# MỤC LỤC

[MỤC LỤC 1](#_Toc463594022)

[DANH MỤC HÌNH ẢNH 8](#_Toc463594023)

[CHƯƠNG 1 TỔNG QUAN VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU 10](#_Toc463594024)

[MỤC ĐÍCH VÀ YÊU CẦU 10](#_Toc463594025)

[NỘI DUNG 11](#_Toc463594026)

[1.1. Giới thiệu hệ quản trị cơ sở dữ liệu 11](#_Toc463594027)

[1.1.1. Giới thiệu chung 11](#_Toc463594028)

[1.1.2. Một số khái niệm cơ bản 11](#_Toc463594029)

[1.1.3. Các tính năng và đặc điểm chính của HQT CSDL 13](#_Toc463594030)

[1.1.4. Hệ quản trị cơ sở dữ liệu Microsoft SQL Server 2008 16](#_Toc463594031)

[1.1.5. Các thành phần cơ bản của hệ quản trị cơ sở dữ liệu (HQT CSDL) 26](#_Toc463594032)

[1.1.6. Kiến trúc CSDL quan hệ (relational Database Architecture) 28](#_Toc463594033)

[1.1.7. Các công cụ và trình tiện ích. 33](#_Toc463594034)

[1.2. Tạo và quản trị cơ sở dữ liệu. 35](#_Toc463594035)

[1.2.1. Cơ sở dữ liệu (CSDL) và các tập tin lưu trữ. 35](#_Toc463594036)

[1.2.2. Quản trị database 38](#_Toc463594037)

[1.3. Tạo và quản trị bảng. 39](#_Toc463594038)

[1.3.1. Kiểu dữ liệu. 40](#_Toc463594039)

[1.3.2. Tạo bảng. 41](#_Toc463594040)

[1.3.3. Quản trị bảng 42](#_Toc463594041)

[BÀI TẬP CHƯƠNG 1 44](#_Toc463594042)

[A. CÂU HỎI 44](#_Toc463594043)

[B. BÀI TẬP 44](#_Toc463594044)

[CHƯƠNG 2 SAO LƯU (BACKUP) - PHỤC HỒI (RESTORE) VÀ CHUYỂN ĐỔI DỮ LIỆU (IMPORT/EXPORT) 48](#_Toc463594045)

[MỤC ĐÍCH VÀ YÊU CẦU 48](#_Toc463594046)

[NỘI DUNG 49](#_Toc463594047)

[2.1. Tổng quan về sao lưu và phục hồi dữ liệu 49](#_Toc463594048)

[2.1.1. Mục đích của việc sao lưu và phục hồi dữ liệu 49](#_Toc463594049)

[2.1.2. Các cơ chế sao lưu dữ liệu 49](#_Toc463594050)

[2.1.3. Kịch bản sao lưu dữ liệu 49](#_Toc463594051)

[2.2. Sao lưu dữ liệu (Backup) 50](#_Toc463594052)

[2.3. Phục hồi 51](#_Toc463594053)

[2.4. Attach và detach 53](#_Toc463594054)

[2.5. Import/Export data – Nhập hoặc trích xuất dữ liệu 53](#_Toc463594055)

[Import hoặc Export Wizard 54](#_Toc463594056)

[BÀI TẬP CHƯƠNG 2 58](#_Toc463594057)

[A. CÂU HỎI: 58](#_Toc463594058)

[B. BÀI TẬP: 58](#_Toc463594059)

[CHƯƠNG 3 THIẾT LẬP RÀNG BUỘC TOÀN VẸN (CONSTRAINTS) 62](#_Toc463594060)

[MỤC ĐÍCH VÀ YÊU CẦU 62](#_Toc463594061)

[NỘI DUNG 62](#_Toc463594062)

[3.1. Giới thiệu 62](#_Toc463594063)

[3.1.1. Định nghĩa ràng buộc toàn vẹn (RBTV) 62](#_Toc463594064)

[3.1.2. Các yếu tố của một RBTV 63](#_Toc463594065)

[3.2. Phân loại các RBTV 64](#_Toc463594066)

[3.3. Cài đặt các ràng buộc toàn vẹn. 64](#_Toc463594067)

[3.3.1. Cài đặt RBTV miền giá trị 65](#_Toc463594068)

[3.3.2. Cài đặt RBTV liên bộ 68](#_Toc463594069)

[3.3.3. Ràng buộc tham chiếu – ràng buộc khóa ngoại (foreign key) 71](#_Toc463594070)

[3.4. Xóa bỏ RBTV 73](#_Toc463594071)

[3.4.1. Hủy bỏ các ràng buộc toàn vẹn CONSTRAINT 73](#_Toc463594072)

[3.4.2. Hủy bỏ các ràng buộc toàn vẹn RULE 73](#_Toc463594073)

[BÀI TẬP CHƯƠNG 3 74](#_Toc463594074)

[A. CÂU HỎI 74](#_Toc463594075)

[B. BÀI TẬP 74](#_Toc463594076)

[CHƯƠNG 4 QUẢN LÝ VÀ THAO TÁC DỮ LIỆU 78](#_Toc463594077)

[MỤC ĐÍCH VÀ YÊU CẦU 78](#_Toc463594078)

[NỘI DUNG 79](#_Toc463594079)

[4.1. Truy vấn dữ liệu 79](#_Toc463594080)

[4.1.1. Giới thiệu chung 79](#_Toc463594081)

[4.1.2. Tìm thông tin từ các cột của bảng - mệnh đề SELECT 81](#_Toc463594082)

[4.1.3. Chọn các dòng của bảng – mệnh đề WHERE 84](#_Toc463594083)

[4.1.4. Thứ tự thể hiện các bản ghi - mệnh đề ORDER BY 86](#_Toc463594084)

[4.1.5. Chọn các nhóm thỏa điều kiện – mệnh đề HAVING 89](#_Toc463594085)

[4.2. Cập nhật dữ liệu 91](#_Toc463594086)

[4.2.1. Lệnh INSERT 91](#_Toc463594087)

[4.2.2. Lệnh DELETE 93](#_Toc463594088)

[4.2.3. Lệnh UPDATE 93](#_Toc463594089)

[4.3. Truy vấn lồng 94](#_Toc463594090)

[4.3.1. Phân loại: 94](#_Toc463594091)

[4.3.2. Cú pháp 95](#_Toc463594092)

[4.3.2. Ví dụ minh họa 95](#_Toc463594093)

[4.5. Một số vấn đề cần lưu ý khi thực hiện câu truy vấn 97](#_Toc463594094)

[BÀI TẬP CHƯƠNG 4 99](#_Toc463594095)

[A. CÂU HỎI 99](#_Toc463594096)

[B. BÀI TẬP 99](#_Toc463594097)

[CHƯƠNG 5 LẬP TRÌNH T – SQL 101](#_Toc463594098)

[MỤC ĐÍCH VÀ YÊU CẦU 101](#_Toc463594099)

[NỘI DUNG 102](#_Toc463594100)

[5.1. Tổng quan về ngôn ngữ T – SQL 102](#_Toc463594101)

[5.1.1. Giới thiệu 102](#_Toc463594102)

[5.1.2. Các thành phần cơ bản trong cú pháp của ngôn ngữ T - SQL 103](#_Toc463594103)

[5.2. Biến và cách sử dụng biến trong T-SQL 105](#_Toc463594104)

[5.2.1. Khái niệm biến 105](#_Toc463594105)

[5.2.2. Phân loại biến theo phạm vi hoạt động 105](#_Toc463594106)

[5.2.3. Khai báo và gán giá trị cho biến 107](#_Toc463594107)

[5.3. Các cấu trúc điều khiển 111](#_Toc463594108)

[5.3.1. Cấu trúc IF … ELSE … 111](#_Toc463594109)

[5.3.2. Cấu trúc WHILE 114](#_Toc463594110)

[5.3.3. Cấu trúc CASE 116](#_Toc463594111)

[5.3.4. Một số lệnh điều khiển khác 118](#_Toc463594112)

[5.4. Batch 120](#_Toc463594113)

[5.5. Transact – SQL Scripts: 121](#_Toc463594114)

[BÀI TẬP CHƯƠNG 5 122](#_Toc463594115)

[A. CÂU HỎI 122](#_Toc463594116)

[B. BÀI TẬP 122](#_Toc463594117)

[CHƯƠNG 6 THỦ TỤC – HÀM 123](#_Toc463594118)

[MỤC ĐÍCH VÀ YÊU CẦU 123](#_Toc463594119)

[NỘI DUNG 124](#_Toc463594120)

[6.1. Thủ tục (Stored Procedure - SP) 124](#_Toc463594121)

[6.1.1. Tổng quan về thủ tục 124](#_Toc463594122)

[6.1.2. Thao tác trên thủ tục 125](#_Toc463594123)

[6.1.3. Sử dụng tham số trong thủ tục 129](#_Toc463594124)

[6.2. Hàm (Function) 134](#_Toc463594125)

[6.2.1. Khái niệm 134](#_Toc463594126)

[6.2.2. Phân loại hàm 134](#_Toc463594127)

[6.2.3. Tạo lập và sử dụng hàm 140](#_Toc463594128)

[BÀI TẬP CHƯƠNG 6 145](#_Toc463594129)

[A. CÂU HỎI 145](#_Toc463594130)

[B. BÀI TẬP 145](#_Toc463594131)

[CHƯƠNG 7 RÀNG BUỘC TOÀN VẸN CAO CẤP VỚI TRIGGER. 147](#_Toc463594132)

[MỤC ĐÍCH VÀ YÊU CẦU 147](#_Toc463594133)

[NỘI DUNG 148](#_Toc463594134)

[7.1. Giới thiệu 148](#_Toc463594135)

[7.1.1. Khái niệm: 148](#_Toc463594136)

[7.1.2. Phân loại 149](#_Toc463594137)

[7.2. Quản lý Trigger: 150](#_Toc463594138)

[7.2.1. Tạo Trigger: 150](#_Toc463594139)

[7.2.2. Nguyên tắc hoạt động và các bước cài đặt Trigger 151](#_Toc463594140)

[7.2.3. Sửa, Xóa một Trigger 153](#_Toc463594141)

[7.3. Một số ví dụ về Trigger 154](#_Toc463594142)

[7.4. Trigger lồng 164](#_Toc463594143)

[BÀI TẬP CHƯƠNG 7 166](#_Toc463594144)

[A. CÂU HỎI 166](#_Toc463594145)

[B. BÀI TẬP 166](#_Toc463594146)

[CHƯƠNG 8 CON TRỎ (CURSOR), KHUNG NHÌN (VIEW), CHỈ MỤC (INDEX), GIAO TÁC (TRANSACTION) VÀ KHÓA (LOCK). 169](#_Toc463594147)

[MỤC ĐÍCH VÀ YÊU CẦU 169](#_Toc463594148)

[NỘI DUNG 170](#_Toc463594149)

[8.1. Con trỏ (Cursor) 170](#_Toc463594150)

[8.1.1. Khái niệm về Cursor 170](#_Toc463594151)

[8.1.2. Định nghĩa Cursor 171](#_Toc463594152)

[8.1.3. Mở Cursor 174](#_Toc463594153)

[8.1.4. Đọc và xử lý dữ liệu trong Cursor 174](#_Toc463594154)

[8.1.5. Đóng Cursor 179](#_Toc463594155)

[8.2. Khung nhìn (View) 180](#_Toc463594156)

[8.2.1. Khái niệm 180](#_Toc463594157)

[8.2.2. Phân loại View 180](#_Toc463594158)

[8.2.3. Tạo View 181](#_Toc463594159)

[8.2.4. Quản lý View 184](#_Toc463594160)

[8.3. Chỉ mục (Index) 185](#_Toc463594161)

[8.3.1. Khái niệm 185](#_Toc463594162)

[8.3.2. Phân loại: 186](#_Toc463594163)

[8.3.3. Tạo Index 186](#_Toc463594164)

[8.4. Giao tác (Transaction) và Khóa (Lock) 188](#_Toc463594165)

[8.4.1. Khái niệm về Transaction 188](#_Toc463594166)

[8.4.2. Các lệnh thường sử dụng trong Transaction 189](#_Toc463594167)

[8.4.3. Khóa (Lock) 190](#_Toc463594168)

[BÀI TẬP CHƯƠNG 8. 192](#_Toc463594169)

[A. CÂU HỎI 192](#_Toc463594170)

[B. BÀI TẬP 192](#_Toc463594171)

[CHƯƠNG 9 BẢO MẬT (SECURITY) 196](#_Toc463594172)

[MỤC ĐÍCH VÀ YÊU CẦU 196](#_Toc463594173)

[NỘI DUNG 197](#_Toc463594174)

[9.1. Cơ chế bảo mật trong SQL Server 197](#_Toc463594175)

[9.1.1. Hai lớp bảo mật trong SQL Server 197](#_Toc463594176)

[9.1.2. Các khái niệm về bảo mật trong SQL Server 198](#_Toc463594177)

[9.2. Tạo các tài khoản đăng nhập 202](#_Toc463594178)

[BÀI TẬP CHƯƠNG 9 205](#_Toc463594179)

[A. CÂU HỎI 205](#_Toc463594180)

[B. BÀI TẬP 205](#_Toc463594181)

# DANH MỤC HÌNH ẢNH

[Hình 1.1. Lược đồ 3 mức trừu tượng dữ liệu 13](#_Toc459057632)

[Hình 1. 2. Nền tảng dữ liệu của Microsoft 15](#_Toc459057633)

[Hình 1. 3. Hộp thoại Installation 17](#_Toc459057634)

[Hình 1. 4. Hộp thoại Setup Support file 18](#_Toc459057635)

[Hình 1. 5. Hộp thoại Installation Type 19](#_Toc459057636)

[Hình 1. 6. Hộp thoại nhập Product Key 20](#_Toc459057637)

[Hình 1. 7. Hộp thoại Feature Selection 21](#_Toc459057638)

[Hình 1. 8. Hộp thoại Instance Configuration 22](#_Toc459057639)

[Hình 1. 9. Hộp thoại Server Configuration 23](#_Toc459057640)

[Hình 1. 10. Hộp thoại Database Engine Configuration 24](#_Toc459057641)

[Hình 1. 11. Hộp thoại Reporting Services Configuration 25](#_Toc459057642)

[Hình 1. 12. Sơ đồ cấu trúc lưu trữ database 28](#_Toc459057643)

[Hình 1. 13. Cấu trúc lưu trữ vật lý trong CSDL 31](#_Toc459057644)

[Hình 1. 14. Giao diện Enterprise Manager 32](#_Toc459057645)

[Hình 1. 15. Cửa sổ Query Analyzer 33](#_Toc459057646)

[Hình 1. 16. Thao tác tạo Database 35](#_Toc459057647)

[Hình 1. 17. Màn hình khai báo thông tin cho database 35](#_Toc459057648)

[Hình 2. 1. Thao tác import dữ liệu 53](#_Toc459057649)

[Hình 2. 2. Hộp thoại SQL Server Import and Export Wizard 53](#_Toc459057650)

[Hình 2. 3. Màn hình chọn dữ liệu nguồn 54](#_Toc459057651)

[Hình 2. 4. Hộp thoại chọn dữ liệu đích 55](#_Toc459057652)

[Hình 2. 5. Giao diện chọn bảng cần Import 56](#_Toc459057653)

[Hình 7. 1. Cơ chế hoạt động của Trigger After/for 148](#_Toc459057665)

[Hình 7. 2. Cơ chế hoạt động của Trigger instead of 148](#_Toc459057666)

[Hình 7. 3. Minh họa trigger cho thao tác Insert 150](#_Toc459057667)

[Hình 7. 4. Thêm một dòng dữ liệu 150](#_Toc459057668)

[Hình 7. 5. Cài đặt trigger minh họa thêm 1 nhân viên 153](#_Toc459057669)

[Hình 7. 6. Cài đặt trigger có dùng lệnh ROLLBACK TRANSACTION 154](#_Toc459057670)

[Hình 7. 7. Minh họa cài đặt trigger cho thao tác Update 155](#_Toc459057671)

[Hình 7. 8. Minh họa cài đặt trigger instead of cho hành động insert 155](#_Toc459057672)

[Hình 7. 9. Minh họa cài đặt trigger cho hành động Delete 157](#_Toc459057673)

[Hình 7. 10. Minh họa cài đặt trigger instead of cho hành động Delete 158](#_Toc459057674)

[Hình 7. 11. Minh họa cài đặt trigger after cho hành động Insert 159](#_Toc459057675)

[Hình 7. 12. Minh họa cài đặt trigger after cho hành động Update 160](#_Toc459057676)

[Hình 7. 13. Minh họa cài đặt trigger after cho hành động Insert, Update 161](#_Toc459057677)

[Hình 7. 14. Minh họa hiệu chỉnh cài đặt trigger after cho hành động Insert 162](#_Toc459057678)

[Hình 7. 15. Minh họa cài đặt trigger after cho hành động Delete 162](#_Toc459057679)

[Hình 8. 1. Cơ chế duyệt mẫu tin của một cursor 170](#_Toc459057680)

[Hình 8. 2. Minh họa định hướng duyệt tin của cursor 174](#_Toc459057681)

[Hình 8. 3. Minh họa cài đặt biến Cursor 176](#_Toc459057682)

[Hình 8. 4. Minh họa thao tác duyệt mẫu tin và thêm dữ liệu bằng Cursor 178](#_Toc459057683)

[Hình 8. 5. Kết quả sau khi thêm dòng dùng biến Cursor 178](#_Toc459057684)

[Hình 8. 6. Minh họa cài đặt View 182](#_Toc459057685)

[Hình 8. 7. Minh họa cài đặt View có tùy chọn With check option 182](#_Toc459057686)

[Hình 8. 8. Màn hình chi tiết tiến trinh Excution plan 186](#_Toc459057687)

[Hình 8. 9. Minh họa cài đặt Transaction 188](#_Toc459057688)

[Hình 9. 1. Thao tác tạo tài khoản đăng nhập 201](#_Toc459057689)

[Hình 9. 2. Màn hình thiết lập tài khoản đăng nhập 202](#_Toc459057690)

[Hình 9. 3. Kiểm tra thời hạn hiệu lực của password đăng nhập 203](#_Toc459057691)

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

## MỤC ĐÍCH VÀ YÊU CẦU

**Mục đích:**

Nội dung chương 1 nhằm cung cấp cho sinh viên các kiến thức tổng quan về:

* Giới thiệu tổng quan về vai trò, chức năng và đặc trưng của hệ quản trị cơ sở dữ liệu (HQT CSDL).
* Giới thiệu chung về HQT CSDL Microsoft SLQ SERVER 2008: cách cài đặt, vận hành, thao tác trên phần mềm.
* Cú pháp lệnh tạo ra cơ sở dữ liệu và lệnh tạo bảng.
* Các phương pháp phân tích, tổ chức cơ sở dữ liệu trong SQL Server.

**Yêu cầu:**

Sau khi học xong chương này sinh viên có khả năng:

* Biết cách cài đặt và sử dụng phần mềm MS SQL SERVER 2008, hiểu cơ chế vận hành và nguyên tắc hoạt động của một HQT CSDL;
* Biết sử dụng thành thạo công cụ tạo cơ sở dữ liệu;
* Biết và hiểu rõ cách sử dụng câu lệnh để tạo cơ sở dữ liệu, tạo bảng;
* Biết dùng lệnh để thay đổi cấu trúc bảng trên cơ sở dữ liệu

## NỘI DUNG

## Giới thiệu hệ quản trị cơ sở dữ liệu

### Giới thiệu chung

**SQL (Structured Query Language, đọc là "sequel") là tập lệnh truy xuất trên cơ sở dữ liệu (CSDL) quan hệ.**

Ngôn ngữ SQL được IBM sử dụng đầu tiên trong hệ quản trị CSDL System R vào giữa những năm 70, hệ ngôn ngữ SQL đầu tiên (SEQUEL2) được IBM công bố vào tháng 11 năm 1976. Năm 1979, tập đoàn ORACLE giới thiệu thương phẩm đầu tiên của SQL, SQL cũng được cài đặt trong các hệ quản trị CSDL như DB2 của IBM và SQL/DS.

Ngày nay, SQL được sử dụng rộng rãi và đuợc xem là ngôn ngữ chuẩn để truy cập CSDL quan hệ. Và SQL là ngôn ngữ vận hành chủ yếu trên các phần mềm được gọi là hệ quản trị cơ sở dữ liệu.

Hệ quản trị cơ sở dữ liệu (Database Management System – DBMS) là một hệ thống các phần mềm cho phép người dùng tạo lập cơ sở dữ liệu (định nghĩa, xây dựng, thao tác và quản trị) trên nó và được quyền điều khiển mọi truy cập trên cơ sở dữ liệu đó.

Hiện nay có rất nhiều phần mềm HQT CSDL được phát triển như: ORACLE, MS ACCESS, DB2, SQL SERVER, …

HQT CSDL quan hệ là HQT CSDL được phát triển theo mô hình quan hệ 🡪 được sử dụng rộng rãi và phổ biến hiện nay.

### Một số khái niệm cơ bản

**Cơ sở dữ liệu (CSDL) – database**

**Một CSDL (database) tương tự một tập tin dữ liệu, nơi để lưu trữ dữ liệu. Giống như một tập tin dữ liệu, một CSDL không trình bày thông tin một cách trực tiếp đến người dùng, muốn xem dữ liệu, người dùng phải chạy một ứng dụng truy xuất dữ liệu từ CSDL và khi đó dữ liệu được trình bày trong một dạng dễ hiểu và thuận tiện.**

Khi dữ liệu được lưu trong các tập tin thì phải có ứng dụng (được lập trình) để làm việc với cấu trúc đặc biệt của mỗi tập tin dữ liệu. Ngược lại, một CSDL chứa dựng một catalog mà các ứng dụng có thể dùng để xác định dữ liệu được tổ chức như thế nào. Các ứng dụng CSDL cùng loại có thể dùng catalog để trình bày đến người dùng các dữ liệu từ các CSDL khác nhau một cách linh động mà không bị phụ thuộc vào một dạng mẫu đặc biệt.

Các hệ thống CSDL mạnh tức là các tập tin dữ liệu được tổ chức theo một mức độ cao. Một CSDL được thiết kế tốt sẽ không có sự trùng lắp thông tin khi người dùng hoặc ứng dụng cập nhật tại cùng thời điểm. Các mảnh thông tin liên quan được kết nhóm với nhau trong một cấu trúc đơn hoặc mẫu tin và các mối quan hệ có thể được định nghĩa giữa các cấu trúc này và các mẫu tin này.

**CSDL quan hệ (Relational Database)**

Mặc dù có nhiều cách khác nhau để tổ chức dữ liệu trong một CSDL, các CSDL quan hệ là CSDL hiệu quả nhất. Các hệ thống CSDL quan hệ là một ứng dụng thuộc về lý thuyết tập hợp toán học đối với vấn đề tổ chức dữ liệu một cách hiệu quả. Trong một CSDL quan hệ, dữ liệu được thu thập vào các bảng (table).

Một bảng trình bày lớp các đối tượng quan trọng đối với một tổ chức. Mỗi bảng được xây dựng thành các cột và các dòng (được gọi là các thuộc tính và các bộ trong lý thuyết quan hệ). Mỗi cột trình bày vài thuộc tính của đối tượng được thể hiện trong bảng.

Ví dụ, bảng NHANVIEN sẽ có các cột điển hình đối với các thuộc tính như họ nhân viên, tên lót, tên nhân viên, mã nhân viên, phòng ban, bậc lương, công việc … Mỗi dòng trình bày một thể hiện của đối tượng thể hiện trong bảng.

Khi tổ chức dữ liệu vào các bảng, ta có nhiều cách khác nhau để định nghĩa các bảng. Lý thuyết CSDL quan hệ định nghĩa một phương thức gọi là dạng chuẩn (normalization) mà nó đảm bảo rằng tập các bảng mà ta định nghĩa sẽ được tổ chức dữ liệu một cách hiệu quả.

**Ngôn ngữ truy vấn dữ liệu SQL**

Để làm việc với dữ liệu trong một CSDL, ta phải dùng một tập các lệnh và câu lệnh (một ngôn ngữ) được hỗ trợ bởi DBMS. Ta có thể dùng vài ngôn ngữ khác nhau với CSDL quan hệ; thông thường nhất là SQL. Cú pháp lệnh của SQL được hỗ trợ bởi SQL Server được gọi Transact-SQL, và Transact-SQL là ngôn ngữ chính được dùng bởi hệ quản trị CSDL SQL Server.

**XML**

Extensible Markup Language (XML) là một ngôn ngữ lập trình siêu văn bản, dùng để mô tả nội dung của của tập dữ liệu và cách dữ liệu sẽ được kết xuất ra thiết bị hoặc hiển thị trên trang web.

### Các tính năng và đặc điểm chính của HQT CSDL

#### Các tính năng HQT CSDL

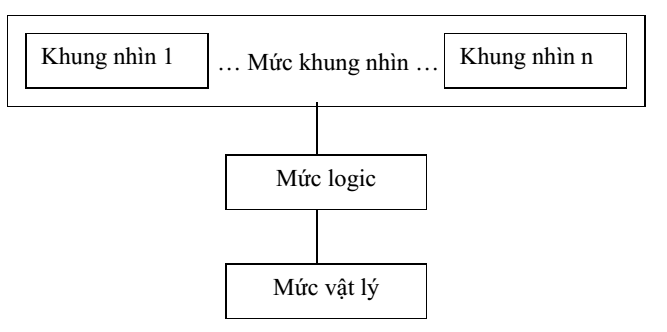
HQT CSDL có 8 tính năng chính như sau:

1. Khả năng quản lý dữ liệu tồn tại lâu dài: đặc điểm này chỉ ra rằng có một cơ sở dữ liệu tồn tại trong một thời gian dài, nội dung của cơ sở dữ liệu này là các dữ liệu mà hệ quản trị CSDL truy nhập và quản lý.
2. Khả năng truy nhập các khối lượng dữ liệu lớn một cách hiệu quả.
3. Hỗ trợ ít nhất một mô hình dữ liệu hay một sự trừu tượng toán học mà qua đó người sử dụng có thể quan sát dữ liệu.
4. Ðảm bảo tính độc lập dữ liệu hay sự bất biến của chương trình ứng dụng đối với các thay đổi về cấu trúc trong mô hình dữ liệu.
5. Hỗ trợ các ngôn ngữ cao cấp nhất định cho phép người sử dụng định nghĩa cấu trúc dữ liệu, truy nhập dữ liệu và thao tác dữ liệu.
6. Quản lý giao dịch, có nghĩa là khả năng cung cấp các truy nhập đồng thời, đúng đắn đối với CSDL từ nhiều người sử dụng tạicùng một thời điểm.
7. Ðiều khiển truy nhập, có nghĩa là khả năng hạn chế truy nhập đến các dữ liệu bởi những người sử dụng không được cấp phép và khả năng kiểm tra tính đúng đắn của CSDL.
8. Phục hồi dữ liệu, có nghĩa là có khả năng phục hồi dữ liệu, không làm mất mát dữ liệu với các lỗi hệ thống.

#### Đặc điểm của một HQT CSDL

* **Sự trừu tượng hoá dữ liệu:**

Mục đích của việc sử dụng CSDL quan hệ thay cho hình thức lưu trữ tập tin truyền thống là vấn đề lưu trữ và truy cập thông tin một cách dễ dàng và nhanh chóng. Chính vì vậy, việc thiết kế dữ liệu sẽ trở nên phức tạp để biểu diễn dữ liệu trong từng CSDL được tạo ra. Người phát triển che dấu tính phức tạp này thông qua một số mức trừu tượng để đơn giản hoá các tương tác của người sử dụng đối với hệ thống.



Hình 1.1. Lược đồ 3 mức trừu tượng dữ liệu

**Trong lược đồ trên:**

* ***Mức vật lý:*** Mức thấp nhất của sự trừu tượng mô tả dữ liệu được lưu trữ một cách thực sự như thế nào. Tại mức vật lý, các cấu trúc dữ liệu mức thấp phức tạp được mô tả chi tiết.
* ***Mức logic:*** Mức cao tiếp theo của sự trừu tượng hoá mô tả những dữ liệu nào được lưu trữ và các mối quan hệ nào tồn tại giữa các dữ liệu này. Mức logic của sự trừu tượng được xác định người quản trị CSDL, cụ thể phải quyết định những thông tin gì đượclưu trữ trong CSDL.
* ***Mức khung nhìn:*** Mức cao nhất của sự trừu tượng mô tả chỉ một phần của toàn bộ CSDL. Mặc dù sử dụng các cấu trúc đơn giản mức logic, một số phức tạp vẫn còn tồn tại do kích thước lớn của CSDL. Thực chất những người sử dụng chỉ cần truy nhập đến một phần CSDL, do vậy sự tương tác của họ với hệ thống này là đơn giản hoá và mức khung nhìn của sự trừu tượng được xác định. Hệ thống có thể được cung cấp nhiều khung nhìn đối với cùng một cơ sở dữ liệu.
* **Ngôn ngữ cơ sở dữ liệu**

Mỗi HQT CSDL luôn cung cấp cho người dùng một ngôn ngữ để thực hiện xây dựng, định nghĩa, truy vấn và quản trị thông tin trên cơ sở dữ liệu tương ứng được xây dựng trên nó.

Để làm việc với dữ liệu trong một CSDL, ta phải dùng một tập các lệnh và câu lệnh (một ngôn ngữ) được hỗ trợ bởi DBMS. Ta có thể dùng vài ngôn ngữ khác nhau với CSDL quan hệ; thông thường nhất là SQL. Cú pháp lệnh của SQL được hỗ trợ bởi SQL Server được gọi Transact-SQL, và Transact-SQL là ngôn ngữ chính được dùng bởi hệ quản trị CSDL SQL Server. Các câu lệnh T-SQL dùng để hiệu chỉnh hoặc truy xuất dữ liệu có thể được chia thành ba nhóm khác nhau:

1. *Ngôn ngữ định nghĩa dữ liệu* (Data Definition Language – DDL): Cho phép khai báo cấu trúc các bảng của CSDL, khai báo các mối liên hệ của dữ liệu (relationship) và các quy tắc áp đặt lên các dữ liệu đó (đã đề cập ở chương 1)
2. *Ngôn ngữ thao tác dữ liệu* (Data Manipullation Language - DML) cho phép người sử dụng có thể thêm (insert), xoá (delete), sửa (update) dữ liệu trong CSDL.
3. *Ngôn ngữ truy vấn dữ liệu (*hay ngôn ngữ hỏi đáp có cấu trúc (Structured Query Language - SQL)): Cho phép người sử dụng khai thác CSDL để truy vấn các thông tin cần thiết trong CSDL.
4. *Ngôn ngữ quản lý dữ liệu* (Data Control Language - DCL): Cho phép những người quản trị hệ thống thay đổi cấu trúc của các bảng dữ liệu (sẽ đề cập trong chương 1), khai báo bảo mật thông tin và cấp quyền khai thác CSDL cho người sử dụng (sẽ đề cập trong chương 9)

* **Xử lý yêu cầu truy vấn**

Công việc của bộ xử lý câu hỏi là biến đổi một truy vấn hay một thao tác CSDL có thể được biểu diễn ở các mức cao thành một dãy các yêu cầu đối với các dữ liệu lưu trữ trong CSDL.

Thường phần khó nhất của nhiệm vụ xử lý câu hỏi là tối ưu hoá câu hỏi, có nghĩa là lựa chọn một kế hoạch tốt nhất đối với hệ thống lưu trữ để trả lời truy vấn này nhanh nhất.

* **Quản trị giao dịch**

Thông thường một số thao tác trên CSDL hình thành một đơn vị logic công việc. Ðiều này có nghĩa là hoặc tất cả các thao tác được thực hiện hoặc không thao tác nào được thực hiện. Hơn nữa sự thực hiện các thao tác này phải đảm bảo tính nhất quán của CSDL.

Một giao dịch là một tập hợp các thao tác mà xử lý như một đơn vị không chia cắt được. Các hệ quản trị CSDL điển hình cho phép người sử dụng một hay nhiều nhóm thao tác tra cứu hay thay đổi CSDL thành một giao dịch.

* **Quản trị lưu trữ dữ liệu**

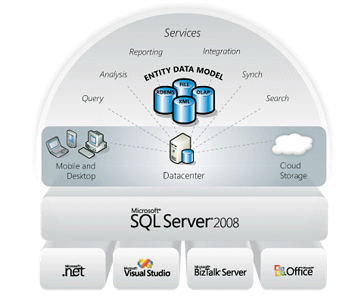
Các CSDL thường đòi hỏi một khối lượng lớn không gian lưu trữ. Do bộ nhớ chính của máy tính không thể lưu trữ nhiều thông tin như vậy, các thông tin này được lưu trữ ở các thiết bị nhớ ngoài như đĩa cứng, đĩa mềm,.v.v…

Khi xử lý, dữ liệu cần phải được di chuyển từ đĩa từ vào bộ nhớ chính; sự di chuyển này là khá chậm so với tốc độ xử lý của bộ nhớ trung tâm, do vậy các hệ CSDL phải tổ dữ liệu vật lý sao cho tốt, tối thiểu hoá số yêu cầu chuyển dữ liệu giữa đĩa từ vào bộ nhớ chính.

### Hệ quản trị cơ sở dữ liệu Microsoft SQL Server 2008

#### Đặc trưng của Microsoft SQL Server 2008

Microsoft SQL Server 2008 là phần mềm được sử dụng khá phổ biến và rộng rãi hiện nay. Với vai trò là một HQT CSDL, phiên bản MS SQL Server 2008 được biết đến với một số tính năng nổi bật như sau:



Hình 1. 2. Nền tảng dữ liệu của Microsoft

* **Nền tảng chính cho các nhiệm vụ then chốt trong việc lưu trữ và tổ chức dữ liệu như**:

1. Cho phép chạy hầu hết các ứng dụng phức tạp của người dùng trên một nền tảng an toàn, tin cậy và có khả năng mở rộng.
2. Giảm được sự phức tạp trong việc quản lý cơ sở hạ tầng dữ liệu.
3. Cung cấp một nền tảng tin cậy và an toàn bằng cách bảo đảm những thông tin có giá trị trong các ứng dụng đang tồn tại và nâng cao khả năng sẵn có của dữ liệu.
4. Giới thiệu một cơ chế quản lý cách tân dựa trên chính sách, cơ chế này cho phép các chính sách có thể được định nghĩa quản trị tự động cho các thực thể máy chủ trên một hoặc nhiều máy chủ.
5. Cho phép thi hành truy vấn dự báo với một nền tảng tối ưu.

