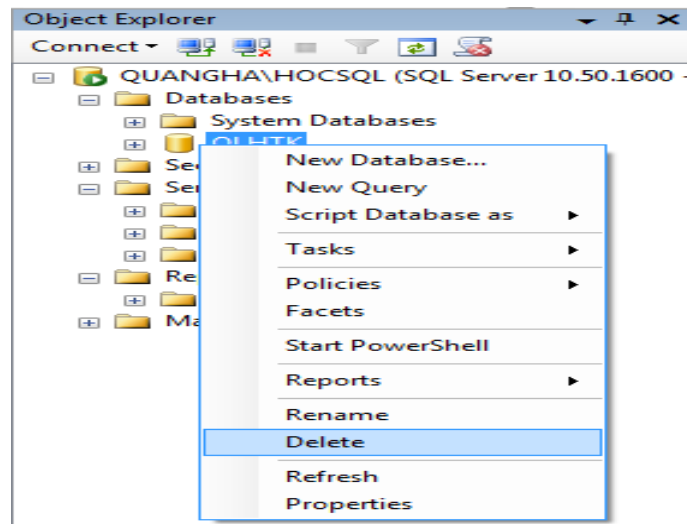
Kết quả chúng ta sẽ thấy xuất hiện File Backup mới như hình ảnh trên.

3.3 Xóa Database (Delete)

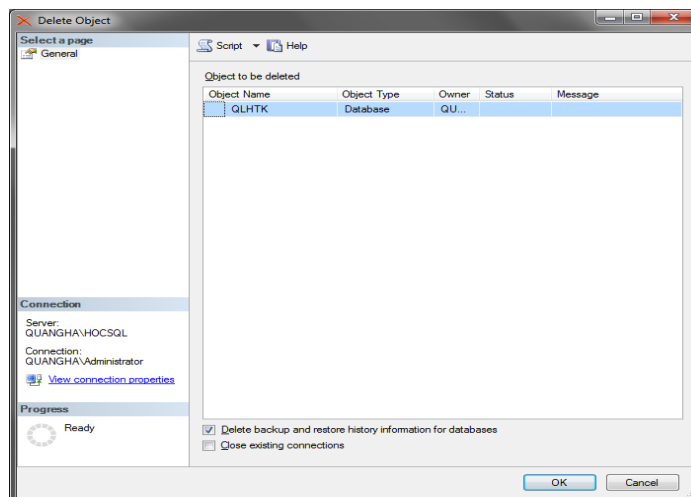
Xóa vĩnh viễn Database khỏi hệ quản trị cơ sở dữ liệu Sql Server 2008 và máy tính, các bước thực hiện như sau:

Bước 1: Chọn Database cần xóa trong Object Explorer.

Bước 2: Nhấp phải chuột vào Database này chọn Delete



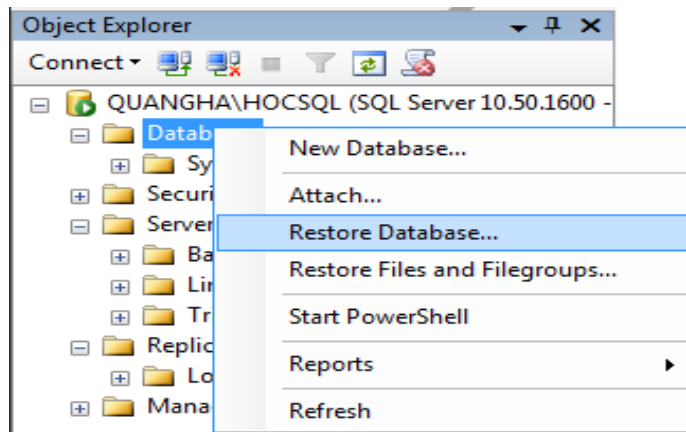
Bước 3: Hộp thoại Delete Object xuất hiện nhấp Ok để thực hiện xóa.



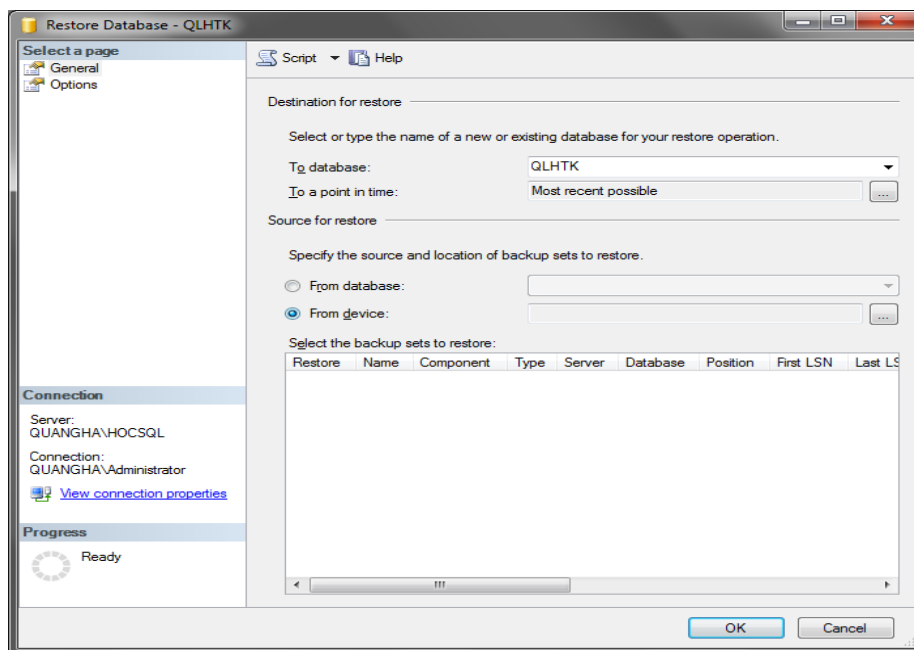
3.4 Phục hồi Database (Restore)

Phục hồi Database Backup, chỉ thực hiện được khi chúng ta có File Backup về Database đó, cách thực hiện như sau:

Bước 1: Nhấp phải chuột vào mục Database chọn Restore Database...

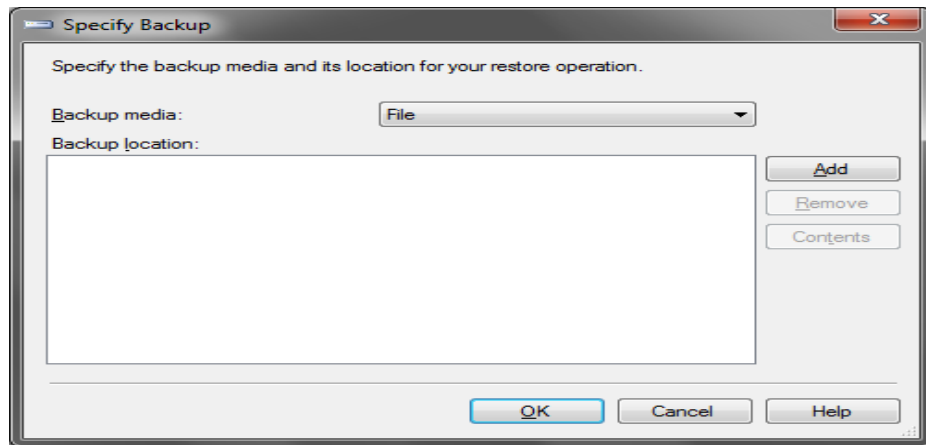


Bước 2: Hộp thoại Restore Database xuất hiện như hình dưới đây:

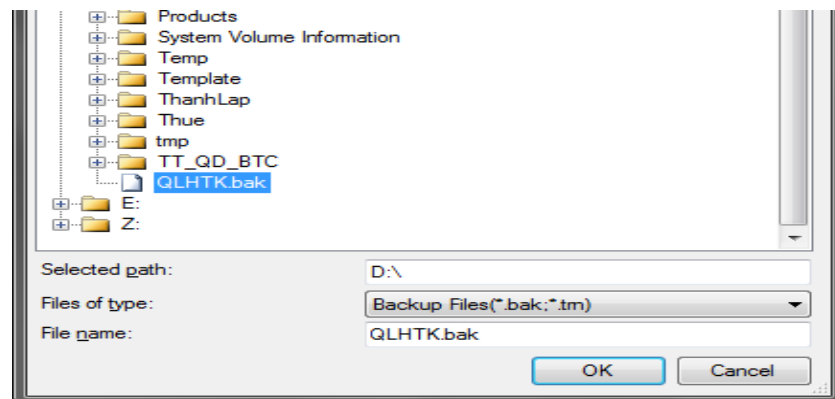


Chúng ta có thể chọn Database để phục hồi dữ liệu từ File Backup hoặc phục hồi dữ liệu Backup thành một File mới, dưới đây chúng ta sẽ thực hiện theo trường hợp thứ 2. Như vậy tại Tab General trong mục To Database hãy nhập trực tiếp vào tên Database mới cần tạo từ File backup ví dụ như hình ảnh trên chúng ta sẽ tạo ra một Database mới có tên QLHTK.

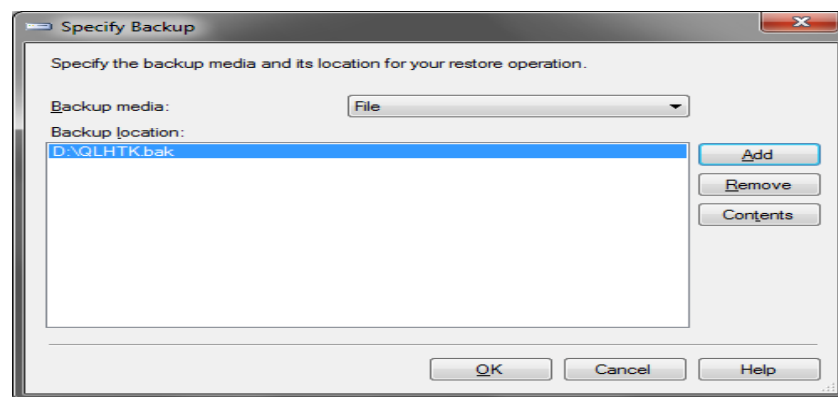
Bước 3: Cũng tại Tab General chọn From device sau đó nhấp chọn nút (...) hộp thoại Specify backup xuất hiện như hình dưới đây:



Nhấp Add để tìm tới File Backup cần Restore.

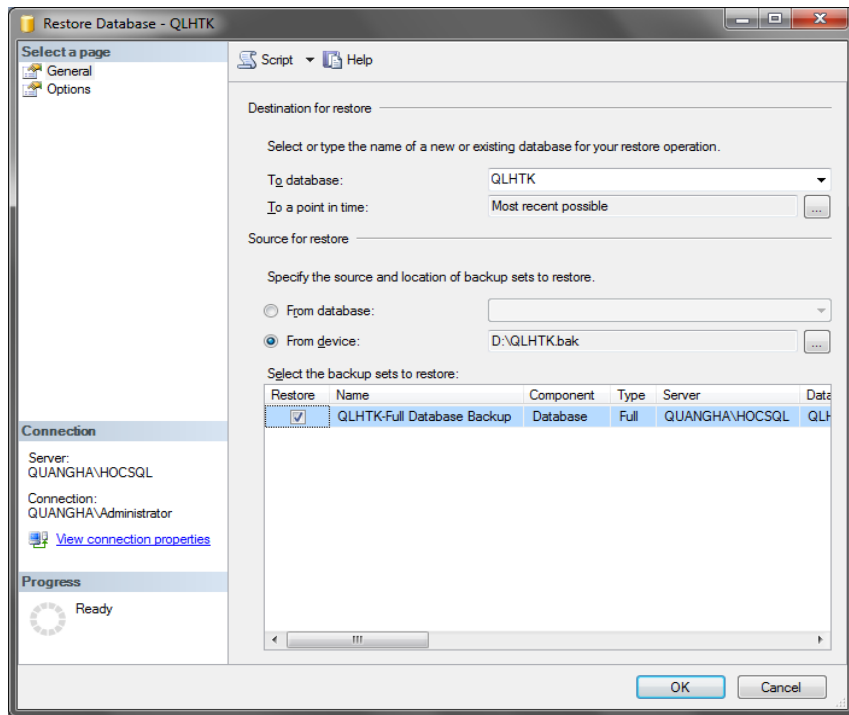


Giả sử chúng ta tìm thấy File cần Restore có tên là QLHTK.bak được lưu tại ổ D:\, nhấp chọn File này và nhấp Ok.

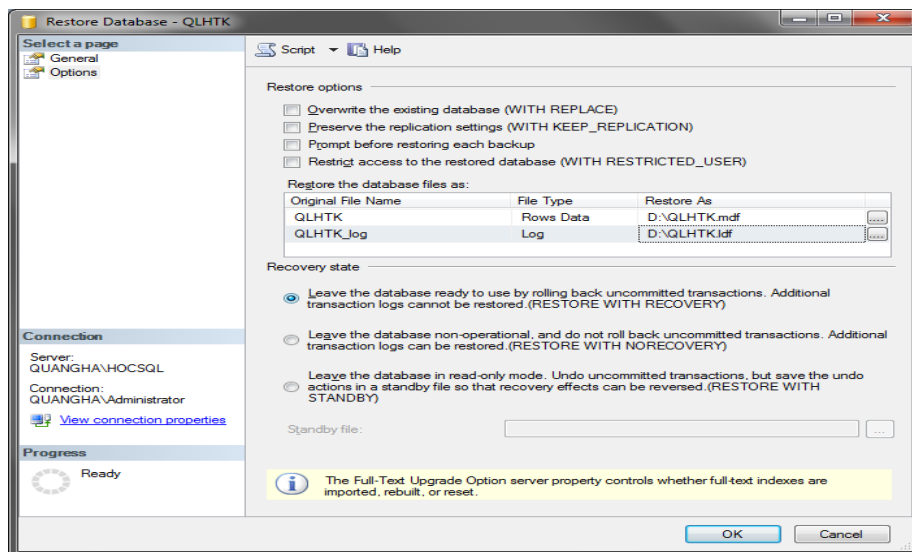


Trở lại với cửa sổ Specify Backup, ta nhấp Ok để xác nhận đường dẫn vừa chọn.

Bước 4: Khi đó màn hình Restore Database sẽ xuất hiện với những thông tin về Database backup, bạn hãy tích chọn vào cột Restore trong mục Select the backup set to restore.



Bước 5: Tích chọn Tab Options trong mục Select a page



Trong mục Restore the Database file as ở cột Restore As hãy chọn tới đường dẫn cần lưu File Database khi Restore, đây là đường dẫn đầy đủ bao gồm cả tên File và phần mở rộng. Thông thường ở dòng thứ nhất tên File có phần mở rộng là .mdf, dòng thứ 2 lưu File Log có phần mở rộng là .ldf.

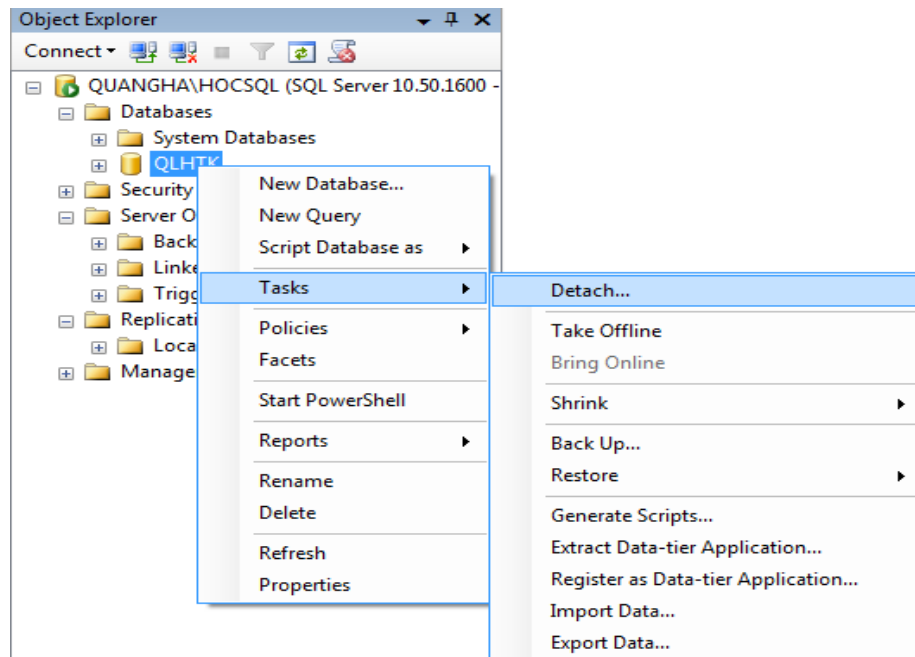
Bước 6: Cuối cùng nhấp Ok để thực hiện Restore. Chúng ta có thể kiểm tra việc thực thi chức năng như phần thêm mới một Database.

3.5 Gỡ bỏ Database (Detach)

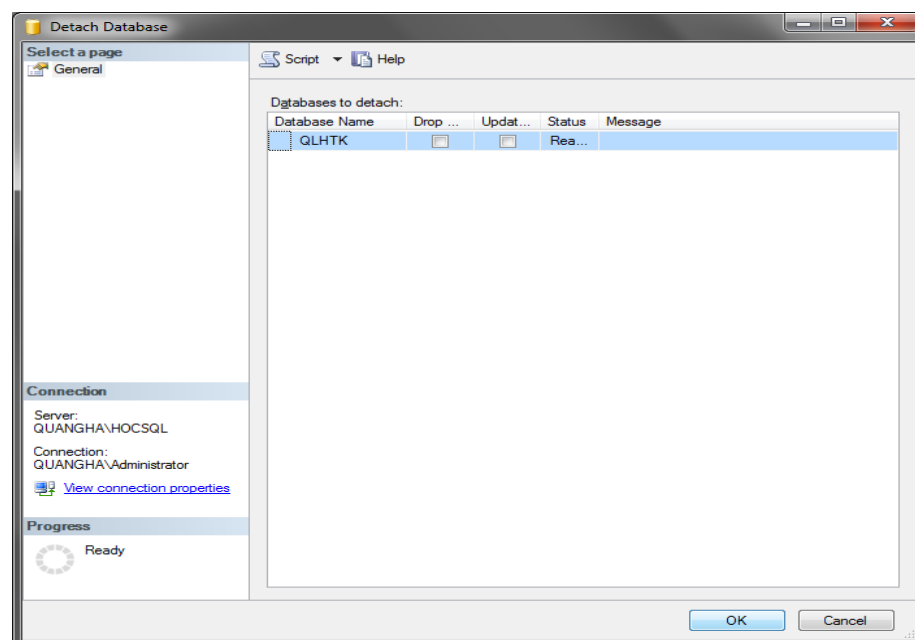
Khác với việc xóa Database, Detach chỉ gỡ bỏ Database khỏi sự quản lý của hệ quản trị cơ sở dữ liệu SQL Server 2008 hiện thời. Các bước thực hiện như sau:

Bước 1: Chọn Database cần Detach trong cửa sổ Object Explorer

Bước 2: Nhấp phải chuột chọn Task/Detach...



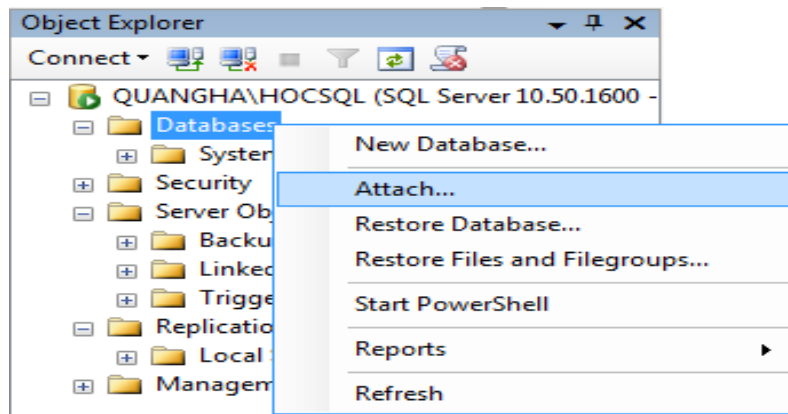
Bước 3: Hộp thoại Detach Database xuất hiện nhấp Ok để thực hiện Detach



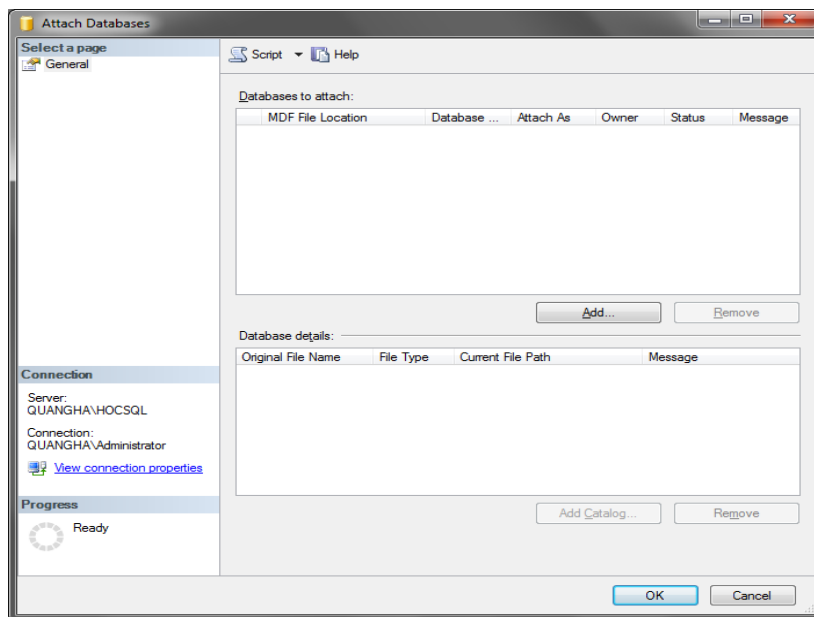
3.6 Đính kèm Database (Attach)

Gắn Database được Detach vào hệ quản trị cơ sở dữ liệu (HQTCSDL) SQL Server 2008, Attach chỉ thực hiện với những Database đã được Detach trước đó. Các bước thực hiện như sau:

Bước 1: Nhấp phải chuột vào mục Database trong cửa sổ Object Explorer chọn Attach...

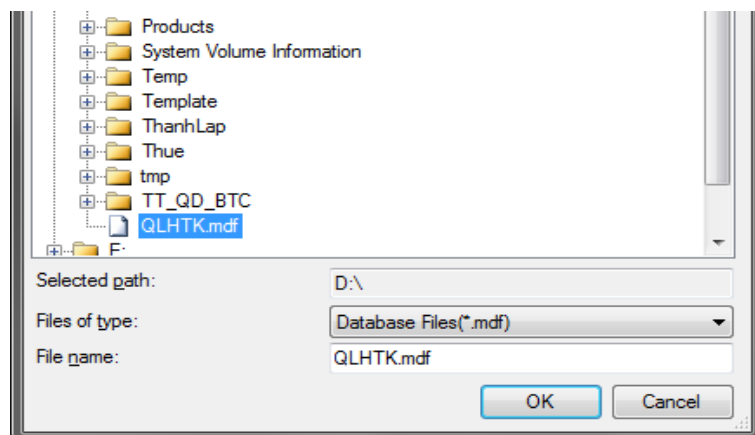


Bước 2: Hộp thoại Attach Database xuất hiện như hình dưới đây:



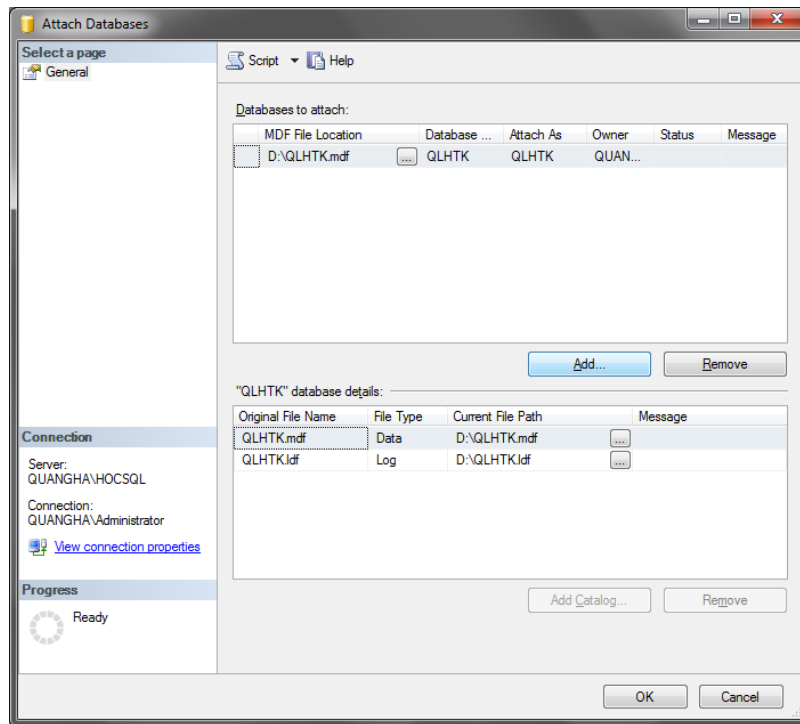
Nhấp Add... để chọn tới Database cần Attach.

Bước 3: Sau bước 2 hộp thoại chọn Database xuất hiện



Như hình ảnh trên chúng ta chọn tới Database QLHTK.mdf lưu tại ổ D: của máy tính và vừa được Detach từ phần trước. Nhấp Ok để xác nhận chọn Database.

Bước 4: Bây giờ trở lại với cửa sổ Attach Database



Nhấp Ok để thực hiện Attach.

4. CÂU HỎI ÔN TẬP

- 1/- Hãy kể tên các bước tạo cơ sở dữ liệu trong MS SQL Server?
- 2/- Hãy kể tên các bước sửa, xóa cơ sở dữ liệu trong MS SQL Server?
- 3/- Hãy kể tên các bước lưu trữ, phục hồi cơ sở dữ liệu trong MS SQL Server?
- 4/- Hãy kể tên các bước gỡ bỏ và đính kèm cơ sở dữ liệu vào MS SQL Server?
- 5/- Hãy thao tác chuột thông qua giao diện của Management Studio để làm bài tập 1 và 2 phần A luyện tập TẠO CƠ SỞ DỮ LIỆU.

