**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

### ĐẠI HỌC CÔNG NGHỆ TP.HCM

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

**Biên Soạn:**

**ThS. Võ Hoàng Khang**

##### [www.hutech.edu.vn](http://www.hutech.edu.vn/)

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Ấn bản 2015

### MỤC LỤC

[MỤC LỤC I](#_bookmark0)

[HƢỚNG DẪN IV](#_bookmark1)

[BÀI 1: TỔNG QUAN VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU 1](#_bookmark2)

* 1. [TỔNG QUAN 1](#_bookmark3)
  2. [GIỚI THIỆU MICROSOFT SQL SERVER 6](#_bookmark4)
  3. [TRANSACT-SQL NÂNG CAO 15](#_bookmark5)
     1. [Khai báo và sử dụng biến 15](#_bookmark6)
     2. [Các lệnh điều khiển 17](#_bookmark7)

[TÓM TẮT 19](#_bookmark8)

[CÂU HỎI ÔN TẬP 20](#_bookmark9)

[BÀI 2: INDEX (CHỈ MỤC) 21](#_bookmark10)

* 1. [KHÁI NIỆM 21](#_bookmark11)
  2. [PHÂN LOẠI 22](#_bookmark12)
  3. [TẠO INDEX 22](#_bookmark13)
  4. [XÓA INDEX 22](#_bookmark14)

[TÓM TẮT 23](#_bookmark15)

[CÂU HỎI ÔN TẬP 23](#_bookmark16)

[BÀI 3: VIEW (KHUNG NHÌN) 24](#_bookmark17)

* 1. [KHÁI NIỆM 24](#_bookmark18)
  2. [TẠO VIEW 25](#_bookmark19)
  3. [HIỆU CHỈNH VIEW 26](#_bookmark20)
  4. [XÓA VIEW 27](#_bookmark21)

[TÓM TẮT 28](#_bookmark22)

[CÂU HỎI ÔN TẬP 28](#_bookmark23)

[BÀI 4: STORED PROCEDURE (THỦ TỤC NỘI TẠI) 29](#_bookmark24)

* 1. [KHÁI NIỆM 29](#_bookmark25)
  2. [PHÂN LOẠI SP 30](#_bookmark26)
  3. [TẠO SP 32](#_bookmark27)
  4. [SỬA SP 33](#_bookmark28)
  5. [XÓA SP 34](#_bookmark29)

[TÓM TẮT 37](#_bookmark30)

[CÂU HỎI ÔN TẬP 38](#_bookmark31)

[BÀI 5: FUNCTION (HÀM) 39](#_bookmark32)

* 1. [KHÁI NIỆM 39](#_bookmark33)
  2. [PHÂN LOẠI 40](#_bookmark34)
  3. [TẠO FUNCTION 40](#_bookmark35)
  4. [HIỆU CHỈNH FUNCTION 43](#_bookmark36)
  5. [XÓA FUNCTION 44](#_bookmark37)

[TÓM TẮT 45](#_bookmark38)

[CÂU HỎI ÔN TẬP 46](#_bookmark39)

[BÀI 6: TRIGGER (BẪY LỖI) 47](#_bookmark40)

* 1. [KHÁI NIỆM 47](#_bookmark41)
  2. [SỬ DỤNG TRIGGER 47](#_bookmark42)
  3. [NGUYÊN TẮC HOẠT ĐỘNG 48](#_bookmark43)
  4. [PHÂN LOẠI 49](#_bookmark44)
  5. [TẠO TRIGGER 50](#_bookmark45)
  6. [HIỆU CHỈNH TRIGGER 51](#_bookmark46)
  7. [XÓA TRIGGER 51](#_bookmark47)
  8. [MỘT VÀI VÍ DỤ 51](#_bookmark48)

[TÓM TẮT 54](#_bookmark49)

[CÂU HỎI ÔN TẬP 54](#_bookmark50)

[BÀI 7: KIỂU DỮ LIỆU CURSOR 55](#_bookmark51)

* 1. [ĐẶT VẤN ĐỀ 55](#_bookmark52)
  2. [KHÁI NIỆM 56](#_bookmark53)
  3. [KHAI BÁO CURSOR 57](#_bookmark54)
  4. [PHÂN LOẠI CURSOR 58](#_bookmark55)
  5. [THAY ĐỔI DỮ LIỆU TẠI VỊ TRÍ CURSOR 59](#_bookmark56)
  6. [DUYỆT CURSOR 59](#_bookmark57)
  7. [QUY TRÌNH SỬ DỤNG CURSOR 60](#_bookmark58)
  8. [BIẾN CURSOR 64](#_bookmark59)
  9. [KẾT HỢP CURSOR VỚI STORED PROCEDURE 64](#_bookmark60)
  10. [KẾT LUẬN 66](#_bookmark61)

[TÓM TẮT 67](#_bookmark62)

[CÂU HỎI ÔN TẬP 67](#_bookmark63)

[BÀI 8: QUẢN LÝ GIAO TÁC 68](#_bookmark64)

* 1. [GIỚI THIỆU 68](#_bookmark65)
  2. [KHÁI NIỆM 69](#_bookmark66)
  3. [MỘT SỐ THUẬT NGỮ LIÊN QUAN ĐẾN GIAO TÁC 73](#_bookmark67)
  4. [CÁC VẤN DỀ LIÊN QUAN ĐẾN XỬ LÝ TRUY XUẤT DỒNG THỜI 74](#_bookmark68)
     1. [Lost Update (mất dữ liệu cập nhật) 74](#_bookmark69)
     2. [Đọc dữ liệu chưa commit (Uncommitted data, Dirty read) 74](#_bookmark70)
     3. [Thao tác đọc không thể lặp lại (Unrepeatable data) 74](#_bookmark71)
     4. [Bóng ma (Phantom) 74](#_bookmark72)
     5. [Cách giải quyết các vấn đề 74](#_bookmark73)
  5. [CHẾ ĐỘ KHÓA 75](#_bookmark74)
  6. [KHAI BÁO TƢỜNG MINH GIAO TÁC 76](#_bookmark75)
     1. [Mức cô lập cho giao tác: 77](#_bookmark76)
     2. [Các cấp độ khóa: 78](#_bookmark77)
     3. [Dead lock: 79](#_bookmark78)

[TÓM TẮT 81](#_bookmark79)

[CÂU HỎI ÔN TẬP 81](#_bookmark80)

[BÀI 9: BẢO MẬT VÀ QUẢN TRỊ 82](#_bookmark81)

* 1. [GIỚI THIỆU 82](#_bookmark82)
  2. [XÁC THỰC SQL SERVER 82](#_bookmark83)
  3. [TẠO USER TRONG WINDOWS 84](#_bookmark84)
  4. [TẠO GROUP TRONG WINDOWS 85](#_bookmark85)
  5. [VAI TRÒ (ROLE) 96](#_bookmark86)
  6. [QUYỀN 104](#_bookmark87)

[TÓM TẮT 111](#_bookmark88)

[CÂU HỎI ÔN TẬP 111](#_bookmark89)

[BÀI 10: MÃ HÓA DỮ LIỆU TRONG SQL (BÀI ĐỌC THÊM) 112](#_bookmark90)

* 1. [ĐẶT VẤN ĐỀ 112](#_bookmark91)
  2. [CÁC KỸ THUẬT MÃ HÓA 112](#_bookmark92)

[TÓM TẮT 120](#_bookmark93)

[CÂU HỎI ÔN TẬP 120](#_bookmark94)

[BÀI 11: BÀI TẬP THỰC HÀNH 121](#_bookmark95)

[TÀI LIỆU THAM KHẢO 146](#_bookmark96)

### HƢỚNG DẪN

##### MÔ TẢ MÔN HỌC

Môn học nhằm cung cấp cho sinh viên các khái niệm về Hệ quản trị cơ sở dữ liệu (HQTCSDL), vai trò và chức năng của HQTCSDL, phân loại HQTCSDL, các thành phần của một HQTCSDL. Sử dụng được một HQTCSDL (cụ thể là MS SQL Server) để quản trị một hệ cơ sở dữ liệu (CSDL) như: cài đặt hệ CSDL trên máy chủ, tạo kết nối từ các máy người dùng, thiết kế, phát triển, quản trị, sử dụng và tạo mới các đối tượng do HQTCSDL hỗ trợ, quản lý giao tác, quản lý tài khoản người dùng, phân quyền, sao lưu và phục hồi hệ thống. Bước đầu làm quen với lập trình CSDL. Học xong môn học này, học viên có thể nắm vững các nguyên lý hoạt động của một hệ quản trị CSDL, và có thể thao tác thành thạo các chức năng cơ bản, thường dùng của phần mềm quản trị CSDL MS SQL Server.

Khối kiến thức của môn học này đóng vai trò quan trọng giúp cho học viên sau khi ra trường có thể làm việc với vai trò của người quản trị hệ thống server trong các công ty, xí nghiệp... có sử dụng Database Server.

##### NỘI DUNG MÔN HỌC

* Bài 1: Tổng quan về Hệ quản trị CSDL: Bài này cung cấp cho học viên các khái niệm về HQTCSDL, vai trò và chức năng của HQTCSDL, phân loại HQTCSDL, sử dụng một HQTCSDL cụ thể (Microsoft SQL Server) để quản trị một hệ CSDL như cài đặt, đăng nhập, làm quen với một số thành phần. Ôn tập các kiểu dữ liệu của SQL mà học viên đã học trong môn CSDL, một số thao tác như: tạo mới, hiệu chỉnh, xóa Database; tạo mới, chỉnh sửa, xóa Table; cách thức tạo Diagram, và các kiến thức về sao lưu, phục hồi dữ liệu... từ đó có thể vận hành, quản lý CSDL.
* Bài 2: Index: Bài này giới thiệu cho học viên khái niệm và cách sử dụng Index trong SQL Server. Index cung cấp một phương pháp giúp chúng ta nhanh chóng tìm kiếm dữ liệu dựa trên các giá trị trong các cột. Nếu ta tạo ra một Index trên cột khóa chính và sau đó tìm kiếm một dòng dữ liệu dựa trên một trong các giá trị của cột này, đầu tiên SQL Server sẽ tìm giá trị này trong Index, sau đó nó sử dụng

Index để nhanh chóng xác định vị trí của dòng dữ liệu bạn cần tìm. Nếu không có Index, SQL Server sẽ thực hiện động tác quét qua toàn bộ bảng (table scan) để xác định vị trí dòng cần tìm, mà table scan là một trong những động tác có hại nhất cho hiệu suất của SQL Server.

* + Bài 3: View: Giới thiệu cho học viên khái niệm về View (khung nhìn), View là một bảng tạm thời, có cấu trúc như một bảng, View không lưu trữ dữ liệu mà nó được tạo ra khi sử dụng, View là đối tượng thuộc CSDL. View được tạo ra từ câu lệnh truy vấn dữ liệu (lệnh Select), truy vấn từ một hoặc nhiều bảng dữ liệu. View được sử dụng khai thác dữ liệu như một bảng dữ liệu, chia sẻ nhiều người dùng, an toàn trong khai thác, không ảnh hưởng dữ liệu gốc. Có thể thực hiện truy vấn dữ liệu trên cấu trúc của View.
  + Bài 4: Stored Procedure: Bắt đầu từ chương này, học viên làm quen với lập trình trên SQL Server. Trong bài học trước, chúng ta đã tìm hiểu qua View, rõ ràng ưu điểm của View rất nhiều khi quản trị một CSDL. Tuy nhiên, trong thực tế, view gặp bất lợi khi có cùng một công việc nhưng chúng ta phải tạo ra nhiều View (chỉ khác nhau rất ít). Thủ tục lưu trữ (Stored procedure - SP) sẽ giúp chúng ta giải quyết vấn đề này. SP cũng tương tự như thủ tục trong các ngôn ngữ lập trình khác, nó có thể: Chấp nhận các thông số đầu vào và trả về nhiều giá trị trong các hình thức của các thông số đầu ra, chứa các câu lệnh thực hiện các công việc trong CSDL, bao gồm cả việc gọi các thủ tục khác, trả về giá trị thông báo trạng thái việc gọi một thủ tục hay khối lệnh (batch) thành công hay thất bại (và lý do của sự thất bại). Ta có thể sử dụng câu lệnh EXECUTE để thi hành một SP. SP khác với hàm (function) ở chỗ nó không trả về giá trị thay cho tên của nó và nó không thể được sử dụng trực tiếp trong một biểu thức.
  + Bài 5: Function: Hầu hết các ngôn ngữ lập trình, hay CSDL lớn, luôn luôn có một phần mở rộng cho phép người dùng tự định nghĩa một số quy tắc, hàm hoặc thủ tục. User-defined function giống như Stored procedure của SQL server. Hàm người dùng cũng có thể truyền tham số nhưng không được mang thuộc tính OUTPUT, thay vào đó chúng ta dùng câu lệnh RETURN. Hàm cho phép nhận tham số vào, gọi hàm bằng lệnh EXECUTE giống như gọi SP (đối với hàm trả về kiểu vô hướng - Scalar Function).
* Bài 6: Trigger: Trigger là một Stored procedure đặc biệt tự động được chạy mỗi khi có một hành động nào đó xảy ra có liên quan đến nó. Các hành động xảy ra giúp TRIGGER hoạt động: Insert, Delete, Update. Trigger không được thực thi một cách tường minh nên cần thận trọng khi dùng Trigger. Trigger hoạt động gần giống các hàm bắt sự kiện trong javascript. Nó chỉ được kích hoạt khi xảy ra một hành động mà cụ thể là các thao tác Insert, Delete, Update trên một table. Nó hoạt động một các thụ động nên chúng ta không thể biết trigger thực thi khi nào.
* Bài 7: Kiểu dữ liệu Cursor: Phần lớn các CSDL quan hệ thường làm việc trên dữ liệu của nhiều dòng mẫu tin, còn gọi là một bộ các mẫu tin. Ví dụ lệnh SELECT kết quả luôn trả về nhiều dòng dữ liệu hơn là một dòng dữ liệu. Ngược lại đối với một số ngôn ngữ lập trình hoặc bên trong các ứng dụng thì người lập trình vẫn còn các thói quen xử lý và tính toán dữ liệu trên từng dòng riêng lẻ. Để đáp ứng được yêu cầu này của các người lập trình, muốn làm việc chỉ trên từng dòng dữ liệu tại thời điểm hiện hành, Microsoft SQL Server tạo ra một kiểu dữ liệu đó chính là kiểu cursor (kiểu con trỏ).
* Bài 8: Quản lý giao tác: Học viên học xong bài này có thể nắm được khái niệm về giao tác (transaction), các tính chất một giao tác cần phải có để sự hoạt động của nó đảm bảo cho CSDL luôn ở trạng thái nhất quán. Hiểu được các khái niệm về khóa, các phương thức khóa cơ bản, các vấn đề liên quan đến truy xuất đồng thời..., nắm được các mức cô lập cho giao tác.
* Bài 9: Bảo mật và phân quyền sử dụng: Bảo mật và quản trị nhằm kiểm soát ai truy cập và truy cập dữ liệu gì. Người quản trị phải làm cho dễ dàng và thuận lợi để người dùng chỉ truy xuất được phần dữ liệu mà họ được phép, đồng thời phải ngăn chặn được những người không được phép truy xuất vào dữ liệu hoặc cố tình phá hoại dữ liệu. Bài học này giúp học viên nắm được cách thức tạo user, group và các bước phân quyền cụ thể trên CSDL...
* Bài 10: Mã hóa dữ liệu trong SQL: Sau khi học xong bài này, học viên có thể nắm được lý do cần phải mã hóa thông tin, hiểu được các kỹ thuật mã hóa của SQL Server hiện đang được sử dụng. Vận dụng được một kỹ thuật mã hóa và giải mã cụ thể: mã hóa bằng mật khẩu. Với kiến thức này, học viên có thể ứng dụng vào việc bảo mật thông tin nhằm đảm bảo an toàn dữ liệu đối với một CSDL cụ thể.

Mỗi bài học sẽ có các ví dụ minh họa và bài tập đề nghị, học viên cần thực hiện trên lớp và trong giờ thực hành.

##### KIẾN THỨC TIỀN ĐỀ

Điều kiện tiên quyết: CSDL.

Sinh viên đã có khả năng lập trình bằng một số ngôn ngữ như: C++, Visual Basic, VB.NET, C#.NET…

##### YÊU CẦU MÔN HỌC

Người học phải dự học đầy đủ các buổi lên lớp và thực hiện các bài tập trong giờ thực hành.

##### CÁCH TIẾP NHẬN NỘI DUNG MÔN HỌC

Vào buổi học đầu tiên, giảng viên sẽ giới thiệu tài liệu liên quan đến môn học, cung cấp phần mềm cho học viên cùng bài tập thực hành, hướng dẫn sơ bộ cách cài đặt phần mềm trên máy tính.

Để học tốt môn này, học viên cần ôn tập môn học CSDL đã học từ trước, xem qua trước bài mới và tìm hiểu thêm các thông tin liên quan đến bài học.

Đối với mỗi bài học, học viên đọc trước mục tiêu và tóm tắt, sau đó đọc nội dung. Kết thúc mỗi ý của bài học, cần trả lời các câu hỏi ôn tập, để củng cố kiến thức, học viên làm các bài tập cho sẵn.

##### PHƢƠNG PHÁP ĐÁNH GIÁ MÔN HỌC

Môn học được đánh giá gồm:

* Điểm quá trình: 30%. Học viên làm bài kiểm tra trên máy (vào buổi cuối cùng của buổi học thực hành). Giảng viên dạy thực hành ra đề thi, tổ chức thi, chấm thi và công bố điểm tại chỗ.
* Điểm thi: 70%. Học viên dự thi kết thúc môn học theo hình thức trắc nghiệm, thời lượng 60 phút, nội dung trải đều trong tất cả các bài đã học. Từ phương án ra đề

của giảng viên giảng dạy lý thuyết, Phòng Khảo thí sẽ rút trích ngẫu nhiên số lượng câu hỏi theo chỉ định từ ngân hàng câu hỏi của nhà trường.

# BÀI 1: TỔNG QUAN VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Sau khi học xong bài này, học viên có thể:

* + *Nắm được tổng quan về một Hệ quản trị CSDL, cấu trúc các thành phần hệ thống của CSDL và nguyên lý hoạt động của nó.*
  + *Với Hệ quản trị CSDL cụ thể MS SQL Server 2005 (hay 2008/2012), học viên biết cách cài đặt, sử dụng các đối tượng một cách cơ bản nhất.*
  1. **TỔNG QUAN**

Một hệ quản trị CSDL (Database Management System – DBMS) là một phần mềm cho phép tạo ra, định nghĩa và thao tác với CSDL. Trên thực tế DBMS là một công cụ được sử dụng để thực hiện bất kỳ loại hoạt động trên dữ liệu trong CSDL. DBMS cũng cung cấp giải pháp bảo vệ và an toàn cho CSDL, nó duy trì tính nhất quán dữ liệu trong trường hợp của nhiều người dùng. Hiện nay có nhiều hệ quản trị CSDL phổ biến như: MS SQL Server, MySQL, Oracle, Sysbase, Microsoft Access, DB2, v.v...

Các thành phần của hệ thống CSDL:

Users

Database

Application

Database

Database Management System (DBMS)

**Hình 1.1: Các thành phần hệ thống**

* **Users:** Người dùng, bao gồm nhà quản trị, phát triển hệ thống và người sử dụng cuối.
* **Database application:** Ứng dụng CSDL, có thể là cá nhân, phòng ban, doanh nghiệp và nội bộ.
* **DBMS:** Phần mềm cho phép người dùng xác định, tạo và quản lý truy cập CSDL, ví dụ: MS SQL Server, MySQL, Access, Oracle, DB2...
* **Database:** Tập dữ liệu logic được lưu trữ trên máy tính.

Chức năng của DBMS:

* Cung cấp dữ liệu độc lập.
* Kiểm soát đồng thời.
* Cung cấp dịch vụ phục hồi hệ thống.
* Cung cấp các dịch vụ tiện ích.
  + Cung cấp một cái nhìn rõ ràng và hợp lý của quá trình thao tác dữ liệu.

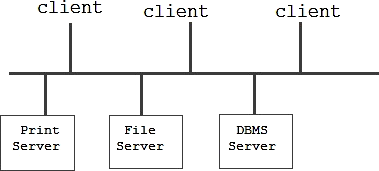
|  |  |
| --- | --- |
| **Ƣu điểm của DBMS:**   * Phân biệt chương trình ứng dụng. * Tối thiểu trùng lắp dữ liệu. * Thu hồi dễ dàng dữ liệu. * Giảm thời gian phát triển và bảo trì khi cần thiết. | **Nhƣợc điểm của DBMS:**   * Phức tạp. * Tốn kém. * Kích thước lớn. |

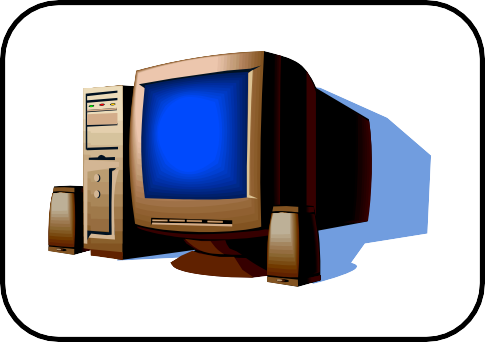
Kiến trúc CSDL:

Chia thành hai loại:

* + Kiến trúc 2 tầng.
  + Kiến trúc 3 tầng.

**Kiến trúc 2 tầng:**



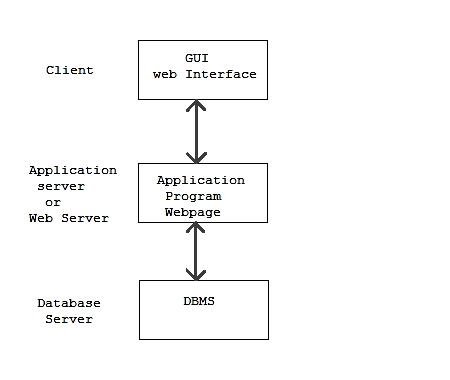
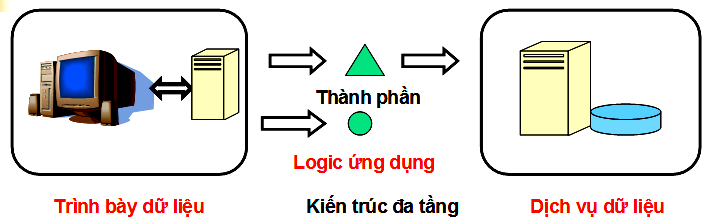
**2-Tier Model**

**Trình bày dữ liệu Dịch vụ dữ liệu**

Hình 1.2: Kiến trúc 2 tầng

* Dữ liệu chia sẻ giữa máy chủ trung tâm và máy client.
* Máy khách gửi yêu cầu đến máy chủ, máy chủ xử lý yêu cầu và trả kết quả cho máy khách.
* Kiến trúc này còn được gọi là kiến trúc client-server.

Kiến trúc 3 tầng:



**Hình 1.3: Kiến trúc 3 tầng**

* Được gọi là kiến trúc ứng dụng
* Kiến trúc ứng dụng chia ứng dụng thành 3 nhóm dịch vụ:
* Tầng dịch vụ người dùng được gọi là tầng trình bày, trình bày dữ liệu cho người dùng.
  + Tầng dịch vụ tác nghiệp được gọi là tầng logic ứng dụng. Đó là tầng trung gian giữa giao diện và CSDL. Chức năng: xử lý kiểm tra dữ liệu hợp lệ, logic truy cập dữ liệu.
  + Tầng dịch vụ tương tác trực tiếp với dữ liệu nguồn như: thêm mới, sửa, xóa, rút trích.

Các mô hình CSDL:

Một mô hình CSDL xác định thiết kế hợp lý của dữ liệu. Mô hình mô tả mối quan hệ giữa các phần khác nhau của dữ liệu. Hiện có nhiều loại mô hình dữ liệu, có thể nêu ra các loại mô hình dữ liệu sau đây là các mô hình dữ liệu cơ bản đang được sử dụng nhiều:

* + Mô hình phân cấp (Hierarchical model): Mô hình dữ liệu là một cây, trong đó các nút biểu diễn các tập thực thể, giữa các nút con và nút cha được liên hệ theo một mối quan hệ xác định.
  + Mô hình lưới (Network model): Mô hình dữ liệu được biểu diễn là một đồ thị có hướng.
  + Mô hình quan hệ (Relational model): Mô hình này dự trên cơ sở khái niệm lý thuyết tập hợp của các quan hệ, tức là tập các k-bộ với k cố định.
  + Mô hình thực thể kết hợp (Entity Relationship model)
  + Mô hình hướng đối tượng (Object Oriented model): Là các mô hình có hỗ trợ:
    - Đặc tính nhận dạng đối tượng
    - Các đối tượng phức
    - Phân cấp theo kiểu
  + Mô hình cơ sở tri thức: Thường được sử dụng trong các hệ chuyên gia.
  + Mô hình CSDL Web.

#### GIỚI THIỆU MICROSOFT SQL SERVER

Trong môn học này, cuốn giáo trình hướng dẫn tổng thể trên MS Server SQL 2005 (cho phù hợp với phòng máy thực hành của nhà trường). Hiện tại, các công ty, xí nghiệp bên ngoài đã và đang sử dụng MS SQL Server 2008/2012, tuy nhiên, với kiến thức đã học, học viên có thể học hỏi thêm và vận hành tốt với các phiên bản sau của MS SQL Server.

SQL Server 2005 hỗ trợ các loại CSDL dưới đây:

CSDL người dùng

CSDL hệ thống

|  |
| --- |
| * Tạo ra bởi SQL Server để lưu trữ thông tin về SQL Server * Để quản lý CSDL người dùng |
| * Do người dùng tạo ra * Lưu trữ dữ liệu người dùng |
| * SQL Server phân phối kèm theo một số CSDL mẫu * AdventureWorks là CSDL mẫu mới được giới thiệu trong SQL Server 2005. |

SQL Server 2005 hỗ trợ một số tính năng sau:

CSDL mẫu

* Dễ cài đặt
* Tích hợp với Internet
* Khả cỡ và khả dụng
* Kiến trúc mô hình Client/Server
* Tương thích hệ điều hành
* Nhà kho dữ liệu
* Tương thích chuẩn ANSI, SQL-92
* Nhân bản dữ liệu
* Tìm kiếm Full-Text
* Sách trực tuyến

Tính năng căn bản

* + Dịch vụ thông báo
  + Dịch vụ báo cáo
  + Dịch vụ môi giới
  + Cải tiến của Database Engine

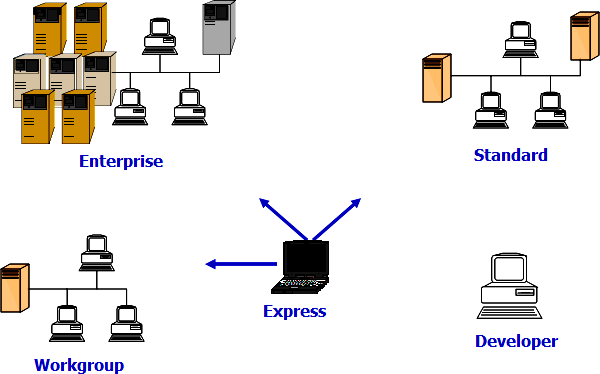
Ghi chú:

Tính năng nâng cao

Dịch vụ thông báo là một nền tảng cho phép phát triển ứng dụng gửi nhận thông báo với tính khả cỡ cao.

Cơ chế xử lý CSDL của SQL Server 2005 được bổ sung tính năng mới cũng như nâng cao những khả năng về lập trình như: bổ sung kiểu dữ liệu mới, kiểu dữ liệu XML và một số kiểu dữ liệu khác.

Phiên bản của SQL Server 2005:



**Hình 1.4: Các phiên bản của SQL**

* + Enterprise Edition:
    - Hỗ trợ: 32-bit and 64-bit
    - Không hạn chế kích thước CSDL
    - Hỗ trợ Xử lý giao tác trực tuyến (OLTP)
    - Khả dụng và khả cỡ cao
* Standard Edition:
* Đủ cho các công ty vừa và nhỏ
* Gồm các tính năng cơ bản như: thương mại điện tử, nhà kho dữ liệu, giải pháp ứng dụng doanh nghiệp
* Workgroup Edition:
* Dòng sản phẩm cho nhóm làm việc
* Cho các ứng dụng và hệ thống của các tổ chức nhỏ
* Không hạn chế kích thước CSDL và số lượng người dùng
* Developer Edition
* Có các chức năng để xây dựng và kiểm thử ứng dụng trên phiên bản SQL Server Expression
* Express Edition
* Phiên bản nhỏ gọn có thể download từ Internet
* Chỉ có phần dịch vụ CSDL, không hỗ trợ những công cụ quản lý

Ƣu điểm của SQL Server 2005:

* Nâng cao quản lý dữ liệu doanh nghiệp
* Nâng cao hiệu suất cho người lập trình
* Hỗ trợ tốt hệ thống phân tích, hỗ trợ ra quyết định
* Các tính năng có tính cạnh tranh

Các thành phần của SQL Server 2005:

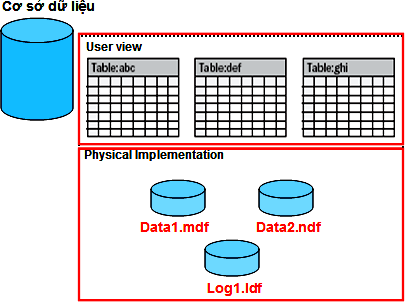
* Dịch vụ CSDL quan hệ
* Dịch vụ phân tích
* Dịch vụ chuyển đổi dữ liệu - DTS
* Dịch vụ thông báo
* Hỗ trợ dịch vụ HTTP
  + Tích hợp .NET CLR
  + Dịch vụ báo cáo
  + Dịch vụ môi giới
  + SQL Server Agent
  + Nhân bản
  + Tìm kiếm Full-Text (toàn văn)

CSDL hệ thống (System Database):

* + Trong SQL Server 2005, tất cả thông tin hệ thống được lưu trong CSDL hệ thống
  + Tất cả CSDL người dùng được quản lý bởi CSDL hệ thống
  + SQL Server 2005 cung cấp và hỗ trợ 4 CSDL hệ thống sau:

|  |  |
| --- | --- |
| **Database** | **Mô tả** |
| master | Chứa tất cả những thông tin cấp hệ thống (system-level information) bao gồm thông tin về các database khác trong hệ thống như vị trí của các data files, các login account và các thiết đặt cấu hình hệ thống của SQL Server (system configuration settings). |
| msdb | Database này được SQL Server Agent sử dụng để hoạch định các báo động và các công việc cần làm (schedule alerts and jobs). |
| model | Database này đóng vai trò như một bảng kẻm (template) cho các database khác. Nghĩa là khi một user database được tạo ra thì SQL Server sẽ copy toàn bộ các system objects (tables, stored procedures...) từ Model database sang database mới vừa tạo. |
| tempdb | Chứa tất cả những table hay stored procedure được tạm thời tạo ra trong quá trình làm việc bởi user hay do bản thân SQL Server engine. Các table hay stored procedure này sẽ biến mất khi khởi động lại SQL Server hay khi ta disconnect. |
| **Bảng 1.1: CSDL hệ thống của SQL Server 2005** | |

Cấu trúc vật lý của một SQL Server Database:



Hình 1.5: Cấu trúc vật lý của một SQL Server

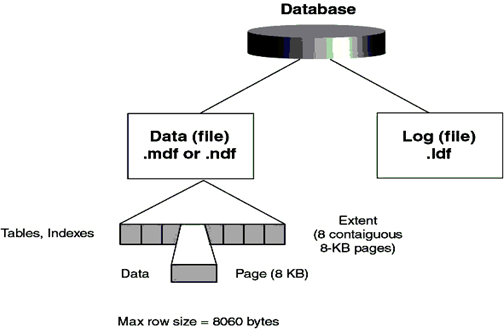
Mỗi một database trong SQL Server đều chứa ít nhất một data file chính (Primary), có thể có thêm một hay nhiều data file phụ (Secondary) và một transaction log file (tập tin ghi nhận giao tác).

Primary data file (có phần mở rộng **.mdf**): đây là file chính chứa data và những system tables.

Secondary data file (có phần mở rộng **.ndf**): đây là file phụ thường chỉ sử dụng khi database được phân chia để chứa trên nhiều đĩa.

Transaction log file (có phần mở rộng **.ldf**): đây là file ghi lại tất cả những thay đổi diễn ra trong một database và chứa đầy đủ thông tin để có thể roll back hay roll forward khi cần.

Data trong SQL Server được chứa thành từng **Page** 8KB và 8 page liên tục tạo thành một **Extent** như hình vẽ dưới đây:



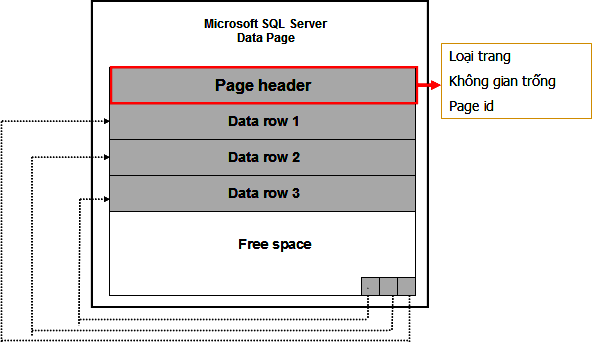
Hình 1.6: Pages và Extents

Trước khi SQL Server muốn lưu data vào một table nó cần phải dành riêng một khoảng trống trong data file cho table đó. Những khoảng trống đó chính là các Extents. Có 2 loại Extents:

* + **Mixed Extents** (loại hỗn hợp) dùng để chứa data của nhiều tables trong cùng một Extent
  + **Uniform Extent** (loại thuần nhất) dùng để chứa data của một table. Ðầu tiên SQL Server dành các Page trong Mixed Extent để chứa data cho một table sau đó khi data tăng trưởng thì SQL dành hẳn một Uniform Extent cho table đó.