* **Đặc trưng bởi nền tảng phát triển động:**

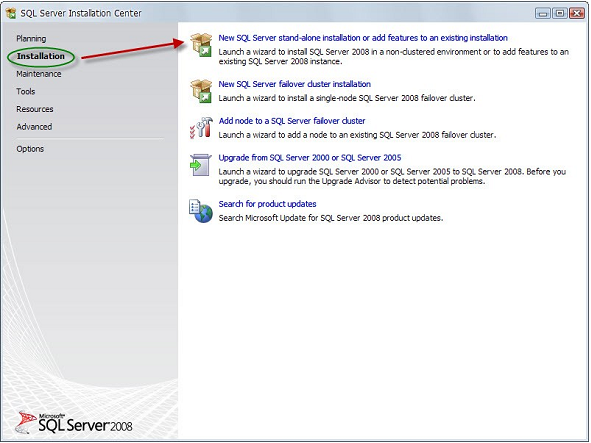
SQL Server 2008 cùng với .NET Framework đã giảm được sự phức tạp trong việc phát triển các ứng dụng mới. ADO.NET Entity Framework cho phép các chuyên gia phát triển phần mềm có thể nâng cao năng suất bằng làm việc với các thực thể dữ liệu logic đáp ứng được các yêu cầu của doanh nghiệp thay vì lập trình trực tiếp với các bảng và cột. Các mở rộng của ngôn ngữ truy vấn tích hợp (LINQ) mới trong .NET Framework đã cách mạng hóa cách các chuyên gia phát triển truy vấn dữ liệu bằng việc mở rộng Visual C#® và Visual Basic® .NET để hỗ trợ cú pháp truy vấn giống SQL vốn đã có. Hỗ trợ cho các hệ thống kết nối cho phép chuyên gia phát triển xây dựng các ứng dụng cho phép người dùng mang dữ liệu cùng với ứng dụng này vào các thiết bị và sau đó đồng bộ dữ liệu của chúng với máy chủ trung tâm.

* **Phát triển dữ liệu quan hệ mở rộng:** cho phép các chuyên gia phát triển khai thác triệt để và quản lý bất kỳ kiểu dữ liệu nào từ các kiểu dữ liệu truyền thống đến dữ liệu không gian địa lý mới.
* **Hỗ trợ thông tin trong toàn bộ doanh nghiệp:** cung cấp một cơ sở hạ tầng có thể mở rộng, cho phép quản lý các báo cáo, phân tích với bất kỳ kích thước và sự phức tạp nào, bên cạnh đó nó cho phép người dùng dễ dàng hơn trong việc truy cập thông tin thông qua sự tích hợp sâu hơn với Microsoft Office. Điều này cho phép CNTT đưa được thông tin của doanh nghiệp rộng khắp trong tổ chức. SQL Server 2008 tạo những bước đi tuyệt vời trong việc lưu trữ dữ liệu, cho phép người dùng hợp nhất các trung tâm dữ liệu vào một nơi lưu trữ dữ liệu tập trung của toàn doanh nghiệp.

#### Hướng dẫn cài đặt Microsoft SQL Server 2008

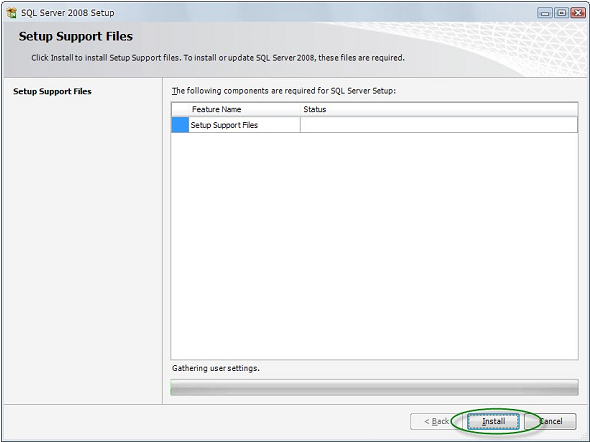
Các bước cài đặt Microsoft SQL Server 2008:

**Bước 1:** Chạy file setup.exe để cài đặt, chọn Installation 🡪New SQL Server stand-alone ...



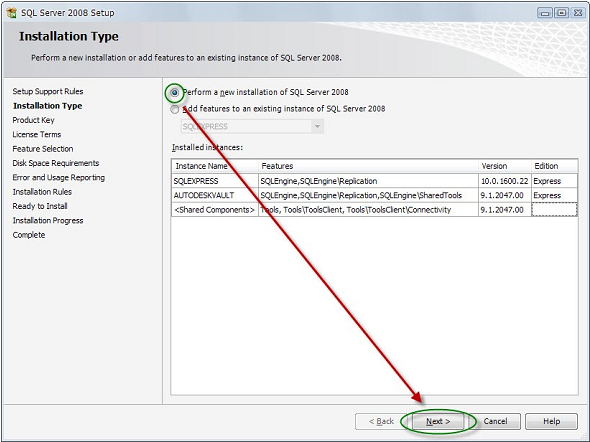
Hình 1. 3. Hộp thoại Installation

**Bước 2:** Chọn OK 🡪 Nhấn Next 🡪 Xuất hiện hộp thoại Setup Support Files 🡪 Chọn Install



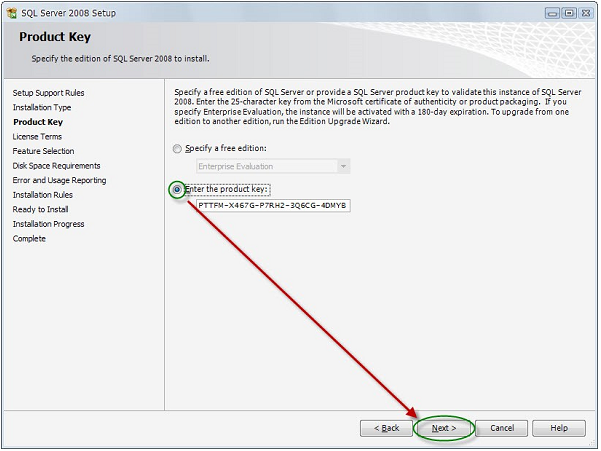
Hình 1. 4. Hộp thoại Setup Support file

**Bước 3:** Chọn Perform a new installation of SQL Server 2008 🡪 Nhấn Next



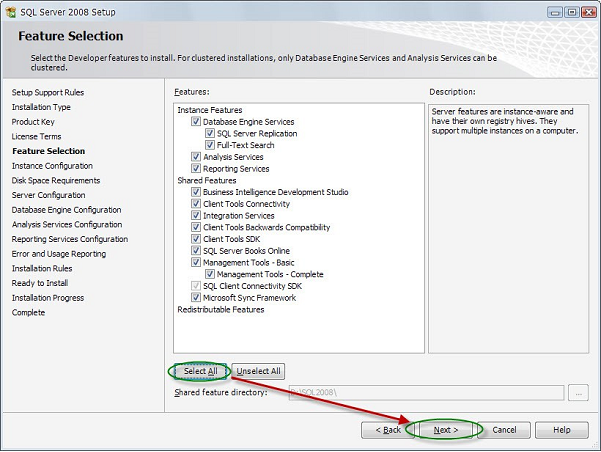
Hình 1. 5. Hộp thoại Installation Type

**Bước 4:** Nhập Product Key 🡪 Nhấn Next



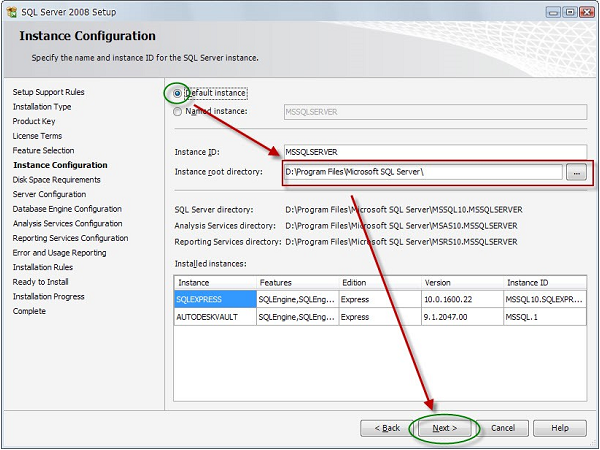
Hình 1. 6. Hộp thoại nhập Product Key

**Bước 5:** Sau khi đồng ý License Terms 🡪 chọn các thành phần cài đặt



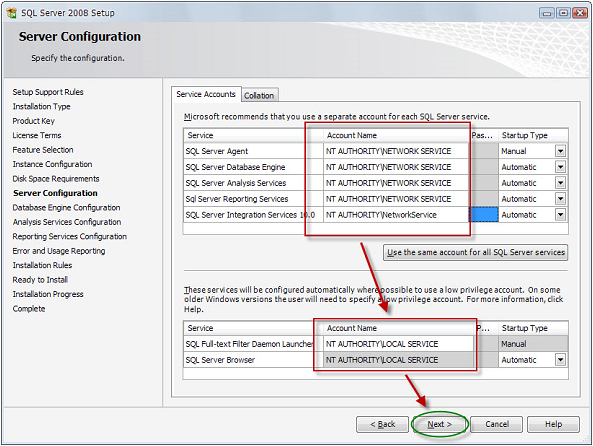
Hình 1. 7. Hộp thoại Feature Selection

**Bước 6:** Thiết lập cài đặt chọn Default instance



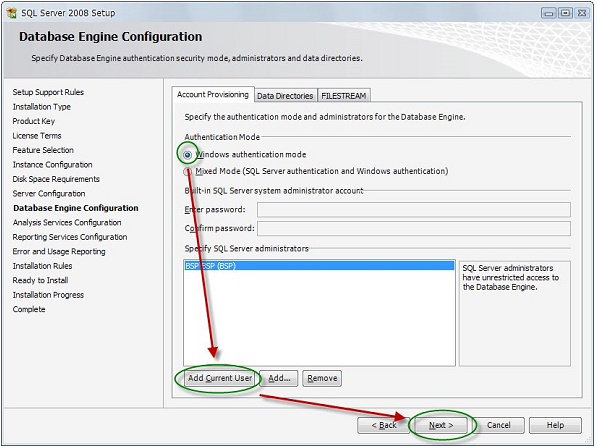
Hình 1. 8. Hộp thoại Instance Configuration

**Bước 7:** Cấu hình Server



Hình 1. 9. Hộp thoại Server Configuration

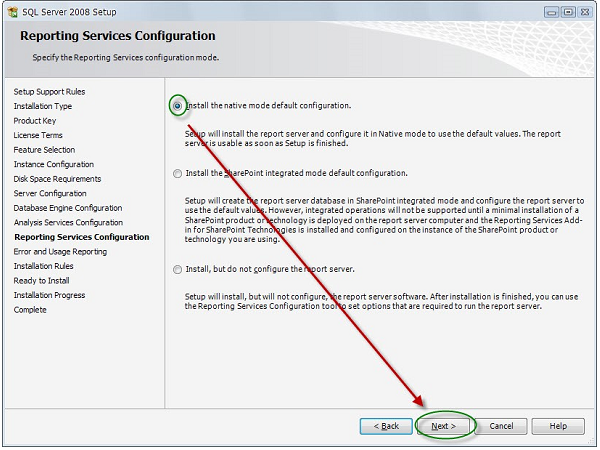
**Bước 8:** Cấu hình dữ liệu như sau chọn Window Authentication và Add current User



Hình 1. 10. Hộp thoại Database Engine Configuration

**Bước 9:** Cấu hình analysis services Add Current User

**Bước 10:** Cấu hình report: chọn option như hình 🡪 nhấn Next, Next … Cho đến khi hoàn tất



Hình 1. 11. Hộp thoại Reporting Services Configuration

### Các thành phần cơ bản của hệ quản trị cơ sở dữ liệu (HQT CSDL)

#### Phân loại Server Type

Để kết nối đến một database, máy tính cần kết nối đến server chứa database đó. Tùy theo mục đích, thao tác làm việc với database mà SQL Server 2008 hỗ trợ các thể hiện của Server đó như sau:

**Database Engine**: là bộ máy cốt lõi trong việc quản lý các database trong SQL Server 2008, cho phép thực hiện thao tác tạo mới cũng như cập nhật CSDL đã có. Ngoài ra Database Engine còn hỗ trợ gắn kết một database mới vào Server cũng như tháo gỡ một database ra khỏi Server.

**Analysis Services**: hỗ trợ các thao tác phân tích dữ liệu, cho phép xây dựng một data source (nguồn dữ liệu) với cấu trúc của một cube có nhiều dimension nhằm bổ nghĩa thông tin đầy đủ cho một database.

**Intergration Services**: Khi cần tập hợp dữ liệu từ nhiều nguồn khác nhau hay xây dựng một data warehouse (kho dữ liệu) từ nhiều nguồn dữ liệu (data source), hoặc khi cần đồng nhất việc quản trị một database đang được lưu trữ dưới nhiều dạng khác nhau (như file excel, Access, SQL, Oracle,...), Intergration Services sẽ hỗ trợ các thao tác để xử lý công việc trên.

**Reporting Services**: Hỗ trợ thao tác tạo, xuất bản cũng như quản lý số lượng lớn các báo cáo liên quan đến việc sử dụng một database.

#### Các thành phần của Server

Để Server có thể được thực thi, các service tương ứng cần được khởi động và ở trạng thái Started. Dưới đây là một số service quan trọng của SQL Server 2008:

**MSSQL Server (SQL Server):** Là một database engine. Để kết nối được SQL Server 2008 thì MSSQL Server phải đang được chạy. Phần lớn các chức năng của database engine đều được điều khiển bởi service này.

**SQLServerAgent (SQL Server Agent):** Phụ trách thực thi các thao tác theo lịch định sẵn như backup, import/export , …. Nếu có giao tiếp với mạng hoặc truy cập các file hệ thống, credential của service này thường được dùng.

**SQLBrowser (SQL Server Browser):** Mỗi instance trong SQL Server đều có port riêng khi giao tiếp với client, instance mặc định (lúc cài SQL Server) sẽ có port cố định là 1433. Các instance về sau sẽ có port ngẫu nhiên, SQLBrowser có chức năng dò port của các instance đó mỗi khi khởi động SQL Server.Nếu vì một lý do nào đó mà SQLBrowser không hoạt động hoặc bị disable thì ta sẽ phải gán thủ công port cho từng instance của SQL Server và điều chỉnh lại bên phía ứng dụng client.

**SQLWriter (SQL Server VSS Writer):** Dùng để back-up data file và log file trong thời gian thực (mặc dù các file này vẫn đang đựơc sử dụng khi  SQL Server đang vận hành)

**SQL Server Analysis Services, SQL Server Intergration Services, SQL Reporting** Servcies: Điểu khiển các chức năng tương ứng của một database

#### Các công cụ đồ họa (Graphical Tools)

**SQL Server Management Studio:** công cụ quản trị CSDL và server chính, cung cấp một giao diện Microsoft Management Console (MMC).

**SQL Profiler**: tạo cơ hội các người quản trị một công cụ tinh vi để theo dõi và phân tích giao thông mạng đến và đi từ một server đang chạy SQL Server.

**SQL Server Configuration Manager**: được dùng để start, stop, và pause các dịch vụ của SQL Server.

**SQL Server Books online**: là một tài liệu trực tuyến hỗ trợ với Microsoft® SQL Server™. Ta có thể tìm thông tin trong SQL Server Books Online bằng cách:

* Điều hướng thông qua bảng nội dung.
* Gõ một từ khóa trong index.
* Gõ một từ hoặc một cụm từ và thực hiện việc tìm kiếm.

### Kiến trúc CSDL quan hệ (relational Database Architecture)

Trong SQL Server, dữ liệu được lưu trong các CSDL.Về vật lý, một CSDL bao gồm hai hoặc nhiều hơn hai tập tin trên một hoặc nhiều đĩa, sự thể hiện vật lý này chỉ thấy được bởi nhà quản trị và nó trong suốt đối với người dùng. Sự chọn lựa vật lý của CSDL là trách nhiệm chính của người quản trị CSDL. Về mặc logic, một CSDL được xây dựng thành các thành phần (component) được hiển thị đối với người dùng, chẳng hạn các bảng (Table), các khung nhìn (view) và các thủ tục (stored procudure). Sự chọn lựa logic của CSDL (chẳng hạn như thiết kế các bảng và các chỉ mục) là trách nhiệm chính của người thiết kế CSDL.

Các CSDL chứa các dữ liệu của người dùng được gọi là CSDL của người dùng (userdatabase). Ngoài ra, các hoạt động của SQL Server tùy thuộc vào bốn CSDL hệ thống (system Database): master, model, Tempdb, vàmsdb. Các CSDL này tồn tại trong mỗi thể hiện của SQL Server.

CSDL hệ thống và CSDL người dùng

**CSDL Master**: đuợc dùng để điều khiển các thao tác và thông tin mức hệ thống của SQL Server. Lưu trữ thông tin hệ thống trong 16 bảng hệ thống, gọi là system catalog. System catalog lưu trữ các thông tin tài khoản người dùng, bao gồm các mục như là bảo mật, ID, mật khẩu, các CSDL lưu trên server, các biến môi trường, các thông điệp lỗi hệ thống, và các thủ tục hệ thống.

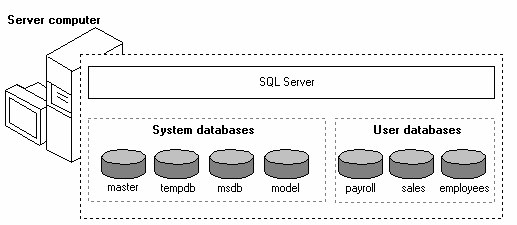
**CSDL Model**: được dùng như là một CSDL mẫu cho tất cả các CSDL mới được tạo ra trong hệ thống CSDL Model có thể được tùy biến để tạo một cấu trúc CSDL mặc định mới cho CSDL mới. Ví dụ, nếu một bảng cần phải tồn tại trong CSDL mới sau này thì các bảng đó nên được tạo trong CSDL model. Các CSDL mới vừa được tạo sẽ có kích cở ít nhất bằng kích cở CSDL model.

**CSDL Tempdb**: được dùng như là vùng lưu chứa tạm thời đối với các bảng và các thủ tục tạm. SQL Server hỗ trợ hai loại bảng tạm: bảng tạm toàn cục (globaltemporarytable) và bảng tạm cục bộ (local temporary table). Tên của bảng tạm toàn cục được bắt đầu##, có thể được truy xuất đối với tất cả các client, trong khi tên bảng tạm cụ bộ bắt đầu#, chỉ có thể truy xuất đối với những client mà tạo chúng. Làm việc trong Tempdb rất nhanh vì các hoạt động không được ghi nhận lại.Tuy nhiên, khi client kết thúc kết nối đến Server thì toàn bộ các bảng và thủ tục trong TempDB sẽ bị xóa.

**CSDL Msdb**: SQL Server Agent dùng CSDL msdb cho các tác vụ khác nhau như lập biểu, cảnh báo và ghi nhận các thao tác. Dữ liệu được lưu trong các bảng hệ thống trong CSDL msdb. Các bảng hệ thống được dùng bởi SQL Server Agent là *sysalerts*, *syscategories*, *sysdownloadlist*, *sysjobhistory*, *sysjobs*, *sysjobschedules*, *sysjobservers*, *sysjobsteps*, *sysnotifications*, *sysoperators*, *systargetservergroupmembers*, *systargetservergroups*, *systargetservers và systaskids*. Ngoài ra, CSDL msdb có 7 bảng dùng cho thao tác dự phòng và phục hồi dữ liệu: *backupfile, backupmediafamily, backupmediaset, backupset, restorefile, restorefilegroup, vàrestorehistory.* SQL Server có thể tự động tăng hoặc giảm kích cở của CSDL khi các dòng được thêm vào hoặc xóa đi.

**CSDL Pubs, Northwind**: là hai CSDL ví dụ được dùng trong các tài liệu của SQL Server (trong phiên bản MS SQL Server 2003 trở về trước)

**Các CSDL khác là các CSDL do người dùng tạo ra.**



Hình 1. 12. Sơ đồ cấu trúc lưu trữ database

**Một số thuật ngữ quan trọng trong cơ sở dữ liệu quan hệ: Databases, Tables, Columns, Views, Datatypes, và Database Schemas**

Dữ liệu trong một RDBMS được sắp xếp trong một số đối tượng, dễ thấy nhất là bảng (table). Dữ liệu được lưu trong các Table theo dòng và cột. Dữ liệu liên quan đến các mục thực tế như các nhân viên, các sản phẩm,việc gửi hàng, người tham gia…. được lưu trong các table riêng biệt.

Ví dụ, hệ thống quản lý nhân viên của một công ty có thể có một bảng gọi là nhân viên, bảng này dùng để lưu chi tiết tất cả của các nhân viên trong công ty. Chi tiết của nhân viên bao gồm Họ nhân viên, tên nhân viên, địa chỉ, thành phố, số chứng minh nhân dân. Thỉnh thoảng cần ẩn các thông tin riêng tư từ người dùng bằng cách dùng một truy vấn. Các truy vấn là các câu lệnh SQL lưu trong CSDL và có thể được tham chiếu đến các câu lệnh SQL theo cách như các table. SQL Server cũng sử dụng các kiểu dữ liệu, mỗi cột có thể có các kiểu khác nhau. Tất cả các kiểu khác nhau này của các đối tượng, giống như bảng, truy vấn và kiểu dữ liệu được lưu trong một CSDL.Trong môi trường SQL Server, một lược đồ CSDL (database schema) là tập hợp các đối tượng CSDL có liên quan đến việc sử dụng một tên duy nhất và nó thuộc về một người dùng đơn lẻ.

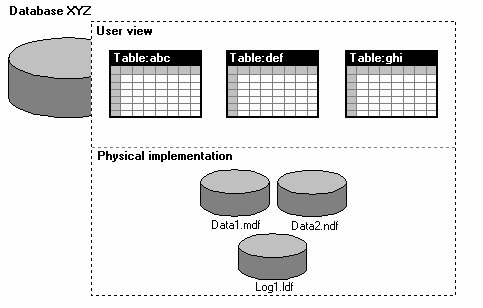
**Danh mục các đối tượng trong cơ sở dữ liệu**

|  |  |
| --- | --- |
| **Đối tượng** | **Mô tả** |
| Bảng - Table | Một table là một tập hợp của dữ liệu lưu trữ trong nhiều dòng nhiều cột 🡪 Được gọi là một quan hệ trong CSDL |
| Cột - Column | Cột là các đối tượng dùng lưu trữ các phân tử riêng biệt bên trong một dòng dữ liệu 🡪 Được gọi là thuộc tính trong quan hệ |
| Dòng – Row | Dòng là một mục đơn của dữ liệu bên trong một bảng, nó bao gồm tập các cột 🡪 Được gọi là một bộ trong quan hệ |
| Kiểu dữ liệu - Datatype | Kiểu dữ liệu là các đối tượng dùng để xác định loại của dữ liệu có thể được lưu trong cột |
| Chỉ mục – Index | Chỉ mục là một đối tượng của CSDL, dùng để tăng tốc các truy vấn bằng cách dò tìm dữ liệu theo giá trị khóa thay cho việc quét toàn bộ bảng.  SQL Server hỗ trợ các chỉ mục clustered và non-clustered. Chỉ mụcClustered (Clusteredindex) là các chỉ mục mà sắp xếp các dòng theo cách dùng B-Tree. |
| Bẫy lỗi – Trigger | Trigger được ưa chuộng bởi vài nhà lập trình. Chúng chính là các thủ tục mà có thể chạy khi dữ liệu trong bảng được hiệu chỉnh như là sửa, xóa, chèn. Các trigger thường được dùng để đảm bảo tính thống nhất giữa các bảng liên kết. Các trigger có thể được tạo trên bảng hoặc truy vấn. |
| Quy tắc - Rule | Rule là đối tượng CSDL, nó được gắn kết với một cột hoặc một định nghĩa người nhằm hạn chế các giá trị. Ví dụ, một rule có thể đảm bảo số chứng minh nhân dân chỉ gồm các chữ số. |
| Thủ tục-Stored  Procedures | Thủ tục là một tập các câu lệnh T-SQL được dịch trước (pre-compile) và được thực thi như một đối tượng. Do chúng đã được dịch trước nên các thủ tục sẽ thực thi nhanh. |

Quản lý dữ liệu chính là lưu trữ với sự trợ giúp của một số đối tượng được cung cấp cung cấp bởi SQL Server 2008. Bảng liệt kê các đối tượng chính trong một database của SQL Server 2008.

**Cấu trúc vật lý của một CSDL**

Mỗi CSDL bao gồm ít nhất một tập tin dữ liệu (datafile) và một tập tin log (transaction logfile). Các tập tin này không được chia sẻ với các CSDL khác. Để tối ưu hiệu năng và để hỗ trợ khả năng chịu lỗi (faulttolerance), tập tin dữ liệu và log được trải dài trên nhiều đĩa và thông thường dùng một Raid(Redundantarrayofindependent disks)



Hình 1. 13. Cấu trúc lưu trữ vật lý trong CSDL

**Bảo mật trong SQL Server**

SQL Server 2008 dùng bốn cơ chế sau để đảm bảo tính bảo mật trong việc quản lý mộtdatabase: Authentication, Login và user, Server Role, Database Role**Authentication**

SQL Server xác thực các đăng nhập bằng hai cơ chế, windows authentication và SQL Server authentication. Việc này bắt nguồn từ khi tạo login trong SQL Server. Khi ta tạo một login mới, SQL Server sẽ yêu cầu ta chọn cơ chế xác thực windows hay SQL Server cho login đó. Nếu chọn windows, ta cần cung cấp một windows account và SQL Server sẽ chỉ lưu tên của account đó trong danh sách login. Nếu chọn SQL Server authentication, ta cần cung cấp login name và password và cả hai đều được lưu trong SQL Server.

Khi đăng nhập vào SQL Server, ta cũng sẽ phải chọn một trong hai cơ chế xác thực. Nếu chọn windows, chính account hiện đang đăng nhập vào windows được dùng. Ta không có quyền chọn login và thực tế, SQL Server sẽ vòng xuống windows và kiểm tra xem account nào đang đăng nhập vào windows, và kiểm tra account đó có nằm trong danh sách login hay không, nếu có sẽ cho phép đăng nhập, nếu không sẽ chặn lại.

Đối với SQL Server authentication, cơ chế như sau: ta cung cấp login ID và password. SQL Server sẽ kiểm tra và quyết định có cho phép đăng nhập hay không.

Một điểm cần lưu ý là với login sử dụng windows authentication, SQL Server chỉ lưu login name và không hề lưu password. Và khi đăng nhập nếu ta cố tình chọn sang SQL Server authentication, hệ thống sẽ không xác thực.

Chúng ta sẽ tìm hiểu bảo mật trong SQL Server kỹ hơn ở trong chương bảo mật (chương 9)

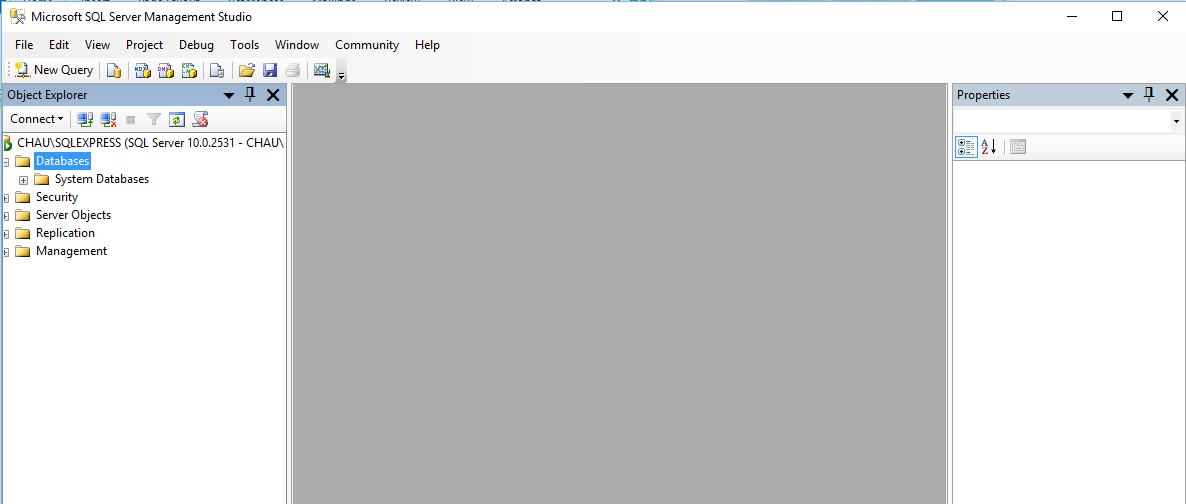
### Các công cụ và trình tiện ích.

#### ENTERPRISE MANAGER

Kể từ phiên bản SQL Server 2008 trở đi, SQL Server Management Studio được xây dựng dựa trên việc tích hợp Enterprise ManagervàSQL Query Analyzer lại trong cùng một giao diện đồ họa.

**Enterprise Manager**: (gọi tắt là EM) là một công cụ chính dành cho nhà quản trị server và CSDL. Enterprise Manager trong SQL Server 2008 chính là Object Explorer cho phép ta dừng và khởi động một Server, cũng như cho phép ta thực hiện các tác vụ sau:

* Đăng ký một server;
* Cấu hình các server cục bộ hoặc từ xa;
* Cấu hình và quản lý một cài đặt với nhiều server (multiple-server);
* Cài đặt các bảo mật đăng nhập (login security), thêm các người dùng (user), các nhà quản trị hệ thống (system administrator), và các điều hành viên;
* Gán một password nhà quản trị hệ thống;
* Tạo và lập biểu cho các công việc (job);
* Tạo các cảnh báo và cấu hình giao tiếp đến nhà quản trị hệ thống thông qua e- mail;
* Cài đặt và quản trị các CSDL, các bảng (table), các chỉ mục (index), các truy vấn (view), các thủ tục (stored procedure), các qui tắc (rule), các bẫy lỗi (trigger), các ràng buộc mặc định (default), các thiết bị dự phòng (backup device), và các vết lỗi (error log).

****

Hình 1. 14. Giao diện Enterprise Manager

#### SQL QUERY ANALYZER

SQL Query Analyzer là giao diện người dùng đồ họa (Graphical User Interface GUI) dành cho các nhà lập trình. Query Analyzer cho phép thực hiện:

* Tạo các truy vấn (query), bó lệnh (script) và thực thi (execute) chúng để tác động đến CSDL của SQL Server;
* Tạo các đối tượng của CSDL một cách nhanh chóng từ những script được định nghĩa trước;
* Sao chép nhanh chóng các đối tượng của CSDL;
* Tạo và thực thi các thủ tục (Stored procedures), hàm người dùng (user-defined function);
* Tìm lỗi (Debug) các thủ tục;
* Tìm lỗi (Debug) các vấn đề hiệu năng của của truy vấn (Show Execution Plan, Show Server Trace, Show Client Statistics, Index Tuning Wizard);
* Định vị các đối tượng trong các CSDL, xem và làm việc với các đối tượng;
* Chèn, cập nhật, xóa các mẫu tin trong table một cách nhanh chóng;

**Thành phần chính của Query Analyzer**

SQL Query Analyzer bao gồm các cửa sổ, hộp thoại, hướng dẫn (wizard) giúp chúng ta thiết kế các tác vụ (Task) cần thiết để tạo các CSDL, lưu trữ, khai thác dữ liệu trong các CSDL đó.