BÀI 3: ĐỊNH NGHĨA DỮ LIỆU VỚI T-SQL

Mã bài: MD15_03

Giới thiệu:

Bài học giúp sinh viên tạo và hiệu chỉnh được các đối tượng trong cơ sở dữ liệu, viết được các câu lệnh để định nghĩa và cập nhật cấu trúc cơ sở dữ liệu.

Mục tiêu:

- Trình bày được cách tạo mới, sửa, xóa các đối tượng trong cơ sở dữ liệu;
- Nêu được các kiểu dữ liệu trong SQL Server đặc biệt là các kiểu dữ liệu mới thêm vào SQL Server;
- Phân loại được các đối tượng trong cơ sở dữ liệu;
- Viết được các câu lệnh để định nghĩa và cập nhật cấu trúc cơ sở dữ liệu;
- Hoàn thành đầy đủ bài tập một cách tích cực, chủ động khoa học và sáng tạo.

SQL chuẩn quy định cách đặt tên đối tượng trong CSDL như sau:

+ Tên đối tượng dài tối đa 32 ký tự và bắt đầu bằng chữ cái, tên không có khoảng trắng hoặc dấu tiếng việt.

+ Nếu là tên bảng hoặc khung nhìn (view) thì phải là duy nhất trong CSDL và không được trùng với các từ khóa trong ngôn ngữ của hệ quản trị cơ sở dữ liệu.

+ Nếu là tên cột thì trong cùng một bảng không được đặt tên trùng nhau. Tuy nhiên các cột có thể trùng tên nếu nó nằm trong các bảng khác nhau. Nên đặt tên cho các cột mang cùng ý nghĩa ở các bảng.

+ Nếu đặt tên cho các ràng buộc trong CSDL thì phải là duy nhất và không được đặt tên trùng nhau.

Để thuận tiện cho ngành lập trình, các cú pháp lệnh ta đều dựa trên Hệ quản trị cơ sở dữ liệu SQL SERVER 2008.

1. NGÔN NGỮ ĐỊNH NGHĨA DỮ LIỆU

1.1 TẠO VÀ SỬA ĐỔI CẤU TRÚC CƠ SỞ DỮ LIỆU.

1.1.1 *Tạo tập tin cơ sở dữ liệu.*

1.1.1.1 *Tạo cơ sở dữ liệu với đường dẫn mặc định*

Tạo cơ sở dữ liệu mới với đường dẫn file data (.mdf) và file log (.ldf) sẽ được lưu vào đường dẫn mặc định của SQL Server.

CREATE DATABASE <Tên cơ sở dữ liệu>

1.1.1.2 *Tạo cơ sở dữ liệu với đường dẫn chỉ định*

Tạo cơ sở dữ liệu mới và chỉ định đường dẫn để lưu cơ sở dữ liệu.

CREATE DATABASE <Tên cơ sở dữ liệu>

ON

PRIMARY (NAME = TenFileCSDL, FILENAME = 'D:\LAPTRINH\TenFileCSDL.mdf', SIZE = 10, MAXSIZE=100|UNLIMITED, FILEGROWTH = 5)

LOG ON (NAME = TenFileCSDL_LOG, FILENAME = 'D:\LAPTRINH\TenFileCSDL_LOG.ldf', SIZE = 10, MAXSIZE = 50, FILEGROWTH = 5).

Trong đó:

- *SIZE (MB): là dung lượng khởi đầu của tập tin.*
- *MAXSIZE (MB): là dung lượng tối đa của tập tin.*
- *FILEGROWTH (MB): là dung lượng được tự động tăng.*
- *UNLIMITED : dung lượng không hạn chế.*

1.1.2 *Khai báo cơ sở dữ liệu đã có vào MS SQL Server*

Đưa cơ sở dữ liệu từ file .mdf vào hệ quản trị cơ sở dữ liệu

CREATE DATABASE <Tên cơ sở dữ liệu>

ON (FILENAME = 'đường dẫn và tên file csdl .mdf')

FOR ATTACH

GO

1.1.3 Đổi tên cơ sở dữ liệu

ALTER DATABASE <Tên cơ sở dữ liệu cũ>

MODIFY NAME = <Tên cơ sở dữ liệu mới>

1.1.4 Tăng dung lượng tập tin dữ liệu

ALTER DATABASE <Tên cơ sở dữ liệu>

MODIFY FILE

(

NAME = <tên file csdl_data>, SIZE = <kích thước mới>

) GO

1.1.5 Dời vị trí tập tin .mdf:

ALTER DATABASE <tên cơ sở dữ liệu>

MODIFY FILE

(

NAME = <Tên cơ sở dữ liệu>, FILENAME = N'Đường dẫn\tên file csdl.mdf'

)GO

1.1.6 Cấu hình CSDL chỉ đọc thì dùng SET

ALTER DATABASE <Tên cơ sở dữ liệu>

SET READ_ONLY

GO

1.1.7 Chỉ định cơ sở dữ liệu được dùng cho các lệnh truy vấn.

USE <Tên cơ sở dữ liệu chỉ định dùng>

-- Lưu ý: không có từ Database

1.1.8 Xóa cơ sở dữ liệu

Để xóa cơ sở dữ liệu thì cơ sở dữ liệu không được đang truy cập.

DROP DATABASE <Tên cơ sở dữ liệu muốn xóa>

1.1.9 Lưu ý về Quyền tạo CSDL :

- + Tài khoản đăng nhập phải là : *sysadmin, dbcreator*
- + Hoặc có một trong các quyền : *Create Database/ Any Adatabase/ Alter Any Database*
- + Kiểm tra quyền : *Security/ Logins/ Tên tài khoản/ R_Click/ Properties/ Server Roles*
- + Tạo tài khoản : *Security/ Logins/ R_Click/ New Login*. Sau đó vào ngăn *Server Roles* để chọn quyền tạo csdl là *dbCreator*. Nếu check vào *SysAdmin* thì tài khoản có quyền như *Administrator*.

1.2 TẠO BẢNG TRONG CƠ SỞ DỮ LIỆU.

Lưu ý: Khi tạo hai bảng có quan hệ với nhau ta phải tạo bảng bên một trước, bên nhiều sau.

1.2.1 Các ràng buộc toàn vẹn trong tạo bảng:

- *NOT NULL* : không cho phép chứa giá trị null
- Các tùy chọn khác : *SET NULL, CASCADE* và *SET DEFAULT ...*
- *Khóa chính* :
 - **PRIMARY KEY** (<các thuộc tính tham gia khóa chính>)
 - Giá trị khóa chính phải luôn luôn khác null.
 - Bộ giá trị cho tập thuộc tính khóa chính phải là duy nhất.
- *Tính duy nhất* :
 - **UNIQUE** (<các thuộc tính hoặc khóa đề nghị>)
 - Ràng buộc *UNIQUE* : có thể được phép chứa giá trị null (khóa đề nghị)
 - Giá trị cho thuộc tính khóa đề nghị phải là duy nhất.

- *Ràng buộc khóa ngoại hay toàn vẹn tham chiếu :*
 - **FOREIGN KEY** <thuộc tính khóa ngoại> **REFERENCES** <tên bảng tham chiếu> (<thuộc tính khóa chính tương ứng>)
 - ▲ *Khóa ngoại có thể chứa giá trị null.*
 - ▲ *Có thể được thêm vào ràng buộc khóa ngoại để tạo ra tự động cập nhật bằng : ON DELETE, ON UPDATE thông qua [On Delete Cascade On Update Cascade]*
- **CHECK** (<điều kiện>)
 - *Điều kiện phải là một điều kiện đơn giản.*
- *Các giá trị mặc định :*
 - *Được chỉ ra bằng : **DEFAULT** <giá trị mặc định>*
 - *Được dùng khi không có giá trị tường minh nào gán cho thuộc tính.*
 - *Sẽ mang giá trị null nếu không có chỉ ra giá trị khác.*
- *Trong SQL mỗi ràng buộc có thể được đặt tên bằng cách sử dụng:*
 - **CONSTRAINT** <tên đặt cho ràng buộc> <kiểu ràng buộc>
 - *Tên ràng buộc phải duy nhất trong một lược đồ csdl.*

1.2.2 Một số kiểu dữ liệu thường dùng.

Kiểu dữ liệu dùng để định nghĩa cho các cột trong các bảng khi tạo bảng hoặc định dạng dữ liệu nhập vào.

Kiểu dữ liệu	Mô tả	Cú pháp
Kiểu dữ liệu Binary: kiểu dữ liệu này bao gồm các số hexa có 3 dạng như sau:	Binary	binary[(n)]
	Varbinary	Varbinary[(n)]
	Image	Image
Kiểu dữ liệu Text: Đây là kiểu dữ liệu kí tự, kiểu này bao gồm một tổ hợp các chữ cái, các ký hiệu hoặc	Char	Char[(n)]
	Varchar	Varchar[(n)]

các chữ số Trong đó kiểu dữ liệu Nchar & nvarchar lưu các kí tự theo chuẩn Unicode	Nvarchar	Nvarchar[0]
	Text	Text
	Ntext	ntext
Date và time: Đây là kiểu dữ liệu ngày tháng, kiểu dữ liệu này được đặt trong dấu ‘ ‘ (dấu nháy đơn)	Datetime	Datetime
	Smalldatetime	Smalldatetime
Monetary : Kiểu dữ liệu này lưu giữ các giá trị tiền tệ với độ chính xác là 4 chữ sau dấu phẩy (,)	Money	Money
	Smallmoney	Smallmoney
Kiểu Numeric: Kiểu dữ liệu này lưu giữ các giá trị số(số dương, số âm, số thập phân)	Interger: gồm các số nguyên dương, số nguyên âm như: -23,0,10,34.... <i>Kiểu dữ liệu này có 4 dạng :</i> int , smallint, tinyint, bigint	Int Smallint Tinyint Bitint
	Decimal gồm: float,real.	Decimal(18,0) Float real
Sql_variant: Đây là kiểu dữ liệu lưu giá trị của các kiểu dữ liệu khác trong SQL server.	Sql_variant	Sql_variant
Bit: Kiểu dữ liệu này chỉ lưu các giá trị 1 hoặc 0	Bit	bit

1.2.3 Lệnh tạo bảng

1.2.3.1 Cú pháp tạo bảng

CREATE TABLE <tên bảng>

(
 <tên_thuộc_tính_1> <kiểu dữ liệu> [<Khai báo ràng buộc thuộc
 tính>],
 ...
 [Constraint <tên_rb_pk> PRIMARY KEY (thuộc tính tham gia khóa
 chính),]
 [Constraint <tên_rb_fk> FOREIGN KEY (<thuộc tính khóa ngoại>)
 REFERENCES <bảng>(khóa chính tương ứng) [On Delete Cascade On
 Update Cascade],]
 [Constraint <tên_rb_mgt> CHECK (<điều kiện đơn>)]
 --Lưu ý: dòng lệnh cuối cùng không có dấu phẩy.
);

1.2.3.2 *Hiển thị tên tất cả các bảng trong cơ sở dữ liệu*

Xem tên tất cả các bảng trong CSDL

Exec sp_Tables

Go

1.2.3.3 *Hiển thị cấu trúc của bảng*

Xem cấu trúc của bảng

Exec sp_Columns <Tên bảng>

Go

Trong đó:

- + *Exec* : viết tắt của từ *Execute* (thực thi)
- + *sp_tables, sp_columns* : là thủ tục hệ thống do SQL Server hỗ trợ.
- + *Go* : để phân cách các lệnh với nhau. Khi lệnh đang thực thi sai thì lệnh
 go cho thi hành lệnh kế tiếp sau.

1.2.3.4 *Hiển thị thông tin của cột dữ liệu trong bảng*

Xem thông tin của cột dữ liệu trong bảng

*Select * From syscolumns*

Where id=(Select id From sysobjects Where name='Tên bảng')

Hoặc SQL Server 2005:

*Select * From sys.columns*

Where object_id = (Select object_id from sys.tables where name = 'Tên bảng')

Hoặc :

*Select * from sysobjects where xtype/type='U'*

1.2.3.5 Hiển thị tên các khoá chính -

Xem tên các khóa chính của bảng

Exec sp_pkeys <tên bảng>

1.2.3.6 Hiển thị tên các khoá ngoại

Xem tên các khóa ngoại của bảng

Exec sp_fkeys <tên bảng>

1.2.3.7 Hiển thị tên các ràng buộc

Xem các ràng buộc trên bảng

Exec sp_helpConstraint <tên bảng>

1.2.4 Lệnh xóa bảng

Lưu ý: khi xóa bảng ta phải xóa bảng bên nhiều trước, bên một sau.

DROP TABLE <Tên bảng cần xóa>

1.2.5 Lệnh Thêm, Xóa, Sửa cột

○ Cú pháp thêm cột :

ALTER TABLE <tên bảng> ADD <Tên cột> <Kiểu dữ liệu>

Ví dụ: Thêm cột DienThoai có kiểu dữ liệu nvarchar 20 ký tự vào bảng SINHVIEN:

alter table SINHVIEN add DienThoai nvarchar(20)

○ Cú pháp xóa cột :

ALTER TABLE <tên bảng> **DROP** COLUMN <Tên cột>

Ví dụ: Xóa cột của một bảng:

alter table SINHVIEN drop column DienThoai

- Cú pháp sửa kiểu dữ liệu cột :

ALTER TABLE <tên bảng> **ALTER** COLUMN <Tên cột> <Kiểu dữ liệu mới>

Ví dụ: sửa kiểu dữ liệu của cột DienThoai:

alter table SINHVIEN alter column DienThoai double

1.2.6 Lệnh Thêm, Xóa, Sửa, Vô hiệu hóa ràng buộc.

- Cú pháp thêm ràng buộc :

ALTER TABLE <tên bảng> **ADD** CONSTRAINT <tên ràng buộc> <khai báo ràng buộc>

Ví dụ: Thêm ràng buộc khóa chính:

alter table SINHVIEN add constraint pk_sv primary key (MaSV)

Trường hợp thêm ràng buộc miền giá trị:

ALTER TABLE <Tên bảng> ADD CONSTRAINT <Tên ràng buộc> **CHECK**
(điều kiện đơn)

Ví dụ: alter table CHITIETXUAT add constraint c_soluong **check** (SoLuong > 0)

- Cú pháp xóa ràng buộc :

ALTER TABLE <tên bảng> **DROP** CONSTRAINT <tên ràng buộc>

Ví dụ: Xóa ràng buộc khóa chính:

alter table SINHVIEN drop constraint pk_sv

- Vô hiệu hóa ràng buộc trong bảng

Alter Table <tên bảng> **NoCheck** Constraint <tên ràng buộc>

- Vô hiệu hóa tất cả các ràng buộc trong bảng

Alter Table <tên bảng> **NoCheck** Constraint **All**

- Kích hoạt ràng buộc

Alter Table <tên bảng> **Check** Constraint <tên ràng buộc>

- Kích hoạt tất cả các ràng buộc

Alter Table <tên bảng> **Check** Constraint **All**

2. CÂU HỎI ÔN TẬP

- 1/- Hãy nêu cú pháp tạo, đổi tên, xóa tập tin cơ sở dữ liệu bằng phát biểu T-SQL?
- 2/- Hãy nêu các loại ràng buộc toàn vẹn trong tạo bảng?
- 3/- Hãy nêu một số kiểu dữ liệu thường dùng?
- 4/- Hãy nêu cú pháp tạo, sửa, xóa bảng bằng phát biểu T-SQL?
- 5/- Hãy nêu cú pháp thêm, sửa, xóa cột trong bảng bằng phát biểu T-SQL?
- 6/- Hãy nêu cú pháp thêm, sửa, xóa ràng buộc trong bảng bằng phát biểu T-SQL?
- 7/- Hãy dùng cửa sổ soạn thảo (Query Editor) để làm bài tập 1 và 2 phần A..

BÀI 4: THAO TÁC DỮ LIỆU VỚI T-SQL

Mã bài: MD15_04

Giới thiệu:

Bài học giúp sinh viên xây dựng được các câu truy vấn lồng nhau và truy vấn trên nhiều bảng với các điều kiện từ đơn giản đến phức tạp.

Mục tiêu:

- Trình bày được cú pháp các câu truy vấn;
- Sử dụng kết hợp mềm dẻo các tiện ích khác nhau khi tạo câu truy vấn;
- Xây dựng chính xác các điều kiện truy vấn;
- Xây dựng được các câu truy vấn lồng nhau và truy vấn trên nhiều bảng;
- Hoàn thành đầy đủ bài tập một cách tích cực, chủ động khoa học và sáng tạo.

SQL được xem như là công cụ hữu hiệu để thực hiện các yêu cầu truy vấn và thao tác trên dữ liệu. Trong chương này, ta sẽ bàn luận đến nhóm các câu lệnh trong SQL được sử dụng cho mục đích này. Nhóm các câu lệnh này được gọi chung là ngôn ngữ thao tác dữ liệu (DML: Data Manipulation Language) bao gồm: thao tác cập nhật, thao tác truy vấn, ...

1. NGÔN NGỮ THAO TÁC CẬP NHẬT DỮ LIỆU (action query, data modification query).

1.1 Thêm dữ liệu

Thêm trực tiếp một bộ giá trị:

Cú pháp :

insert into <tên bảng> [(<tên cột 1>, <tên cột 2>, ... <tên cột n>)]
values (<biểu thức 1>, <biểu thức 2>, ... <biểu thức n>)

Lưu ý :

+ Nếu cột được khai báo kiểu dữ liệu chuỗi hoặc kiểu ngày, thì khi nhập dữ liệu, giá trị của cột phải được đặt trong cặp dấu nháy đơn ('... '), nếu cột được khai báo kiểu dữ liệu số thì giá trị để bình thường.

+ Nếu ta chèn đầy đủ giá trị cho các cột thuộc tính trong bảng thì không cần phải khai báo các cột. Các giá trị chèn vào phải tương ứng với miền giá trị và thứ tự của các cột khi tạo bảng.

+ Nếu ta chèn không đầy đủ giá trị cho các cột thuộc tính trong bảng thì phải khai báo các cột thuộc tính mà có giá trị sẽ chèn vào và thứ tự các giá trị chèn vào phải tương ứng với thứ tự các cột được khai báo.

+ Nếu chèn dữ liệu chuỗi có dấu tiếng Việt thì thêm ký tự N trước chuỗi : N' ... '

+ Nếu dữ liệu chèn vào hoặc điều kiện có kiểu ngày tháng ta nên định dạng ngày tháng trước khi thực hiện câu lệnh bằng dòng lệnh sau: **SET DATEFORMAT DMY**

Ví dụ 1: insert into KHOA(MaKhoa, TenKhoa, SoCBGD)
 values ('CNTT', N'Công nghệ thông tin', 11)

Tương đương : (bởi vì khoa có tất cả 3 cột)

Ví dụ 2: insert into KHOA
 values ('CNTT', N'Công nghệ thông tin', 11)

Trường hợp chèn không đầy đủ bộ giá trị thì phải khai báo danh sách thuộc tính tương ứng.

Vd3 : insert into KHOA (MaKhoa, SoCBGD)
 values ('CNTT', 11)

1.2 Xóa dữ liệu

Cú pháp xóa dữ liệu.

delete from <tên bảng>

[where <điềuKiện>]

Lưu ý : Nếu không có mệnh đề Where thì tất cả các bộ dữ liệu trong bảng sẽ bị xóa.

Vd : delete from KETQUA

where MaSV = '99022'

Vd : xóa toàn bộ dữ liệu trong bảng KETQUA

delete from KETQUA

1.3 Sửa dữ liệu

Cú pháp cập nhật sửa dữ liệu.

update <tên bảng>

set <field1> = <biểuThức1> ,

<field2> = <biểuThức2> ,

...

<fieldn> = <biểu thức n>

[where <điềuKiện>]

Vd: Cập nhật số cá bộ giảng dạy bằng 15 cho khoa có mã là “CNTT”

update KHOA

set SoCBGD = 15,

where MaKhoa = 'CNTT'

2. NGÔN NGỮ TRUY VẤN DỮ LIỆU

2.1 Truy vấn chọn (select query)

Khi có nhu cầu thể hiện các dòng của bảng hay của nhiều bảng dưới dạng một bảng có số cột và số dòng theo ý muốn như bảng điểm của sinh viên, danh sách sinh viên thì ta sử dụng truy vấn chọn.

Để dùng truy vấn chọn ta sử dụng câu lệnh SQL sau:

--Trình bày theo cách tổng quát đơn giản:

Select [* |**Distinct** | **top k**] <các cột cần hiển thị> [, biểu thức, hàm kết hợp]

From <các bảng tham gia> [, (truy vấn con) **As** <tên bảng mới>, điều kiện kết]
[**Where** <điều kiện lọc mẫu tin nguồn hoặc điều kiện kết>]
[**Group By** <tên các cột cần gom nhóm>]
[**Having** <Điều kiện lọc mẫu tin tổng hợp gom nhóm>]
[**Order By** <thứ tự các cột được xét dữ liệu tăng(Asc)/ giảm(Desc) dần>];

Trong đó:

Select : chọn các cột cần hiển thị kết quả.

From : từ những bảng tham gia trong câu truy vấn.

Where : những điều kiện dữ liệu cần thỏa hoặc điều kiện kết.

Group By : tên các cột cần gom nhóm.

Having : Điều kiện lọc mẫu tin tổng hợp theo các hàm gom nhóm.

Order By: sắp kết quả hiển thị theo thứ tự cột giá trị tăng dần (Asc, mặc định) hoặc giảm dần (Desc)

Thứ tự thực hiện các mệnh đề trong câu truy vấn:

From -> Where -> Group by -> Having ->Order by -> Select

***** : lấy hết tất cả các cột trong quan hệ nguồn.

Distinct: loại bỏ các bộ trùng trong quan hệ đích (quan hệ kết quả).

Điều kiện kết : <Quan hệ 1> **Inner Join** <Quan hệ 2> **ON** <QuanHe1>.ThuocTinh = QuanHe2.ThuocTinh : mỗi mẫu tin của quan hệ 1 nối với bất kỳ mẫu tin nào của quan hệ 2 mà dữ liệu của thuộc tính thuộc quan hệ 1 thỏa mãn điều kiện kết so sánh với dữ liệu của thuộc tính thuộc quan hệ 2 tạo thành mẫu tin của truy vấn.

Chú ý:

Khi nêu rõ một thuộc tính thuộc về quan hệ nào ta viết theo cú pháp tênQuanHệ.tênThuộcTính.

Ví dụ : cho lược đồ quan hệ QUẢN LÝ ĐIỂM SAU (QLD)

KHOA (MaKhoa, TenKhoa, NgayThanhLap, DienThoai, SoCBGD)

Mỗi khoa có một MaKhoa duy nhất. Mỗi mã khoa xác định tất cả các thuộc tính còn lại của khoa đó.

LOP (MaLop, TenLop, SiSo, MaKhoa)

Mỗi lớp có một mã lớp duy nhất, mỗi lớp chỉ thuộc về một khoa nào đó.

SINHVIEN (MaSV, HoTen, Nu, NgaySinh, MaLop, HocBong)

Mỗi sinh viên có một MaSV duy nhất. Mỗi MaSV xác định tất cả các thuộc tính còn lại của sinh viên đó.

MONHOC (MaMH, TenMH, SoTiet)

Mỗi môn học có một Mamh duy nhất. Mỗi Mamh xác định tất cả các thuộc tính còn lại của môn học.

KETQUA (MaSV, MaMH, NgayDangKy, DiemThi)

Mỗi sinh viên cùng với một môn học và một ngày đăng ký xác định duy nhất một điểm thi.

Ví dụ 1: Lập danh sách sinh viên gồm MaSV, HoTen, HocBong

```
select MaSV, HoTen, HocBong  
from SINHVIEN;
```

Ví dụ 2: Lập danh sách sinh viên nữ , có học bổng thuộc lớp có mã lớp là “CDTH2B”.

```
select MaSV, HoTen, HocBong  
from SINHVIEN  
where Nu and MaLop = 'CDTH2B' and HocBong > 0;
```

Ví dụ 3: Lập danh sách sinh viên gồm MaSV, HoTen, HocBong trong đó học bổng được sắp giảm dần

```
select MaSV, HoTen, HocBong  
from SINHVIEN  
order by HocBong desc;
```

Ví dụ 4: Lập danh sách sinh viên gồm MaSV, HoTen, TenLop

Cách 1:

```
select MaSV, HoTen, TenLop  
from LOP inner join SINHVIEN on LOP.MaLop =  
SINHVIEN.MaLop;
```

Cách 2 :

```
select MaSV, HoTen, TenLop  
from SINHVIEN, LOP  
where LOP.MaLop = SINHVIEN.MaLop
```

Ví dụ 5: Lập danh sách sinh viên có MaSV, HoTen, Nu, NgaySinh, DiemThi với điểm thi môn 'CSDL' >= 8

Cách 1:

```
select KETQUA.MaSV, HoTen, Nu, NgaySinh, DiemThi  
from SINHVIEN inner join KETQUA on SINHVIEN.MaSV =  
KETQUA.MaSV  
where KETQUA.MaMH = 'CSDL' and KETQUA.DiemThi >= 8  
order by DiemThi desc, NgaySinh asc;
```

Cách 2 :

```
select KETQUA.MaSV, HoTen, Nu, NgaySinh, DiemThi
```

from SINHVIEN, KETQUA

where SINHVIEN.MaSV = KETQUA.MaSV and KETQUA.MaMH = 'CSDL' and KETQUA.DiemThi >= 8

order by DiemThi desc, NgaySinh asc;

Ví dụ 6: Lập danh sách sinh viên có họ là “Nguyễn”

select * from SINHVIEN

where HoTen like N'Nguyễn%';

*(dùng * hoặc % để đại diện cho nhiều ký tự)*

2.2 Truy vấn nhóm dữ liệu (Select query có group by)

Khi có nhu cầu chia các bộ theo từng nhóm rồi tính toán trên từng nhóm này như tính điểm trung bình của tất cả các môn học của từng sinh viên, hay là cần tính số lượng sinh viên của mỗi lớp, hay là cần biết tổng số môn mà một sinh viên đã đăng ký học thì ta sử dụng truy vấn theo nhóm.

Để truy vấn nhóm ta lưu ý:

+ **Group By <tên các cột cần gom nhóm>**: tên các cột mà các mẫu tin có dữ liệu giống nhau trên cột được xếp vào cùng nhóm.

+ **Having <điều kiện lọc mẫu tin sau tổng hợp>**: điều kiện mà các mẫu tin tổng hợp đã gom nhóm phải thỏa mãn.