Ghi chú thêm về Pages and Extents:

* + Đơn vị lưu trữ dữ liệu cơ sở trong SQL Server là page. Không gian đĩa được cấp phát cho tập tin dữ liệu (.mdf hoặc .ndf) của một CSDL được phân chia logic thành các page và được đánh số liên tục từ 0 đến n. SQL Server thực hiện đọc ghi dữ liệu trên các page này.
  + Mỗi page có kích thước là 8 Kbs (1MB có 128 pages). Trong đó 96 byte đầu tiên của page (page header) lưu trữ các thông tin hệ thống về page như: số trang, loại trang (data, Index, text/image...), chỉ số ID của trang.
* Một Extent là 8 page liên tiếp, tức là 64 KB. Có 16 Extent trên mỗi MB.



Hình 1.7: Minh họa thêm về Pages and Extents

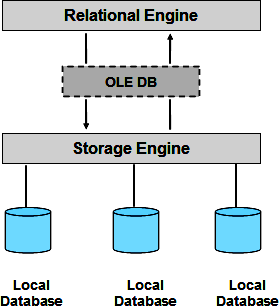
* Database Files (các tập tin CSDL) and Filegroups:

(Khái niệm filegroup tương tự như folder trong Windows, là một tên logic để nhóm các data file chứ không bản thân nó chứa dữ liệu. Filegroup chứa một hoặc nhiều data file và các data file này mới thực sự chứa dữ liệu. Khi mới tạo database, một filegroup có tên PRIMARY tự động được tạo và chứa data file chính (.mdf). Tuy nhiên, ta không nên dùng PRIMARY filegroup vào việc phân đoạn mà nên tạo filegroup mới)

* Space Allocation and Reuse (không gian phân bổ và dùng lại):
* SQL Server phân bổ vùng nhớ cho các đối tượng và lấy lại vùng nhớ khi các hàng, đối tượng bị xóa.
* Table and Index Architecture (kiến trúc chỉ mục):
* Mỗi bảng và chỉ mục, các hàng dữ liệu được lưu trữ trên 8KB gọi là page.
* Transaction Log Architecture (kiến trúc ghi nhận giao dịch):
* Mỗi CSDL trong SQL Server đều bao gồm file log để ghi lại tất cả các giao tác và thao tác CSDL. Các thao tác hỗ trợ bởi giao tác gồm:
  + Khôi phục lại từng giao tác
  + Khôi phục lại tất cả các giao tác chưa hoàn tất khi SQL Server khởi động
  + Khôi phục dữ liệu về trước vị trí mà hệ thống bị lỗi

Kiến trúc CSDL quan hệ:

Relational database engine có hai phần chính: relational engine và storage engine.



Hình 1.8: Kiến trúc CSDL

* + Query Processor Architecture (kiến trúc bộ xử lý truy vấn):
    - Các câu lệnh gửi từ ứng dụng người dùng đến SQL Server là các câu lệnh SQL.
    - Các câu lệnh SQL được SQL Server thực hiện theo một trong những cách sau:
      * Xử lý từng câu lệnh
      * Xử lý lô/tập lệnh
      * Thực thi Stored Procedure và Trigger
      * Kế hoạch thực thi và dùng lại nơi lưu trữ
      * Xử lý truy vấn song song
* Memory Architecture (kiến trúc bộ nhớ):
* Microsoft SQL Server 2005 cấp phát vùng nhớ động khi cần, giảm đọc/ghi đĩa.
* Thread and Task Architecture (luồng và kiến trúc tác vụ):
* Thread là tính năng của hệ điều hành hỗ trợ để xử lý nhiều công việc đồng thời.
* Mỗi process có một thread thực hiện riêng. Ứng dụng có thể làm được điều này bằng cách khởi động một process cho một công việc
* Input/Output (I/O) Architecture:
* Việc đọc/ghi đĩa tốn nhiều tài nguyên và mất nhiều thời gian để giải phóng tài nguyên.
* SQL Server 2005 phân bố không gian bộ nhớ đệm lớn để lưu trữ tạm và tránh việc đọc ghi đĩa thường.
* Distributed Query Architecture (kiến trúc phân tán truy vấn):
* Truy vấn phân tán có thể truy cập dữ liệu hỗn tạp trên nhiều nguồn dữ liệu trên cùng hoặc các máy khác nhau.
* SQL Server 2005 hỗ trợ truy vấn phân tán bằng việc sử dụng Microsoft OLE DB Provider.

Các đối tƣợng của CSDL:

|  |  |
| --- | --- |
| **Đối tƣợng** | **Mô tả** |
| Table | Bảng |
| Data Type | Kiểu dữ liệu |
| View | Bảng ảo |
| Stored procedure | Thủ tục nội tại |
| Function | Hàm |
| Index | Chỉ mục - đối tượng CSDL nhằm truy cập dữ liệu nhanh hơn |
| Constraint | Ràng buộc dữ liệu, được định nghĩa trên một cột hoặc nhiều cột dữ liệu để đảm bảo toàn vẹn dữ liệu |
| Trigger | Là loại thủ tục lưu trữ đặc biệt, được thực thi khi dữ liệu trong bảng thay đổi |
| Logins | Một người dùng được định danh bởi Login ID để kết nối đến SQL Server. SQL Server hỗ trợ 2 cơ chế chứng thực đó là Windows Authentication và SQL Server Authentication |
| Users | Nhận diện mỗi người dùng trong mỗi CSDL. Quyền truy cập của người dùng dựa trên đối tượng này |
| Roles | Vai trò, là nhóm người dùng cùng chức năng |
| Groups | Nhóm nhiều SQL Server lại thành một Groups |
| **Bảng 1.2: Các đối tƣợng của CSDL** | |

**Cài đặt MS SQL Server 2005/2008:**

Phần này, giảng viên sẽ hướng dẫn cài đặt chi tiết trong giờ học lý thuyết hay thực hành.

#### TRANSACT-SQL NÂNG CAO

###### Khai báo và sử dụng biến:

Biến cục bộ:

Là một đối tượng có thể chứa giá trị thuộc một kiểu dữ liệu nhất định.

* + Tên biến: Bắt đầu bằng một ký tự @
  + Tầm vực của biến: Biến cục bộ có giá trị trong một query batch hoặc trong một stored procedure/function.
* Khai báo biến cục bộ: Dùng lệnh Declare, cung cấp tên biến và kiểu dữ liệu: Declare Tên\_biến Kiểu\_dữ\_liệu.

Ví dụ:

Declare @MaSinhVien varchar(10) Declare @HoTen nvarchar(50) Declare @Sum float, @Count int

* Gán giá trị cho biến cục bộ: Dùng lệnh set để gán giá trị cho biến. Giá trị gán cho biến phải phù hợp với kiểu dữ liệu của biến:

Set tên\_biến = giá\_trị Set tên\_biến = tên\_biến Set tên\_biến = biểu\_thức

Set tên\_biến = kết\_quả\_truy\_vấn Ví dụ:

Set @MaLop = ‘09DTH01’

Set @SoSV = (select count (\*) from SINHVIEN) Set @MaLop = N‘Thân chào lớp ’ + @Malop

* Đưa kết quả truy vấn vào biến: Ví dụ :

Cho table SV(MSSV: int; HoTen: nvarchar(50), Tuoi int).

Select @Var1 = HoTen, @Var2 = Tuoi from SV where MaSV = 1

**Lƣu ý:** nếu câu truy vấn trả về nhiều dòng, các biến chỉ nhận giá trị tương ứng của dòng đầu tiên.

Biến toàn cục:

Là các biến hệ thống do SQL Server cung cấp.

* Tên biến bắt đầu bằng @@ (2 ký tự @). SQL tự cập nhật giá trị cho các biến này, người dùng không thể gán giá trị trực tiếp.
  + Bản chất là 1 hàm (function)

Một số biến hệ thống thường dùng:

* + @@error
  + @@rowcount
  + @@trancount
  + @@fetch\_status

###### Các lệnh điều khiển:

If…else

Cú pháp:

If *biểu\_thức\_điều kiện*

Lệnh | Khối\_lệnh

[Else Lệnh | Khối\_lệnh]

Khối lệnh là một hoặc nhiều lệnh nằm trong cặp từ khóa begin…end.

While

Cú pháp:

While *biểu\_thức\_điều\_kiện*

Lệnh| Khối lệnh

Có thể sử dụng Break và Continue trong khối lệnh của While:

* + Break: thoát khỏi vòng while hiện hành.
  + Continue: trở lại đầu vòng while, bỏ qua các lệnh sau đó.

Case

Kiểm tra một dãy các điều kiện và trả về kết quả phù hợp với điều kiện đúng. Có thể được sử dụng như một hàm trong câu lệnh Select.

Cú pháp: Có hai dạng.

* + Dạng 1 (simple case):

Case *Biểu\_thức\_đầu\_vào* When *Giá\_trị* then *Kết\_quả* [...n]

[Else *Kết\_quả\_khác*] End

* Dạng 2 (searched case):

Case

When *Biểu\_thức\_điều kiện* then *Kết\_quả*

[...n]

[Else *Kết\_quả\_khác*] End

### TÓM TẮT

*SQL Server là hệ quản trị CSDL quan hệ và cũng đóng vai trò là nền tảng dữ liệu của doanh nghiệp.*

*Một số tính năng cải tiến của SQL Server 2005 là: dịch vụ thông báo, dịch vụ môi giới, dịch vụ báo cáo, và bộ máy xử lý CSDL.*

*Dịch vụ thông báo là một thành phần mới của SQL Server 2005, thông báo người dùng về bất kỳ sự kiện nào xảy ra trên CSDL.*

*Tầng dịch vụ dữ liệu trong kiến trúc 3 tầng tương tác trực tiếp với dữ liệu. Bốn chức năng cơ bản của dịch vụ này là: rút trích, thêm, sửa, xóa.*

*SQL Server 2005 có hai loại CSDL: CSDL hệ thống và CSDL người dùng. CSDL hệ thống lưu trữ thông tin để vận hành và quản lý SQL Server. CSDL người dùng do người dùng tạo ra để lưu trữ dữ liệu.*

*Người quản trị CSDL có thể sao lưu hoặc khôi phục từng file riêng lẻ thay vì toàn bộ CSDL. Việc sao lưu hoặc khôi phục từng file rất hữu ích khi CSDL quá lớn.*

*Cơ chế chứng thực người dùng của SQL Server có 2 chế độ: SQL Server authentication, Windows NT authentication.*

*Bộ máy CSDL của SQL Server 2005 gồm các thành phần chính:*

* + *Kiến trúc bộ xử lý truy vấn*
  + *Kiến trúc bộ nhớ*
  + *Kiến trúc luồng và công việc*
  + *Kiến trúc vào ra*
  + *Kiến trúc truy vấn phân tán*

### CÂU HỎI ÔN TẬP

**Câu 1:** Hãy nêu khái niệm về một Hệ quản trị CSDL?

**Câu 2:** Hãy nêu cấu trúc vật lý của một Database?

**Câu 3:** Cho biết các đối tượng của CSDL?

**Câu 4:** Hãy thực hiện tạo Database của bài thực hành số 1, sau đó tiến hành Detach/Attach, Back up/Restore Database:

1. Bằng lệnh T-SQL.
2. Bằng công cụ SQL Server Management Studio.

## BÀI 2: INDEX (CHỈ MỤC)

Sau khi học xong bài này, học viên có thể:

*Nắm được khái niệm về khóa Index (chỉ mục), là một khóa quan trọng đối với CSDL, đặc biệt là CSDL lớn. Index được thiết lập từ một hoặc nhiều cột dữ liệu của bảng dữ liệu. Các giá trị của khóa Index sẽ được sắp xếp và lưu trữ theo một danh sách (bảng khác). Mỗi giá trị trong khóa Index là duy nhất trong danh sách, mỗi giá trị khóa Index sẽ liên kết đến giá trị trong bảng dữ liệu (liên kết dạng con trỏ). Việc lưu trữ dữ liệu của bảng có khóa Index được thực hiện theo cấu trúc cấy B-Tree nhằm tăng tốc độ truy xuất dữ liệu đối với ổ đĩa (thiết bị thứ cấp).*

#### KHÁI NIỆM

Index (chỉ mục) thực chất là một dạng tương tự như phần mục lục của một cuốn sách hay của một cuốn từ điển, các giá trị trong bảng được sắp theo cột, các giá trị trong cột đó sẽ được sắp xếp theo một trật tự dựa vào dữ liệu của bảng đó, giúp cho việc truy xuất trở nên nhanh hơn. Một chỉ mục trên một thuộc tính A của bảng T sẽ giúp tăng tốc quá trình tìm kiếm các giá trị cố định trên thuộc tính A.

LỢI ÍCH CỦA INDEX

* + Là 1 trong những cách tốt nhất để giảm hoạt động đọc/ghi của đĩa cứng.
  + Cho phép SQL Server tìm thấy dữ liệu mà không cần phải duyệt qua toàn bộ table.

NHƢỢC ĐIỂM CỦA INDEX

Việc tạo Index đôi khi rất tốn thời gian. Đối với dự án có CSDL nhỏ, vừa thì không đáng kể nhưng với CSDL lớn thì việc này rất tốn thời gian.

Do index là tạo 1 cấu trúc bảng, nên dẫn đến tốn tài nguyên. Nếu có sự thay đổi về dữ liệu thì index cũng được update theo.

Trong thực tế, hầu hết các trường hợp, việc sử dụng index vẫn là lựa chọn hàng đầu.

#### PHÂN LOẠI

Có 2 loại Index chính thường được dùng là Clustered Index (chỉ mục kết cụm) và Nonclustered Index (chỉ mục không kết cụm).

**Clustered Index**: Mặc định khi tạo khóa chính cho 1 table nào đó tức là ta đã tạo 1 Clustered Index. Một bảng mà không có Clustered Index được gọi là 1 Heap table.

Trong clustered index, các dữ liệu ở cấu trúc bảng trong được sắp xếp một cách vật

lý, tức là dữ liệu bảng trong được sắp xếp đúng theo thư mục cây dựa vào bảng chữ cái.

**Non-Clustered Index**: không sắp xếp dữ liệu theo một trật tự vật lý như clustered mà là "loạn xà ngầu" trong bảng thông tin, miễn sao nó nằm trong một logic do index qui định. Có thể có các biến thể như Covering Index (chỉ mục bao phủ), Filtered Index (chỉ mục lọc)… Ngoài ra còn có các loại Index khác như Full Text Index (chỉ mục toàn văn), Spatial Index (chỉ mục không gian), XML Index (chỉ mục XML).

#### TẠO INDEX

**CREATE INDEX** <Tên\_index>

**ON** <Tên\_bảng> (Cột1 [ASC|DESC], ..., n)

Ví dụ:

CREATE INDEX ID\_PHG ON NHANVIEN (PHG)

--

CREATE INDEX ID\_PHANCONG

ON PHANCONG (SODA, MA\_NVIEN DESC)

#### XÓA INDEX

**DROP INDEX** <Tên\_index> **ON** <Tên\_bảng>

**Ví dụ:**

DROP INDEX ID\_PHG ON NHANVIEN DROP INDEX ID\_PHANCONG ON PHANCONG

### TÓM TẮT

*Trong bài này, học viên làm quen với các khái niệm về Index. Sử dụng Index là một cách rất hữu hiệu để tăng hiệu năng thực hiện các câu lệnh SQL. Index trong SQL cũng giống như mục lục của những quyển sách, giúp SQL Server xác định chính xác nơi dữ liệu được lưu trữ. Index là 1 trong những cách tốt nhất để giảm hoạt động của đĩa cứng (disk I/O) và logical reads, cho phép SQL Server tìm thấy dữ liệu mà không cần phải quét toàn bộ Table.*

*Việc tạo Index đôi khi rất tốn thời gian. Đối với dự án có CSDL nhỏ, vừa thì không đáng kể nhưng với CSDL lớn thì việc này là rất mất thời gian. Do index là tạo 1 cấu trúc bảng nên dẫn đến tốn tài nguyên. Nếu có sự thay đổi về dữ liệu thì index cũng được update theo. Tuy nhiên, trong thực tế, trong hầu hết các trường hợp, việc sử dụng index vẫn là lựa chọn hàng đầu.*

*Tóm lại, chúng ta đã tìm hiểu các vấn đề cơ bản của Index trong SQL Server và các vấn đề cần lưu ý khi triển khai Index. Điều đó không có nghĩa là chúng ta đã nắm được đầy đủ về Index trong Server. Có một số loại Index mà chúng ta cần tìm hiểu thêm, như index cho cột có kiểu là XML, Filtered Index, Spatial Index được hỗ trợ trong SQL Server 2008.*

### CÂU HỎI ÔN TẬP

Khái niệm Index? Khi nào dùng Index?

**Câu 1:** Nêu ưu và nhược điểm của Index?

## BÀI 3: VIEW (KHUNG NHÌN)

Sau khi học xong bài này, học viên có thể:

* *Nắm được khái niệm về View (khung nhìn), view là một bảng tạm thời, có cấu trúc như một bảng, view không lưu trữ dữ liệu mà nó được tạo ra khi sử dụng, view là đối tượng thuộc CSDL. View được tạo ra từ câu lệnh truy vấn dữ liệu (lệnh Select), truy vấn từ một hoặc nhiều bảng dữ liệu.*
* *View được sử dụng khai thác dữ liệu như một bảng dữ liệu, chia sẻ nhiều người dùng, an toàn trong khai thác, không ảnh hưởng dữ liệu gốc. Có thể thực hiện truy vấn dữ liệu trên cấu trúc của View.*

#### KHÁI NIỆM

View có thể được coi như là một bảng ảo (virtual table) hay một truy vấn được lưu trữ. Các dữ liệu truy cập thông qua một view không được lưu trữ trong CSDL như là một đối tượng riêng biệt. Dữ liệu được lưu trữ trong CSDL thông qua câu lệnh SELECT. Tập kết quả của câu lệnh SELECT lưu trữ trong bảng ảo thông qua view. Người dùng có thể sử dụng bảng ảo này bằng cách tham chiếu đến tên view thông qua câu lệnh Transact-SQL, cách thực hiện giống như cách tham chiếu đến một table.

View được dùng trong các trường hợp sau:

Hạn chế người dùng truy cập đến các dòng trong một bảng. Ví dụ, chỉ cho phép một nhân viên chỉ thấy các dòng thông tin liên quan đến công việc của mình trong một bảng theo dõi công việc.

Hạn chế người dùng truy cập đến các cột nào đó trong một bảng. Ví dụ, cho phép các nhân viên xem tên, văn phòng, điện thoại nơi làm việc, bộ phận trong một bảng nhân viên, nhưng lại không cho phép họ nhìn thấy bất kỳ cột thông tin về tiền lương hoặc thông tin riêng của cá nhân nào.

Liên kết nhiều cột dữ liệu từ các bảng để tạo ra bảng mới, trông giống như một bảng duy nhất.

Tạo ra thông tin tổng hợp thay vì cung cấp chi tiết. Ví dụ, hiển thị giá trị tổng, giá trị lớn nhất, nhỏ nhất của một cột...

View được tạo ra bằng cách dùng câu lệnh SELECT để lấy các dữ liệu. Các bảng dữ liệu tham chiếu bởi các câu lệnh SELECT được gọi là các bảng cơ sở.

#### TẠO VIEW

CREATE VIEW [< *owner* > **.** ] *view\_name* [ **(** *column* [

**,**...*n* ] **)** ]

[ WITH < view\_attribute > [ **,**...n ] ] AS

Giải thích:

*Owner:* Là tên của người dùng đã sở hữu view..

*view\_name*: Tên của view.

*column:* Là tên cột của view, nếu cột không được chỉ định, các cột sẽ có cùng tên với các cột trong câu lệnh SELECT.

*n:* có thể chỉ ra nhiều cột.

AS: bắt đầu phần thân của view, nơi đây sẽ là câu lệnh select. Trong đó, có thể tham chiếu đến nhiều table và view khác. Trong thân của view không được dùng các từ khóa sau:

* COMPUTE hoặc COMPUTE BY
* Chỉ dùng ORDER BY nếu có phát biểu TOP phía trên câu lệnh SELECT
* INTO
* Tham chiếu một bảng tạm thời hoặc một biến bảng

Một view có thể được tạo ra chỉ trong CSDL hiện tại, có thể tham chiếu tối đa là

1.024 cột.

Cho database QLBH gồm 4 bảng sau:

KHACHHANG(MAKH, TENKH, DCKH, DTKH) HOADON(SOHD, NGAY, MAKH) MATHANG(MAMH, TENMH, QUYCACH, GIAMUA) CTHD(SOHD, MAMH, SL, GIABAN, THANHTIEN)

Trong ví dụ dưới đây, view có tên “Tonghop*”* trong database “QLBH*”* sẽ lấy dữ liệu từ 3 bảng cơ sở để thống kê tổng trị giá của từng hóa đơn:

CREATE VIEW Tonghop AS

SELECT H.SOHD, sum(GIABAN\*SL) as [Tổng trị giá] FROM HOADON H, MATHANG M, CTHD C

WHERE H.SOHD = C.SOHD and M.MAMH = C.MAMH GROUP BY H.SOHD

Ta có thể tham chiếu đến view “Tonghop” giống như cách tham chiếu đến một table:

SELECT \*

FROM Tonghop

Một view có thể tham chiếu đến một view khác, ở DB trên, ta có thể xây dựng view thống kê tổng tiền của từng khách hàng đã mua hàng. Ta xây dựng view có tên “Thongke”, câu lệnh select bên trong sẽ gọi lại view “Tonghop”:

CREATE VIEW Thongke AS

SELECT K.MAKH, TENKH, sum([Tổng trị giá]) as [Tổng tiền] FROM KHACHHANG K, HOADON H, Tonghop T

WHERE K.MAKH = H.MAKH and H.SOHD = T.SOHD GROUP BY K.MAKH, TENKH

#### HIỆU CHỈNH VIEW

ALTER VIEW [< *owner* > **.** ] *view\_name* [ **(** *column* [

**,**...*n* ] **)** ]

[ WITH < view\_attribute > [ **,**...n ] ] AS

*select\_statement*

* 1. **XÓA VIEW**

DROP VIEW [ schema\_name. ] view\_name [ ...,n ] [ ; ]

schema\_name: Là tên của các lược đồ mà view trực thuộc. view\_name: Là tên của view cần xóa.

### TÓM TẮT

*View (khung nhìn/virtual table - bảng ảo) thực chất là một đối tượng mà bên trong nó chỉ lưu trữ duy nhất một câu lệnh SELECT dùng để chỉ định các cột, các dòng dữ liệu bên dưới các bảng dữ liệu mà nó chọn lựa ra để hiển thị cho người sử dụng xem hoặc cập nhật. Với nguyên tắc này, bạn có thể hiển thị ra đúng các thông tin tối thiểu mà người sử dụng cần dùng, không cần thiết phải hiển thị ra tất cả các thông tin hiện đang được lưu trữ bên trong bảng (đáp ứng được tính bảo mật thông tin). Ngoài ra View còn giúp những người sử dụng dễ dàng truy xuất đến các thông tin mà họ đang cần, khi đó đơn giản sẽ thông qua việc thực hiện các truy vấn trực tiếp đến các bảng ảo mà không cần quan tâm các thông tin này đang được lưu trữ trong những bảng dữ liệu nào (đáp ứng được tính dễ sử dụng).*

*Trong thực tế, bạn thường tạo ra các bảng ảo để lưu trữ các thông tin cho các loại báo cáo đơn giản hoặc dữ liệu của các màn hình nhập liệu phức tạo có liên kết dữ liệu với nhiều bảng khác hoặc các màn hình tra cứu thông tin cho các người sử dụng.*

*Lưu ý: bảng ảo hoàn toàn không lưu trữ dữ liệu một cách riêng lẻ. Các dữ liệu được hiển thị trong bảng ảo sẽ được lấy từ bên dưới dữ liệu của các bảng cơ sở trong CSDL hiện hành. Tuy nhiên bạn vẫn có thể cập nhật (thêm, sửa, xóa) dữ liệu trong các bảng ảo như là đang cập nhật dữ liệu trong các bảng cơ sở.*

### CÂU HỎI ÔN TẬP

**Câu 1:** Khái niệm View? Khi nào dùng View? Nêu ưu và nhược điểm của View? Làm các câu hỏi phần View trong bài thực hành số 2 và 3.

## BÀI 4: STORED PROCEDURE (THỦ TỤC NỘI TẠI)

Sau khi học xong bài này, học viên có thể:

* *Nắm được khái niệm cơ bản về chương trình con (Sub-Program) trong ngôn ngữ lập trình, cách xây dựng và vận dụng.*
* *Hiểu rõ ý nghĩa và cách sử dụng thủ tục nội tại (Stored Procedure): cách tạo, sửa và xóa, đặc biệt là cách thức truyền tham số đầu vào và đầu ra.*
* *Phân biệt những điểm giống và khác nhau giữa View và Stored Procedure.*

#### KHÁI NIỆM

Trong bài học trước, chúng ta đã tìm hiểu qua View, rõ ràng ưu điểm của View rất nhiều khi quản trị một CSDL. Tuy nhiên, trong thực tế, view gặp bất lợi khi có cùng một công việc nhưng chúng ta phải tạo ra nhiều View (chỉ khác nhau rất ít). Ví dụ như, công ty muốn đếm xem có bao nhiêu người đi làm vào ngày hôm qua, hôm nay hay một ngày cụ thể trong tháng... Ta phải xây dựng rất nhiều View tương ứng với từng ngày cụ thể. Công việc đếm này chỉ khác nhau về ngày cần thống kê. Thủ tục nội tại (Stored procedure) sẽ giúp chúng ta giải quyết vấn đề này.

Thủ tục nội tại trong Microsoft SQL Server cũng tương tự như thủ tục trong các ngôn ngữ lập trình khác, nó có thể:

* Chấp nhận các thông số đầu vào và trả về nhiều giá trị trong các hình thức của các thông số đầu ra.
* Chứa các câu lệnh thực hiện các công việc trong CSDL, bao gồm cả việc gọi các thủ tục khác.
* Trả về giá trị thông báo trạng thái việc gọi một thủ tục hay khối lệnh (batch) thành công hay thất bại (và lý do của sự thất bại).
* Ta có thể sử dụng câu lệnh EXECUTE để thi hành một thủ tục lưu trữ. Thủ tục lưu trữ khác với hàm (function) ở chỗ nó không trả về giá trị thay cho tên của nó và nó không thể được sử dụng trực tiếp trong một biểu thức.

Những lợi ích của việc sử dụng thủ tục lưu trữ:

* Đã được đăng ký tại các máy chủ.
* Có thể có các thuộc tính bảo mật (chẳng hạn như các quyền) và quyền sở hữu, các chứng thực có thể được gắn kèm với nó.
* Người dùng có thể được cấp phép để thi hành một thủ tục lưu trữ mà không cần phải có quyền truy cập trực tiếp vào các đối tượng tham chiếu trong thủ tục.
* Có thể nâng cao tính bảo mật của ứng dụng.
* Tham số của thủ tục lưu trữ có thể giúp bảo vệ ứng dụng từ các cuộc tấn công SQL Injection. (*Để biết thêm thông tin, xem SQL Injection*).
* Cho phép lập trình mô-đun.
* Ta có thể tạo ra các thủ tục một lần, và gọi nó trong chương trình. Điều này có thể cải thiện khả năng bảo trì của ứng dụng và cho phép ứng dụng truy cập CSDL một cách thống nhất.
* Có thể làm giảm lưu lượng mạng.

Thủ tục lưu trữ có thể bảo vệ người dùng không thể biết các chi tiết của các bảng trong CSDL. Nếu một tập hợp các thủ tục lưu trữ hỗ trợ tất cả các chức năng kinh doanh người dùng cần phải thực hiện, người dùng không bao giờ cần truy cập vào bảng trực tiếp, họ có thể thực hiện các thủ tục được lưu trữ để thi hành các công việc kinh doanh mà họ đã quen thuộc.

#### PHÂN LOẠI SP

Stored Procedure có thể được chia thành 5 nhóm như sau:

1. **System Stored Prcedure**: Là những stored procedure chứa trong Master database và thường bắt đầu bằng tiếp đầu ngữ **sp\_**. Các stored procedure này thuộc loại built-in và chủ yếu dùng trong việc quản lý database (administration) và security.

Ví dụ, bạn có thể kiểm tra tất cả các processes đang được sử dụng bởi user

DomainName\Administrators, cú pháp:

sp\_who @loginame='DomainName\Administrators'.

Có hàng trăm system stored procedure trong SQL Server.

1. **Local Stored Procedure**: Ðây là loại thường dùng nhất. Chúng được chứa trong user database và thường được viết để thực hiện một công việc nào đó. Thông thường người ta nói đến stored procedure là nói đến loại này. Local stored procedure thường được viết bởi DBA hoặc programmer.
2. **Temporary Stored Procedure**: Là những stored procedure tương tự như local stored procedure nhưng chỉ tồn tại cho đến khi connection đã tạo ra chúng bị đóng lại hoặc SQL Server shutdown. Các stored procedure này được tạo ra trên TempDB của SQL Server nên chúng sẽ bị delete khi connection tạo ra chúng bị cắt đứt hay khi SQL Server down. Temporary stored procedure được chia làm 3 loại: **local** (bắt đầu bằng #), **global** (bắt đầu bằng ##) và stored procedure được tạo ra trực tiếp trên TempDB. Loại local chỉ được sử dụng bởi connection đã tạo ra chúng và bị xóa khi disconnect, còn loại global có thể được sử dụng bởi bất kỳ connection nào. Permission cho loại global là dành cho mọi người (public) và không thể thay đổi. Loại stored procedure được tạo trực tiếp trên TempDB khác với 2 loại trên ở chỗ ta có thể set permission, chúng tồn tại kể cả sau khi connection tạo ra chúng bị cắt đứt và chỉ biến mất khi SQL Server shut down.
3. **Extended Stored Procedure**: Ðây là một loại stored procedure sử dụng một chương trình ngoại vi (external program) vốn được compiled thành một DLL để mở rộng chức năng hoạt động của SQL Server. Loại này thường bắt đầu bằng tiếp đầu ngữ **xp\_**. Ví dụ, *xp\_sendmail* dùng đề gởi mail cho một người nào đó hay *xp\_cmdshell* dùng để chạy một DOS command... Ví dụ *xp\_cmdshell 'dir c:\'*. Nhiều loại extend stored procedure được xem như system stored procedure và ngược lại.
4. **Remote Stored Procedure**: Những stored procedure gọi stored procedure ở server khác.

#### TẠO SP

Sử dụng lệnh CREATE PROCEDURE để tạo SP, SP được lưu ở DB hiện hành.

Nếu trong SP có tạo một bảng tạm, thì bảng tạm chỉ tồn tại khi thực thi SP, bảng tạm sẽ tự động bị xóa khi thi hành xong SP (bảng tạm có tên bắt đầu bằng ký hiệu #, ví dụ: #NHANVIEN)

Trong SP không được chứa các câu lệnh: CREATE PROCEDURE, CREATE RULE, CREATE VIEW, CREATE TRIGGER.

Để thi hành lệnh CREATE PROCEDURE, người dùng phải là thành viên của một trong các role: sysadmin, db\_owner, db\_ddladmin hoặc được cấp quyền CREATE PROCEDURE.

Để thi hành lệnh CREATE PROCEDURE, người dùng phải là thành viên của một trong các role: sysadmin, db\_owner, db\_ddladmin hoặc được cấp quyền CREATE PROCEDURE.

Cú pháp:

CREATE PROC [EDURE] <Tên\_Procedure>

[@<Tham\_số> <Kiểu\_dữ\_liệu> [ = <Giá\_trị\_mặc\_định> ] [ OUTPUT ] ]

[ ,...n ]

[ WITH {RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION} ] AS

<Các\_câu\_lệnh\_SQL>

Giải thích:

* Tên\_Procedure: tên procedure, các SP tạm cục bộ có ký hiệu # trước tên của SP
* @<Tham\_số>: tên tham số của procedure.
* <Kiểu\_dữ\_liệu>: kiểu dữ liệu của tham số.
* <Giá\_trị\_mặc\_định>: giá trị mặc định của tham số.
* OUTPUT: cho phép tham số nhận giá trị trả về.
  + RECOMPILE: nếu có thêm tùy chọn này thì mỗi lần thi hành SQL Server sẽ biên dịch lại SP và mã của SP không được lưu vào vùng đệm của thủ tục.
  + ENCRYPTION: nếu có thêm tùy chọn này thì văn bản lệnh được mã hóa và lưu trong syscomments.

Ví dụ:

CREATE PROC DS\_NHANVIEN AS

SELECT \* FROM NHANVIEN

Thực thi SP:

EXEC[UTE] <Tên\_Procedure> [Danh\_sách\_tham\_số]

Ví dụ:

**EXEC DS\_NHANVIEN**

#### SỬA SP

Cú pháp:

HIỆU CHỈNH SP:

ALTER PROC [EDURE] <Tên\_Procedure>

[ @<Tham\_so> <Kieu\_du\_lieu> [ = <Gia\_tri\_mac\_dinh> ] [ OUTPUT ] ]

[ ,...n ]

[ WITH {RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION} ] AS

<Cac\_cau\_lenh\_SQL>

Ví dụ:

ALTER PROC DS\_NHANVIEN AS

SELECT \* FROM nhanvien WHERE PHAI= ‘Nam’

#### XÓA SP

Cú pháp:

DROP PROC [EDURE] <Tên\_Procedure>

Ví dụ:

DROP PROC DS\_NHANVIEN

MỘT VÀI VÍ DỤ:

**Ví dụ 1 – Có tham số vào, default:** Xem danh sách nhân viên theo Phòng: CREATE PROC DSNV\_THEOPHONG

@Phong int = 1

AS

SELECT \* FROM NHANVIEN

WHERE PHG = @Phong

Thực thi SP:

* Cách 1:

EXEC DSNV\_THEOPHONG 4

* Cách 2:

DECLARE @P int Set @P = 5

EXEC DSNV\_THEOPHONG @P

Ví dụ 2 – Có tham số vào, ra:

Xóa thân nhân theo MANV:

CREATE PROC XOA\_THANNHAN\_THEOMANV

@MANV nvarchar(20), @SoNVXoa int OUTPUT AS

DELETE THANNHAN

WHERE MA\_NVIEN = @MaNV

SET @SoNVXoa = @@rowcount

Thực thi SP:

DECLARE @SoNVXoa int

EXEC XOA\_THANNHAN\_NHANVIEN ‘123’, @SoNVXoa OUTPUT

PRINT 'So mau tin bi xoa:' + str(@SoNVXoa,3)

Ví dụ 3 – Có recompile, encryption:

Tăng lương cho nhân viên theo phòng lên @Tyle lần.