Hình 1. 15. Cửa sổ Query Analyzer

**Màu của mã lệnh trong Query Analyzer:**

|  |  |
| --- | --- |
| Màu | Ý nghĩa |
| Đỏ | Chuỗi ký tự |
| Đỏ đậm | Thủ tục |
| Xanh lá | Bảng hệ thống |
| Xanh lá đậm | Chú giải |
| Đỏ tươi | Hàm hệ thống |
| Xanh | Từ khóa |
| Xám | Toán tử |

* **Nhóm phím tắt dùng trong Query Analyzer**

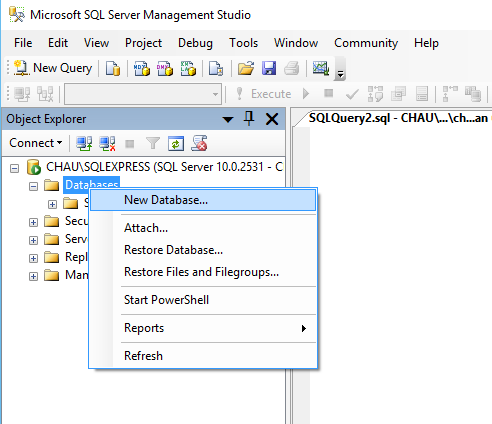
|  |  |
| --- | --- |
| **Hành động** | **Phím nóng** |
| Thực thi | CTRL + E/F5 |
| Tìm kiếm | CTRL + F |
| Chuyển văn bản đang chọn thành chữ hoa | CTRL + SHIFT + U |
| Chuyển văn bản đang chọn thành chữ thường | CTRL + SHIFT + L |
| Các kết quả dạng văn bản | CTRL + T |
| Những kết quả trong lưới | CTRL + D |
| Trợ giúp trong Query Analyzer | F1 |
| Trợ giúp một câu lệnh T-SQL được chọn | SHIFT + F1 |

## Tạo và quản trị cơ sở dữ liệu.

### Cơ sở dữ liệu (CSDL) và các tập tin lưu trữ.

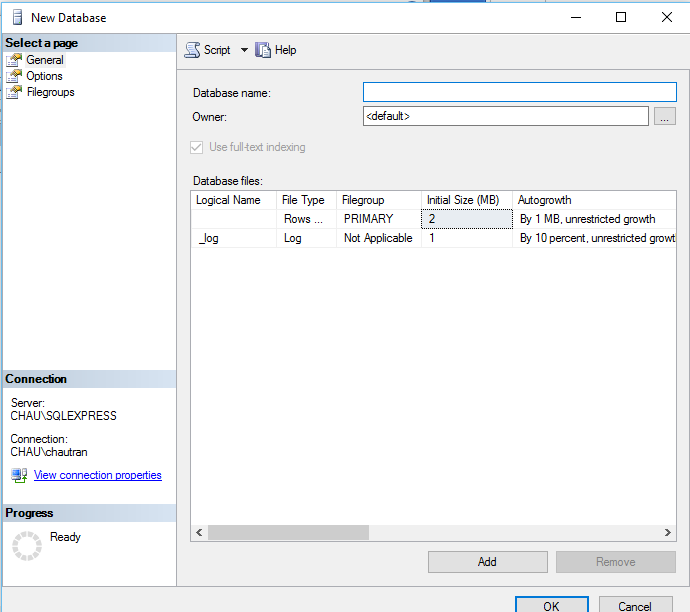
#### Tạo ở tại cửa sổ Object Explorer

**Bước 1:** Click\_Right 🡪 chọn Database



Hình 1. 16. Thao tác tạo Database

**Bước 2:** Khai báo thông tin cho Database cần tạo



Hình 1. 17. Màn hình khai báo thông tin cho database

General

* Database Name:<Tên logic của CSDL>
* Logical Name:<Tên của tập tin Datafile,vị trí lưu tập tin trên đĩa>
* Initialsize:<Kích cỡ khởi tạoCSDL>
* FileGroup:<Tên của FileGroup>
* Autogrowth: Khai báo một số thuộc tính khác như tỉ lệ gia tăng (File
* Growth), kích cỡ tối đa (maximunsize)

**Bước 3:** nhấn OK

#### Tạo và cấu hình CSDL bằng viết lệnh.

**Cú pháp đơn giản**

**CREATE DATABASE** *database\_name*

Trong đó:

* *database\_name*: tên cơ sở dữ liệu cần tạo – được đặt tên theo quy tắc đặt tên định danh

**Ví dụ 1:** tạo CSDL  **CREATE DATABASE QUANLYDEANCONGTY**

**Cú pháp chi tiết**

**CREATE DATABASE** *database\_name*

**ON PRIMARY**

(

[NAME = *logical\_file\_name,*] FILENAME = '*os\_file\_name*'

[, SIZE = *size*][, MAXSIZE = {*max\_size*|UNLIMITED}]

[, FILEGROWTH = *growth\_increment*])[, ...*n*]<filegroup> ::= FILEGROUP filegroup*\_name* <filespec>[, ...*n*])

**LOG ON**

**(**

[NAME = *logical\_file\_name,*] FILENAME = '*os\_file\_name*'

[, SIZE = *size*][, MAXSIZE = {*max\_size*|UNLIMITED}]

[, FILEGROWTH = *growth\_increment*])[, ...*n*] <filegroup> ::= FILEGROUP *filegroup\_name*<filespec>[, ...*n*])

**Ví dụ 2: tạo database quản lý đề án công ty**

**CREATE DATABASE** QUANLYDEANCONGTY

**ON PRIMARY**

( NAME= QUANLYDEANCONGTY \_DATA,

FILENAME=’C\MSSQL\DATA\ QUANLYDEANCONGTY \_DATA.MDF’,

SIZE=50MB,

MAXSIZE=200MB,

FILEGROWTH=10% )

LOG ON

(

NAME= QUANLYDEANCONGTY \_LOG,

FILENAME=’C: \MSSQL\DATA\ QUANLYDEANCONGTY \_LOG.LDF’,

SIZE=10MB,

MAXSIZE=UNLIMITED,

FILEGROWTH=5MB

)

### Quản trị database

Một số thao tác giúp người dùng quản trị một database bao gồm: kiểm tra sự tồn tại của database đã tạo, thêm – xóa hoặc sửa một database có sẵn.

Cụ thể như sau:

* Kiểm tra sự tồn tại của database

**Cú pháp**

**SP\_HELP DATABASE\_NAME**

**Ví dụ:** Kiểm tra sự tồn tại của database quản lý đề án công ty

sp\_help QUANLYDEANCONGTY

* Chỉnh sửa database

**Ví dụ 1:** Chỉnh sửa kích cỡ của tập tin log file của QUANLYDUAN thành 10MB

ALTER DATABASE QUANLYDEANCONGTY

MODIFY FILE (NAME = ‘QUANLYDUAN\_log’, size = 10MB)

**Ví du 2:** Bổ sung thêm tập tin dữ liệu QUANLYDEAN\_data

ALTER DATABASE QUANLYDEANCONGTY

ADD FILE (NAME = QUANLYDUAN\_Data2,

FILENAME =‘C:\Program Files\Microsoft SQL Server\MSSQL\Data\ QUANLYDUAN\_Data2.MDF’, SIZE = 10MB, MAXSIZE = 20MB)

* **Xóa database**

**Cú pháp**

**DROP DATABASE DATABASE\_NAME [, …n]**

**Ví dụ:** Xóa database QUANLYDEANCONGTY

DROP DATABASE QUANLYDEANCONGTY

* Đổi tên database

**Cú pháp**

**SP\_RENAMEDB ‘OLD\_NAME’, ‘NEW\_NAME’**

**Ví dụ:** Đổi tên QUANLYDEANCONGTY thành QLDA

SP\_RENAMEDB ‘QUANLYDEANCONGTY’, ‘QLDA’

## Tạo và quản trị bảng.

Trong CSDL, để tạo và quản trị các bảng dữ liệu, người người dùng cần phải trả lời được các câu hỏi sau:

1. Bảng dùng để lưu trữ thông tin gì? 🡪 Đặt tên bảng mang tính gợi nhớ tương ứng với thông tin mà bảng lưu trữ. Ví dụ: bảng NHANVIEN lưu trữ tất cả các thông tin chi tiết của một nhân viên.
2. Với chức năng được xác định trong câu hỏi 1, xác định các thông tin cần được lưu trữ trong bảng đó 🡪 các thuộc tính cần thiết trong bảng. Ví dụ: bảng NHANVIEN lưu trữ thông tin: họ nhân viên, tên nhân viên, năm sinh, địa chỉ, mức lương, …
3. Kiểu dữ liệu cần thiết cho từng cột (thuộc tính) trong từng bảng (quan hệ) là gì? 🡪 xác định kiểu dữ liệu cho các cột trong từng bảng. Ví dụ như họ nhân viên thường được lưu trữ kiểu nvarchar, lương nhân viên được lưu trữ kiểu double hoặc money, …
4. Bảng sau khi đã tạo có thể thêm – xóa hay sửa đổi cấu trúc bảng hay không? Nếu có thì chỉnh sửa bằng cách nào? (tương ứng với cú pháp lệnh hoặc thao tác như thế nào?)

Để trả lời các câu hỏi trên, ta lần lượt đi xét các nội dung liên quan như sau:

### Kiểu dữ liệu.

#### 1.3.1.1 Các kiểu dữ liệu số nguyên:

* Bigint: là kiểu dữ liệu số nguyên (8 bytes) có giá trị từ -263 đến 263 – 1.
* Int: là kiểu dữ liệu số nguyên (4 bytes) có giá trị từ -231 đến 231 – 1.
* Smallint: là kiểu dữ liệu số nguyên (2 bytes) có giá trị từ -215 đến 215 – 1.
* Tinyint: là kiểu dữ liệu số nguyên (1 byte) có giá trị từ 0 đến 255.
* Bit: là kiểu dữ liệu số nguyên chỉ nhận giá trị 0 hoặc 1.
* Numeric, Decimal: là kiểu dữ liệu số nguyên có giá trị từ -1038 + 1 đến 1038 – 1.

#### 1.3.1.2 Các kiểu dữ liệu số thực:

* Float: là kiểu dữ liệu số thực có giá trị từ -1.79E +308 đến 1.79E + 308
* Real: là kiểu dữ liệu số thực có giá trị từ -3.40E + 38 đến 3.40E + 38.

#### 1.3.1.3 Các kiểu dữ liệu tiền tệ:

* Money: có giá trị từ -263 đến 263 – 1, với độ chính xác đến hai chữ số sau dấu thập phân.
* Smallmoney: có giá trị từ -214.748.3648 đến +214.748.3647, với độ chính xác đến hai chữ số sau dấu thập phân.

#### 1.3.1.4 Các kiểu dữ liệu ngày giờ:

* Datetime: là kiểu dữ liệu biểu diễn ngày giờ, có giá trị từ 1/1/1753 đến 31/12/9999 với độ chính xác là 3/100 giây hay 3.33 miliseconds.
* Smalldatetime: là kiểu dữ liệu biểu diễn ngày giờ, có giá trị từ 1/1/1900 đến 6/6/2079, với độ chính xác là một phút.

#### 1.3.1.5 Các kiểu dữ liệu chuỗi:

* Char: kiểu dữ liệu biểu diễn chuỗi ký tự có độ dài cố định (không theo Unicode), tối đa 8000 ký tự.
* Varchar: kiểu dữ liệu biểu diễn chuỗi ký tự có độ dài không cố định (không theo Unicode), tối đa 8000 ký tự. (Sự khác nhau giữa char và varchar thể hiện khi ta insert dữ liệu, nếu ta insert dữ liệu kiểu char, ta phải nhập đủ độ dài đã khai báo, nếu không SQL Server sẽ tự động thêm vào những khoảng trắng, ngược lại khi insert dữ liệu kiểu varchar, dù ta không insert đủ độ dài đã khai báo, SQL Server sẽ vẫn chỉ ghi nhận lại chuỗi dữ liệu ta đã nhập).
* Text: kiểu dữ liệu biểu diễn chuỗi ký tự có độ dài không cố định (không theo Unicode), tối đa 231 – 1 ký tự.
* NChar: kiểu dữ liệu biểu diễn chuỗi ký tự có độ dài cố định (Unicode), tối đa 8000 ký tự.
* NVarchar: kiểu dữ liệu biểu diễn chuỗi ký tự có độ dài không cố định (Unicode), tối đa 8000 ký tự. (Sự khác nhau giữa nchar và nvarchar giống như char và varchar).
* NText: kiểu dữ liệu biểu diễn chuỗi ký tự có độ dài không cố định (không theo Unicode), tối đa 231 – 1 ký tự.

#### 1.3.1.6 Các kiểu dữ liệu khác:

* Binary: kiểu dữ liệu số có độ dài cố định (tối đa 8000 bytes).
* Varbinary: kiểu dữ liệu số có độ dài không cố định (tối đa 8000 bytes).
* Image: kiểu dữ liệu số có độ dài thay đổi (tối đa 231 – 1 ).
* Cursor: kiểu dữ liệu con trỏ

### Tạo bảng.

Áp dụng database Quản lý đề án công ty như sau:

NHANVIEN (**MANV**, HONV, TENLOT, TENNV, NGSINH, DIACHI, LUONG, PHAI, MA\_NQL, PHOG)

**Cú pháp**

**CREATE TABLE TABLE\_NAME**

**(**

Column\_name1 datatype [Constraint],

Column\_name2 datatype [Constraint],

[, …n]

**)**

**Trong đó:**

* *TABLE\_NAME*: tên bảng đặt theo quy tắc định danh
* *Column\_name*: tên cột đặt theo quy tắc định danh
* *Constraint*: các ràng buộc toàn vẹn (sẽ được đề cập chi tiết trong chương 3)

**Ví dụ:** Tạo bảng nhân viên lưu trữ thông tin của các nhân viên bao gồm: mã số nhân viên, họ nhân viên, tên lót, tên nhân viên, ngày sinh, địa chỉ

CREATE TABLE NHANVIEN

(

MANV VARCHAR(10) PRIMARY KEY,

HONV NVARCHAR(10) NOT NULL,

TENLOT NVARCHAR(20),

TENNV NVARCHAR(20) NOT NULL,

NGAYSINH DATETIME,

DIACHI NVARCHAR(100) NOT NULL

)

### Quản trị bảng

* Thêm một cột mới trong bảng

**Cú pháp**

**ALTER TABLE** TABLE\_NAME

**ADD** COLUMN\_NAME DATATYPE

**Ví dụ:** Hãy thêm cột phái vào bảng nhân viên

ALTER TABLE NHANVIEN

ADD PHAI NVARCHAR (3)

* Xóa cột bên trong bảng

**Cú pháp**

**ALTER TABLE** TABLE\_NAME

**DROP COLUMN** COLUMN\_NAME [, …n]

**Ví dụ:** Hãy xóa cột phái trong bảng nhân viên

ALTER TABLE NHANVIEN

DROP COLUMN PHAI

* Sửa đổi kiểu dữ liệu của cột

**Cú pháp**

**ALTER TABLE** TABLE\_NAME

**ALTER COLUMN** COLUMN\_NAME DATATYPE

**Ví dụ:** Thay đổi dữ liệu của cột địa chỉ thành kiểu dữ liệu nvarchar có độ rộng tối đa 150 ký tự

ALTER TABLE NHANVIEN

ALTER COLUMN DIACHI NVARCHAR (150)

* Đổi tên cột

**Cú pháp**

**EXEC SP\_RENAME** ‘TABLE\_NAME.TABLE\_COLUMN’, ‘NEW\_NAME’, ‘COLUMN’

**Ví dụ:** đổi tên cột DIACHI trong bảng nhân viên thành DCHI

EXEC SP\_RENAME ‘NHANVIEN.DIACHI’, ‘DCHI’, ‘COLUMN’

* **Đổi tên bảng**

**Cú pháp**

**EXEC SP\_RENAME** ‘OLD\_ NAME’, ‘NEW\_NAME’

**Ví dụ:** Đổi tên bảng NHANVIEN thành NVIEN

EXEC SP\_RENAME ‘NHANVIEN’, ‘NVIEN’

* **Xóa bảng**

**Cú pháp**

**DROP TABLE** TABLE\_NAME

**Ví dụ:** Hãy xóa bảng nhân viên trong database

DROP TABLE NHANVIEN

# BÀI TẬP CHƯƠNG 1

## CÂU HỎI

**Câu hỏi 1:** Trình bày tóm tắt các thành phần cơ bản của hệ quản trị cơ sở dữ liệu SQL Server ?

**Câu hỏi 2:** Trình bày các đối tượng của một cơ sở dữ liệu quan hệ trong SQL Server.

**Câu hỏi 3:** Phân biệt các loại cơ sở dữ liệu trong SQL Server.

**Câu hỏi 4:** Trình bày cú pháp các câu lệnh tạo và quản trị một Database trong SQL Server.

**Câu hỏi 5:** Trình bày cú pháp các câu lệnh tạo và quản trị một Table trong SQL Server.

## BÀI TẬP

**Bài tập 1:** Hãy xây dựng cơ sở dữ liệu cho lược đồ cơ sở dữ liệu ‘QUẢN LÝ ĐỀ ÁN CÔNG TY’ được mô tả như sau:

**NHANVIEN** (MANV, HONV, TENLOT, TENNV, PHAI, DIACHI, LUONG, NGAYSINH, PHG)

***Mô tả*:** Mỗi nhân viên có một mã số duy nhất để phân biệt với các nhân viên khác. Thông tin mô tả nhân viên bao gồm: họ, tên lót, tên nhân viên, phái, địa chỉ, mức lương, ngày tháng năm sinh, và mã phòng mà nhân viên trực thuộc.

**PHONGBAN** (MAPHG, TENPHG, DIADIEM\_PHG)

***Mô tả*:** Mỗi phòng trong công ty có một mã số duy nhất để phân biệt với các phòng khác. Mỗi phòng được mô tả bởi tên phòng (là duy nhất), địa điểm phòng.

**DEAN** (MADA, TENDA, PHONG, DIADIEM\_DA)

***Mô tả*:** Mỗi đề án có một mã số duy nhất để phân biệt với các đề án khác. Mỗi có tên đề án, mã phòng ban chủ trì đề án và địa điểm triển khai đề án.

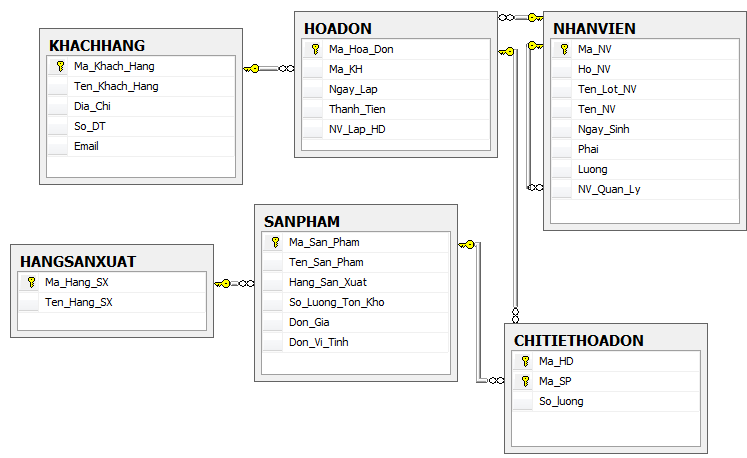
**PHANCONG** (MA\_NVIEN, SODA, THOIGIAN)

***Mô tả*:** Lưu trữ thông tin chi tiết của việc phân công các nhân viên (MA\_NVIEN) sẽ tham gia đề án nào (SODA) với thời gian tham gia đề án tương ứng.

**Yêu cầu:**

1. Hãy xác định khóa chính, khóa ngoại và các điều kiện ràng buộc cần thiết khác cho lược đồ cơ sở dữ liệu mô tả như trên.
2. Hãy tạo cơ sở dữ liệu trong HQT CSDL MS SQL SERVER (phiên bản 2008 trở lên) với bằng các câu lệnh định nghĩa (CREATE DATABASE, CREATE TABLE, ALTER TABLE, …) và kiểu dữ liệu thích hợp cho từng thuộc tính.
3. Hãy nhập dữ liệu cho các bảng (mỗi bảng ít nhất 10 dòng dữ liệu) (INSERT, UPDATE, …)

**Bài tập 2: Cho lược đồ cơ sở dữ liệu sau:**



**Với nội dung miêu tả như sau:**

* **SANPHAM:** Mô tả thông tin chi tiết của một sản phẩm
  + **Ma\_San\_Pham:** Mã số của sản phẩm
  + **Ten\_San\_Pham:** Tên của sản phẩm
  + **Hang\_San\_Xuat:** Mã sốhãng sản xuất của sản phẩm
  + **So\_Luong\_Ton\_Kho:** Số lượng tồn kho của sản phẩm
  + **Don\_Gia:** Đơn giá của sản phẩm
  + **Don\_Vi\_Tinh:** Đơn vị tính của sản phẩm
* **HANGSANXUAT:** Mô tả thông tin của hãng sản xuất
  + **Ma\_Hang\_SX:** Mã số của hãng sản xuất
  + **Ten\_Hang\_SX:** Tên của hãng sản xuất
* **HOADON:** Mô tả thông tin của một hóa đơn bán hàng
  + **Ma\_Hoa\_Don:** Mã số của hóa đơn
  + **Ma\_KH:** Mã số của khách hàng nhận hóa đơn
  + **Ngay\_Lap:** Ngày lập hóa đơn
  + **Thanh\_Tien:** Thành tiền của hóa đơn
  + **NV\_Lap\_HD:** Mã số của nhân viên lập hóa đơn
* **CHITIETHOADON:** Mô tả thông tin chi tiết của một hóa đơn
  + **Ma\_HD:** Mã số của hóa đơn
  + **Ma\_SP:** Mã số của sản phẩm có trong hóa đơn
  + **So\_Luong:** Số lượng của sản phẩm có trong hóa đơn
* **NHANVIEN:** Mô tả thông tin của một nhân viên
  + **Ma\_NV:** Mã số của nhân viên
  + **Ho\_NV:** Họ của nhân viên
  + **Ten\_Lot\_NV:** Tên lót của nhân viên
  + **Ten\_NV:** Tên của nhân viên
  + **Ngay\_Sinh:** Ngày sinh của nhân viên
  + **Phai:** Phái của nhân viên
  + **Luong:** Lương của nhân viên
  + **NV\_Quan\_Ly:** Mã số của nhân viên quản lý

**Hãy thực hiện các công việc sau đây:**

1. Viết câu lệnh tạo cơ sở dữ liệu QLBH với hai tập tin quản lý là “QLBH.mdf” và “QLBH\_log.ldf” được lưu trong thư mục “D:\HQTCSDL\BT\Chuong1”. Biết rằng kích thước khởi tạo của file QLBH.mdf là 500MB, kích thước tối đa là không giới hạn, kích thước tăng trưởng là 100MB, file QLBH\_log.ldf có kích thước khởi tạo là 800MB, kích thước tối đa là không giới hạn, kích thước tăng trưởng là 10%
2. Viết câu lệnh chỉnh sửa cấu trúc cơ sở dữ liệu đã tạo ở câu 1 với nội dung như sau: “Chỉnh sửa kích thước khởi tạo của file QLBH.mdf thành 250MB, kích thước tối đa là 800MB. Thêm một tập tin mới “QLBH\_Data2.ndf” của cơ sở dữ liệu QLBH vào cùng thư mục “D:\HQTCSDL\BT\Chuong1” với kích thước khởi tạo là 200MB, kích thước tối đa 800MB, kích thước tăng trưởng 10%”
3. Viết câu lệnh tiến hành xóa cơ sỡ dữ liệu QLBH đã tạo ở hai câu trên.
4. Viết các câu lệnh tạo cấu trúc các bảng như trong lược đồ quan hệ, yêu cầu tạo đầy đủ khóa chính và khóa ngoại cho các bảng

# CHƯƠNG 2 SAO LƯU (BACKUP) - PHỤC HỒI (RESTORE) VÀ CHUYỂN ĐỔI DỮ LIỆU (IMPORT/EXPORT)

# MỤC ĐÍCH VÀ YÊU CẦU

**Mục đích**:

Chương 2 nhằm cung cấp cho sinh viên về những kiến thức cơ bản về:

* Phương pháp sao lưu và phục hồi cơ sở dữ liệu;
* Phương pháp chuyển đổi dữ liệu từ SQL Server sang các phần mềm khác và ngược lại.

**Yêu cầu**:

Sau khi học xong chương 2 sinh viên có khả năng:

* Biết cách thực hiện được thao tác sao lưu và phục hồi cơ sở dữ liệu;
* Biết cách thực hiện được thao tác chuyển đổi dữ liệu từ SQL Server sang các phần mềm khác và ngược lại;
* Hiểu rõ cơ chế hoạt động của SQL Server 2008 khi người dùng thực thi sao lưu – phục hồi hay chuyển đổi kiểu dữ liệu.

# NỘI DUNG

## 2.1. Tổng quan về sao lưu và phục hồi dữ liệu

### 2.1.1. Mục đích của việc sao lưu và phục hồi dữ liệu

Sao lưu và khôi phục dữ liệu là vấn đề rất quan trọng khi quản trị CSDL. Hoạt động này đảm bảo cho dữ liệu khi có sự cố xảy ra, ta có thể khôi phục lại trạng thái của CSDL trước khi xảy ra sự cố. Công việc này phải dựa vào cơ chế sao lưu (Backup) và phục hồi (Restore) của các HQT CSDL (hệ quản trị cơ sở dữ liệu), và để thực hiện tốt công việc này thì người quản trị CSDL phải có kịch bản sao lưu và khôi phục CSDL cho hệ thống một cách cẩn thận và chi tiết.

### 2.1.2. Các cơ chế sao lưu dữ liệu

SQL Server cung cấp ba cơ chế sao lưu như sau:

* **Full Backup**: Backup toàn bộ dữ liệu tại thời điểm thực hiện.
* **Differential Backup**: Backup các dữ liệu mới được cập nhật kể từ lần full backup trước đó.
* **Transaction Log Backup**: Backup các *log record* hiện có trong log file. Cơ chế sao lưu này sẽ sao lưu các *hành động* (các thao tác xảy ra đối với database) chứ không sao lưu dữ liệu. Đồng thời cơ chế này cũng tiến hành ghi đè lên các *log record* đã được back up, còn gọi là thao *tác Log Truncation*, nhờ vậy vùng nhớ của log file được giải phóng để ghi nhận các *log record* mới. Transaction log backup chỉ sao lưu các log record kể từ lần transaction log backup trước đó. Khi thấy log file tăng quá lớn, có nhiều khả năng là ta chưa từng backup transaction log bao giờ.

### 2.1.3. Kịch bản sao lưu dữ liệu

Nguyên tắc chung để giảm thiểu mất mát dữ liệu khi có sự cố là tăng tần suất backup. Tuy nhiên với một database có dung lượng lớn và được cập nhật liên tục, việc thực hiện full backup với tần suất cao là không khả thi, vì cơ chế này dùng rất nhiều tài nguyên (CPU, I/O). Nhờ có differential backup và transaction log backup, ta có thể tạo lập các phương án kịch bản sao lưu thích hợp, đảm bảo dữ liệu được backup thường xuyên hơn mà không chiếm nhiều tài nguyên của hệ thống.

*Ví dụ, một kịch bản sao lưu và khôi phục dữ liệu tham khảo như sau:*

* **Full backup:** Một lần mỗi ngày vào 2h sáng.
* **Differential backup:** vào các thời điểm 6h, 10h, 14h, 18h, 22h (5 lần/ngày).
* **Transaction log backup:** 15 phút một lần vào các thời điểm 5′, 20′, 35′, và 50′ của mỗi giờ (4 lần/ giờ).

***Lưu ý***  *Differential backup* luôn sao lưu các dữ liệu đã thay đổi kể từ lần full backup trước (trong ví dụ trên là các dữ liệu đã  thay đổi kể từ 2h), mà không phải từ lần differential backup trước đó. Vì thế bản backup lúc 10h sẽ bao gồm các dữ liệu đã được sao lưu trong bản backup lúc 6h, bản backup lúc 14h gồm các dữ liệu đã có trong bản 10h…

*Giả sử Database bị hỏng vào thời điểm 10h55′, ta cần khôi phục lại database theo trình tự sau:*

**Bước 1***.* Khôi phục từ bản full backup gần với thời điểm có sự cố nhất (bản full backup lúc 2h).

**Bước 2*.*** Khôi phục từ bản differential backup gần với thời điểm có sự cố nhất (bản lúc 10h).

**Bước 3*.*** Khôi phục tất cả các transaction log backup kể từ sau lần diferential backup gần nhất, lần lượt theo trình tự thời gian. Đó là các bản tại các thời điểm 10h5′, 10h20′, 10h35′, và 10h50′.

Bước 1 và 2 sẽ đưa database trở lại trạng thái giống như lúc 10h. Ở bước 3, với mỗi lần khôi phục transaction log thì các thao tác chứa trong đó được đem ra thực hiện lại trên database (gọi là log forwarding) và do đó đưa database về trạng thái gần hơn thời điểm xảy ra sự cố. Như vậy sau khi hoàn tất khôi phục bốn bản transaction log backup thì database sẽ ở vào trạng thái giống như lúc 10h50′. Tuy nhiên các thay đổi diễn ra trong 5 phút sau đó (từ 10h50′ đến 10h55′) đã bị mất.

## 2.2. Sao lưu dữ liệu (Backup)

Cú pháp

**BACKUP DATABASE DATABASE\_NAME**

TO <thiết bị lưu>

[WITH {DIFFERENTIAL | <tham số>}]

Trong đó:

* **DIFFERENTIAL**: Sao lưu một phần, mặc nhiên là sao lưu toàn phần.

<thiết bị lưu> := { DISK | TAPE } = ‘tên\_thiết\_bị’

* **Tham số** sao lưu:
* DESCRIPTION = *ghichu* (ghi chú bản lưu)
* NAME = *ten* (ghi tên bản lưu)
* PASSWORD = *matkhau* (ghi mật khẩu bản lưu)
* RETAINDAYS = *songay* (ghi số ngày hết hạn của bản lưu tính từ thời điểm sao lưu)
* EXPIREDATE = *ngay* (hạn cuối của bảng sao lưu)
* STOP\_ON\_ERROR | CONTINUE\_AFTER\_ERROR (ngừng khi gặp lỗi và ngược lại)
* STATS = *số* (phần trăm thể hiện trong quá trình sao lưu)

Ví dụ minh họa

**Ví dụ 1:** Hãy viết lệnh thực hiện công việc sao lưu toàn phần cho CSDL tên là ‘QUANLYDEANCONGTY’

BACKUP DATABASE QUANLYDEANCONGTY

TO DISK = ‘D: \QUANLYDEANCONGTY.BAK’

WITH NAME = ‘QUANLYDEANCONGTY backup’,

STATS = 10

**Ví dụ 2:** Hãy viết lệnh thực hiện công việc sao lưu một phần cho CSDL tên là ‘QUANLYDEANCONGTY’

BACKUP DATABASE QUANLYDEANCONGTY

TO DISK = ‘D: \QUANLYDEANCONGTY.BAK’

WITH DIFFERENTIAL, NAME = ‘QUANLYDEANCONGTY backup’,

STATS = 10

## 2.3. Phục hồi

**Cú pháp**

**RESTORE DATABASE NEW\_DATABASE\_NAME**

**FROM**  <thiết bị lưu> [WITH {<tham số>}]

Trong đó:

* **Lưu ý**: Quá trình sao lưu 1 phần chỉ được phục hồi liền sau quá trình phục hồi toàn phần (với tham số norecovery)
* **Tham số** phục hồi
* STOP\_ON\_ERROR|CONTINUE\_AFTER\_ERROR  **(ngừng khi gặp lỗi và ngược lại)**
* FILE = { tên | số thứ tự } (Chỉ bản sao lưu được phục hồi trong tập tin sao lưu)
* PASSWORD = matkhau (ghi mật khẩu bảng lưu)
* RECOVERY | NORECOVERY (trạng thái của datbase sau khi thực hiện xong lệnh phục hồi.
  + Với trạng thái **Recovery**, database sẽ chấm dứt quá trình phục hồi, có thể được truy cập bình thường.
  + Với trạng thái **NoRecovery**, quá trình phục hồi chưa chấm dứt, các lệnh phục hồi có thể được tiếp thục thực hiện trên database này, và database này sẽ không thể truy cập cho đến khi có một lệnh quá trình phục hồi **Recovery** được thực hiện trên nó.

**Ví dụ minh họa**

**Ví dụ 1**: Phục hồi từ tập tin sao lưu ‘QUANLYDEANCONGTY.bak’ thành database tên là ‘QUANLYDUAN’. (chưa chấm dứt phục hồi)

RESTORE DATABASE QUANLYDEANCONGTY

FROM DISK = ‘D: \QUANLYDEANCONGTY.BAK’

WITH FILE = 1, NORECOVERY, STATS=10;

**Ví dụ 2**: Phục hồi từ tập tin ‘QUANLYDEANCONGTY.bak’ thành database tên là ‘QUANLYDUAN’. (chấm dứt phục hồi)

RESTORE DATABASE QUANLYDEANCONGTY

FROM DISK = ‘D: \QUANLYDEANCONGTY.BAK’

WITH FILE = 1, RECOVERY, STATS=10;

**Ví dụ 3**: Phục hồi từ tập tin ‘QUANLYDEANCONGTY.bak’ thành database tên là ‘QUANLYDUAN2’ và đường dẫn 2 file .mdf và .ldf lưu qua vị trí khác.

RESTORE DATABASE QUANLYDUAN2

FROM DISK = ‘D:\ QUANLYDEANCONGTY.bak’

WITH FILE = 1, RECOVERY, STATS=10,

MOVE ‘QUANLYDEANCONGTY \_data' TO ‘D:\ QUANLYDEANCONGTY.mdf',

MOVE ' QUANLYDEANCONGTY \_log' TO ‘D:\ QUANLYDEANCONGTY.ldf';

## 2.4. Attach và detach

Dữ liệu được lưu trữ thành tập tin có đuôi là .mdf và các vết lưu quá trình làm việc, giao tác được lưu với tập tin có đuôi là .ldf . Để sao lưu dữ liệu, ta có thể thực hiên một cách khác, đó chính là sao chép và giữ lại các tập tin .mdf và .ldf quan trọng của một database. Trên thực tế ta có thể chỉ cần tập tin .mdf là có thể phục hồi lại database, khi đó ta sẽ dùng một số kỹ thuật nhất định để tạo ra một file log .ldf tương ứng. Lệnh sp\_detach\_db giúp ta gỡ bỏ một database ra khỏi hệ thống, đồng thời lệnh sp\_attach\_db giúp ta gắn kết một database vào hệ thống thông qua file .mdf.