+ Các hàm dùng trong gom nhóm : MAX, MIN, AVG, COUNT, COUNT_BIG, VAR, CHECKSUM, CHECKSUM_AGG, STDEV, STDEVP, GROUPING, VARP. Ngoại trừ hàm Count, các hàm còn lại bỏ qua cột có giá trị null.

Lưu ý : + Các cột được đưa vào mệnh đề Select (ngoại trừ biểu thức) phải giống các cột trong mệnh đề Group. Trong đó các cột chính phải có giá trị tương quan 1-1 với các cột phụ (thường trong cùng một quan hệ). Nếu

không sẽ làm cho ý nghĩa của việc gom nhóm bị sai lệch dẫn đến kết quả sai.

Ví dụ 1: Lập danh sách sinh viên có đăng ký ít nhất là ba môn học

```
select KETQUA.MaSV, HoTen, Nu, NgaySinh, count(MaMH) as SLMH
from SINHVIEN inner join KETQUA on SINHVIEN.MaSV =
KETQUA.MaSV
group by KETQUA.MaSV, HoTen, Nu, NgaySinh
having count(MaMH) >= 3;
```

Ví dụ 2 (đơn giản): KETQUA(MaSV, MaMH, DiemThi)

```
select MaSV, count(MaMH)
from KETQUA
group by MaSV
```

Ví dụ 3: Tổng số lượng sinh viên nữ của mỗi khoa

```
select KHOA.MaKhoa, TenKhoa, count(KHOA.MaKhoa) as SoLuong
from (SINHVIEN inner join LOP on SINHVIEN.MaLop = LOP.MaLop)
inner join KHOA on LOP.MaKhoa = KHOA.MaKhoa
where Nu
group by KHOA.MaKhoa, TenKhoa;
```

2.3 Truy vấn lồng nhau (nested query, sub query)

Truy vấn lồng nhau là những câu lệnh truy vấn mà trong thành phần WHERE hay HAVING có chứa thêm một câu lệnh Select khác. Câu lệnh select khác này gọi là subquery. Ta lồng câu Select vào phần Where hay Having theo cú pháp sau:

- **biểu thức toán Tử So Sánh [ANY | ALL | SOME] (câuLệnhSQL)**

ANY, SOME : bất kỳ.

ALL : tất cả.

Các mẫu tin của query chính thỏa mãn toán tử so sánh với bất kỳ/ tất cả mẫu tin nào của subquery

- **bieuThuc [NOT] IN (cauLenhSQL) : so sánh cột giá trị**

Các mẫu tin của query chính có giá trị bằng với một giá trị trong subquery

- **[NOT] EXISTS (cauLenhSQL) : so sánh tồn tại mẫu tin.**

Các mẫu tin của query chính thỏa mãn khi subquery có mẫu tin.

Ví dụ 1: Lập danh sách sinh viên thuộc lớp có mã lớp 'cdltmtk3a' có học bổng cao hơn tất cả các sinh viên thuộc lớp có mã lớp 'cdltmtk3b'

```
select *  
from SINHVIEN SV  
where MaLop = 'cdltmtk3a' and SV.HocBong > all (  
    select HocBong  
    from SINHVIEN  
    where MaLop = 'cdltmtk3b'  
);
```

Ví dụ 2: Lập danh sách sinh viên có điểm thi môn CSDL cao nhất

```
select KETQUA.MaSV, HoTen, Nu, NgaySinh, DiemThi  
from SINHVIEN inner join KETQUA on SINHVIEN.MaSV =  
KETQUA.MaSV  
where MaMH = 'CSDL' and DiemThi >= all  
    ( select DiemThi  
      from KETQUA  
      where MaMH = 'CSDL'  
    );
```

Lưu ý :

+ Miền giá trị của các cột trong mệnh đề *select* trong câu truy vấn con phải bằng miền giá trị của các cột trong mệnh đề *where* trong câu truy vấn cha.

+ Nếu một bảng được dùng 2 lần trong câu truy vấn thì nên đổi tên quan hệ trong câu truy vấn cha.

Hiểu và vận dụng tốt lệnh truy vấn dữ liệu là một việc làm cực kỳ cần thiết để tạo ra các kết quả cho báo cáo, thống kê số liệu.

2.4 Truy vấn hợp (union [all]), trừ (Except), giao (Intersect)

Khi có nhu cầu thực hiện truy vấn có kết quả như toán tử hợp, ta sử dụng câu lệnh SQL sau:

Select

...

UNION [ALL| EXCEPT| INTERSECT]

Select

...

Ví dụ 1: Lệnh sau đây gấp đôi danh sách sinh viên

select MaSV, HoTen, NgaySinh

from SINHVIEN

union all

select MaSV, HoTen, NgaySinh

from SINHVIEN

Ví dụ 2: Tập hợp các sinh viên thuộc lớp có mã 'cdltmtk3a' và 'cdltmtk3b'

select *

from SINHVIEN

where MaLop = 'cdltmtk3a'

union

select *

from SINHVIEN

where MaLop = 'cdltmtk3b'

Ví dụ 3: lấy danh sách sinh viên có mã lớp là 'CDLTMTK2' và 'CDCNTTK2'

select *

from SINHVIEN

where MaLop = 'CDLTMTK2' or MaLop = 'CDCNTTK2'

Cách 2 :

(select *

from SINHVIEN

where MaLop = 'CDLTMTK2'

)

union

(select *

from SINHVIEN

where MaLop = 'CDCNTTK2'

)

3. CẬP NHẬT DỮ LIỆU ĐƯỢC TỔNG HỢP TỪ NHỮNG DỮ LIỆU CÓ SẴN

3.1 Thêm một hay nhiều bộ giá trị vào một bảng từ bảng có sẵn.

Cú pháp:

INSERT INTO <tên bảng> [(<tên cột 1>, <tên cột 2>, ... <tên cột n>)]

```
SELECT <biểu thức 1>, <biểu thức 2>, ... <biểu thức n>  
FROM <danh sách các bảng nguồn>
```

3.2 Tạo một bảng mới với các bộ giá trị được lấy từ một bảng đã có sẵn.

Cú pháp:

```
SELECT <biểu thức 1>, <biểu thức 2>, ... <biểu thức n> INTO  
<đặt tên bảng mới>  
FROM <danh sách các bảng nguồn>  
[WHERE <điều kiện>]
```

Ví dụ 1: lấy danh sách các sinh viên thuộc khoa “Công nghệ thông tin” và đưa vào một bảng mới có tên là “SVKCNTT”

```
select SV.MaSV, SV.HoTen, SV.NgaySinh, LOP.TenLop,  
KHOA.TenKhoa into SVKCNTT  
from (SINHVIEN SV inner join LOP on SV.MaLop = LOP.MaLop)  
inner join KHOA on LOP.MaKhoa = KHOA.MaKhoa  
where KHOA.TenKhoa = N'Công nghệ thông tin' and SV.MaLop =  
LOP.MaLop and LOP.MaKhoa = KHOA.MaKhoa
```

Ví dụ 2: cho biết số học sinh của mỗi khoa và đưa vào bảng mới “SHSMK”

```
select KHOA.MaKhoa, KHOA.TenKhoa, count(KHOA.MaKhoa)  
into SHSMK  
from SINHVIEN SV, LOP, KHOA  
where SV.MaLop = LOP.MaLop and LOP.MaKhoa =  
KHOA.MaKhoa  
group by KHOA.MaKhoa, KHOA.TenKhoa  
(phép kết nằm trong mệnh đề Where)
```

4. CÂU HỎI ÔN TẬP

- 1/- Hãy nêu cú pháp thêm, sửa, xóa dữ liệu bằng phát biểu T-SQL?
- 2/- Hãy nêu cú pháp truy vấn chọn, nhóm, lồng nhau, tập hợp dữ liệu bằng phát biểu T-SQL?
- 3/- Hãy nêu cú pháp truy vấn cập nhật dữ liệu được tổng hợp từ những dữ liệu có sẵn?
- 4/- Hãy dùng cửa sổ soạn thảo (Query Editor) để làm bài tập 1 và 2 phần B luyện tập TRUY VẤN DỮ LIỆU.

BÀI 5: BẢO MẬT CƠ SỞ DỮ LIỆU TRONG MS SQL SERVER

Mã bài: MD15_05

Giới thiệu:

Bài học giúp sinh viên định nghĩa và phân quyền người sử dụng từ đó thiết lập được bảo mật cho cơ sở dữ liệu mong muốn.

Mục tiêu:

- Trình bày được các giải pháp bảo mật cơ sở dữ liệu trong SQL Server;
- Nêu được vai trò của người dùng trên hệ quản trị cơ sở dữ liệu và trên cơ sở dữ liệu;
- Sử dụng một số công cụ để định nghĩa và phân quyền người sử dụng;
- Rèn luyện thái độ làm việc nghiêm túc, tỉ mỉ, cẩn thận

1. TẠO VÀ CẤP QUYỀN TÀI KHOẢN ĐĂNG NHẬP SQL SERVER

Trong bất kỳ hệ thống cơ sở quản trị dữ liệu nào, vấn đề an ninh, bảo mật luôn được đặt lên hàng đầu. Vì vậy, với SQL Server, nếu chúng ta tận dụng được nhiều ưu điểm của ứng dụng này, những người làm công việc quản trị hệ thống sẽ giảm bớt được rất nhiều gánh nặng cũng như áp lực trong công việc. Hướng dẫn sau đây sẽ giới thiệu với các bạn một số thông tin cơ bản cũng như cách sử dụng, quản lý role của SQL Server.

Bước đầu tiên cần thực hiện trong toàn bộ quá trình bảo mật dữ liệu cho người dùng là xác định rõ ràng những tài khoản nào sẽ được quyền truy cập, xem hoặc chỉnh sửa dữ liệu. Ví dụ, các trưởng bộ phận có thể xem được tài khoản lương của nhân viên, trong khi các bậc quản lý ở cấp cao hơn có quyền xem và chỉnh sửa, trong khi đó nhân viên chỉ có thể xem được tài khoản của chính họ.

Tiếp theo, cần phải xác định rõ tài khoản nào sẽ được cấp quyền điều chỉnh, thay đổi cơ sở dữ liệu. Do vậy, tùy từng mô hình hệ thống, quy mô của công ty, tổ chức mà khối lượng công việc của người quản trị cũng sẽ tăng lên. Bên cạnh đó, do kiến thức và kinh nghiệm sử dụng của mỗi người khác nhau, việc làm sao đảm bảo được mức an toàn tối thiểu cho toàn bộ nhân viên trở nên vất vả hơn rất nhiều. Sau đây, chúng ta sẽ đề cập đến khái niệm role (Role là một công cụ cho phép ta tập hợp các người dùng (user id) vào cùng một nhóm nhờ đó ta có thể gán quyền chung cho cả nhóm đó) và mối quan hệ với Windows Group, làm thế nào để gán thêm hoặc từ chối quyền truy cập của một hoặc nhiều tài khoản người dùng trong thời gian hoạt động.

Về mặt bản chất, role là 1 phần của **tiered security model**:

- **Login security**: thực hiện quá trình kết nối tới server
- **Database security**: nhận quyền truy cập tới cơ sở dữ liệu
- **Database object**: nhận quyền truy cập tới từng đối tượng và dữ liệu riêng biệt trong toàn hệ thống.

Trước tiên, người dùng phải tiến hành đăng nhập vào server bằng cách nhập mật khẩu, sau khi quá trình kết nối này hoàn tất, việc truy cập tới các cơ sở dữ liệu đã lưu trữ sẽ được thực hiện qua việc chỉ định tài khoản. Khi người quản trị đã hoàn tất việc gán quyền cho những tài khoản này, họ sẽ không thể truy cập đến những vùng dữ liệu không được phép khác.

Tuy nhiên, ưu điểm lớn nhất của việc sử dụng role chính là hiệu quả trong quá trình quản lý. Các bạn hãy thử tưởng tượng, nếu 1000 nhân viên cần xem hoặc chỉnh sửa dữ liệu cá nhân của họ sớm nhất có thể, thì bộ phận quản trị hệ thống chỉ việc lựa chọn trong Windows Group có sẵn, và sau đó là gán toàn bộ vào **SQL Server role** tương ứng – thay vì việc phải chỉnh sửa 1000 tài khoản lần lượt theo cách thủ công. Hiểu rõ hơn về trường hợp này, Windows Group có chứa toàn bộ tài khoản người dùng với quyền truy cập tương ứng tới hệ thống mạng trên Windows, và các role của **SQL Server** sẽ phụ thuộc toàn bộ vào những thành phần có liên quan. Do vậy, chúng ta chỉ cần gán quyền theo nhu cầu với cơ sở dữ liệu của SQL Server và tài khoản tương ứng trên Windows.

Các role của server thường được giám sát và quản lý bởi **Database Administrator – DBA** và áp dụng được với toàn bộ server, chứ không riêng gì đối với từng thành phần riêng rẽ. Ở chế độ mặc định, các role này được thiết lập public đối với tất cả các tài khoản, và toàn bộ những tài khoản sau khi thêm vào SQL Server cũng sẽ tự động được gán role public.

Với các role cố định trong Server Role thì chúng được áp dụng trên toàn bộ mô hình của hệ thống. Khi tạo một tài khoản đăng nhập (login user) ta cấp quyền đăng nhập SQL Server với một số điểm lưu ý chung như sau (Server role):

- **SysAdmin**: toàn bộ người dùng đều có thể thực hiện các thao tác trên server.
- **ServerAdmin**: toàn bộ người dùng đều có thể thiết lập, tùy chỉnh các phương án lựa chọn trên server.

- **SetupAdmin:** toàn bộ người dùng đều có thể quản lý các server đã kết nối, những tùy chọn và tác vụ hoạt động của SQL Server.
- **SecurityAdmin:** toàn bộ người dùng đều có thể quản lý các thành phần có liên quan đến an ninh, bảo mật.
- **ProcessAdmin:** toàn bộ người dùng đều có thể tắt hoặc tạm dừng bất kỳ tiến trình nào hoạt động trên SQL Server.
- **DbCreator:** toàn bộ người dùng có thể tạo, thay đổi, xóa hoặc khôi phục cơ sở dữ liệu.
- **DiskAdmin:** toàn bộ người dùng có thể quản lý các file của SQL Server.
- **BulkAdmin:** toàn bộ người dùng có thể thực hiện nhiều lệnh insert khác nhau.
- **Public:** toàn bộ người dùng có quyền hạn chế nhất.

Việc tạo cơ sở dữ liệu là của riêng người quản trị nhưng các bạn cần lưu ý một số điểm sau về quy chuẩn chung khi cấp quyền người dùng trên cơ sở dữ liệu (User Mapping):

- **db_owner:** toàn bộ người dùng có quyền full – access.
- **db_accessadmin:** người dùng có quyền quản lý các Windows Group và tài khoản SQL Server đăng nhập.
- **db_datareader:** người dùng có thể đọc được toàn bộ dữ liệu.
- **db_datawriter:** người dùng có quyền thêm, xóa hoặc chỉnh sửa dữ liệu trong bảng.
- **db_ddladmin:** người dùng có thể sử dụng các file dynamic – link library (DLL).
- **db_securityadmin:** người dùng có thể chỉnh sửa vai trò role và quản lý các bậc quản lý, phân quyền khác.
- **db_backupoperator:** người dùng có thể sao lưu cơ sở dữ liệu.
- **db_denydatareader:** người dùng không thể xem dữ liệu trong bảng.
- **db_denydatawriter:** người dùng không thể xem, thay đổi hoặc xóa dữ liệu trong bảng.

Để có thể hình dung dễ dàng cách thức phân quyền của SQL Server, chúng ta cần nêu ra hai cách thức phân quyền sau:

- + Phân quyền thông qua giao diện của Microsoft SQL Server.
- + Phân quyền bằng phát biểu SQL (viết code).

1.1 Tạo tài khoản đăng nhập

1.1.1 Tạo tài khoản đăng nhập SQL Server

Nếu tài khoản dùng để đăng nhập SQL Server (login user) được cấp quyền tạo cơ sở dữ liệu (DbCreator) khi tài khoản được tạo ra thì người sử dụng đăng nhập bằng tài khoản này có thể tạo ra các cơ sở dữ liệu.

Khi tạo cơ sở dữ liệu trong SQL Server, tài khoản đang đăng nhập sẽ là chủ nhân (dbo) của cơ sở dữ liệu vừa tạo ra. Như vậy, một khi login user là dbo thì họ có toàn quyền trên cơ sở dữ liệu do họ làm chủ nhân.

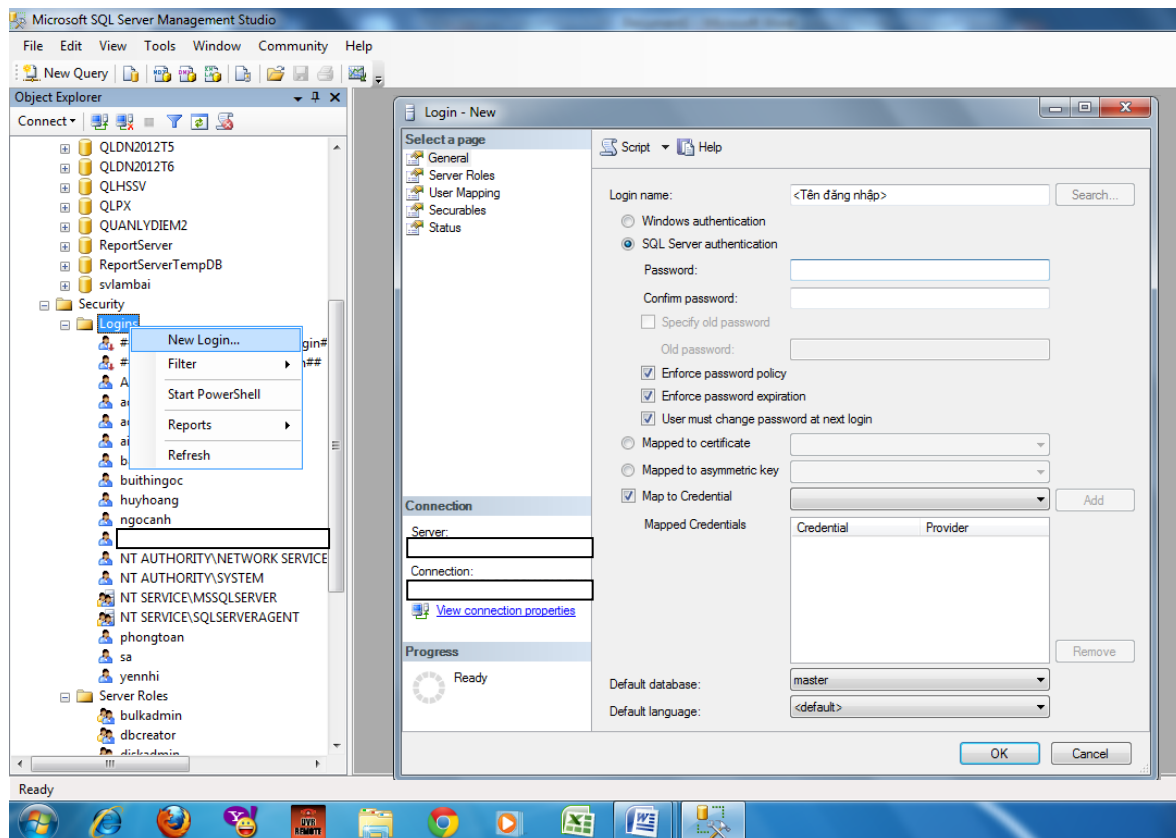
Tuy nhiên, nếu một login user khác muốn làm việc trên cơ sở dữ liệu mà họ không phải là dbo thì họ cần phải được sự cho phép của admin hay dbo.

Trong SQL Server, bạn cũng có thể khai báo nhiều login user cùng là chủ sở hữu một cơ sở dữ liệu bằng cách chọn vào tùy chọn db_owner từ giao diện đồ họa hay phát biểu SQL.

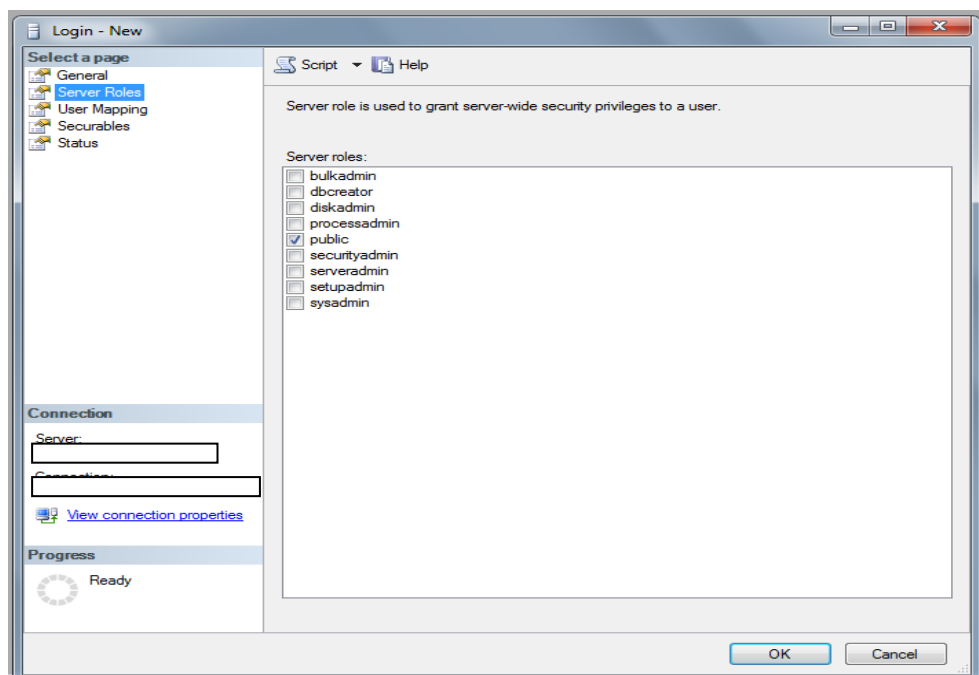
Tài khoản nhóm SysAdmin hay sa sẽ có toàn quyền trên các cơ sở dữ liệu trong SQL Server.

Để tạo tài khoản đăng nhập SQL Server: Security --> Login --> New Login...:

- + General: chọn tên đăng nhập lấy từ Windows User hoặc tạo tên đăng nhập mới theo chứng thực trong SQL Server.

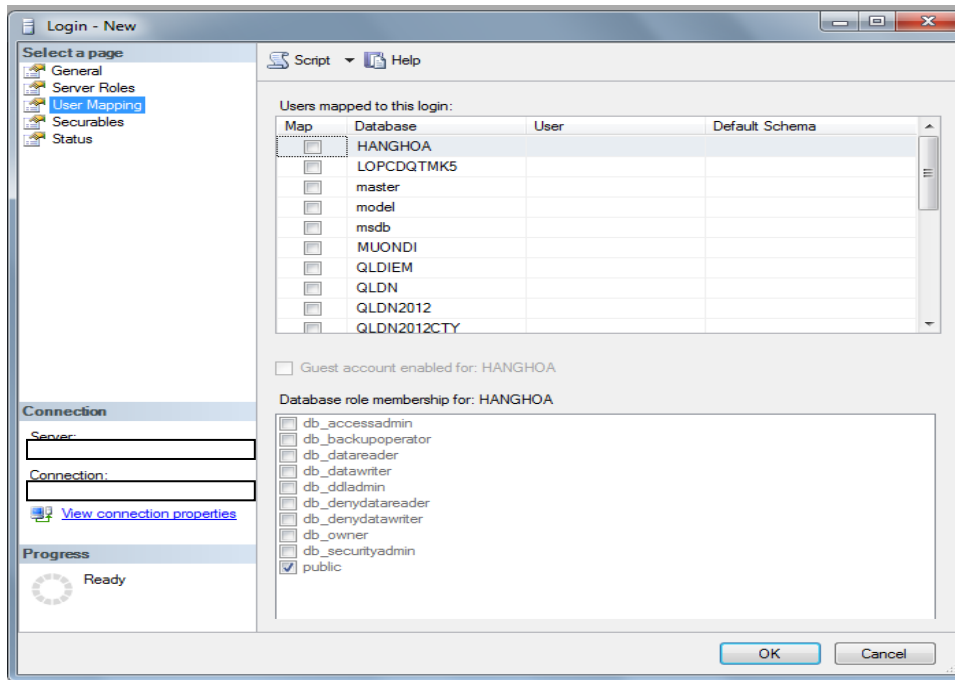


+ Server Roles: cấp quyền đăng nhập SQL Server cho login user đang tạo. Quyền mặc định là public (chỉ mỗi có quyền đăng nhập vào SQL Server, không được tạo các đối tượng khác...). Để có thể thao tác trên SQL Server ta phải cấp quyền cho nó qua mục Server role.



+ User Mapping: chọn cơ sở dữ liệu đã tồn tại và cấp quyền truy cập trên cơ sở dữ liệu đã tồn tại cho login user đang tạo. Quyền mặc định là public (chỉ mỗi có

quyền truy cập vào cơ sở dữ liệu, không được tạo các đối tượng khác...) (xem mục 2)



1.1.2 Tạo tài khoản đăng nhập bằng phát biểu SQL

Cú pháp tạo login user theo chứng thực SQL Server:

Create Login <đặt_tên_login_user>

WITH Password = 'mật_khẩu' [MUST_CHANGE]

[, DEFAULT_DATABASE = <tên_cơ_sở_dữ_liệu>]

[, CHECK_EXPIRATION = ON|OFF]

Cú pháp tạo login user theo user được lấy từ hệ điều hành Windows:

Create Login <đặt_tên_login_user_lấy_từ_user_windows>

FROM WINDOWS

[, DEFAULT_DATABASE = <tên_cơ_sở_dữ_liệu>]

Trong đó:

+ **WITH Password = 'mật_khẩu'**: là các tùy chọn tương ứng với mật khẩu của login user đặt mới nếu nó không thuộc hệ điều hành.

- + **MUST_CHANGE**: SQL Server sẽ yêu cầu phải đổi mật khẩu truy cập lần đầu tiên vào SQL Server với login user này.
- + **DEFAULT_DATABASE** = <tên_cơ_sở_dữ_liệu>: là khai báo đăng nhập vào cơ sở dữ liệu ngay sau khi tạo login.
- + **CHECK_EXPIRATION** = ON|OFF: là khai báo được (ON) hoặc tạm ngưng (OFF) thời hạn đăng nhập.