CREATE PROC TANGLUONG\_NHANVIEN

@Phong int, @Tyle Decimal(3,1), @So\_NV\_Tang int OUTPUT WITH RECOMPILE, ENCRYPTION

AS

UPDATE NHANVIEN

SET LUONG = LUONG \* @Tyle

WHERE PHG = @Phong

SET @So\_NV\_Tang = @@rowcount

Ví dụ 4 – Có chặn lỗi:

Thêm phòng ban:

CREATE PROC THEM\_PHONGBAN @TENPHG NVARCHAR(40), @MAPHG INT, @TRPHG NVARCHAR(20),

@NG\_NHANCHUC SMALLDATETIME, @Loi int OUTPUT AS

BEGIN TRY

INSERT PHONGBAN VALUES

(@TENPHG, @MAPHG, @TRPHG, @NG\_NHANCHUC) END TRY

BEGIN CATCH

SET @Loi = @@error

RAISERROR(N'Lỗi thêm dữ liệu !',10,1) RETURN

END CATCH

SET @Loi = @@error

### TÓM TẮT

*Stored Procedure (Thủ tục nội tại) là một nhóm câu lệnh Transact-SQL đã được biên dịch và chứa trong SQL Server dưới một tên nào đó và được xử lý như một đơn vị (chứ không phải nhiều câu SQL riêng lẻ). Stored Procedure được kết cấu từ một kịch bản câu lệnh T-SQL, có những đặc điểm cơ bản sau:*

* + *Truyền tham số.*
  + *Gọi thủ tục khác.*
  + *Trả về các giá trị tham số, chuyển giá trị tham số cho các thủ tục được gọi.*
  + *Trả về giá trị trạng thái thủ tục là thành công hay không thành công.*

*Stored Procedure có nhiều ưu điểm so với thực hiện câu lệnh T-SQL từ các máy khách:*

* + *Lập trình theo module: Thủ tục được thiết lập trong từng CSDL một lần, có thể gọi thực hiện nhiều lần trong một ứng dụng, có thể gọi từ nhiều ứng dụng.*
  + *Thực hiện nhanh hơn: Khi cần thực hiện một lượng lớn câu lệnh T-SQL, thủ tục lưu trữ thực hiện nhanh hơn vì khi máy chủ nhận được nhiều cầu lệnh cùng một lúcđều phải kiểm tra tính hợp lệ quyền của tài khoản từ máy khách và các tham số khác. Khi thủ tục cần gọi nhiều lần trên các máy khách thì thủ tục thực hiện một lầnđầu tiên, những lần sau máy khách sẽ chạy thủ tục đã được biên dịch.*
  + *Làm giảm lưu lượng trên mạng: Thay vì máy khách phải gửi nhiều dòng lệnh từ các ứng dụng đến máy chủ, khi sử dụng thủ tục thì nó chỉ cần gửi một lệnh, từ đó dẫn đến lưu lượng thông tin lệnh truyền qua mạng giảm.*
  + *An ninh bảo mật hơn: Khi không muốn cho một user trực tiếp khai thác một đối tượng hay bảng dữ liệu nào đó, mà cần cho user đó được khai thác thì thủ tục có thể giúp bạn gán quyền khai thác cho người đó. Việc gán quyền khai thác như nói trên sẽ giúp cho vấn đề an ninh bảo mật trong CSDL tốt hơn.*

### CÂU HỎI ÔN TẬP

**Câu 1:** Stored Procedure là gì? Nêu những điểm giống nhau và khác nhau giữa View và Stored Procedure?

**Câu 2:** Cho biết nguyên lý hoạt động của Stored Procedure? Hãy phân loại Stored Procedure? Ta dùng từ khóa OUTPUT phía sau tham số trong trường hợp nào?

**Câu 3:** Thực hiện các câu hỏi trong bài thực hành số 2 và 3 liên quan đến Stored Procedure.

## BÀI 5: FUNCTION (HÀM)

Sau khi học xong bài này, học viên có thể:

* + Nắm được các khái niệm cơ bản về chương trình con (bổ trợ kiến thức từ bài học trước), cấu trúc và nguyên lý hoạt động của hàm nội tại.
  + Cách thức tạo, sửa, xóa và cách thức gọi một hàm do người dùng định nghĩa.
  + Phân biệt được các loại hàm người dùng và cách thức vận dụng.
  + Hiểu rõ hơn về cách sử dụng, truyền tham số cho hàm, giá trị trả về của hàm.

#### KHÁI NIỆM

Hầu hết các ngôn ngữ lập trình, hay CSDL lớn, luôn luôn có một phần mở rộng cho phép người dùng tự định nghĩa một số quy tắc, hàm hoặc thủ tục. Hàm người dùng (User-defined function) giống như Stored Procedure của SQL server. Hàm người dùng cũng có thể truyền tham số nhưng không được mang thuộc tính OUTPUT, thay vào đó chúng ta dùng câu lệnh RETURN.

Function cho phép nhận tham số vào, gọi hàm bằng lệnh EXECUTE giống như gọi PROCEDURE (đối với hàm trả về kiểu vô hướng – Scalar-valued Functions).

Chú ý:

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

#### PHÂN LOẠI

Có hai loại hàm do người dùng định nghĩa:

* Hàm người dùng trả về kiểu vô hướng (Scalar-valued Functions)
* Hàm người dùng trả về một bảng dữ liệu (Table-valued Functions)

#### TẠO FUNCTION

* Cú pháp hàm người dùng trả về giá trị vô hướng (Scalar-valued Functions):

CREATE FUNCTION <Tên\_hàm> ([@<Tham\_số> <Kiểu\_dữ\_liệu> [ = <Giá\_trị\_mặc\_định>]]

[ ,...n ])

RETURNS <Kiểu\_dữ\_liệu\_trả\_về> [ AS ]

BEGIN

<Thân\_hàm>

RETURN <giá\_trị\_trả\_về>

END

Ví dụ 1: Tính tổng 2 số nguyên.

CREATE FUNCTION TONG(@a INT, @b INT) RETURNS INT AS

RETURN @a+@b

Gọi hàm:

* **Cách 1:**

PRINT 'TONG: ' + STR(DBO.TONG(6,3))

* **Cách 2:**

DECLARE @TONG INT

SET @TONG = DBO.TONG(6,3) SELECT @TONG

Ví dụ 2: Hàm tự viết: Chuyển ngày về dạng DDMMYYYY.

Create Function dbo.dngayDDMMYYYY(@Date smalldatetime) returns varchar(10)

As Begin

return convert(varchar(10),@Date,103)

End

Gọi hàm:

Select MaNV, Hoten, dbo.dngayDDMMYYYY(Ngaysinh), Phai From NHANVIEN

* + Cú pháp hàm người dùng trả về một bảng dữ liệu (Table-valued Functions):

CREATE FUNCTION [owner\_name.] Tên\_hàm

(@tên\_biến [AS] Kiểu\_giá\_trị\_biến [= default ],...) RETURNS TABLE

AS

RETURN (1 câu\_lệnh\_select)

Chú ý: phần thân của hàm chỉ cho phép sự xuất hiện duy nhất của câu lệnh RETURN. Ngoài ra, không sử dụng bất kỳ câu lệnh nào khác trong phần thân của hàm.

Ví dụ 1:

create function dbo.HienthiNV() returns table

as

return (Select \* from NHANVIEN)

Gọi hàm:

Select \* from dbo.HienthiNV()

Ví dụ 2: Ứng dụng hàm người dùng trả về bảng dữ liệu (trường hợp có tham số truyền vào):

create function dbo.HienthiNV(@MaNV varchar(4)) returns Table

as return

(select MaNV, Hoten, Ngaysinh, Phai from NHANVIEN

where MaNV like @MaNV)

Trong trường hợp cần phải sử dụng đến nhiều câu lệnh trong phần thân của hàm, ta sử dụng cú pháp như sau để định nghĩa hàm:

CREATE FUNCTION tên\_hàm([danh\_sách\_tham\_số]) RETURNS @biến\_bảng TABLE định\_nghĩa\_bảng

AS BEGIN

Các\_câu\_lệnh\_trong\_thân\_hàm RETURN

END

Khi định nghĩa hàm dạng này cần lưu ý một số điểm sau:

* Cấu trúc của bảng trả về bởi hàm được xác định dựa vào định nghĩa của bảng trong mệnh đề RETURNS.
* Biến @biến\_bảng trong mệnh đề RETURNS có phạm vi sử dụng trong hàm và được sử dụng như là một tên bảng.
* Câu lệnh RETURN trong thân hàm không chỉ định giá trị trả về.
* Giá trị trả về của hàm chính là các dòng dữ liệu trong bảng có tên là @biến\_bảng được định nghĩa trong mệnh đề RETURNS.

Ví dụ 3:

Create function F4(@MaNV varchar(5)) returns @Tonghop table

(

MaPX int,

MaNV varchar(5)

)

as

begin

if (@MaNV is NULL) INSERT INTO @Tonghop

select MaPX, MaNV from PHIEUXUAT --lấy tất cả else INSERT INTO @Tonghop

select MaPX, N.MaNV from NHANVIEN N, PHIEUXUAT P

where N.MaNV=P.MaNV and N.MaNV=@MaNV return

end

Thực thi:

select \* from F4('NV01') select \* from F4(NULL)

#### HIỆU CHỈNH FUNCTION

Cú pháp (chung cho cả 3 dạng):

ALTER FUNCTION <Tên\_Function>([@<Tham\_số> <Kiểu\_dữ\_liệu> [ = <Giá\_trị\_mặc\_định> ]]

[ ,...n ] )

RETURNS <Kiểu\_dữ\_liệu\_trả\_về> [ AS ]

BEGIN

<Thân\_hàm>

RETURN <giá\_trị\_trả\_về>

END

#### XÓA FUNCTION

Cú pháp:

DROP FUNCTION <Tên\_Function>

Ví dụ:

Drop function TONG

MỘT VÀI VÍ DỤ:

**Tính tổng lƣơng theo Mã phòng:**

CREATE FUNCTION TONG\_LUONG\_PHONG(@PHG INT) RETURNS INT BEGIN

DECLARE @TONGLUONG INT SET @TONGLUONG **=**

(SELECT SUM(LUONG) FROM NHANVIEN WHERE PHG = @PHG)

RETURN @TONGLUONG END

**Tính tổng thời gian theo tên Đề án:**

CREATE FUNCTION TONG\_TG\_DA(@TENDA NVARCHAR(40)) RETURNS FLOAT

BEGIN

DECLARE @TONG\_TG FLOAT SET @TONG\_TG **=**

(SELECT SUM(THOIGIAN) FROM PHANCONG, DEAN WHERE TENDA = @TENDA AND SODA = MADA)

RETURN @TONG\_TG END

**TÓM TẮT**

*Cũng giống như Stored Procedure, Function (Hàm) là một đối tượng trong CSDL bao gồm một tập nhiều câu lệnh SQL được nhóm lại với nhau thành một nhóm. Điểm khác biệt giữa hàm và thủ tục là hàm trả về một giá trị thông qua tên hàm. Điều này cho phép ta sử dụng hàm như là một thành phần của một biểu thức chẳng hạn như trong các câu lệnh truy vấn hay các câu lệnh thực hiện cập nhật dữ liệu.*

*Hàm người dùng cũng có thể truyền tham số nhưng không được mang thuộc tính OUTPUT, thay vào đó chúng ta dùng câu lệnh RETURN.*

*Trong SQL có rất nhiều các hàm được định nghĩa sẵn (Được chia theo nhóm - Trong một Database, ta chọn Programmability/Functions/System Functions) như các hàm về chuỗi (String Functions), các hàm về ngày tháng (Date and Time Functions), Các hàm toán học (Mathematical Functions)... Ngoài những hàm do Hệ quản trị CSDL cung cấp sẵn, ta có thể tự xây dựng các hàm nhằm phục vụ cho mục đích riêng của mình - Các hàm do người dùng định nghĩa. Các hàm do người dùng định nghĩa thường có 2 loại: Loại 1 là Hàm trả về là một giá trị (Scalar-valued Functions) và các hàm này cũng sẽ được Hệ quản trị phân thành 2 nhóm. Loại 2 là Hàm với giá trị trả về là "dữ liệu kiểu bảng" (Table-valued Functions).*

*Function cho phép nhận tham số vào, gọi hàm bằng lệnh EXECUTE giống như gọi PROCEDURE (đối với hàm trả về kiểu vô hướng – Scalar-valued Functions).*

*Chú ý:*

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

### CÂU HỎI ÔN TẬP

**Câu 1:** Function là gì? Nêu những điểm giống nhau và khác nhau giữa Function và Stored Procedure? Cho biết nguyên lý hoạt động của Function? Hãy phân loại Function?

**Câu 2:** Trong function, ta có dùng từ khóa OUTPUT phía sau tham số không? Giải thích?

**Câu 3:** Thực hiện các câu hỏi trong bài thực hành số 2 và 3 liên quan đến Function.

## BÀI 6: TRIGGER (BẪY LỖI)

Sau khi học xong bài này, học viên có thể:

* + *Nắm được cơ chế hoạt động của một table.*
  + *Các biến cố tác động lên một table. Liên kết hoạt động giữa 2 hay nhiều table có mối kết hợp với nhau.*
  + *Hiểu rõ tính ràng buộc toàn vẹn của CSDL, từ đó có thể xây dựng và vận hành các trigger nhằm đảm bảo sự toàn vẹn dữ liệu và an toàn cho hệ thống.*

#### KHÁI NIỆM

Trigger là một Stored procedure đặc biệt tự động được chạy mỗi khi có một hành động nào đó xảy ra có liên quan đến nó. Các hoạt động xảy ra giúp trigger hoạt động: Insert, Delete, Update (DML). TRIGGER không được thực thi một cách tường minh nên cần thận trọng khi dùng trigger.

Trigger hoạt động gần giống các hàm bắt sự kiện trong javascript. Nó chỉ được kích hoạt khi xảy ra một hành động mà cụ thể là các thao tác Insert, Delete, Update trên một table. Nó hoạt động một các thụ động nên chúng ta không thể biết trigger thực thi khi nào.

#### SỬ DỤNG TRIGGER

Sử dụng trigger nhằm mục đích:

* + Đảm bảo tính ràng buộc toàn vẹn cho CSDL.
  + Kiểm soát dữ liệu hiện có trong CSDL khi có thay đổi giá trị của một mẩu tin trong bảng.
  + Kiểm tra dữ liệu mới nhập vào có thỏa mãn điều kiện không.
  + Kiểm chứng khi xóa mẩu tin trong bảng.
* Tự động cập nhật dữ liệu cho bảng B khi dữ liệu bảng A thay đổi (khi 2 bảng có mối quan hệ với nhau).

#### NGUYÊN TẮC HOẠT ĐỘNG

* Triggers được thực hiện tự động sau khi lệnh INSERT, UPDATE, hoặc DELETE được thực hiện trên một table mà trigger đó được định nghĩa. Còn các constraints và INSTEAD OF trigger sẽ được kiểm tra trước khi lệnh INSERT, UPDATE, hoặc DELETE thực hiện.
* Constraints sẽ được kiểm tra trước trigger.
* Một table có thể có nhiều Triggers cho một action. Một trigger có thể được định nghĩa cho nhiều action.
* Thứ tự thi hành sẽ là: trigger INSTEAD OF, các constraint, và sau cùng là trigger AFTER. Khi có nhiều trigger trong một table, thì table owner có thể dùng procedure hệ thống sp\_settriggerorder để chỉ định trigger đầu và trigger cuối để thực thi. Thứ tự của các trigger còn lại không thể sắp xếp được.
* User phải có quyền để thực hiện tất cả các lệnh mà được định nghĩa trong Triggers
* Table Owners không thể tạo ra các Triggers trên Views hoặc Temporary Tables nhưng có thể tham chiếu đến view và temporary.
* Khi có nhiều trigger trong một table, thì table owner có thể dùng procedure hệ thống **sp\_settriggerorder** để chỉ định trigger đầu và trigger cuối để thực thi. Thứ tự của các trigger còn lại không thể sắp xếp được.
* User phải có quyền để thực hiện tất cả các lệnh mà được định nghĩa trong Triggers.
* Table Owners không thể tạo ra các Triggers trên Views hoặc Temporary Tables nhưng có thể tham chiếu đến view và temporary.
* Triggers không trả kết quả về.
* Triggers có thể điều khiển multi-row actions: một hành động INSERT, UPDATE, hoặc DELETE gọi một trigger có thể ảnh hưởng lên nhiều dòng dữ liệu, Ta có thể chọn:
* Xử lý tất cả các dòng cùng với nhau trong trường hợp các dòng ảnh hưởng phải thỏa điều kiện của trigger.
* Xử lý từng dòng thỏa điều kiện.

#### PHÂN LOẠI

Có 4 loại trigger sau:

1. Insert trigger:

Được thực hiện mỗi khi mẩu tin mới được chèn vào bảng. Một bảng tạm Inserted sẽ được sinh ra để chứa mẩu tin cần chèn.

1. Delete trigger:

Được thực hiện mỗi khi các mẩu tin trong bảng bị xóa. Một bảng tạm Deleted được sinh ra để lưu các mẩu tin bị xóa.

1. Update trigger:

Được thực hiện khi các bản các mẩu tin của bảng được cập nhật. Hai bảng Inserted và Deleted sẽ được sinh ra. Bảng Inserted sẽ lưu thông tin các mẩu tin mới được sửa, bảng Deleted sẽ lưu thông tin các mẩu tin cũ.

1. Instead of trigger:

Trigger cho phép cập nhật dữ liệu các bảng thông qua view có liên kết nhiều bảng, hoặc bẫy trước khi các constraint có tác dụng.

Chú ý:

* + Một bảng có nhiều trigger.
  + Mỗi một trigger có tên duy nhất.
  + Trong trigger thường dùng mệnh đề IF EXISTS.
  + Sử dụng trigger trong toàn vẹn dữ liệu.
  + Sử dụng trigger trong ràng buộc tham chiếu.

#### TẠO TRIGGER

Cú pháp:

CREATE TRIGGER <Tên\_ Trigger> ON <Tên\_bảng> FOR < [INSERT] [,] [UPDATE] [,] [DELETE]> AS

<Các\_câu\_lệnh\_SQL>

Chú thích:

<Tên\_ Trigger>: tên Trigger.

<Tên\_bảng>: tên bảng cần định nghĩa Trigger.

INSERT, UPDATE, DELETE: định nghĩa thao tác để kích hoạt Trigger.

**Ví dụ 1: Tạo ràng buộc lƣơng nhân viên phải là số dƣơng.**

CREATE TRIGGER LUONG\_DUONG ON NHANVIEN FOR INSERT, UPDATE

AS BEGIN

IF (SELECT COUNT(\*) FROM INSERTED WHERE LUONG < 0) > 0 BEGIN

Print N'Lương phải là số dương' Rollback Tran

END END

**Ví dụ 2: Ràng buộc 0 <= LUONG <= 100000**

CREATE TRIGGER LUONG\_DUONG ON NHANVIEN FOR INSERT, UPDATE

AS BEGIN

IF (SELECT COUNT(\*) FROM INSERTED

WHERE LUONG < 0 OR LUONG > 100000) > 0 BEGIN

Print ‘0 <= LUONG <= 100000’

Rollback Tran

END END

#### HIỆU CHỈNH TRIGGER

Cú pháp:

ALTER TRIGGER <Tên\_ Trigger> ON <Tên\_bảng> FOR <[INSERT] [,] [UPDATE] [,] [DELETE]> AS

<Các\_câu\_lệnh\_SQL>

#### XÓA TRIGGER

Cú pháp:

DROP TRIGGER <Ten\_ Trigger>

Ví dụ:

DROP TRIGGER LUONG\_DUONG

#### MỘT VÀI VÍ DỤ

* + **Ràng buộc lƣơng nhân viên phải tăng:**

CREATE TRIGGER LUONG\_TANG ON NHANVIEN FOR UPDATE

AS

IF UPDATE(LUONG) BEGIN

IF (SELECT COUNT(\*) FROM INSERTED I, DELETED D

WHERE D.LUONG >= I.LUONG AND I.MANV=D.MANV) > 0 BEGIN

Print N'Lương nhân viên phải tăng' Rollback Tran

END

END

* **Ràng buộc nhân viên phải từ 18 tuổi trở lên:**

CREATE TRIGGER TUOI\_18\_TROLEN ON NHANVIEN FOR INSERT, UPDATE

AS

IF EXISTS (SELECT \* FROM INSERTED

WHERE DATEADD(YY, 18, NGSINH) > GETDATE()) BEGIN

Print N'Nhân viên phải từ 18 tuổi trở lên' Rollback Tran

END

* **Ràng buộc không cho phép thêm, sửa bảng NHANVIEN nếu PHG không có trong bảng PHONGBAN:**

CREATE TRIGGER TONTAI\_PHONG ON NHANVIEN FOR INSERT, UPDATE

AS

IF NOT EXISTS (SELECT \* FROM INSERTED, PHONGBAN WHERE PHG = MAPHG)

BEGIN

Print N’Mã phòng chưa tồn tại’ Rollback Tran

END

* **Ràng buộc không cho sửa MAPHG trong bảng PHONGBAN:**

CREATE TRIGGER KHONG\_SUA\_KHOACHINH ON PHONGBAN FOR UPDATE

AS

IF UPDATE(MAPHG) BEGIN

Print N‘Không được sửa khóa chính’ Rollback Tran

END

* + **Sửa MAPHG trong bảng PHONGBAN thì phải sửa luôn những mẫu tin có liên quan trong bảng NHANVIEN:**

CREATE TRIGGER SUA\_DAYCHUYEN ON PHONGBAN FOR UPDATE

AS

IF UPDATE(MAPHG) BEGIN

UPDATE NHANVIEN SET PHG = I.MAPHG

FROM NHANVIEN, DELETED D, INSERTED I WHERE PHG = D.MAPHG

END

* + **Không cho xóa NHANVIEN nếu nhân viên đó có trong bảng PHANCONG:**

CREATE TRIGGER KHONG\_XOA\_NV\_CO\_PHANCONG ON NHANVIEN FOR DELETE

AS

IF (SELECT COUNT(\*) FROM DELETED, PHANCONG WHERE MANV = MA\_NVIEN) > 0

BEGIN

Print N’Nhân viên có trong phân công’ Rollback Tran

END

**TÓM TẮT**

*Trigger là một loại đặc biệt của thủ tục lưu trữ, nó tự động thực thi khi một sự kiện xảy ra trong các máy chủ CSDL. DML (Data Manipulation Language – ngôn ngữ thao tác dữ liệu) gây nên khi người dùng cố gắng sửa đổi dữ liệu thông qua một ngôn ngữ thao tác dữ liệu (sự kiện INSERT, UPDATE, hoặc DELETE trên một Table hoặc View). Trigger không được thực thi một cách tường minh nên cần thận trọng khi dùng trigger.*

*Trigger là một thủ tục không có tham số. Một table có thể có nhiều trigger. Trigger thường dùng để kiểm tra các ràng buộc mà không thể khai báo trên table như các ràng buộc liên thuộc tính-liên quan hệ, liên bộ-liên quan hệ, ràng buộc do sự hiện diện của chu trình.*

*Sử dụng lệnh ROLLBACK TRAN trong trigger để bãi bỏ phát biểu cập nhật khi cần thiết.*

*Các loại trigger: Insert trigger, Delete trigger, Update trigger và Istead of trigger.*

### CÂU HỎI ÔN TẬP

**Câu 1:** Khái niệm về trigger? Tại sao phải dùng trigger trong một hệ thống CSDL?

**Câu 2:** Phân loại trigger và cho biết nguyên lý hoạt động của từng loại?

**Câu 3:** Ta sử dụng Check Constraint lúc tạo field cho table khi nào? Trong ngữ cảnh đó, ta dùng trigger được không? Giải thích?

**Câu 4:** Thực hiện các câu hỏi trong bài thực hành số 2 và 3 liên quan đến trigger.

## BÀI 7: KIỂU DỮ LIỆU CURSOR

Sau khi học xong bài này, học viên có thể:

* + *Nắm được các khái niệm cơ bản kiểu dữ liệu cursor: khái niệm, cách dùng, ý nghĩa.*
  + *Phân biệt được các loại cursor và cách thức sử dụng.*
  + *Hiểu và vận dụng tốt quy trình sử dụng cursor.*

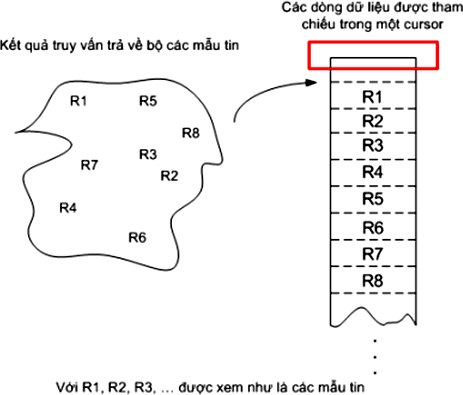
#### ĐẶT VẤN ĐỀ

Phần lớn các CSDL quan hệ thường làm việc trên dữ liệu của nhiều dòng mẫu tin, còn gọi là một bộ các mẫu tin. Ví dụ lệnh SELECT kết quả luôn trả về nhiều dòng dữ liệu hơn là một dòng dữ liệu.

Ngược lại đối với một số ngôn ngữ lập trình hoặc bên trong các ứng dụng thì người lập trình vẫn còn các thói quen xử lý và tính toán dữ liệu trên từng dòng riêng lẻ. Để đáp ứng được yêu cầu này của các người lập trình, muốn làm việc chỉ trên từng dòng dữ liệu tại thời điểm hiện hành, Microsoft SQL Server tạo ra một kiểu dữ liệu đó chính là kiểu cursor (kiểu con trỏ).

Có thể hình dung kiểu dữ liệu cursor giống như một cuốn sổ danh bạ chứa thông tin liên lạc của các khách hàng giao tác trong một công ty. Bằng cách dò tìm thủ công, ta sẽ phải sử dụng đến mắt và tay để tham chiếu đến tên của các khách hàng bất bỳ trong sổ danh bạ đó. Ta có thể di chuyển lên, xuống hoặc qua trang để tìm ra các khách hàng mong muốn, nhưng tại thời điểm hiện hành, tay và mắt của mình chỉ đứng tại một khách hàng mà thôi.

Hoạt động của kiểu dữ liệu cursor trong Transaction-SQL hoàn toàn giống như ví dụ minh họa ở trên. Tuy nhiên, cursor có nhiều kiểu khác nhau cho phép ta có thể chọn lựa để định nghĩa theo đúng yêu cầu mà mình mong muốn. Tùy thuộc vào kiểu cursor đã định nghĩa mà việc đọc và cập nhật dữ liệu sẽ có hiệu lực như thế nào.



**Hình 7.1: So sánh cơ chế cursor và bộ các mẩu tin**

#### KHÁI NIỆM

Cursor là một cấu trúc dữ liệu ánh xạ đến một tập các dòng dữ liệu kết quả của một câu truy vấn (select), cho phép duyệt tuần tự qua tập các dòng dữ liệu và đọc giá trị từng dòng. Thể hiện của cursor là 1 biến, nhưng biến này không bắt đầu bằng „@‟.

Vị trí hiện hành của cursor có thể được dùng như điều kiện trong mệnh đề where của lệnh update hoặc delete: cho phép cập nhật/xóa dữ liệu (dữ liệu thật sự trong CSDL) tương ứng với vị trí hiện hành của cursor.

#### KHAI BÁO CURSOR

Cú pháp SQL92 chuẩn:

Declare *cursor\_name* [Insensitive] [Scroll] Cursor For *select\_statement*

[For {Read only | Update [of *column\_name* [,…n]]}]

Cú pháp T\_SQL mở rộng:

Declare *cursor\_name* Cursor [Local | Global] [Forward\_only | Scroll] [Static | Dynamic] [Read\_only]

For *select\_statement*

[For Update [of column\_name [,…n]]]

Lưu ý: *Tên cursor trong các cách khai báo* ***không*** *bắt đầu bằng ký tự* „@‟.

Giải thích:

* + Insensitive/static: nội dung của cursor không thay đổi trong suốt thời gian tồn tại, trong trường hợp này, cursor chỉ là read only.
  + Dynamic: trong thời gian tồn tại, nội dung của cursor có thể thay đổi nếu dữ liệu trong các bảng liên quan có thay đổi.
  + Local: cursor cục bộ, chỉ có thể sử dụng trong phạm vi một khối (query batch) hoặc một thủ tục/hàm.
  + Global: cursor toàn cục, có thể sử dụng trong một thủ tục/hàm hay một query batch bất kỳ hoặc đến khi bị hủy một cách tường minh.
  + Forward\_only: cursor chỉ có thể duyệt một chiều từ đầu đến cuối.
  + Scroll: có thể duyệt lên xuống cursor tùy ý (duyệt theo đa chiều).
  + Read only: có thể đọc từ cursor, không thể sử dụng cursor để update dữ liệu trong các bảng liên quan (ngược lại với “for update…”).
* Update [of column\_name [,…n]]: danh sách các field có thể update dữ liệu. Nếu chỉ có update thì tất cả các field đều có thể cập nhật dữ liệu.
* Mặc định khai báo cursor nếu không chỉ ra các tùy chọn thì cursor có các tính chất:
* Global
* Forward\_only
* Read only hay “for update” tùy thuộc vào câu truy vấn
* Dynamic

#### PHÂN LOẠI CURSOR

Có 3 loại cursor:

* Static – Con trỏ tĩnh
* Keyset – Con trỏ keyset
* Dynamic – Con trỏ động

Static cursor:

Khi con trỏ này được tạo ra, những mẫu tin được copy vào một bảng tạm thời trong CSDL temdp. Con trỏ làm việc với bảng tạm thời, do đó những thay đổi dữ liệu trong bảng gốc không có tác động đến con trỏ. Con trỏ static không cho phép cập nhật dữ liệu.

Keyset cursor:

Tập dữ liệu trong khóa được copy vào bảng tạm trong temdp, tập dữ liệu trong khóa phải duy nhất. Cho phép thay đổi các field dữ liệu không phải là khóa. Việc thay đổi các field không phải là khóa sẽ có tác động ngay đến con trỏ. Việc xóa mẫu tin hoặc thay đổi khóa có thể làm con trỏ bị lỗi. Việc thêm mới mẫu tin vào bảng gốc không tác động đến con trỏ.

Dynamic cursor:

Con trỏ Dynamic không sử dụng đến bảng tạm trong temdp mà thao tác trực tiếp với bảng gốc. Khác với con trỏ Keyset, dynamic không yêu cầu unique index. Mọi thao tác thêm, sửa, xóa đều có tác động ngay lập tức đến con trỏ Dyanamic.

#### THAY ĐỔI DỮ LIỆU TẠI VỊ TRÍ CURSOR

Có thể sử dụng Update hoặc Delete để sửa đổi hoặc xóa mẫu tin tại vị trí hiện hành của con trỏ với mệnh đề:

where current of Tên\_Cursor

Lưu ý: Chỉ áp dụng với con trỏ dạng Keyset và Dynamic.

#### DUYỆT CURSOR

Dùng lệnh Fetch để duyệt tuần tự qua cursor theo cú pháp:

Fetch

[[Next | Prior | First | Last | Absolute n | Relative n]

From] Tên\_cursor [Into Tên\_biến [,…n]]

Các từ khóa NEXT, PRIOR, FIRST, LAST: dùng để đọc dữ liệu của dòng dữ liệu kế tiếp (next), dòng dữ liệu phía trước (prior), dòng dữ liệu đầu tiên (first), dòng dữ liệu cuối cùng (last) so với dòng dữ liệu hiện hành bên trong cursor. Sau khi đọc dữ liệu thành công, dòng dữ liệu hiện hành sẽ bị thay đổi chính là dòng vừa mới được đọc.

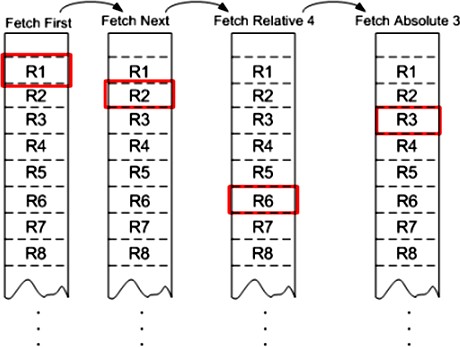
Từ khóa ABSOLUTE: dùng để chỉ định dữ liệu chính xác thứ n bên trong cursor. Với n là số nguyên dương dùng chỉ định việc đọc dữ liệu tại dòng thứ n được đếm từ dòng đầu tiên, với n là số nguyên âm dùng chỉ định việc đọc dữ liệu tại dòng thứ n được đếm ngược từ dòng cuối cùng trở lên.

Từ khóa RELATIVE: dùng để chỉ định việc đọc dữ liệu tại một dòng tương đối so với dòng dữ liệu hiện hành. Với n là một số nguyên có thể dương hoặc âm để chỉ định việc đọc theo chiều tới hoặc lui so với dòng dữ liệu hiện hành.

Mặc định: fetch next.

Đối với cursor dạng forward\_only, chỉ có thể fetch next.

Biến hệ thống @@fetch\_status cho biết lệnh fetch vừa thực hiện có thành công hay không, giá trị của biến này là cơ sở để biết đã duyệt đến cuối cursor hay chưa.