**Cú pháp**

**SP\_DETACH\_DB** ‘DATABASE\_NAME’

**SP\_ATTACH\_DB** ‘DATABASE\_NAME’

, ‘PATH FILENAME.MDF’, ‘PATH FILENAME.LDF’

**Ví dụ:**

**SP\_DETACH\_DB** ‘QUANLYDEANCONGTY’

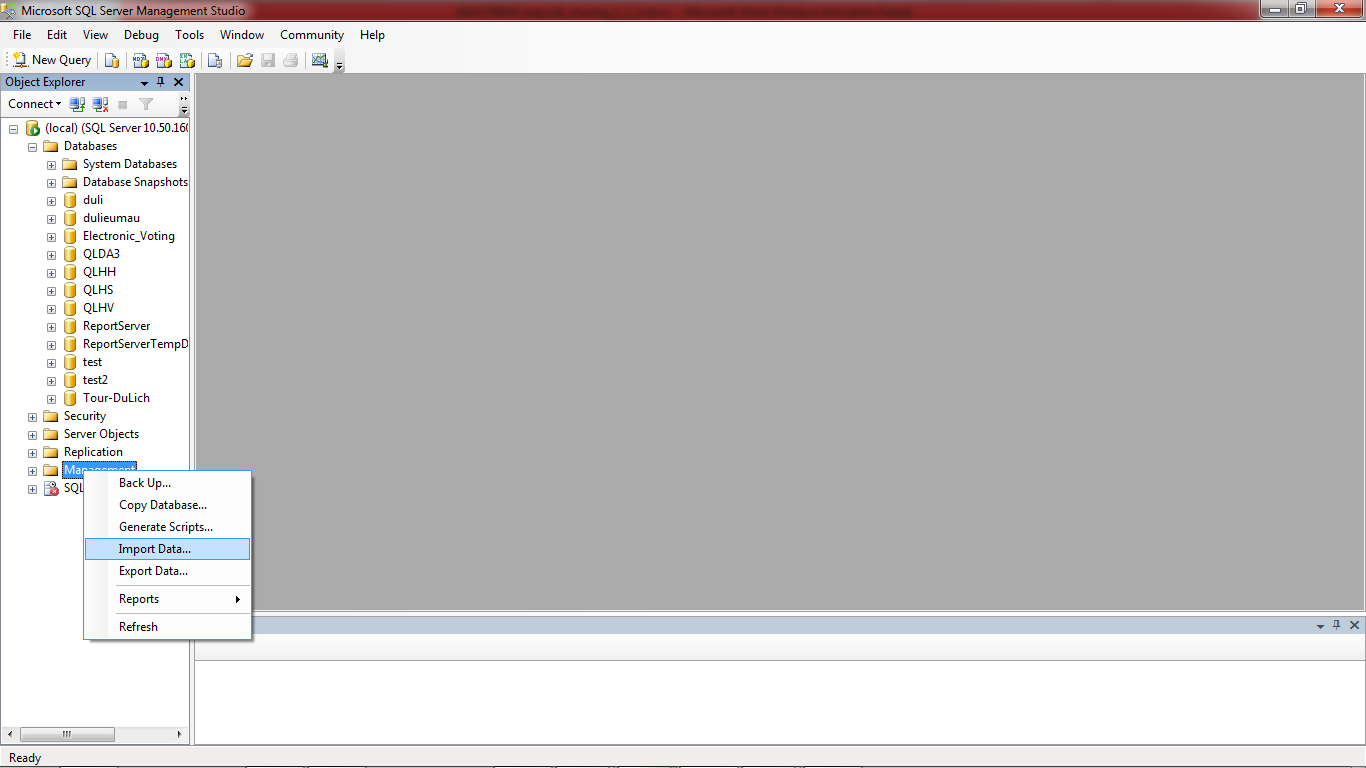
**SP\_ATTACH\_DB** ‘QUANLYDEANCONGTY’

, ‘D:\ QUANLYDEANCONGTY.MDF’

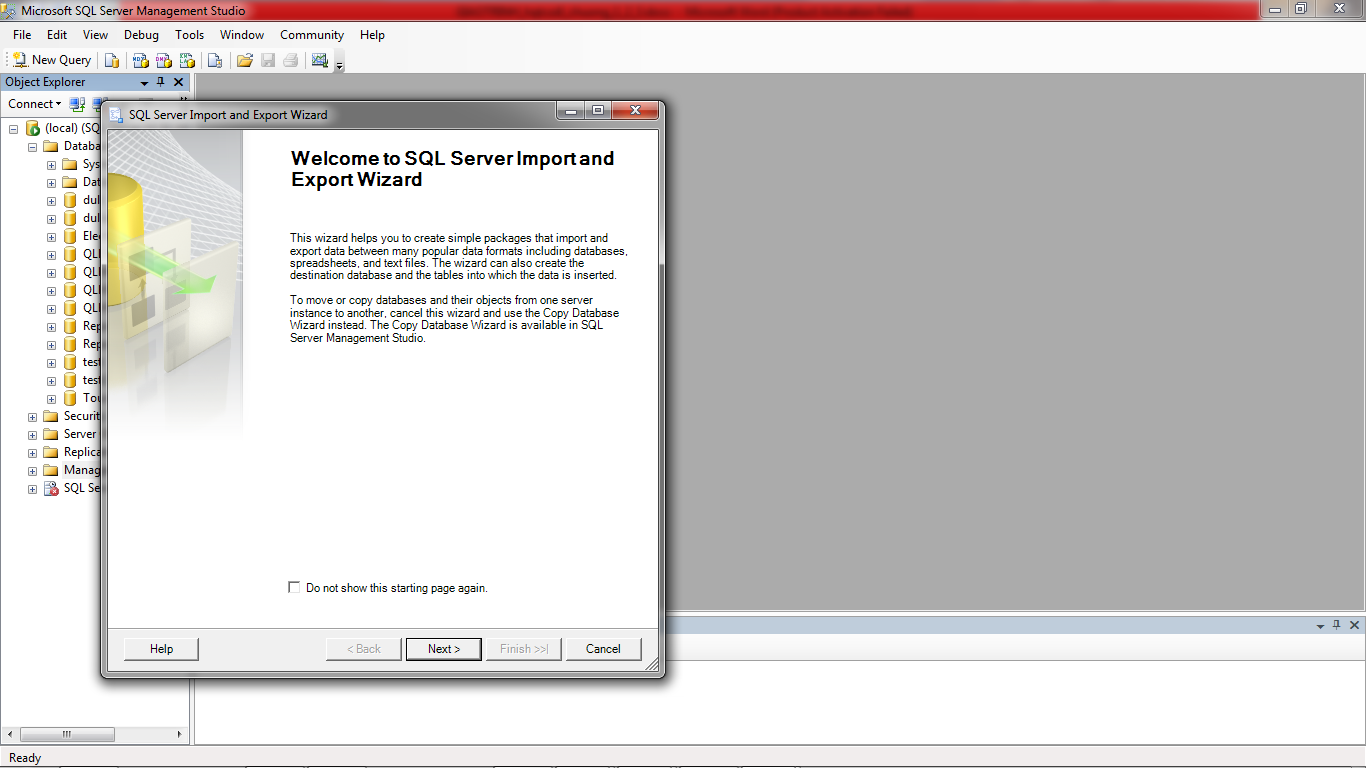
, ‘D:\ QUANLYDEANCONGTY.LDF’

## 2.5. Import/Export data – Nhập hoặc trích xuất dữ liệu

**Bước 1**: Trong cửa sổ Object Explorer chọn **Import and Export Data** xuất hiện hộp thoại **SQL Server Import and Export Wizard**🡪 nhấn **Next;**



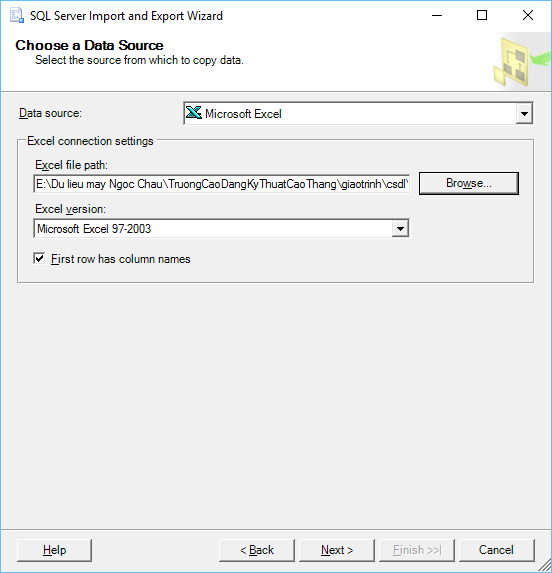
Hình 2. 1. Thao tác import dữ liệu



Hình 2. 2. Hộp thoại SQL Server Import and Export Wizard

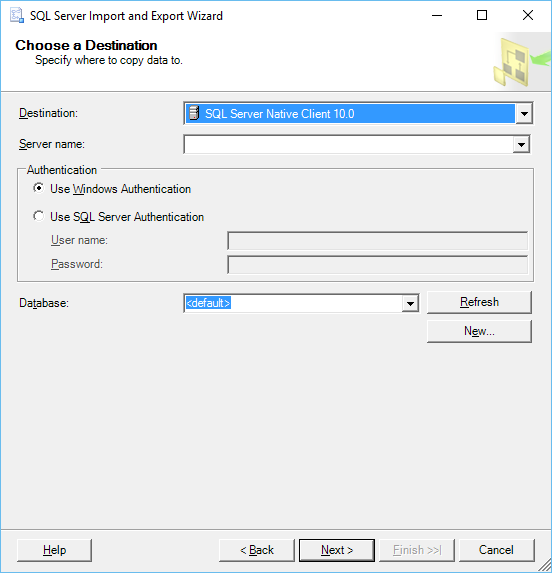
Import hoặc Export Wizard

**Bước 2:** Chọn nguồn dữ liệu nếu từ SQL ra ngoài các dạng khác thì chọn nguồn là SQL còn ngược lại thì chọn nguồn Excel, Access … chẳng hạn như ví dụ hình dưới đây là Excel. Sau khi chọn file nguồn 🡪 nhấn **Next**



Hình 2. 3. Màn hình chọn dữ liệu nguồn

**Bước 3:**. Tại cửa sổ chọn Server đích 🡪 nhấn Next (2 lần)

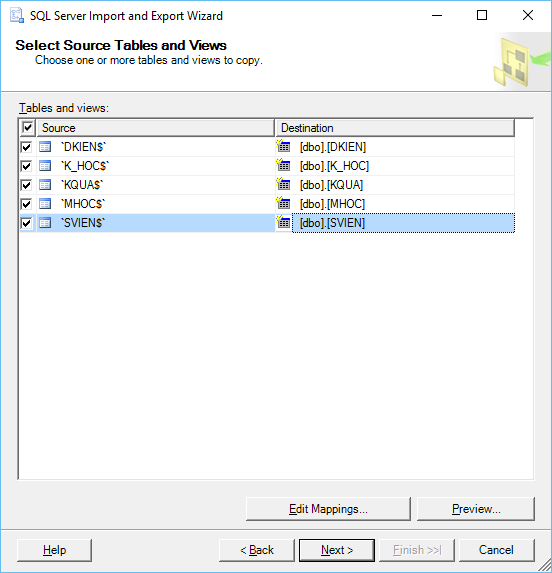


Tạo Database mới

Chọn tên server

Hình 2. 4. Hộp thoại chọn dữ liệu đích

**Bước 4:** Check chọn tất cả các bảng trong Source và bỏ dấu $ bên Destination. Sau đó nhấn Next (2 lần).



Hình 2. 5. Giao diện chọn bảng cần Import

**Bước 5**: Nhấn Finish để hoàn tất quá trình

# BÀI TẬP CHƯƠNG 2

## CÂU HỎI:

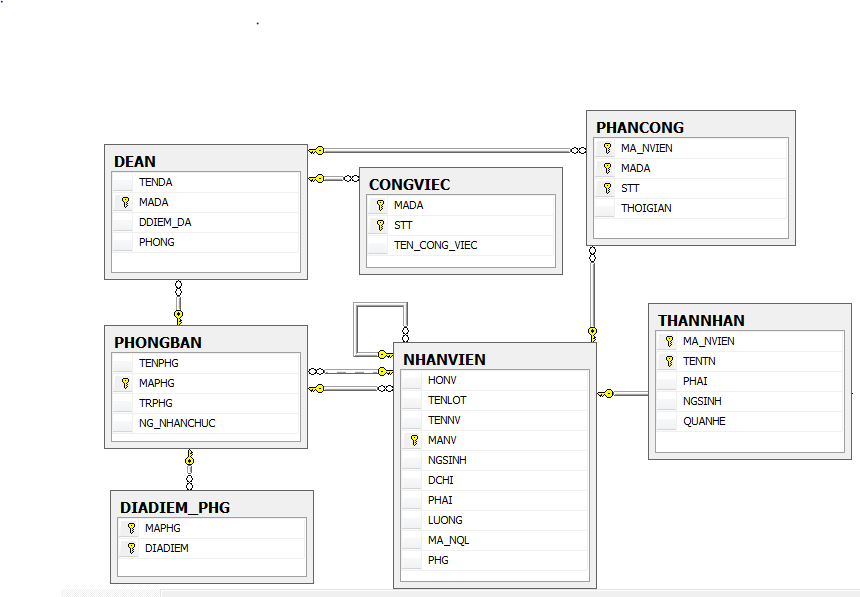
**Câu hỏi 1:** Định nghĩa sao lưu dữ liệu? Vì sao ta cần tiến hành sao lưu dữ liệu.

**Câu hỏi 2:** Phân biệt hai cơ chế sao lưu dữ liệu: sao lưu toàn phần và sao lưu một phần

**Câu hỏi 3:** Trình bày vai trò của kịch bản sao lưu trong quá trình sao lưu và phục hồi dữ liệu

## BÀI TẬP:

**Sử dụng lược đồ sau để thực hiện các bài tập 1, bài tập 2 (mô tả chi tiết về lược đồ quan hệ xem lại ở bài tập chương 1)**



**Bài tập 1: Cho kịch bản sao lưu theo các bước như sau:**

***Bước 1:*** Với cơ sở dữ liệu QLDA đã được tạo đầy đủ các bảng (chưa có dữ liệu), tiến hành sao lưu toàn phần với tập tin sao lưu được đặt tên là QLDA\_BACKUP\_FULL1.bak và được đặt trong thư mục “D:\HQTCSDL\BT\Chuong2”

***Bước 2:*** Trong bảng THANNHAN, bổ sung thuộc tính DIACHI, sau đó tiến hành sao lưu một phần với tập tin sao lưu được đặt tên là QLDA\_BACKUP\_DIFF1\_1.bak và được đặt trong thư mục “D:\HQTCSDL\BT\Chuong2”

***Bước 3:*** Trong bảng NHANVIEN, bổ sung thuộc tính thuộc tính PHUCAP và xóa thuộc tính LUONG, sau đó tiến hành sao lưu một phần với tập tin sao lưu được đặt tên là QLDA\_BACKUP\_DIFF1\_2.bak và được đặt trong thư mục “D:\HQTCSDL\BT\Chuong2”

Hãy thực hiện các yêu cầu sau:

1. Viết các câu lệnh và thực hiện các thao tác theo đúng kịch bản sao lưu trên
2. Viết câu lệnh phục hồi cơ sở dữ liệu về trạng thái ở bước 1
3. VIết câu lệnh phục hồi cơ sở dữ liệu về trạng thái ở bước 2 và đưa ra nhận xét
4. Viết câu lệnh phục hồi cơ sở dữ liệu về trạng thái ở bước 3 và đưa ra nhận xét

**Bài tập 2: Cho kịch bản sao lưu theo các bước như sau:**

Bước 1: Với cơ sở dữ liệu QLDA đã được tạo đầy đủ các bảng (chưa có dữ liệu), tiến hành sao lưu toàn phần với tập tin sao lưu được đặt tên là QLDA\_BACKUP\_FULL2.bak và được đặt trong thư mục “D:\HQTCSDL\BT\Chuong2”

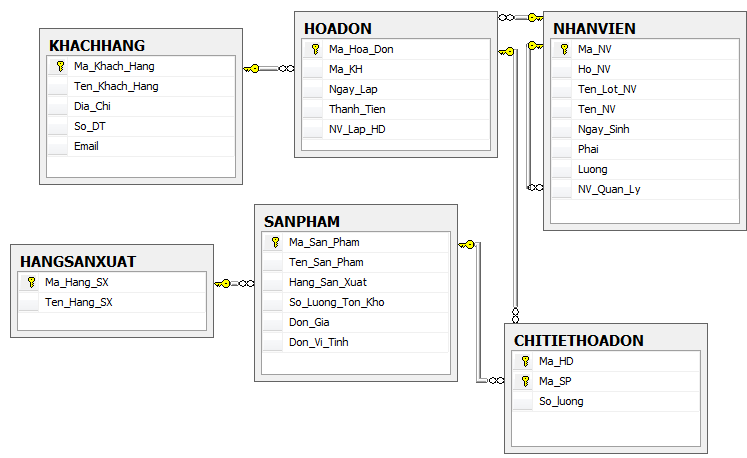
Bước 2: Viết các câu lệnh thêm ba bộ dữ liệu bất kỳ vào bảng PHONGBAN, sau đó tiến hành sao lưu một phần với tập tin sao lưu được đặt tên là QLDA\_BACKUP\_DIFF2\_1.bak và được đặt trong thư mục “D:\HQTCSDL\BT\Chuong2”

Bước 3: Viết các câu lệnh thêm ba bộ dữ liệu bất kỳ vào bảng DEAN và xóa tất cả dữ liệu ở bảng PHONGBAN, sau đó tiến hành sao lưu một phần với tập tin sao lưu được đặt tên là QLDA\_BACKUP\_DIFF2\_2.bak và được đặt trong thư mục “D:\HQTCSDL\BT\Chuong2”

Hãy thực hiện các yêu cầu sau:

1. Viết các câu lệnh và thực hiện các thao tác theo đúng kịch bản sao lưu trên
2. Viết câu lệnh phục hồi cơ sở dữ liệu về trạng thái ở bước 1
3. VIết câu lệnh phục hồi cơ sở dữ liệu về trạng thái ở bước 2 và đưa ra nhận xét
4. Viết câu lệnh phục hồi cơ sở dữ liệu về trạng thái ở bước 3 và đưa ra nhận xét

**Sử dụng lược đồ sau để thực hiện các bài tập 3, bài tập 4 (mô tả chi tiết về lược đồ quan hệ xem lại ở bài tập chương 1)**



**Bài tập 3: Cho kịch bản sao lưu theo các bước như sau:**

***Bước 1:*** Với cơ sở dữ liệu QLBH đã được tạo đầy đủ các bảng (chưa có dữ liệu), tiến hành sao lưu toàn phần với tập tin sao lưu được đặt tên là QLBH\_BACKUP\_FULL1.bak và được đặt trong thư mục “D:\HQTCSDL\BT\Chuong2”

***Bước 2:*** Trong bảng NHANVIEN, bổ sung thuộc tính DIACHI, sau đó tiến hành sao lưu một phần với tập tin sao lưu được đặt tên là QLBH\_BACKUP\_DIFF1\_1.bak và được đặt trong thư mục “D:\HQTCSDL\BT\Chuong2”

***Bước 3:*** Tạo thêm một bảng mới để lưu thông tin tài khoản của nhân viên: TAIKHOAN(Username, Password), đồng thời bổ sung thuộc tính TENTK trong bảng NHANVIEN, sau đó tiến hành sao lưu một phần với tập tin sao lưu được đặt tên là QLBH\_BACKUP\_DIFF1\_2.bak và được đặt trong thư mục “D:\HQTCSDL\BT\Chuong2”

Hãy thực hiện các yêu cầu sau:

1. Viết các câu lệnh và thực hiện các thao tác theo đúng kịch bản sao lưu trên
2. Viết câu lệnh phục hồi cơ sở dữ liệu về trạng thái ở bước 1
3. VIết câu lệnh phục hồi cơ sở dữ liệu về trạng thái ở bước 2 và đưa ra nhận xét
4. Viết câu lệnh phục hồi cơ sở dữ liệu về trạng thái ở bước 3 và đưa ra nhận xét

**Bài tập 4: Cho kịch bản sao lưu được thực hiện tiếp theo kịch bản sao lưu của bài tập 3:**

***Bước 4:*** Với cơ sở dữ liệu QLBH đã trải qua các bước thay đổi và sao lưu ở bài tập 3, tiến hành sao lưu toàn phần với tập tin sao lưu được đặt tên là QLBH\_BACKUP\_FULL2.bak và được đặt trong thư mục “D:\HQTCSDL\BT\Chuong2”

***Bước 5:*** Thêm ba bộ dữ liệu bất kỳ trong bảng TAIKHOAN, sau đó tiến hành sao lưu một phần với tập tin sao lưu được đặt tên là QLBH\_BACKUP\_DIFF2\_1.bak và được đặt trong thư mục “D:\HQTCSDL\BT\Chuong2”

***Bước 6:*** Thêm ba bộ dữ liệu bất kỳ trong bảng NHANVIEN, sau đó tiến hành sao lưu một phần với tập tin sao lưu được đặt tên là QLBH\_BACKUP\_DIFF2\_2.bak và được đặt trong thư mục “D:\HQTCSDL\BT\Chuong2”

Hãy thực hiện các yêu cầu sau:

1. Viết các câu lệnh và thực hiện các thao tác theo đúng kịch bản sao lưu trên
2. Viết câu lệnh phục hồi cơ sở dữ liệu về trạng thái ở bước 4
3. VIết câu lệnh phục hồi cơ sở dữ liệu về trạng thái ở bước 5 và đưa ra nhận xét
4. Viết câu lệnh phục hồi cơ sở dữ liệu về trạng thái ở bước 6 và đưa ra nhận xét

Thực hiện export dữ liệu ở câu 8 ra file Excel và lưu thành tập tin “QLBH.xls” trong thư mục “D:\HQTCSDL\BT\Chuong2”

**Bài tập 5:** Hãy thực hiện sao chép cơ sở dữ liệu ‘QUẢN LÝ ĐỀ ÁN CÔNG TY’ và ‘QUẢN LÝ BÁN HÀNG’ bằng phương pháp Attach và Detach dữ liệu.

**Bài tập 6:** Hãy trích xuất dữ liệu (EXPORT) từ cơ sở dữ liệu ‘QUẢN LÝ ĐỀ ÁN CÔNG TY’ và ‘QUẢN LÝ BÁN HÀNG’ sang HQT CSDL Access, sau đó nhập dữ liệu (IMPORT) từ Access sang SQL SERVER.

# CHƯƠNG 3 THIẾT LẬP RÀNG BUỘC TOÀN VẸN (CONSTRAINTS)

# MỤC ĐÍCH VÀ YÊU CẦU

**Mục đích**:

Chương 3 nhằm cung cấp cho sinh viên về những kiến thức cơ bản về ràng buộc toàn vẹn dữ liệu.Ý nghĩa của các loại ràng buộc và cách cài đặt các loại ràng buộc.

**Yêu cầu:**

Sau khi học xong chương 3 sinh viên có khả năng:

* Biết và hiểu được ý nghĩa của ràng buộc dữ liệu.
* Biết phân biệt được các loại ràng buộc.
* Biết cách cài đặt các loại ràng buộc.

# NỘI DUNG

## 3.1. Giới thiệu

### 3.1.1. Định nghĩa ràng buộc toàn vẹn (RBTV)

Trong một CSDL, luôn luôn tồn tại rất nhiều mối liên hệ ảnh hưởng qua lại lẫn nhau giữa các thuộc tính của một quan hệ, giữa các bộ giá trị trong một quan hệ và giữa các thuộc tính của các bộ giá trị trong các quan hệ với nhau. Các mối quan hệ phụ thuộc lẫn nhau này chính là những điều kiện bất biến mà tất cả các bộ của những quan hệ có liên quan trong cơ sở dữ liệu đều phải thỏa mãn ở bất kỳ thời điểm nào. Ràng buộc toàn vẹn còn được gọi là các quy tắc quản lý (Rules) được áp đặt lên trên các đối tượng của thế giới thực. Như vậy, ràng buộc toàn vẹn(RBTV) là một điều kiện bất biến không được vi phạm trong một CSDL. RBTV được xác định khi định nghĩa lược đồ CSDL, đồng thời khi các quan hệ có sự thay đổi, cập nhật dữ liệu thông qua các thao tác insert, delete, update,v.v..., các RBTV sẽ được kiểm tra

**Ví dụ:** Xét CSDL Quản lý đề án công ty, ta có các phát biểu RBTV như sau:

* R1: Lương của mỗi nhân viên không được vượt quá 80000
* R2: Mỗi nhân viên phải có một mã nhân viên riêng biệt và không trùng nhau
* R3: Các đề án có tên là “Sản phẩm X” phải có địa điểm khác “TPHCM”
* R4: Mỗi nhân viên phải thuộc một phòng ban
* R5: Lương của nhân viên không được cao hơn lương của trưởng phòng
* R6: Ngày nhận chức của trưởng phòng phải lớn hơn ngày sinh

### 3.1.2. Các yếu tố của một RBTV

Một RBTV bao gồm ba yếu tố:

* **Nội dung**: nội dung của RBTV có thể được thể hiện một cách dễ hiểu và tường minh bằng ngôn ngữ tự nhiên hoặc thể hiện một cách tổng quát hơn bằng đại số quan hệ, phép toán quan hệ, mã giả ...
* **Bối cảnh**: Là những quan hệ mà RBTV có hiệu lực. Bối cảnh của một RBTV có thể là một quan hệ hoặc nhiều quan hệ.
* **Bảng tầm ảnh hưởng**: RBTV có thể bị vi phạm khi thực hiện các thao tác cập nhật dữ liệu (thêm, xóa, sửa) trên bối cảnh. Bảng tầm ảnh hưởng dùng để xác định thời điểm cần kiểm tra RBTV.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên** RBTV | **Thêm** | **Xóa** | **Sửa** |
| Quan hệ 1 | + | - | + |

Trong đó:

* Quan hệ 1,..., Quan hệ k: Bối cảnh của RBTV
* + : Cần phải kiểm tra RBTV
* - : Không cần kiểm tra RBTV

**Ví dụ**: Xét RBTV: mỗi nhân viên phải có một mã nhân viên riêng biệt và không trùng nhau

* Nội dung: Mỗi nhân viên phải có một mã nhân viên riêng biệt và không trùng nhau.
* Bối cảnh: Quan hệ NHANVIEN
* Bảng tầm ảnh hưởng

|  |  |  |  |
| --- | --- | --- | --- |
| **R1** | **Thêm** | **Xóa** | **Sửa** |
| NHANVIEN | + | - | + |

## 3.2. Phân loại các RBTV

Nhằm giúp người thiết kế CSDL có thể phát hiện đầy đủ các RBTV, dựa theo bối cảnh của RBTV ta có các loại RBTV như sau:

* RBTV có bối cảnh là một quan hệ
  + RBTV miền giá trị (Ràng buộc toàn vẹn R1)
  + RBTV liên bộ (Ràng buộc toàn vẹn R2)
  + RBTV liên thuộc tính (Ràng buộc toàn vẹn R3)
* RBTV có bối cảnh là nhiều quan hệ
  + RBTV tham chiếu (Ràng buộc toàn vẹn R4)
  + RBTV liên bộ liên quan hệ (Ràng buộc toàn vẹn R5)
  + RBTV liên thuộc tính liên quan hệ (Ràng buộc toàn vẹn R6)
* RBTV thuộc tính tổng hợp

## 3.3. Cài đặt các ràng buộc toàn vẹn.

Để cài đặt RBTV ta có hai lựa chọn cài đặt như sau:

* **Create Table**: Định nghĩa RBTV trong lúc thiết kế cấu trúc bảng.
* **Alter Table**: Định nghĩa RBTV trong khi hiệu chỉnh bảng.

Để kiểm tra hoặc xem các RBTV của một table ta có hai cách như sau:

* **Sp\_HelpConstraint** <Tên Table>

Bật cửa số Object Explorer 🡪 trong mục Database chọn database cần tạo ràng buộc 🡪 mở nhánh Constraint của từng bảng.

### 3.3.1. Cài đặt RBTV miền giá trị

#### 3.3.1.1. Ràng buộc NOT NULL

Khi khai báo ràng buộc NOT NULL trong định nghĩa của thuộc tính thì ta bắt buộc thuộc tính này phải có giá trị. Nếu ta không chỉ thị gì trong định nghĩa của thuộc tính thì nó có thể có hoặc không có giá trị.

Có 2 tình huống tạo ràng buộc NOT NULL cho một thuộc tính trong một quan hệ:

* Tạo ràng buộc khi tạo mới bảng
* Tạo ràng buộc đối với 1 bảng đã có sẵn.

**Tình huống 1: Tạo ràng buộc khi tạo mới bảng**

* **Cú pháp**:

**CREATE TABLE <TABLE\_NAME>**

(

COLUMN\_NAME DATATYPE NOT NULL

)

* **Ví dụ:** Tạo ràng buộc NOT NULL cho cột tên nhân viên

CREATE TABLE NHANVIEN

(

TENNV NVARCHAR(20) NOT NULL

)

**Tình huống 2: tạo RBTV khi bảng đã tồn tại**

* **Cú pháp**:

**ALTER TABLE** <TABLE\_NAME>

**ADD COLUMN** COLUMN\_NAME DATATYPE NOT NULL

* **Ví dụ:** Tạo ràng buộc NOT NULL cho cột tên nhân viên

ALTER TABLE <TABLE\_NAME>

ADD COLUMN TENNV NVARCHAR(20) NOT NULL

#### 3.3.1.2. Ràng buộc mặc định.

Ràng buộc DEFAULT được cài đặt trong trường hợp ta muốn gán một giá trị mặc định tự động cho một thuộc tính khi người dùng không nhập dữ liệu.

**Tình huống 1: tạo RBTV khi tạo mới bảng**

* **Cú pháp:**

**CREATE TABLE <TABLE\_NAME>**

(

COLUMN\_NAME DATATYPE DEFAULT value|function

)

* **Ví dụ:** Hãy ràng buộc cho cột lương có giá trị mặc định là 2000

CREATE TABLE NHANVIEN

(

LUONG INT DEFAULT 2000

)

**Tình huống 2: tạo RBTV khi bảng đã tồn tại**

* **Cú pháp**:

**ALTER TABLE** <TABLE\_NAME>

**ADD COSTRAINT** CONTRAINT\_NAME **DEFAULT** value|function **FOR** COLUMN\_NAME

* **Ví dụ**: tạo ràng buộc cho cột lương có giá trị mặc định là 2000

ALTER TABLE NHANVIEN

ADD COSTRAINT DF\_LUONG DEFAULT 2000 FOR LUONG

#### 3.3.1.3. Ràng buộc CHECK

Ràng buộc CHECK được sử dụng để ràng buộc các giá trị trong cột phải thỏa mãn một điều kiện hoặc quy tắc theo định nghĩa.

**Tình huống 1: tạo RBTV khi tạo mới bảng**

* **Cú pháp**:

**CREATE TABLE TABLE\_NAME**

(

COLUMN\_NAME DATATYPE [DEFAULT| …],

[, … n],

[CONSTRAINT NAME] **CHECK** (expression)

)

* **Ví dụ**: tạo ràng buộc cho cột lương có giá trị nhỏ hơn 8000

CREATE TABLE NHANVIEN

(

MANV VARCHAR (20) NOT NULL,

LUONG INT DEFAULT 2000

CONSTRAINT CK\_LUONG CHECK (LUONG < 8000)

)

**Tình huống 2: tạo RBTV khi bảng đã tồn tại**

* **Cú pháp**:

**ALTER TABLE** TABLE\_NAME

**ADD** [CONSTRAINT CT\_NAME] CHECK (Expression)

* **Ví dụ:** tạo ràng buộc cho cột lương có giá trị nhỏ hơn 8000

ALTER TABLE NHANVIEN

ADD CONSTRAINT CT\_LUONG CHECK (LUONG<8000)

#### 3.3.1.4. Ràng buộc RULE

Rule cũng được sử dụng để cài đặt, định nghĩa một RBTV miền giá trị, tuy nhiên khác với CHECK vốn trình bày trực tiếp nội dung của RBTV trên một quan hệ, RULE lại chỉ ra một quy tắc trên một miền dữ liệu, quy tắc này sao đó sẽ được áp dụng cụ thể vào một hoặc nhiều quan hệ. Ưu điểm của RULE so với CHECK trong trường hợp cài đặt RBTV miền giá trị là khi có sự thay đổi về nội dung của RBTV, nếu như cài đặt bằng CHECK ta phải chỉnh sửa lại ở tất cả các quan hệ ta đã cài đặt RBTV thì đối với RULE, ta chỉ cần thay đổi nội dung của RULE, các quan hệ áp dụng quy tắc của RULE đó sẽ tự động cập nhật sự thay đổi này. Một điểm khác biệt nữa giữa CHECK và RULE đó chính là CHECK bản thân là một đối tượng CONSTRAINT tồn tại trong một bảng của database, trong khi RULE là một đối tượng của cả database.

Quy trình để cài đặt và thực thi một RULE gồm 3 bước:

**Bước 1**: Định nghĩa đối tượng RULE:

**Cú pháp:** CREATE RULE RULE\_NAME

AS

<EXPRESSION>

**Bước 2:** Áp dụng RULE cho các cột trên các quan hệ

**Cú pháp:**

EXEC SP\_BINDRULE RULE\_NAME TABLE\_NAME.COLUMN\_NAME

**Ví dụ:** Cài đặt ràng buộc miền giá trị “Lương của một nhân viên phải ít hơn 8000”

**Bước 1**: Định nghĩa RULE

DECLARE @range INT

CREATE RULE RANGE\_LUONG

AS

@range < 8000

**Bước 2**: Thực thi RULE

EXEC SP\_BINDRULE RANGE\_LUONG NHANVIEN.LUONG

### 3.3.2. Cài đặt RBTV liên bộ

#### 3.3.2.1. Ràng buộc Unique (duy nhất)

**Unique Constraints** dùng để đảm bảo không có giá trị trùng ở các cột Mặc dù cả Unique constraint và Primary key constrain đều tuân theo tín duy nhất, nhưng hãy sử dụng Unique contarint khi ta muốn bảo đảm tính duy nhất của:

* Một cột, hay sự kết hợp giữa các cột, vốn không phải là khóa chính.
* Một cột chấp nhận giá trị null, trong khi đó constraint primary key không thể ấn định trên cột này.
* Một bảng có thể có nhiều Unique constraint.

**Tình huống 1: tạo RBTV khi tạo mới bảng:**

* **Cú pháp**

**CREATE TABLE TABLE\_NAME**

(

COLUMN\_NAME DATATYPE [NOT NULL| …],

[, … n],

[CONSTRAINT NAME] **UNIQUE** (COLUMN\_NAME)

)

* **Ví dụ**: tạo ràng buộc tên phòng ban là duy nhất

CREATE TABLE PHONGBAN

(

MAPB VARCHAR(20) NOT NULL,

TENPHG NVARCHAR(30) UNIQUE

)

**Hoặc**

CREATE TABLE PHONGBAN

(

MAPB VARCHAR(20) NOT NULL,

TENPHG NVARCHAR(30),

CONSTRAINT UQ\_TENPHG UNIQUE (TENPHG)

)

**Tình huống 2: tạo RBTV khi bảng đã tồn tại**

* **Cú pháp**

**ALTER TABLE** TABLE\_NAME

**ADD** [CONSTRAINT CT\_NAME] UNIQUE (COLUMN\_NAME)

* **Ví dụ:**

ALTER TABLE PHONGBAN

ADD CONSTRAINT CT\_TENPHG UNIQUE (TENPHG)

#### 3.3.2.2. Ràng buộc khóa chính (primary key)

Một quan hệ thường có một hay nhiều thuộc tính có các giá trị riêng để nhận biết hai bộ bất kỳ trong cùng một quan hệ. Các thuộc tính này được gọi là khóa chính (Primarykey) của quan hệ và bảo đảm tính toàn vẹn thực thể trong quan hệ đó.

* SQL Server tự động tạo một chỉ mục cho quan hệ ứng với các thuộc tính tham gia primary key constraint.
* Một quan hệ chỉ có một constraint Primay key.
* Một thuộc tính nằm trong constraint Primay key không thể chấp nhận giá trị Null, trùng lắp. Bởi vì Primay key constraint bảo đảm tính duy nhất của dữ liệu nên chúng thường được ấn định cho cột nhận dạng (identity column).
* Nếu một Primay key constraint được ấn định trên nhiều cột, các giá trị có thể được lặp lại trong một cột, nhưng mỗi sự kết hợp giá trị từ tất cả các cột trong Primay key constraint phải là sự kết hợp duy nhất.