Ví dụ 1: tạo login user theo cú pháp:

Create login <tên_login> With password = <'mật khẩu'>

Create login TruongThiThanhThao With password = '123cdndl'

Ví dụ 2: tạo login user đồng thời khai báo đăng nhập cơ sở dữ liệu HANGHOA:

Create login TruongThiThanhThao1 With password = '123cdndl'
MUST_CHANGE,

DEFAULT_DATABASE = HANGHOA,

CHECK_EXPIRATION = ON

Ví dụ 3: tạo login user từ các user của hệ điều hành windows theo cú pháp:

Create login [tên_máy\tên_windows_user] From WINDOWS

Create login [TRUONGTHITHANHTHAO-PC\TruongThiThanhThao]

From WINDOWS

Ví dụ 4: Liệt kê danh sách các login user:

Select name, createdate From syslogins

Ví dụ 5: Liệt kê danh sách các login user có nguồn gốc từ SQL Server:

Select name, createdate From syslogins Where isntname = 0

Ví dụ 6: Liệt kê danh sách các login user có nguồn gốc từ Windows:

```
Select name, createdate From syslogins Where isntname = 1
```

Tương tự, chúng ta có thể sửa đổi hoặc xóa login user như sau:

+ Sửa đổi login user: Alter login thay cho Create login trong cú pháp tạo login.

Ví dụ 7: sửa đổi mật khẩu login user TruongThiThanhThao:

```
Alter login TruongThiThanhThao With password = 'cdnd1123cdnd1'
```

+ Xóa login user: Drop login <tên_login>.

Ví dụ 8: xóa login user TruongThiThanhThao

```
Drop login TruongThiThanhThao
```

Ngoài hai cách tạo login như trên, chúng ta có thể sử dụng thủ tục nội tại của hệ thống do SQL Server cung cấp sp_AddLogin để tạo và sp_DropLogin để xóa login user:

+ Tạo login user: sp_AddLogin 'tên_login_user', 'mật_khẩu'| null, 'tên_cơ_sở_dữ_liệu'.

Ví dụ 9: tạo login user TruongThiThanhThao, mật khẩu rỗng vào HANGHOA

```
exec sp_addlogin 'TruongThiThanhThao', null, 'HANGHOA'
```

+ Xóa login user: sp_DropLogin <tên_login_user>.

Ví dụ 10: xóa login user TruongThiThanhThao

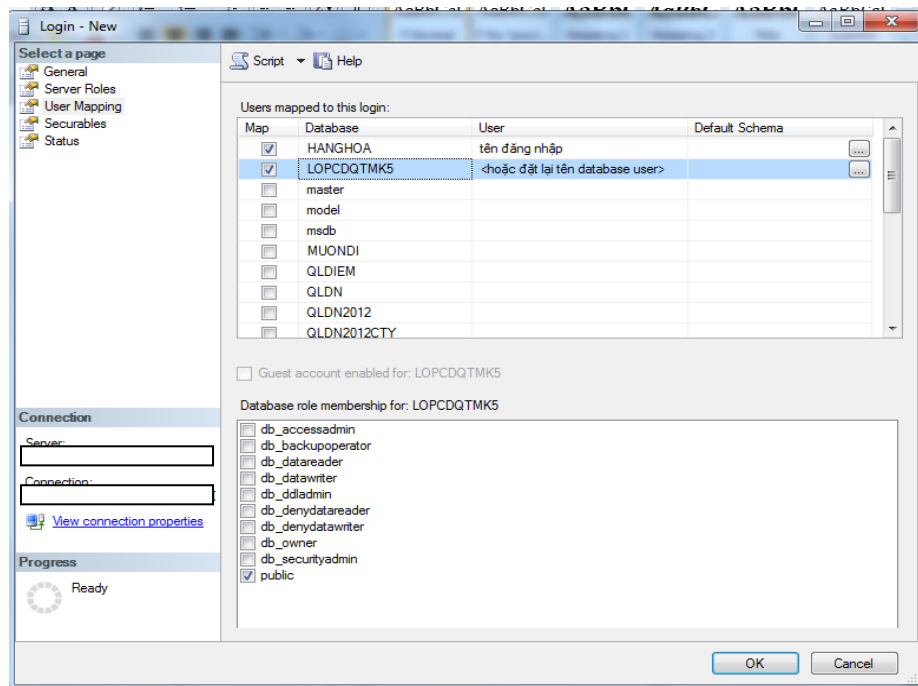
```
exec sp_droplogin TruongThiThanhThao
```

1.1.3 Tạo tài khoản truy cập cơ sở dữ liệu.

1.1.3.1 Tạo tài khoản truy cập SQL Server

Khi login user được thêm vào cơ sở dữ liệu thì login user này được gọi là database user. Để database user này có thể tạo được các đối tượng trên cơ sở dữ

liệu đã chọn thì ta phải cấp quyền cho nó. Quyền mặc định là public. Nếu muốn cấp quyền như chủ nhân của cơ sở dữ liệu thì ta chọn quyền db_owner.



Sau khi login user được khai báo cơ sở dữ liệu và database user để truy cập cơ sở dữ liệu xong thì trong mục Security\User của cơ sở dữ liệu này xuất hiện tên của database user vừa được tạo để truy cập vào cơ sở dữ liệu. Tuy nhiên để có thể thao tác trên cơ sở dữ liệu ta phải cấp quyền cho database user này qua mục Database role membership (xem mục 2.3).

Ngoài ra chúng ta có thể vào trực tiếp trong mỗi cơ sở dữ liệu để tạo database user bằng cách: Chọn cơ sở dữ liệu --> Security --> User --> Click phải chuột chọn New User.

1.1.3.2 Tạo tài khoản truy cập cơ sở dữ liệu bằng phát biểu SQL

Ta có thể khai báo database user từ login user bằng phát biểu SQL dạng Create User với cú pháp như sau:

Create user <đặt_tên_user>

FOR LOGIN <tên_login_đã_có>

[WITH DEFAULT_SCHEMA = <tên_schema>]

Ví dụ 1: tạo database user cho cơ sở dữ liệu hiện tại theo cú pháp:

Create user <đặt_tên_user> FOR LOGIN <tên_login_tương_ứng_đã_có>

Create user TruongThiThanhThao FOR LOGIN TruongThiThanhThao

(gợi ý: thông thường người ta tạo tên database user giống tên login user)

Ví dụ 2: tạo database user và đồng thời khai báo default scheme theo cú pháp:

Create login [tên_máy\tên_windows_user] From WINDOWS

Create user TruongThiThanhThao FOR LOGIN TruongThiThanhThao

WITH DEFAULT_SCHEMA = dbo;

Tương tự, chúng ta có thể sửa đổi hoặc xóa database user như sau:

+ Sửa đổi database user: Alter user thay cho Create user trong cú pháp tạo user.

Ví dụ 3: đặt login user TruongThiThanhThao1 cho database user TruongThiThanhThao

USE HANGHOA;

Alter user TruongThiThanhThao FOR LOGIN TruongThiThanhThao1

+ Xóa database user: Drop user <tên_user>.

Ví dụ 4: xóa database user TruongThiThanhThao

USE HANGHOA;

Drop user TruongThiThanhThao

Ngoài hai cách tạo database user như trên, chúng ta có thể sử dụng thủ tục nội tại của hệ thống do SQL Server cung cấp để tạo và xóa database user:

+ Tạo database user: sp_AddUser.

Ví dụ 5: tạo database user TruongThiThanhThao trên cơ sở dữ liệu HANGHOA

use HANGHOA

exec sp_adduser TruongThiThanhThao

+ Xóa database user: sp_DropUser.

Ví dụ 6: xóa database user TruongThiThanhThao trên cơ sở dữ liệu HANGHOA

use HANGHOA

exec sp_dropuser TruongThiThanhThao

+ Ta cũng có thể liệt kê danh sách các database user:

Ví dụ 7: lấy ra loại tài khoản là người sử dụng (gid = 0)

Select name, createdate

From sysusers

Where gid = 0

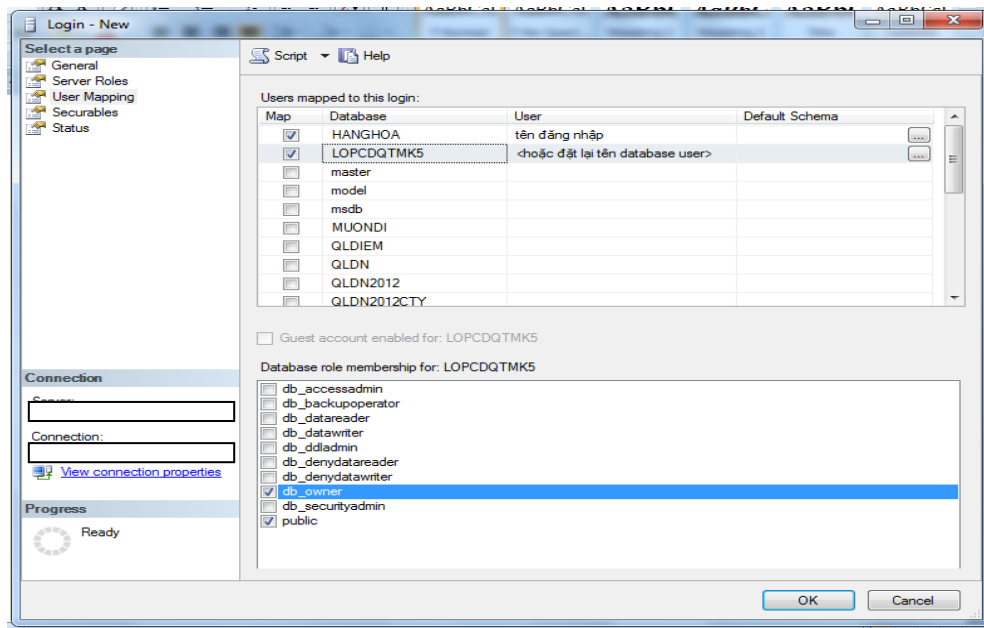
+ Để tìm hiểu các thủ tục nội tại ta dùng phát biểu: sp_helptext <tên_thủ_tục>

Ví dụ: exec sp_helptext sp_addlogin

2. CẤP QUYỀN NGƯỜI SỬ DỤNG TRÊN ĐỐI TƯỢNG CƠ SỞ DỮ LIỆU.

2.1 Cấp quyền người sử dụng trên đối tượng cơ sở dữ liệu bằng SQL Server.

Sau khi cấp quyền người sử dụng (database user) truy cập cơ sở dữ liệu xong, để có thể thao tác trên cơ sở dữ liệu ta phải cấp quyền cho database user này qua mục Database role membership.



2.2 Cấp quyền người sử dụng trên đối tượng cơ sở dữ liệu bằng lệnh SQL

Nếu muốn cho phép database user có thể tạo các đối tượng cơ sở dữ liệu như Table, View, Stored Procedure, Login thì ta phải cấp quyền cơ bản như: CREATE DATABASE, CREATE TABLE, CREATE VIEW, CREATE PROCEDURE, CREATE LOGIN bằng phát biểu GRANT theo cú pháp sau:

GRANT <permission> [, n ...]

TO <database_principal>[, n ...]

[WITH GRANT OPTION] [AS <database_principal>]

Trong đó:

+ Permission: chỉ định các quyền cho phép người sử dụng bao gồm các hành động ứng với các phát biểu DDL như: CREATE, ALTER, DROP và EXECUTE trên đối tượng cơ sở dữ liệu và các phát biểu DML ứng với thao tác dữ liệu như: SELECT, INSERT, DELETE và UPDATE.

Permission::= permission | ALL [PRIVILEGES]

+ Tùy chọn ALL cho phép người sử dụng thực hiện các phát biểu như: BACKUP DATABASE, BACKUP LOG, CREATE DATABASE, CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE RULE, CREATE TABLE và CREATE VIEW.

+ database_principal: cấp các quyền trên các database user, database role, or application role.

Ví dụ 1: cho phép database user TruongThiThanhThao được tạo bảng và khung nhìn trong cơ sở dữ liệu HANGHOA.

Use HANGHOA;

GRANT

CREATE TABLE, CREATE VIEW

TO TruongThiThanhThao

GO

Ví dụ 2: sau khi đã cấp phát quyền ta có thể thu hồi quyền bằng phát biểu REVOKE thay cho GRANT

Use HANGHOA;

REVOKE

CREATE TABLE, CREATE VIEW

TO TruongThiThanhThao

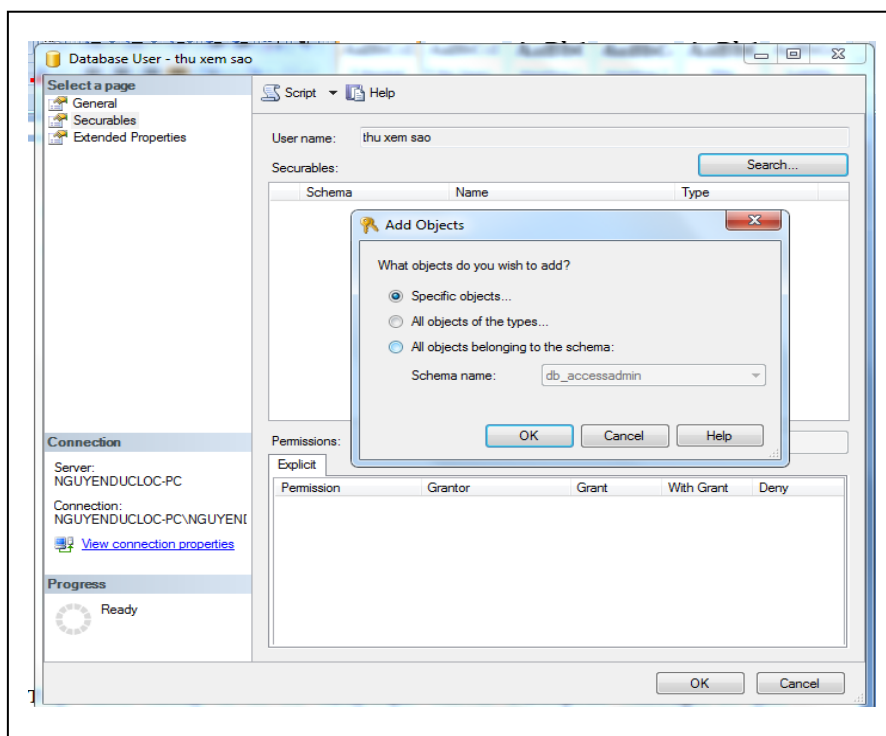
GO

3. CẤP QUYỀN NGƯỜI SỬ DỤNG TRÊN DỮ LIỆU

3.1 Cấp quyền người sử dụng trên dữ liệu

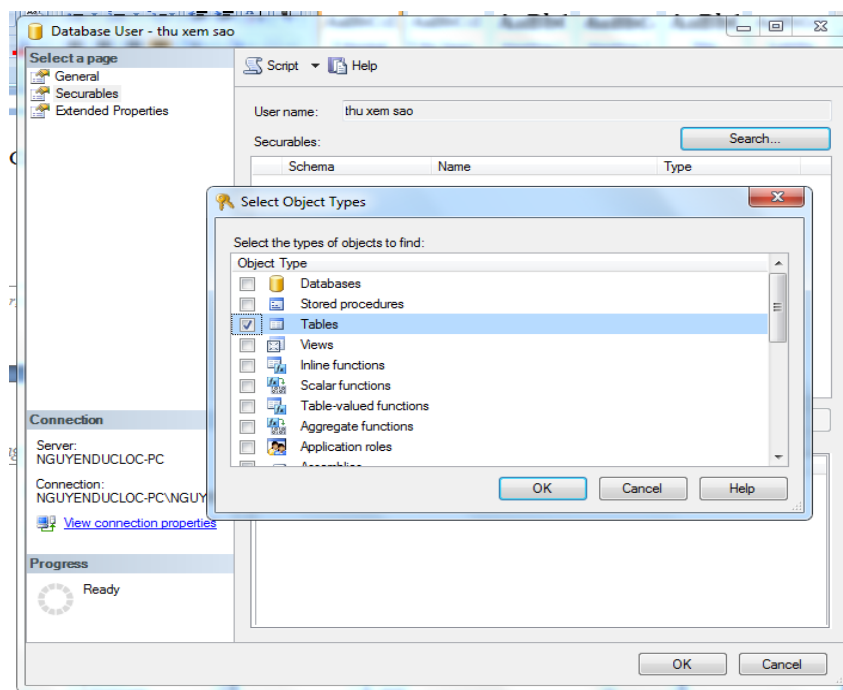
Nếu ta muốn cấp quyền cho từng người dùng đối với từng đối tượng trong một cơ sở dữ liệu thì ta chọn: Cơ sở dữ liệu --> Security --> User --> Phải chuột trên Database User cần cấp quyền --> Properties --> Securables --> Search --> All

objects of the types (hoặc Specific objects) --> OK để chọn loại đối tượng trong cơ sở dữ liệu cần cấp quyền.

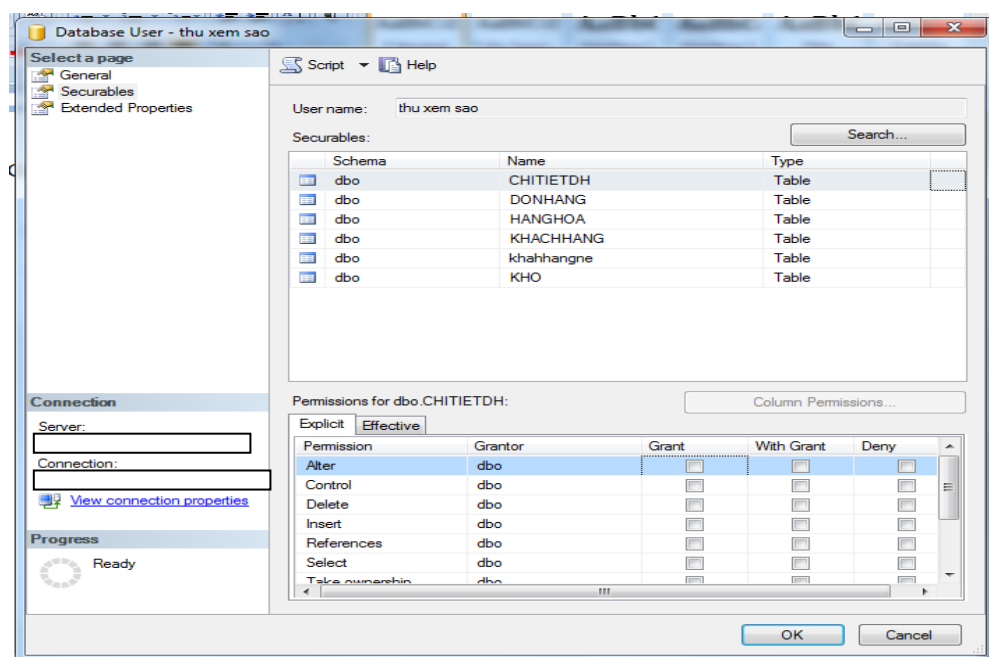


Các loại đối tượng thông thường được chọn gồm

tables, stored procedures, view. Giả sử ở đây ta chọn loại đối tượng tables và nhấn OK.



Tiếp theo ta chọn từng đối tượng dữ liệu trong phần Securables và cấp quyền (Grant) hoặc từ chối quyền (Deny) cho người sử dụng (database user) đã chọn để thiết lập quyền trên dữ liệu.



Trong trường hợp chúng ta muốn tạo một role mới để cấp cho một nhóm người dùng có quyền giống nhau, ta có thể tạo qua: Chọn cơ sở dữ liệu --> Security --> phải chuột trên Roles --> New --> New database role... --> đặt tên role

--> thao tác tương tự như cấp quyền chi tiết cho từng đối tượng dữ liệu (mục 4 trên).

3.2 Cấp quyền người sử dụng trên dữ liệu bằng ngôn ngữ SQL

Ngoại trừ chủ nhân (dbo) hay sysadmin, mọi database user khác phải được cấp quyền truy cập trên dữ liệu mới có thể thao tác dữ liệu.

```
GRANT {ALL [PRIVILEGES]}  
    | permission [(column [...n])] [, n...] [ON [CLASS::] SECURABLE]  
    TO principal [, n...] [WITH GRANT OPTION] [AS principal]
```

Ví dụ 1: khai báo cho database user TruongThiThanhThao truy vấn và xóa dữ liệu bảng KHACHHANG:

```
GRANT SELECT, DELETE  
ON KHACHHANG  
TO TruongThiThanhThao;  
GO
```

Ví dụ 2: Ta có thể sử dụng phát biểu DENY để nghiêm cấm quyền người dùng trên đối tượng cơ sở dữ liệu hay xử lý dữ liệu.

```
DENY DELETE  
ON NHANVIEN  
TO TruongThiThanhThao;  
GO
```

4. CÂU HỎI ÔN TẬP

- 1/- Hãy nêu các quyền có thể cấp cho tài khoản đăng nhập?
- 2/- Hãy nêu các quyền có thể cấp cho tài khoản người dùng?
- 3/- Hãy nêu các bước tạo tài khoản đăng nhập thông qua giao diện MS SQL Server?
- 4/- Hãy nêu các bước tạo tài khoản truy cập cơ sở dữ liệu thông qua giao diện MS SQL Server?

5/- Hãy nêu các bước cấp quyền người sử dụng trên đối tượng cơ sở dữ liệu thông qua giao diện MS SQL Server?

6/- Hãy nêu các bước cấp quyền người sử dụng trên dữ liệu thông qua giao diện MS SQL Server?

7/- Hãy nêu cú pháp lệnh tạo tài khoản đăng nhập bằng phát biểu SQL?

8/- Hãy nêu cú pháp lệnh tạo tài khoản truy cập cơ sở dữ liệu bằng phát biểu SQL?

9/- Hãy nêu cú pháp lệnh cấp quyền người sử dụng trên đối tượng cơ sở dữ liệu bằng phát biểu SQL?

10/- Hãy nêu cú pháp lệnh cấp quyền người sử dụng trên dữ liệu bằng phát biểu SQL?

11/- Hãy thực hiện các yêu cầu về cấp quyền trong bài 1 và 2 phần C luyện tập **SECURITY** bằng giao diện MS SQL Server và bằng phát biểu SQL?

BÀI 6 : THỦ TỤC LƯU TRỮ

Mã bài: MD15_06

Giới thiệu:

Bài học giúp sinh viên tạo và sử dụng được thủ tục trong SQL Server.

Mục tiêu:

- Trình bày được cú pháp và cách tạo thủ tục thông qua công cụ Management Studio;
- Tạo và sử dụng được thủ tục;
- Rèn luyện thái độ nghiêm túc, khoa học, sáng tạo.

1. CÂU LỆNH ĐIỀU KHIỂN

Khi chúng ta thực hiện một chương trình, Theo chế độ mặc định các câu lệnh được thực hiện tuần tự. Chúng ta có thể điều khiển luồng thực hiện của chương trình bằng cách sử dụng các câu lệnh điều khiển.

Trong SQL, các câu lệnh DML có thể sử dụng để thực hiện một hành động trên một tập các hàng hoặc tất cả các hàng. Tuy nhiên, một tập hợp các hành động lại được yêu cầu thực hiện trên nhiều hàng một cách lặp đi lặp lại hoặc có điều kiện. Để thực hiện điều này, SQL Server đã cung cấp ngôn ngữ lập trình có cấu trúc. Các cấu trúc này được gọi là các câu lệnh điều khiển bởi vì chúng xử lý một cách chính xác thứ tự thực hiện các đoạn code. Nhóm luồng các câu lệnh điều khiển này thực hiện các nhiệm vụ sử dụng các kỹ thuật như là quá trình xử lý lặp hoặc điều kiện giống như các ngôn ngữ lập trình.

Bảng dưới đây tóm tắt các câu lệnh điều khiển khác nhau có sẵn trong T-SQL.

<i>Cú pháp của các cấu trúc điều khiển</i>	Diễn tả
BEGIN <sql statements> END	Các câu lệnh nằm trong khối BEGIN...END được xem như một đơn vị xử lý đơn
RETURN	Sử dụng để thoát khỏi thủ tục hoặc các truy vấn.

GOTO <label> <label>:	Chuyển luồng xử lý tới một vị trí xác định được chỉ ra bởi nhãn.
IF <condition> <sql statements> ELSE <sql statements>	Cấu trúc này có thể sử dụng để thực hiện một hoặc một tập hợp các câu lệnh được lựa chọn dựa trên giá trị true-false của điều kiện.
WHILE <condition> BEGIN <sql statements> <on some condition> BREAK <on some condition> CONTINUE END	Trong cấu trúc này, một tập hợp các lệnh được thực hiện lặp đi lặp lại trong khi giá trị của điều kiện là true . Điều kiện được kiểm tra trước tiên sau đó bắt đầu xử lý khối lệnh.
BREAK	Câu lệnh này chuyển điều khiển ra ngoài vòng lặp trong luồng xử lý hiện tại.
CONTINUE	Câu lệnh này có thể sử dụng để bắt đầu thực hiện các câu lệnh trong vòng lặp tiếp theo bỏ qua các câu lệnh nằm sau lệnh này trong vòng lặp hiện tại.
CASE <expression> WHEN <expression1> THEN <expression1> [WHEN <expression2> THEN <expression2>] [ELSE <expression> END	Sử dụng cấu trúc này, tập hợp các lệnh khác nhau được xử lý dựa trên giá trị của điều kiện.

1.1 Khối lệnh BEGIN ... END

Chúng ta sử dụng BEGIN ... END để bao khối lệnh thành đơn vị xử lý đơn

Cú pháp:

```
BEGIN
{
  <Các phát biểu, thuật toán xử lý...>
}
END
```

1.2 Lệnh GOTO và RETURN

Sử dụng lệnh GOTO bất kỳ đâu trong một khối câu lệnh hay thủ tục để có thể thay đổi luồng thực hiện tới một vị trí khác (nhãn). Các câu lệnh sau từ khoá GOTO bị bỏ qua và quá trình thực hiện được tiếp tục bởi các câu lệnh sau nhãn được chỉ ra trong lệnh GOTO. Ta có thể sử dụng nhiều câu lệnh GOTO lồng nhau.

Cú pháp:

GOTO <Label>

Label là nhãn xác định vị trí mới mà tại đó các câu lệnh tiếp tục được thực hiện. Định nghĩa (*label*) phải tuân theo quy luật như định danh. GOTO có thể tồn tại trong câu lệnh điều kiện, trong khối, hoặc thủ tục, nhưng nó không thể tồn tại ở bên ngoài batch. Lệnh GOTO thay đổi luồng thực hiện tới một nhãn, nhãn có thể được định nghĩa trước hoặc sau lệnh GOTO.