**Hình 7.2: Minh họa việc đọc dữ liệu theo các thứ tự khác nhau**

#### QUY TRÌNH SỬ DỤNG CURSOR

* Khai báo cursor.
* “Mở” cursor bằng lệnh Open: Open Tên\_cursor
* Khai báo các biến tạm để chứa phần tử hiện hành (đang được xử lý) của cursor:
* Các biến tạm phải có cùng kiểu dữ liệu với các field tương ứng của phần tử trong cursor.
* Có n field trong phần tử của cursor thì có đủ n biến tạm tương ứng
* Fetch (next,…) cursor để chuyển đến vị trí phù hợp:
* Có thể đưa các giá trị của dòng hiện hành vào các biến thông qua mệnh đề into của lệnh fetch.
  + Có thể sử dụng vị trí hiện tại như là điều kiện cho mệnh đề where của câu delete/update (nếu cursor không là read\_only)
  + Lặp lại việc duyệt và sử dụng cursor, có thể sử dụng biến @@fetch\_status để biết đã duyệt qua hết cursor hay chưa:
    - @@FETCH\_STATUS = 0: lấy mẫu tin thành công.
    - @@FETCH\_STATUS = -1: lấy mẫu tin thất bại với lý do là con trỏ đã ra quá vùng giới hạn BOF hoặc EOF.
    - @@FETCH\_STATUS = -2: lấy mẫu tin thất bại với lý do là mẫu tin không tồn tại.
  + Đóng cursor bằng lệnh Close: Close Tên\_cursor

Lưu ý: Sau khi đóng, vẫn có thể mở lại nếu cursor chưa bị hủy.

* + Hủy cursor bằng lệnh deallocate: Deallocate Tên\_cursor Ví dụ 1:

Xét loại quan hệ: SANPHAM(MASP,TENSP)

Declare cur\_SP cursor

For select MASP, TENSP from SANPHAM

Declare @Masp varchar(10), @Tensp nvarchar(30) Open cur\_SP

Fetch Next from cur\_SP into @Masp, @Tensp Print N‘Mã sản phẩm: ’ + @Masp +

N‘ Tên sản phẩm: ‘ + @Tensp Close cur\_SP

Deallocate cur\_SP

Có nhận xét gì trong ví dụ trên?

Cải tiến Ví dụ 1 ở trên:

Xét loại quan hệ: SANPHAM(MASP,TENSP)

Declare cur\_SP cursor

For select MASP, TENSP from SANPHAM

Declare @Masp varchar(10), @Tensp nvarchar(30) Open cur\_SP

Fetch Next from cur\_SP into @Masp, @Tensp While (@@fetch\_status = 0)

begin

Print N‘Mã sản phẩm: ’ + @Masp + N‘ Tên sản phẩm: ‘ + @Tensp

Fetch Next from cur\_SP into @Masp, @Tensp end

Close cur\_SP Deallocate cur\_SP Ví dụ 2:

--1. Khai báo biến Cursor

Declare cur\_Vattu CURSOR KEYSET

For

select \* from VT where TENVT Like ‘DA%’ order by MAVT

--2. Mở cursor

Open cur\_Vattu

Fetch next from cur\_Vattu

--3. Đọc dữ liệu

While @@FETCH\_STATUS=0 Begin

--Đọc các dòng kế tiếp

Fetch next from cur\_Vattu End

--4. Đóng cursor

Close cur\_Vattu Deallocate cur\_Vattu Ví dụ 3:

Xét CSDL gồm 2 bảng sau:

PHIEUNHAP(SOPN, TGNHAP), CTPHIEUNHAP(SOPN, SLNHAP, DGNHAP).

Để cập nhật giá trị dữ liệu cho cột TGNHAP (trị giá nhập) trong bảng PHIEUNHAP bằng cách duyệt qua từng phiếu nhập, tính ra trị giá nhập của từng phiếu căn cứ vào số lượng nhập và đơn giá nhập của từng vật tư trong bảng CTPNHAP, sau cùng cập nhật vào cột TGNHAP.

--1. Khai báo biến cursor, các biến cục bộ

Declare cur\_PNHAP CURSOR Forward\_only FOR

select SOPN from PHIEUNHAP

Declare @SoPN char(4), @TongTG money

--2. Mở cursor

Open cur\_PNHAP

--3. Đọc dữ liệu và cập nhật giá trị

While 0=0 Begin

Fetch next from cur\_PNHAP INTO @SoPN If @@FETCH\_STATUS<>0 Break

Select @TongTG=SUM(SLNHAP\*DGNHAP) from CTPHIEUNHAP where SOPN=@SoPN

Print N‘Đang nhập số phiếu nhập: ’ + @SoPN + ‘…’ Update PHIEUNHAP

Set TGNHAP = @TongTG

Where CURRENT OF cur\_PNHAP End

--4. Đóng cursor

Close cur\_PNHAP Deallocate cur\_PNHAP

Nhận xét:

Trong ví dụ 2 ở trên, sử dụng vòng lặp WHILE mà điều kiện lặp là 0 = 0 để chỉ định điều kiện so sánh vòng lặp là luôn luôn đúng. Do đó, bên trong vòng lặp này bạn bắt buộc phải thoát khỏi vòng lặp bằng lệnh BREAK và điều kiện thoát là khi việc đọc dữ liệu bị lỗi (@@FETCH\_STATUS <> 0). Ngoài việc cập nhật dữ liệu bằng lệnh UPDATE mà trong mệnh đề WHERE có sử dụng từ khóa CURRENT OF

<Tên cursor> dùng để chỉ định việc cập nhật dữ liệu trên dòng dữ liệu hiện hành của cursor.

#### BIẾN CURSOR

Ta có thể khai báo một biến kiểu cursor và gán cho nó tham chiếu đến một cursor đang tồn tại.

* Biến cursor có thể được xem như là con trỏ cursor.
* Biến cursor là một biến cục bộ.
* Biến cursor sau khi gán giá trị được sử dụng như một cursor thông thường.

Ví dụ:

Declare @cur\_var cursor

Set @cur\_var = my\_cur --my\_cur là một cursor đang tồn tại

Hoặc:

Declare @cur\_var cursor

Set @cur\_var = cursor for select\_statement

#### KẾT HỢP CURSOR VỚI STORED PROCEDURE

Xây dựng SP tính điểm trung bình và xếp loại cho sinh viên thuộc một Lớp cho trước. Giả sử có các loại quan hệ như sau:

SINHVIEN(MASV, HOTEN, DTB, XEPLOAI, LOP)

MONHOC(MAMH, TENMH) KETQUA(MAMH, MASV, LANTHI, DIEM)

Biết rằng:

Điểm thi chỉ tính lần sau cùng. Xếp loại: Xuất sắc [9, 10], Giỏi [8, 8.9], Khá [7, 7.9], Trung bình [5, 6.9], Yếu [0, 4.9]. Kết quả ghi vào CSDL, đồng thời xuất ra tổng số SV xếp loại Giỏi của lớp đó.

Phân tích ví dụ:

Lớp cần xét có nhiều SV, từng SV cần được xử lý thông qua 3 bước:

* + Tính điểm trung bình cho SV, điểm trung bình phải là điểm của lần thi sau cùng. Có thể tái sử dụng thủ tục XepLoaiSVLop.
  + Dựa vào điểm trung bình của SV để xếp loại.
  + Cập nhật điểm và xếp loại vào bảng SINHVIEN. Từ phân tích trên ta thấy:
  + Cần xử lý nhiều phần tử (các SV).
  + Mỗi phần tử xử lý tương đối phức tạp (truy vấn, tính toán, gọi thủ tục khác, điều kiện rẽ nhánh, cập nhật dữ liệu…).
  + Cách xử lý các phần tử là như nhau.

Suy ra: sử dụng cursor là thích hợp. Cursor chứa các SV của lớp cần xét, chỉ cần chứa MASV là đủ thông tin.

Xây dựng thủ tục:

Create procedure XepLoaiSVLop

@Lop nvarchar(10), @SoSVGioi int OUT As

declare @DTB float

declare @XepLoai nvarchar(20) declare @MaSV varchar(10) declare cur\_SV cursor

for (select MASV from SINHVIEN where LOP=@Lop) open cur\_SV

Fetch Next from cur\_SV into @MaSV While (@@FETCH\_STATUS=0)

Begin

Exec XepLoaiSV @MaSV, @DTB output, @XepLoai output

Update SINHVIEN set DTB=@DTB,

XEPLOAI=@XepLoai where MASV=@MaSV

Fetch Next from cur\_SV into @MaSV

End

Close cur\_SV Deallocate cur\_SV

Set @SoSVGioi = (select count(\*) from SINHVIEN)

where LOP=@Lop and XEPLOAI like N’Giỏi’)

#### KẾT LUẬN

Tiện ích duy nhất khi làm việc với cursor là chúng cho phép bạn có thể duyệt tuần tự trên các dòng dữ liệu (giống như đối tượng recordset của ADO hoặc DAO trong ngôn ngữ lập trình Visual Basic, Access, Delphi…), tuy nhiên Microsft SQL Server khuyến cáo bạn không nên lạm dụng quá nhiều các hành động cập nhật dữ liệu bằng cursor bởi vì nó sẽ làm cho các xử lý này chậm. Bằng chứng là bạn có thể tính toán giá trị cho cột trị giá nhập (TGNHAP) trong bảng PNHAP bằng một lệnh UPDATE duy nhất thay vì phải sử dụng quá nhiều lệnh cho các xử lý trong cursor.

Không dùng cursor:

Update PHIEUNHAP Set TGNHAP =

(Select Sum(SLNHAP\*DGNHAP) from CTPHIEUNHAP

where PN.SOPN=CTPHIEUNHA.SOPN) From PHIEUNHAP PN

**TÓM TẮT**

*Chúng ta chỉ sử dụng kiểu dữ liệu cursor trong Transaction-SQL để giải quyết các vấn đề:*

* + *Microsoft SQL Server là một hệ quản trị CSDL quan hệ (Relational DataBase Managament System), do đó chúng luôn chọn các giải pháp làm việc trên bộ các mẫu tin.*
  + *Khi cần giải quyết các vấn đề cập nhật dữ liệu thì bạn luôn luôn ưu tiên chọn ra các hướng giải quyết trên bộ các mẫu tin bởi vì khi đó sẽ làm cho các xử lý được nhanh hơn.*
  + *Sau cùng, các hướng giải quyết theo kiểu cursor chỉ là giải pháp sau cùng nhất để chọn lựa khi không còn giải pháp nào tốt hơn.*

*Tiện ích duy nhất khi làm việc với cursor là chúng cho phép chúng ta có thể duyệt tuần tự trên các dòng dữ liệu, tuy nhiên Microsft SQL Server khuyến cáo không nên lạm dụng quá nhiều các hành động cập nhật dữ liệu bằng cursor bởi vì nó sẽ làm cho các xử lý này chậm.*

### CÂU HỎI ÔN TẬP

**Câu 1:** Hãy nêu khái niệm về Cursor? Cho biết các loại Cursor?

**Câu 2:** Khi nào chúng ta dùng Cursor?

**Câu 3:** Thực hiện các câu hỏi trong bài thực hành số 2 và 3 liên quan đến Cursor.

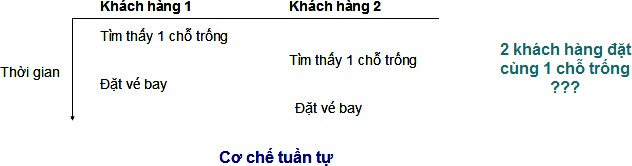
## BÀI 8: QUẢN LÝ GIAO TÁC

Sau khi học xong bài này, học viên có thể:

* *Nắm được khái niệm về giao tác (transaction), các tính chất một giao tác cần phải có để sự hoạt động của nó trong môi trường có "biến động" vẫn đảm bảo cho CSDL luôn ở trạng thái nhất quán.*
* *Hiểu được các khái niệm về khóa, các phương thức khóa cơ bản, các vấn đề liên quan đến truy xuất đồng thời...*
* *Nắm được các mức cô lập cho giao tác.*

#### GIỚI THIỆU

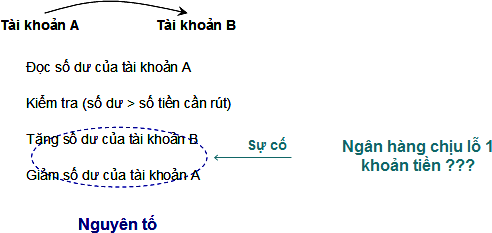
Xét hệ thống đặt vé máy bay:



Hình 8.1: Hệ thống đặt vé máy bay

 DBMS là môi trường đa người dùng, nhiều thao tác truy xuất lên cùng một đơn vị dữ liệu, nhiều thao tác thi hành đồng thời.

Xét hệ thống chuyển khoản trong tài khoản ngân hàng:



Hình 8.2: Hệ thống chuyển khoản trong tài khoản ngân hàng

 Khi DBMS gặp sự cố, các thao tác có thể làm cho trạng thái CSDL không chính xác.

#### KHÁI NIỆM

Giao tác là một đơn vị xử lý nguyên tố gồm một chuỗi các hành động tương tác lên CSDL. Một đơn vị xử lý nguyên tố phải thể hiện bốn tính chất: tính nguyên tố, tính nhất quán, tính cô lập, và tính bền vững.

Atomicity (Nguyên tố):

Một giao tác phải là một đơn vị nguyên tố (không thể phân chia được nữa) của công việc: hoặc là tất cả các thay đổi dữ liệu của nó được thực hiện, hoặc không ai trong số họ được thực hiện.

Ghi chú: Để đảm bảo tính Automicity của giao tác, người sử dụng phải tường minh điều khiển sự rollback của giao tác. Cần kiểm tra lỗi sau khi thực hiện mỗi thao tác trong giao tác để có thể xử lý rollback kịp thời.

Ví dụ: Trong SQL Server, dùng biến toàn cục @@error và @@rowcount. Dùng Stored Procedure:

Create Proc…

Begin

/\*Gọi lệnh insert thêm mới vào SinhVien\*/

/\* Đọc và kiểm tra SiSo lớp \*/

/\* Nếu SiSo >= Max , báo lỗi \*/ rollback transaction

/\*Cập nhật tăng sỉ số\*/

/\* Nếu có lỗi khi thực hiện thao tác:\*/ rollback transaction

End

Ví dụ: Kiểm lỗi bằng @@error.

Create proc sp\_ThemSV @MaSV int, @MaLop int As

begin transaction declare @SiSo int select @SiSo = SiSo from LOP

where MaLop = @MaLop if(@SiSo>=Max)

begin

rollback transaction return

end

Insert into SINH\_VIEN(MaSV,MaLop) values(@MaSV,@MaLop) if(@@error<>0)

begin

rollback transaction return

end

update LOP

set SiSo = SiSo+1 where MaLop = @MaLop if(@@error<>0) begin

rollback transaction return

end

commit transaction /\* hoàn tất giao tác\*/

/\*end stored proc\*/

Consistency (Nhất quán):

Khi hoàn tất, một giao tác cần phải trả về tập dữ liệu trong một khối thống nhất. Trong một CSDL quan hệ, tất cả các luật (rules) phải được áp dụng để sự thay đổi của giao tác vẫn duy trì tính toàn vẹn dữ liệu. Tất cả các cấu trúc dữ liệu nội bộ, chẳng hạn như chỉ mục B-cây hoặc danh sách liên kết đôi, phải được chính xác tại thời điểm kết thúc giao tác.

Isolation (Cô lập):

Cho dù nhiều giao tác có thể thực hiện đồng thời, hệ thống phải đảm bảo rằng đối với mỗi cặp giao tác Ti, Tj, hoặc Tj kết thúc thực hiện trước khi Ti khởi động hoặc Tj bắt đầu thực hiện sau khi Ti kết thúc. Như vậy mỗi giao tác không cần biết đến các giao tác khác đang thực hiện đồng thời trong hệ thống.

Durability (Bền vững):

Sau khi giao tác hoàn thành, các thay đổi đã được tạo ra cho CSDL vẫn tồn tại ngay cả trong trường hợp hệ thống bị lỗi.

Các tính chất này thường được gọi là các tính chất ACID (Các chữ cái đầu của 4 tính chất).

Hãy xét một ví dụ sau: Một hệ thống nhà băng gồm một số tài khoản và một tập các giao tác truy xuất và cập nhật các tài khoản. Tại thời điểm hiện tại, ta giả thiết rằng CSDL nằm trên đĩa, nhưng một vài phần của nó đang nằm tạm thời trong bộ nhớ. Các truy xuất CSDL được thực hiện bởi hai hoạt động sau:

* READ(X): chuyển hạng mục dữ liệu X từ CSDL đến buffer của giao tác thực hiện hoạt động READ này.
* WRITE(X): chuyển hạng mục dữ liệu X từ buffer của giao tác thực hiện WRITE đến CSDL.

Trong hệ CSDL thực, hoạt động WRITE không nhất thiết dẫn đến sự cập nhật trực tiếp dữ liệu trên đĩa; hoạt động WRITE có thể được lưu tạm thời trong bộ nhớ và được thực hiện trên đĩa muộn hơn. Trong ví dụ này, ta giả thiết hoạt động WRITE cập nhật trực tiếp CSDL.

T là một giao tác chuyển 50 từ tài khoản A sang tài khoản B. Giao tác này có thể được xác định như sau:

T: READ(A, t); t := t - 50; WRITE(A, t) READ(B, t); t := t + 50; WRITE(B, t);

Ta xem xét mỗi một công việc trong các yêu cầu ACID:

***Tính nhất quán:*** Tổng A+B là không đổi. Nếu CSDL nhất quán trước khi T được thực hiện thì sau khi T hoàn tất, CSDL vẫn còn nhất quán.

***Tính nguyên tử:*** Giả sử rằng ngay trước khi thực hiện giao tác T, giá trị của các tài khoản A và B tương ứng là 100 và 200, vậy A+B=300.

Tại thời điểm sau khi WRITE(A):

A=50, B=200, (A+B=250)  CSDL không nhất quán. Tại thời điểm sau khi WRITE(B):

A=50, B=250, (A+B=300)  CSDL nhất quán.

Nếu T không bao giờ bắt đầu thực hiện hoặc T được đảm bảo phải hoàn tất thì trạng thái không nhất quán sẽ không xuất hiện.

Đảm bảo tính nguyên tử là trách nhiệm của hệ CSDL, và được quản lý bởi một thành phần được gọi là thành phần quản trị giao tác (transaction-management component).

***Tính bền vững:*** Khi T kết thúc thành công, dữ liệu sẽ không thể nào bị mất bất chấp có sự cố hệ thống xảy ra.

Đảm bảo tính bền vững là trách nhiệm của một thành phần của hệ CSDL, gọi là thành phần quản trị phục hồi (recoverymanagement component). Hai thành phần quản trị giao tác và quản trị phục hồi quan hệ mật thiết với nhau.

***Tính cô lập:*** Giả sử có 1 giao tác T‟ thực hiện phép toán A+B và chen vào giữa thời gian thực hiện của T:

T: READ(A, t); t := t - 50;

WRITE(A, t);

T’

READ(B, t); t := t + 50; WRITE(B, t);

T‟ kết thúc: A+B=50+200=250. T kết thúc: A+B=50+250=300.

Hệ thống của các giao tác thực hiện đồng thời có trạng thái tương đương với trạng thái hệ thống của các giao tác thực hiện tuần tự theo một thứ tự nào đó.

Một giải pháp cho vấn đề các giao tác thực hiện đồng thời là thực hiện tuần tự các giao tác, tuy nhiên giải pháp này làm giảm hiệu năng của hệ thống. Tính cô lập của một giao tác đảm bảo rằng sự thực hiện đồng thời các giao tác dẫn đến một trạng thái hệ thống tương đương với một trạng thái có thể nhận được bởi thực hiện các giao tác này một tại một thời điểm theo một thứ nào đó. Đảm bảo tính cô lập là trách nhiệm của một thành phần của hệ CSDL, được gọi là thành phần quản trị cạnh tranh (concurrency-control component).

#### MỘT SỐ THUẬT NGỮ LIÊN QUAN ĐẾN GIAO TÁC

* Begin [transaction/tran]: bắt đầu một transaction
* Commit [transaction/tran]: hoàn tất một transaction
* Rollback [transaction/tran]: quay lui, hủy bỏ toàn bộ phần giao tác đã thực hiện trước đó.

#### CÁC VẤN DỀ LIÊN QUAN ĐẾN XỬ LÝ TRUY XUẤT DỒNG THỜI

###### Lost Update (mất dữ liệu cập nhật):

Tình trạng này xảy ra khi có nhiều hơn một giao tác cùng thực hiện cập nhật trên 1 đơn vị dữ liệu. Khi đó, tác dụng của giao tác cập nhật thực hiện sau sẽ đè lên tác dụng của thao tác cập nhật trước.

###### Đọc dữ liệu chƣa commit (Uncommitted data, Dirty read)

Xảy ra khi một giao tác thực hiện đọc trên một đơn vị dữ liệu mà đơn vị dữ liệu này đang bị cập nhật bởi một giao tác khác nhưng việc cập nhật chưa được xác nhận đã hoàn tất.

###### Thao tác đọc không thể lặp lại (Unrepeatable data):

Tình trạng này xảy ra khi một giao tác T1 vừa thực hiện xong thao tác đọc trên một đơn vị dữ liệu (nhưng chưa commit) thì giao tác khác (T2) lại thay đổi (ghi) trên đơn vị dữ liệu này. Điều này làm cho lần đọc sau đó của T1 không còn nhìn thấy dữ liệu ban đầu nữa.

###### Bóng ma (Phantom):

Là tình trạng mà một giao tác đang thao tác trên một tập dữ liệu nhưng giao tác khác lại chèn thêm hoặc xóa đi các dòng dữ liệu vào tập dữ liệu mà giao tác kia quan tâm.

###### Cách giải quyết các vấn đề:

Dùng khái niệm giao tác và dùng cơ chế khóa:

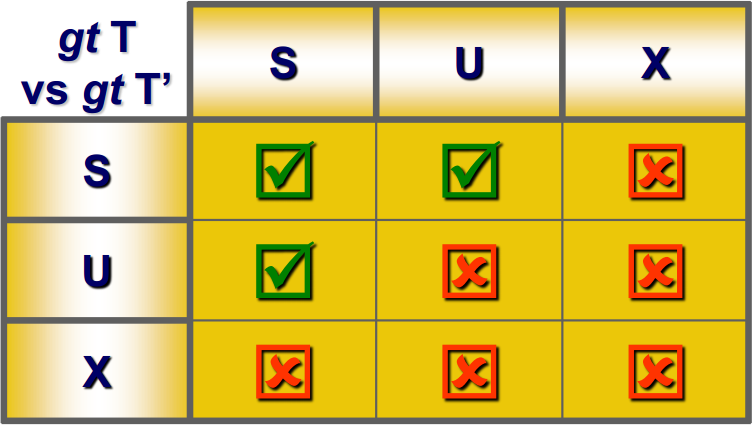
* Write lock (exclusive lock)
* Read lock (shared lock)
* Các hệ quản trị cụ thể còn có những loại khoá mở rộng khác: updlock, holdlock…
* Đặt khóa vào đơn vị dữ liệu:
* Tự động (đặt mức cô lập cho giao tác)
* Thủ công (đặt cấp độ khóa trong câu SELECT)

#### CHẾ ĐỘ KHÓA

Các loại khóa:

* Khóa chia sẻ (shared lock): Còn gọi là khóa đọc (read lock). Gọi tắt: Khóa S.
* Khóa dự định ghi (Intend to write lock): Còn gọi là khóa cập nhật (update lock). Gọi tắt: Khóa U.
* Khóa độc quyền (exclusive lock): Còn gọi là khóa ghi (write lock). Gọi tắt: Khóa X.
  + Khóa X luôn được phát ra khi ghi, bất kể thông số hệ thống đang thiết lập thế nào.
  + Sở dĩ như vậy vì hệ quản trị CSDL hỗ trợ xử lý đồng thời các thao tác đọc nhưng ghi phải ghi lần lượt.

Bảng tương thích giữa các chế độ khóa:



Bảng 8.3 - Bảng tƣơng thích giữa các chế độ

Tương thích: T‟ không phải chờ T

Không tương thích: T‟ phải chờ T giải phóng khóa.

Khi một transaction Ti cần truy cập và thao tác trên một đơn vị dữ liệu X, nó sẽ đòi phát khóa A trên X. Nhưng khi ấy nếu transaction Tj đang giữ khóa B trên X (và khóa A với khóa B không tương thích) thì Ti phải đợi Tj giải phóng khóa B trên X nó mới phát được khóa A trên X  Hiện tượng chờ đợi lẫn nhau.

#### KHAI BÁO TƢỜNG MINH GIAO TÁC

Trong SQL Server, ta có thể khai báo tường minh các giao tác, có thể là trên một khối lệnh độc lập hay trong thân một thủ tục thường trú.

Ngoài ra SQL Server còn có thể phát sinh các giao tác ngầm định (Ví dụ: Trigger).

Các chỉ thị:

* Begin tran: Đặt trước dòng lệnh đầu tiên của giao tác (1 chỉ thị duy nhất cho 1 giao tác).
* Commit tran: Đặt sau dòng lệnh cuối cùng hoàn tất giao tác (1 chỉ thị duy nhất cho 1 giao tác).
* Rollback tran: Đặt tại các vị trí kiểm lỗi hay các nhánh rẽ logic ứng với trường hợp nghiệp vụ thất bại (nhiều chỉ thị cho 1 giao tác).

Kiểm lỗi trong giao tác:

Lỗi có thể sảy ra sau các thao tác:

* Insert, Update (trùng khóa chính, sai kiểu dữ liệu, sai định dạng ngày tháng…)
* Delete (ràng buộc tồn tại…)
* Select (login không có quyền trên object…)

Sau mỗi thao tác trên phải kiểm lỗi bằng biến hệ thống @@error (=0  không có lỗi, ≠0  có lỗi).

Khi lỗi xảy ra (@@error ≠ 0), cần thực hiện các công việc:

* Báo lỗi (nếu cần) bằng các lệnh print hay raise error
* Rollback tran (bắt buộc)
* Nếu Tran khai báo trong SP thì thông thường gọi lệnh *return* để kết thúc SP (tính Automicity)

Ví dụ: đoạn lệnh kiểm lỗi trong giao tác.

If (@@error <> 0) Begin

Print ‘…’ Rollback tran Return

End

###### Mức cô lập cho giao tác:

Mục đích: Tự động đặt khóa cho các thao tác (đọc) trong kết nối dữ liệu hiện hành. Ghi chú:

* + Tầm vực của Isolation level là ở mức connection chứ không phải mức transaction.
    - Khi 1 connection N được đặt mức cô lập X thì X sẽ phát huy hiệu lực trên tất cả các transaction Ti chạy trên N.
  + Mức cô lập chỉ quyết định cách phát và giữ khóa S của transaction (vì khóa X luôn được phát ra khi ghi). Mức cô lập không quan tâm khóa U.

Bốn mức cô lập SQL cung cấp:

1. **Read Uncommited**: Không phát S khi đọc. Có nhu cầu đọc tức thời, ví dụ như khi cần đánh giá tổng quát toàn hệ thống. Tuy nhiên, không giải quyết được bất cứ vấn đề xử lý đống thời nào.
2. **Read Commited**: Phát S khi đọc, giải phóng S ngay sau khi đọc. Chỉ giải quyết được Dirty Read.
3. **Repeatable Read**: Phát S khi đọc và giữ S đến khi transaction kết thúc. Không ngăn chặn lệnh insert/delete dữ liệu thỏa điều kiện thiết lập S. Giải quyết được Dirty Read và Unrepeatable Read.
4. **Seralizable**: Giống Repeatable Read nhưng có ngăn chặn lệnh insert/delete dữ liệu thỏa điều kiện thiết lập S. Giải quyết được Dirty Read và Unrepeatable Read và Phantom.

Thiết lập:

* Đặt lệnh thiết lập cần thiết trước khi bắt đầu giao tác.
* Thiết lập mặc định là Read Commited.
* Lệnh thiết lập:

Set transaction Isolation level Tên\_MứcCôLập

Ví dụ:

Set transaction Isolation level Read Commited Set transaction Isolation level Serializable

###### Các cấp độ khóa:

Đặt vấn đề: Mức cô lập là chưa đủ.

* Mức cô lập quyết định cách phát và giữ khóa S trong một transaction và có hiệu lực trên tất cả các thao tác đọc trong transaction đó.
* Thực tế, ta cần phát và giữ khóa S theo các cách khác nhau cho các thao tác đọc khác nhau trong cùng một transaction.
* Ngoài ra, ta cũng cần dùng nhiều dạng khóa linh động hơn là chỉ 1 loại khóa S đơn giản.

Khái niệm:

* Cấp độ khóa là các loại khóa khác nhau (không chỉ khóa S) được gắn vào từng table trong mệnh đề From của từng thao tác Select.
* Ngoài lệnh Select, cấp độ khóa còn có thể gắn vào các câu lệnh cập nhật, tuy nhiên ở đây ta chỉ quan tâm câu Select.

Các cấp độ khóa:

|  |  |
| --- | --- |
| * Read Uncommited / No lock * Read Commited (mặc định) * Repeatable * Serializable / Hold lock | * Updlock * Tablock * TablockX * ReadPast… |

Cách thiết lập:

Select …

From {Tab1 Alias1 with Lock\_mode [,…n]} [,…n] Where …

Ví dụ:

Select SV.HoVaTen, K.TenKhoa

From SinhVien SV with ReadCommite, Khoa K with Updlock Where SV.MaKhoa = K.MaKhoa And Year(NgaySinh) >= 1983

Phối hợp với Isolation Level:

* + Trong transaction luôn có các thao tác yêu cầu bảo vệ nghiêm ngặt và các thao tác ít yêu cầu bảo vệ nghiêm ngặt.
  + Dùng Isolation level ứng với yêu cầu bảo vệ ít nghiêm ngặt nhất.
  + Bổ sung lock mode vào các thao tác yêu cầu bảo vệ nghiêm ngặt hơn mức mà Isolation level đó cung cấp.

Khóa với dữ liệu trong cursor:

* + Nếu cursor là loại tĩnh (static) thì các đơn vị dữ liệu đọc ra sẽ được lock ngay khi vừa Open cursor.
  + Nếu cursor là loại động (dynamic) thì fetch đến đâu sẽ khóa đến đó.
  + Cách phát khóa và giữ khóa là do mức cô lập của connection và các lock mode trong câu select định nghĩa cursor quyết định.

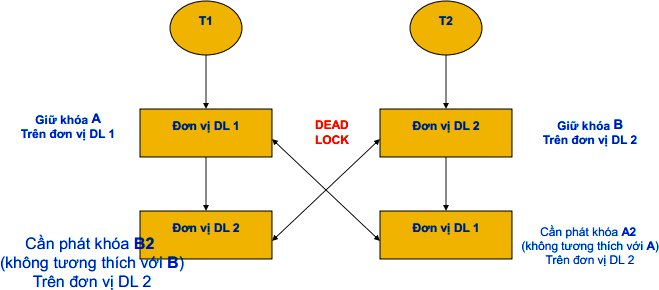
###### Dead lock:

Khái niệm:

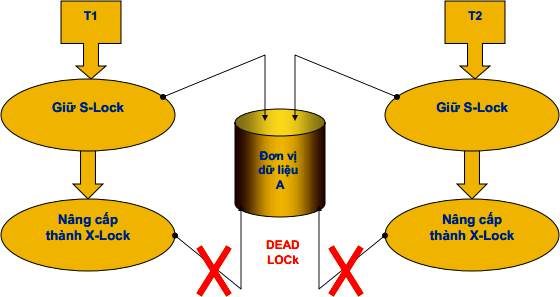
* + Khi xử lý đồng thời, không tránh khỏi việc transaction này phải chờ đợi transaction khác.
  + Nếu vì lý do gì đó mà hai transaction lại chờ lẫn nhau vĩnh viễn, không cái nào trong hai có thể hoàn thành được thì ta gọi đó là hiện tượng Dead Lock.

Phân loại:

* Cyclic Deadlock.
* Conversion Deadlock.



Hình 8.3: Cyclic Deadlock



**Hình 8.4: Conversion Deadlock**

Khi dead lock xảy ra:

* SQL Server sẽ chọn 1 trong 2 transaction gây dead lock để hủy bỏ, khi đó transaction còn lại sẽ được tiếp tục thực hiện cho đến khi hoàn tất.
* Transaction bị chọn hùy bỏ là transaction mà SQL ước tính chi phí cho phần việc đã làm được ít hơn transaction còn lại.

### TÓM TẮT

*Qua bài học này, học viên hiểu rõ khái niệm về giao tác và tính chất của nó. Nắm được cơ chế về khóa, các phương thức khóa cơ bản, các vấn đề liên quan đến truy xuất đồng thời..., hiểu được các mức cô lập cho giao tác.*

*Vì các phương thức khóa được thiết lập trên một đơn vị dữ liệu cụ thể, nên để hiểu được và các phương thức khóa trước tiên cần tìm hiểu về khái niệm đơn vị dữ liệu. Đơn vị dữ liệu có thể được chia thành nhiều cấp độ sau:*

* + *Một dòng dữ liệu.*
  + *Một trang (page) (8KB)*
  + *Một bảng (table) trong CSDL.*
  + *Một CSDL (database).*

*Khi có 2 transaction (của 2 connection khác nhau) có ít nhất 1 thao tác ghi trên cùng một đơn vị dữ liệu sẽ xảy ra tình trạng tranh chấp. Nếu để tình trạng tranh chấp này xảy ra sẽ dẫn đến những sai sót trên CSDL.*

*Để giải quyết các vấn đề tranh chấp nêu trên, hệ quản trị CSDL cần sử dụng các phương thức khóa, nhờ vậy mà khi có tranh chấp xảy ra, hệ quản trị CSDL có thể quyết định transaction nào được ưu tiên và transaction nào phải chờ.*

### CÂU HỎI ÔN TẬP

**Câu 1:** Tại sao lại cần các phương thức khóa?

**Câu 2:** Trình bày các phương thức khóa cơ bản?

**Câu 3:** Hãy cho biết các mức cô lập và các ưu/khuyết điểm của chúng?

**Câu 4:** Trong giờ thực hành, học viên làm bài thực hành giao tác Quản lý Thư viện (do GV cung cấp), từ đó hiểu rõ hơn về các mức cô lập:

1. So sách mức cô lập READ UNCOMMITTED và READ COMMITTED.
2. So sách mức cô lập READ COMMITTED và REPEATABLE READ.
3. So sách mức cô lập REPEATABLE READ và SERIALIZABLE

## BÀI 9: BẢO MẬT VÀ QUẢN TRỊ

Sau khi học xong bài này, học viên có thể:

* *Nắm được cách thức xác thực khi đăng nhập vào SQL Server.*
* *Cách tạo user và cách phân quyền cho từng user.*
* *Cách tạo group và add/delete user trong group.*
* *Hiểu được nhóm quyền CSDL và cách thức thêm/bớt quyền người dùng trên một CSDL.*

#### GIỚI THIỆU

Bảo mật và quản trị nhằm kiểm soát ai truy cập và truy cập dữ liệu gì. Người quản trị phải làm cho dễ dàng và thuận lợi để người dùng chỉ truy xuất được phần dữ liệu mà họ được phép, đồng thời phải ngăn chặn được những người không được phép truy xuất vào dữ liệu hoặc cố tình phá hoại dữ liệu.