**Tình huống 1: tạo RBTV khi tạo mới bảng**

* **Cú pháp**

**CREATE TABLE TABLE\_NAME**

(

COLUMN\_NAME DATATYPE [NOT NULL| …],

[, … n],

[CONSTRAINT NAME] **PRIMARY KEY** (COLUMN\_NAME)

)

* **Ví dụ:** tạo ràng buộc khóa chính cho thuộc tính mã phòng

CREATE TABLE PHONGBAN

(

MAPB VARCHAR(20) PRIMARY KEY,

TENPHG NVARCHAR(30) UNIQUE

)

**Hoặc**

CREATE TABLE PHONGBAN

(

MAPB VARCHAR(20),

TENPHG NVARCHAR(30),

CONSTRAINT PK\_MAPB PRIMARY KEY (MAPB)

)

**Tình huống 2: tạo RBTV khi bảng đã tồn tại**

* **Cú pháp**

**ALTER TABLE** TABLE\_NAME

**ADD** [CONSTRAINT PK\_NAME] PRIMARY KEY (COLUMN\_NAME)

* **Ví dụ:** tạo ràng buộc khóa chính cho thuộc tính mã phòng

ALTER TABLE PHONGBAN

ADD CONSTRAINT PK\_MAPB PRIMARY KEY (MAPB)

#### 3.3.2.3. Ràng buộc liên thuộc tính

Ta có thể sử dụng CHECK để cài đặt các RBTV liên thuộc tính

**Ví dụ:** Trong quan hệ NHANVIEN, ta bổ sung thêm thuộc tính NGAYVAOLAM. Cài đặt RBTV NGAYSINH < NGAYVAOLAM

* Tạo RBTV khi tạo mới bảng:

CREATE TABLE NHANVIEN

( [, …n],

NGAYSINH DATETIME,

NGAYVAOLAM DATETIME,

CONSTRAINT CK\_NGAYSINH\_NGAYVAOLAM   
 CHECK (NGAYSINH < NGAY VAO LAM))

* Tạo RBTV khi bảng đã tồn tại rồi:

ALTER TABLE NHANVIEN

ADD CONSTRAINT CK\_NGAYSINH\_NGAYVAOLAM

CHECK (NGAYSINH < NGAYVAOLAM)

### 3.3.3. Ràng buộc tham chiếu – ràng buộc khóa ngoại (foreign key)

Chỉ ra mối liên hệ ràng buộc tham chiếu giữa bảng này với bảng khác. Từ khoá ON DELETE CASCADE được chỉ định trong dạng khoá này để chỉ khi dữ liệu cha bị xoá thì dữ liệu con cũng tự động bị xoá theo.

**Tình huống 1: tạo RBTV khi tạo mới bảng:**

**Cú pháp**

**CREATE TABLE TABLE\_NAME**

(

COLUMN\_NAME DATATYPE [NOT NULL| …],

[, … n],

[CONSTRAINT FK\_NAME] **FOREIGN KEY** (COLUMN\_NAME\_1)

**REFERENCES** TABLE\_NAME (COLUMN\_NAME\_2) )

**Trong đó:**

* *FK\_NAME:* tên khóa ngoại
* *COLUMN\_NAME\_1:* thuộc tính được chọn làm khóa ngoại
* *TABLE\_NAME:* tên bảng tham chiếu (bảng chứa khóa chính)
* *COLUMN\_NAME\_2:* thuộc tính trong bảng chứa khóa chính được tham chiếu

**Ví dụ**: tạo ràng buộc khóa ngoại cho thuộc tính PHONG trong bảng NHANVIEN

NHANVIEN (**MANV**, HONV, TENLOT, TENNV, NGSINH, DIACHI, LUONG, PHAI, MA\_NQL, PHONG)

PHONGBAN (**MAPB**, TENPB, TRG\_PHG, NGAY\_NHAMCHUC)

CREATE TABLE NHANVIEN

(

MANV NVARCHAR(10) NOT NULL,

HOTEN NVARCHAR(40),

CONSTRAINT FK\_PHONG **FOREIGN KEY** (PHONG)

**REFERENCES** PHONGBAN (MAPB)

)

**Tình huống 2: tạo RBTV khi bảng đã tồn tại**

* **Cú pháp**

**ALTER TABLE** TABLE\_NAME

**ADD** [CONSTRAINT FK\_NAME] **FOREIGN KEY** (COLUMN\_NAME\_1)

**REFERENCES** TABLE\_NAME (COLUMN\_NAME\_2)

Trong đó:

* *FK\_NAME:* tên khóa ngoại
* *COLUMN\_NAME\_1:* thuộc tính được chọn làm khóa ngoại
* *TABLE\_NAME:* tên bảng tham chiếu (bảng chứa khóa chính)
* *COLUMN\_NAME\_2:* thuộc tính trong bảng chứa khóa chính được tham chiếu

**Ví dụ:** tạo ràng buộc khóa ngoại cho thuộc tính PHONG trong bảng NHANVIEN

ALTER TABLE NHANVIEN

ADD CONSTRAINT FK\_PHONG **FOREIGN KEY** (PHONG)

**REFERENCES** PHONGBAN (MAPB)

## 3.4. Xóa bỏ RBTV

### 3.4.1. Hủy bỏ các ràng buộc toàn vẹn CONSTRAINT

* **Xóa RBTV**

ALTER TABLE TABLE\_NAME

DROP CONSTRAINT <CONSTRAINT\_NAME>

* **Tắt hiệu lực kiểm tra của các RBTV**

ALTER TABLE TABLE\_NAME

NOCHECK CONSTRAINT ALL|<CONSTRAINT\_NAME>

* **Mở lại hiệu lực kiểm tra của các RBTV**

ALTER TABLE TABLE\_NAME

NOCHECK CONSTRAINT ALL|< CONSTRAINT\_NAME >

### 3.4.2. Hủy bỏ các ràng buộc toàn vẹn RULE

* **Gỡ bỏ một RULE ra khỏi một quan hệ:**

EXEC SP\_UNBINDRULE < TABLE\_NAME >.< COLUMN\_NAME >

* **Xóa bỏ một RULE:**

DROP RULE <RULE\_NAME>

* **Lưu ý**:
* Chỉ có thể xóa một RULE khi RULE đó không còn được gắn kết cho bất kỳ quan hệ nào.
* Từ sau phiên bản SQL Server 2008, lệnh DROP RULE được loại bỏ.

# BÀI TẬP CHƯƠNG 3

## CÂU HỎI

**Câu hỏi 1:** RBTV là gì ? Trình bày các yếu tố của RBTV?.

**Câu hỏi 2:** Hãy phân loại RBTV dựa theo bối cảnh quan hệ?

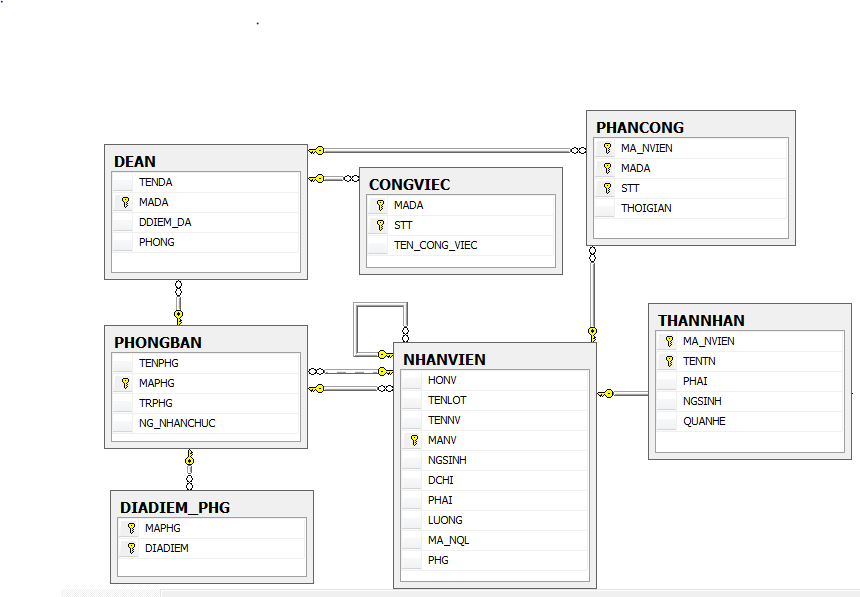
**Câu hỏi 3:** Trình bày hai cách cài đặt RBTV ?

**Câu hỏi 4**: Trình bày các loại ràng buộc miền giá trị và ràng buộc liên thuộc tính?

**Câu hỏi 5:** Phân biệt điểm giống và khác nhau của ràng buộc RULE và CHECK?

## BÀI TẬP

**Bài tập 1: Cho lược đồ Cơ sở dữ liệu sau:**



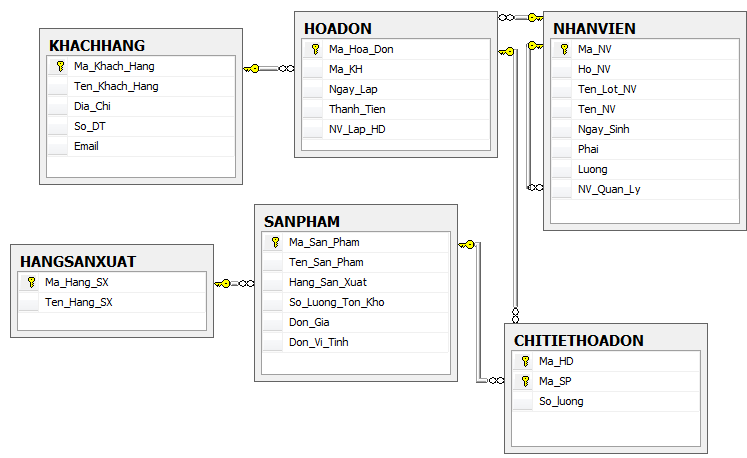
**Tiến hành cài đặt các ràng buộc toàn vẹn sau đây**

1. Phái của Nhân Viên và Thân Nhân phải là “Nam” hoặc “Nữ” (Sử dụng Rule và Check)
2. Lương của Nhân Viên tối thiểu là 500 (Sử dụng Rule và Check)
3. Tên công việc là duy nhất (Ràng buộc Unique)
4. Bổ sung thuộc tính NGAYVAOLAM trong bảng NHANVIEN. Cài đặt ràng buộc toàn vẹn Ngày sinh phải nhỏ hơn Ngày vào làm
5. Cài đặt ràng buộc khóa chính cho các bảng trên trong câu lệnh Create Table
6. Cài đặt ràng buộc khóa chính cho các bảng trên bằng câu lệnh Alter Table sau khi đã tạo bảng không có khóa chính. So sánh với câu 5 và nhận xét
7. Cài đặt ràng buộc khóa ngoại cho các bảng trên trong câu lệnh Create Table
8. Cài đặt ràng buộc khóa ngoại cho các bảng trên bằng câu lệnh Alter Table. So sánh với câu 7 và nhận xét.

**Bài tập 2: Thực hiện các yêu cầu liên quan đến các ràng buộc toàn vẹn đã cài đặt ở bài tập 1 như sau:**

1. Viết câu lệnh thay đổi nội dung ràng buộc ở câu 1 bài tập 1 thành “Phái của nhân viên và thân nhân phải là “NAM” hoặc “NỮ” ( Sử dụng Rule và Check)”
2. Viết câu lệnh thay đổi nội dung ràng buộc ở câu 2 của bài tập 1 thành “Lương của nhân viên cao nhất là 30000 và thấp nhất là 25000 (Sử dụng Rule và Check)”
3. Viết câu lệnh tắt kiểm tra ràng buộc toàn vẹn của của các ràng buộc trên bảng NHANVIEN và bảng THANNHAN. Thêm một bộ dữ liệu trong bảng NHANVIEN có Phái là “NA” và nhận xét.
4. Viết câu lệnh bật kiểm tra ràng buộc toàn vạn của các ràng buộc trên bảng NHANVIEN, sau đó thêm một bộ dữ liệu trong bảng NHANVIEN có Phái là “NA” và nhận xét.

**Sử dụng lược đồ sau để thực hiện các bài tập 3, bài tập 4 (mô tả chi tiết về lược đồ quan hệ xem lại ở bài tập chương 1)**



**Bài tập 3: Tiến hành cài đặt các ràng buộc toàn vẹn như sau (Sử dụng cả Rule và Check)**

1. Nhân viên phải có tuổi ít nhất là 20 tuổi.
2. Số lượng của một chi tiết hóa đơn không được vượt quá 50.
3. Số lượng tồn kho của một sản phẩm không được thấp hơn 10.
4. Ngày lập hóa đơn phải nhỏ hơn hoặc bằng ngày hiện tại.
5. Cài đặt ràng buộc khóa chính cho các bảng trên trong câu lệnh Create Table
6. Cài đặt ràng buộc khóa chính cho các bảng trên bằng câu lệnh Alter Table sau khi đã tạo bảng không có khóa chính. So sánh với câu 5 và nhận xét
7. Cài đặt ràng buộc khóa ngoại cho các bảng trên trong câu lệnh Create Table
8. Cài đặt ràng buộc khóa ngoại cho các bảng trên bằng câu lệnh Alter Table. So sánh với câu 7 và nhận xét.

**Bài tập 4: Thực hiện các yêu cầu liên quan đến các ràng buộc toàn vẹn đã cài đặt ở bài tập 3 như sau**

1. Viết câu lệnh thay đổi nội dung của ràng buộc toàn vẹn ở câu 1 bài tập 3 như sau: “Nhân viên phải có tuổi trong khoảng từ 20 đến 35 tuổi” (Sử dụng Rule và Check)
2. Viết câu lệnh gỡ bỏ hiệu lực của Rule được cài đặt ở câu 3 bài tập 3, sau đó tiến hành thêm một bộ dữ liệu sản phẩm có số lượng tồn kho là 8. Nhận xét kết quả.
3. Viết câu lệnh xóa ràng buộc toàn vẹn Check được cài đặt ở câu 3 bài tập 3 (sau khi thực hiện câu 2 của bài tập 4), sau đó tiến hành thêm một bộ dữ liệu sản phẩm có số lượng tồn kho là 8. Nhận xét kết quả.
4. Viết các câu lệnh xóa bỏ tất cả các ràng buộc toàn vẹn Rule và Check đã được cài đặt ở bài tập 3 và bài tập 4.

# CHƯƠNG 4 QUẢN LÝ VÀ THAO TÁC DỮ LIỆU

# MỤC ĐÍCH VÀ YÊU CẦU

**Mục đích**:

Chương này nhằm cung cấp cho sinh viên những kiến thức chung về:

* Phương pháp phân tích yêu cầu truy vấn, thao tác truy vấn trên cơ sở dữ liệu thông qua mệnh đề Select …
* Phương pháp tương tác giữa người dùng với cơ sở dữ liệu thông qua hoạt động cập nhật (thêm, xóa, sửa) trên dữ liệu có sẵn lưu trữ trên máy.

**Yêu cầu**:

Sau khi học xong chương này sinh viên có khả năng:

* Phân tích các tình huống theo yêu cầu đặt ra;
* Nắm vững cú pháp, phân tích vấn đề và vận dụng thành thạo các câu lệnh để truy vấn và cập nhật dữ liệu.
* Hiểu rõ công dụng của các lệnh Select, Insert, Update, Detele.

# NỘI DUNG

## 4.1. Truy vấn dữ liệu

### 4.1.1. Giới thiệu chung

Như đã đề cập ở trên, hệ quản trị CSDL là một hệ thống phần mềm nhằm cung cấp cho người sử dụng một môi trường vừa thích hợp, vừa hiệu quả để khai thác CSDL theo 3 chức năng mô tả, lưu trữ, tìm kiếm, và cập nhật dữ liệu. Để thực hiện được các tính năng trên, HQT CSDL phải cung cấp một ngôn ngữ hỗ trợ tương tác giữa người sử dụng với CSDL. Ngôn ngữ đó bao gồm:

*Ngôn ngữ định nghĩa dữ liệu* (Data Definition Language – DDL): Cho phép khai báo cấu trúc các bảng của CSDL, khai báo các mối liên hệ của dữ liệu (relationship) và các quy tắc áp đặt lên các dữ liệu đó (đã đề cập ở chương 1)

*Ngôn ngữ thao tác dữ liệu* (Data Manipullation Language - DML) cho phép người sử dụng có thể thêm (insert), xoá (delete), sửa (update) dữ liệu trong CSDL.

*Ngôn ngữ truy vấn dữ liệu (*hay ngôn ngữ hỏi đáp có cấu trúc (Structured Query Language - SQL)): Cho phép người sử dụng khai thác CSDL để truy vấn các thông tin cần thiết trong CSDL.

*Ngôn ngữ quản lý dữ liệu* (Data Control Language - DCL): Cho phép những người quản trị hệ thống thay đổi cấu trúc của các bảng dữ liệu (đã đề cập ở chương 1), khai báo bảo mật thông tin và cấp quyền khai thác CSDL cho người sử dụng (sẽ đề cập trong chương 9)

Trong phạm vi chương 4, chúng ta sẽ tìm hiểu ngôn ngữ thao tác và truy vấn dữ liệu – SQL.

Cú pháp tổng quát của một câu lệnh truy vấn có dạng như sau:

**SELECT distinct /\*/danh sách thuộc tính/ <biểu thức>, ...**

**FROM <danh sách các quan hệ>**

**WHERE <biểu thức điều kiện>**

**GROUP BY <danh sách thuộc tính>**

**HAVING <điều kiện nhóm>**

**ORDER BY <danh sách các thuộc tính [desc]>**

Trong đó:

* ***<biểu thức>*** (expression) là sự kết hợp một cách hợp lệ giữa các thuộc tính, các toán tử và các hàm. Sau đây sẽ là các toán tử và hàm thông dụng nhất. (cần chú ý rằng cách sử dụng các toán tử và các hàm này còn tuỳ thuộc vào câu lệnh SELECT của ngôn ngữ được sử dụng).

*Các toán tử số học: ^ (luỹ thừa),\* (nhân), / (chia), mod (phần dư), + (cộng), - (trừ)*

*Các toán tử luận lý: not (phủ định), and (phép hội), or (phép tuyển)*

*Các toán tử tập hợp: In (danh sách các giá trị), LIKE, NOT LIKE, union (phép hợp), intersect (phép giao), minus (phép trừ)*

*Các toán tử giới hạn miền giá trị: between … and (giá trị nằm trong khoảng từ … đến …), not between … and (giá trị không nằm trong khoảng từ … đến …)*

*Các toán tử so sánh : =, <>, >, <, >=, <=*

*Các hàm xử lý ngày tháng*

* **Date**(): Trả về ngày tháng năm của hệ thống
  + **Time**(): Trả về giờ phút giây của hệ thống
  + **Day** (biểu thức ngày): Trả về một trị số từ 1 đến 31 của biểu thức ngày
  + **Month** (biểu thức ngày): Trả về một số từ 1 đến 12 - là tháng của biểu thức ngày
  + **Year** (biểu thức ngày): Trả về năm của biểu thức ngày
  + **Len** (biểu thức chuỗi): Trả về chiều dài của chuỗi

*Các hàm tính toán theo nhóm*

* **Sum** (thuộc tính): tính tổng giá trị của các bộ theo thuộc tính đã chỉ ra.
* **Max** (thuộc tính): cho biết giá trị lớn nhất của các bộ theo thuộc tính đã chỉ ra.
* **Min** (thuộc tính): cho biết giá trị nhỏ nhất của các bộ theo thuộc tính đã chỉ ra.
* **Avg** (thuộc tính): cho biết giá trị trung bình của các bộ theo thuộc tính đã chỉ ra.
* **Count** (thuộc tính) / distinct <thuộc tính>
* **count** \*: đếm tất cả các bộ
* **count** (thuộc tính):chỉ đếm những bộ mà giá trị của thuộc tính là khác NULL
* **count distinct**<thuộc tính>

**Lưu ý**: Ngôn ngữ SQL ***không phân biệt*** chữ hoa hay chữ thường.

Để minh họa cho các ví dụ trong chương này, chúng ta sẽ dùng lại lược đồ cơ sở dữ liệu: QUẢN LÝ ĐỀ ÁN CÔNG TY

**NHANVIEN** (MANV, HONV, TENLOT, TENNV, NGSINH, DIACHI, LUONG, PHAI, MA\_NQL, PHOG)

**PHONGBAN** (MAPB, TENPB, TRG\_PHG, NGAY\_NHAMCHUC)

**DEAN** (MADA, TENDA, PHONG, DIADIEM\_DA)

**PHANCONG** (MA\_NV, MA\_DA, THOIGIAN)

**DIADIEM\_PHONG** (MAPHG, DIADIEM)

**THANNHAN** (MA\_NVIEN, HOTN, TENLOTTN, TENTN, NGSINH, DIACHI, PHAI)

### 4.1.2. Tìm thông tin từ các cột của bảng - mệnh đề SELECT

**a. Chức năng**

Rút trích một vài thông tin, một vài cột từ các quan hệ tham gia câu truy vấn. Câu lệnh SELECT tương ứng với chức năng của phép chiếu trong NN ĐSQH.

**b. Cú pháp**

**SELECT distinct /\*/danh sách thuộc tính/ <biểu thức>, ...**

**FROM <danh sách các quan hệ>**

**Trong đó:**

* *Danh sách thuộc tính:* bao gồm tất cả các cột muốn rút trích thông tin từ các quan hệ trong CSDL. Các thuộc tính cách nhau bởi dấu phẩy “**,**”. Các thuộc tính trùng tên phải đặt sau tên quan hệ gắn liền với thuộc tính đó (theo quy tắc: TEN\_QUANHE.TEN\_THUOCTINH)
* *Danh sách các quan hệ:* bao gồm tất cả các quan hệ tham gia hỗ trợ thực hiện yêu cầu của câu truy vấn. Các quan hệ cách nhau bởi dấu phẩy “**,**”.

**c. Ví dụ minh họa**

**Cho quan hệ NHANVIEN có thể hiện như sau:**

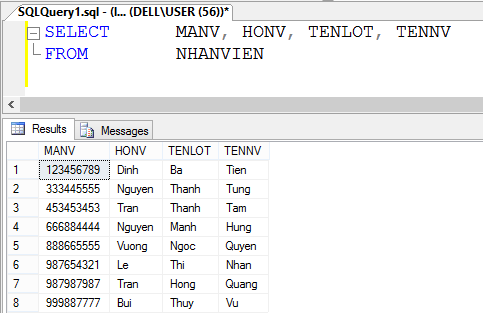
****

**Ví dụ 1**: Hãy cho biết thông tin mã nhân viên, họ và tên đầy đủ của tất cả các nhân viên trong công ty.

**SELECT** MANV, HONV, TENLOT, TENNV

**FROM** NHANVIEN

**Kết quả:**

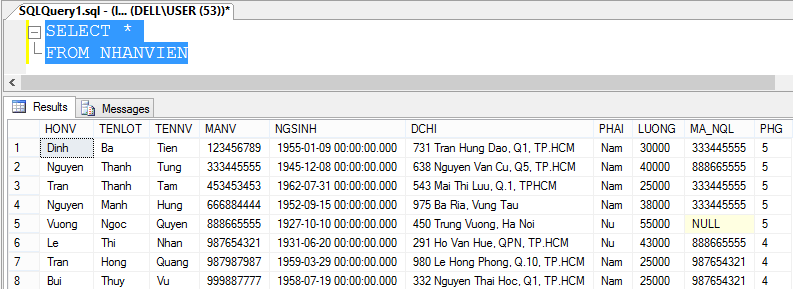


**Ví dụ 2**: Hãy cho biết *tất cả* các thông tin của các nhân viên trong công ty.

**SELECT** NHANVIEN.\*

**FROM** NHANVIEN

**Kết quả:**

****

**d. Một số tình huống truy vấn**

**TH1**: Hãy cho biết thông tin mã nhân viên, họ và tên đầy đủ của tất cả các nhân viên trong công ty. Thông tin thể hiện ra ngoài có dạng:

|  |  |
| --- | --- |
| **Mã nhân viên** | **Họ và tên** |
| NV01 | Nguyễn Văn A |

**SELECT** MANV as ‘Mã nhân viên’, HONV+‘ ’+TENLOT+‘ ’+TENNV as ‘Họ và tên’

**Tên bí danh& phép nối chuỗi**

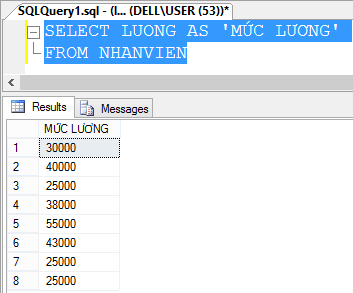
**FROM** NHANVIEN

**TH2**: Hãy cho biết tất cả các mức lương trong công ty:

**SELECT** LUONG AS ‘MỨC LƯƠNG’

**FROM** NHANVIEN

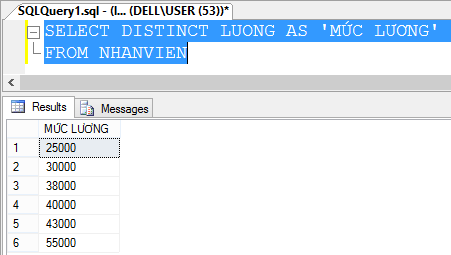
**Kết quả:**



**SELECT DISTINCT** LUONG AS ‘MỨC LƯƠNG’

**FROM** NHANVIEN

**Kết quả:**



### 4.1.3. Chọn các dòng của bảng – mệnh đề WHERE

**a. Chức năng**

Rút trích các dòng từ các quan hệ tham gia câu truy vấn thỏa mãn điều kiện trong mệnh đề WHERE. Các biểu thức điều kiện trong mệnh đề WHERE tương ứng với các điều kiện của phép chọn trong NN ĐSQH.

**b. Cú pháp**

**SELECT** distinct /\*/danh sách thuộc tính/ <biểu thức>, ...

**FROM**  <danh sách các quan hệ>

**WHERE**  <biểu thức điều kiện>

**Trong đó:**

* *Danh sách thuộc tính:* bao gồm tất cả các cột muốn rút trích thông tin từ các quan hệ trong CSDL. Các thuộc tính cách nhau bởi dấu phẩy “**,**”. Các thuộc tính trùng tên phải đặt sau tên quan hệ gắn liền với thuộc tính đó (theo quy tắc: TEN\_QUANHE.TEN\_THUOCTINH).
* *Danh sách các quan hệ:* bao gồm tất cả các quan hệ tham gia hỗ trợ thực hiện yêu cầu của câu truy vấn. Các quan hệ cách nhau bởi dấu phẩy “**,**”.

*Biểu thức điều kiện:* các dòng dữ liệu được rút ra từ CSDL phải thỏa mãn các biểu thức điều kiện đã quy định trong mệnh đề WHERE. Các điều kiện được nối với nhau bởi AND/OR.

**c. Ví dụ minh họa**

**Ví dụ 1**: Cho biết họ và tên các nhân viên có mức lương trên 3000000.

**SELECT** MANV, HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE** LUONG>3000000

**Ví dụ 2**: Cho biết họ và tên các nhân viên có mức lương từ 5000000 đến 7000000.

**SELECT** MANV, HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE** LUONG >= 5000000 **AND** LUONG <= 7000000

**Ví dụ 3**: Cho biết họ tên các nhân viên cùng tên phòng mà nhân viên đó trực thuộc.

**SELECT** HONV, TENLOT, TENNV, TENPB

**FROM** NHANVIEN, PHONGBAN

**WHERE** PHOG = MAPB

**Ví dụ 4**: Cho biết họ tên các nhân viên nữ có mức lương dưới 3000000 hoặc trên 5000000.

**SELECT** MANV, HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE** PHAI = ‘Nữ’ **and (**LUONG < 3000000 **or** LUONG > 5000000)

**d. Một số tình huống truy vấn:**

**TH1**: Cho biết họ và tên các nhân viên có mức lương **từ** 5000000 **đến** 7000000.

**Sử dụng toán tử BETWEEN … AND …**

**SELECT** MANV, HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE** LUONG **BETWEEN** 5000000 **AND** 7000000

* Cho biết họ tên các nhân viên nữ có mức lương dưới 3000000 hoặc trên 5000000.

**Sử dụng toán tử NOT BETWEEN … AND …**

**SELECT** MANV, HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE** PHAI = ‘Nữ’ **and (**LUONG **NOT BETWEEN** 3000000 **AND** 5000000)

**TH2**: Cho biết họ và tên của tất cả các nhân viên có họ là Nguyễn.

***Cách 1***: So sánh bằng:

**SELECT** MANV, HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE** HONV **=** ‘Nguyễn’

***Cách 2*:** Sử dụng toán tử so sánh chuỗi LIKE

**SELECT** MANV, HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE** HONV **LIKE** ‘Nguyễn’

**TH3**: Cho biết họ và tên của tất cả các nhân viên có họ bắt đầu là ký tự N.

Trong trường hợp chỉ so sánh một hoặc một vài ký tự được trích ra từ một chuỗi (tức chuỗi có giá trị không trọn vẹn) ta không được sử dụng toán tử ‘=’ hoặc ‘<>’để so sánh mà phải sử dụng toán tử LIKE hoặc NOT LIKE.

Khi sử dụng toán tử LIKE/ NOT LIKE, người ta quy ước như sau:

* Dấu %: đại diện cho một chuỗi con bất kỳ.
* Dấu \_: đại diện cho một ký tự bất kỳ.

Trong ví dụ trên, ta có thể viết được câu truy vấn như sau:

**SELECT** MANV, HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE** HONV **LIKE** ‘N%’

* Cho biết họ và tên của tất cả các nhân viên có họ bắt đầu là ký tự N, sau ký tự N là 2 kí tự và kết thúc chuỗi.

**SELECT** MANV, HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE** HONV **LIKE** ‘N\_ \_’

### 4.1.4. Thứ tự thể hiện các bản ghi - mệnh đề ORDER BY

**a. Chức năng**

Sắp xếp theo thứ tự tăng dần/ giảm dần các dòng trong quan hệ kết quả theo tiêu chí sắp xếp.

**b. Cú pháp**

**SELECT distinct /\*danh sách thuộc tính/ <biểu thức>, ...**

**FROM <danh sách các quan hệ>**

**[WHERE <biểu thức điều kiện>]**

**ORDER BY <danh sách các thuộc tính [desc/asc]>**

**Trong đó:**

* *Danh sách thuộc tính:* bao gồm tất cả các cột muốn rút trích thông tin từ các quan hệ trong CSDL. Các thuộc tính cách nhau bởi dấu phẩy “**,**”. Các thuộc tính trùng tên phải đặt sau tên quan hệ gắn liền với thuộc tính đó (theo quy tắc: TEN\_QUANHE.TEN\_THUOCTINH).
* *Danh sách các quan hệ:* bao gồm tất cả các quan hệ tham gia hỗ trợ thực hiện yêu cầu của câu truy vấn. Các quan hệ cách nhau bởi dấu phẩy “**,**”.
* *Biểu thức điều kiện:* các dòng dữ liệu được rút ra từ CSDL phải thỏa mãn các biểu thức điều kiện đã quy định trong mệnh đề WHERE. Các điều kiện được nối với nhau bởi AND/OR.
* *Mệnh đề ORDER BY:* danh sách các thuộc tính được chọn làm tiêu chí sắp xếp. Mặc định là sắp xếp theo thứ tự tăng dần.

**c. Ví dụ minh họa:**

**Ví dụ 1:** Cho biết họ tên của các nhân viên, danh sách được sắp theo thứ tự tăng dần của mức lương.

**SELECT** MANV, HONV, TENLOT, TENNV

**FROM** NHANVIEN

**ORDER BY** LUONG

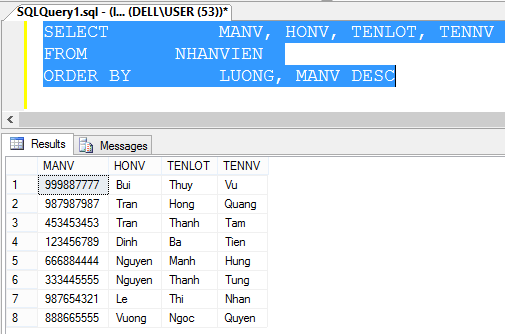
**Ví dụ 2**: Cho biết họ tên của các nhân viên, danh sách được sắp theo thứ tự tăng dần của mức lương, nếu cùng một mức lương thì sắp xếp theo thứ tự giảm dần của mã nhân viên.

**SELECT** MANV, HONV, TENLOT, TENNV

**FROM** NHANVIEN

**ORDER BY** LUONG, MANV **DESC**

**Kết quả:**



* + 1. Gom nhóm dữ liệu – mệnh đề GROUP BY

#### a. Chức năng

Gom nhóm các dòng dữ liệu theo các tiêu chí gom nhóm đặt trong mệnh đề GROUP BY nhằm hỗ trợ phục vụ các mục đích của câu truy vấn.

#### b. Cú pháp

**SELECT distinct /\*/danh sách thuộc tính/ <biểu thức>, ...**

**FROM <danh sách các quan hệ>**

**[WHERE <biểu thức điều kiện>]**

**GROUP BY <danh sách thuộc tính>**

**Trong đó:**

* *Danh sách thuộc tính:* bao gồm tất cả các cột muốn rút trích thông tin từ các quan hệ trong CSDL. Các thuộc tính cách nhau bởi dấu phẩy “**,**”. Các thuộc tính trùng tên phải đặt sau tên quan hệ gắn liền với thuộc tính đó (theo quy tắc: TEN\_QUANHE.TEN\_THUOCTINH).
* *Danh sách các quan hệ:* bao gồm tất cả các quan hệ tham gia hỗ trợ thực hiện yêu cầu của câu truy vấn. Các quan hệ cách nhau bởi dấu phẩy “**,**”.
* *Biểu thức điều kiện:*các dòng dữ liệu được rút ra từ CSDL phải thỏa mãn các biểu thức điều kiện đã quy định trong mệnh đề WHERE. Các điều kiện được nối với nhau bởi AND/OR.
* *Mệnh đề GROUP BY:*danh sách các thuộc tính được chọn làm tiêu chí gom nhóm. Tất cả các thuộc tính đơn lẻ (ngoại trừ các hàm kết hợp và gom nhóm)xuất hiện trong mệnh đề SELECT phải đặt vào GROUP BY.