Chúng ta có thể sử dụng lệnh RETURN tại bất kỳ điểm nào để thoát khỏi một khối lệnh hay thủ tục. Các câu lệnh sau RETURN ở trong khối hoặc thủ tục không được thực hiện.

Cú pháp:

RETURN [integer_expression]

Tham số trong lệnh RETURN chứa giá trị nguyên trả về. Stored procedures có thể trả về một giá trị nguyên cho thủ tục hoặc chương trình gọi nó.

1.3 Lệnh IF ELSE

Chúng ta có thể thực hiện một tập hợp các câu lệnh khác của SQL dựa trên điều kiện chỉ ra. Câu lệnh SQL sau từ khoá *IF* và điều kiện chỉ được thực hiện nếu điều kiện đúng. Trong trường hợp, điều kiện trong mệnh đề *IF* không thoả mãn (nó chứa trị sai), thì tập hợp các lệnh SQL sau từ khoá *ELSE* được thực hiện (nếu có).

Cú pháp:

```
IF <Biểu thức điều kiện>
{ <Các phát biểu SQL hoặc Khối lệnh> }
[ ELSE
{ <Các phát biểu SQL hoặc Khối lệnh> } ]
```

Biểu thức được sử dụng với IF phải trả về *TRUE* hoặc *FALSE*. Trong trường hợp chúng ta sử dụng câu lệnh SELECT trong biểu thức, câu lệnh nên được nằm trong dấu ngoặc đơn. Chúng ta cũng có thể sử dụng câu lệnh IF...ELSE lồng nhau và không có giới hạn về số mức lồng nhau.

1.4 Vòng lặp WHILE

Chúng ta có thể thực hiện một lệnh SQL hoặc một khối lệnh một cách lặp đi lặp lại bằng cách dựa trên điều kiện. Sử dụng từ khoá WHILE chúng ta có thể bảo đảm các câu lệnh được thực hiện lặp đi lặp lại trong khi điều kiện chỉ ra có giá trị đúng. Chúng ta có thể sử dụng từ khoá *CONTINUE* và *BREAK* trong vòng lặp *WHILE* để điều khiển sự thực thi của các câu lệnh.

Cú pháp:

```
WHILE <biểu thức điều kiện đúng>
{ <Các phát biểu SQL hoặc Khối lệnh> }
[ BREAK ]
{ <Các phát biểu SQL hoặc Khối lệnh> }
[ CONTINUE ]
```

Biểu thức điều kiện, nó có thể trả về giá trị *ĐÚNG* hoặc *SAI*. Trong trường hợp biểu thức điều kiện chứa câu lệnh SELECT, nó phải được bao trong dấu ngoặc đơn.

Từ khoá *BREAK* dùng để thoát khỏi vòng lặp WHILE trong cùng ngược lại từ khoá *CONTINUE* dùng để điều khiển vòng lặp được thực hiện tiếp tục bỏ qua một loạt các lệnh sau từ khoá *CONTINUE*.

Nếu chúng ta có các vòng lặp WHILE lồng nhau, câu lệnh BREAK ở vòng lặp phía trong sẽ chuyển quyền thực hiện ra vòng lặp phía ngoài cùng. Đầu tiên, các câu lệnh tại vòng lặp phía trong được kết thúc, sau đó các câu lệnh của vòng lặp phía ngoài được thực hiện.

Ví dụ 1: Sử dụng từ khoá BREAK và CONTINUE với các lệnh IF ...ELSE lồng nhau và WHILE

Trong ví dụ này, nếu giá trung bình của các cuốn sách trong bảng *titles* nhỏ hơn \$30, vòng lặp WHILE sẽ thực hiện cập nhật giá của các cuốn sách đó gấp 2 lần và sau đó lựa chọn giá lớn nhất. Nếu giá lớn nhất vừa tìm được nhỏ hơn hoặc bằng \$50, vòng lặp WHILE sẽ lặp lại và gấp đôi giá lần nữa. Vòng lặp này sẽ tiếp tục gấp đôi giá của các cuốn sách tới khi giá lớn nhất lớn hơn \$50, thì thoát khỏi vòng lặp và hiển thị một thông báo.

1. USE pubs
2. GO
3. WHILE (SELECT AVG(price) FROM titles) < \$30
4. BEGIN
5. UPDATE titles
 SET price = price * 2
6. SELECT MAX(price) FROM titles
7. IF (SELECT MAX(price) FROM titles) > \$50
 BREAK
8. ELSE
 CONTINUE
9. END
- 10.PRINT 'Too much for the market to bear'

1.5 Lệnh rẽ nhánh CASE

Từ khoá CASE cho phép chúng ta trả về giá trị dựa trên giá trị của biểu thức điều kiện là đúng. Nó có thể sử dụng bất kỳ ở nhánh nào khi biểu thức điều kiện thoả mãn.

Cú pháp:

CASE expression

WHEN expression1 THEN expression1

[[WHEN expression2 THEN expression2] [...]]

[ELSE expression]

END

Trong đó:

Biểu thức: là một hằng số, tên cột, một hàm, truy vấn con, hoặc bất kỳ biểu thức nào có thể so sánh với biểu thức khác.

WHEN..THEN: Mệnh đề WHEN giữ giá trị điều kiện hoặc biểu thức cần so sánh. Nếu biểu thức so sánh trả về giá trị ĐÚNG thì mệnh đề THEN được thực hiện. Mệnh đề ELSE chỉ được thực hiện nếu tất cả giá trị trong mệnh đề WHEN trả về SAI.

Lệnh lựa chọn cơ bản:

SELECT au_fname, au_lname,

CASE state

WHEN 'OR' THEN 'Oregon'

END AS StateName

FROM authors

Cập nhật nhiều cột

Chúng ta có thể sử dụng câu lệnh CASE để cập nhật nhiều cột trong bảng, thậm chí sử dụng điều kiện cập nhật riêng cho mỗi cột. Ví dụ này cập nhật vào bảng publishers để thiết lập cột state là “_” cho những công ty không ở USA, và thay đổi thành phố cho một nhà xuất bản cụ thể, tất cả dữ liệu trong bảng được cập nhật.

UPDATE publishers

SET state =

CASE

```

    WHEN country <> "USA"
    THEN "--"
    ELSE state
END,
city =
CASE
    WHEN pub_id = "9999"
    THEN "LYON"
    ELSE city
END
WHERE country <> "USA" OR
pub_id = "9999"

```

Định dạng giống nhau sẽ cập nhật 3 hoặc nhiều hàng cùng một lúc với các điều kiện cập nhật khác nhau.

2. TẠO THỦ TỤC LƯU TRỮ (STORED PROCEDURE)

2.1 Khái niệm:

Thủ tục lưu trữ là một đối tượng trong CSDL, bao gồm nhiều câu lệnh T-SQL được tập hợp lại với nhau thành một nhóm, và tất cả các lệnh này sẽ được thực thi khi thủ tục lưu trữ được thực thi.

Cung cấp cách thức lưu trữ, truy xuất và cập nhật dữ liệu vào mỗi bảng thông qua qui trình xác định. Trong qui trình đó, thủ tục có thể có chứa tham số và tùy theo mỗi giá trị trong tham số sẽ cho ta kết quả tương ứng.

Với thủ tục lưu trữ, một phần nào đó khả năng của ngôn ngữ lập trình được đưa vào trong ngôn ngữ SQL.

Thủ tục lưu trữ có thể có các thành phần sau:

- + Các cấu trúc điều khiển (IF, CASE, WHILE, FOR) có thể được sử dụng trong thủ tục.

- + Bên trong thủ tục lưu trữ có thể sử dụng các biến như trong ngôn ngữ lập trình nhằm lưu giữ các giá trị tính toán được, các giá trị được truy xuất từ cơ sở dữ liệu.

+ Một tập các câu lệnh SQL được kết hợp lại với nhau thành một khối lệnh bên trong một thủ tục. Một thủ tục có thể nhận các tham số truyền vào cũng như có thể trả về các giá trị thông qua các tham số (như trong các ngôn ngữ lập trình). Khi một thủ tục lưu trữ đã được định nghĩa, nó có thể được gọi thông qua tên thủ tục, nhận các tham số truyền vào, thực thi các câu lệnh SQL bên trong thủ tục và có thể trả về các giá trị sau khi thực hiện xong.

Lợi ích của việc sử dụng thủ tục lưu trữ:

+ SQL Server chỉ biên dịch các thủ tục lưu trữ một lần và sử dụng lại kết quả biên dịch này trong các lần tiếp theo trừ khi người dùng có những thiết lập khác. Việc sử dụng lại kết quả biên dịch không làm ảnh hưởng đến hiệu suất hệ thống khi thủ tục lưu trữ được gọi liên tục nhiều lần.

+ Thủ tục lưu trữ được phân tích, tối ưu khi tạo ra nên việc thực thi chúng nhanh hơn nhiều so với việc phải thực hiện một tập rời rạc các câu lệnh SQL tương đương theo cách thông thường.

+ Thủ tục lưu trữ cho phép chúng ta thực hiện cùng một yêu cầu bằng một câu lệnh đơn giản thay vì phải sử dụng nhiều dòng lệnh SQL. Điều này sẽ làm giảm thiểu sự lưu thông trên mạng.

+ Thay vì cấp phát quyền trực tiếp cho người sử dụng trên các câu lệnh SQL và trên các đối tượng cơ sở dữ liệu, ta có thể cấp phát quyền cho người sử dụng thông qua các thủ tục lưu trữ, nhờ đó tăng khả năng bảo mật đối với hệ thống.

Các thủ tục lưu trữ trả về kết quả theo 4 cách:

- + Sử dụng các tham số output.
- + Sử dụng các lệnh trả về giá trị, các lệnh này luôn trả về giá trị số nguyên.
- + Tập các giá trị trả về của mỗi câu lệnh SELECT có trong thủ tục lưu trữ hoặc của quá trình gọi một thủ tục lưu trữ khác trong một thủ tục lưu trữ.
- + Một biến con trỏ toàn cục có thể tham chiếu từ bên ngoài thủ tục.

2.2 Tạo thủ tục lưu trữ

Thủ tục lưu trữ được tạo thông qua câu lệnh CREATE PROCEDURE:

Cú pháp:

CREATE PROCEDURE <tên_thủ_tục>

[(danh_sách_tham_số)]

[WITH RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION]

AS

Các_câu_lệnh_của_thủ_tục

Trong đó:

- + WITH RECOMPILE: yêu cầu SQL Server biên dịch lại thủ tục lưu trữ mỗi khi được gọi.
- + WITH ENCRYPTION: yêu cầu SQL Server mã hóa thủ tục lưu trữ.
- + *Các_câu_lệnh_của_thủ_tục*: Các lệnh T-SQL. Nếu nhiều hơn một câu lệnh thì các lệnh này phải nằm trong cặp BEGIN...END.

Ví dụ: Giả sử cần thực hiện các công việc theo thứ tự như sau:

Nhập một đơn đặt hàng mới của khách hàng có mã khách hàng là 3. Biết rằng đơn hàng gồm thông tin Mã khách hàng, Ngày đặt hàng.

Nhập các chi tiết đơn đặt hàng cho đơn đặt hàng trên. Biết rằng Chi tiết đơn hàng gồm thông tin Mã đơn hàng, Mã hàng, Số lượng mỗi mã hàng.

Để thực hiện các công việc trên chúng ta cần các câu lệnh như sau: Trước tiên nhập đơn đặt hàng cho khách hàng có mã khách hàng là 3

insert into DONHANG

values(3, '7/22/2008')

Tiếp theo thêm các chi tiết đơn đặt hàng cho đơn hàng này. Giả sử rằng đơn đặt hàng có mã là 4 và khách hàng đặt một mặt hàng có mã là 1 với số lượng 10.

insert into CHITIETDONHANG

values(4, 1, 10)

+ Hạn chế: Cách viết như trên có hạn chế là trong quá trình làm việc sẽ có rất nhiều đơn đặt hàng mới, do đó người dùng sẽ phải viết đi viết lại những câu lệnh tương tự nhau cho các khách hàng khác nhau với dữ liệu khác nhau.

+ Cách giải quyết vấn đề này: là dùng thủ tục lưu trữ và dùng tham số để nhận các thông tin thay đổi.

```
create procedure spDONHANGCHITIETinsert
@maKhachHang int,
@ngayDatHang datetime,
@maDonHang int,
@maHang int,
@soLuong decimal
as
begin
    insert into DONHANG
    values (@maKhachHang, @ngayDatHang)
    insert into CHITIETDONHANG
    values(@maDonHang, @maHang, @soLuong)
end
```

Thực hiện thủ tục lưu trữ này như sau:

```
spDONHANGCHITIETinsert '3', '22/7/2008', '4', '1', '10'
```

2.3 Lời gọi thủ tục lưu trữ

Thủ tục lưu trữ được gọi theo cấu trúc:

<Tên_thủ_tục_lưu_trữ> [danh_sách_tham_số]

Cần lưu ý:

+ Danh sách tham số truyền vào trong lời gọi phải theo đúng thứ tự khai báo các tham số trong thủ tục lưu trữ.

+ Nếu thủ tục được gọi từ một thủ tục khác, thực hiện bên trong một trigger hay phối hợp với câu lệnh SELECT, sử dụng cấu trúc như sau:

exec <Tên_thủ_tục_lưu_trữ> [danh_sách_tham_số]

Ví dụ: *exec spDONHANGCHITIETinsert '3', '22/7/2008', '4', '1', '10'*

2.4 Định nghĩa biến trong thủ tục lưu trữ

Trong thủ tục lưu trữ có thể có các biến nhằm lưu các kết quả tính toán hay truy xuất từ CSDL. Các biến trong thủ tục được khai báo bằng từ khóa DECLARE theo cấu trúc như sau:

DECLARE <@tên_biến> <kiểu_dữ_liệu>

Ví dụ:

create procedure spKhachTuoiLonNhat

as

begin

declare @tuoiLonNhat int

select @tuoiLonNhat = max(year(getdate())-year(NgaySinh))

from KHACHHANG

select TenKhachHang, NgaySinh

from KHACHHANG

where year(getdate())-year(NgaySinh)=@tuoiLonNhat

end

Một biến là một đối tượng có thể chứa dữ liệu. Chúng ta có thể truyền dữ liệu tới các câu lệnh SQL bằng cách sử dụng các biến cục bộ. Trong T-SQL, chúng ta có thể tạo lập các biến tạm sử dụng trong khi thực hiện một khối các lệnh SQL. Khi biến được khai báo, câu lệnh T-SQL đầu tiên trong khối có thể thiết lập giá trị cho biến. Câu lệnh tiếp theo trong khối có thể nhận giá trị từ biến và trả về kết quả.

2.4.1 Các biến cục bộ và các biến toàn cục

SQL Server hỗ trợ 2 kiểu của biến trong T-SQL: *Biến toàn cục* do hệ thống định nghĩa và duy trì, *Biến cục bộ* do chúng ta tạo lập để chứa kết quả tức thời. Trong phần này chúng ta sẽ học về cách sử dụng biến toàn cục.

2.4.2 Sử dụng biến toàn cục

Tất cả biến toàn cục trong SQL Server có 2 ký hiệu @ làm tiền tố ở đầu. Chúng ta có thể nhận giá trị của bất kỳ biến nào từ một truy vấn SELECT đơn giản

Danh sách các biến toàn cục:

Danh sách các biến	Mục đích
@@CONNECTIONS	Số lượng các kết nối tới server kể từ lần khởi động cuối cùng.
@@CPU_BUSY	Số lượng một phần nghìn giây của hệ thống đang xử lý kể từ khi SQL Server được khởi động.
@@CURSOR_ROWS	Số hàng trong thời gian truy cập gần nhất.
@@DATEFIRST	The current value of SET DATEFIRST parameter which controls the first day that's considered to be the first day of the week.
@@ERROR	Mã lỗi của phát biểu T-SQL vừa thực hiện.
@@FETCH_STATUS	0 if the last fetch status was successful. -1 if there was an error
@@IDENTITY	Last inserted identity value
@@LANGUAGE	Name of the language currently in use.
@@MAX_CONNECTIONS	Số lượng các kết nối hiện tại tối đa có thể tạo được.
@@ROWCOUNT	Số lượng các hàng bị tác động bởi câu lệnh SQL gần nhất.
@@SERVERNAME	Tên của máy chủ cục bộ
@@SERVICENAME	Tên của dịch vụ SQL trên máy tính
@@TIMETICKS	Số lượng phần triệu giây trên mỗi giây tại máy tính hiện thời
@@TRANSCOUNT	Số lượng các giao dịch mở trên kết nối hiện tại
@@VERSION	Thông tin về phiên bản của SQL Server

2.4.3 Khai báo và khởi tạo giá trị cho các biến cục bộ

Giống như các ngôn ngữ lập trình khác, T-SQL cho phép chúng ta tạo lập và sử dụng các biến cục bộ để chứa tạm thời trong khi chúng ta chạy một khối các lệnh SQL.

Tên của các biến cục bộ phải bắt đầu với ký hiệu '@'. Để tạo lập một biến cục bộ, chúng ta sử dụng câu lệnh DECLARE, với cú pháp như sau:

Cú pháp:

```
DECLARE {  
    @local_variable name DATA TYPE  
}
```

Ví dụ, chúng ta khai báo một biến để chứa 5 ký tự.

```
DECLARE @CUST VARCHAR(5)
```

Để gán một giá trị cho một biến, ta có thể sử dụng lệnh SET hoặc lệnh SELECT

Cú pháp:

```
SET @local_variable name = value OR  
SELECT @local_variable name = value
```

Ví dụ, chúng ta gán giá trị cho biến đã khai báo trước đó.

```
SET @CUST='FRANK'  
  
SELECT      CUSTOMERID,COMPANYNAME      FROM  
CUSTOMERS  
  
WHERE CUSTOMERID = @CUST
```

2.4.4 Tham số với giá trị mặc định

Các tham số được khai báo trong thủ tục có thể nhận các giá trị mặc định. Giá trị mặc định sẽ được gán cho tham số trong trường hợp không truyền đối số cho tham số khi có lời gọi đến thủ tục.

Tham số với giá trị mặc định được khai báo theo cú pháp như sau:

<@tên_tham_số> <kiểu_dữ_liệu> = <giá_trị_mặc_định>

Ví dụ:

```
create procedure spTestDefault
```

```
@maKhachHang int = 3
```

```
as
```

```
begin
```

```
    select *
```

```
    from KHACHHANG
```

```
    where MaKhachHang = @maKhachHang
```

```
end
```

Thực thi thủ tục lưu trữ theo giá trị mặc định của tham số.

```
spTestDefault
```

MaKhachHang	TenKhachHang	NgaySinh	GioiTinh	DiaChi
3	Lê Thị Hoa	1972-01-01 00:00:00.000	1	46 Ngô Quyền

Thực thi thủ tục và truyền giá trị cho tham số:

```
spTestDefault 4
```

MaKhachHang	TenKhachHang	NgaySinh	GioiTinh	DiaChi
-------------	--------------	----------	----------	--------

4	Nguyễn Văn Hải	1979-02-01 00:00:00.000	1	46 Lê Lợi
---	-------------------	----------------------------	---	--------------

2.5 Giá trị trả về trong thủ tục lưu trữ

Trong các ví dụ trước, nếu đổi số truyền cho thủ tục khi có lời gọi đến thủ tục là biến thì những thay đổi giá trị của biến trong thủ tục sẽ không được giữ lại khi kết thúc quá trình thực hiện thủ tục.

Ví dụ: Có thủ tục lưu trữ như sau:

```
create procedure spTestOutput
```

```
@a int, @b int, @c int
```

```
as
```

```
select @c = @a + @b
```

Thực thi thủ tục:

```
Declare @tong int
```

```
set @tong = 0
```

```
spTestOutput 100, 200, @tong
```

```
select @tong
```

Kết quả là 0. (giá trị không thay đổi)

+ Sử dụng tham số OUTPUT

Trong trường hợp cần phải giữ lại giá trị của đối số sau khi kết thúc thủ tục, ta phải khai báo tham số của thủ tục theo cú pháp như sau:

```
<@tên_tham_số> <kiểu_dữ_liệu> OUTPUT
```

Ví dụ trên được viết lại như sau:

```
create procedure spTestOutput
```

```
@a int,
```

```
@b int,  
@c int output  
as  
select @c = @a + @b
```

Thực thi thủ tục:

```
Declare @tong int  
set @tong = 0  
spTestOutput 100, 100, @tong output  
select @tong
```

Kết quả là 200. (giá trị đã thay đổi)

Lưu ý: cách gọi thủ tục có tham số output: *spTestOutput 100, 100, @tong output*

+ Sử dụng lệnh **RETURN**

Tương tự như việc sử dụng tham số OUTPUT, câu lệnh RETURN trả về giá trị cho đối tượng thực thi stored procedure thông qua tham số hoặc biểu thức.

```
RETURN <@tên_tham_số>
```

Ví dụ:

```
create procedure spTestReturn  
as  
begin  
    declare @out int  
    select @out = count(*)  
    from KHACHHANG  
    return @out  
end
```

Thực thi thủ tục lưu trữ

declare @a int

exec @a = spTestReturn

select @a

Lưu ý: cách gọi thủ tục có return: ***exec @a = spTestReturn***

2.6 Sửa và xóa thủ tục lưu trữ

2.6.1 Sửa đổi thủ tục lưu trữ

Khi một thủ tục đã được tạo ra, ta có thể tiến hành định nghĩa lại thủ tục đó bằng câu lệnh ALTER PROCEDURE có cú pháp như sau:

ALTER PROCEDURE <tên_thủ_tục>

[(danh_sách_tham_số)]

[WITH RECOMPILE|ENCRYPTION|RECOMPILE,ENCRYPTION]

AS

Các_câu_lệnh_của_thủ_tục

Câu lệnh này sử dụng tương tự như câu lệnh CREATE PROCEDURE. Việc sửa đổi lại một thủ tục đã có (tức là không đổi tên thủ tục) không làm thay đổi đến các quyền đã cấp phát trên thủ tục cũng như không tác động đến các thủ tục khác hay trigger phụ thuộc vào thủ tục này.

2.6.2 Xóa thủ tục lưu trữ

Để xóa một thủ tục đã có, ta sử dụng câu lệnh DROP PROCEDURE với cú pháp như sau:

DROP PROCEDURE <tên_thủ_tục>

Ví dụ: drop procedure spTestDefault

Khi xóa một thủ tục, tất cả các quyền đã cấp cho người sử dụng trên thủ tục đó cũng đồng thời bị xóa bỏ. Do đó, nếu tạo lại thủ tục, ta phải tiến hành cấp phát lại các quyền trên thủ tục đó.

3. CÂU HỎI ÔN TẬP THỦ TỤC

- 1/- Hãy nêu cú pháp tạo thủ tục lưu trữ?
- 2/- Hãy nêu cách đặt tham số trong thủ tục lưu trữ?
- 3/- Hãy nêu cú pháp lời gọi thủ tục lưu trữ?
- 4/- Hãy nêu cách đặt biến trong thủ tục lưu trữ?
- 5/- Hãy nêu các cách lấy giá trị trả về của thủ tục lưu trữ?
- 6/- Hãy nêu cú pháp sửa thủ tục lưu trữ?
- 7/- Hãy nêu cú pháp xóa thủ tục lưu trữ?
- 8/- Hãy áp dụng làm bài luyện tập phần thủ tục lưu trữ. (xem phần luyện tập)?

BÀI 7: HÀM DO NGƯỜI DÙNG ĐỊNH NGHĨA (User Defined Function-UDF)

Mã bài: MD15_07

Giới thiệu:

Bài học giúp sinh viên tạo và sử dụng được hàm trong SQL Server

Mục tiêu:

- Trình bày được cú pháp và cách tạo các hàm thông qua công cụ Management Studio;
- Vận dụng được cách truyền tham số;
- Tạo và sử dụng được hàm;
- Rèn luyện thái độ nghiêm túc, khoa học, sáng tạo.

Hàm do người dùng định nghĩa được chia làm 3 loại: (1) scalar (hàm vô hướng), (2) inline table-valued (hàm nội tuyến, giá trị trả về dạng bảng), (3) multi-statement table-valued (hàm bao gồm nhiều câu lệnh SQL bên trong, trả về giá trị dạng bảng)

Scalar UDF: được sử dụng để trả về duy nhất một giá trị dựa trên một hoặc các tham số truyền vào. Ví dụ: ta có thể tạo ra một UDF vô hướng nhận MaKhachHang là tham số và trả về TenKhachHang.

Inline table-valued: trả về một bảng dựa trên một câu lệnh SQL duy nhất định nghĩa các dòng và các cột trả về.

Multi-statement table-value: cũng trả về kết quả là một tập hợp nhưng có thể dựa trên nhiều câu lệnh SQL.

1. HÀM VÔ HƯỚNG - SCALAR UDF

1.1 Sử dụng hàm

1.1.1 Sử dụng các hàm hệ thống

T-SQL cung cấp một số lượng lớn các hàm. Các hàm này có thể hữu ích khi chúng ta tính toán hoặc thao tác dữ liệu. Các hàm T-SQL có thể chia làm 3 loại:

- Nhóm hàm Rowset - Những hàm này có thể sử dụng định danh tên bảng trong SQL

- Nhóm hàm Aggregate (Hàm tập hợp) -Những hàm này được sử dụng để tính toán các phép tính số học (ví dụ một phép tính tổng hoặc một sự lệch chuẩn.
- Tập hợp hàm Scalar (Những hàm số này thao tác trên một giá trị đơn, và trả về một giá trị đơn. Chúng ta có thể sử dụng các hàm số này trong biểu thức

1.1.1.1 Các hàm phổ biến được SQL server hỗ trợ và các dạng phân loại các hàm này

Trong SQL, các hàm thao tác trên dữ liệu hoặc một nhóm dữ liệu, để trả về một giá trị. Chúng có thể sử dụng trong một danh sách SELECT, hoặc bất kỳ đâu khi biểu thức thoả mãn. SQL phân chia thành 7 loại hàm số:

- Hàm Aggregate trả về tổng các giá trị
- Hàm Conversion chuyển đổi một kiểu dữ liệu thành một kiểu khác
- Hàm Date làm việc trên các dữ liệu dạng ngày giờ.
- Hàm Mathematical thực hiện các phép toán số học.
- Hàm String thực hiện các phép toán trên dữ liệu dạng ký tự, chuỗi, nhị phân.