#### XÁC THỰC SQL SERVER

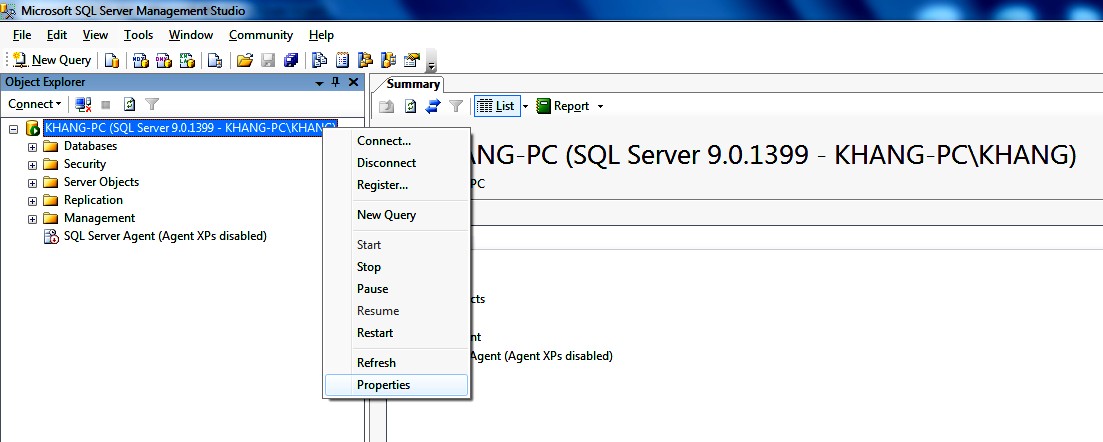
Xác thực là đảm bảo biết người dùng là ai. Sau khi đăng nhập SQL Server, người dùng có thể thực hiện bất cứ hành động được phép nào với đăng nhập của mình hoặc những hành động được phép với nhóm mà họ là thành viên.

Có 2 kiểu xác thực:

SQL Server and Windows Authentication mode và Windows Authentication mode.

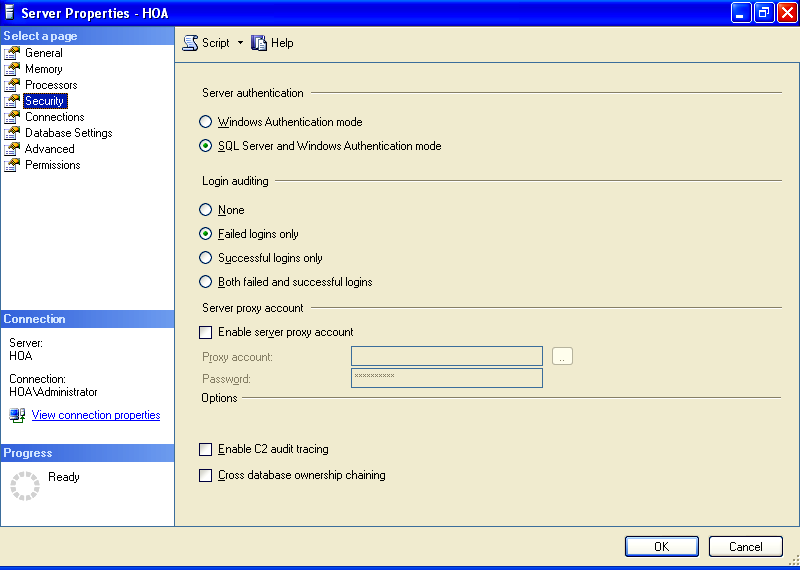
**Thiết lập kiểu xác thực SQL Server:**

* Khởi động Management Studio.
* Click phải vào server mà bạn muốn thay đổi kiểu xác thực  Chọn Propeties.



**Hình 9.1: Xác lập kiểu xác thực SQL Server**

* + Chọn Security  Chọn kiểu xác thực  Chọn OK.



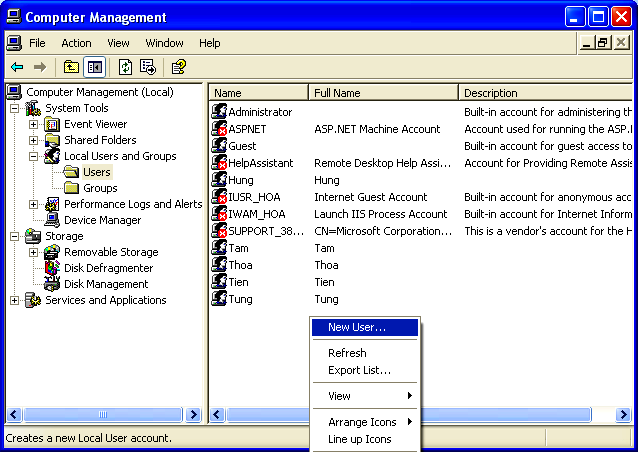
**Hình 9.2: Chọn kiểu xác thực**

#### TẠO USER TRONG WINDOWS

* Click phải vào My Computer  Chọn Manage.

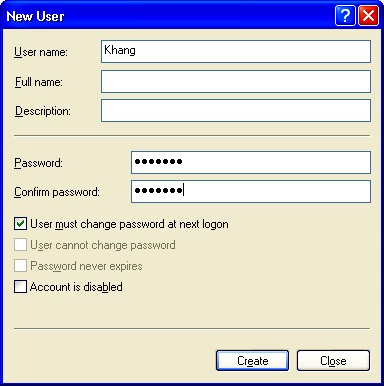


Hình 9.3: Tạo User

* Chọn System Tools  Local Users and Groups  Users  Click phải vào vùng trống  Chọn New User.

Hình 9.4: Chỉ định New User

* + Nhập tên User, Password… như hình sau  Chọn Create.



**Hình 9.5: Tạo password cho User**

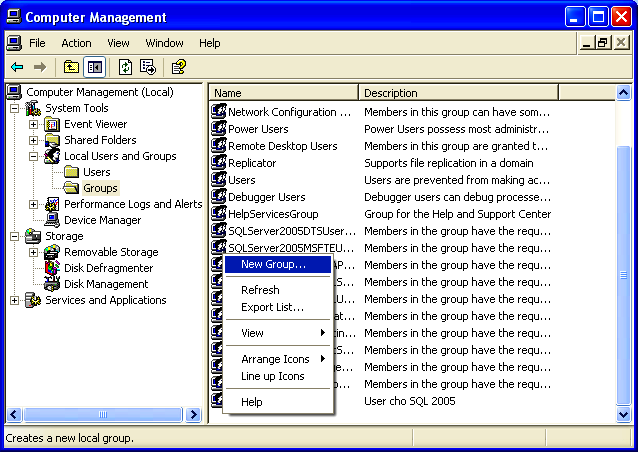
#### TẠO GROUP TRONG WINDOWS

- Click phải vào My Computer  Chọn Manage.



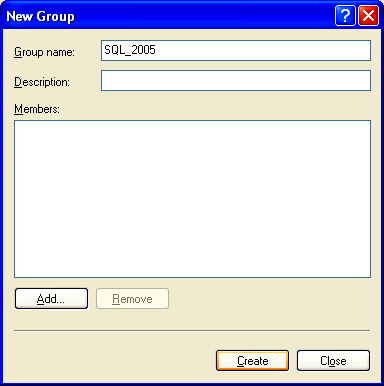
Hình 9.6: Tạo Group trong Windows

* Chọn System Tools  Local Users and Groups  Groups  Click phải vào vùng trống  Chọn New Group.



Hình 9.7: Chỉ định New Group

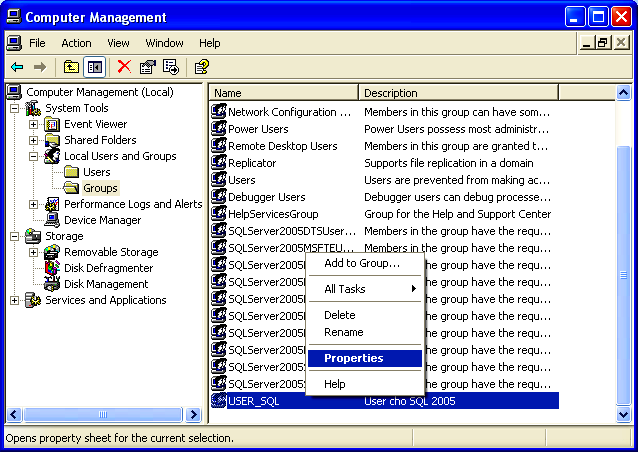
* Nhập tên nhóm  Chọn Create.



Hình 9.8: Chỉ định tên Group

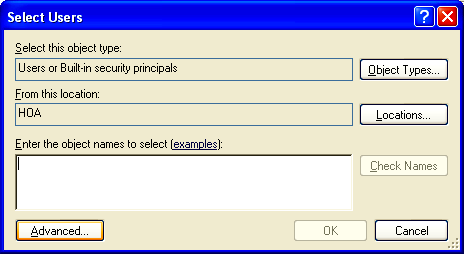
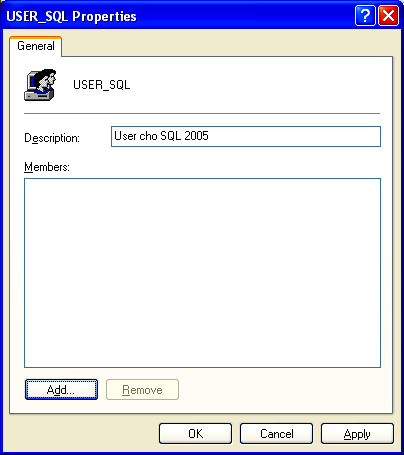
**Thêm User vào Group:**

* + Click phải vào Group cần thêm User  Chọn Propeties.



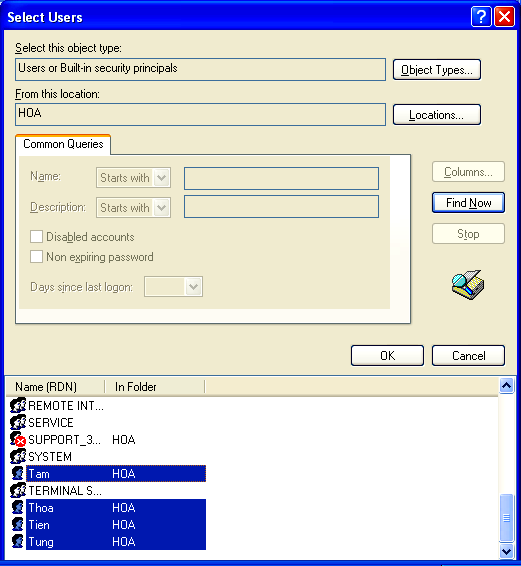
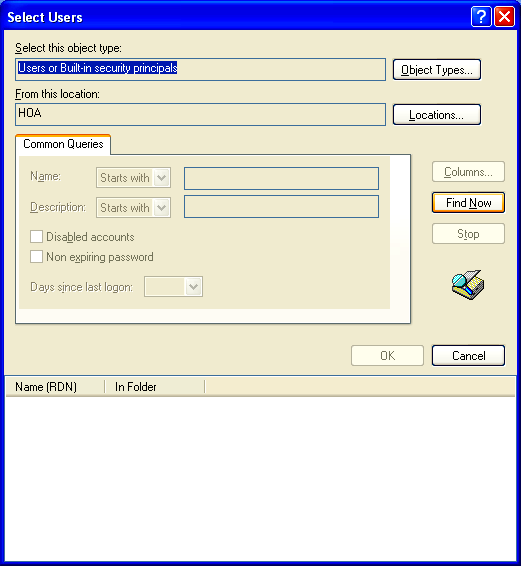
Hình 9.9: Chọn Properties

* + Chọn Add  Chọn Advanced.



Hình 9.10: Thêm User vào Group

* Find Now  Chọn các user cần thêm  Chọn OK.

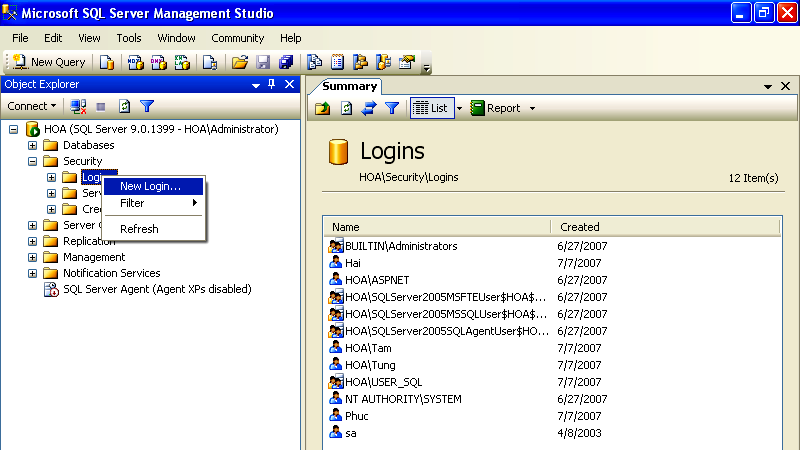


Hình 9.11: Chọn User đƣa vào Group

* Click OK và tiếp tục chọn OK.

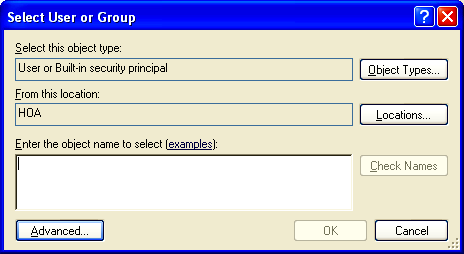
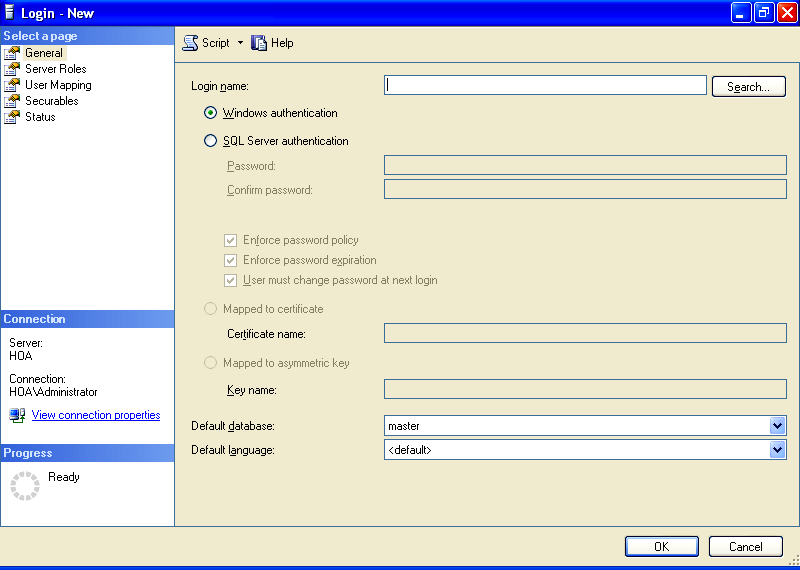
Tạo đăng nhập Windows:

* Mở Security cấp Server  Click phải vào Logins  Chọn New Login.



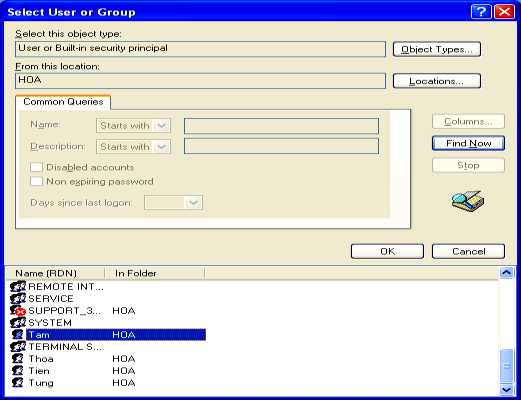
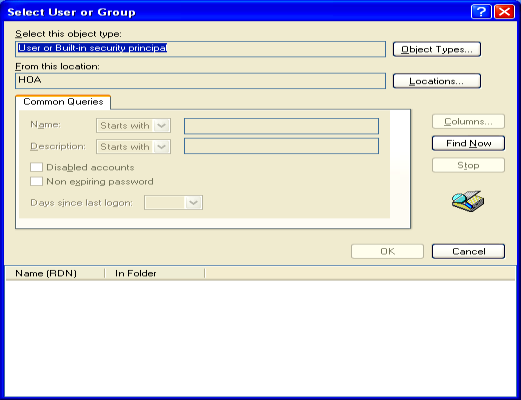
Hình 9.12: Tạo đăng nhập cho Windows

* + Search  Advanced.



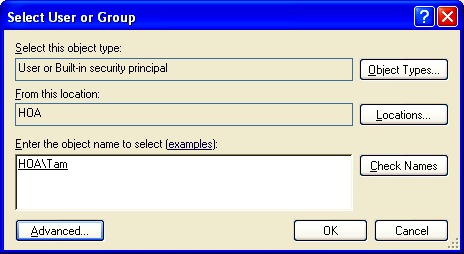
Hình 9.13: Chọn Advanced sau khi chọn Seach

* + Find Now  Chọn user  Chọn OK.



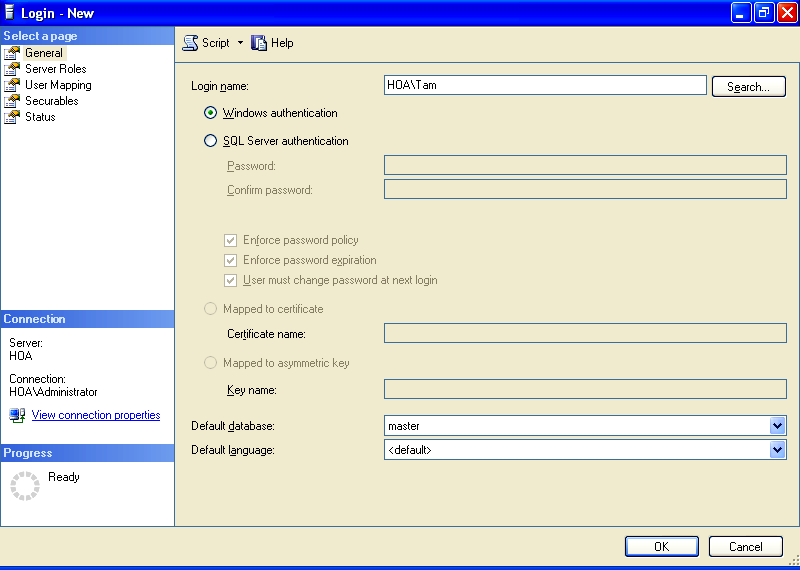
Hình 9.14: Chọn User

* + Chọn OK.



Hình 9.15: Xác nhận OK sau khi chọn User

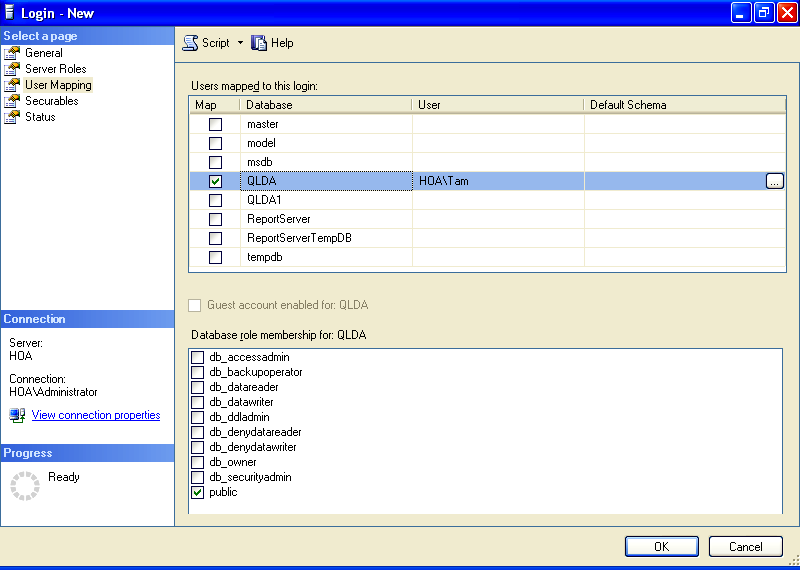
* Chọn Windows Authentication.



Hình 9.16: Chọn hình thức đăng nhập

* Chọn User Mapping  Check chọn các Database mà user được phép truy cập 

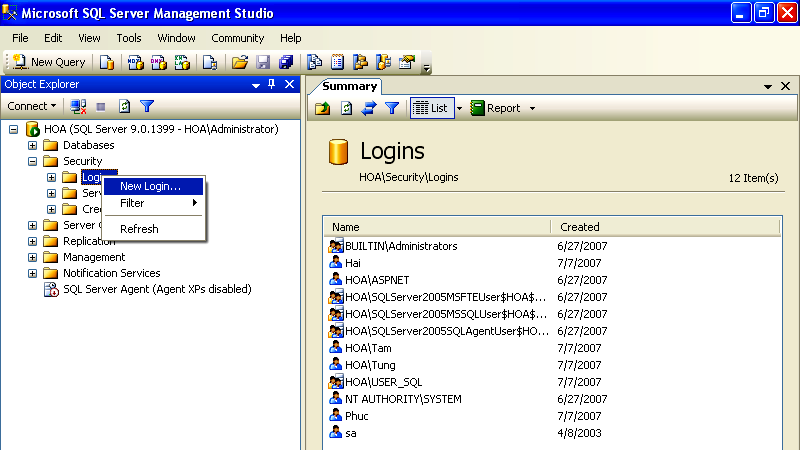
Chọn OK.



Hình 9.17: Chọn các Database

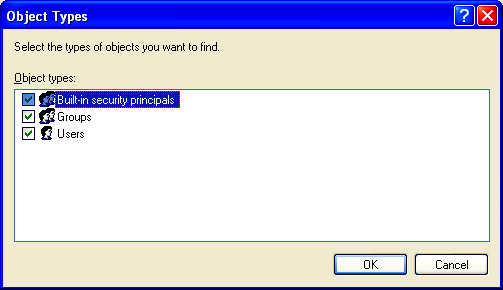
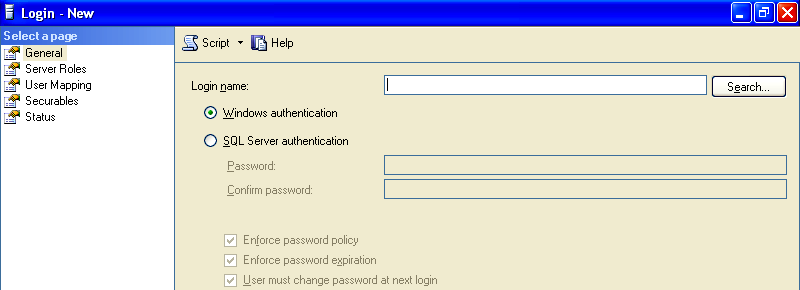
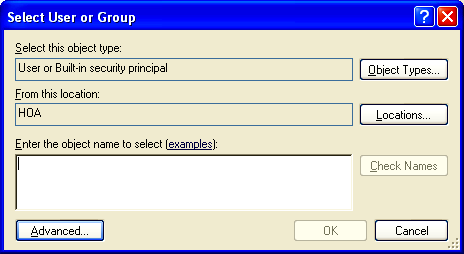
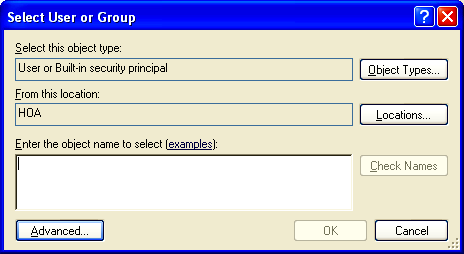
**Tạo đăng nhập Windows cho nhóm:**

Mở Security cấp Server  Click phải vào Logins  Chọn New Login.



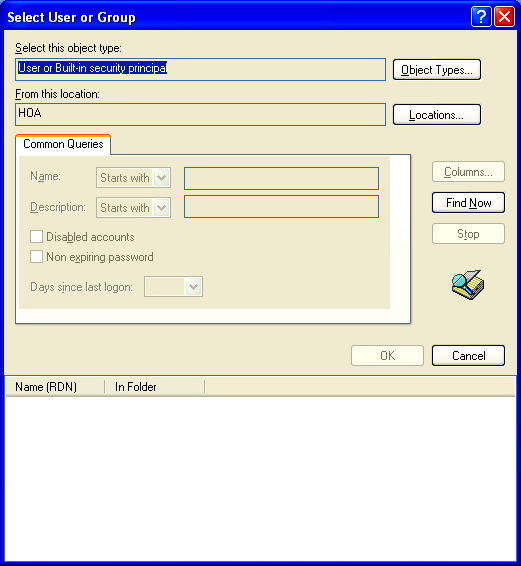
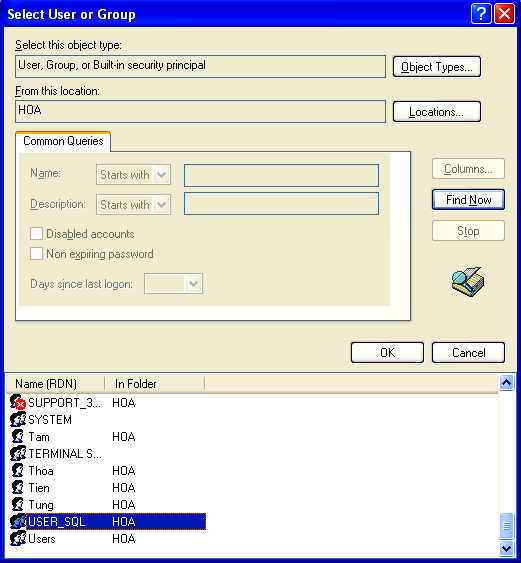
Hình 9.18: Tạo đăng nhập Windows cho nhóm

* + Search  Object Types  Check chọn Groups  OK  Advanced.



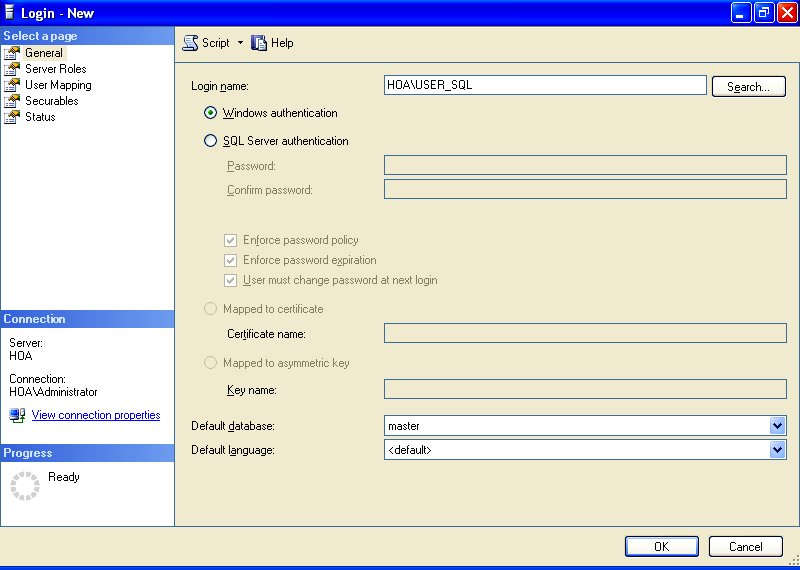
Hình 9.19: Chọn Group

* Find Now  Chọn nhóm  Chọn OK.



Hình 9.20: Chỉ định nhóm

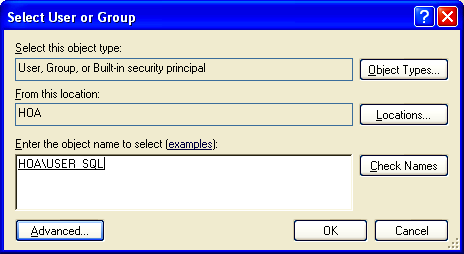
* Chọn OK.
* Chọn Windows Authentication.



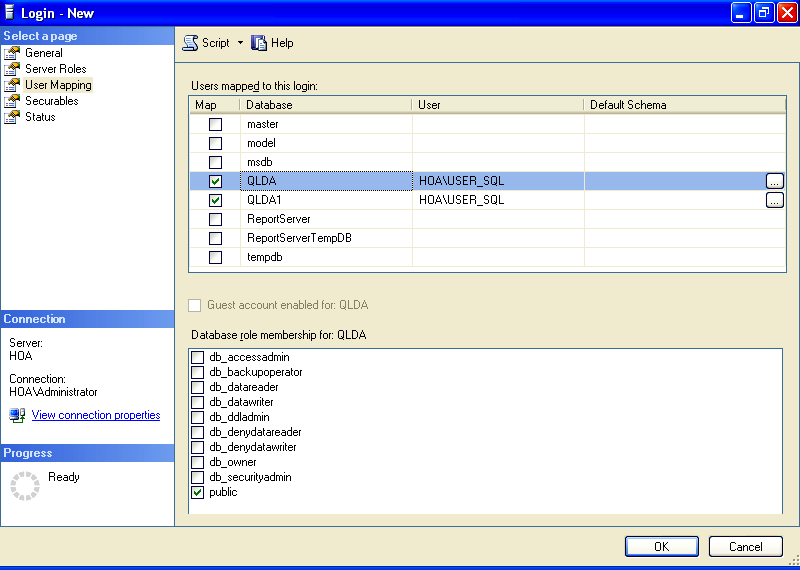
Hình 9.21: Chỉ định hình thức đăng nhập

* + Chọn User Mapping  Check chọn các Database mà nhóm được phép truy cập 

Chọn OK.



Hình 9.22: Xác nhận nhóm đã chọn

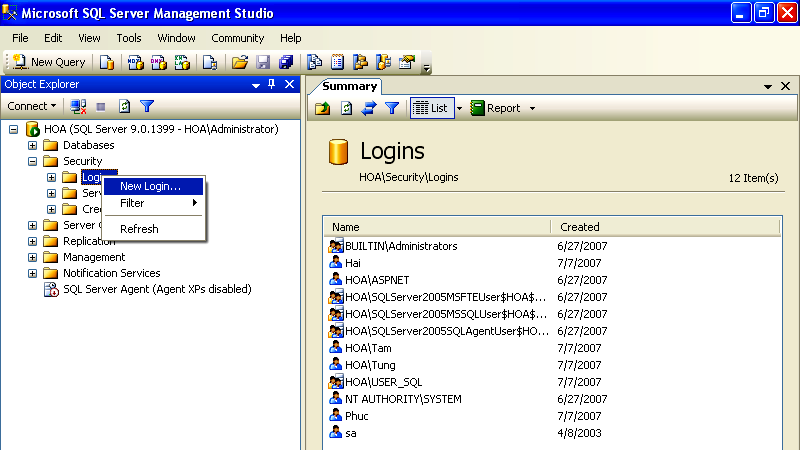


**Hình 9.23: Chọn các Database mà nhóm đƣợc phép truy cập**

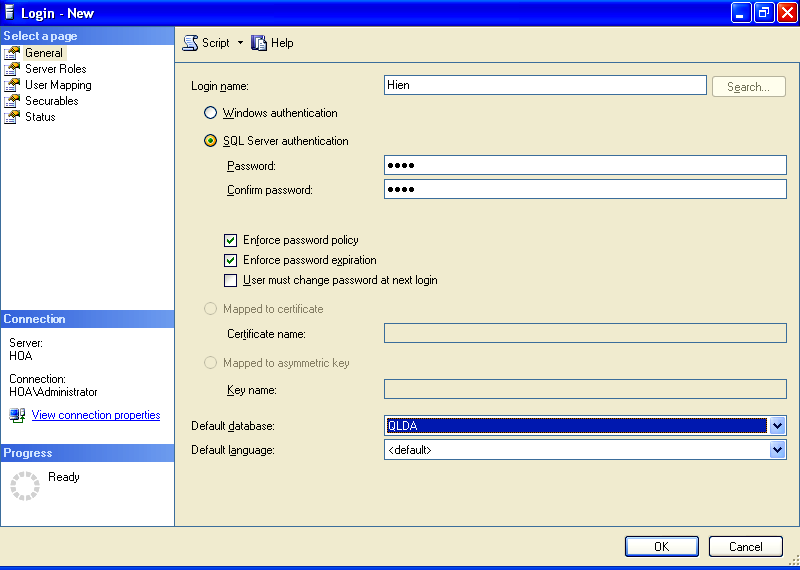
Tạo đăng nhập SQL Server:

* Mở Security cấp Server  Click phải vào Logins  Chọn New Login.
* Nhập tên đăng nhập, password… như hình minh họa bên dưới:
* Chọn User Mapping  Check chọn các Database mà user được phép truy cập 

Chọn OK.