#### c. Ví dụ minh họa

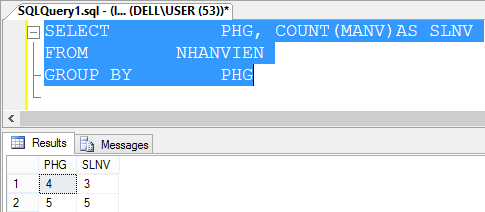
**Ví dụ 1**: Cho biết số lượng nhân viên của từng phòng ban.

**SELECT** PHOG, COUNT(MANV)

**FROM** NHANVIEN

**GROUP BY** PHOG

**Kết quả:**



**Ví dụ 2**: Cho biết tên phòng cùng số lượng nhân viên của phòng tương ứng.

**SELECT** TENPB, COUNT(MANV)

**FROM** NHANVIEN, PHONGBA**N**

**WHERE** PHOG = MAPB

**GROUP BY** PHOG, TENPB

### 4.1.5. Chọn các nhóm thỏa điều kiện – mệnh đề HAVING

**a. Chức năng**

Cho kết quả là các dữ liệu thỏa mãn các điều kiện theo từng nhóm quy định trong GROUP BY.

**b. Cú pháp**

**SELECT distinct /\*/danh sách thuộc tính/ <biểu thức>, ...**

**FROM <danh sách các quan hệ>**

**[WHERE <biểu thức điều kiện>]**

**GROUP BY <danh sách thuộc tính>**

**HAVING <điều kiện trên từng nhóm>**

**Trong đó:**

* *Danh sách thuộc tính:* bao gồm tất cả các cột muốn rút trích thông tin từ các quan hệ trong CSDL. Các thuộc tính cách nhau bởi dấu phẩy “**,**”. Các thuộc tính trùng tên phải đặt sau tên quan hệ gắn liền với thuộc tính đó (theo quy tắc: TEN\_QUANHE.TEN\_THUOCTINH).
* *Danh sách các quan hệ:* bao gồm tất cả các quan hệ tham gia hỗ trợ thực hiện yêu cầu của câu truy vấn. Các quan hệ cách nhau bởi dấu phẩy “**,**”.
* *Biểu thức điều kiện:* các dòng dữ liệu được rút ra từ CSDL phải thỏa mãn các biểu thức điều kiện đã quy định trong mệnh đề WHERE. Các điều kiện được nối với nhau bởi AND/OR.
* *Mệnh đề GROUP BY:* danh sách các thuộc tính được chọn làm tiêu chí gom nhóm. Tất cả các thuộc tính đơn lẻ (ngoại trừ các hàm kết hợp và gom nhóm) xuất hiện trong mệnh đề SELECT phải đặt vào GROUP BY.
* *Điều kiện trên từng nhóm:* dữ liệu kết quả phải thỏa mãn các biểu thức điều kiện đã quy định trong mệnh đề HAVING, điều kiện này được thực hiện trên từng nhóm trong theo tiêu chí gom nhóm trong mệnh đề GROUP BY. Các điều kiện được nối với nhau bởi AND/OR.

**c. Ví dụ minh họa**

**Ví dụ 1:** Cho biết các phòng có mức lương trung bình trên 7000000.

**SELECT** PHOG, AVG (LUONG)

**FROM** NHANVIEN

**GROUP BY** PHOG

**HAVING** AVG (LUONG) > 7000000

**Ví dụ 2**: Cho biết tên phòng có số lượng nhân viên từ 3 nhân viên trở lên.

**SELECT** TENPB, COUNT(MANV)

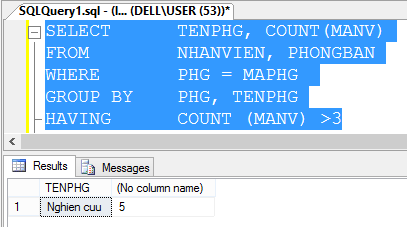
**FROM** NHANVIEN, PHONGBA**N**

**WHERE** PHOG = MAPB

**GROUP BY** PHOG, TENPB

**HAVING** COUNT (MANV) >3

**Kết quả:**



## 4.2. Cập nhật dữ liệu

### 4.2.1. Lệnh INSERT

**a. Chức năng**

Cho phép thêm một hoặc nhiều dòng dữ liệu vào bảng đã tạo sẵn trong CSDL.

**b. Cú pháp**

Có hai hình thức thêm dữ liệu:

* Thêm 1 dòng dữ liệu mới;
* Thêm nhiều dòng dữ liệu từ những dữ liệu đã có sẵn.

Tương ứng với 2 cách thêm, cú pháp tổng quát được xây dựng như sau:

**Thêm một dòng dữ liệu:**

**INSERT INTO**<Tên bảng> [Danh sách các cột trong bảng đã định nghĩa]

**VALUES** (Giá trị cần thêm)

**Thêm nhiều dòng dữ liệu:**

**INSERT INTO**<Tên bảng> [Danh sách các cột trong bảng đã định nghĩa]

***Câu lệnh SELECT***

Trong đó:

* ***Tên bảng***: là tên quan hệ muốn thêm dữ liệu vào;
* ***Danh sách các cột trong bảng đã định nghĩa:*** có thể có hoặc không liệt kê danh sách này. Khi người dùng có liệt kê danh sách các cột, danh sách này không bắt buộc phải theo đúng trình tự trong khai báo định nghĩa ban đầu 🡪 theo trình tự khai báo trong câu lệnh INSERT và ngược lại (nếu không có khai báo danh sách các cột thì giá trị cần thêm cần được sắp sếp theo đúng thứ tự đã khai báo trong định nghĩa cấu trúc bảng)

**c. Ví dụ minh họa**

**Ví dụ 1:** Hãy thêm một dòng vào bảng PHONGBAN

**PHONGBAN** (MAPB, TENPB, TRG\_PHG, NGAY\_NHAMCHUC)

Dữ liệu cần thêm như sau: MP07, kế hoạch, NV008, 06/06/2016

**Cách 1**:

INSERT PHONGBAN

VALUES (‘MP07’, N‘kế hoạch’, ‘NV008’, ‘06/06/2016’)

**Cách 2**:

INSERT PHONGBAN(MAPB, TENPB, TRG\_PHG, NGAY\_NHAMCHUC)

VALUES (‘MP07’, N‘kế hoạch’, ‘NV008’, ‘06/06/2016’)

**Cách 3**:

INSERT PHONGBAN(MAPB, TRG\_PHG, TENPB, NGAY\_NHAMCHUC)

VALUES (‘MP07’, ‘NV008’, N‘kế hoạch’, ‘06/06/2016’)

**Ví dụ 2**: Hãy thống kê số lượng nhân viên của mỗi phòng ban, biết cấu trúc bảng thống kê như sau: THONGKE (MAPB, TENPB, SLNV)

**INSERT** THONGKE

SELECT PHOG, TENPB, COUNT (MANV) AS SLNV

FROM NHANVIEN, PHONGBAN

WHERE PHOG = MAPB

GROUP BY PHOG)

### 4.2.2. Lệnh DELETE

**a. Chức năng**

Cho phép người dùng xóa một hoặc nhiều dòng trên bảng dữ liệu đã có sẵn trong CSDL.

**b. Cú pháp**

**DELETE FROM** <Tên bảng>

[**WHERE** <Điều kiện>]

**c. Ví dụ minh họa**

Hãy thực hiện hai tình huống cập nhật dữ liệu trên bảng NHANVIEN như sau:

**NHANVIEN** (MANV, HONV, TENLOT, TENNV, NGSINH, DIACHI, LUONG, PHAI, MA\_NQL, PHOG)

**Ví dụ 1**: Hãy xóa danh sách các nhân viên:

**DELETE FROM** NHANVIEN

**Ví dụ 2**: Hãy xóa danh sách các nhân viên có mã phòng PB01

**DELETE FROM** NHANVIEN

**WHERE** PHOG = ‘PB01’

### 4.2.3. Lệnh UPDATE

**a. Chức năng**

Cho phép người dùng cập nhật giá trị của thuộc tính trên các dòng của bảng dữ liệu đã có sẵn trong CSDL.

**b. Cú pháp**

**UPDATE** <Tên bảng>

**SET** <Tên thuộc tính> = <giá trị mới>,

<Tên thuộc tính> = <giá trị mới>, …

[**WHERE** <Điều kiện>]

Trong đó:

* ***Tên bảng***: là tên quan hệ muốn cập nhật dữ liệu;
* ***Tên thuộc tính***: tên cột muốn cập nhật dữ liệu;
* ***Giá trị mới***: giá trị sẽ thay thế dữ liệu cũ;
* ***Điều kiện***: chỉ cập nhật các dòng thỏa mãn điều kiện (nếu không có mệnh đề WHERE, tất cả các dòng dữ liệu trong bảng sẽ được cập nhật theo giá trị mới)

**c. Ví dụ minh họa:**

Hãy thực hiện hai tình huống cập nhật dữ liệu trên bảng NHANVIEN như sau:

**NHANVIEN** (MANV, HONV, TENLOT, TENNV, NGSINH, DIACHI, LUONG, PHAI, MA\_NQL, PHOG)

**Ví dụ 1**: Hãy tăng lương cho tất cả các nhân viên 10%

**UPDATE** NHANVIEN

**SET** LUONG = LUONG\*1.1

**Ví dụ 2**: Hãy tăng lương cho các nhân viên Nữ 10%

**UPDATE** NHANVIEN

**SET** LUONG = LUONG\*1.1

**WHERE** PHAI = ‘Nữ’

## 4.3. Truy vấn lồng

### 4.3.1. Phân loại:

Truy vấn lồng là những câu lệnh mà trong thành phần WHERE có chứa thêm một câu lệnh SELECT khác nữa. Câu lệnh này thường gặp khi dữ liệu cần thiết phải duyệt qua nhiều lần. Đây là một trong những vấn đề khó khăn nhất khi truy vấn dữ liệu.

* Các câu lệnh SELECT có thể lồng nhau ở nhiều mức
* Các câu truy vấn con trong cùng một mệnh đề WHERE được kết hợp bằng phép nối logic
* Câu truy vấn con thường trả về một tập các giá trị
* Mệnh đề WHERE của câu truy vấn cha
  + <biểu thức><so sánh tập hợp><truy vấn con>
  + So sánh tập hợp thường đi cùng với một số toán tử
    - IN, NOT IN
    - ALL
    - ANY hoặc SOME
  + Kiểm tra sự tồn tại
    - EXISTS
    - NOT EXISTS
* **Có 2 loại truy vấn lồng**
  + ***Lồng phân cấp***
    - Mệnh đề WHERE của truy vấn con không tham chiếu đến thuộc tính của các quan hệ trong mệnh đề FROM ở truy vấn cha.
    - Khi thực hiện, câu truy vấn con sẽ được thực hiện trước.
  + ***Lồng tương quan***
    - Mệnh đề WHERE của truy vấn con tham chiếu ít nhất một thuộc tính của các quan hệ trong mệnh đề FROM ở truy vấn cha.

Khi thực hiện, câu truy vấn con sẽ được thực hiện nhiều lần, mỗi lần **tương ứng với một bộ của truy vấn cha.**

### 4.3.2. Cú pháp

**SELECT**<danh sách các cột>

**FROM**<danh sách các bảng>

**WHERE**<so sánh tập hợp> (

**SELECT**<danh sách các cột>

**FROM**<danh sách các bảng>

**WHERE**<điều kiện>)

### 4.3.2. Ví dụ minh họa

**Ví dụ 1:** Tìm họ và tên của các nhân viên trực thuộc các phòng ban ở địa bàn TP. HCM.

Hướng dẫn giải:

*Bước 1*: Tìm tất cả các phòng có địa điểm ở TP. HCM

**SELECT**  MAPHG

**FROM**  DIADIEM\_PHONG

**WHERE**  DIADIEM = ‘TP. HCM’

*Bước 2*: Tìm nhân viên trực thuộc các phòng trong bước 1.

**SELECT**  HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE**  PHG **IN** ( **SELECT**  MAPHG

**FROM**  DIADIEM\_PHONG

**WHERE**  DIADIEM = ‘TP. HCM’)

**Ví dụ 2**: Tìm họ và tên của các nhân viên không tham gia vào đề án 322.

Hướng dẫn giải:

*Bước 1*: Tìm tất cả các nhân viên tham gia vào đề án 322.

**SELECT** MA\_NV

**FROM**  PHANCONG

**WHERE** MA\_DA = ‘322’

*Bước 2*: Tìm nhân viên không thuộc danh sách trong bước 1.

**SELECT** HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE** MANV **NOT IN** ( **SELECT** MA\_NV

**FROM**  PHANCONG

**WHERE** MA\_DA = ‘322’)

**Ví dụ 3**: Tìm những nhân viên có lương lớn hơn lương của ít nhất một nhân viên phòng 4

**Cách 1: Viết câu truy vấn dạng lồng phân cấp:**

*Bước 1*: Tìm lương của tất cả các nhân viên phòng số 4.

**SELECT** LUONG

**FROM** NHANVIEN

**WHERE** PHG = 4

*Bước 2*: Tìm nhân viên có mức lương lớn hơn ít nhất 1 mức lương trong bước 1.

**SELECT** HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE** LUONG > ANY ( **SELECT** LUONG

**FROM** NHANVIEN

**WHERE** PHG = 4)

**Cách 2:** Viết câu truy vấn dạng lồng tương quan:

**SELECT**  \*

**FROM** NHANVIEN NV1

**WHERE**  *EXISTS* (

**SELECT**  \*

**FROM** NHANVIEN NV2

**WHERE** NV2.PHG=4 AND NV1.LUONG>NV2.LUONG)

**Ví dụ 4:** Tìm những nhân viên có lương lớn hơn lương của tất cả các nhân viên phòng 4

*Bước 1*: Tìm lương của tất cả các nhân viên phòng số 4.

**SELECT** LUONG

**FROM** NHANVIEN

**WHERE** PHG = 4

*Bước 2*: Tìm nhân viên có mức lương lớn hơn tất cả các mức lương trong bước 1.

**SELECT** HONV, TENLOT, TENNV

**FROM** NHANVIEN

**WHERE** LUONG > ALL ( **SELECT** LUONG

**FROM** NHANVIEN

**WHERE** PHG = 4)

## 4.5. Một số vấn đề cần lưu ý khi thực hiện câu truy vấn

* **Toán tử IN:**
  + <tên cột> IN <câu truy vấn con>
  + Thuộc tính ở mệnh đề SELECT của truy vấn con phải có cùng kiểu dữ liệu với thuộc tính ở mệnh đề WHERE của truy vấn cha
* **Toán tử EXISTS**
  + Không cần có thuộc tính, hằng số hay biểu thức nào khác đứng trước
  + Không nhất thiết liệt kê tên thuộc tính ở mệnh đề SELECT của truy vấn con.
  + *Những câu truy vấn có = ANY hay IN đều có thể chuyển thành câu truy vấn có* ***EXISTS***
* **Mệnh đề GROUP BY**
  + Các thuộc tính trong mệnh đề SELECT (trừ những thuộc tính trong các hàm kết hợp) phải xuất hiện trong mệnh đề GROUP BY
* **Mệnh đề HAVING**
  + Sử dụng các hàm kết hợp trong mệnh đề SELECT để kiểm tra một số điều kiện nào đó
  + Chỉ kiểm tra điều kiện trên nhóm, không là điều kiện lọc trên từng bộ
  + Sau khi gom nhóm điều kiện trên nhóm mới được thực hiện.
* **Thứ tự thực hiện câu truy vấn có mệnh đề GROUP BY và HAVING**

(1) Chọn ra những dòng thỏa điều kiện trong mệnh đề WHERE

(2) Những dòng này sẽ được gom thành nhiều nhóm tương ứng với mệnh đề GROUP BY

(3) Áp dụng các hàm kết hợp cho mỗi nhóm

(4) Bỏ qua những nhóm không thỏa điều kiện trong mệnh đề HAVING

(5) Rút trích các giá trị của các cột và hàm kết hợp trong mệnh đề SELECT

# BÀI TẬP CHƯƠNG 4

## CÂU HỎI

**Câu hỏi 1:** Khi thực hiện truy vấn, các biểu thức điều kiện (nếu có) thường nằm trong hai mệnh đề nào? Hãy phân loại các điều kiện của từng mệnh đề.

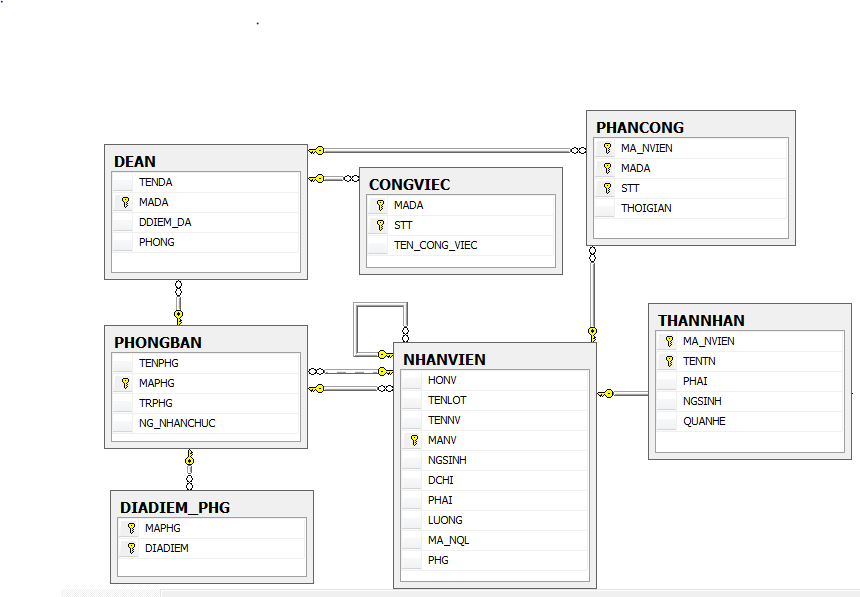
**Câu hỏi 2:** Trong quá trình xây dựng câu truy vấn, các hàm kết hợp (MIN, MAX, SUM, AVG, …) thường xuất hiện trong các mệnh đề nào?

**Câu hỏi 3:** Hãy nêu trình tự thực hiện công việc truy vấn của các mệnh đề.

**Câu hỏi 4:** Hãy trình bày các lưu ý cần thiết khi sử dụng mệnh đề GROUP BY trong mệnh đề truy vấn.

## BÀI TẬP

Cho lược đồ cơ sở dữ liệu ‘QUẢN LÝ ĐỀ ÁN CÔNG TY’ như sau:

Với cơ sở dữ liệu cho trên, hãy thực hiện các yêu cầu truy vấn sau:

1. Cập nhật lại dữ liệu cho các bảng trong CSDL đã tạo ở chương 3.
2. Cho biết thông tin chi tiết của các nhân viên.
3. Cho biết tên các nhân viên ở phòng số 5.
4. Cho biết tên phòng mà nhân viên tên Khoa trực thuộc
5. Cho biết tên của trường phòng phòng nghiên cứu.
6. Cho biết mức lương trung bình của từng phòng ban.
7. Cho biết tên các phòng có mức lương trung bình cao nhất.
8. Cho biết tên các nhân viên không tham gia đề án nào.
9. Cho biết tên các nhân viên tham gia tất cả các đề án.
10. Cho biết tên các nhân viên tham gia trên 3 đề án ở TPHCM.
11. Cho biết tên nhân viên có mức lương cao nhất ở phòng quản lý.
12. Hãy xóa các đề án ở Vũng Tàu.
13. Tăng lương cho các nhân viên phòng quản lý 10%.
14. Đối với các nhân viên tham gia trên 3 đề án, hãy tăng 10% lương cho họ.

# CHƯƠNG 5 LẬP TRÌNH T – SQL

# MỤC ĐÍCH VÀ YÊU CẦU

**Mục đích**:

Chương này nhằm cung cấp cho sinh viên những kiến thức chung về:

* Cách khai báo biến, lựa chọn kiểu dữ liệu thích hợp, …
* Sử dụng một số câu lệnh cơ bản và các hàm – thủ tục có sẵn trong SQL Server.
* Cú pháp các lệnh trong ngôn ngữ T - SQL.

**Yêu cầu**:

Sau khi học xong chương này sinh viên có khả năng:

* Phân tích các tình huống theo yêu cầu đặt ra;
* Nắm vững cú pháp, phân tích vấn đề và vận dụng thành thạo các câu lệnh để xử lý dữ liệu theo yêu cầu.
* Hiểu rõ công dụng của các câu lệnh, hàm, thủ tục hỗ trợ sẵn trong SQL Server.

# NỘI DUNG

## 5.1. Tổng quan về ngôn ngữ T – SQL

### 5.1.1. Giới thiệu

T – SQL (Transact SQL) là ngôn ngữ SQL mở rộng dựa trên SQL chuẩn của ISO (International Organization for Standardization) và ANSI (American National Standards Institute). Trong SQL, ngôn ngữ T – SQL được chia thành 3 nhóm chính:

* Data Definition Language (DDL) – ngôn ngữ định nghĩa dữ liệu;
* Data Control Language (DCL) – ngôn ngữ điều khiển dữ liệu;
* Data Manipulation Language (DML) – ngôn ngữ thao tác dữ liệu.

Trong đó:

**Data Definition Language (DDL)** – ngôn ngữ định nghĩa dữ liệu: Là tập hợp các lệnh chuyên dùng để quản lý các thuộc tính của một CSDL như: định nghĩa các dòng (bộ) hoặc cột (thuộc tính) của một bảng (quan hệ) hay vị trí data file của một CSDL. Bao gồm: Create *object\_Name,* Alter *object\_Name,* Drop *object\_Name.*

*Lưu ý*: *object\_Name* có thể là một table, view, stored procedure, indexes...

**Data Control Language (DCL)** – ngôn ngữ điều khiển dữ liệu: Là tập hợp các lệnh dùng quản lý các quyền truy cập trên từng object (table, view, stored procedure,...). Bao gồm: Deny, Revoke, Grant.

**Data Manipulation Language (DML)** – ngôn ngữ thao tác dữ liệu: Là tập hợp các lệnh cho phép người dùng thao tác trên CSDL như: Select, Update, Delete, Insert.

Trong chương này, để để minh họa cho các thao tác trên thủ tục, nhóm tác giả sử dụng một phần cơ sở dữ liệu “Quản lý đề án công ty” như sau:

**NHANVIEN** (MANV, HONV, TENLOT, TENNV, PHAI, DIACHI, LUONG, MA\_NQL, NGAYSINH, PHG)

**PHONGBAN** (MAPHG, TENPHG, TRUONGPHONG, DIADIEM\_PHG)

**DEAN** (MADA, TENDA, PHONG)

**PHANCONG** (MANV, SODA, THOIGIAN)

### 5.1.2. Các thành phần cơ bản trong cú pháp của ngôn ngữ T - SQL

Các thành phần cơ bản cấu thành nên cú pháp trong ngôn ngữ T – SQL bao gồm: định danh (indentifier), biến (variables), hàm (functions), biểu thức (expressions), kiểu dữ liệu (data type), lệnh và cấu trúc điều khiển (control of flow), chú thích (comment), …

**Identifiers:** là tên của các CSDL object (table, view, stored procedure, index.....), được dùng để xác định một object. PHONGBAN, MAPB, TENPB là những identifiers trong ví dụ bên dưới.

**Ví dụ:**

CREATE TABLE PHONGBAN

(

MAPHG NVARCHAR(12) PRIMARY KEY,

TENPHG NVARCHAR(50)

)

Có hai loại Identifiers: **Regular Identifier** và **Delimited Identifier.** Trong đó:

**Regular Identifier:** khai báo thông thường theo quy tắc đặt tên định danh (không khoảng trắng, không có dấu tiếng việt, không trùng với tên từ khóa)

**Delimited Identifier:** có thể có khoảng trắng, có thể dùng dấu tiếng việt, có thể trùng với từ khóa trong SQL Server. Đối với Delimited Identifier khi khai báo bắt buộc có dấu "" hay dấu [] để ngăn cách.

**Ví dụ:**

SELECT \* FROM [My Table]

WHERE [Order] = 10

Trong ví dụ trên từ My Table có chữ Table trùng với từ khóa đồng thời giữa hai chữ có khoảng trắng, tương tự Order trùng với keyword Order 🡪 bắt buộc phải đặt trong cặp dấu ngoặc vuông [].

**Variables (Biến)**

Biến trong T-SQL cũng có chức năng tương tự như trong các ngôn ngữ lập trình khác nghĩa là cần khai báo trước loại dữ liệu trước khi sử dụng. Biến được bắt đầu bằng dấu @ (Ðối với các global variable thì có hai dấu @@)

**Ví dụ:**

USE QLDEAN

DECLARE @MaPhong SMALLINT

SET @MaPhong = 3

SELECT \*

FROM PHONGBAN

WHERE MAPHG = @MaPhong + 1

**Functions (Hàm)**

Có 2 loại hàm: built-in và user-defined.

* Các hàm **Built-In**(được định nghĩa, xây dựng và tích hợp sẵn trong hệ quản trị cơ sở dữ liệu SQL Server khi người dùng cài đặt). Được chia làm 3 nhóm:
* **Rowset Functions**: trả về một object và được xem như một table. Ví dụ như hàm OPENQUERY sẽ trả về một recordset và có thể đứng vị trí của một table trong câu lệnh Select.
* **AggregateFunctions**: tính toán và thao tác trên các giá trị, kết quả trả về một giá trị đơn hay là các giá trị tổng hợp. Ví dụ như hàm AVG sẽ trả về giá trị trung bình của một cột.
* **Scalar Functions**: thao tác trên một giá trị đơn và trả về một giá trị đơn. Trong loại này lại chia làm nhiều loại nhỏ như các hàm về toán học, về thời gian, xử lý kiểu dữ liệu String....Ví dụ như hàm MONTH('2002-09-30') sẽ trả về tháng 9.
* Các hàm **User-Defined**(do người dùng tự định nghĩa, xây dựng và gắn liền với CSDL cài đặt tương ứng. Được tạo ra bởi câu lệnh CREATE FUNCTION và phần body thường được nằm trong cặp lệnh BEGIN...END) cũng được chia làm các nhóm như sau:
* **Scalar Functions**: trả về một giá trị đơn bằng câu lệnh RETURNS.
* **Table Functions**: kết quả trả về một table.

**Data Type (Kiểu dữ liệu)**

Các kiểu dữ liệu trong SQL Server đã được đề cập chi tiết trong phần tạo và quản lý bảng (Chương 1 – Tổng quan về hệ quản trị cơ sở dữ liệu SQL Server 2008).

**Expressions (Biểu thức)**

Các Expressions có dạng **Identifier** + **Operators** (như +,-,\*,/,=...) + **Value**

**Các thành phần Control-Of Flow**

Bao gồm: BEGIN...END, BREAK, CONTINUE, GOTO, IF...ELSE, RETURN, WHILE....

**Comments (Chú thích)**

T-SQL dùng dấu -- để đánh dấu phần chú thích cho câu lệnh đơn và dùng **/\*...\*/** để chú thích cho một nhóm lệnh.

## 5.2. Biến và cách sử dụng biến trong T-SQL

### 5.2.1. Khái niệm biến

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

Biến có thể được dùng để:

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

### 5.2.2. Phân loại biến theo phạm vi hoạt động

Phân loại theo phạm vi hoạt động, cũng như một số ngôn ngữ lập trình khác, T – SQL có 2 loại biến: biến cục bộ (local) và biến toàn cục (global).

#### 5.2.2.1. Global variable (biến toàn cục)

Được SQL Server cung cấp sẵn và người dùng có thể sử dụng bất cứ khi nào cần mà không cần khai báo (được xem như là những hàm chuẩn của SQL Server). Từ SQL Server 7.0 biến Global được định nghĩa như là hàm hệ thống. Mỗi kết nối sẽ tạo ra một session, SQL Server tạo ra sẵn một số biến có sẵn rất tiện ích trong việc lập trình và quản trị hệ thống. Các biến này không có kiểu, tên bắt đầu @@. Một số biến (hàm hệ thống) thường dùng như:

|  |  |
| --- | --- |
| @@VERSION | Xem phiên bản của SQL Server và hệ điều hành |
|  | SELECT @@VERSION |
| @@TRANCOUNT | Xem có bao nhiêu transaction đang mở |
|  | IF (@@TRANCOUNT > 0)  BEGIN  RAISERROR('Task cannot be executed within a transaction.', 10, 1)  RETURN  END |
| @@ROWCOUNT | Trả về số dòng bị ảnh hưởng đối với lệnh thực thi gần nhất |
|  | **Ví dụ 1:**  USE QLDEAN  UPDATE NHANVIEN SET TENNV = 'Anh'  WHERE HONV = 'Nguyen'  IF (@@ROWCOUNT = 0)  BEGIN  PRINT 'Warning: No rows were updated'  RETURN  END  **Ví dụ 2:**  UPDATE NHANVIEN  SET LUONG = LUONG\*1.1  WHERE PHG = 5  PRINT @@ROWCOUNT |
| @@IDENTITY | trả về số Identity phát sinh sau cùng |
|  | CREATE TABLE TABLE\_HD  ( mahd int Identity Primary Key,  Ghichu varchar(20) )  CREATE TABLE TABLE\_CTHD  ( Mahd int, Masp char(10), Soluong int)  declare @maso Int  Insert into Table\_HD Values ('Mau tin 1')  Insert into Table\_HD Values ('Mau tin 2')  set @maso= @@IDENTITY  Insert into Table\_CTHD  values (@maso, 'sp001',5)  Insert into Table\_CTHD  values (@maso, 'sp002',10)  -- Kiểm tra  SELECT \* FROM Table\_HD  SELECT \* FROM Table\_CTHD |
| @@ERROR | Trả về lỗi số (STT lỗi) của lệnh sau cùng mà SQL thực thi, là 0 có nghĩa là câu lệnh thực thi hoàn thành. |
| @@FETCH\_STATUS | Trả về trạng thái của lệnh Fetch của biến con trỏ có thành công hay không (0: Thành công, -1: bị lỗi hoặc vượt quá phạm vi; -2: Thất bại) |

#### 5.2.2.2. Local variable (biến cục bộ)

Biến phải được khai báo trước khi người dùng muốn sử dụng.

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

### 5.2.3. Khai báo và gán giá trị cho biến

**Cú pháp khai báo biến:**

**DECLARE @Var\_name Var\_type**

*Trong đó*:

* **DECLARE**: từ khóa bắt buộc khi khai báo biến (có thể viết hoa hoặc viết thường)
* **Var\_name**: tên biến được khai báo theo quy tắc đặt tên định danh
* **Var\_type**: tên kiểu dữ liệu

**Gán giá trị cho biến:**

Biến mới khai báo mặc định được gán giá trị là NULL.

Để gán giá trị cho biến ta có thể dùng lệnh SET hoặc dùng câu lệnh SELECT

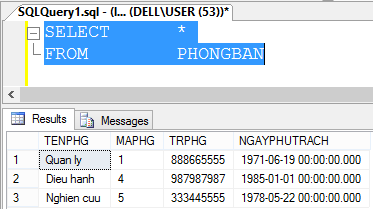
**Cú pháp gán giá trị cho biến:**

SET @Var\_name = *expression*

SELECT { @*Var\_name* = *expression* } [,...*n* ]

**Cho quan hệ NHANVIEN, PHONGBAN có thể hiện như sau:**

****



**Ví dụ 1:**

USE QLDEAN

DECLARE @MaPhong SMALLINT

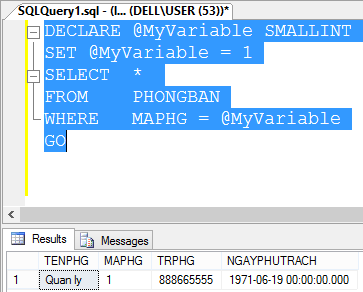
SET @MaPhong = 1

SELECT \*

FROM NHANVIEN

WHERE PHG = @MaPhong

**Kết quả:**

****

**Ví dụ 2:**

DECLARE @MyVariable SMALLINT

SET @MyVariable = 1

GO – điểm kết thúc một lô batch.

-- @MyVariable đã vượt quá phạm vi và nó đã hết tồn tại

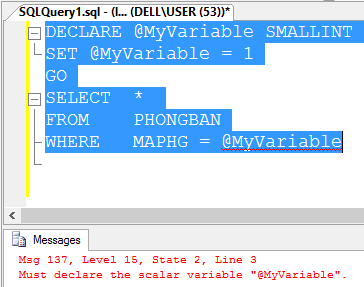
-- Câu lệnh SELECT sau sẽ nhận lỗi sai cú pháp bởi vì nó không tham chiếu được biến @MyVariable.

SELECT \*

FROM PHONGBAN

WHERE MAPHG = @MyVariable

**Kết quả:**



**Ví dụ 3:**

USE QLDEAN

GO

-- Khai báo 2 biến.

DECLARE @TENNV NVARCHAR (20)

DECLARE @DIACHI NVARCHAR (30)

-- Gán giá trị cho 2 biến.

SET @TENNV = 'Khoa'

SET @DIACHI = 'TP.HCM'

-- Dùng chúng trong mệnh đề WHERE của lệnh SELECT.

SELECT HONV, TENLOT, TENNV

FROM NHANVIEN

WHERE TENNV = @TENNV

OR DCHI = @DIACHI

GO

**Ví dụ 4:**

USE QLDEAN

GO

-- Khai báo 1 biến

DECLARE @Luong INT

-- Gán giá trị biến bằng câu lệnh Select

SELECT @Luong = MAX (LUONG)

FROM NHANVIEN

GO

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

**Ví dụ 5:**

DECLARE @Luong int

SELECT @Luong = LUONG

FROM QLDEAN.NHANVIEN

**Ví dụ 6:**

USE QLDEAN

GO

DECLARE @Luong INT

SELECT @Luong = MAX(LUONG)

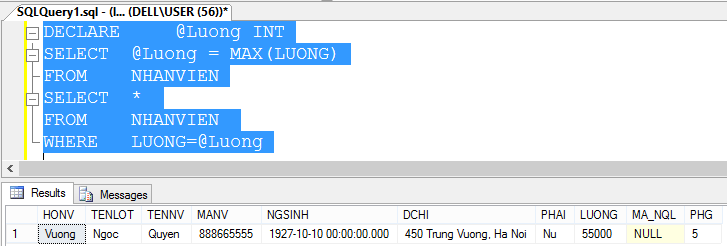
FROM NHANVIEN

SELECT \*

FROM NHANVIEN

WHERE LUONG=@Luong

**Kết quả:**



## 5.3. Các cấu trúc điều khiển

Tương tự một số ngôn ngữ lập trình khác, ngôn ngữ T-SQL cũng cung cấp một số cấu trúc điều kiển cơ bản để hỗ trợ người dùng thực thi một hoặc một khối lệnh dựa trên kết quả của các phép so sánh.