Hàm System trả về tập hợp các thông tin đặc biệt từ CSDL tham chiếu từ các giá trị SQL Server, các đối tượng, hoặc các thông số cài đặt.

1.1.1.1.1 Hàm Aggregate

SQL cung cấp các hàm **aggregate** để hỗ trợ việc tập hợp một lượng lớn dữ liệu. Chúng ta sẽ học về các hàm cộng và tính giá trị trung bình, đếm các bản ghi thoả mãn điều kiện chỉ ra và tìm số lớn nhất hoặc nhỏ nhất trong bảng.

Tên Hàm	Giá trị trả về	Ví dụ
Sum(tên cột)	Trả về tổng của tập hợp giá trị trong cột.	SELECT SUM(Quantity) AS Total FROM WidgetOrders Trả về giá trị tổng của trường 'Quantity' từ bảng WidgetOrders
Avg(tên cột)	Trả về giá trị trung bình của tập hợp giá trị trong cột.	SELECT AVG(UnitPrice * Quantity) As AveragePrice FROM WidgetOrders WHERE Continent = "North

Tên Hàm	Giá trị trả về	Ví dụ
		America” Trả về giá trung bình
COUNT(*)	Trả về số bản ghi trong bảng thoả mãn điều kiện đã chỉ ra.	SELECT COUNT(*) AS 'Number of Large Orders' FROM WidgetOrders WHERE Quantity > 100 Trả về số lượng mặt hàng yêu cầu có quantity > 100
Max(tên cột)	Trả về giá trị lớn nhất trong tập hợp giá trị của cột.	SELECT MAX(Quantity * UnitPrice)As 'Largest Order' FROM WidgetOrders Trả về giá trị lớn nhất trong dãy số đã tính toán.
Min(tên cột)	Trả về giá trị nhỏ nhất trong tập hợp giá trị của cột.	SELECT MIN(Quantity * UnitPrice)As 'Smallet Order' FROM WidgetOrders Trả về giá trị nhỏ nhất trong dãy số đã tính toán.

1.1.1.1.2 Hàm Conversion

Hàm conversion function sử dụng để chuyển kiểu giá trị thành kiểu khác. Thêm vào đó, chúng ta có thể sử dụng nó để định dạng nhiều dạng khác nhau của kiểu ngày tháng. SQL Server cung cấp một hàm chuyển đổi, đó là CONVERT().

Cú pháp:

CONVERT(datatype[(length)], expression [,style])

Trong đó:

datatype: Xác định kiểu dữ liệu hệ thống. Chúng ta không thể chỉ ra kiểu người dùng định nghĩa ở đây.

Length: Chỉ ra độ dài nếu kiểu dữ liệu là ký tự, nhị phân hoặc biến. Độ dài tối đa cho phép là 255.

Expression: Có thể là tên cột, một hằng số, một hàm số, một biến, một truy vấn con, hoặc bất kỳ sự kết hợp của các tên cột, hằng số, và các hàm có toán tử hoặc một truy vấn con được kết nối.

Style: Chỉ ra ngày đại diện mà muốn chuyển đổi dữ liệu ngày tháng thành ký tự.

1.1.1.1.3 Hàm Date

Hàm Date được sử dụng để thao tác với các giá trị ngày tháng. Chúng dùng để tính toán trên dữ liệu dạng ngày tháng. Chúng ta không thể thực hiện các hàm toán học trực tiếp trên các giá trị ngày tháng. Ví dụ, nếu chúng ta thực hiện câu lệnh giá trị ngày +1, SQL Server sẽ không hiểu chúng ta muốn cộng 1 ngày, 1 tháng hay 1 năm. Hàm Date giúp chúng ta trích lọc ngày, tháng và năm của chuỗi giá trị ngày. Vì vậy chúng ta có thể thao tác với chúng một cách riêng biệt.

Hầu hết các hàm ngày tháng liệt kê trong phần này đều sử dụng một tham số gọi là 'datepart'. Nó tham chiếu tới một phần dữ liệu ngày tháng chúng ta muốn trích lọc và sử dụng. Chúng ta có thể sử dụng các datepart và giá trị này trong hàm số

DatePart	Viết tắt	Giá trị	Datepart	Viết tắt	Giá trị
Hour	Hh	0-23	Week	wk	1-53
Minute	Mi	0-59	Weekday	dw	1-7
Second	Ss	0-59	Month	mm	1-12
Millisecond	Ms	0-999	Quarter	qq	1-4
Day of year	Dy	1-366	Year	yy	1753-9999
Day	Dd	1-31			

Các hàm ngày tháng trong SQL Server.

Hàm	Giá trị trả về	Ví dụ
GETDATE()	Ngày hiện tại	SELECT GETDATE()
DATEADD(datepart, number, date)	Giá trị ngày (date) cộng với một số (number).	SELECT DATEADD(mm,4,'01/01/99') Trả về ngày 05/01/99 trong định dạng ngày hiện tại.
DATEDIFF(datepart, date1,date2)	Giá trị chênh lệch giữa hai ngày.	SELECT DATEDIFF (mm, '01/01/99','05/01/99') Trả về 4
DATENAME(datepart, date)	Trả về giá trị ngày, với định dạng ngày tương ứng trong datepart, date dưới dạng chuỗi ký tự.	SELECT DATENAME (dw, '01/01/2000') Trả về Saturday
DATEPART(datepart, date)	Một số nguyên chứa giá trị ngày tương ứng với định dạng trong (datepart) của giá trị ngày (date).	SELECT DATEPART(day,'01/15/2000') Trả về 15

1.1.1.2 Các hàm toán học

Các hàm toán học thực hiện các phép toán đại số trên các giá trị số. Bảng 5.6 liệt kê các hàm toán học phổ biến có sẵn trong SQL Server.

Hàm	Giá trị trả về	Ví dụ
ABS(num_expr)	Giá trị tuyệt đối của num_expr	SELECT ABS(-43) Trả về 43
CEILING(num_expr)	Số nguyên nhỏ nhất lớn hơn hoặc bằng giá trị số	SELECT CEILING(43.5)

	trong biểu thức (num_expr).	Trả về 44
FLOOR(num_expr)	Số nguyên lớn nhất nhỏ hơn hoặc bằng giá trị số chỉ ra trong biểu thức.	SELECT FLOOR(43.5) Trả về 43
POWER(num_expr,y)	Trả về giá trị lũy thừa y của số (num_expr).	SELECT POWER(5,2) Trả về 25
ROUND(num_expr,length)	Trả về giá trị làm tròn của số trong biểu thức (num_expr) được làm tròn đến phần thập phân có độ dài chỉ ra (length)	SELECT ROUND (43.543,1) Trả về 43.5
Sign(num_expr)	+1 cho số dương, -1 cho số âm , và 0 cho số 0	SELECT SIGN(-43) Trả về -1
Sqrt(float_expr)	Trả về giá trị căn bậc hai của biểu thức chỉ ra (float expression)	SELECT SQRT(9) Trả về 3

1.1.1.3 Các hàm hệ thống

SQL Server cung cấp các hàm hệ thống để trả về thông tin cơ bản hoặc các cấu hình cài đặt. Các hàm hệ thống thông dụng sau:

Hàm	Giá trị trả về
DB_ID(['database_name'])	Số định danh CSDL
DB_NAME([database_id])	Tên CSDL
HOST_ID()	Số định danh của máy làm việc
HOST_NAME()	Tên máy làm việc

ISNULL(expr,value)	Giá trị của biểu thức(expr) rỗng sẽ được thay thế với giá trị này (value)
OBJECT_ID('obj_name')	Số định danh của đối tượng CSDL
OBJECT_NAME(object_id)	Tên của đối tượng CSDL
SUSER_SID(['login_name'])	Số định danh bảo mật (SID) đối với tên đăng nhập của người sử dụng (user's login name)
SUSER_ID(['login_name'])	Số định danh đăng nhập của người dùng. Chức năng này giống như SUSER_SID(), và nó tiếp tục tương thích ở phía sau.
SUSER_SNAME([server_user_id])	Trả về tên định danh đăng nhập từ số định danh bảo mật của người sử dụng (SID).
SUSER_NAME([server_user_id])	Tên định danh đăng nhập của người sử dụng.
USER_ID(['user_name'])	Số định danh CSDL của người sử dụng.
USER_NAME([user_id])	Tên CSDL của người sử dụng.

1.2 Hàm vô hướng

Scalar UDF được tạo ra bằng câu lệnh CREATE FUNCTION có cấu trúc như sau:

CREATE FUNCTION < tên_hàm >

[(danh_sách_tham_số)]

RETURNS (kiểu_trả_về_của_hàm)

AS

BEGIN

các_câu_lệnh_của_hàm

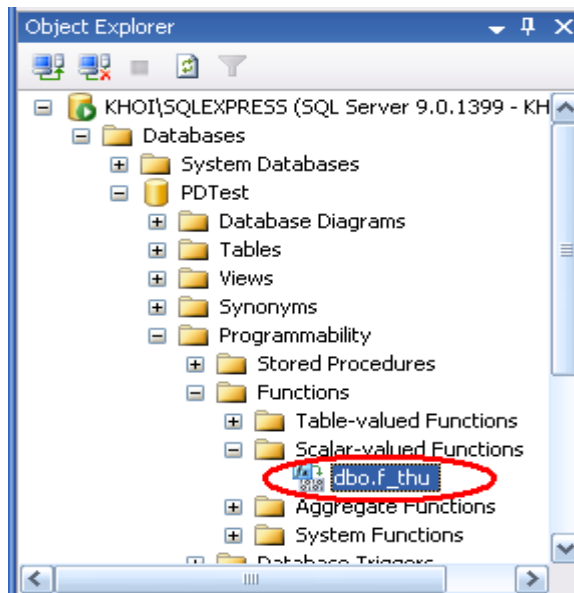
END

Ví dụ:

Câu lệnh dưới đây định nghĩa hàm tính ngày trong tuần (thứ trong tuần) của một giá trị kiểu ngày

```
create function f_thu  
( @ngay datetime)  
returns nvarchar(10)  
as  
begin  
  
    declare @st nvarchar(10)  
  
    select @st=case datepart(dw,@ngay)  
  
        when 1 then N'chủ nhật'  
  
        when 2 then N'thứ hai'  
  
        when 3 then N 'thứ ba'  
  
        when 4 then N 'thứ tư'  
  
        when 5 then N 'thứ năm'  
  
        when 6 then N 'thứ sáu'  
  
        else N 'thứ bảy'  
  
    end  
  
    return (@st) /* trị trả về của hàm */  
  
end
```

Sau khi chạy thành công, hàm trở thành một đối tượng trong CSDL và có thể được truy xuất như các hàm được xây dựng sẵn trong SQL Server 2005 Express Edition.



Ví dụ:

```
select TenKhachHang, dbo.f_thu(NgaySinh)
from KHACHHANG
```

TenKhachHang	(No column name)
Nguyễn Thanh Hải	thứ năm
Trần Thiên Phước	thứ hai
Lê Thị Hoa	thứ năm

2. HÀM NỘI TUYẾN - INLINE UDF

Hàm nội tuyến được định nghĩa bằng lệnh CREATE FUNCTION.

CREATE FUNCTION <tên_hàm>

([danh_sách_tham_số])

RETURNS TABLE

AS

RETURN (câu_lệnh_select)

Cú pháp của hàm nội tuyến phải tuân theo các qui tắc sau:

+ Kiểu trả về của hàm phải được chỉ định bởi mệnh đề RETURNS TABLE.

+ Trong phần thân của hàm chỉ có duy nhất một câu lệnh RETURN xác định giá trị trả về của hàm thông qua duy nhất một câu lệnh SELECT. Ngoài ra, không sử dụng bất kỳ câu lệnh nào khác trong phần thân của hàm.

Ví dụ: Ví dụ dưới đây lấy ra các khách hàng tùy thuộc vào giá trị mã khách hàng truyền vào cho tham số.

```
create function fnKHACHHANG (@maKhachHang int)
returns TABLE
as
return (select *
        from KHACHHANG
        where MaKhachHang = @maKhachHang)
```

Việc gọi các hàm nội tuyến cũng tương tự như việc gọi các hàm vô hướng.

Ví dụ:

```
select KH.TenKhachHang, DH.NgayDonHang
from DONHANG DH inner join dbo.fnKHACHHANG(3) as KH on
DH.MaKhachHang = KH.MaKhachHang
```

TenKhachHang	NgayDonHang
Lê Thị Hoa	2007-12-06 00:00:00.000

3. HÀM BAO GỒM NHIỀU CÂU LỆNH BÊN TRONG – MULTI STATEMENT UDF

Hàm này cũng được định nghĩa bằng lệnh CREATE FUNCTION

CREATE FUNCTION <tên_hàm>

([danh_sách_tham_số])

RETURNS @biến_bảng **TABLE**

(định_nghĩa_bảng_bằng_cách_tên_cột_kiểu)

AS

BEGIN

các_câu_lệnh_trong_thân_hàm

RETURN

END

Lưu ý: sau từ khóa **RETURNS** là một biến bảng được định nghĩa. Và sau từ khóa **RETURN** ở cuối hàm không có tham số nào đi kèm.

Ví dụ:

create function fnKHACHHANG

(@maKhachHang int)

RETURNS @KHACHHANG table

(

MaKhachHang int,

TenKhachHang nvarchar(50),

NgayDonHang datetime

)

AS

begin

if @maKhachHang = 0

insert into @KHACHHANG

```

select KH.MaKhachHang, KH.TenKhachHang, DH.NgayDonHang
from KHACHHANG KH inner join DONHANG DH on
DH.MaKhachHang = KH.MaKhachHang
else
insert into @KHACHHANG
select KH.MaKhachHang, KH.TenKhachHang, DH.NgayDonHang
from KHACHHANG KH inner join DONHANG DH on
DH.MaKhachHang = KH.MaKhachHang
where KH.MaKhachHang = @MaKhachHang
return
end

```

Việc gọi hàm multi statement UDF cũng tương tự các loại hàm khác:

```
select * from fnKHACHHANG(0)
```

MaKhachHang	TenKhachHang	NgayDonHang
3	Lê Thị Hoa	2007-12-06 00:00:00.000
3	Lê Thị Hoa	2007-12-07 00:00:00.000
4	Nguyễn Thanh Hải	2007-11-07 00:00:00.000
5	Trần Thiên Phước	2007-11-06 00:00:00.000

```
select * from fnKHACHHANG(3)
```

MaKhachHang	TenKhachHang	NgayDonHang
3	Lê Thị Hoa	2007-12-06 00:00:00.000

3	Lê Thị Hoa	2007-12-07 00:00:00.000
---	------------	----------------------------

4. SỬA VÀ XÓA HÀM

4.1 Sửa đổi hàm

Dùng lệnh **ALTER FUNCTION** để thay đổi định nghĩa hàm. Cấu trúc của câu lệnh ALTER FUNCTION tương tự như CREATE FUNCTION

Ví dụ:

```
alter function fnKHACHHANG (@maKhachHang int)
```

```
returns table
```

```
AS
```

```
return (select *
```

```
from KHACHHANG
```

```
where MaKhachHang = @maKhachHang)
```

4.2 Xóa hàm

Dùng lệnh DROP FUNCTION để xóa hàm. Cấu trúc DROP FUNCTION như sau:

```
DROP FUNCTION < tên_hàm >
```

Ví dụ:

```
drop function f_thu
```

Tương tự như thủ tục lưu trữ, khi hàm bị xóa các quyền cấp cho người dùng trên hàm đó cũng bị xóa. Do đó khi định nghĩa lại hàm này, ta phải cấp lại quyền cho các người dùng.

5. THỰC THI CÁC HÀM DO NGƯỜI DÙNG ĐỊNH NGHĨA

Không như thủ tục, khi đã được người dùng định nghĩa tùy đối tượng trả về là giá trị hay là bảng dữ liệu mà ta sử dụng câu lệnh truy vấn SQL để thực thi hàm.

6. CÂU HỎI ÔN TẬP HÀM

- 1/- Hãy cho biết có bao nhiêu loại hàm do người dùng định nghĩa?
- 2/- Hãy nêu các thành phần trong cú pháp tạo hàm của mỗi loại?
- 3/- Hãy chỉ ra những điểm khác nhau trong cú pháp tạo hàm của mỗi loại?
- 4/- Hãy nêu cú pháp sửa hàm và xóa hàm?
- 5/- Hãy áp dụng làm bài luyện tập phần hàm. (xem phần luyện tập)?

BÀI 8: TRIGGER

Mã bài: MD15_08

Giới thiệu:

Bài học giúp sinh viên phân tích và sử dụng được công cụ để tạo trigger.

Mục tiêu:

- Trình bày được cú pháp và cách tạo và công dụng của trigger;
- Phân biệt sự khác nhau giữa trigger với hàm, thủ tục;
- Phân tích được các tình huống nên sử dụng trigger;
- Sử dụng được công cụ để tạo trigger;
- Rèn luyện thái độ nghiêm túc, khoa học, sáng tạo.

1. GIỚI THIỆU VỀ TRIGGER

Trigger là một dạng đặc biệt của thủ tục lưu trữ, được thực thi một cách tự động khi có sự thay đổi dữ liệu (do tác động của câu lệnh INSERT, UPDATE, DELETE) trên một bảng nào đó.

1.1 Các đặc điểm của trigger

Trigger chỉ thực thi tự động thông qua các sự kiện mà không thực hiện bằng tay. Trigger sử dụng được với khung nhìn.

Khi trigger thực thi theo các sự kiện Insert hoặc Delete thì dữ liệu khi thay đổi sẽ được chuyển sang các bảng **INSERTED** và **DELETED**, là 2 bảng tạm thời chỉ chứa trong bộ nhớ, các bảng này chỉ được sử dụng với các lệnh trong trigger. Các bảng này thường được sử dụng nhằm mục đích khôi phục lại phần dữ liệu đã thay đổi (roll back) hoặc tự động thực thi dữ liệu tương ứng đảm bảo ràng buộc toàn vẹn trong quá trình thay đổi dữ liệu.

Trigger chia thành 2 loại INSTEAD OF và AFTER: INSTEAD OF là loại trigger mà hoạt động của sự kiện gọi trigger sẽ bị bỏ qua và thay vào đó là các lệnh trong trigger được thực hiện trước. AFTER trigger là loại ngầm định, khác với loại INSTEAD OF thì loại AFTER trigger này sẽ thực hiện các lệnh bên trong sau khi đã thực hiện xong sự kiện kích hoạt trigger.

1.2 Các trường hợp sử dụng trigger

Sử dụng Trigger khi các biện pháp bảo đảm toàn vẹn dữ liệu khác không bảo đảm được. Các công cụ này sẽ thực hiện kiểm tra tính toàn vẹn trước khi

đưa dữ liệu vào cơ sở dữ liệu, còn Trigger thực hiện kiểm tra tính toàn vẹn khi công việc đã thực hiện.

Khi CSDL chưa được chuẩn hóa (Normalization) thì có thể xảy ra dữ liệu thừa, chứa ở nhiều vị trí trong CSDL thì yêu cầu đặt ra là dữ liệu cần cập nhật thống nhất trong mọi nơi. Trong trường hợp này ta phải sử dụng Trigger.

Khi xảy ra thay đổi dây chuyền dữ liệu giữa các bảng với nhau (khi dữ liệu bảng này thay đổi thì dữ liệu trong bảng khác cũng được thay đổi theo).

1.3 Khả năng của trigger

Một trigger có thể nhận biết, ngăn chặn và huỷ bỏ được những thao tác làm thay đổi trái phép dữ liệu trong cơ sở dữ liệu.

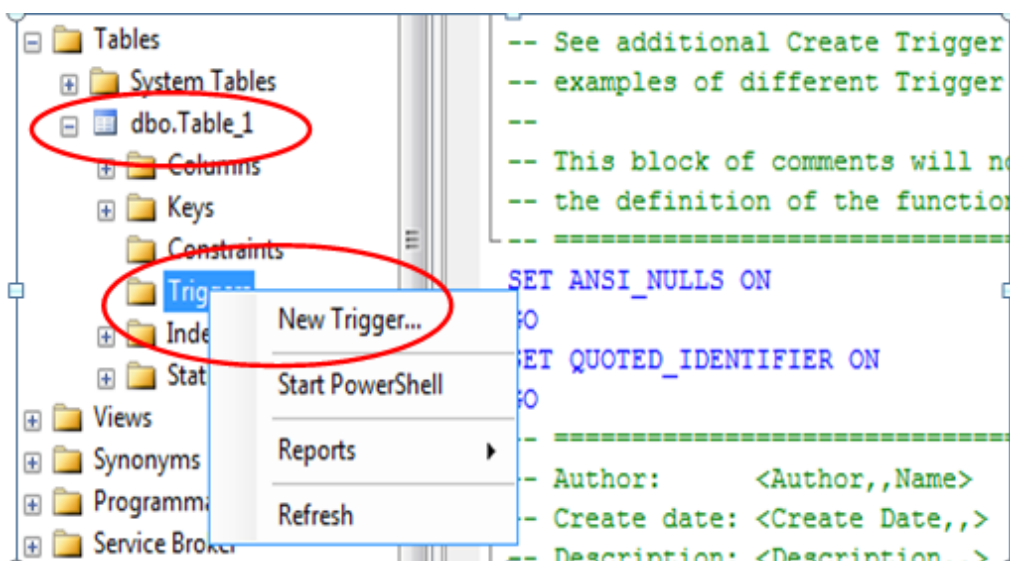
Các thao tác trên dữ liệu (xóa, cập nhật và bổ sung) có thể được trigger phát hiện ra và tự động thực hiện một loạt các thao tác khác trên cơ sở dữ liệu nhằm đảm bảo tính hợp lệ của dữ liệu.

Thông qua trigger, ta có thể tạo và kiểm tra được những mối quan hệ phức tạp hơn giữa các bảng trong cơ sở dữ liệu mà bản thân các ràng buộc không thể thực hiện được.

2. TẠO, THAY THẾ, LOẠI BỎ TRIGGER

2.1 Tạo trigger

Chọn bảng cần tạo trigger, kích phải chuột trên mục trigger và chọn New trigger



Câu lệnh CREATE TRIGGER được sử dụng để định nghĩa trigger và có cấu trúc như sau:

```
CREATE TRIGGER <tên_trigger>  
ON <tên_bảng|tên_view>  
FOR {[INSERT][,][UPDATE][,][DELETE]}  
AS  
[IF UPDATE (tên_cột)  
[AND UPDATE(tên_cột)|OR UPDATE(tên_cột)]  
...]  
các_câu_lệnh_của_trigger
```

Lưu ý: Như đã nói ở trên, chuẩn SQL định nghĩa hai bảng logic INSERTED và DELETED để sử dụng trong các trigger. Cấu trúc của hai bảng này tương tự như cấu trúc của bảng mà trigger tác động. Dữ liệu trong hai bảng này tùy thuộc vào câu lệnh tác động lên bảng làm kích hoạt trigger; Cụ thể trong các trường hợp sau:

+ Khi câu lệnh DELETE được thực thi trên bảng, các dòng dữ liệu bị xóa sẽ được sao chép vào trong bảng DELETED. Bảng INSERTED trong trường hợp này không có dữ liệu.

+ Dữ liệu trong bảng INSERTED sẽ là dòng dữ liệu được bổ sung vào bảng gây nên sự kích hoạt đối với trigger bằng câu lệnh INSERT. Bảng DELETED trong trường hợp này không có dữ liệu.

+ Khi câu lệnh UPDATE được thực thi trên bảng, các dòng dữ liệu cũ chịu sự tác động của câu lệnh sẽ được sao chép vào bảng DELETED, còn trong bảng INSERTED sẽ là các dòng sau khi đã được cập nhật.

Hoạt động	Bảng INSERTED	Bảng DELETED
INSERT	dữ liệu được insert	không có dữ liệu
DELETE	không có dữ liệu	dữ liệu bị xóa

UPDATE	dữ liệu được cập nhật	dữ liệu trước khi cập nhật
--------	-----------------------	----------------------------

Ví dụ 1: Ví dụ dưới đây minh họa việc trigger được kích hoạt khi thêm dữ liệu vào bảng KHACHHANG

--Kiểm tra sự tồn tại của trigger

if exists (select name from sysobjects

where name = 'trKiemTraTenKhachHang' and type = 'TR')

drop trigger trKiemTraTenKhachHang

go

--Tạo trigger

create trigger trKiemTraTenKhachHang

on KHACHHANG

for insert

as

declare @chieuDaiTen int

select @chieuDaiTen = len(inserted.TenKhachHang)

from inserted

if @chieuDaiTen <=1 *--tên hợp lệ ít nhất phải 2 ký tự.*

print N'Tên không hợp lệ'

rollback tran

go

Thêm một khách hàng mới có tên là A

insert into KHACHHANG

values('A', '5/5/1978', 'True', '35 Hung Vuong')

Tên không hợp lệ

Msg 3609, Level 16, State 1, Line 2

The transaction ended in the trigger. The batch has been aborted.

Ví dụ 2: Ví dụ dưới đây minh họa trigger được kích hoạt khi có sự thay đổi mang tính dây chuyền giữa các bảng. Giả sử ta có vắn tắt cơ sở dữ liệu như sau:

HANGHOA(MaHang: nvarchar(25), SoLuongTon: int).

Trong đó: MaHang là not null primary key, SoLuongTon mặc định khi khởi tạo là default 0 check (sl>=0).

NHAP(MaDonHang: nvarchar(12), MaHang: nvarchar(25), SoLuongNhap: int).

Trong đó: MaDonHang là primary key, MaHang là foreign key, SoLuongNhap phải check (SoLuongNhap>0)

XUAT (MaDonHang: nvarchar(12), MaHang: nvarchar(25), SoLuongXuat: int)

Trong đó: MaDonHang là primary key, MaHang là foreign key, SoLuongXuat phải check (SoLuongXuat>0)

Hãy viết các trường hợp làm ảnh hưởng đến số lượng tồn trong HANGHOA?

Phân tích: ta có 4 trường hợp sau đối với việc thêm và xóa:

- + Khi ta thêm một mã hàng trong bảng NHAP thì số lượng tồn trong bảng HANGHOA sẽ bằng SoLuongTon + INSERTED.SoLuongNhap
- + Khi ta xóa một mã hàng trong bảng NHAP thì số lượng tồn trong bảng HANGHOA sẽ bằng SoLuongTon - DELETED.SoLuongNhap (tức là bỏ nhập)
- + Khi ta thêm một mã hàng trong bảng XUAT thì số lượng tồn trong bảng HANGHOA sẽ bằng SoLuongTon - INSERTED.SoLuongXuat
- + Khi ta xóa một mã hàng trong bảng XUAT thì số lượng tồn trong bảng HANGHOA sẽ bằng SoLuongTon + DELETED.SoLuongXuat (tức là bỏ xuất)

Viết lệnh tạo trigger cho 4 trường hợp trên:

+/ Thêm nhập.