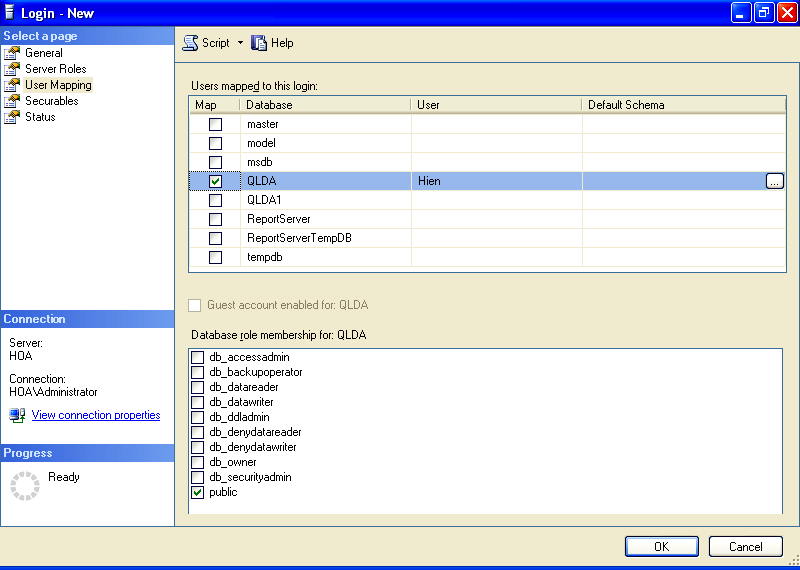


Hình 9.24: Tạo đăng nhập SQL Server



**Hình 9.25: Chỉ định tên và hình thức đăng nhập**

**Hình 9.26: Chỉ định Database**



#### VAI TRÒ (ROLE)

SQL Server phân làm 3 nhóm quyền chính:

* Nhóm quyền Server (Server Roles): nhóm này cho phép người dùng thành viên thực hiện các tác vụ quản trị cấp Server.
* Nhóm quyền CSDL (Database Roles): nhóm này cho phép người dùng thành viên thực hiện các tác vụ liên quan đến CSDL.
* Nhóm quyền CSDL do người dùng định nghĩa: cho phép người dùng tự định nghĩa nhóm quyền riêng cho mình.
* sysadmin: có toàn quyền
* securityadmin: gồm các quyền liên quan đến user
* dbcreator: các quyền liên quan đến Database
* …

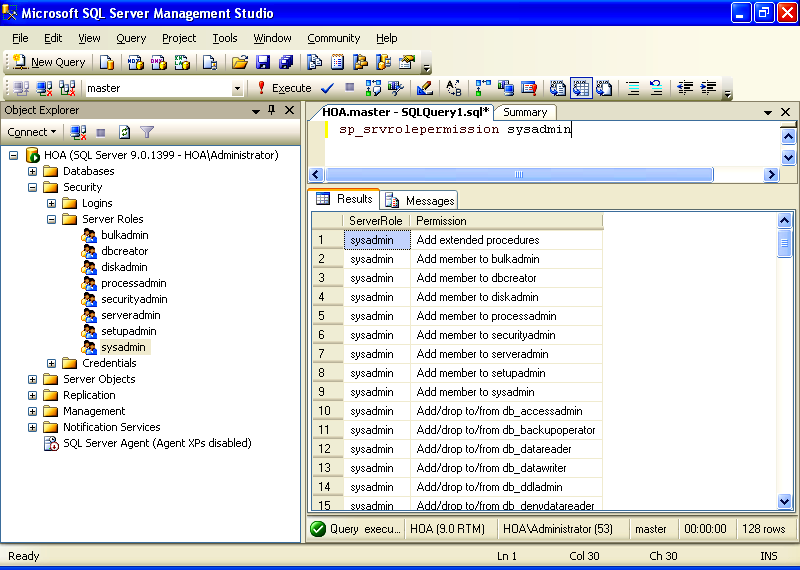
Để xem các quyền của nhóm quyền Server, ta dùng lệnh:

sp\_srvrolepermission 'Ten\_Role'

Ví dụ:

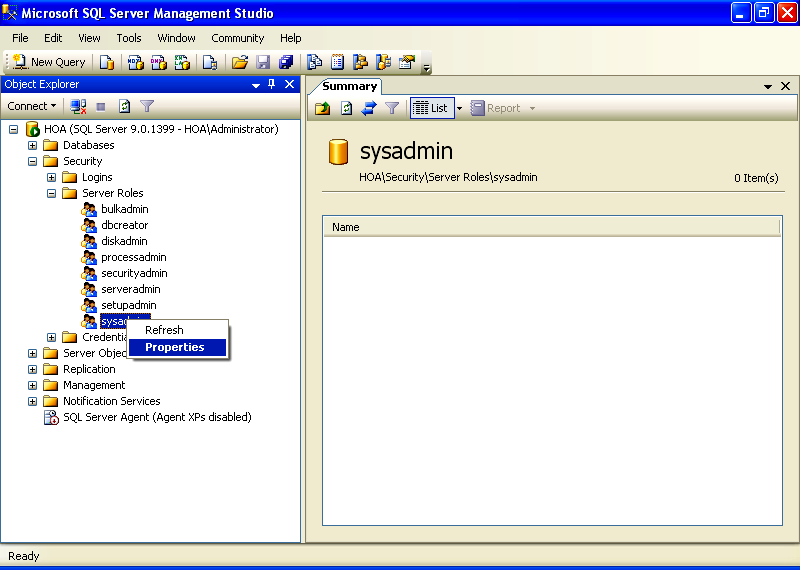
sp\_srvrolepermission sysadmin

Các quyền của sysadmin:



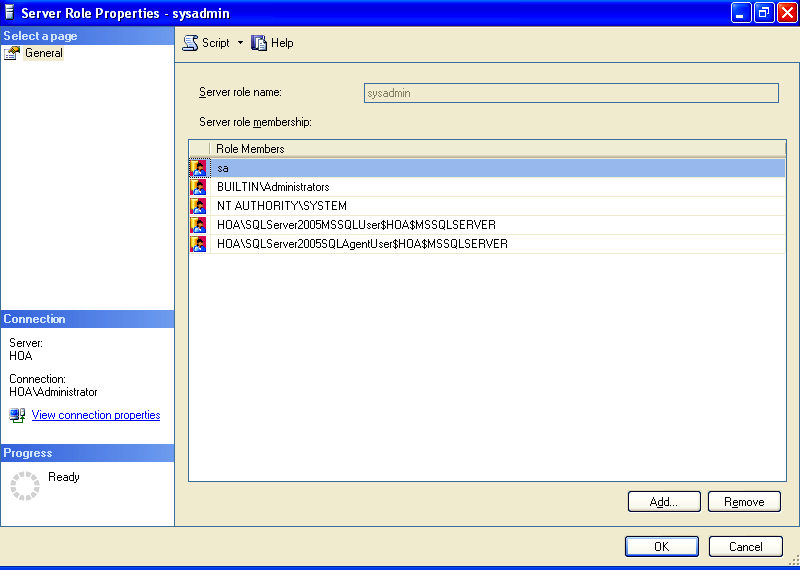
Hình 9.27: Các quyền của sysadmin

**Thêm ngƣời dùng vào nhóm quyền Server:**

* + Mở Security cấp Server  Server Roles  Click phải vào nhóm muốn cấp  Chọn Propeties.

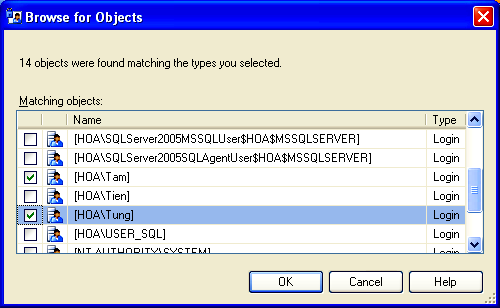
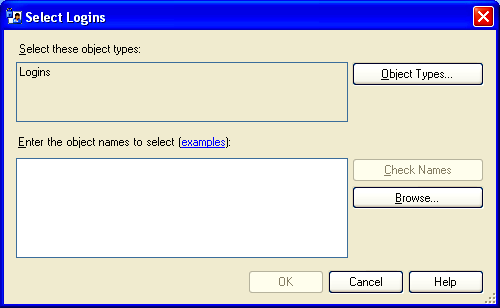
Hình 9.28: Thêm ngƣời dùng vào nhóm quyền Server

* Chọn Add.



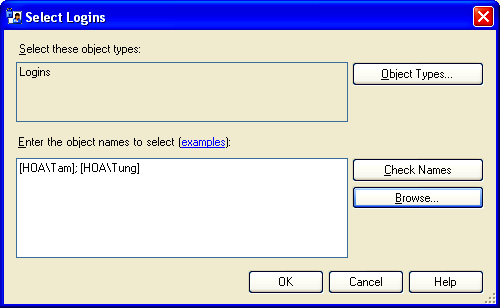
Hình 9.29: Chỉ định vai trò

* Browse  Check chọn những user cần thêm  OK.



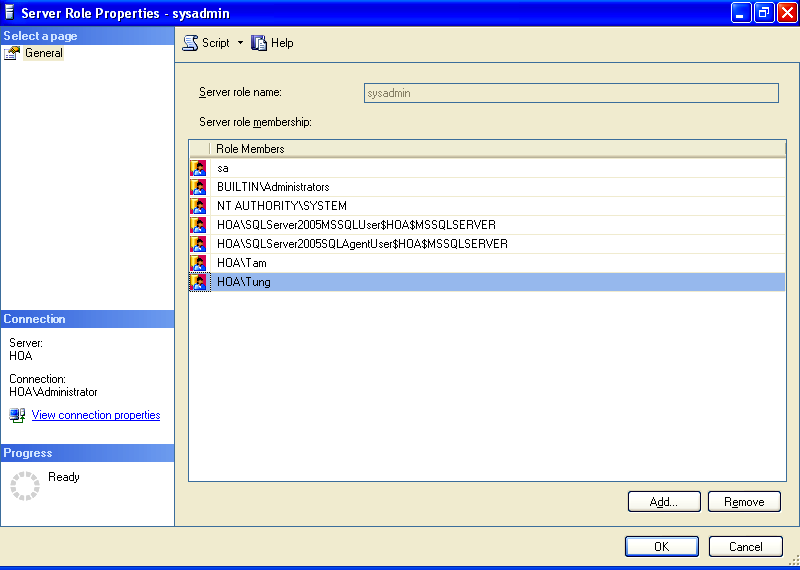
Hình 9.30: Chỉ định user

* Chọn OK.



Hình 9.31: Xác nhận sau khi chọn user

* + Tiếp tục chọn OK.



Hình 9.32: Xác nhận hoàn tất

**Nhóm quyền CSDL:**

* + db\_owner: quyền sở hữu Database
  + db\_dlladmin: cho phép thay đổi cấu trúc Database
  + db\_datareader: cho phép đọc dữ liệu
  + db\_datawrite: cho phép ghi dữ liệu

- …

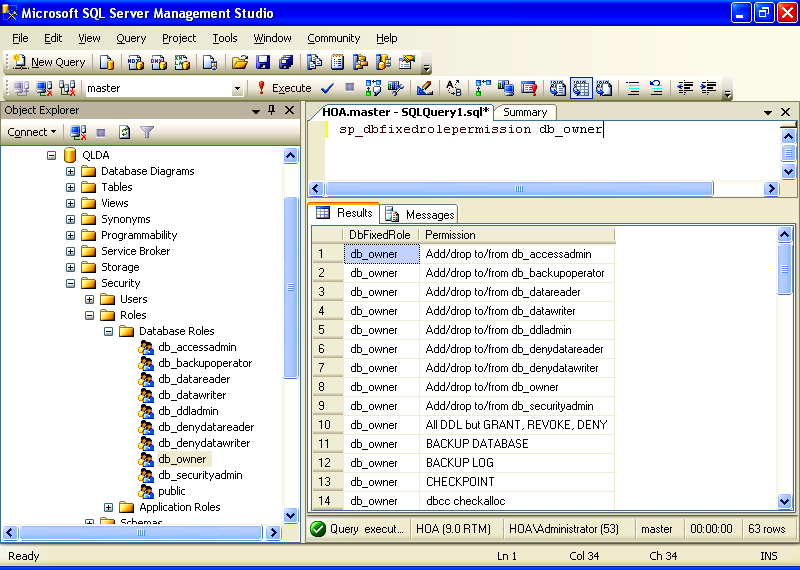
Để xem các quyền của nhóm quyền CSDL, ta dùng lệnh:

sp\_dbfixedrolepermission 'Ten\_Role'

Ví dụ:

sp\_dbfixedrolepermission db\_owner

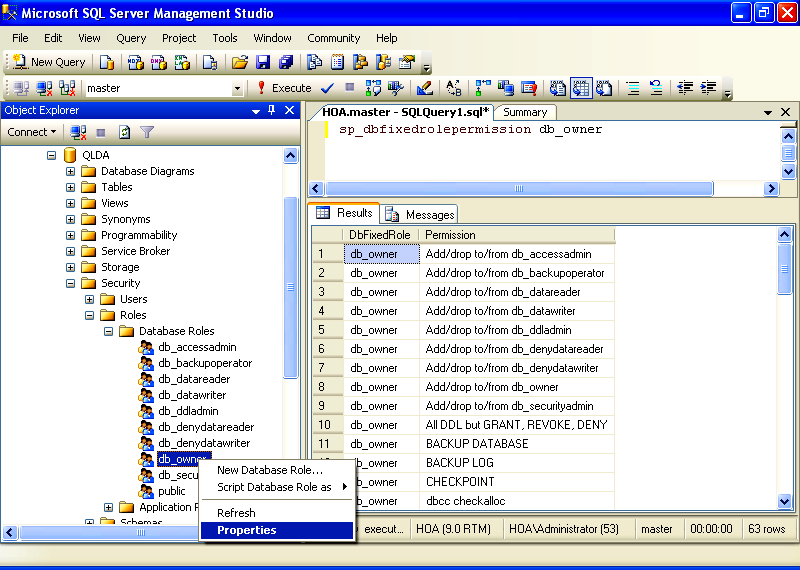
* + Các quyền của db\_owner:



Hình 9.33: Các quyền của db\_owner

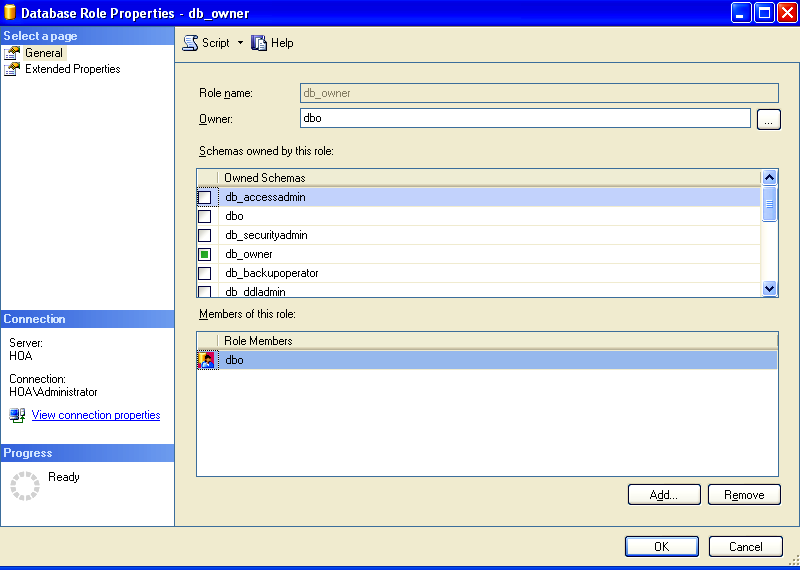
**Thêm ngƣời dùng vào nhóm quyền CSDL:**

* Mở CSDL bạn muốn thêm người dùng (QLDA)  Security  Roles  Database Roles  Click phải vào nhóm muốn cấp  Chọn Propeties.



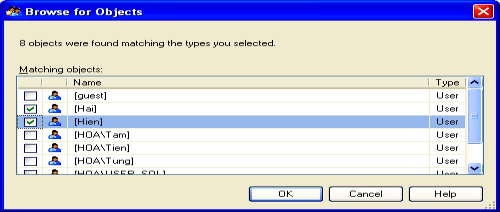
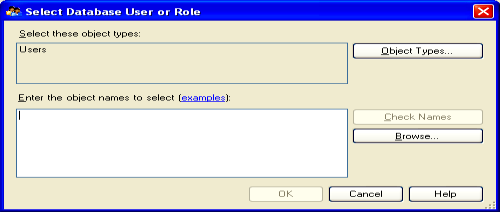
Hình 9.34: Thêm ngƣời dùng vào nhóm quyền CSDL

* + Chọn Add.



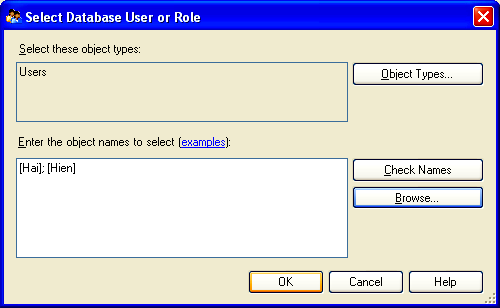
Hình 9.35: Chỉ định thêm ngƣời dùng vào nhóm quyền CSDL

* + Browse  Check chọn những user cần thêm  OK.



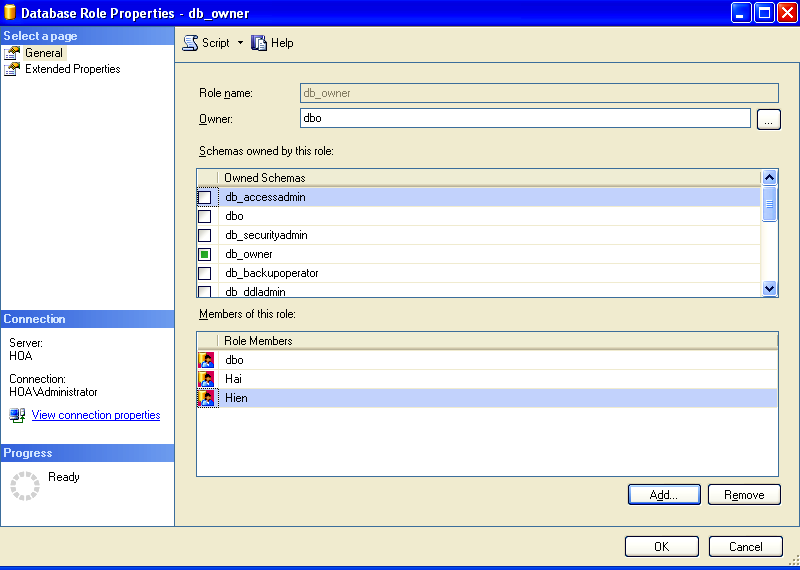
Hình 9.36: Chọn những user cần thêm

* + Chọn OK.



Hình 9.37: Xác nhận những user cần thêm

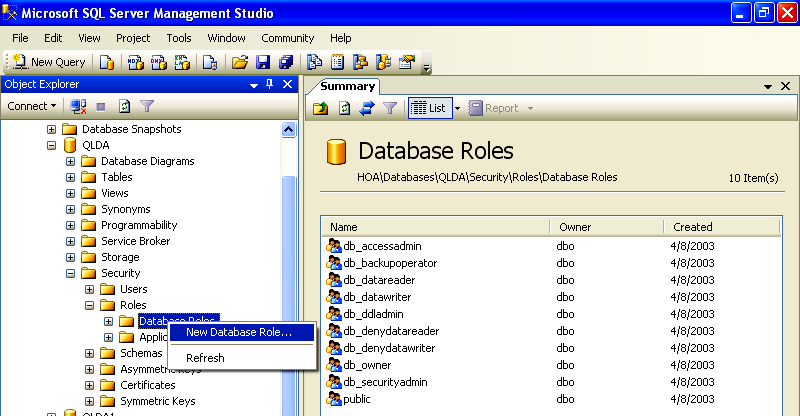
* Tiếp tục chọn OK.



Hình 9.38: Xác nhận hoàn tất

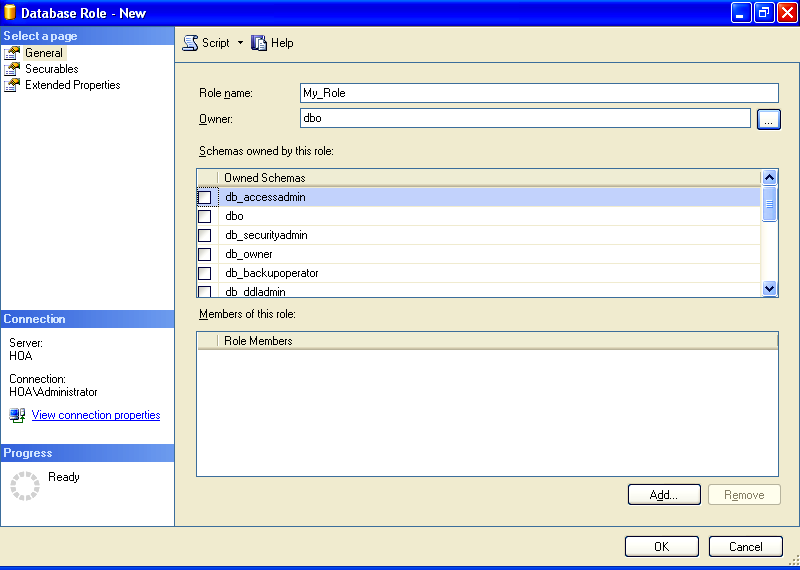
**Tạo nhóm quyền do ngƣời dùng định nghĩa:**

* Mở CSDL (QLDA) muốn tạo nhóm quyền  Security  Roles  Click phải vào Database Roles  New Database Role.



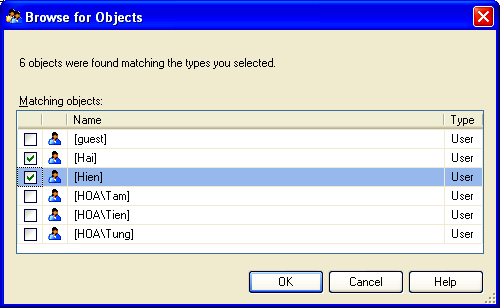
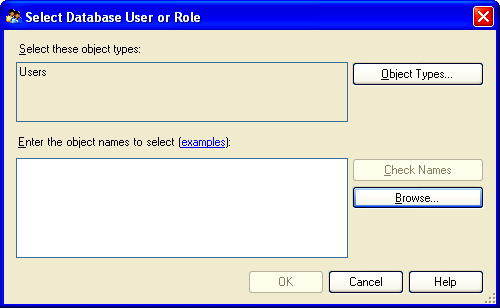
Hình 9.39: Tạo nhóm quyền do ngƣời dùng định nghĩa

* + Nhập tên Role  Chọn Owner  Add.



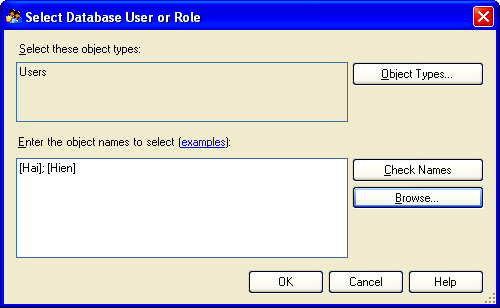
Hình 9.40: Nhập tên Role và chỉ định Owner

* + Browse  Chọn các user cần thêm vào nhóm  OK.



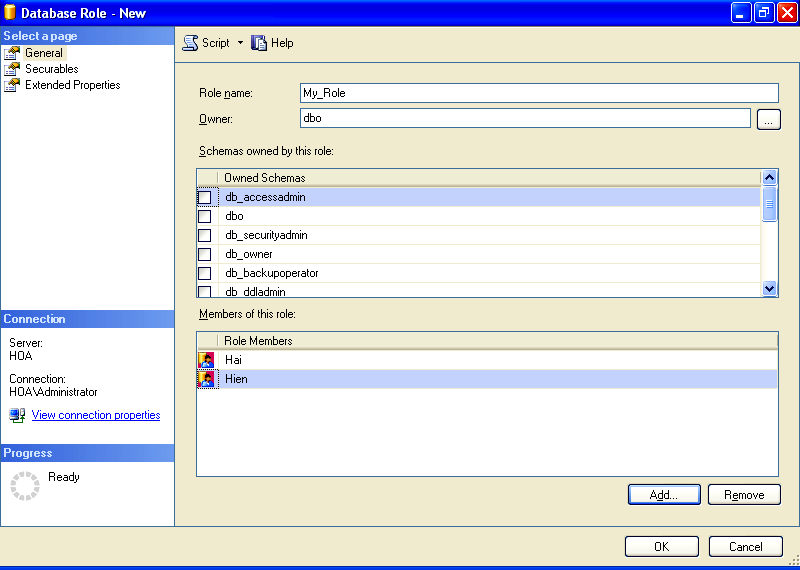
Hình 9.41: Chọn các user

* + Chọn OK.



Hình 9.42: Xác nhận các user

* Tiếp tục chọn OK.



**Hình 9.43: Hoàn tất việc tạo nhóm quyền**

#### QUYỀN

SQL Server chia thành 3 loại quyền:

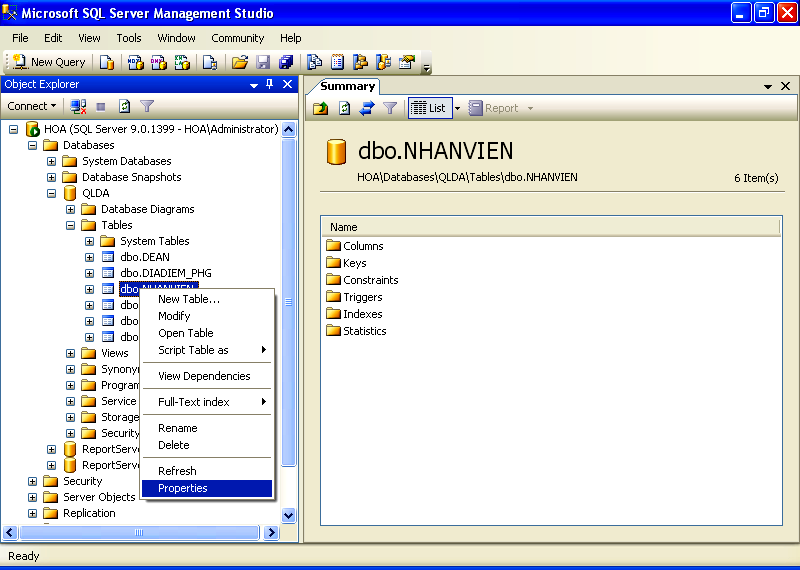
* Quyền phát biểu: cho phép người dùng tạo CDSL mới, tạo các đối tượng trong phạm vi CSDL như: CREATE DATABASE, CREATE TABLE, CREATE RULE, CREATE PROCEDURE…
* Quyền về đối tượng: cho phép người dùng thực hiện các tác vụ trên từng đối tượng như: SELECT, INSERT, UPDATE, DELETE…
* Quyền thừa hưởng: đây là những quyền người dùng có được do là thành viên của nhóm hoặc bởi vì người dùng là chủ đối tượng.

Các phát biểu:

* GRANT: cấp quyền
* WITH GRANT: cho phép người dùng cấp quyền cho người dùng khác
* DENY: thu hồi quyền Các quyền:
  + Alter: cho phép thay đổi cấu trúc của bảng
  + Control: cho phép quản trị đối tượng
  + Select: cho phép xem dữ liệu
  + Delete: cho phép xóa dữ liệu
  + Insert: cho phép thêm dữ liệu vào bảng Cấp quyền lên đối tượng, có 2 cách:
  + Gán quyền cho người dùng hoặc nhóm quyền CSDL lên đối tượng (cách 1)
  + Gán quyền cho tất cả các đối tượng đến người dùng hoặc nhóm quyền CSDL (cách 2)

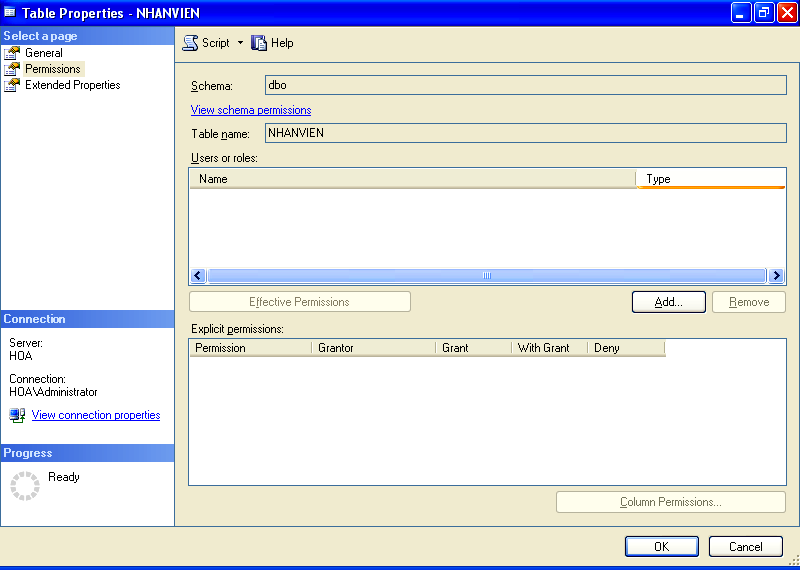
Cấp quyền lên đối tƣợng cách 1:

* + Click phải vào đối tượng cần cấp quyền  Chọn Properties.



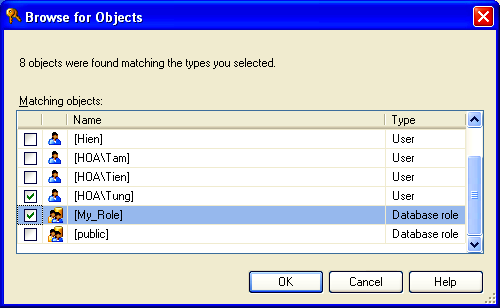
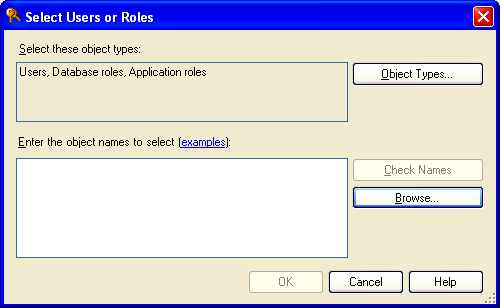
Hình 9.44: Cấp quyền lên đối tƣợng cách 1

Permission  Add



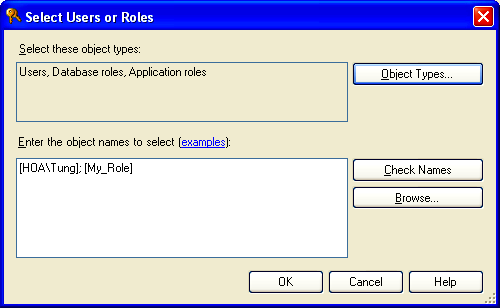
Hình 9.45: Chỉ định quyền

* Browse  Chọn các user cần cấp quyền  Chọn OK.



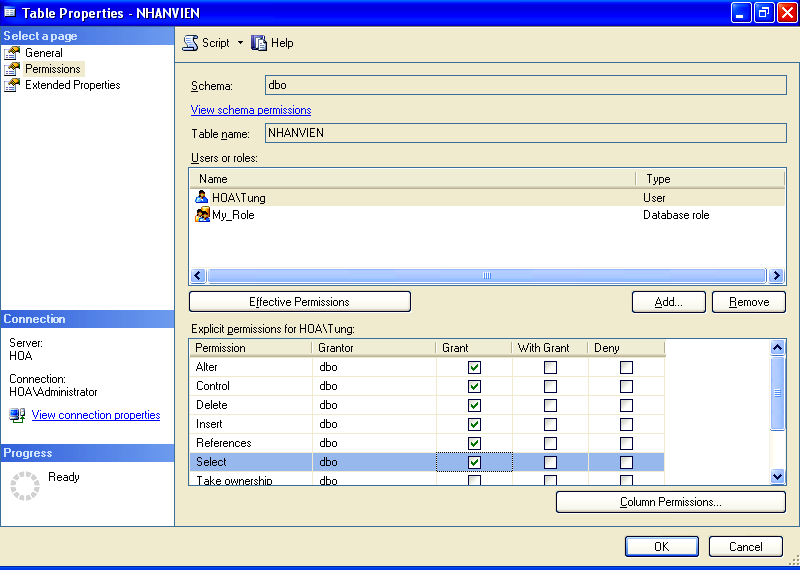
Hình 9.46: Chỉ định user cần cấp quyền

* + Chọn OK.



Hình 9.47: Xác nhận chọn user

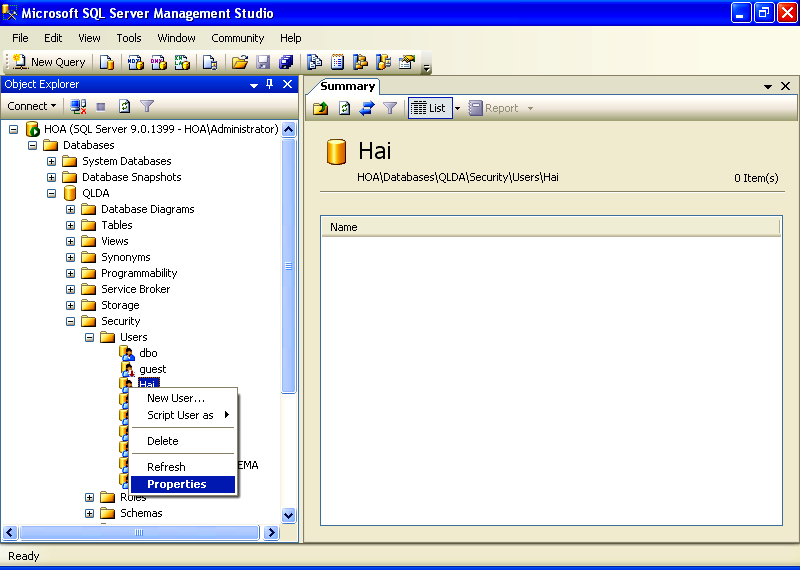
* + Cấp các quyền cho từng người dùng  Chọn OK.