Một số cấu trúc điều khiển thường dùng bao gồm: cấu trúc điều kiện **IF … ELSE**, cấu trúc lặp **WHILE,** cấu trúc lựa chọn **CASE.**

### 5.3.1. Cấu trúc IF … ELSE …

Là một cấu trúc điều kiện, cho phép thực thi một hoặc nhiều lệnh tùy thuộc vào một hoặc nhiều điều kiện nào đó.

**Cú pháp**:

IF <LOGIC\_EXPRESSION>

<STATEMENT\_SQL>

[ ELSE

< STATEMENT\_SQL >]

Trong đó:

* Điều kiện: là một biểu thức logic, có giá trị *True* hoặc *False.*Tùy thuộc vào *điều kiện,* một trong hai khối lệnh sẽ được thực thi.
* Lệnh: nếu nhiều lệnh (khối lệnh) yêu cầu đồng thời được thực thi khi thỏa mãn điều kiện thì các lệnh này phải đặt trong cặp từ khóa BEGIN … END

**Ví dụ 1:** Kiểm tra xem trong bảng NHANVIEN có các nhân viên địa chỉ ở Hà Nội không?

USE QLDEAN

IF ((SELECT COUNT (\*) FROM NHANVIEN

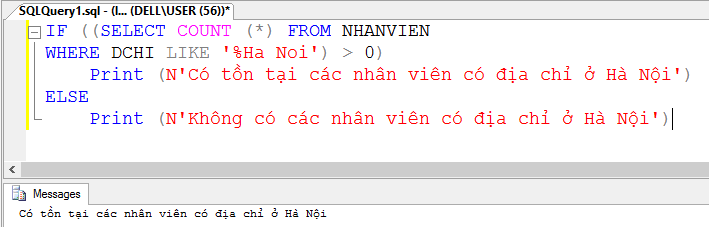
WHERE DCHI LIKE '%Ha Noi') > 0)

Print (N'Có tồn tại các nhân viên có địa chỉ ở Hà Nội’)

ELSE

Print (N'Không có các nhân viên có địa chỉ ở Hà Nội')

**Kết quả:**



**Ví dụ 2:** Kiểm tra xem trong NHANVIEN có các nhân viên có họ bắt đầu là 2 ký tự NG và có lương nằm trong khoảng từ 3000 đến 5000 hay không? Nếu có hãy xuất ra tên các nhân viên tìm được.

USE QLDEAN

GO

DECLARE @msg varchar (255)

IF ( SELECT COUNT (MANV) FROM NHANVIEN

WHERE HONV LIKE 'NG%' AND LUONG BETWEEN 30000 AND 50000) > 0

BEGIN

SET NOCOUNT ON

SET @msg = 'Có NV có lương từ 30000 đến 40000. Bao gồm:'

PRINT @msg

SELECT HONV+‘ ’+TENLOT+‘ ’+ TENNV

FROM NHANVIEN

WHERE HONV LIKE 'NG%' AND LUONG BETWEEN 30000 AND 40000

END

ELSE

BEGIN

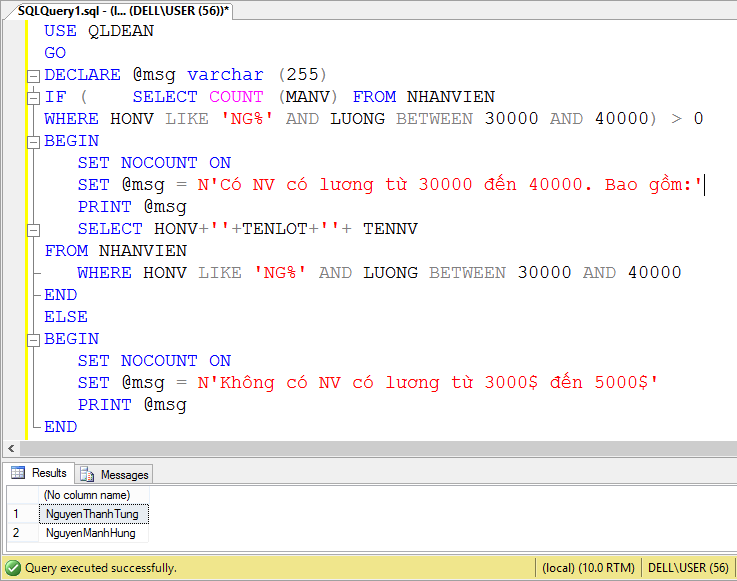
SET NOCOUNT ON

SET @msg = 'Không có NV có lương từ 3000$ đến 5000$'

PRINT @msg

END

**Kết quả:**

****

### 5.3.2. Cấu trúc WHILE

Là cấu trúc lặp, hành động lặp sẽ được thực hiện cho đến khi biểu thức điều kiện (*Boolean\_expression*) trong While mang giá trị False. Biểu thức điều kiện lặp có thể là một câu SELECT.

**Cú pháp**

WHILE <Boolean\_expression>

BEGIN

sql\_statement | statement\_block

[BREAK]

sql\_statement | statement\_block

[CONTINUE]

END

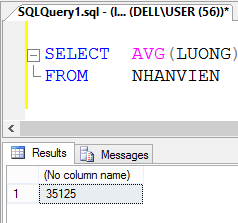
**Trong đó:**

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

**Lưu ý:** thông thường 2 từ khóa Break và Continue phải nằm trong trong cấu trúc If … Else …

**Ví dụ:** Trong khi lương trung bình vẫn còn nhỏ hơn 40000 thì cập nhật các lương tăng lên 10% so với lương cũ.

**Ban đầu:**



USE QLDEAN

GO

WHILE (SELECT AVG (LUONG) FROM NHANVIEN) < 30000

BEGIN

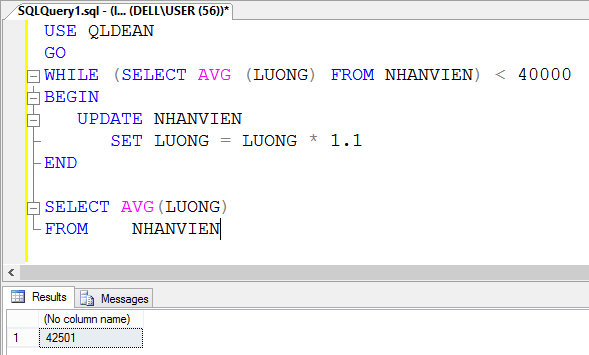
UPDATE NHANVIEN

SET LUONG = LUONG \* 1.1

SELECT MAX (LUONG) FROM NHANVIEN

END

**Sau điều chỉnh:**



### 5.3.3. Cấu trúc CASE

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

**Cú pháp:**

CASE

WHEN <Boolean\_expression>THEN <Result\_Expression\_1>

ELSE < Result\_Expression\_2>

END

Trong đó:

* Boolean\_expression: điều kiện của biểu thức WHEN
* Result\_Expression\_1: thực hiện biểu thức khi điều kiện trong WHEN thỏa
* Result\_Expression\_2: thực hiện biểu thức khi với điều kiện trong WHEN không thỏa

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

DECLARE @a AS INT, @b AS INT, @ketqua as NVARCHAR (30)

SET @a=3

SET @b=5

SET @ketqua = CASE

WHEN @a<@b THEN N'A nhỏ hơn B'

-- When chỉ dùng trong case

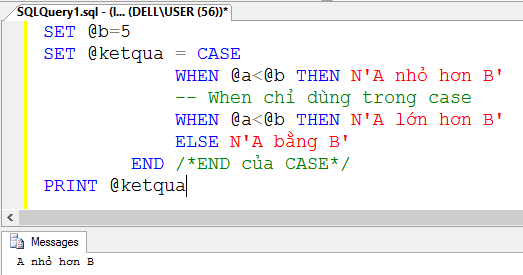
WHEN @a<@b THEN N'A lớn hơn B'

ELSE N'A bằng B'

END /\*END của CASE\*/

PRINT @ketqua

**Kết quả:**

****

**Ví dụ 2:** Dựa vào lương của các nhân viên cho biết lương đó như thế nào?

Use QLDEAN

SELECT  TENNV, LUONG,'XẾP LOẠI'=

CASE

     WHEN LUONG < 30000 THEN ‘Lương thấp'

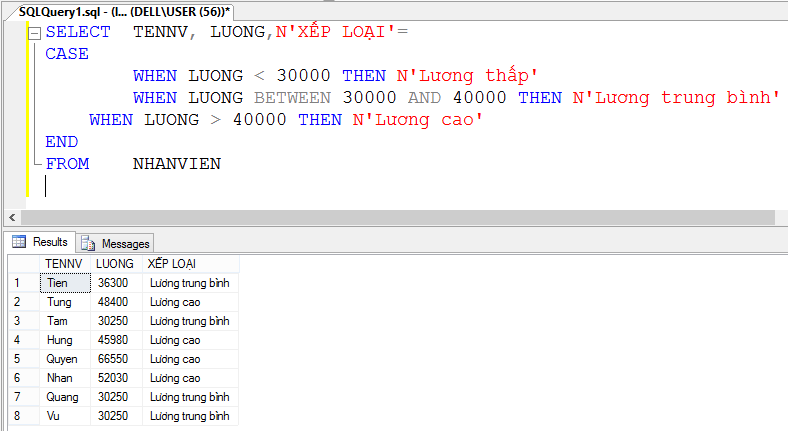
     WHEN LUONG BETWEEN 30000 AND 40000 THEN ‘Lương trung bình'

WHEN LUONG > 40000 THEN 'Lương cao'

END

FROM  NHANVIEN

**Kết quả:**



### 5.3.4. Một số lệnh điều khiển khác

Lệnh RETURN

RETURN [integer\_expression]

Khi gặp câu lệnh Return, toàn bộ quá trình xử lý sẽ kết thúc. Trong một số trường hợp, người dùng có thể sử dụng lệnh RETURN trong cài đặt thủ tục để thủ tục có thể trở thành hàm như các ngôn ngữ khác.

Lệnh WAITFOR

Là câu lệnh trong SQL Server có tác dụng như một chỉ thị giúp cho SQL Server tạm dừng một thời gian trước khi xử lý tiếp các phát biểu sau đó.

WAITFOR DELAY '*time*' | TIME '*time*'

Trong đó:

* ‘Time’: được viết thao dạng hh:mm:ss, tối đa là 24 giờ
* DEPLAY ‘Time’: Hệ thống tạm dừng trong khoảng thời gian ‘Time’.
* TIME ‘Time’: Hệ thống được tạm dựng đến thời gian ‘Time’ chỉ ra.

**Ví dụ:**

WAITFOR ‘02:10’

**Lệnh RAISERROR**

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

RAISERROR (*msg\_id* | *msg\_str*, *security,state*[*argument* [,...*n* ] ])

[WITH *option* [,...*n* ] ]

Trong đó:

* *msg\_id:* là mã thông báo, nó được lưu trong bảng sysmessage. Mã thông báo của người dùng định nghĩa phải được bắt đầu từ trên 50000
* *msg\_str* : Nội dung thông báo, tối đa 400 ký tự.

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

Loại ký tự là d, i, o, x, X, hoặc u

|  |  |
| --- | --- |
| **Các ký tự** | **Mô tả** |
| d hoặc I | Biểu hiện là số nguyên (integer) |
| O | Octal không dấu |
| P | Con trỏ |
| S | Chuỗi |
| U | So nguyên không dấu |
| x or X | Hexadecimal không dấu |

**Tham số loại ký tự**

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

* *security:* Độ nghiêm ngặt của thông báo

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

* Mức lỗi **10** là lỗi về thông tin và biểu thị nguyên nhân do thông tin nhập vào.
* Mức lỗi từ **11 đến 16** thì thông thường là do các user.
* Mức từ **17 đến 25** do lỗi phần mềm hoặc phần cứng. Người dùng nên báo cho nhà quản trị hệ thống bất cứ khi nào sự cố xảy ra. Nhà quản trị hệ thống phải giải quyết sự cố đó và theo dõi chúng thường xuyên. Khi mức lỗi 17, 18, 19 xảy ra, người dùng có thể tiếp tục làm việc mặc dù họ không thể thực thi lệnh đặc biệt.
* **Mức lỗi 17:** Những thông báo này cho biết rằng câu lệnh nguyên nhân SQL Server cạn kiệt tài nguyên hoặc vượt quá tập giới hạn bởi nhà quản trị.
* Người quản trị hệ thống nên giám sát tất cả các sự cố mà được phát ra mức trầm trọng từ **17 đến 25** và in ra giải thích lỗi mà bao gồm các thông tin để quay lại từ lỗi.

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

* Mức lỗi từ **20 đến 25** chỉ ra sự cố hệ thống. Đó là lỗi không tránh được, mà có nghĩa là tiến trình không còn đang chạy. Tiến trình tê liệt trước khi nó dừng, ghi nhận thông tin về cái gì xảy ra, và sau đó kết thúc.
* *State*: Là một số nguyên tùy ý từ 1 đến 127 mà nó mô tả thông tin diễn giải về trạng thái lỗi.
* *Argument:* Là tham số dùng trong việc thay thế cho biến để định nghĩa thông báo lỗi hoặc thông báo tương ứng với mã lỗi msg\_id. Có thể không hoặc có nhiều tham số. Tuy nhiên, không được quá 20. Mỗi tham số thay thế có thể là một biến local hoặc bất kỳ một trong các kiểu dữ liệu int, char, varchar, binary, varbinary. Các kiểu khác không được cung cấp.

## 5.4. Batch

Batch là một tập các phát biểu (tập các lệnh yêu cầu thực thi liên tiếp) trong ngôn ngữ T-SQL nằm liên tiếp, kết thúc bởi phát biểu GO và được biên dịch đồng thời bởi SQL Server.

Việc gửi đến server một tập lệnh SQL thay vì từng lệnh SQL làm tăng hiệu năng của server...

**Ví dụ:**

USE QLDEAN

GO /\* Signals the end of the batch \*/

CREATE VIEW NHANVIEN

AS

SELECT \*

FROM NHAVIEN

GO /\* Signals the end of the batch \*/

SELECT \*

FROM PHONGBAN

GO /\* Signals the end of the batch \*/

**Lưu ý:**

* Các phát biểu trong 1 batch được biên dịch thành một nhóm.
* Nếu một trong phát biểu của batch bị lỗi thì batch cũng xem như lỗi.
* Các phát biểu CREATE bị ràng buộc trong một batch đơn, tức trong batch đó chỉ có phát biểu CREATE. Các phát biểu đó là: CREATE DATABASE, CREATE TABLE

## 5.5. Transact – SQL Scripts:

Một script là tập các câu lệnh T-SQL được lưu trữ trong một file, gồm 1 hoặc nhiều batch

*Script được sử dụng nhằm*:

* Lưu giữ bản sao các bước tạo CSDL trên server.
* Di chuyển các câu lệnh từ máy này sang máy khác.

# BÀI TẬP CHƯƠNG 5

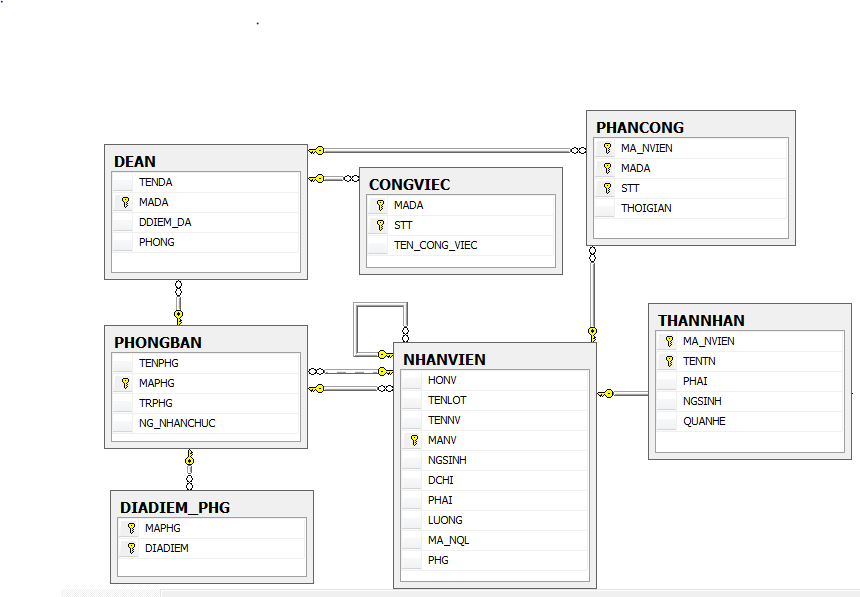
## CÂU HỎI

Câu hỏi 1: Hãy phân loại biến theo phạm vi hoạt động trong ngôn ngữ T-SQL.

Câu hỏi 2: Hãy liệt các thành phần cơ bản trong trong cú pháp ngôn ngữ T-SQL.

## BÀI TẬP

Cho lược đồ cơ sở dữ liệu ‘QUẢN LÝ ĐỀ ÁN CÔNG TY’ như sau:



Với cơ sở dữ liệu cho trên, hãy viết lệnh thực hiện các yêu cầu truy vấn sau:

1. Cho biết có bao nhiêu nhân viên thuộc phòng nghiên cứu có mức lương trên 30000, nếu không có nhân viên nào hãy xuất ra thông báo.
2. Trong khi nhân viên có mức lương cao nhất ở phòng quản lý chưa đạt 50000, hãy tăng lương cho nhân viên này 10%, hãy xuất ra mức lương sau khi tăng.
3. Hãy xuất ra thông tin các nhân viên đến tuổi về hưu. Biết rằng quy định về hưu đối với nam là 60 và đối với nữ là 50.
4. Cho biết có bao nhiêu nhân viên có mức lương trên 50000.
5. Trong khi mức lương trung bình của các nhân viên phòng nghiên cứu dưới 30000, hãy tăng cho mỗi người thêm 10%. Xuất ra mức lương trung bình mới sau khi tăng.

# CHƯƠNG 6 THỦ TỤC – HÀM

# MỤC ĐÍCH VÀ YÊU CẦU

**Mục đích**:

Chương này nhằm cung cấp cho sinh viên những kiến thức chung về:

* Cú pháp khai báo chung và cài đặt hàm – thủ tục
* Cách khai báo tham số và sử dụng tham số cài đặt hàm – thủ tục
* Phân loại được các loại hàm, các loại thủ tục, cú pháp khai báo và cách cài đặt cho từng loại.
* Biết cách hiệu chỉnh, xóa thủ tục – hàm đã tạo.

**Yêu cầu:**

Sau khi học xong chương này sinh viên có khả năng:

* Phân tích các tình huống theo yêu cầu đặt ra;
* Nắm vững cú pháp, phân tích vấn đề và vận dụng thành thạo các câu lệnh để xử lý dữ liệu theo yêu cầu.
* Hiểu rõ công dụng của các câu lệnh, hàm, thủ tục hỗ trợ sẵn trong SQL Server.

# NỘI DUNG

## 6.1. Thủ tục (Stored Procedure - SP)

### 6.1.1. Tổng quan về thủ tục

#### 6.1.1.1. Khái niệm

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

Một thủ tục có thể được gọi thực thi hoặc tự động thực khi SQL Server khởi động.

#### 6.1.1.2. Đặc điểm và tính năng

* Thủ tục trong SQL Server được lưu trữ tại server khi nó được tạo ra. Vì vậy, khi thủ tục được thi hành thì nó được chạy tại Server. Có thể gọi thủ tục chạy bằng một lệnh đơn giản và trong thủ tục có thể chứa rất nhiều lệnh của T-SQL.
* Trước khi thủ tục được tạo, SQL Server sẽ kiểm tra tính đúng đắn của các cú pháp lệnh. Nếu không có lỗi về cú pháp thì thủ tục được tạo, tên của thủ tục được lưu trong bảng hệ thống **SysObjects** và nội dung được lưu trong bảng hệ thống **SysCommanes**.
* Trước khi thủ tục được thực thi, một kế hoạch thực thi (Execution plain) được tạo ra và thủ tục được biên dịch. Từ đó trở về sau tiến trình dịch thủ tục nhanh hơn bởi vì SQL Server sẽ không kiểm tra tính đúng đắng của các câu lệnh nữa, chỉ tạo lại execution plan và biên dịch lại thủ tục.
* Security (an toàn): thủ tục có một đặc tính quan trọng là nó có thể được nâng cao an toàn thông tin thông qua isolation (cô lập) hoặc encryption (mã hóa). Người dùng CSDL có thể được cho quyền thực thi thủ tục nhưng sẽ không có quyền trực tiếp truy xuất các đối tượng của thủ tục. Một thủ tục có thể được mã hóa ngay khi được tạo hoặc chỉnh sửa vì thế người sử dụng không thể đọc được các câu lệnh trong thủ tục.
* SP thi hành nhanh vì: SQL Server sau khi phân tích và biên dịch SP sẽ giữ execution plans trong procedure cache. Các lần gọi sử dụng SP sau đó sẽ sử dụng lại execution plans đã có.
* Khi viết ứng dụng, gọi SP sẽ dễ bảo trì lệnh SQL trong SP hơn.
* SP cho phép thực thi các lệnh SQL theo gói (batch) Và gửi trả tập kết quả cùng một lúc, làm giảm lưu thông trên mạng.
* Việc sử dụng SP sẽ tiết kiệm chi phí và thời gian cho người lập trình khi muốn thực hiện cùng một tính năng cho các đối tượng khác nhau. Ngoài ra, việc cài đặt và sử dụng SP giúp người lập trình dễ dàng quản lý lệnh SQL khi có nhu cầu cải tiến hoặc khắc phục sự cố.

#### 6.1.1.3. Phân loại thủ tục

* **System sp:** được lưu trữ trong CSDL master và được đặt tên với tiếp đầu ngữ là **sp**. Chúng đóng vai trò khác nhau của các tác vụ được cung cấp trong SQL Server.
  + Ví dụ: Sp\_help, Sp\_helpConstraint, ….
* **Local sp:** được lưu trữ trong các CSDL của người dùng, nó thực thi các tác vụ (Task) trong CSDL chứa nó. Một Local sp có thể được người sử dụng tạo hoặc từ các sp hệ thống.
* **Temporary sp:** giống như là một local sp, nhưng nó chỉ hiện hữu cho đến khi kết nối tạo ra nó bị đóng. Nó được nằm trong CSDL TempDB. Có 3 loại temporary sp: local (private), Global, sp tạo trực tiếp trong TempDB.
* **Extended sp:** Là một thủ tục được tạo từ các ngôn ngữ lập trình khác (không phải SQL Server) và nó được triển khai tính năng của một thủ tục trong SQL Server. Các thủ tục này có tên với tiếp đầu ngữ là sp.
* **Remote sp:** là một thủ tục được gọi thực thi từ một server từ xa.

### 6.1.2. Thao tác trên thủ tục

Trong chương này, để để minh họa cho các thao tác trên thủ tục, nhóm tác giả sử dụng một phần cơ sở dữ liệu “Quản lý đề án công ty” như sau:

**NHANVIEN** (MANV, HONV, TENLOT, TENNV, PHAI, DIACHI, LUONG, MA\_NQL, NGAYSINH, PHG)

**PHONGBAN** (MAPHG, TENPHG, TRUONGPHONG, NGAY\_NHAMCHUC, DIADIEM\_PHG)

**DEAN** (MADA, TENDA, PHONG)

**PHANCONG** (MANV, SODA, THOIGIAN)

#### 6.1.2.1. Tạo và thực thi thủ tục

Tùy theo mục đích và yêu cầu mà người dùng xây dựng thủ tục phù hợp.

**Cú pháp tạo thủ tục:**

**CREATE PROCEDURE|PROC Procedure\_name** [<parameter>[, …n][ OUTPUT ]]

**[WITH option]**

**AS**

**sql\_statement [...]**

Trong đó:

* **[<parameter>[, …n]]**: Các tham số cách nhau bởi dấu phẩy. Khai báo của mỗi một tham số tối thiểu phải bao gồm hai phần:tên tham số được bắt đầu bởi dấu @, kiểu dữ liệu của tham số;
* **[OUTPUT|OUT]**: Đối với các thủ tục trả về giá trị thông qua kết quả của tham số thì đặt từ khóa OUTPUT|OUT sau tham số trong quá trình cài đặt và gọi thực thi thủ tục 🡪 sẽ đề cập chi tiết hơn trong phần 1.3 cùng chương.
* **Option**: có 2 tùy chọn:
* Nếu tuỳ chọn WITH RECOMPILE được chỉ định, thủ tục sẽ được dịch lại mỗi khi được gọi
* Thủ tục sẽ được mã hoá nếu tuỳ chọn WITH ENCRYPTION được chỉ định.
* **Sql\_statement:** Tập hợp các câu lệnh sử dụng trong nội dung thủ tục. Các câu lệnh này có thể đặt trong cặp từ khoá BEGIN...END hoặc có thể không.

Tùy theo yêu cầu và mục đích sử dụng mà ta chọn hình thức SP tự động thực thi khi SQL Server khởi động hay chỉ thực thi khi được gọi. Tương ứng với 2 cách cài đặt như sau:

**Cú pháp thực thi thủ tục khi được gọi:**

**EXECUTE|EXEC Procedure\_name**[<parameter>[, …n][ OUTPUT ]]

**Cú pháp thực thi thủ tục tự động:** Store procedure phải nằm ở CSDL **Master.**

Dùng thủ tục **Sp\_procoption** để gán thuộc tính tự động thực thi

USE Master

EXEC Sp\_procoption [ @ProcName = ] '*procedure*'

, [@OptionName =] Startup

, [@OptionValue =] True

*(Ví dụ: EXECute Sp\_procoption dbo.overdueOrders Startup, True)*

**Ví dụ 1:** Hãy xây dựng thủ tục cho phép xuất ra thông tin của các nhân viên phòng nghiên cứu

**CREATE PROCEDURE DSNV\_PHONGNGHIENCUU**

**AS**

**SELECT** NHANVIEN.\*

**FROM** NHANVIEN, PHONGBAN

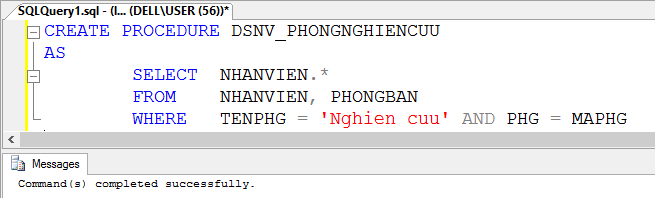
**WHERE** TENPHG = ‘Nghien cuu’ AND PHG = MAPHG

**EXECUTE** DSNV\_PHONGNGHIENCUU

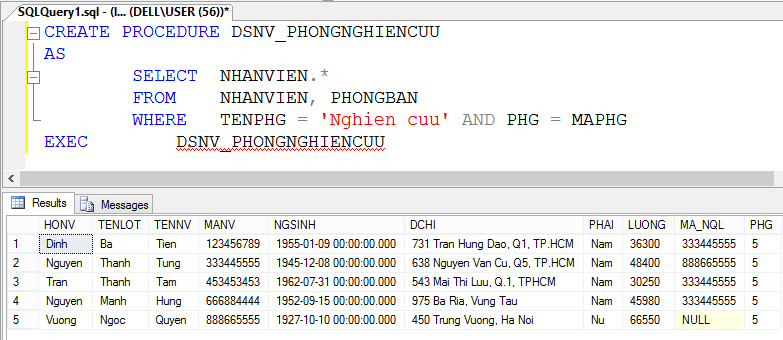
**Hoặc**

**EXEC** DSNV\_PHONGNGHIENCUU

**Kết quả cài đặt Procedure**

****

**Kết quả khi thực thi Procedure**

****

**Ví dụ 2:** Hãy xây dựng thủ tục cho phép xuất ra thông tin của các nhân viên của một phòng bất kỳ theo yêu cầu người truy vấn.

**CREATE PROCEDURE DSNV\_PHONG** *(@tenphong varchar (12))*

**AS**

**SELECT** NHANVIEN.\*

**FROM**  NHANVIEN, PHONGBAN

**WHERE** TENPHG = @*tenphong* AND PHG = MAPB

**EXECUTE** DSNV\_PHONG ‘Nghiên cứu’

**Hoặc**

**EXEC** DSNV\_PHONG ‘Nghiên cứu’

#### 6.1.2.2. Sửa và xóa thủ tục

**Cú pháp sửa thủ tục:**

**ALTER PROCEDURE|PROC Procedure\_name**

**AS**

**sql\_statement [...]**

Trong đó: **sql\_statement [...]** là câu lệnh chỉnh sửa theo nhu cầu

**Cú pháp xóa thủ tục:**

**DROP Procedure\_name**

**Ví dụ:**

*Yêu cầu 1*: Hãy cài đặt thủ tục cho biết số nhân viên của các phòng ban. Thông tin bao gồm: tên phòng, số nhân viên tương ứng của phòng đó.

**CREATE PROC** SLNV\_PHOG

**AS**

**BEGIN**

**SELECT** TENPHG, COUNT (MANV) AS SLNV

**FROM** NHANVIEN, PHONGBAN

**WHERE** PHG = MAPHG

**GROUP BY** PHG, TENPHG

**END**

*Yêu cầu 2*: Hãy sửa lại thủ tục từ yêu cầu 1, hãy cho biết tên phòng kèm số nhân viên của phòng đó. Chỉ xuất ra thông tin của các phòng có trên 5 nhân viên.

**ALTER PROC** SLNV\_PHOG

**AS**

**BEGIN**

**SELECT** TENPHG, COUNT (MANV) AS SLNV

**FROM** NHANVIEN, PHONGBAN

**WHERE** PHG = MAPHG

**GROUP BY** PHG, TENPHG

**HAVING** COUNT (MANV) > 5

**END**

*Yêu cầu 3:* Hãy xóa thủ tục vừa tạo

**DROP** SLNV\_PHOG

### 6.1.3. Sử dụng tham số trong thủ tục

#### 6.1.3.1. Input Parameter/ Output Parameter

**Input Parameter:** tham số nhập, cầu nối giữa giá trị nhập từ yêu cầu người truy vấn với điều kiện để thực hiện lọc kết quả trong CSDL thông qua tham số được cài đặt trong thủ tục.

**Ví dụ:** tham số tenphong trong ví dụ bên dưới chính là input parameter

**CREATE PROCEDURE DSNV\_PHONG** *(@tenphong varchar(12))*

**AS**

**SELECT** NHANVIEN.\*

**FROM**  NHANVIEN, PHONGBAN

**WHERE** TENPHG = @*tenphong* AND PHG = MAPHG

**Output Parameter:** tham số xuất chứa giá trị trả về của thủ tục. Thông thường, sử dụng output parameter khi có nhu cầu sử dụng giá trị trả về của thủ tục cho một/nhiều hành động kế tiếp.

**Ví dụ**: tham số luongmax trong ví dụ bên dưới chính là output parameter

CREATE PROCEDURE LuongCaoNhat\_phong (@tenphong varchar (12), @luongmax int OUTPUT)

AS

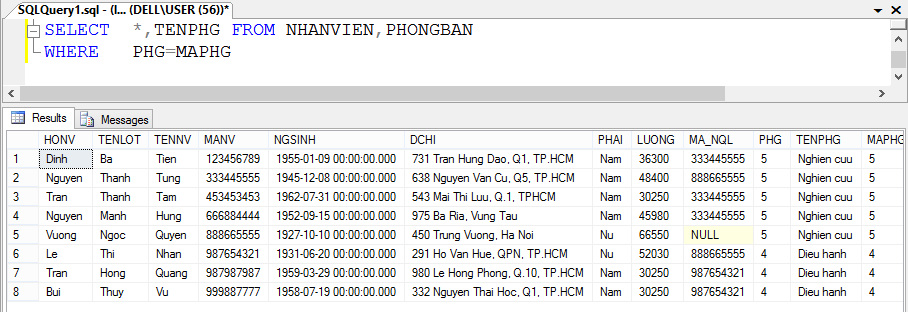
SELECT  @luongmax = max (LUONG)

FROM  NHANVIEN, PHONGBAN

WHERE PHG = MAPHG AND TENPHG = @tenphong

GO

**Dữ liệu ban đầu:**

****

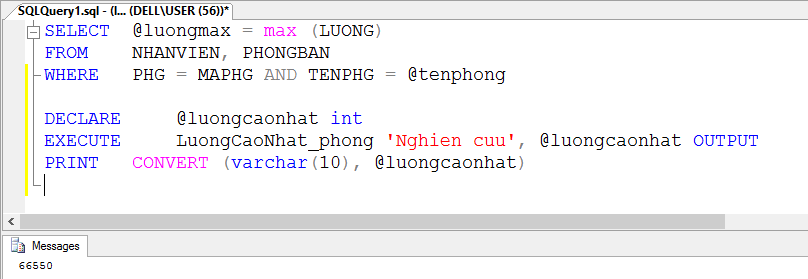
**Thực thi:** Bắt buộc phải khai báo một biến trước khi gọi thủ tục thực thi

DECLARE  @luongcaonhat int

EXECUTE LuongCaoNhat\_phong ‘Nghien cuu’, @luongcaonhat OUTPUT

PRINT  CONVERT (varchar(10), @luongcaonhat)

**Kết quả:**

****

#### 6.1.3.2. Cài đặt thủ tục theo mục đích sử dụng tham số

Một vấn đề khó khăn chung cho người lập trình, đặc biệt trong quá trình cài đặt thủ tục là việc xác định:

* Có sử dụng tham số hay không? (sử dụng tham số trong trường hợp nào?)
* Nếu có thì bao nhiêu tham số là đủ?
* Khai báo tham số như thế nào?
* Việc khai báo tham số phải đảm bảo ràng buộc gì?
* Khi nào sử dụng input parameter hay output parameter?
* …

Trong nội dung này, chúng ta sẽ cùng phân tích và trả lời các câu hỏi trên.