Create trigger trNHAPInsert

ON NHAP

FOR Insert

AS

Update HANGHOA

Set HANGHOA.SoLuongTon = HANGHOA.SoLuongTon +
INSERTED.SoLuongNhap

From HANGHOA inner join INSERTED on HANGHOA.MaHang =
INSERTED.MaHang

+/ Xóa nhập.

Create trigger trNHAPDelete

ON NHAP

FOR Delete

AS

Update HANGHOA

Set HANGHOA.SoLuongTon = HANGHOA.SoLuongTon -
DELETED.SoLuongNhap

From HANGHOA inner join DELETED on HANGHOA.MaHang =
DELETED.MaHang

+/ Thêm xuất.

Create trigger trXUATInsert

ON XUAT

FOR Insert

AS

Update HANGHOA

Set HANGHOA.SoLuongTon = HANGHOA.SoLuongTon -
INSERTED.SoLuongXuat

From HANGHOA inner join INSERTED on HANGHOA.MaHang =
INSERTED.MaHang

+/ Xóa xuất.

Create trigger trXUATDelete

ON XUAT

FOR Delete

AS

Update HANGHOA

Set HANGHOA.SoLuongTon = HANGHOA.SoLuongTon +
DELETED.SoLuongXuat

From HANGHOA inner join DELETED on HANGHOA.MaHang =
DELETED.MaHang

2.2 Sửa trigger

Sau khi sửa nội dung trong trigger đã tạo (không đổi tên trigger), ta phải thay từ CREATE bởi từ ALTER trước trigger và biên dịch lại trigger

ALTER TRIGGER <Tên trigger>

Nếu muốn đổi tên trigger ta phải xóa tên cũ với từ khóa DROP trước khi tạo lại theo tên mới với từ khóa CREATE.

2.3 Xóa trigger

Để xóa trigger đã có ta dùng cú pháp lệnh:

DROP TRIGGER <Tên trigger>

3. KÍCH HOẠT TRIGGER DỰA TRÊN SỰ THAY ĐỔI DỮ LIỆU TRÊN CỘT (IF UPDATE).

Thay vì chỉ định một trigger được kích hoạt trên một bảng, ta có thể chỉ định trigger được kích hoạt và thực hiện những thao tác cụ thể khi việc thay đổi dữ liệu chỉ liên quan đến một số cột nhất định nào đó của bảng. Trong trường hợp này, ta sử dụng mệnh đề **IF UPDATE** trong trigger. IF UPDATE không sử dụng được đối với câu lệnh DELETE.

Trở lại ví dụ 2 trong phần định nghĩa trigger:

Phân tích: ngoài 4 trường hợp trên (thêm, xóa trong bảng nhập và xuất) ta còn 2 trường hợp cho việc sửa cột mã hàng hoặc số lượng trong bảng nhập và xuất.

+ Khi ta sửa tại cột MaHang hoặc SoLuongNhap của một mã hàng trong bảng NHAP thì có hai sự kiện xảy ra là INSERTED và DELETED. Lúc đó số lượng tồn trong bảng HANGHOA sẽ thực hiện hai bước:

- $SoLuongTon + INSERTED.SoLuongNhap$ (tức là thêm số lượng mới)
- $SoLuongTon - DELETED.SoLuongNhap$ (tức là bỏ thêm số lượng cũ)

+ Khi ta sửa tại cột MaHang hoặc SoLuongXuat của một mã hàng trong bảng XUAT thì có hai sự kiện xảy ra là INSERTED và DELETED. Lúc đó số lượng tồn trong bảng HANGHOA sẽ thực hiện hai bước:

- $SoLuongTon + DELETED.SoLuongXuat$ (tức là bỏ xuất số lượng cũ)
- $SoLuongTon - INSERTED.SoLuongXuat$ (tức là xuất số lượng mới)

Viết lệnh tạo trigger cho 2 trường hợp trên:

+/- Sửa nhập.

Create trigger trNHAPUpdate

ON NHAP

FOR Update

AS

--Sửa cột mã hàng hoặc số lượng:

If update (MaHang) OR update (SoLuongNhap)

BEGIN

Update HANGHOA

Set HANGHOA.SoLuongTon = HANGHOA.SoLuongTon +
INSERTED.SoLuongNhap

From HANGHOA inner join INSERTED on HANGHOA.MaHang =
INSERTED.MaHang

```

Update HANGHOA
Set    HANGHOA.SoLuongTon    =    HANGHOA.SoLuongTon    -
DELETED.SoLuongNhap
From  HANGHOA inner join DELETED on HANGHOA.MaHang =
DELETED.MaHang
END

```

+/ Sửa xuất.

Create trigger trXUATUpdate

ON XUAT

FOR Update

AS

--Sửa cột mã hàng hoặc số lượng đều bị tác động như nhau:

If update (MaHang) OR update (SoLuongXuat)

BEGIN

Update HANGHOA

```

Set    HANGHOA.SoLuongTon    =    HANGHOA.SoLuongTon    +
DELETED.SoLuongXuat

```

```

From  HANGHOA inner join DELETED on HANGHOA.MaHang =
DELETED.MaHang

```

Update HANGHOA

```

Set    HANGHOA.SoLuongTon    =    HANGHOA.SoLuongTon    -
INSERTED.SoLuongXuat

```

```

From  HANGHOA inner join INSERTED on HANGHOA.MaHang =
INSERTED.MaHang

```

END

Mệnh đề IF UPDATE có thể xuất hiện nhiều lần trong phần thân của trigger. Khi đó, mệnh đề IF UPDATE nào đúng thì phần câu lệnh của mệnh đề đó sẽ được thực thi khi trigger được kích hoạt.

4. SỬ DỤNG TRIGGER VÀ GIAO TÁC (TRANSACTION)

4.1 Giao tác:

Khi một trigger được kích hoạt, SQL Server luôn tạo ra một giao tác theo dõi những thay đổi do câu lệnh kích hoạt trigger hoặc do bản thân trigger gây ra. Sự theo dõi này cho phép cơ sở dữ liệu quay trở lại trạng thái trước đó bởi lệnh ROLLBACK TRAN (hoặc ROLLBACK TRANSACTION).

Ví dụ: từ ví dụ phân định nghĩa trigger, sau đây ta xây dựng trigger không cho phép thêm vào một bản ghi trong bảng XUAT khi số lượng hàng xuất lớn hơn SoLuongTon trong bảng HANGHOA.

```
create trigger trKIEMTRASOLUONGXUAT
ON XUAT
FOR Insert
AS
BEGIN
    declare @soLuongXuat decimal(18,0)
    declare @soLuongTon decimal(18,0)
    declare @maHang int
    select @maHang = MaHang from INSERTED
    select @soLuongXuat = SoLuongXuat from INSERTED
    select @soLuongTon = SoLuongTon from HANGHOA
    where MaHang = @maHang
    if(@soLuongTon < @soLuongXuat)
    Begin
        print N'số lượng nhập vào lớn hơn số lượng hiện có'
        rollback tran --hoặc RollBack Transaction
    End
END
```

Tiến hành thêm vào bảng XUAT số liệu như sau:

insert into XUAT

values(DHX02, 5, 1000)

```
{1 row(s) affected}  
số lượng nhập vào lớn hơn số lượng hiện có  
Msg 3609, Level 16, State 1, Line 2  
The transaction ended in the trigger. The batch has been aborted.
```

4.2 Ddl trigger

Được giới thiệu trong SQL Server 2005, khác với DML trigger được kích hoạt khi có sự thay đổi dữ liệu trên bảng, DDL trigger được thiết kế để đáp ứng lại các sự kiện diễn ra trên server hay trên CSDL. Một DDL trigger có thể được kích hoạt khi người dùng thực hiện các lệnh CREATE TABLE hay DROP TABLE. Ở cấp độ server, DDL trigger có thể được kích hoạt khi có một tài khoản mới được tạo ra

DDL trigger được lưu trữ trong CSDL mà DDL trigger được gắn vào. Với các Server DDL Trigger theo dõi các thay đổi ở cấp độ Server, được lưu trữ trong CSDL master.

DDL trigger được tạo ra bằng câu lệnh CREATE TRIGGER với cấu trúc như sau:

```
CREATE TRIGGER <tên_trigger>  
ON { ALL SERVER | DATABASE }  
FOR { loại_sự_kiện } [ ,...n ]  
AS { các_câu_lệnh_SQL }
```

Trong đó:

ALL SERVER | DATABASE: quy định trigger sẽ kích hoạt dựa trên các sự kiện diễn ra trên Server hay các sự kiện diễn ra trên CSDL.

loại_sự_kiện: là một sự kiện đơn ở cấp độ Server hay cấp độ CSDL làm kích hoạt DDL trigger như: CREATE_TABLE, ALTER_TABLE, DROP_TABLE...

Ví dụ 1: Câu lệnh dưới đây xây dựng một trigger được kích hoạt khi xảy ra các sự kiện ở cấp độ CSDL. Trigger này sẽ ngăn chặn các lệnh DROP TABLE và ALTER TABLE.

```

create trigger trSUANTOAN
on database
for CREATE_TABLE, DROP_TABLE
as
    print N'Phải xóa trigger trSUANTOAN trước khi ALTER hay DROP
    bảng'
rollback tran

```

Ví dụ 2: Câu lệnh dưới đây xây dựng một trigger được kích hoạt khi xảy ra các sự kiện ở cấp độ Server. Trigger này sẽ ngăn chặn việc tạo ra một account login mới

```

IF EXISTS (SELECT * FROM sys.server_triggers
           WHERE name = 't_DoNotAllowCreateNewLogin')
DROP TRIGGER t_DoNotAllowCreateNewLogin
ON ALL SERVER
GO
CREATE TRIGGER t_DoNotAllowCreateNewLogin
ON ALL SERVER
FOR CREATE_LOGIN
AS
    PRINT N'Phải DROP trigger t_DoNotAllowCreateNewLogin trước khi
    tạo account'
rollback
GO

```

Tiến hành tạo một account login mới:

```
create login test with password = '123456'
```

```

Phải DROP trigger t_DoNotAllowCreateNewLogin trước khi tạo account
Msg 3609, Level 16, State 2, Line 1
The transaction ended in the trigger. The batch has been aborted.

```

5. VÔ HIỆU HÓA HAY KÍCH HOẠT TRIGGER (DISABLE/ENABLE TRIGGER)

Trigger cần bị vô hiệu hóa trong một số trường hợp:

- Trigger gây ra lỗi trong quá trình xử lý CSDL.
- Quá trình nhập hay khôi phục những dữ liệu không thỏa trigger.

Vô hiệu hóa trigger bằng lệnh **DISABLE TRIGGER** có cấu trúc như sau:

DISABLE TRIGGER < *tên_trigger* >

ON { *tên_đối_tượng* | **DATABASE** | **SERVER** }

Ví dụ 1: Ví dụ này sẽ vô hiệu hóa trigger *t_DoNotAllowCreateNewLogin*

disable trigger t_DoNotAllowCreateNewLogin

on all server

Tiến hành tạo một account login mới:

create login newLogin with password = '12345'

Command(s) completed successfully.

Ví dụ 2: Ví dụ này sẽ khôi phục lại trigger *t_DoNotAllowCreateNewLogin*

ENABLE *trigger t_DoNotAllowCreateNewLogin*

on all server

Tiến hành tạo một account login mới:

create login newLogin1 with password = '12345'

Phải DROP trigger t_DoNotAllowCreateNewLogin trước khi tạo account
Msg 3609, Level 16, State 2, Line 1
The transaction ended in the trigger. The batch has been aborted.

6. CÂU HỎI ÔN TẬP

1/- Trigger sẽ được kích hoạt trong trường hợp nào?

2/- Hãy nêu các đặc điểm của trigger?

3/- Hãy nêu các khả năng của trigger?

4/- Hãy nêu cú pháp lệnh tạo trigger đối với tác động của thao tác dữ liệu?

5/- Hãy nêu cú pháp lệnh tạo trigger đối với tác động của định nghĩa dữ liệu?

6/- Hãy nêu cú pháp lệnh vô hiệu hóa và kích hoạt lại trigger?

7/- Hãy làm các yêu cầu thuộc phần trigger trong phần luyện tập?

BÀI 9: QUẢN LÝ GIAO DỊCH VÀ KHÓA

Mã bài: MD15_09

Giới thiệu:

Bài học giúp sinh viên thấy được tầm quan trọng của việc quản lý giao dịch và viết được một số module để quản lý giao dịch.

Mục tiêu:

- Trình bày được khái niệm giao dịch;
- Trình bày được tầm quan trọng của việc quản lý giao dịch;
- Nêu được vai trò của khóa trong quản lý giao dịch;
- Viết được một số module để quản lý giao dịch;
- Hoàn thành đầy đủ bài tập một cách tích cực, chủ động khoa học và sáng tạo.

1. GIAO DỊCH

1.1 Khái niệm

Trong hệ thống quản lý CSDL đa người dùng, các mục dữ liệu chứa trong nguồn tài nguyên chính có thể bị truy cập đồng thời bởi những chương trình của người sử dụng, chúng lấy dữ liệu một cách liên tục từ CSDL và thay đổi CSDL. Sự thi hành của chương trình truy cập hoặc thay đổi nội dung của CSDL được gọi là giao dịch (transaction).

Một hệ quản trị CSDL là hệ thống một người dùng (single-user) nếu tại một thời điểm chỉ có nhiều nhất một người sử dụng hệ thống, và hệ quản trị CSDL được gọi là hệ thống đa người dùng (multi-user) nếu nhiều người dùng có thể sử dụng hệ thống cùng một lúc.

Ví dụ, một hệ thống đặt vé máy bay được sử dụng bởi hàng trăm đại lý và đồng thời các người dùng riêng. Các hệ thống trong các ngân hàng, các đại lý bảo hiểm và các sổ giao dịch chứng khoán cũng được thực hiện bởi nhiều người sử dụng và những người dùng này xác nhận các giao dịch cùng một lúc. Nhiều người sử dụng có thể sử dụng hệ thống máy tính một cách đồng thời bởi vì khái niệm đa chương trình cho phép máy tính xử lý nhiều chương trình trong cùng một lúc.

Một giao dịch sẽ bao gồm các thao tác đọc và ghi. Các giao dịch được xác nhận bởi nhiều người sử dụng khác nhau có thể thực hiện đồng thời và có thể truy cập, cập nhật cùng các mục giống nhau trong CSDL.

1.2 Các thuộc tính của giao dịch

Mặc dù có vẻ dễ dàng định nghĩa, các giao dịch lại không dễ dàng để thi hành trong một ứng dụng. Tách rời việc viết mã, một nền tảng chắc chắn được yêu cầu để hỗ trợ các giao dịch.

Trước khi chúng ta thảo luận về nền tảng, chúng ta tìm hiểu các thuộc tính của giao dịch. Một giao dịch có bốn thuộc tính cụ thể là, tính hoàn chỉnh (Atomicity), tính nhất quán (Consistency), tính riêng biệt (Isolation) và tính bền vững (Durability). Các thuộc tính này được gọi chung là thuộc tính ACID.

□ **Atomicity:** Một giao dịch phải là một thao tác trọn vẹn. Các thành phần của một giao dịch là không thể phân chia được (nguyên tử). Tất cả các thành phần trong một giao dịch đều phải xác nhận (commit) hoặc hủy bỏ (rollback). Nếu có bất kỳ thành phần nào trong giao dịch lỗi, toàn bộ giao dịch sẽ hủy đi.

Ví dụ trong một ngân hàng, có 2 số dư tài khoản đang được cập nhật (một tài khoản là có và tài khoản kia là nợ) thông qua một giao dịch. Nếu giao dịch được xác nhận thì cả 2 tài khoản được cập nhật dữ liệu. Nếu vì một lý do nào đó, giao dịch bị dừng trước khi việc cập nhật trên cả 2 tài khoản thành công thì không có số dư nào được cập nhật và những thay đổi được làm trên bất kỳ số dư tài khoản nào được hủy bỏ. Một giao dịch không thể được xác nhận theo từng phần riêng biệt.

□ **Consistency:** Khi hoàn thành, giao dịch phải để dữ liệu ở dạng nhất quán. Nói một cách

khác, trước khi giao dịch bắt đầu, dữ liệu trong kho chứa dữ liệu ở trong trạng thái nhất quán. Trong một giao dịch đang thực hiện, dữ liệu có thể ở dạng không nhất quán (ví dụ, dữ liệu có thể bị thay đổi từng phần). Tuy nhiên, khi một giao tác hoàn tất thành công, dữ liệu sẽ một lần nữa trở về dạng nhất quán. Những sự thay đổi, được làm với dữ liệu bởi giao dịch không làm hỏng dữ liệu hoặc một giao dịch không nên để dữ liệu ở dạng không ổn định.

Một lần nữa lấy ví dụ về ngân hàng. Trước khi giao dịch bắt đầu, tổng của tất cả số dư tất cả các tài khoản ở dạng nhất quán. Trong quá trình thực hiện một nửa giao dịch, số dư của một tài khoản bị giảm và số dư của các tài khoản khác vẫn không bị thay đổi. Do đó, tổng số dư của tất cả tài khoản không nhất quán (không có giá trị bằng ban đầu). Sau khi giao tác hoàn thành, tổng số dư của các tài khoản lại trở về trạng thái nhất quán.

□ **Isolation:** Tất cả các giao dịch mà thay đổi dữ liệu trong kho chứa dữ liệu là riêng biệt với nhau, điều này có nghĩa các giao dịch phải đứng một mình; nó

không phụ thuộc hoặc ảnh hưởng đến các giao dịch khác. Một giao dịch đang thay đổi có thể truy cập dữ liệu trước khi (một giao dịch khác, bắt đầu sử dụng cùng dữ liệu đó) hoặc sau khi (giao dịch khác sử dụng cùng dữ liệu đó hoàn thành) giao tác khác đã hoàn tất. Thêm vào đó, khi giao dịch thay đổi dữ liệu, nếu bất kỳ quá trình nào khác sử dụng cùng dữ liệu đồng thời thì dữ liệu đã thay đổi phải không ảnh hưởng trừ khi giao dịch khác đã xác nhận thành công. Nói một cách khác, các thành phần của hệ thống không bao giờ được xem như là một giao dịch riêng biệt.

□ **Durability:** Giao dịch bền vững chỉ sau khi giao dịch được xác nhận, một thao tác sẽ hoàn thành. Nếu hệ thống bị hỏng hoặc khởi động lại sau khi giao dịch đó vừa hoàn thành, sự thay đổi đã làm cho dữ liệu phải được đảm bảo vẫn tồn tại sau khi hệ

1.3 Các kiểu giao dịch

Các giao dịch có thể được phân chia thành 3 loại:

- ✓ Giao dịch tường minh
- ✓ Giao dịch ngầm định
- ✓ Giao dịch tự động xác nhận

Chúng ta hãy học về các kiểu giao dịch này chi tiết dưới đây.

1.3.1 *Giao dịch tường minh*

Một giao dịch tường minh là một giao dịch mà bạn khai báo tường minh cả điểm bắt đầu và kết thúc của giao dịch. Một giao dịch tường minh xảy ra khi bạn chỉ thị lệnh BEGIN TRANS và COMMIT. Chế độ giao dịch tường minh tồn tại trong suốt giao dịch. Người dùng khởi đầu kiểu giao dịch. Một giao dịch tường minh cụ thể được diễn tả như dưới đây.

BEGIN TRAN

INSERTRECORD

DELETE RECORD

COMMITTRAN

1.3.2 *Giao dịch ngầm định*

Một giao dịch ngầm định xảy ra khi bạn gửi lệnh thay đổi dữ liệu tới SQL Server mà không chỉ ra cụ thể khối BEGIN TRAN ...COMMIT TRAN bao bọc các lệnh này. SQL Server thêm khối một cách tự động hoặc ngầm định giống như

kiểu của giao dịch được khởi tạo bằng SQL Server thông qua lệnh SETIMPLICIT_TRANSACTION ON

1.3.3 Giao dịch xác nhận tự động

Mọi câu lệnh T-SQL đều được xác nhận hoặc huỷ bỏ khi nó hoàn thành. Nếu câu lệnh thực hiện thành công, nó được xác nhận; nếu nó bắt gặp bất kỳ lỗi nào, nó sẽ bị huỷ bỏ. SQL Server thực hiện các thao tác kết nối trong cơ chế tự động bất kỳ khi nào cơ chế mặc định không bị ghi đè bởi các giao dịch tường minh hoặc ngầm định. Nó là cơ chế quản lý giao dịch mặc định của SQL Server.

1.4 Các cấp độ giao dịch riêng

Theo chuẩn ANSI SQL định nghĩa bốn kiểu mức độ riêng của các giao tác. Các cấp độ này xác định cách có một giao tác cho dữ liệu bị sai. Những cấp độ riêng đó là:

1. Đọc giao dịch chưa xác nhận (ReadUncommitted)
2. Đọc giao dịch đã xác nhận (Read Committed)
3. Đọc lặp lại (RepeatableRead)
4. Serializable

1.4.1 ReadUncommitted

Trong trường hợp này SQL Server không đưa ra các khoá dùng chung trong khi đọc dữ liệu vì vậy bạn có thể đọc một giao dịch chưa xác nhận mà có thể bị huỷ bỏ sau đó. Mức độ riêng biệt này cũng được gọi là mức đọc bị nhiễm bẩn. Đây là mức thấp nhất. Nó chỉ đảm bảo dữ liệu hư hỏng ở mức vật lý không thể được đọc.

1.4.2 ReadCommitted

Đây là một mức mặc định riêng trong SQL. Trong trường hợp này, SQL Server sẽ sử dụng các khoá dùng chung trong quá trình đọc dữ liệu. Nó đảm bảo dữ liệu hỏng ở mức vật lý sẽ không được đọc và sẽ không bao giờ đọc dữ liệu ứng dụng khác đã thay đổi và chưa được xác nhận. Nó chỉ đọc dữ liệu khi nó đã được xác nhận (committed).

1.4.3 RepeatableRead

Trong trường hợp này, việc đọc dữ liệu không sạch và không lặp không thể xảy ra. Điều này có nghĩa là các khoá sẽ được đặt trên tất cả dữ liệu được sử dụng trong truy vấn, và giao dịch khác không thể cập nhật dữ liệu.

1.4.4 Serializable

Đây là mức hạn chế nhất. Trong trường hợp này, các giá trị ảo không thể xảy ra. Nó ngăn chặn các người sử dụng khác trong lúc đang cập nhật hoặc chen các bản ghi vào dữ liệu cho đến khi giao dịch hoàn thành.

Chú ý: Hành động ảo xảy ra khi một giao dịch cố gắng chọn một hàng không tồn tại và một giao dịch thứ hai chen bản ghi trước khi giao dịch đầu tiên kết thúc. Nếu bản ghi được chen vào, bản ghi này sẽ xuất hiện ảo trong giao dịch đầu tiên, xuất hiện không nhất quán và biến mất.

Mức riêng thấp hơn này tăng đồng thời và giảm sự chờ đợi các giao dịch khác. Mặt khác việc tăng này cũng tăng nguy cơ đọc dữ liệu sai. Tuy nhiên có một mức cao hơn giảm đồng thời và tăng trong khi đợi giao dịch khác, nhưng giảm nguy cơ đọc dữ liệu sai. Mức cao hơn dành được khi các giao dịch được hoàn thành.

1.5 Cách sử dụng tệp nhật ký giao dịch (transaction log)

Nhật ký giao dịch là một tệp riêng (hoặc một vùng lưu trữ riêng) nơi máy chủ CSDL lưu trữ các thao tác được thực hiện.

Nhật ký giao dịch là một phần quan trọng để bảo vệ dữ liệu trong CSDL. Bằng cách lưu trữ các dấu vết của thao tác trong nhật ký, máy chủ CSDL làm cho nó có khả năng phục hồi lại khi có sự cố xảy ra. Giả sử đĩa cứng lưu trữ dữ liệu bị hỏng. Nếu có sao lưu nhật ký giao dịch của CSDL trên ổ cứng riêng biệt, trước tiên là có thể dễ dàng phục hồi lại dữ liệu trên máy sao lưu và sau đó ra lệnh cho server lập lại tất cả các thay đổi đã được ghi lại trong nhật ký giao dịch sau khi phục hồi xong dữ liệu. Nhật ký giao dịch đảm bảo tính hoàn chỉnh và bền vững trên CSDL.

SQL Server ghi lại mọi thay đổi trên CSDL vào nhật ký giao dịch, do đó nếu một giao dịch bắt đầu thi hành nhưng không được hoàn thành, bạn có thể lấy lại các sự thay đổi này từ tệp nhật ký và thực hiện lại hoặc khôi phục lại chúng. Khả năng để khôi phục những thay đổi có nghĩa là các Server giao tác còn nguyên vẹn như khi chưa thực hiện. Khi một giao dịch được xác nhận, SQL Server lập tức ghi tất cả thành các bản ghi nhật ký tương ứng với các giao dịch đó và lưu trữ vĩnh viễn trên đĩa. Kể cả trong trường hợp, thậm chí hệ thống bị hỏng trước khi SQL Server kịp ghi bất kỳ sự thay đổi dữ liệu nào lên đĩa, các bản ghi nhật ký đã được lưu trên đĩa. Khi SQL Server khởi động lại, các bản ghi nhật ký này đưa ra đủ thông tin để phục hồi, hoặc quay lại trạng thái trước đó, kể cả bất kỳ giao tác nào đã hoàn thành nhưng dữ liệu tương ứng của chúng chưa được ghi lên đĩa. Khả

năng để phục hồi các giao dịch có nghĩa là các giao dịch trên SQL Server có tính bền vững.

Một ưu điểm khác của nhật ký giao dịch là bạn có thể ghi dự phòng CSDL tăng (một thủ tục ghi dự phòng, trong đó chương trình ghi dự phòng đĩa cứng chỉ ghi những tệp đã có thay đổi kể từ lần ghi dự phòng cuối cùng). Bằng cách sao lưu nhật ký giao dịch nhưng không sao lưu toàn bộ CSDL, mà chỉ lưu giữ những thay đổi kể từ lần ghi dự phòng cuối cùng. Trình tự của việc sao lưu là các bản ghi lưu trữ sự thay đổi của CSDL sau lần sao lưu cuối cùng.