Hình 9.48: Hoàn tất việc cấp quyền

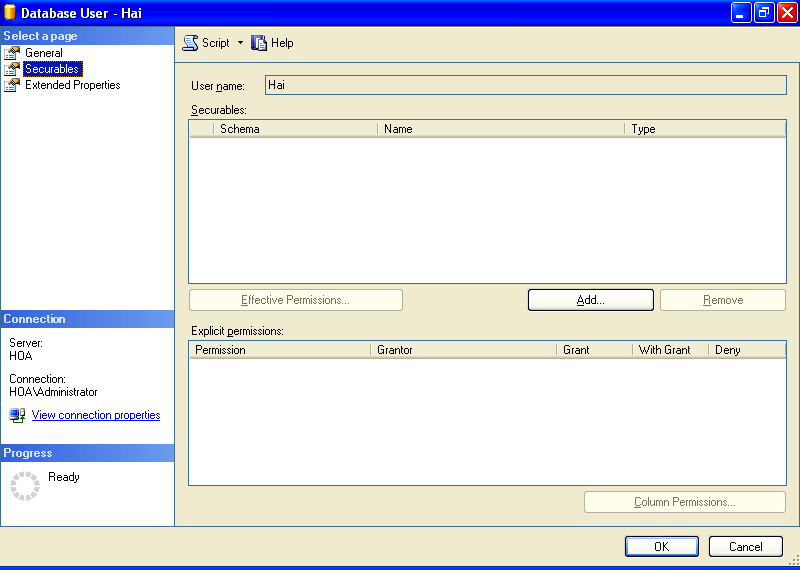
**Cấp quyền lên đối tƣợng cách 2:**

* + Mở CSDL muốn cấp quyền (QLDA)  Security  Users hoặc Roles  Click phải vào người dùng cần cấp quyền  Properties.



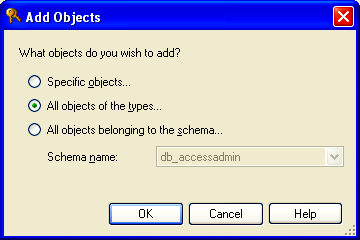
Hình 9.49: Cấp quyền lên đối tƣợng cách 2

* Securables  Chọn Add.



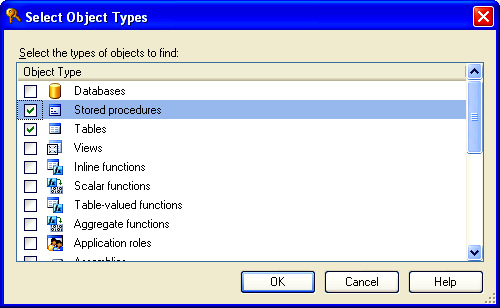
Hình 9.50: Chỉ định Add

* + Chọn All objects of the types  Chọn OK.



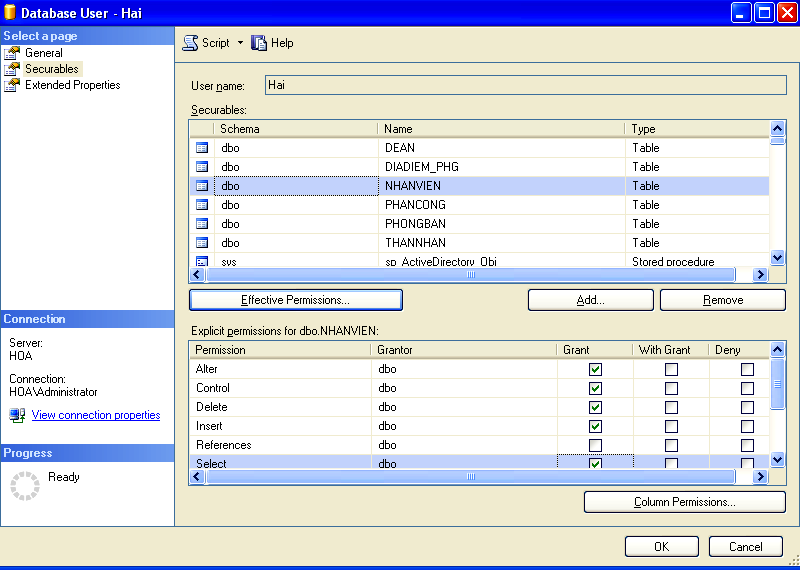
Hình 9.51: Chỉ định kiểu Object

Chọn các loại đối tượng cần cấp quyền  Chọn OK.



Hình 9.52: Chọn các kiểu Object

* + Chọn đối tượng cần cấp quyền  Check vào các quyền cần cấp  Chọn OK.



**Hình 9.53: Chỉ định các quyền cần cấp**

### TÓM TẮT

*Phân quyền là sự phân chia khả năng quản trị và sử dụng hệ quản trị CSDL SQL Server. Hình thành theo cơ cấu:*

* + *Người đăng nhập (login)*
  + *Người dùng (user)*
  + *Quyền hạn (permission)*
  + *Nhóm quyền (role)*
  + *Người đăng nhập được thể hiện là mỗi một người dùng với một số quyền hạn ứng với một dữ liệu.*

*Mỗi Database nên có một hệ thống bảo mật đáng tin cậy (reliable security system) để giám sát mọi hoạt động cũng như các thông tin cần được xem và chỉnh sửa. Một hệ thống bảo mật đáng tin cậy phải bảo đảm được việc bảo vệ dữ liệu bất kể việc user đã dùng cách nào để truy xuất vào Database. SQL áp dụng các quyền bảo mật vào các mức: mức Database, mức các đối tượng và mức các cột của bảng.*

*User có thể truy xuất vào Databasethông qua 1 account đăng nhập (login ID) hợp lệ, nhờ đó user có khả năng kết nối vào database server. Quá trình này được gọi là authentication (xác thực).*

*Tài khoản đăng nhập (Login ID) sẽ được ánh xạ với tài khoản user (user ID) để cho phép user được quyền truy xuất trong một database. Quá trình này gọi là authorization (cấp phép) (hay permission validation). User sẽ không thể truy xuất vào database ngay cả khi họ có tài khoản đăng nhập hợp lệ.*

### CÂU HỎI ÔN TẬP

**Câu 1:** Tại sao cần thiết phải tạo người dùng và phân quyền cho người dùng? Từ kiến thức đã học, học viên áp dụng làm bài thực hành số 4.

## BÀI 10: MÃ HÓA DỮ LIỆU TRONG SQL (BÀI ĐỌC THÊM)

Sau khi học xong bài này, học viên có thể:

* *Nắm được lý do cần phải mã hóa thông tin.*
* *Hiểu được các kỹ thuật mã hóa của SQL Server.*
* *Vận dụng được một kỹ thuật mã hóa và giải mã cụ thể: mã hóa bằng mật khẩu.*

#### ĐẶT VẤN ĐỀ

Lý do cần phải mã hóa thông tin:

* Mã hóa là một phương pháp quan trọng nhằm bảo mật dữ liệu.
* Những dữ liệu nhạy cảm như số CMND, số thẻ tín dụng, mật khẩu… cần phải được bảo vệ trước vô vàn mối nguy hiểm tấn công hiện nay. Phần lớn các CSDL quan hệ thường làm việc trên dữ liệu của nhiều dòng mẫu tin, còn gọi là một bộ các mẫu tin. Ví dụ lệnh SELECT kết quả luôn trả về nhiều dòng dữ liệu hơn là một dòng dữ liệu.
* Trong SQL Server 2000 bạn có thể tự tạo các hàm của riêng mình hoặc sử dụng các DLL ngoài để mã hóa dữ liệu.
* Trong SQL Server 2005, các hàm và phương thức này được mặc định cho phép sẵn.

#### CÁC KỸ THUẬT MÃ HÓA

SQL Server 2005 cung cấp các kỹ thuật sau để mã hóa dữ liệu:

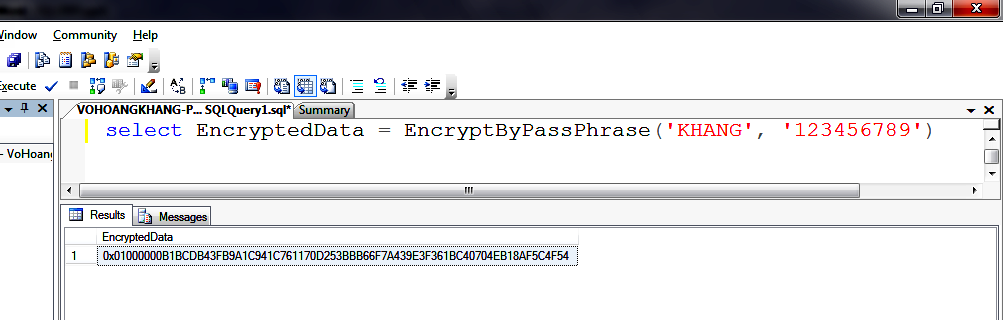
* Mã hóa bằng mật khẩu
* Mã hóa khóa đối xứng
* Mã hóa khóa không đối xứng
  + Mã hóa chứng nhận

Trong bài này, giới thiệu cách sử dụng kỹ thuật mã hóa bằng mật khẩu và phương pháp giải mã nó. SQL Server 2005 cung cấp 2 hàm cho việc mã hóa: một cho việc mã hóa và một cho việc giải mã. “Mã hóa bằng mật khẩu” là phương pháp mã hóa dữ liệu cơ bản thông qua mật khẩu. Dữ liệu có thể được giải mã nếu nhập đúng mật khẩu đã sử dụng khi mã hóa. Chúng ta sẽ thử một ví dụ về việc mã hóa và giải mã dữ liệu bằng kỹ thuật mã hóa thông qua mật khẩu.

Xét ví dụ sau:

select EncryptedData = EncryptByPassPhrase(‘KHANG', '123456789')

Kết quả:



Hình 10.1: Ví dụ về mã hóa

Bây giờ, chúng ta sẽ thực thi 3 lần hàm Encryptbypassphrase trên theo ví dụ sau:

declare @count int

declare @SocialSecurityNumber varchar(500) declare @password varchar(12)

set @count =1 while @count<=3 begin

set @SocialSecurityNumber = '123456789'

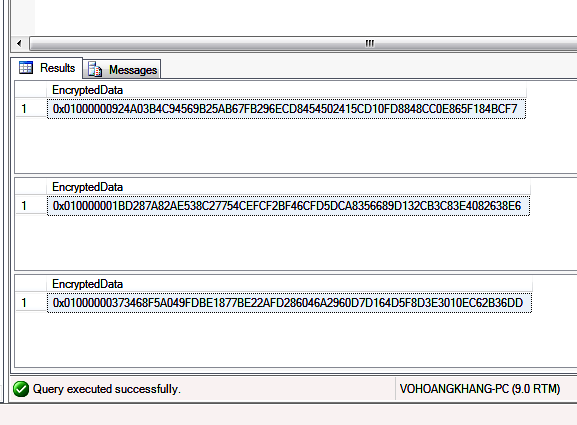
end

set @Password = ‘KHANG'

select EncryptedData = EncryptByPassPhrase(@password, @SocialSecurityNumber )

set @count=@count+1

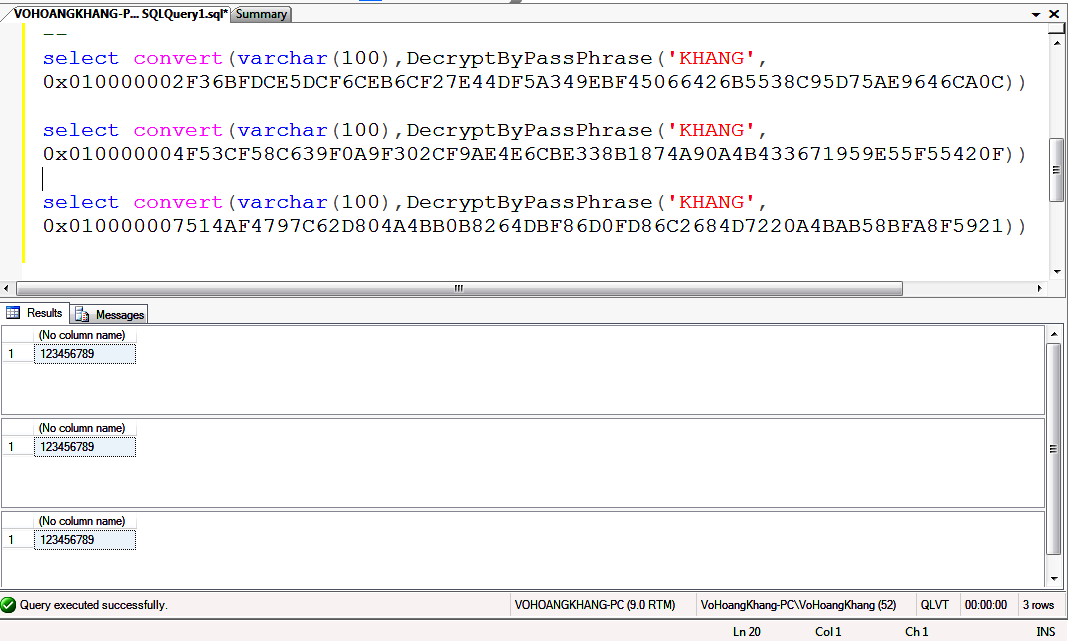
Kết quả:



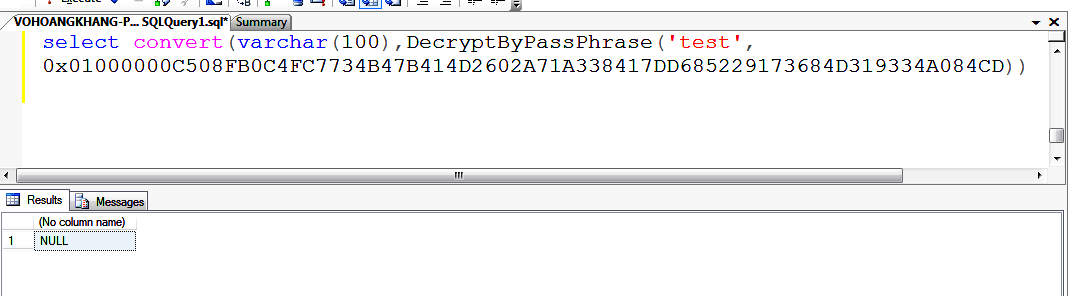
Hình 10.2: Kết quả sau khi mã hóa

Lưu ý:

“123456789” ở đây có thể là số thẻ tín dụng và “KHANG” là mật khẩu. Kết quả của Encryptbypassphrase sau mỗi lần thực thi hàm là khác nhau. Tuy nhiên, khi chúng ta giải mã dữ liệu thì nó vẫn ra kết quả như ban đầu trước khi mã hóa. Bây giờ chúng ta sẽ thử giải mã dữ liệu đã được mã hóa ở trên với hàm DecryptByPassPhrase:



Hình 10.3: Kết quả sau khi giải mã đúng

Thử giải mã dữ liệu đã được mã hóa với một mật khẩu khác. Thực thi theo câu lệnh sau:

Hình 10.4: Kết quả sau khi giải mã sai

Kết quả cho thấy SQL Server trả lại giá trị NULL nếu mật khẩu sai.

Bây giờ, chúng ta sẽ thử tạo một bảng chứa số thẻ tín dụng và số CMND, sau đó sẽ mã hóa dữ liệu này thông qua phương pháp mã hóa mật khẩu:

USE [master] Go

IF EXISTS (SELECT name FROM sys.databases WHERE name = N'Customer DB')

DROP DATABASE [Customer DB]

Go

Create database [Customer DB]

Go

use [Customer DB] Go

Create table [Customer data] ([customer id] int,

[Credit Card Number] bigint, [Social Security Number] bigint) Go

Chèn dữ liệu vào bảng:

insert into [Customer data] values (1, 1234567812345678, 123451234)

insert into [Customer data] values (2, 1234567812345378, 323451234)

insert into [Customer data] values (3, 1234567812335678, 133451234)

insert into [Customer data] values (4, 1234567813345678, 123351234)

insert into [Customer data] values (5, 1234563812345678, 123431234)

Go

Tạo hai cột để lưu dữ liệu đã được mã hóa:

use [Customer DB]

Go

Alter table [Customer Data]

Add [Encrypted Credit Card Number] varbinary(MAX) Go

Alter table [Customer Data

Add [Encrypted Social Security Number] varbinary(MAX) Go

Cập nhật dữ liệu đã được mã hóa vào hai cột vừa tạo:

use [Customer DB] Go

Update [Customer Data]

set [Encrypted Credit Card Number] = EncryptByPassPhrase('Credit Card', convert(varchar(100),[Credit Card Number]))

Go

Update [Customer Data] set [Encrypted Social Security Number]

=

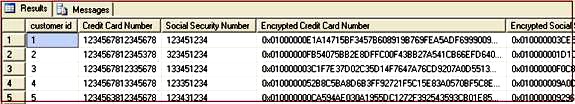
EncryptByPassPhrase('Social Security', convert(varchar(100),[Social Security Number])) Go

Truy vấn bảng bằng các lệnh sau:

Use [Customer DB] Go

Select \* from [customer data] Go

Kết quả:



Hình 10.5: Kết quả khi truy vấn 2 cột lƣu dữ liệu mã hóa

Xóa bỏ cột chứa dữ liệu chưa được mã hóa:

Use [Customer DB] Go

Alter table [Customer Data]

drop column [Credit Card Number] Go

Alter table [Customer Data]

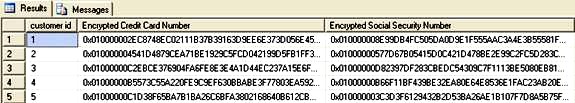
drop column [Social Security Number] Go

Truy vấn bảng theo các lệnh sau:

Use [Customer DB] Go

Select \* from [customer data] Go

Kết quả:



Hình 10.6: Kết quả sau khi xóa cột chƣa đƣợc mã hóa

Giải mã dữ liệu trên bảng thông qua hàm Decryptbypassphrase như sau:

Use [Customer DB] Go

Select [customer id],

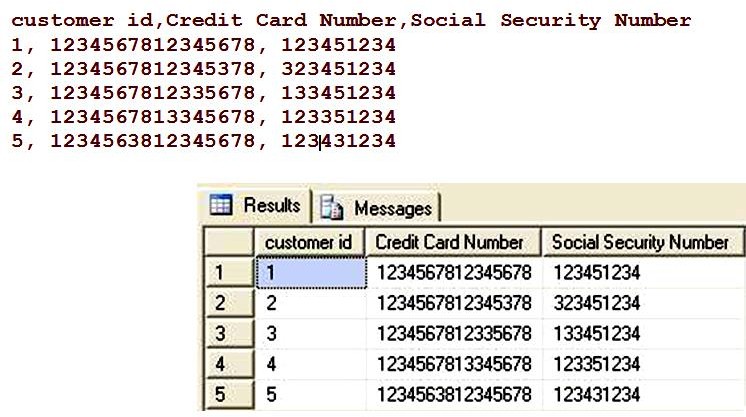
convert(bigint,convert(varchar(100), decryptbypassphrase('Credit Card',

[Encrypted Credit Card Number]) )) as [Credit Card Number], convert(bigint,convert(varchar(100), decryptbypassphrase('Social Security',

[Encrypted Social Security Number])))

as [Social Security Number] from [customer data] Go

Kết quả:



**Hình 10.7: Kết quả sau khi giải mã dữ liệu thông qua hàm Decryptbypassphrase**

### TÓM TẮT

*Mã hóa dữ liệu là công việc thực sự rất quan trọng. Bài này đã giới thiệu một trong 4 kỹ thuật mã hóa sẵn có trong SQL Server 2005 – kỹ thuật mã hóa bằng mật khẩu và phương pháp giải mã nó.*

*Học viên có thể tìm hiểu thêm về các kỹ thuật mã hóa khác.*

*Mặc dù mã hóa là một công cụ có giá trị để bảo đảm an ninh, nó không nên được xem xét cho tất cả các dữ liệu hoặc các kết nối. Khi ta quyết định có nên thực hiện mã hóa hay không, trước tiên hãy xem xét cách thức người dùng truy cập dữ liệu. Nếu người dùng truy cập dữ liệu qua một mạng công cộng, mã hóa dữ liệu có thể được yêu cầu nhằm tăng cường an ninh. Tuy nhiên, nếu tất cả các truy cập chỉ liên quan trong mạng nội bộ (đã đảm bảo an toàn), mã hóa có thể không được yêu cầu.*

### CÂU HỎI ÔN TẬP

**Câu 1:** Mã hóa là gì? Tại sao cần phải mã hóa thông tin?

**Câu 2:** Hãy nêu các kỹ thuật mã hóa đang được sử dụng hiện tại?

## BÀI 11: BÀI TẬP THỰC HÀNH

Bài tập thực hành bao gồm 5 bài cụ thể, mỗi bài chứa các nội dung gắn liền với bài học lý thuyết, có độ khó tăng dần. Học viên lần lượt làm tuần tự các bài tập trong các buổi thực hành tại phòng máy với sự hướng dẫn của giảng viên giảng dạy thực hành.

Cuối mỗi buổi thực hành, cần sao lưu dữ liệu (Detach/Backup) để về nhà làm tiếp hay có thể thực hành vào các buổi tiếp theo (Attach/Restore). Điểm quá trình (30%) của môn học này được tính trong quá trình thực hành (do giảng viên giảng dạy thực hành quy định), học viên cố gắng đi đầy đủ các buổi.

##### BÀI 1 - ĐỊA LÝ VIỆT NAM

Học viên ôn tập lại câu lệnh SQL, luyện tập các thao tác Detach, Attach, Backup, Restore database cho thuần thục nhằm hỗ trợ tốt cho các buổi thực hành tiếp theo.

**Yêu cầu:**

* + - Tạo DataBase DIALYVN. Chú ý: Kiểu thuộc tính và độ rộng của Khóa chính và Khóa ngoại phải giống nhau.
    - Nhập dữ liệu:
* Nhập bằng cách Copy và Paste
* Hãy tiến hành Detach và Attach DB.
* Hãy tiến hành Backup và Restore DB. Lưu ý: Xóa DB trước khi Restore.
  + - Dặn dò:

Sau mỗi buổi thực hành, SV có cơ chế sao lưu để phục hồi dữ liệu khi cần thiết (mang về nhà hay làm tiếp vào các buổi thực hành kế tiếp).

Hãy tạo CSDL Địa lý Việt Nam có cấu trúc như sau:

**TINH\_TP**(MA\_T\_TP, TEN\_T\_TP, DT, DS, MIEN)

Mô tả: Mỗi tỉnh-thành phố (TINH\_TP) có mã số tỉnh thành phố (MA\_T\_TP) duy nhất để phân biệt với các tỉnh-thành phố khác, có tên gọi (TEN\_T\_TP) cùng với diện tích (DT) và dân số (DS) tương ứng, và thuộc về một miền (MIEN) cụ thể: Bắc, Trung, Nam.

**BIENGIOI**(NUOC, MA\_T\_TP)

Mô tả: Biên giới (BIENGIOI) lưu trữ các nước (NUOC) lân cận giáp ranh biên giới với các tỉnh-thành phố của Việt Nam.

**LANGGIENG**(MA\_T\_TP, LG)

Mô tả: Lưu trữ các tỉnh-thành phố ở Việt Nam là láng giềng với nhau (LANGGIENG). Thuộc tính LG: chính là MA\_T\_TP.

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Mô tả** | **Kiểu** | **Kích thƣớc** |
| MA\_T\_TP | Mã số tỉnh, thành phố | varchar | 3 |
| TEN\_T\_TP | Tên tỉnh, Thành phố | nvarchar | 20 |
| DT | Diện tích (Km2) | float |  |
| DS | Dân số (Người) | bigint |  |
| MIEN | Miền (Bắc, Trung, Nam) | nvarchar | 10 |
| NUOC | Nước có biên giới | nvarchar | 15 |
| LG | Giống như MA\_T\_TP |  |  |

Nhập dữ liệu vào các bảng như bên dưới: TINH\_TP:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MA\_TTP** | **TEN\_TTP** | **DT** | **DS** | **MIEN** |
| AG | [An Giang](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/An_Giang/) | 3406 | 2142709 | Nam |
| BR | [Bà Rịa - Vũng Tàu](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Ba_Ria-Vung_Tau/) | 1982 | 996682 | Nam |
| BC | [Bắc Cạn](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Bac_Kan/) | 4857 | 293826 | Bắc |
| BG | [Bắc Giang](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Bac_Giang/) | 3823 | 1554131 | Bắc |
| BL | [Bạc Liêu](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Bac_Lieu/) | 2526 | 856518 | Nam |
| BN | [Bắc Ninh](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Bac_Ninh/) | 808 | 1024472 | Bắc |
| BT | [Bến Tre](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Ben_Tre/) | 2322 | 1255946 | Nam |
| BDI | [Bình Định](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Binh_Dinh/) | 6025 | 1486465 | Trung |
| BD | [Bình Dương](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Binh_Duong/) | 2696 | 1481550 | Nam |
| BP | [Bình Phước](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Binh_Phuoc/) | 6857 | 873598 | Nam |
| BTH | [Bình Thuận](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Binh_Thuan/) | 7828 | 1167023 | Trung |
| CM | [Cà Mau](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Ca_Mau/) | 5202 | 1206938 | Nam |
| CT | [Cần Thơ](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Can_Tho/) | 1390 | 1188435 | Nam |
| CB | [Cao Bằng](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Cao_Bang/) | 6691 | 507183 | Bắc |
| DN | [Đà Nẵng](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Thanh_pho_Da_Nang/) | 1256 | 887435 | Trung |
| DL | [Đắc Lắc](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Dak_Lak/) | 13085 | 1733624 | Trung |
| DNO | [Đắc Nông](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/ak_Nong/) | 6515 | 489392 | Trung |
| DB | [Điện Biên](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Dien_Bien/) | 9560 | 490306 | Bắc |
| DNA | [Đồng Nai](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Dong_Nai/) | 5895 | 2486154 | Nam |
| DT | [Đồng Tháp](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Dong_Thap/) | 3246 | 1666467 | Nam |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MA\_TTP** | **TEN\_TTP** | **DT** | **DS** | **MIEN** |
| GL | [Gia Lai](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Gia_Lai/) | 15495 | 1274412 | Trung |
| HG | [Hà Giang](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Ha_Giang/) | 7884 | 724537 | Bắc |
| HNA | [Hà Nam](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Tinh_Ha_Nam/) | 852 | 784045 | Bắc |
| HN | [Hà Nội](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Thanh_pho_Ha_Noi/) | 3334,47 | 6451909 | Bắc |
| HT | Hà Tây | 2193 | 2047000 | Bắc |
| HTI | Hà Tĩnh | 6055,70 | 1227554 | Bắc |
| HD | [Hải Dương](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Tinh_Hai_Duong/) | 1648 | 1705059 | Bắc |
| HP | [Hải Phòng](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Bai_bien_Do_Son/) | 1526 | 1837173 | Bắc |
| HAG | [Hậu Giang](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/20100327-Hau_Giang/) | 1608 | 757300 | Nam |
| HB | [Hòa Bình](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Hoa_Binh/) | 4663 | 785217 | Bắc |
| HY | [Hưng Yên](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Hung_Yen/) | 923 | 1127903 | Bắc |
| KH | [Khánh Hòa](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Khanh_Hoa/) | 5198 | 1157604 | Trung |
| KG | [Kiên Giang](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Kien_Giang/) | 6268 | 1688248 | Nam |
| KT | [Kon Tum](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Kon_Tum/) | 9615 | 430133 | Trung |
| LC | [Lai Châu](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Lai_Chau/) | 9059 | 370502 | Bắc |
| LD | [Lâm Đồng](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Lam_ong/) | 9765 | 1187574 | Trung |
| LS | [Lạng Sơn](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Lang_Son/) | 8305 | 732515 | Bắc |
| LCA | [Lào Cai](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Huyen_Sa_Pa/) | 6357 | 614595 | Bắc |
| LA | [Long An](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Long_An/) | 4491 | 1436066 | Nam |
| ND | [Nam Định](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Nam_Dinh/) | 1641 | 1828111 | Bắc |
| NA | Nghệ An | 16487 | 3113055 | Trung |
| NB | [Ninh Bình](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Ninh_Binh/) | 1384 | 898999 | Bắc |
| NT | [Ninh Thuận](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Ninh_Thuan/) | 3360 | 564993 | Trung |
| PT | [Phú Thọ](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Phu_Tho/) | 3520 | 1316389 | Bắc |
| PY | [Phú Yên](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Phu_Yen/) | 5045 | 862231 | Trung |
| QB | Quảng Bình | 8051,80 | 846924 | Trung |
| QN | [Quảng Nam](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Quang_Nam/) | 10407 | 1422319 | Trung |
| QNG | [Quảng Ngãi](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Quang_Ngai/) | 5138 | 1216773 | Trung |
| QNI | [Quảng Ninh](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Quang_Ninh/) | 5900 | 1144988 | Bắc |
| QT | Quảng Trị | 4745,70 | 597985 | Bắc |
| ST | [Sóc Trăng](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Soc_Trang/) | 3223 | 1292853 | Nam |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MA\_TTP** | **TEN\_TTP** | **DT** | **DS** | **MIEN** |
| SL | [Sơn La](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Son_La/) | 14055 | 1076055 | Bắc |
| TN | [Tây Ninh](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Tay_Ninh/) | 4030 | 1066513 | Nam |
| TB | [Thái Bình](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Thai_Binh/) | 1545 | 1781842 | Bắc |
| TNG | [Thái Nguyên](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Thai_Nguyen/) | 3543 | 1123116 | Bắc |
| TH | Thanh Hóa | 11116 | 3400595 | Bắc |
| TTH | Thừa Thiên Huế | 5053,99 | 1087579 | Trung |
| TG | [Tiền Giang](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Tien_Giang/) | 2367 | 1672271 | Nam |
| HCM | [TP Hồ Chí Minh](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Thanh_pho_Ho_Chi_Minh/) | 2095 | 7162864 | Nam |
| TV | [Trà Vinh](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Tra_Vinh/) | 2215 | 1003012 | Nam |
| TQ | [Tuyên Quang](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Tuyen_Quang/) | 5868 | 724821 | Bắc |
| VL | [Vĩnh Long](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Vinh_Long/) | 1475 | 1024707 | Nam |
| VP | [Vĩnh Phúc](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Vinh_Phuc/) | 1371 | 999786 | Bắc |
| YB | [Yên Bái](http://www.dulichvietnam.com.vn/Desktop.aspx/64-tinh/64-tinh/Yen_Bai/) | 6883 | 740387 | Bắc |

**LANGGIENG**:

|  |  |
| --- | --- |
| **MA\_T\_TP** | **LG** |
| AG | DT |
| AG | CT |
| AG | KG |
| BR | HCM |
| BR | BTH |
| BR | DNA |
| DNA | BTH |
| DNA | BR |
| DNA | HCM |
| BD | TN |
| BD | DNA |
| BD | BP |
| BD | HCM |
| BP | DL |
| BP | BD |
| BP | LD |
| BP | DNA |
| BL | KG |
| BL | CT |
| BL | CM |
| BL | ST |
| BT | TG |
| BT | VL |
| BT | TV |
| BT | HCM |

**LANGGIENG**: (tt)

**LANGGIENG**: (tt)

|  |  |
| --- | --- |
| **MA\_T\_TP** | **LG** |
| TG | LA |
| TG | DT |
| TG | BT |
| TG | VL |
| TN | BD |
| TN | BP |
| TN | LA |
| TN | HCM |
| BTH | DNA |

**BIENGIOI:**

|  |  |
| --- | --- |
| **MA\_T\_TP** | **LG** |
| CM | KG |
| CM | BL |
| CT | AG |
| CT | DT |
| CT | VL |
| CT | KG |
| CT | BL |
| CT | ST |
| DT | AG |
| DT | LA |
| DT | TG |
| HAG | CT |
| HAG | ST |
| HAG | VL |
| HAG | KG |
| HAG | BL |
| LA | TN |
| LA | TG |
| LA | DT |
| LA | HCM |
| HCM | BD |
| HCM | TN |
| HCM | DNA |
| HCM | BR |
| HCM | LA |
| HCM | TG |

|  |  |
| --- | --- |
| **NUOC** | **MA\_T\_TP** |
| CPC | AG |
| CPC | DL |
| CPC | DT |
| CPC | GL |
| CPC | KG |
| CPC | LA |
| CPC | BP |
| CPC | TN |
| LAO | GL |
| LAO | LC |
| LAO | NT |
| LAO | BT |
| TQ | CB |
| TQ | LC |

**Câu hỏi:**

Hãy thực hiện các câu truy vấn sau:

1. Xuất ra tên tỉnh, TP cùng với dân số của tỉnh,TP:
   1. Có diện tích >= 5000 Km2
   2. Có diện tích >= [input] (SV nhập một số bất kỳ từ bàn phím)
2. Xuất ra tên tỉnh, TP cùng với diện tích của tỉnh,TP:
   1. Thuộc miền Bắc
   2. Thuộc miền [input] (SV nhập một miền bất kỳ)
3. Xuất ra các Tên tỉnh, TP biên giới thuộc miền [input] (SV cho một miền bất kỳ)
4. Cho biết diện tích trung bình của các tỉnh, TP (Tổng DT/Tổng số tỉnh\_TP).
5. Cho biết dân số cùng với tên tỉnh của các tỉnh, TP có diện tích > 7000 Km2.
6. Cho biết dân số cùng với tên tỉnh của các tỉnh miền „Bắc‟.
7. Cho biết mã các nước biên giới của các tỉnh miền „Nam‟.
8. Cho biết diện tích trung bình của các tỉnh, TP. (Sử dụng hàm)
9. Cho biết mật độ dân số (DS/DT) cùng với tên tỉnh, TP của tất cả các tỉnh, TP.
10. Cho biết tên các tỉnh, TP láng giềng của tỉnh „Long An‟.
11. Cho biết số lượng các tỉnh, TP giáp với „CPC‟.
12. Cho biết tên những tỉnh, TP có diện tích lớn nhất.
13. Cho biết tỉnh, TP có mật độ DS đông nhất.
14. Cho biết tên những tỉnh, TP giáp với hai nước biên giới khác nhau.
15. Cho biết danh sách các miền cùng với các tỉnh, TP trong các miền đó.
16. Cho biết tên những tỉnh, TP có nhiều láng giềng nhất.
17. Cho biết những tỉnh, TP có diện tích nhỏ hơn diện tích trung bình của tất cả tỉnh, TP.
18. Cho biết tên những tỉnh, TP giáp với các tỉnh, TP ở miền „Nam‟ và không phải là miền „Nam‟.
19. Cho biết tên những tỉnh, TP có diện tích lớn hơn tất cả các tỉnh, TP láng giềng của nó.
20. Cho biết tên những tỉnh, TP mà ta có thể đến bằng cách đi từ „TP.HCM‟ xuyên qua ba tỉnh khác nhau và cũng khác với điểm xuất phát, nhưng láng giềng với nhau.

##### BÀI 2 - QUẢN LÝ BÁN HÀNG

**Yêu cầu:**

Tạo database QLBANHANG. Tạo các Table có cấu trúc như sau:

(Lưu ý: các field có gạch chân là khóa chính, gạch nét đứt là khóa ngoại)

Table: KHACHHANG

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Size** |
| MAKH | varchar | 5 |
| TENKH | nvarchar | 30 |
| DIACHI | nvarchar | 50 |
| DT | varchar | 11 |
| EMAIL | varchar | 30 |

**Ràng buộc:** TENKH not null, DT có thể có từ 8 chữ số đến 11 chữ số.

Table: VATTU

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Size** |
| MAVT | varchar | 5 |
| TENVT | Nvarchar | 30 |
| DVT | Nvarchar | 20 |
| GIAMUA | Money |  |
| SLTON | Int |  |

**Ràng buộc:** TENVT not null, GIAMUA > 0, SLTON >= 0.

Table: HOADON

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Size** |
| MAHD | varchar | 10 |
| NGAY | SmallDateTime |  |
| MAKH | varchar | 5 |
| TONGTG | Float |  |

**Ràng buộc:** Giá trị nhập vào cho field NGAY phải trước ngày hiện hành.

|  |  |  |
| --- | --- | --- |
| **Table: CTHD**  **Name** | **Type** | **Size** |
| MAHD | varchar | 10 |
| MAVT | varchar | 5 |
| SL | int |  |
| KHUYENMAI | Float |  |
| GIABAN | Float |  |

**Ràng buộc:** Giá trị nhập vào cho field SL phải lớn hơn 0.

**Câu hỏi:**

**Câu 1: Nhập dữ liệu vào các Table**: Table VATTU:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MAVT | TENVT | DVT | GIAMUA | SLTON |
| VT01 | Xi măng | Bao | 50000 | 5000 |
| VT02 | Cát | Khối | 45000 | 50000 |
| VT03 | Gạch | Viên | 120 | 800000 |
| VT04 | Gạch thẻ | Viên | 110 | 800000 |
| VT05 | Đá lớn | Khối | 25000 | 100000 |
| VT06 | Đá nhỏ | Khối | 33000 | 100000 |
| VT07 | Lam gió | Cái | 15000 | 50000 |

Table KHACHHANG:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MAKH | TENKH | DIACHI | DT | EMAIL |
| KH01 | Nguyễn Thị Bé | Tân Bình | 38457895 | [bnt@yahoo.com](mailto:bnt@yahoo.com) |
| KH02 | Lê Hoàng Nam | Bình Chánh | 39878987 | [namlehoang@gmail.com](mailto:namlehoang@gmail.com) |
| KH03 | Trần Thị Chiêu | Tân Bình | 38457895 | *NULL* |
| KH04 | Mai Thị Quế Anh | Bình Chánh | *NULL* | *NULL* |
| KH05 | Lê Văn Sáng | Quận 10 | *NULL* | [sanglv@hcm.vnn.vn](mailto:sanglv@hcm.vnn.vn) |
| KH06 | Trần Hoàng | Tân Bình | 38457897 | *NULL* |

Table HOADON:

|  |  |  |  |
| --- | --- | --- | --- |
| MAHD | NGAY | MAKH | TONGTG |
| HD001 | 12/05/2010 | KH01 | *NULL* |
| HD002 | 25/05/2010 | KH02 | *NULL* |
| HD003 | 25/05/2010 | KH01 | *NULL* |
| HD004 | 25/05/2010 | KH04 | *NULL* |
| HD005 | 26/05/2010 | KH04 | *NULL* |
| HD006 | 02/06/2010 | KH03 | *NULL* |
| HD007 | 22/06/2010 | KH04 | *NULL* |
| HD008 | 25/06/2010 | KH03 | *NULL* |
| HD009 | 15/08/2010 | KH04 | *NULL* |
| HD010 | 30/09/2010 | KH01 | *NULL* |

Table CTHD:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MAHD | MAVT | SL | KHUYENMAI | GIABAN |
| HD001 | VT01 | 5 |  | 52000 |
| HD001 | VT05 | 10 |  | 30000 |
| HD002 | VT03 | 10000 |  | 150 |
| HD003 | VT02 | 20 |  | 55000 |
| HD004 | VT03 | 50000 |  | 150 |
| HD004 | VT04 | 20000 |  | 120 |
| HD005 | VT05 | 10 |  | 30000 |
| HD005 | VT06 | 15 |  | 35000 |
| HD005 | VT07 | 20 |  | 17000 |
| HD006 | VT04 | 10000 |  | 120 |
| HD007 | VT04 | 20000 |  | 125 |
| HD008 | VT01 | 100 |  | 55000 |
| HD008 | VT02 | 20 |  | 47000 |
| HD009 | VT02 | 25 |  | 48000 |
| HD010 | VT01 | 25 |  | 57000 |

Câu 2: Tạo các view sau:

1. Hiển thị danh sách các khách hàng có địa chỉ là “Tân Bình” gồm mã khách hàng, tên khách hàng, địa chỉ, điện thoại, và địa chỉ E-mail.
2. Hiển thị danh sách các khách hàng gồm các thông tin mã khách hàng, tên khách hàng, địa chỉ và địa chỉ E-mail của những khách hàng chưa có số điện thoại.
3. Hiển thị danh sách các khách hàng chưa có số điện thoại và cũng chưa có địa chỉ Email gồm mã khách hàng, tên khách hàng, địa chỉ.
4. Hiển thị danh sách các khách hàng đã có số điện thoại và địa chỉ E-mail gồm mã khách hàng, tên khách hàng, địa chỉ, điện thoại, và địa chỉ E-mail.
5. Hiển thị danh sách các vật tư có đơn vị tính là “Cái” gồm mã vật tư, tên vật tư và giá mua.
6. Hiển thị danh sách các vật tư gồm mã vật tư, tên vật tư, đơn vị tính và giá mua mà có giá mua trên 25000.
7. Hiển thị danh sách các vật tư là “Gạch” (bao gồm các loại gạch) gồm mã vật tư, tên vật tư, đơn vị tính và giá mua.
8. Hiển thị danh sách các vật tư gồm mã vật tư, tên vật tư, đơn vị tính và giá mua mà có giá mua nằm trong khoảng từ 20000 đến 40000.
9. Lấy ra các thông tin gồm Mã hóa đơn, ngày lập hóa đơn, tên khách hàng, địa chỉ khách hàng và số điện thoại.
10. Lấy ra các thông tin gồm Mã hóa đơn, tên khách hàng, địa chỉ khách hàng và số điện thoại của ngày 25/5/2010.
11. Lấy ra các thông tin gồm Mã hóa đơn, ngày lập hóa đơn, tên khách hàng, địa chỉ khách hàng và số điện thoại của những hóa đơn trong tháng 6/2010.
12. Lấy ra danh sách những khách hàng (tên khách hàng, địa chỉ, số điện thoại) đã mua hàng trong tháng 6/2010.
13. Lấy ra danh sách những khách hàng không mua hàng trong tháng 6/2010 gồm các thông tin tên khách hàng, địa chỉ, số điện thoại.
14. Lấy ra các chi tiết hóa đơn gồm các thông tin mã hóa đơn, mã vật tư, tên vật tư, đơn vị tính, giá bán, giá mua, số lượng, trị giá mua (giá mua \* số lượng), trị giá bán (giá bán \* số lượng).
15. Lấy ra các chi tiết hóa đơn gồm các thông tin mã hóa đơn, mã vật tư, tên vật tư, đơn vị tính, giá bán, giá mua, số lượng, trị giá mua (giá mua \* số lượng), trị giá bán (giá bán \* số lượng) mà có giá bán lớn hơn hoặc bằng giá mua.
16. Lấy ra các thông tin gồm mã hóa đơn, mã vật tư, tên vật tư, đơn vị tính, giá bán, giá mua, số lượng, trị giá mua (giá mua \* số lượng), trị giá bán (giá bán \* số lượng) và cột khuyến mãi với khuyến mãi 10% cho những mặt hàng bán trong một hóa đơn lớn hơn 100.
17. Tìm ra những mặt hàng chưa bán được.
18. Tạo bảng tổng hợp gồm các thông tin: mã hóa đơn, ngày hóa đơn, tên khách hàng, địa chỉ, số điện thoại, tên vật tư, đơn vị tính, giá mua, giá bán, số lượng, trị giá mua, trị giá bán.
19. Tạo bảng tổng hợp tháng 5/2010 gồm các thông tin: mã hóa đơn, ngày hóa đơn, tên khách hàng, địa chỉ, số điện thoại, tên vật tư, đơn vị tính, giá mua, giá bán, số lượng, trị giá mua, trị giá bán.
20. Tạo bảng tổng hợp quý 1 – 2010 gồm các thông tin: mã hóa đơn, ngày hóa đơn, tên khách hàng, địa chỉ, số điện thoại, tên vật tư, đơn vị tính, giá mua, giá bán, số lượng, trị giá mua, trị giá bán.
21. Lấy ra danh sách các hóa đơn gồm các thông tin: Số hóa đơn, ngày, tên khách hàng, địa chỉ khách hàng, tổng trị giá của hóa đơn.
22. Lấy ra hóa đơn có tổng trị giá lớn nhất gồm các thông tin: Số hóa đơn, ngày, tên khách hàng, địa chỉ khách hàng, tổng trị giá của hóa đơn.
23. Lấy ra hóa đơn có tổng trị giá lớn nhất trong tháng 5/2010 gồm các thông tin: Số hóa đơn, ngày, tên khách hàng, địa chỉ khách hàng, tổng trị giá của hóa đơn.
24. Đếm xem mỗi khách hàng có bao nhiêu hóa đơn.
25. Đếm xem mỗi khách hàng, mỗi tháng có bao nhiêu hóa đơn.
26. Lấy ra các thông tin của khách hàng có số lượng hóa đơn mua hàng nhiều nhất.
27. Lấy ra các thông tin của khách hàng có số lượng hàng mua nhiều nhất.
28. Lấy ra các thông tin về các mặt hàng mà được bán trong nhiều hóa đơn nhất.
29. Lấy ra các thông tin về các mặt hàng mà được bán nhiều nhất.
30. Lấy ra danh sách tất cả các khách hàng gồm Mã khách hàng, tên khách hàng, địa chỉ, số lượng hóa đơn đã mua (nếu khách hàng đó chưa mua hàng thì cột số lượng hóa đơn để trống)

Câu 3: Tạo các procedure sau:

1. Lấy ra danh các khách hàng đã mua hàng trong ngày X, với X là tham số truyền vào.
2. Lấy ra danh sách khách hàng có tổng trị giá các đơn hàng lớn hơn X (X là tham số).
3. Lấy ra danh sách X khách hàng có tổng trị giá các đơn hàng lớn nhất (X là tham số).
4. Lấy ra danh sách X mặt hàng có số lượng bán lớn nhất (X là tham số).
5. Lấy ra danh sách X mặt hàng bán ra có lãi ít nhất (X là tham số).
6. Lấy ra danh sách X đơn hàng có tổng trị giá lớn nhất (X là tham số).
7. Tính giá trị cho cột khuyến mãi như sau: Khuyến mãi 5% nếu SL > 100, 10% nếu SL > 500.
8. Tính lại số lượng tồn cho tất cả các mặt hàng (SLTON = SLTON – tổng SL bán được).
9. Tính trị giá cho mỗi hóa đơn.
10. Tạo ra table KH\_VIP có cấu trúc giống với cấu trúc table KHACHHANG. Lưu các khách hàng có tổng trị giá của tất cả các đơn hàng >=10.000.000 vào table KH\_VIP.

Câu 4: Tạo các function sau:

1. Viết hàm tính doanh thu của năm, với năm là tham số truyền vào.
2. Viết hàm tính doanh thu của tháng, năm, với tháng và năm là 2 tham số truyền vào.
3. Viết hàm tính doanh thu của khách hàng với mã khách hàng là tham số truyền vào.
4. Viết hàm tính tổng số lượng bán được cho từng mặt hàng theo tháng, năm nào đó. Với mã hàng, tháng và năm là các tham số truyền vào, nếu tháng không nhập vào tức là tính tất cả các tháng.
5. Viết hàm tính lãi ((giá bán – giá mua) \* số lượng bán được) cho từng mặt hàng, với mã mặt hàng là tham số truyền vào. Nếu mã mặt hàng không truyền vào thì tính cho tất cả các mặt hàng.

Câu 5: Tạo các trigger để thực hiện các ràng buộc sau:

1. Thực hiện việc kiểm tra các ràng buộc khóa ngoại.
2. Không cho phép CASCADE DELETE trong các ràng buộc khóa ngoại. Ví dụ không cho phép xóa các HOADON nào có SOHD còn trong table CTHD.
3. Không cho phép user nhập vào hai vật tư có cùng tên.
4. Khi user đặt hàng thì KHUYENMAI là 5% nếu SL >100, 10% nếu SL > 500.
5. Chỉ cho phép mua các mặt hàng có số lượng tồn lớn hơn hoặc bằng số lượng cần mua và tính lại số lượng tồn mỗi khi có đơn hàng.
6. Không cho phép user xóa một lúc nhiều hơn một vật tư.
7. Mỗi hóa đơn cho phép bán tối đa 5 mặt hàng.
8. Mỗi hóa đơn có tổng trị giá tối đa 50000000.
9. Không được phép bán hàng lỗ quá 50%.
10. Chỉ bán mặt hàng Gạch (các loại gạch) với số lượng là bội số của 100.

##### BÀI 3 - QUẢN LÝ SẢN XUẤT

**Yêu cầu:**

Hãy tạo lược đồ CSDL như mô tả sau, học viên tự chọn kiểu dữ liệu và thuộc tính sao cho phù hợp.

**Loai**(MaLoai, TenLoai)

**SanPham**(MaSP, TenSP, MaLoai) **NhanVien**(MaNV, HoTen, NgaySinh, Phai) **PhieuXuat**(MaPX, NgayLap, MaNV) **CTPX**(MaPX, MaSP, SoLuong)

Các ràng buộc:

* Tên sản phẩm là duy nhất nhưng không là khóa chính.
* Mã phiếu xuất là số nguyên tự động tăng.
* Các mã số của các bảng còn lại là phải nhập vào và có kiểu là char(5).
* Số lượng xuất kho là số nguyên.
* Phái của nhân viên có giá trị là 1 (nếu là nam) hoặc 0 (nếu là nữ), mặc định là 0.
* Giá trị cho ngày sinh của nhân viên là >=18 tuổi và <=55 tuổi.

**Câu hỏi:**

**Câu 1**: Viết SQL Script tạo các bảng trên theo đúng các quy tắc ràng buộc trên.

**Câu 2**: Viết Script để nhập dữ liệu cho các bảng:

Bảng LOAI:

|  |  |
| --- | --- |
| **Mã loại** | **Tên loại** |
| 1 | Vật liệu xây dựng |
| 2 | Hàng tiêu dùng |
| 3 | Ngũ cốc |

Bảng SANPHAM:

|  |  |  |
| --- | --- | --- |
| **Mã SP** | **Tên SP** | **Mã loại** |
| 1 | Xi măng | 1 |
| 2 | Gạch | 1 |
| 3 | Gạo nàng hương | 3 |
| 4 | Bột mì | 3 |
| 5 | Kệ chén | 2 |
| 6 | Đậu xanh | 3 |

Bảng NHANVIEN:

|  |  |  |  |
| --- | --- | --- | --- |
| **Mã NV** | **Họ tên** | **Ngày sinh** | **Phái** |
| NV01 | Nguyễn Mai Thi | 15/5/1982 | 0 |
| NV02 | Trần Đình Chiến | 2/12/1980 | 1 |
| NV03 | Lê Thị Chi | 23/1/1979 | 0 |

Bảng PHIEUXUAT:

|  |  |  |
| --- | --- | --- |
| **Mã phiếu xuất** | **Ngày lập** | **Mã NV** |
| 1 | 12/3/2010 | NV01 |
| 2 | 3/2/2010 | NV02 |
| 3 | 1/6/2010 | NV03 |
| 4 | 16/6/2010 | NV01 |

Bảng CTPX:

|  |  |  |
| --- | --- | --- |
| **MAPX** | **MASP** | **Số lƣợng** |
| 1 | 1 | 10 |
| 1 | 2 | 15 |
| 1 | 3 | 5 |
| 2 | 2 | 20 |
| 3 | 1 | 20 |
| 3 | 3 | 25 |
| 4 | 5 | 12 |

**Câu 3:** Tạo các view theo yêu cầu sau:

* 1. Cho biết mã sản phẩm, tên sản phẩm, tổng số lượng xuất của từng sản phẩm trong năm 2010. Lấy dữ liệu từ View này sắp xếp tăng dần theo tên sản phẩm.
  2. Cho biết mã sản phẩm, tên sản phẩm, tên loại sản phẩm mà đã được bán từ ngày 1/1/2010 đến 30/6/2010.
  3. Cho biết số lượng sản phẩm trong từng loại sản phẩm gồm các thông tin: mã loại sản phẩm, tên loại sản phẩm, số lượng các sản phẩm.
  4. Cho biết tổng số lượng phiếu xuất trong tháng 6 năm 2010.
  5. Cho biết thông tin về các phiếu xuất mà nhân viên có mã NV01 đã xuất.
  6. Cho biết danh sách nhân viên nam có tuổi trên 25 nhưng dưới 30.
  7. Thống kê số lượng phiếu xuất theo từng nhân viên.
  8. Thống kê số lượng sản phẩm đã xuất theo từng sản phẩm.
  9. Lấy ra tên của nhân viên có số lượng phiếu xuất lớn nhất.
  10. Lấy ra tên sản phẩm được xuất nhiều nhất trong năm 2010.

**Câu 4**: Tạo các Function sau:

1. Function F1 có 2 tham số vào là: tên sản phẩm, năm. Function cho biết: số lượng xuất kho của tên sản phẩm này trong năm này. (Chú ý: Nếu tên sản phẩm này không tồn tại thì phải trả về 0)
2. Function F2 có 1 tham số nhận vào là mã nhân viên. Function trả về số lượng phiếu xuất của nhân viên truyền vào. Nếu nhân viên này không tồn tại thì trả về 0.
3. Function F3 có 1 tham số vào là năm, trả về danh sách các sản phẩm được xuất trong năm truyền vào.
4. Function F4 có một tham số vào là mã nhân viên để trả về danh sách các phiếu xuất của nhân viên đó. Nếu mã nhân viên không truyền vào thì trả về tất cả các phiếu xuất.
5. Function F5 để cho biết tên nhân viên của một phiếu xuất có mã phiếu xuất là tham số truyền vào.
6. Function F6 để cho biết danh sách các phiếu xuất từ ngày T1 đến ngày T2. (T1, T2 là tham số truyền vào). Chú ý: T1 <= T2.
7. Function F7 để cho biết ngày xuất của một phiếu xuất với mã phiếu xuất là tham số truyền vào.

**Câu 5:** Tạo các Procedure sau:

1. Procedure tên là P1 cho có 2 tham số sau:

* 1 tham số nhận vào là: tên sản phẩm.
* 1 tham số trả về cho biết: tổng số lượng xuất kho của tên sản phẩm này trong năm 2010 (Không viết lại truy vấn, hãy sử dụng Function F1 ở câu 4 để thực hiện)

1. Procedure tên là P2 có 2 tham số sau:

* 1 tham số nhận vào là: tên sản phẩm.
* 1 tham số trả về cho biết: tổng số lượng xuất kho của tên sản phẩm này trong khoảng thời gian từ đầu tháng 4/2010 đến hết tháng 6/2010 (Chú ý: Nếu tên sản phẩm này không tồn tại thì trả về 0)

1. Procedure tên là P3 chỉ có duy nhất 1 tham số nhận vào là tên sản phẩm. Trong Procedure này có khai báo 1 biến cục bộ được gán giá trị là: số lượng xuất kho của tên sản phẩm này trong khoảng thời gian từ đầu tháng 4/2010 đến hết tháng 6/2010. Việc gán trị này chỉ được thực hiện bằng cách gọi Procedure P2.
2. Procedure P4 để INSERT một record vào trong table LOAI. Giá trị các field là tham số truyền vào.
3. Procedure P5 để DELETE một record trong Table NhânViên theo mã nhân viên. Mã NV là tham số truyền vào.

**Câu 6:** Viết các trigger để thực hiện các ràng buộc sau:

1. Chỉ cho phép một phiếu xuất có tối đa 5 chi tiết phiếu xuất.
2. Chỉ cho phép một nhân viên lập tối đa 10 phiếu xuất trong một ngày.
3. Khi người dùng viết 1 câu truy vấn nhập 1 dòng cho bảng chi tiết phiếu xuất thì CSDL kiểm tra nếu mã phiếu xuất mới đó chưa tồn tại trong bảng phiếu xuất thì

CSDL sẽ không cho phép nhập và thông báo lỗi “Phiếu xuất này không tồn tại”. Hãy viết 1 trigger đảm bảo điều này.

##### BÀI 4 - QUẢN LÝ SẢN PHẨM

Yêu cầu:

Hãy tạo lược đồ CSDL sau, học viên tự chọn kiểu dữ liệu và thuộc tính sao cho phù hợp.

**LOAISP**(MaLoai, TenLoai) **SANPHAM**(MASP, TenSP, Mota, Gia, Maloai) **KHACHHANG**(MAKH, TenKH, DC, DT) **DONDH**(SoDDH, NgayDat, MAKH) **CTDDH**(SoDDH, MASP, SoLuong) **NGUYENLIEU**(MaNL, TenNL, DVT, Gia) **LAM**(MaNL, MASP, SoLuong)

Giải thích:

* LOAISP: Mỗi loại sản phẩm có mã loại và tên loại.
* SANPHAM: Mỗi sản phẩm có mã số duy nhất, có tên sản phẩm, mô tả sản phẩm và giá của sản phẩm. Mỗi sản phẩm chỉ thuộc 1 loại.
* KHACHHANG: Mỗi khách hàng có mã số, tên, địa chỉ và điện thoại.
* DONDH: Mỗi đơn đặt hàng có số đơn duy nhất, có ngày đặt và do 1 khách hàng nào đó đặt.
* CTDDH: Mỗi đơn đặt hàng có thể đặt nhiều sản phẩm với số lượng khác nhau cho từng sản phẩm.
* NGUYENLIEU: Mỗi nguyên liệu có mã số, tên nguyên liệu, đơn vị tính và giá.
* LAM: Mỗi sản phẩm được làm từ nhiều nguyên liệu với số lượng tương ứng.

**Câu hỏi:**

1. Tạo các ràng buộc khóa chính, khóa ngoại.
2. Xác định và tạo các ràng buộc dữ liệu khác: UNIQUE, CHECK, DEFAULT, RULE.
3. Xác định và tạo các chỉ mục cần thiết cho CSDL.
4. Nhập dữ liệu thử cho CSDL như minh họa bên dưới.
5. **Thực hiện các view sau:**
   1. Danh sách các loại sản phẩm có nhiều sản phẩm nhất (Tên loại SP, số sản phẩm)
   2. Danh sách khách hàng không đặt hàng trong tháng 3/2010 (Tên KH, địa chỉ).
   3. DS khách hàng đặt nhiều đơn đặt hàng nhất trong tháng 3/2010 (Tên KH, địa chỉ).
   4. Danh sách các sản phẩm không được đặt trong tháng 3/2010 (Tên SP, mô tả).
   5. Danh sách khách hàng có đặt trên 10 cái tủ DDA (Tên KH, địa chỉ, tổng số lượng).
   6. DS các sản phẩm được làm từ nhiều loại nguyên liệu nhất (Tên SP, Giá, Số loại).
   7. Danh sách các sản phẩm có giá thành SX hơn 1 triệu (Tên SP, Giá thành SX).
   8. DS các sản phẩm có lãi trên 20% (Tên SP, Giá thành SX, Giá bán, phần trăm lãi).
   9. Danh sách đơn đặt hàng có tổng số tiền lớn hơn 100 triệu (Số DDH, Ngày đặt, Tổng tiền).
   10. Danh sách các loại nguyên liệu dùng để làm tất cả các sản phẩm (TênNL, Giá).
   11. Danh sách khách hàng có đặt tất cả các sản phẩm (Tên KH, DC).
   12. Danh sách các sản phẩm tất cả các khách hàng đều đặt (Tên SP, Mô tả).
   13. Danh sách khách hàng lâu nhất chưa đặt hàng (Tên KH, địa chỉ).
6. **Viết các Stored Procedure sau:**
   1. Liệt kê DS khách hàng (TênKH, DC) có đặt hàng vào Ngày tháng năm X.
   2. Liệt kê DS khách hàng (TênKH, DC) có đặt hàng sản phẩm có mã số X.
   3. Liệt kê DS khách hàng (TênKH, DC) có đặt hàng với tổng số tiền trên X (1 đơn).
   4. Liệt kê DS khách hàng (TênKH, DC) có đặt hàng với tổng số tiền trên X (tất cả).
   5. Liệt kê DS sản phẩm (TênSP, Giá thành SX, Giá) bán lãi trên X.
   6. Liệt kê DS khách hàng (TênKH, DC) đã trên X ngày rồi chưa đặt hàng.
   7. Liệt kê DS sản phẩm (TênSP, Số đơn) có tổng số đơn đặt hàng trên X.
   8. Liệt kê DS sản phẩm (TênSP, Tổng SL) có tổng số lượng đặt hàng trên X.
   9. Liệt kê DS sản phẩm (TênSP, Tổng số tiền) có tổng số tiền đặt hàng trên X.
7. **Trigger:**
   1. Mỗi ngày mỗi khách hàng chỉ đặt tối đa 2 đơn hàng.
   2. Mỗi đơn đặt hàng có tổng số lượng sản phẩm không quá 100.
   3. Đảm bảo rằng mỗi sản phẩm không bị lỗ hơn 50%.
8. **Cursor**
   1. Viết thủ tục liệt kê (theo thứ tự giảm dần của giá) X sản phẩm đầu tiên của sản phẩm có mã loại là Y (X, Y nhập khi gọi thủ tục) theo dạng sau:

Tên loại: Tủ

Sản phẩm 1: Tủ Trang điểm – Mô tả: Cao 1.4m, rộng 2.2m – Giá: 1000000 Sản phẩm 2: Tủ DDA – Mô tả: Cao 1.6m, rộng 2.0m, cửa kiếng – Giá: 800000…

* 1. Viết thủ tục cập nhật giá cho bảng sản phẩm như sau:
* Những sản phẩm có lãi trên 30% thì giảm giá 10%
* Những sản phẩm bị lỗ thì cập nhật giá bằng giá thành SX
* Những sản phẩm khác thì tăng 5%
  1. Viết thủ tục in ra thống kê đặt hàng trong tháng X (nhập khi gọi thủ tục) có dạng như sau:

Ngày 15/3/2010:

1-DH001: KH001, Tổng tiền: 15000000

2-DH002: KH016, Tổng tiền: 27500000

Ngày 16/3/2010:

1-DH003: KH003, Tổng tiền: 8000000

2-DH004: KH012, Tổng tiền: 1100000

…

1. **Bảo mật phân quyền**

Tạo 3 user: Giám đốc, Kế toán và Nhân viên. Thực hiện phân quyền như sau:

* Giám đốc: Toàn quyền trên toàn bộ CSDL
* Kế toán: Chỉ được xem trên bảng SANPHAM, LOAISP. Các bảng khác toàn quyền.
* Nhân viên: Không được truy cập bảng NGUYENLIEU, LAM. Các bảng khác được quyền xem, không được cập nhật.

CÁC BẢNG DỮ LIỆU MẪU:

**LOAISP**(MaLoai, TenLoai)

|  |  |
| --- | --- |
| **MALOAI** | **TENLOAI** |
| L01 | Tủ |
| L02 | Bàn |
| L03 | Giường |

**SANPHAM**(MASP, TenSP, Mota, Gia, Maloai)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MASP** | **TENSP** | **MOTA** | **GIA** | **MALOAI** |
| SP01 | Tủ trang điểm | Cao 1.4m, rộng 2.2m | 1000000 | L01 |
| SP02 | Giường đơn Cali | Rộng 1.4m | 1500000 | L03 |
| SP03 | Tủ DDA | Cao 1.6m, rộng 2.0m, cửa kiếng | 800000 | L01 |
| SP04 | Bàn ăn | 1m x 1.5m | 650000 | L02 |
| SP05 | Bàn uống trà | Tròn, 1.8m | 1100000 | L02 |

**KHACHHANG**(MAKH, TenKH, DC, DT)

|  |  |  |  |
| --- | --- | --- | --- |
| **MAKH** | **TENKH** | **DC** | **DT** |
| KH001 | Trần Hải Cường | 731 Trần Hưng Đạo, Q.1, TP.HCM | 08-9776655 |
| KH002 | Nguyễn Thị Bé | 638 Nguyễn Văn Cừ, Q.5, TP.HCM | 0913-666123 |
| KH003 | Trần Thị Minh Hòa | 543 Mai Thị Lựu, Ba Đình, Hà Nội | 04-9238777 |
| KH004 | Phạm Đình Tuân | 975 Lê Lai, P.3, TP.Vũng Tàu | 064-543678 |
| KH005 | Lê Xuân Nguyện | 450 Trưng Vương, Mỹ Tho, Tiền Giang | 073-987123 |
| KH006 | Văn Hùng Dũng | 291 Hồ Văn Huê, Q.PN, TP.HCM | 08-8222111 |
| KH012 | Lê Thị Hương Hoa | 980 Lê Hồng Phong, TP.Vũng Tàu | 064-452100 |
| KH016 | Hà Minh Trí | 332 Nguyễn Thái Học, TP.Quy Nhơn | 056-565656 |

**DONDH**(SoDDH, NgayDat, MAKH)

|  |  |  |
| --- | --- | --- |
| **SODDH** | **NGAYDAT** | **MAKH** |
| DH001 | 15/03/2010 | KH001 |
| DH002 | 15/03/2010 | KH016 |
| DH003 | 16/03/2010 | KH003 |
| DH004 | 16/03/2010 | KH012 |
| DH005 | 17/03/2010 | KH001 |
| DH006 | 01/04/2010 | KH002 |

**CTDDH**(SoDDH, MASP, SoLuong)

|  |  |  |
| --- | --- | --- |
| **SODDH** | **MASP** | **SOLUONG** |
| DH001 | SP01 | 5 |
| DH001 | SP03 | 1 |
| DH002 | SP02 | 2 |
| DH003 | SP01 | 2 |
| DH003 | SP04 | 10 |
| DH003 | SP05 | 5 |
| DH004 | SP02 | 2 |
| DH004 | SP05 | 2 |
| DH005 | SP03 | 3 |
| DH006 | SP02 | 4 |
| DH006 | SP04 | 3 |
| DH006 | SP05 | 6 |

**NGUYENLIEU**(MaNL, TenNL, DVT, Gia)

|  |  |  |  |
| --- | --- | --- | --- |
| **MANL** | **TENNL** | **DVT** | **GIA** |
| NL01 | Gỗ Lim XP | m3 | 1200000 |
| NL02 | Gỗ Sao NT | m3 | 1000000 |
| NL03 | Gỗ tạp nham | m3 | 500000 |
| NL04 | Đinh lớn | Kg | 40000 |
| NL05 | Đinh nhỏ | Kg | 30000 |
| NL06 | Kiếng | m2 | 350000 |

**LAM**(MaNL, MASP, SoLuong)

|  |  |  |
| --- | --- | --- |
| **MANL** | **MASP** | **SOLUONG** |
| NL01 | SP01 | 1.2 |
| NL03 | SP01 | 0.3 |
| NL06 | SP01 | 2.5 |
| NL02 | SP02 | 1.1 |
| NL04 | SP02 | 2.2 |
| NL02 | SP03 | 0.9 |
| NL05 | SP03 | 2.1 |
| NL02 | SP04 | 1.3 |
| NL04 | SP04 | 1.7 |
| NL03 | SP05 | 0.8 |
| NL05 | SP05 | 0.5 |
| NL06 | SP05 | 2.4 |

-

### TÀI LIỆU THAM KHẢO

1. Võ Hoàng Khang, *Bài giảng Hệ QTCSDL (Lưu hành nội bộ).* ĐH Kỹ Thuật Công Nghệ TP.HCM, 2010.
2. Bộ môn HTTT, Khoa CNTT, *Hệ QTCSDL*, ĐH Khoa học Tự nhiên, ĐH Quốc Gia TP.HCM, 2010.
3. Tạ Thị Thu Phượng. *Hệ quản trị CSDL (Bài giảng tóm tắt)*. ĐH Đà Lạt, 2007.
4. [www.msdn.com](http://www.msdn.com/)