1.6 Mô tả giao dịch bằng câu lệnh T-SQL

T-SQL sử dụng bốn câu lệnh để tạo giao dịch:

- ✓ BEGINTRANSACTION
- ✓ COMMIT TRANSACTION
- ✓ ROLLBACKTRANSACTION
- ✓ SAVETRANSACTION

Các biến dưới đây được sử dụng trong quá trình thực hiện giao dịch

@@ERROR

@@TRANCOUNT – Nó được sử dụng để đếm số lần các giao dịch mở đồng thời.

Chúng ta tìm hiểu cú pháp của những câu lệnh này và học cách sử dụng quá trình chạy bó lệnh (batch) bằng lệnh T-SQL.

1.6.1 BEGIN TRANSACTION

Câu lệnh này được sử dụng trong SQL server để bắt đầu một giao dịch

Cú pháp:

```
BEGIN TRANS[ACTION] [transaction_name | @variable_name]  
[WITH MARK ['description']]
```

- Ta cũng có thể sử dụng các câu lệnh cơ bản BEGIN TRANS hoặc BEGIN TRANSACTION.
- Tiếp theo bạn chỉ ra tên của giao dịch, hoặc tên biến chứa giao dịch để chúng ta có thể tham chiếu tới giao dịch này bằng tên biến đó khi thực hiện xác nhận (commit) hoặc (quay lại) rollback nó.

- Mệnh đề WITH MARK chèn vào một đánh dấu trong nhật ký giao dịch cho CSDL, sử dụng diễn tả thêm về thời điểm hiện tại như một định danh độc lập. Bằng cách sử dụng lệnh RESTORE giúp phục hồi CSDL cũng như các trạng thái trước khi giao dịch hoặc các trạng thái vừa thực hiện sau giao dịch khi bạn đang phục hồi lại một vấn đề nào đó trong CSDL.

- Các giao dịch có thể lồng nhau.

Chúng ta xét ví dụ dưới đây:

Ví dụ 1:

```
BEGIN TRANSACTION
```

```
UPDATE titles
```

```
SET price=20.00
```

```
WHERE title_id = 'TC7777'
```

```
BEGIN TRANSACTION
```

```
UPDATE titles
```

```
SET type='potboiler'
```

```
WHERE title_id='TC7777'
```

```
COMMIT TRANSACTION
```

1.6.2 COMMITTRANSACTION

Câu lệnh này được sử dụng để xác nhận giao dịch. Một khi giao dịch được xác nhận, nó không thể quay trở về trạng thái trước đó (rolled back).

Cú pháp:

```
COMMIT TRANS[ACTION] [transaction_name | @name_variable]
```

```
COMMIT [WORK]
```

1.6.3 ROLLBACK TRANSACTION

Câu lệnh này được sử dụng huỷ bỏ tất cả mọi thay đổi từ một lệnh BEGIN TRANSACTION gần nhất. Cũng có thể cung cấp một tên giao dịch là một hằng hoặc biến, nhưng SQL server bỏ qua tên. Cũng có thể quay lại (roll back) một phần của giao tác bằng cách cung cấp những điểm trở lưu trữ.

Cú pháp:

```
ROLLBACK TRANS[ACTION]
```

[transaction_name | @variablename | savepointname |
@savepoint_variable]

2. KHÓA.

Một khoá là hạn chế quyền truy cập dữ liệu trong môi trường đa người dùng. SQL Server tự động khoá những người sử dụng ra ngoài một bản ghi xác định, trường, hoặc tệp tin, để duy trì bảo mật, hoặc ngăn chặn các vấn đề thao tác với dữ liệu hiện tại.

Microsoft SQL Server 2000 trở đi sử dụng cơ chế khoá để đảm bảo tính toàn vẹn của giao tác và tính nhất quán của CSDL. Cơ chế khoá ngăn chặn người sử dụng đọc dữ liệu bị thay đổi bởi người dùng khác, và ngăn chặn nhiều người dùng cùng một lúc thay đổi cùng một dữ liệu. Nếu cơ chế 100 Database Design and Implementation with SQL Server khoá không được sử dụng, dữ liệu trong CSDL có thể bị sai một cách logic, và các truy vấn được thực thi dựa trên dữ liệu có thể cho kết quả không mong đợi. Mặc dù SQL Server bắt buộc cơ chế khoá một cách tự động thông qua cơ chế khoá động, chúng ta có thể thiết kế các ứng dụng hiệu quả hơn bằng cách tìm hiểu và cấu hình lại cơ chế khoá trong ứng dụng của chúng ta.

Khi nhiều người dùng thay đổi CSDL, dữ liệu phải có vài cơ cấu để đảm bảo những sự thay đổi này nhất quán về mặt dữ liệu. Khái niệm cơ bản đằng sau cơ chế khoá là một người sử dụng sẽ cần một quyền truy cập riêng vào bảng, sau đó máy chủ khoá bảng cho người dùng này. Khi người dùng hoàn thành thao tác trên bảng, khoá được giải phóng. Điều này giúp cho các người sử dụng khác tiếp tục thao tác được với dữ liệu.

2.1 Các kiểu của khoá

2.1.1 *Khoá bi quan (Pessimistic Lock):*

Một khoá được áp dụng ngay khi dữ liệu được thay đổi và được giải phóng ngay khi thao tác thay đổi được hoàn tất. Điều này đảm bảo không có người sử dụng khác thay đổi dữ liệu.

2.1.2 *Khoá tối ưu (Optimistic Lock):*

Một khoá được áp dụng chỉ khi việc thay đổi dữ liệu hoàn thành và dữ liệu thực sự là sẵn sàng để ghi lên bảng thực sự.

2.1.3 Các khoá dùng chung(SharedLocks):

Các khoá dùng chung được sử dụng cho các thao tác không có sự thay đổi dữ liệu hoặc cập nhật dữ liệu, ví dụ như lệnh SELECT. Chúng được sử dụng cho các thao tác chỉ đọc. Các khoá dùng chung cho phép giao dịch hiện tại đọc (SELECT) nguồn tài nguyên. Khi một khoá dùng chung tồn tại trên nguồn tài nguyên, không một giao dịch nào khác có thể thay đổi dữ liệu. Các khoá dùng chung được giải phóng ngay khi dữ liệu đã đọc xong.

2.1.4 Các khoá dành riêng (Exclusive Locks):

Các khoá dành riêng được sử dụng cho các thao tác thay đổi dữ liệu, giống như, cập nhật (UPDATE), xoá (DELETE), và thêm (INSERT). Khoá dành riêng đảm bảo sao cho nhiều thao tác cập nhật không thể thực hiện trên cùng một nguồn tài nguyên một cách đồng thời. Không có giao dịch nào khác có thể đọc hoặc sửa chữa dữ liệu đã bị khoá với khoá dành riêng. Các khoá dành riêng được giữ đến khi giao dịch đó được xác nhận (commit) hoặc quay lại (roll back).

2.1.5 Các khoá cập nhật (UpdateLocks):

Các khoá cập nhật được sử dụng để cập nhật giao dịch. Khi dữ liệu đang bị truy suất, đầu tiên, nó bị khoá bằng một khoá dùng chung. Một khi dữ liệu được định vị, khoá dùng chung trở thành khoá dành riêng, để thay đổi dữ liệu. Một vấn đề nảy sinh khi một trong hai phiên làm việc với các khoá dùng chung trên cùng một nguồn tài nguyên, muốn thay đổi thành khoá dành riêng để thay đổi dữ liệu, nhưng không thể thực hiện bởi vì khoá dành riêng không tương thích với khoá dùng chung. Các khoá cập nhật giúp ta tránh vấn đề khoá chết (deadlock). Tại một thời điểm chỉ một phiên làm việc có thể giành khoá cập nhật trên một nguồn tài nguyên.

2.2 Khoá chết (Deadlock)

Một khoá chết (deadlock) là một điều kiện có thể xảy ra trên bất kỳ một hệ thống CSDL với nhiều người dùng. Một deadlock xảy ra khi hai người dùng (hoặc các phiên làm việc) có các khoá trên các đối tượng riêng biệt, và mỗi người sử dụng muốn đặt một khoá trên đối tượng của người sử dụng khác. Và mỗi người sử dụng đợi người kia giải phóng khoá của họ.

Chúng ta hãy xem xét một ví dụ của deadlock. Transaction_1 có một khoá dành riêng trên bảng Supplier. Transaction_2 giành được một khoá dành riêng trên bảng Part. Và sau đó muốn một khoá trên bảng Supplier. Transaction_2 không thể dành được khoá bởi vì Transaction_1 đã khoá nó. Transaction_1 cũng muốn dành

một khoá trên bảng Part, nhưng không thể dành được nó bởi vì Transaction_2 đã chiếm giữ. Không có giao dịch nào có thể giải phóng khoá nó đang chiếm giữ, tới khi nó xác nhận hoặc quay lui (rollback).

SQL Server tự động giải quyết vấn đề deadlock bằng cách lựa chọn một ứng dụng, và bắt buộc nó giải phóng khoá, trong khi cho phép các ứng dụng khác tiếp tục. Cách tốt nhất đối với deadlock là tránh nó. Bất cứ khi nào một deadlock xảy ra, thời gian và tài nguyên sẽ bị lãng phí. Một cách để tránh vấn đề này là tránh chạy các giao dịch đồng thời. Nếu giao dịch chạy nối tiếp, và nếu chúng cố gắng truy cập cùng dữ liệu, thì giao dịch sau phải đợi giao dịch trước đó thi hành xong.

2.3 Giải quyết vấn đề Deadlock (Resolving Deadlock)

Để giải quyết tình trạng deadlock, chúng ta có thể sử dụng các lệnh dưới đây.

2.3.1.1 SET DEADLOCK_PRIORITY

Điều khiển phương thức phiên làm việc tác động trở lại trong tình trạng deadlock. Các tình trạng Deadlock phát sinh khi hai quá trình xử lý đều khoá dữ liệu, và bất kỳ quá trình xử lý nào cũng không thể giải phóng khoá của nó cho tới khi quá trình khác giải phóng khoá.

Cú pháp:

```
SETDEADLOCK_PRIORITY {LOW | NORMAL | @deadlock_var }
```

Trong đó

- LOW: Chỉ ra phiên làm việc hiện tại được ưu tiên trong quá trình deadlock. Giao dịch bị deadlock được tự động quay lại (rolled back) bởi Microsoft SQL Server, và một lỗi deadlock 1205 được trả về cho ứng dụng tại máy trạm.

- NORMAL: Chỉ ra phiên làm việc được trả về đối với phương thức deadlock mặc định.

@deadlock_var là một biến ký tự xác định phương thức điều khiển deadlock (deadlockhandling). Giá trị của @deadlock_var là 3 nếu LOW được xác định, và là 6 nếu NORMAL được xác định.

Nhận xét

Lệnh SET DEADLOCK_PRIORITY được thực hiện tại thời điểm thi hành hoặc thời điểm chạy và không thực hiện tại thời điểm phân tích.

Các quyền hạn SETDEADLOCK_PRIORITY là quyền hạn mặc định cho tất cả các người dùng.

2.3.1.2 SET LOCK_TIMEOUT

Chỉ ra số phần nghìn của giây mà một câu lệnh đợi một khoá được giải phóng.

Cú pháp:

SETLOCK_TIMEOUT timeout_period

Trong đó

timeout_period: Chỉ ra số phần nghìn giây sẽ trôi qua trước khi Microsoft SQL Server trả về một lỗi khoá. Một giá trị là -1 chỉ ra không có khoảng thời gian nghỉ (time-out) (đó là, đợi mãi mãi). Khi một khoá đợi quá giá trị time-out, một lỗi được trả về. Giá trị 0 có nghĩa không đợi tại tất cả thời điểm và trả về một thông báo ngay khi bắt gặp một khoá.

Nhận xét: Khi bắt đầu một kết nối, sự thiết lập này có giá trị là -1. Sau khi nó bị thay đổi, sự thiết lập mới tác động tới phần còn lại của quá trình kết nối. Sự thiết lập của lệnh SET LOCK_TIMEOUT được thiết lập tại thời điểm thi hành hoặc thời gian chạy và không thiết lập tại thời điểm phân tích. Cơ chế khoá READPAST cung cấp một cơ chế khác để thiết lập tùy chọn này. Các quyền hạn SETLOCK_TIMEOUT cho phép mặc định với tất cả người sử dụng.

BÀI LUYỆN TẬP

Phần luyện tập này được đưa ra nhằm mục đích luyện tập phù hợp với kiến thức đã học và tạo nhiều góc độ nhìn dữ liệu khác nhau dựa theo yêu cầu khác nhau từ phía ứng dụng áp dụng lên Quản trị cơ sở dữ liệu. Tuy nhiên, để có thể làm những dự án lớn kết nối với hệ quản trị cơ sở dữ liệu, chúng ta cần phải kết hợp học môn lập trình windows 2, lập trình web 2.

Để thấu hiểu được kiến thức môn học, chúng ta cần phải tập hợp các yêu cầu của một bài toán thực tế nhằm đi xuyên suốt kiến thức đã học trong quá trình làm bài.

LUYỆN TẬP 1

TẠO CẤU TRÚC CƠ SỞ DỮ LIỆU, CHÈN DỮ LIỆU, TRUY VẤN DỮ LIỆU, THỦ TỤC, TRIGGER, FUNCTION, SECURITY, CONNECTION.

Cho mô hình quan hệ thực thể (E-R) HANGHOA sau :

Create table KHACHHANG

Thuộc tính – field name	Data type	Description
MaKH	nvarchar(25) not null	Mã khách hàng, Primary key
TenKH	nvarchar(35) not null	Tên khách hàng
DiaChi	nvarchar(50)	Địa chỉ khách hàng

Create table HANGHOA

Thuộc tính – field name	Data type	Description
MaHH	nvarchar(25) not null	Mã hàng hóa, Primary key
TenHH	nvarchar(50) not null	Tên hàng hóa
DVT	nvarchar(12) not null	Đơn vị tính
SoLuongTon	int default 0	Số lượng tồn của mặt hàng, default 0

Create table KHO

Thuộc tính – field name	Data type	Description
MaKho	nvarchar(25) not	Mã kho, Primary

	null	key
TenKho	nvarchar(35) not null	Tên kho
DiaChi	nvarchar(50)	Địa chỉ kho

Create table DONHANG

Thuộc tính – field name	Data type	Description
MaDH	nvarchar(12) not null	Mã đơn hàng, Primary key
NgayDH	smallDatetime not null default getdate()	Ngày đơn hàng, default getdate()
MaKH	nvarchar(25) not null	Mã khách hàng, Foreign key
LoaiDH	bit	Loại đơn hàng, qui định : true(!=0): nhập; false(0): xuất

Create table CHITIETDH

Thuộc tính – field name	Data type	Description
MaDH	nvarchar(12) not null	Mã đơn hàng, Primary key, Foreign key
MaHH	nvarchar(25) not null	Mã hàng hóa, Primary key, Foreign key
MaKho	nvarchar(25) not null	Mã kho, Primary key, Foreign key
SoLuong	Int not null	Số lượng, check (SL>0)

DonGia	Decimal (9, 2) not null default 0	Đơn giá, default 0 check (DonGia >= 0)
--------	-----------------------------------	--

Hãy thực hiện viết code lệnh các yêu cầu sau:

A/- Phần luyện tập TẠO CƠ SỞ DỮ LIỆU

- 1/- Tạo cơ sở dữ liệu (tạo file, tạo bảng có ràng buộc thuộc tính, khóa, điều kiện, quan hệ tham chiếu) thông qua giao diện của MS?
- 2/- Tạo cơ sở dữ liệu (tạo file, tạo bảng có ràng buộc thuộc tính, khóa, điều kiện, quan hệ tham chiếu) thông qua lệnh SQL?

B/- Phần luyện tập TRUY VẤN DỮ LIỆU

- 1/- Lấy ra danh sách khách hàng?
- 2/- Lấy ra danh sách khách hàng chưa có đơn hàng nào?
- 3/- Lấy ra tổng tiền của mỗi đơn hàng?
- 4/- Lấy ra các đơn hàng có tổng tiền lớn nhất?
- 5/- Lấy ra khách hàng mà có tổng tiền trên các đơn hàng lớn nhất?

C/- Phần luyện tập STORED PROCEDURE

- 1/- Hãy viết thủ tục cho hành động thêm, sửa, xóa của bảng KHACHHANG. Nếu gặp lỗi hãy trả về thông báo lỗi?
- 2/- Hãy viết thủ tục cho hành động của bảng KHACHHANG:
 - a)- Truy xuất?
 - b)- Truy xuất kiểu tìm kiếm tuyệt đối theo cột MaKH?
 - c)- Truy xuất kiểu tìm kiếm tương đối theo cột MaKH và TenKH?
- 3/- Hãy viết thủ tục cho hành động thêm, sửa, xóa của bảng HANGHOA. Nếu gặp lỗi hãy trả về thông báo lỗi?
- 4/- Hãy viết thủ tục cho hành động của bảng HANGHOA:
 - a)- Truy xuất?

b)- Truy xuất kiểu tìm kiếm tuyệt đối theo cột MaHH?

c)- Truy xuất kiểu tìm kiếm tương đối theo cột MaHH và TenHH?

5/- Hãy viết thủ tục cho hành động thêm, sửa, xóa của bảng KHO. Nếu gặp lỗi hãy trả về thông báo lỗi?

6/- Hãy viết thủ tục cho hành động của bảng KHO:

a)- Truy xuất?

b)- Truy xuất kiểu tìm kiếm tuyệt đối theo cột MaKho?

c)- Truy xuất kiểu tìm kiếm tương đối theo cột MaKho, TenKho và DiaChi?

7/- Hãy viết thủ tục cho hành động thêm, sửa, xóa của bảng DONHANG. Nếu gặp lỗi hãy trả về thông báo lỗi. Nếu thực hiện thành công đối với hành động thêm và sửa thì hãy trả về MaDH đã thành công đó?

8/- Hãy viết thủ tục cho hành động của bảng DONHANG:

a)- Truy xuất, có thêm cột tổng tiền (TongTien)?

b)- Truy xuất kiểu tìm kiếm tuyệt đối theo cột MaDH, có thêm cột tổng tiền (TongTien)?

c)- Truy xuất kiểu tìm kiếm tương đối theo cột MaKH, TenKH và NgayDH (trong đó NgayDH có thời hạn từ ngày đến ngày), có thêm cột tổng tiền (TongTien)?

9/- Hãy viết thủ tục cho hành động thêm, sửa, xóa của bảng CHITIETDH. Nếu gặp lỗi hãy trả về thông báo lỗi?

10/- Hãy viết thủ tục cho hành động của bảng CHITIETDH:

a)- Truy xuất, có thêm cột thành tiền (ThanhTien)?

b)- Truy xuất kiểu tìm kiếm tuyệt đối theo cột MaDH, có thêm cột thành tiền (ThanhTien)?

c)- Truy xuất kiểu tìm kiếm tương đối theo cột MaKH, TenKH và NgayDH (trong đó NgayDH có thời hạn từ ngày đến ngày), có thêm cột thành tiền (ThanhTien)?

D/- Phần luyện tập FUNCTION

1/- Hãy viết hàm tính tổng tiền của một đơn hàng?

2/- Hãy viết hàm tính số lượng tồn của mỗi mặt hàng trong thời gian từ ngày đến ngày?

3/- Hãy viết hàm tính tổng doanh số của một khách hàng trong thời gian từ ngày đến ngày?

E/- Phần luyện tập bảo mật và phân quyền (SECURITY, PERMISSION).

1/- Hãy tạo một login user với tên hanghoauser và mật khẩu 'hh' là chủ nhân của cơ sở dữ liệu HANGHOA?

2/- Hãy tạo database user với tên như login user cho cơ sở dữ liệu HANGHOA?

3/- Hãy tạo database user với tên xemhanghoa và cấp quyền SELECT, UPDATE trên bảng dữ liệu HANGHOA cho database user này?

F/- Phần luyện tập TRIGGER

1/- Hãy chỉ ra các ràng buộc toàn vẹn dữ liệu giữa các bảng để viết trigger cho các hành động trên một đơn hàng và chi tiết đơn hàng?

2/- Hãy viết trigger cho các hành động thêm, sửa, xóa một đơn hàng. Biết rằng đơn hàng loại nhập có giá trị bằng 1, loại xuất có giá trị bằng 0?

3/- Hãy viết trigger cho các hành động thêm, sửa, xóa một chi tiết đơn hàng?

LUYỆN TẬP 2

TẠO CẤU TRÚC CƠ SỞ DỮ LIỆU, CHÈN DỮ LIỆU, TRUY VẤN DỮ LIỆU, THỦ TỤC, TRIGGER, FUNCTION, SECURITY, CONNECTION.

Cho tóm tắt lược đồ cơ sở dữ liệu QLPX sau:

PHANXUONG(#MaPX: nvarchar(25), TenPX: nvarchar(35), SoLuongNhanVien: int)

NHANVIEN(#MaNV: nvarchar(25), TenNV: nvarchar(45), NgaySinh: smalldatetime, GioiTinh: bit, NgayVaoLam: smallDatetime, *MaPX: nvarchar(12))

SANPHAM(#MaSP: nvarchar(25), TenSP: nvarchar(35), SoLuongSanPham: int, DonGiaThanhPham: decimal(9,2), SoLuongThanhPham: int)

THANHPHAM(#MaSP, #MaNV, #NgayTP, SoLuongThanhPham: int)

Ghi chú : dấu # ký hiệu cho thuộc tính khóa chính. Dấu * ký hiệu cho khóa ngoại.

Hãy thực hiện viết code lệnh các yêu cầu sau:

A/- Phần luyện tập TẠO CƠ SỞ DỮ LIỆU

1/- Tạo cơ sở dữ liệu (tạo file, tạo bảng có ràng buộc thuộc tính, khóa, điều kiện, quan hệ tham chiếu) thông qua giao diện của MS?

2/- Tạo cơ sở dữ liệu (tạo file, tạo bảng có ràng buộc thuộc tính, khóa, điều kiện, quan hệ tham chiếu) thông qua lệnh SQL?

B/- Phần luyện tập TRUY VẤN DỮ LIỆU

1/- Hãy cho biết danh sách sinh viên mỗi phân xưởng?

2/- Hãy cho biết phân xưởng có nhân viên nhiều nhất?

3/- Hãy cho biết tổng tiền cho mỗi nhân viên làm xong sản phẩm trong tháng 8?

4/- Hãy cho biết nhân viên nhận được tiền thành phẩm nhiều nhất trong mỗi tháng?

C/- Phần luyện tập STORED PROCEDURE

1/- Hãy viết thủ tục cho hành động thêm, sửa, xóa của bảng NHANVIEN. Nếu gặp lỗi hãy trả về thông báo lỗi?

2/- Hãy viết thủ tục cho hành động của bảng NHANVIEN:

a)- Truy xuất?

b)- Truy xuất kiểu tìm kiếm tuyệt đối theo cột MaNV?

c)- Truy xuất kiểu tìm kiếm tuyệt đối theo MaPX đồng thời tương đối theo cột NgaySinh từ ngày đến ngày?

3/- Hãy viết thủ tục cho hành động thêm, sửa, xóa của bảng PHANXUONG. Nếu gặp lỗi hãy trả về thông báo lỗi?

4/- Hãy viết thủ tục cho hành động của bảng PHANXUONG:

a)- Truy xuất?

b)- Truy xuất kiểu tìm kiếm tuyệt đối theo cột MaPX?

c)- Truy xuất kiểu tìm kiếm tương đối theo cột TenPX?

5/- Hãy viết thủ tục cho hành động thêm, sửa, xóa của bảng SANPHAM. Nếu gặp lỗi hãy trả về thông báo lỗi?

6/- Hãy viết thủ tục cho hành động của bảng SANPHAM:

a)- Truy xuất?

b)- Truy xuất kiểu tìm kiếm tuyệt đối theo cột MaSP?

c)- Truy xuất kiểu tìm kiếm tương đối theo cột TenSP?

7/- Hãy viết thủ tục cho hành động thêm, sửa, xóa của bảng THANHPHAM. Nếu gặp lỗi hãy trả về thông báo lỗi?

8/- Hãy viết thủ tục cho hành động của bảng THANHPHAM:

a)- Truy xuất?

b)- Truy xuất kiểu tìm kiếm tuyệt đối hoặc “tất cả” theo cột MaSP phối hợp MaNV?

c)- Truy xuất kiểu tìm kiếm tương đối theo cột NgayTP (từ ngày, đến ngày)?

D/- Phần luyện tập FUNCTION

1/- Hãy viết hàm tính tiền trong ngày cho mỗi nhân viên biết rằng mỗi MaSP có một DonGiaThanhPham?

2/- Hãy viết hàm tính tiền trên mỗi MaSP cho mỗi nhân viên biết rằng mỗi MaSP có một DonGiaThanhPham?

3/- Hãy viết hàm tính tiền cho mỗi nhân viên theo từng tháng biết rằng mỗi MaSP có một DonGiaThanhPham?

4/- Hãy viết hàm tính tiền cho mỗi nhân viên trong khoản thời gian làm thành phẩm (từ ngày, đến ngày) biết rằng mỗi MaSP có một DonGiaThanhPham?

E/- Phần luyện tập SECURITY

1/- Hãy tạo một login user với tên phanxuonguser và mật khẩu 'px' là chủ nhân của cơ sở dữ liệu QLPX.

2/- Hãy tạo database user với tên như login user cho cơ sở dữ liệu QLPX.

3/- Hãy tạo database user với tên xemphanxuong và cấp quyền SELECT, UPDATE trên bảng dữ liệu PHANXUONG, NHANVIEN, THANHPHAM cho database user này.

F/- Phần luyện tập TRIGGER

1/- Hãy chỉ ra các ràng buộc toàn vẹn dữ liệu giữa các bảng để viết trigger cho các hành động trên bảng NHANVIEN và THANHPHAM?

2/- Hãy viết trigger cho các hành động thêm, sửa, xóa một nhân viên?

3/- Hãy viết trigger cho các hành động thêm, sửa, xóa một thành phẩm?

TÀI LIỆU THAM KHẢO

- [1] Nguyễn Thiên Bằng (2007), *Giáo trình SQL Server 2008*, NXB Lao Động - Xã Hội.
- [2] Trần Nguyên Phong, *Giáo trình điện tử SQL Server 2000* của Đại Học Huế.
- [3] Các tài liệu cập nhật nhanh trên internet: *SQL Server 2005*, *SQL Server 2008*.