Phân loại theo chức năng và mục đích sử dụng tham số, cài đặt thủ tục được chia làm 3 cách tương ứng:

* Thủ tục không dùng tham số
* Thủ tục sử dụng tham số
* Thủ tục sử dụng tham số với giá trị mặc định

Cho một phần cơ sở dữ liệu “Quản lý đề án công ty” như sau:

**NHANVIEN** (MANV, HONV, TENLOT, TENNV, PHAI, DIACHI, LUONG, NGAYSINH, PHG)

**PHONGBAN** (MAPHG, TENPHG, DIADIEM\_PHG)

**DEAN** (MADA, TENDA, PHONG)

**PHANCONG** (MANV, SODA, THOIGIAN)

**Hãy cài đặt thủ tục thực hiện các tình huống như sau:**

*Tình huống 1:* Hãy cho biết số lượng nhân viên của từng phòng

*Tình huống 2:* Hãy cho biết số lượng nhân viên của từng phòng theo yêu cầu truy vấn

*Tình huống 3:* Hãy cho biết số lượng nhân viên của phòng tương ứng theo yêu cầu truy vấn, nếu người dùng không cung cấp thông tin thì mặc định hãy xuất ra số lượng nhân viên của phòng quản lý.

Xét các tình huống trên, ta thấy:

**Tình huống 1:** Hãy cho biết số lượng nhân viên của từng phòng 🡪 Không yêu cầu phòng cụ thể, chỉ cần xuất ra lần lượt số lượng nhân viên của từng phòng 🡪 Trong trường hợp này cài đặt thủ tục không cần dùng tham số.

**Tình huống 2:** Hãy cho biết số lượng nhân viên của từng phòng **theo yêu cầu truy vấn** 🡪 người truy vấn sẽ cung cấp giá trị đầu vào để làm điều kiện lọc kết quả (điều kiện trong mệnh đề WHERE)🡪 cần cài đặt thủ tục sử dụng tham số đầu vào (tham số khai báo trong cài đặt thủ tục theo cú pháp) để tiếp nhận giá trị do người truy vấn cung cấp.

**Tình huống 3:** Hãy cho biết số lượng nhân viên của phòng tương ứng theo yêu cầu truy vấn, nếu người dùng không cung cấp thông tin thì **mặc định** hãy xuất ra số lượng nhân viên của phòng quản lý 🡪 xét về mặt ngữ nghĩa tương tự với yêu cầu 2, ngoài ra còn phát sinh trường hợp MẶC ĐỊNH 🡪 cần gán giá trị mặc định cho tham số khi cài đặt khi người người dùng không cung cấp thông tin đầu vào.

HQT CSDL SQL Server cung cấp 3 cú pháp khai báo thủ tục tương ứng với từng mục đích truy vấn như sau:

* **Cú pháp thủ tục không sử dụng tham số**:

**CREATE PROC Procedure\_name**

**[WITH option]**

**AS**

**sql\_statement [...]**

**Tình huống 1:**Hãy cho biết số lượng nhân viên của từng phòng

**CREATE PROC** SLNV\_PHOG

**AS**

**BEGIN**

**SELECT** TENPB, COUNT (MANV) AS SLNV

**FROM** NHANVIEN, PHONGBAN

**WHERE** PHG = MAPHG

**GROUP BY** PHG, TENPHG

**END**

* **Cú pháp thủ tục có sử dụng tham số**:

**CREATE PROC Procedure\_name (@ts1 data\_type, @ts2 data\_type, …)**

**[WITH option]**

**AS**

**sql\_statement [...]**

**Tình huống 2:** Hãy cho biết số lượng nhân viên của từng phòng theo yêu cầu truy vấn

**CREATE PROCEDURE DSNV\_PHONG** *(@tenphong varchar(12))*

**AS**

**SELECT** TENPB, COUNT (MANV) AS SLNV

**FROM**  NHANVIEN, PHONGBAN

**WHERE** TENPHG = @*tenphong* AND PHG = MAPHG

* **Cú pháp thủ tục sử dụng tham số có giá trị mặc định**:

**CREATE PROC Procedure\_name (@ts1 data\_type = ‘giá trị mặc định’, …)**

**[WITH option]**

**AS**

**sql\_statement [...]**

***Tình huống 3:***Hãy cho biết số lượng nhân viên của phòng tương ứng theo yêu cầu truy vấn, nếu người dùng không cung cấp thông tin thì mặc định hãy xuất ra số lượng nhân viên của phòng quản lý.

**Thực thi:**

**EXEC** SLNV\_PHG ‘Nghiên cứu’ 🡪 Xuất ra số lượng nhân viên của phòng nghiên cứu

**EXEC** SLNV\_PHG 🡪 Xuất ra số lượng nhân viên của phòng quản lý

**Lưu ý:** Tùy theo cách truyền tham số khác nhau khi gọi lệnh thực thi sẽ trả về kết quả không giống nhau.

**Tình huống 4:** Cho thủ tục cài đặt như sau:

**CREATE PROC** sp\_TestDefault (@tenphong NVARCHAR (30) = NULL,

                    @diachi NVARCHAR (100) ='Huế')

AS

     BEGIN

          IF (@tenphong IS NULL)

              SELECT HONV, TENLOT, TENNV

              FROM NHANVIEN, PHONGBAN

              WHERE DCHI=@diachi AND

NHANVIEN.MANV = PHONGBAN.MAPHG

          ELSE

              SELECT HONV, TENLOT, TENNV

              FROM NHANVIEN, PHONGBAN

              WHERE NHANVIEN.PHG = PHONGBAN.MAPHG AND

DCHI = @diachi AND TENPHG = @tenphong

     END

Thủ tục *sp\_TestDefault* với tham số ***@tenphong*** có giá trị mặc định là *NULL* và tham số **@diachi** có giá trị mặc định là *Huế*.

* Cho biết họ tên của các nhân viên có địa chỉ tại *Huế*:

sp\_testdefault

* Cho biết họ tên của các nhân viên phòng quản lý có địa chỉ tại *Huế*:

        sp\_testdefault @tenphong='Quản lý'

* Cho biết họ tên của các nhân viên có địa chỉ tại *Nghệ An*:

        sp\_testDefault @diachi=N'Nghệ An'

* Cho biết họ tên của các nhân viên phòng nghiên cứu có địa chỉ tại *Đà Nẵng*:

        sp\_testdefault @tenphòng='Nghiên cứu',@diachi=N'Đà Nẵng'

## 6.2. Hàm (Function)

### 6.2.1. Khái niệm

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

Hàm được dùng trong:

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

### 6.2.2. Phân loại hàm

Thông thường hàm được chia ra làm 2 dạng: hàm hệ thống và hàm do người dùng tự định nghĩa

#### 6.2.2.1. Hàm hệ thống (Build-in functions)

**Định nghĩa:** Là tập hợp các hàm được định nghĩa sẵn, đi kèm với HQT CSDL cài đặt. Những hàm này hoạt động như là một định nghĩa trong T-SQL và không thể hiệu chỉnh. Hàm hệ thống có thể chỉ được tham chiếu trong các câu lệnh T-SQL. Giá trị trả về của hàm có thể là một Rowset (tập các dòng), argergate (giá trị tổng hợp) và scalar (vô hướng).

Các hàm do hệ thống định nghĩa được chia thành 5 nhóm hàm chính như sau: nhóm hàm xử lý chuỗi (String functions), nhóm hàm về ngày giờ (Date functions), nhóm hàm toán học (Mathematical functions), nhóm hàm kết hợp (aggregate Functions), nhóm hàm hệ thống khác (System functions).

* + - 1. **Nhóm hàm xử lý chuỗi (String functions** thao tác với dữ liệu kiểu ký tự.
* Hàm **CHARINDEX (***string1, string2, start\_position***)**: Tìm vị trí bắt đầu của chuỗi ký tự chỉ định string1 trong chuỗi string2 và bắt đầu tìm ở vị trí start\_position trong chuỗi string2.

***Ví dụ****:* SELECT CHARINDEX ('test', 'This is a test', 1) 🡪 Hàm sẽ trả về giá trị 11, vị trí bắt đầu của chuỗi ‘test’ trong chuỗi 'This is a test'.

* Hàm **LEFT (***string, number\_of\_characters***)**: Trả về chuỗi gồm number\_of\_characters ký tự tính từ trái sang của chuỗi string.

***Ví dụ****:* SELECT LEFT (‘This is a test’, 4) 🡪 Hàm sẽ trả về chuỗi ‘This’

* Hàm **LEN** (*string)*: Xác định độ dài của chuỗi ký tự *string.*

***Ví dụ:***SELECT LEN (‘This is a test’) 🡪 Hàm sẽ trả về giá trị 14

* Hàm **LOWER (***string***)**: Hàm trả về chuỗi ký tự thường.

***Ví dụ:***SELECT LOWER (‘This is a TEST’)🡪 Hàm sẽ trả về chuỗi ‘this is a test’

* Hàm **LTRIM (***string***)**: Cắt bỏ các ký tự trắng bên trái của chuỗi.

***Ví dụ:***SELECT LTRIM (‘ This is a test ’)🡪 Hàm sẽ trả về chuỗi ‘This is a test ’

* Hàm **RIGHT** (*string, number\_of\_characters*): trả về chuỗi gồm number\_of\_characters ký tự tính từ phải sang của chuỗi string.

***Ví dụ:***SELECT RIGHT (‘This is a test’, 4)🡪 Hàm sẽ trả về chuỗi ‘test’

* Hàm **RTRIM** (*string*): Cắt bỏ các ký tự trắng bên phải của chuỗi.

***Ví dụ:***SELECT RTRIM(‘ This is a test ’) 🡪 Hàm sẽ trả về chuỗi ‘ This is a test’

* Hàm **SUBSTRING** (*expression, start, length*): Hàm trả về chuỗi con gồm length ký tự của expression tính từ vị trí start.

***Ví dụ:*** SELECT x = SUBSTRING ('abcdef', 2, 3) 🡪 Hàm sẽ trả về chuỗi ‘bcd’

* Hàm **UPPER** (*string)*: Chuyển đổi các ký thường thành chữ hoa.

***Ví dụ:***SELECT UPPER (‘This is a TEST’) 🡪 Hàm sẽ trả về chuỗi ‘THIS IS A TEST’

* + - 1. **Nhóm hàm về ngày giờ (Date functions):** Là các hàm làm việc với dữ liệu kiểu datetime. Một số hàm làm việc với các kiểu thông tin đặc biệt được gọi là **datepart**. Trước khi đi vào các hàm, ta xét một số ký tự đặc tả của **datepart** cho trong bảng sau:

|  |  |  |
| --- | --- | --- |
| **Datepart** | **Abbreviation** | **Values** |
| Hour | hh | 0 - 23 |
| Minute | Mi | 0 - 59 |
| Second | Ss | 0 - 59 |
| Millisecond | Ms | 0 - 999 |
| Day of Year | Dy | 1 - 366 |
| Day | Dd | 1 - 31 |
| Week | wk | 1 - 53 |
| Weekday | dw | 1 - 7 |
| Month | mm | 1 - 12 |
| Quarter | qq | 1 - 4 |
| Year | yy | 1753 - 9999 |

* Hàm **DATEADD** (*datepart, amount, date*): cộng thêm một số amount thời gian thành phần datepart của date.

***Ví dụ:*** SELECT DATEADD (year, 1, GETDATE())) 🡪 Hàm sẽ trả về ngày hiện tại cộng thêm một năm.

* Hàm **DATEDIFF** (*datepart, date1, date2*): so sánh chênh lệch giữa hai ngày bằng việc sử dụng tham số datepart.

***Ví dụ****:* SELECT DATEDIFF (hour, ‘1/1/2008 12:00:00’, ‘1/1/2008 16:00:00’) 🡪 trả về giá trị 4 🡪 hai ngày chênh nhau 4 giờ.

* Hàm **DATEPART** (*datepart, date*): trả về giá trị của thành phần datepart trong date.

***Ví dụ:*** SELECT DATEPART (month, ‘1/1/2008 16:00:00’) 🡪 trả về giá trị tháng 1.

* Hàm **DAY** (*date*): Xác định số ngày của tháng trong dữ liệu ngày giờ date.

***Ví dụ****:* SELECT DAY ('7/22/1979 00:04:00')🡪 trả về giá trị ngày là 22

* Hàm **GETDATE ():** Trả về giá trị ngày hiện tại của hệ thống.
* Hàm **MONTH** (*date*): Trả về tháng của dữ liệu ngày giờ.
* Hàm **YEAR** (*date*): Trả về năm của dữ liệu ngày giờ.
  + - 1. **Nhóm hàm toán học (Mathematical functions)**
* Hàm **ABS** (*number)*: Trả về giá trị tuyệt đối của số number.
* Hàm **CEILING** *(number):* Trả về số nguyên nhỏ nhất lớn hơn hoặc bằng number.
* Hàm **FLOOR** (*number*): Trả về số nguyên lớn nhất nhỏ hơn hoặc bằng number.
* Hàm **ROUND** (*number, precision*): Hàm làm tròn số number lấy precision chữ số sau dấu thập phân.
* Hàm **SQUARE** (*number*): Hàm trả về giá trị bình phương số number.
* Hàm **SQRT** (number): Hàm trả về giá trị căn bậc hai số number.
  + - 1. **Nhóm hàm kết hợp (aggregate Functions):**

Các hàm tập hợp thực hiện tính toán trên một tập hợp các giá trị và trả về một giá trị đơn. Ngoại trừ hàm COUNT, hàm tập hợp bỏ qua các giá trị NULL.

Các hàm tập hợp thường sử dụng với mệnh đề GROUP BY trong khối câu lệnh SELECT. Hàm tập hợp được phép dùng như là các biểu thức trong trường hợp:

* Trong danh sách select của khối câu lệnh SELECT.
* Trong mệnh đề COMPUTE hoặc COMPUTE BY .
* Trong mệnh đề HAVING

**Sau đây là một số hàm tập hợp hay được sử dụng:**

* Hàm **AVG** *([ALL|DISTINCT] expression*): trả về giá trị trung bình của tập các giá trị trong một nhóm.

Trong đó:

* **ALL**: Áp dụng cho các hàm tập hợp để chỉ định cho tất cả các giá trị. ALL là từ khóa mặc định.
* **DISTINCT**: Chỉ định chỉ lấy một thể hiện duy nhất của một giá trị. Nghĩa là trong tập hợp có nhiều phần tử có cùng một giá trị thì chỉ lấy một giá trị đại diện cho nó.

***Ví dụ: Hãy tìm mức lương trung bình của các phòng***

USE QUANLYDEANCONGTY

SELECT DISTINCT AVG (LUONG)

FROM NHANVIEN

GROUP BY PHG

🡪 Có nhiều phòng có mức lương trung bình bằng nhau thì kết quả chỉ trả về 1 giá trị TB chung (ví dụ tồn tại 2 phòng đều có giá trị lương trung bình là 3000$ thì chỉ trả về 1 dòng 3000$)

* Hàm **COUNT** ({[ALL|DISTINCT] expression]|\*}): trả về kiểu int số các phần tử của một nhóm.

***Lưu ý:***Khi sử dụng hàm COUNT

* **COUNT (\*):** Trả về số các phần tử trong một nhóm bao gồm cả giá trị NULL và giá trị duplicates.
* **COUNT (**ALL *expression*): xác định giá trị cho *expression* tại mỗi dòng trong nhóm và trả về số các giá trị không NULL
* **COUNT** (DISTINCT *expression*): xác định giá trị cho *expression* tại mỗi dòng trong nhóm và trả về số các giá trị không trùng và không NULL
* Hàm **COUNT\_BIG**({[ALL|DISTINCT] expression}|\*): Trả về số các phần tử trong một nhóm. Hàm COUNT\_BIG làm việc như hàm COUNT. Điểm khác nhau giữa chúng là hàm COUNT trả về giá trị kiểu int còn hàm COUNT\_BIG trả về giá trị kiểu bigint.
* Hàm **MAX** ([ALL|DISTINCT] expression): Trả về giá trị lớn nhất trong biểu thức expression.
* Hàm **MIN** ([ALL|DISTINCT] expression): Trả về giá trị lớn nhất trong biểu thức expression
* Hàm **SUM** ([ALL|DISTINCT] expression): Trả về tổng của tất cả các giá trị của biểu thức hoặc tổng các giá trị DISTINCT của biểu thức expression. Hàm SUM chỉ áp dụng cho các cột kiểu số. Các giá trị NULL được bỏ qua.

**Nhóm hàm hệ thống khác (System functions):** Các hàm hệ thống là các hàm lấy thông tin hệ thống về các đối tượng và đã thiết lập trong SQL Server.

* Hàm **CONVERT** (data\_type, expression): Chuyển đổi biểu thức expression thành kiểu dữ liệu data\_type

***Ví dụ:*** SELECT CONVERT (VARCHAR (5), 12345)🡪 trả về chuỗi ‘12345’.

* Hàm **CAST** (expression AS data\_type): Chuyển đổi biểu thức expression thành kiểu dữ liệu data\_type.
* Hàm **CURRENT\_USER**: Trả về người sử dụng hiện tại

***Ví dụ:***SELECT CURRENT\_USER

* Hàm **DATALENGTH** (expression): Trả về số byte được sử dụng trong biểu thức expression.
* Hàm **HOST\_NAME** (): trả về tên máy tính mà người sử dụng hiện tại đang login.

***Ví dụ:***SELECT HOST\_NAME()

* Hàm **SYSTEM\_USER**: Hàm trả về tên của các User đang login hệ thống.

***Ví dụ:***SELECT SYSTEM\_USER

* Hàm **USER\_NAME():** Hàm trả về username khi đưa số user ID.

***Ví dụ:***SELECT name

FROM sysobjects

WHERE USER\_NAME (uid) = 'dbo'

#### 6.2.2.2. Hàm do người dùng định nghĩa (User-defined function - UDFs)

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

UDFs thường được chia thành 3 dạng cơ bản:

**Scalar Functions:**dạng hàm vô hướng trả về một giá trị đơn và có thể được sử dụng như một biến/giá trị trong một biểu thức (câu lệnh Select, mệnh đề SET của câu lệnh Update). Một hàm vô hướng có thể được xem như kết quả của vài phép toán hoặc hàm chuỗi.

**Table-valued Functons:** dạng hàm có giá trị bảng trả về một tập kết quả và có thể được sử dụng như một bảng dữ liệu hay view. Hàm giá trị bảng có thể được tham chiếu trong mệnh đề FROM của câu lệnh SELECT.

**Multistatement Table-valued:** dạng hàm này xây dựng tập kết quả từ một hay nhiều câu lệnh SELECT.

### 6.2.3. Tạo lập và sử dụng hàm

**Cú pháp chung**

**CREATE FUNCTION Function\_name (**[<parameter> [, …n]**)**

**RETURNS (data\_type)**

**AS**

**sql\_statement [...]**

**Trong đó:**

* **Parameter:** Các tham số cách nhau bởi dấu phẩy. Khai báo của mỗi một tham số tối thiểu phải bao gồm hai phần:tên tham số được bắt đầu bởi dấu @, kiểu dữ liệu của tham số;
* **Data\_type:** kiểu dữ liệu trả về của hàm;
* **Sql\_statement:** tập hợp các câu lệnh sử dụng trong nội dung hàm. Các câu lệnh này có thể đặt trong cặp từ khoá BEGIN...END hoặc có thể không.

Dựa trên cơ sở phân loại UDFs thành 3 loại: Scalar Functions, Table-valued Functons, Multistatement Table-valued. Tương ứng có các cú pháp xây dựng hàm như sau:

#### 6.2.3.1. Scalar Functions:

**Đặc điểm:** Dạng hàm vô hướng trả về một giá trị đơn và có thể được sử dụng như một biến/giá trị trong một biểu thức (câu lệnh Select, mệnh đề SET của câu lệnh Update). Một hàm vô hướng có thể được xem như kết quả của vài phép toán hoặc hàm chuỗi.

**Cú pháp**

**CREATE FUNCTION Function\_name (**[<parameter> [, …n]**)**

**RETURNS (data\_type)**

**AS**

**BEGIN**

**<sql\_statement [...]>**

**RETURN (result)**

**END**

**Ví dụ**: Viết hàm trả về thứ của một ngày trong tuần.

**CREATE FUNCTION Thu (@ngay DATETIME)**

RETURNS NVARCHAR(10)

AS

     BEGIN

          DECLARE @st NVARCHAR(10)

          SELECT @st=CASE DATEPART(DW,@ngay)

                        WHEN 1 THEN 'Chu nhật'

                        WHEN 2 THEN 'Thứ hai'

                        WHEN 3 THEN 'Thứ ba'

                        WHEN 4 THEN 'Thứ tư'

                        WHEN 5 THEN 'Thứ năm'

                        WHEN 6 THEN 'Thứ sáu'

                        ELSE 'Thứ bảy'

                      END

          RETURN (@st) /\* Trị trả về của hàm \*/

     END

**Lưu ý:** một hàm khi đã được định nghĩa có thể được sử dụng như các hàm do HQT CSDL cung cấp (thông thường trước tên hàm ta phải chỉ định thêm tên của người sở hữu hàm)

Câu lệnh SELECT dưới đây sử dụng hàm đã được định nghĩa ở ví dụ trước:

SELECT HONV, TENLOT, TENNV, DBO. THU (NGAYSINH)

FROM NHANVIEN

WHERE PHG = 5

#### 6.2.3.2. Table-valued Functons

**Đặc điểm:** Dạng hàm vô hướng trả về một giá trị đơn và có thể được sử dụng như một biến/giá trị trong một biểu thức (câu lệnh Select, mệnh đề SET của câu lệnh Update). Một hàm vô hướng có thể được xem như kết quả của vài phép toán hoặc hàm chuỗi.

**Cú pháp**

**CREATE FUNCTION Function\_name (**[<parameter> [, …n]**)**

**RETURNS TABLE**

**AS**

**BEGIN**

**RETURN (SELECT\_statement)**

**END**

Trong đó:

* Kiểu trả về của hàm phải được chỉ định bởi mệnh đề RETURNS TABLE.
* Trong phần thân của hàm chỉ có duy nhất một câu lệnh RETURN xác định giá trị trả về của hàm thông qua duy nhất một câu lệnh SELECT. Ngoài ra, không sử dụng bất kỳ câu lệnh nào khác trong phần thân của hàm.

**Ví dụ:** Hãy trả về danh sách các nhân viên ở phòng có tên theo yêu cầu người truy vấn

CREATE FUNCTION func\_XemNV (@tenphong nvarchar(30))

RETURNS TABLE

AS

  RETURN ( SELECT HONV, TENLOT, TENNV

         FROM NHANVIEN, PHONGBAN ON PHG = MAPHG

WHERE TENPHG = @tenphong)

Với hàm được định nghĩa như trên, để biết danh sách các nhân viên phòng nghiên cứu, ta sử dụng câu lệnh như sau:

     SELECT \* FROM dbo.func\_XemNV (*‘Nghiên cứu’*)

#### 6.2.3.3. Multistatement Table-valued

**Đặc điểm:** dạng hàm này xây dựng tập kết quả từ một hay nhiều câu lệnh SELECT

**Cú pháp**

CREATE FUNCTION [owner\_name.]function\_name

([{@parameter\_name [AS] data\_type [=default]} [, … n]])

RETURNS @TABLE\_NAME

TABLE ({column\_definition | table\_constraint} [, … n])

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

[AS]

BEGIN

< sql\_statement [...]>

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 @*biếnbảng* được định nghĩa trong mệnh đề RETURNS

**Ví dụ:** hãy viết hàm tạo biến bảng thống kê, biến bảng này mang giá trị là một bảng dữ liệu gồm có 3 cột: mã phòng, tên phòng, tổng số nhân viên của phòng đó.

CREATE FUNCTION Func\_TongNV (@phong SMALLINT)

RETURNS @bangthongke TABLE

     (

          MaPhong    SMALLINT,

          TenPhong   NVARCHAR (50),

          TongSoNV  INT

     )

AS

     BEGIN

          IF @phong=0

              INSERT INTO @bangthongke

              SELECT MAPHG, TENPHG, COUNT (MANV)

              FROM NHANVIEN INNER JOIN PHONGBAN ON PHG=MAPHG

              GROUP BY MAPHG, TENPHG

          ELSE

              INSERT INTO @bangthongke

              SELECT MAPHG, TENPHG, COUNT (MANV)

              FROM NHANVIEN INNER JOIN PHONGBAN ON PHG=MAPHG

              WHERE PHG=@maphong

              GROUP BY MAPHG, TENPHG

          RETURN /\*Trả kết quả về cho hàm\*/

     END

* Để lập bảng thống kê số lượng nhân viên của tất cả các phòng, ta sử dụng câu lệnh:

SELECT \* FROM Func\_TongNV () ORDER BY TongSoNV

* Để lập bảng thống kê số lượng nhân viên của một phòng bất kỳ khi biết mã số phòng chẳng hạn như phòng 5, ta sử dụng câu lệnh như sau:

SELECT \* FROM Func\_TongNV (5)

# BÀI TẬP CHƯƠNG 6

## CÂU HỎI

**Câu hỏi 1:** Phân biệt Hàm và Thủ tục.

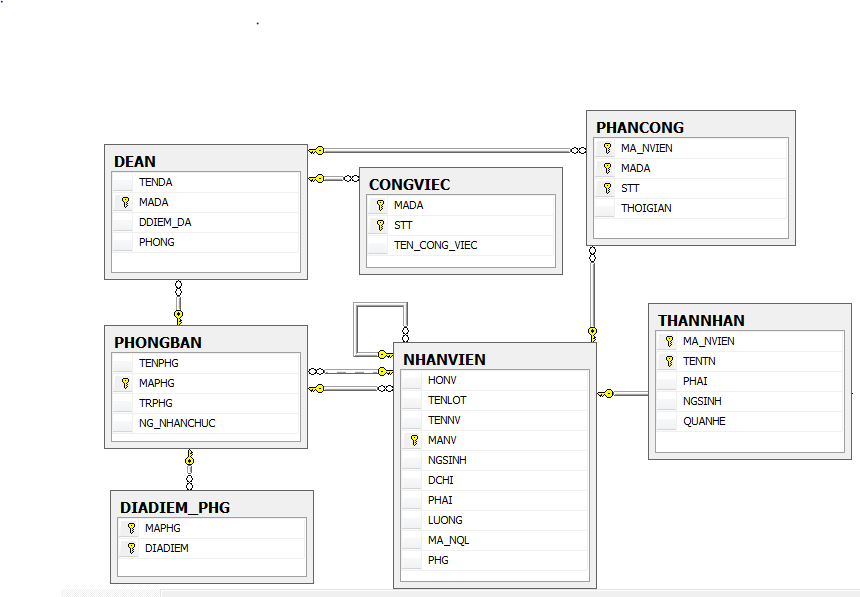
**Câu hỏi 2:** Từ khóa OUT/IN trong khai báo hàm/thủ tục có tác dụng gì?

**Câu hỏi 3:** Khi cài đặt hàm hoặc thủ tục, người dùng dựa vào các tiêu chí nào để xác định có hay không có truyền tham số? Số lượng tham số (nếu có)?

**Câu hỏi 4:** Hãy trình bày cách cài đặt và sử dụng một hàm hay một thủ tục bất kỳ thông qua một ví dụ.

## BÀI TẬP

Cho lược đồ cơ sở dữ liệu ‘QUẢN LÝ ĐỀ ÁN CÔNG TY’ như sau:

Với cơ sở dữ liệu cho trên, hãy viết các thủ tục/hàm thực hiện các yêu cầu truy vấn sau:

* + - 1. Hãy liệt kê danh sách các nhân viên ở phòng nghiên cứu.
      2. Hãy liệt kê danh sách các nhân viên thuộc một phòng bất kỳ theo yêu cầu người truy vấn.
      3. Hãy liệt kê danh sách các nhân viên nữ phòng quản lý có mức lương cao hơn mức lương người yêu cầu truy vấn.
      4. Hãy cho biết thông của nhân viên có lương cao nhất trong phòng mà nhân viên đó trực thuộc, với thông tin tên phòng do người truy vấn yêu cầu.
      5. Hãy viết hàm lấy ra tháng từ ngày sinh của các nhân viên. Hãy xuất ra danh sách các nhân viên sinh vào tháng 05.
      6. Hãy viết hàm trích ra năm của các nhân sinh sinh trước 1975. Hãy xuất ra danh sách các nhân viên nữ sinh trước năm 1975.
      7. Hãy viết hàm hoặc thủ tục cập nhật tăng lương cho các nhân viên tham gia trên 3 đề án với mức tăng 10%. Trong khi lương các nhân viên đó vẫn chưa đạt mức 50000, hãy tiếp tục tăng. Xuất ra mức lương sau khi tăng.
      8. Xuất ra danh sách các nhân viên nữ đang đến độ tuổi về hưu. Biết rằng quy định nữ về hưu ở độ tuổi 55.
      9. Hãy viết hàm trả về giá trị lương trung bình cao nhất trong các phòng. Hãy xuất tên phòng ban có mức lương trung bình cao nhất.
      10. Hãy viết hàm xuất tên phòng có mức lương trung bình thấp nhất so với các phòng khác. Tăng lương cho các nhân viên phòng đó 10% cho đến khi mức lương trung bình cao hơn 35000. Xuất ra mức lương sau khi tăng của từng nhân viên phòng đó.

# CHƯƠNG 7 RÀNG BUỘC TOÀN VẸN CAO CẤP VỚI TRIGGER.

# MỤC ĐÍCH VÀ YÊU CẦU

**Mục đích:**

Chương này nhằm cung cấp cho sinh viên những kiến thức chung về:

* Những kiến thức cơ bản về Trigger, cách cài đặt các Trigger cơ bản và nâng cao.
* Phương pháp giải quyết các ràng buộc toàn vẹn cao cấp bằng Trigger.

**Yêu cầu:**

Sau khi học xong chương này sinh viên có khả năng:

* Phân tích việc xử lý các ràng buộc toàn vẹn cao cấp bằng Trigger.
* Nắm vững cú pháp, phân tích vấn đề và vận dụng thành thạo các câu lệnh xây dựng các Trigger kiểm tra tính toán vẹn dữ liệu.
* Phân biệt được bản chất Trigger For và Trigger After/ Instead of.

# NỘI DUNG

## 7.1. Giới thiệu

### 7.1.1. Khái niệm:

Ở chương 3, chúng ta đã được giới thiệu về ràng buộc toàn vẹn cũng như cách cài đặt các ràng buộc toàn vẹn đơn giản bằng các khái niệm Constraint, Rule, Check. Tuy nhiên đối với những ràng buộc toàn vẹn tương đối phức tạp, cần có những xử lý cập nhật dữ liệu đi kèm khi kiểm tra ràng buộc toàn vẹn, việc sử dụng các khái niệm trên để cài đặt là không khả thi. Khi đó ta sử dụng Trigger để cài đặt ràng buộc toàn vẹn.

Trigger là một loại Stored Procedure đặc biệt được thực thi một cách tự động khi có câu lệnh Update, Insert, Delete tác động vào Table. Trigger là một công cụ mạnh, khi dữ liệu bị sửa đổi, nó sẽ tự động thực hiện việc ép buộc các giao dịch sửa đổi này thực hiện theo quy tắc đã định nhằm đảm bảo tính toàn vẹn dữ liệu. Các Trigger thực thi sau khi các câu lệnh bên trong nó hoàn tất, nếu có lỗi thì Trigger sẽ không thực thi.

 Khi thực hiện một Trigger, hệ quản trị cơ sở dữ liệu SQL Server tự động tạo ra hai bảng Inserted và Deleted trong bộ nhớ chính cũng như bộ nhớ cục bộ cho mỗi Trigger. Như vậy khi áp dụng Trigger trên bảng nào thì bảng Inserted và Deleted sẽ được sử dụng riêng cho bảng đó. Cấu trúc hai bảng Inserted và Deleted được tạo ra sẽ giống cấu trúc của bảng mà Trigger đang thực thi và chỉ tồn tại trong thời gian Trigger đó thực thi.

|  |  |  |  |
| --- | --- | --- | --- |
| TENPHNG | MAPHG | TRPHG | NG\_NHANCHUC |
| TABLE PHONGBAN | | | |
| TENPHNG | **MAPHG** | **TRPHG** | **NG\_NHANCHUC** |
| TABLE INSERTED | | | |
| TENPHNG | **MAPHG** | **TRPHG** | **NG\_NHANCHUC** |
| TABLE DELETED | | | |

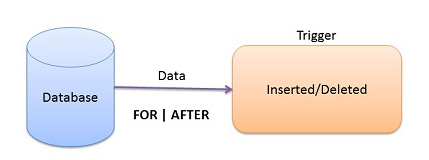
**Trong hình ảnh ví dụ trên, bảng Inserted** chứa các dòng dữ liệu vừa được Insert hay Update vào quan hệ PHONGBAN mà Trigger đang thực thi. Trong khi đó, bảng **Deleted** chứa các dòng dữ liệu mới được xóa khỏi quan hệ PHONGBAN bằng thao tác Delete hay Update.

Thao tác Update trên một bảng về bản chất chính là thao tác Delete những dòng dữ liệu cũ và Insert những dòng dữ liệu mới. Như vậy, khi thực hiện thao tác Update trên quan hệ PHONGBAN, những dòng dữ liệu cũ sẽ được Delete đi và được ghi vào bảng Deleted, những dòng dữ liệu mới sẽ được Insert vào và được ghi vào bảng Inserted.

### 7.1.2. Phân loại

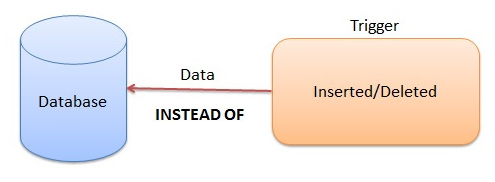
Dựa trên cách thức hoạt động, Trigger được phân làm hai loại:

*Trigger After/For*: Trigger được thực thi sau khi thực hiện thao tác Insert/ Delete các dòng dữ liệu trong Table, gọi là reactive. Khi tạo Trigger nếu không chỉ định rõ thì mặc định Trigger sẽ là After/For Trigger. Trigger After/For chỉ có thể cài đặt trên Table.



Hình 7. 1. Cơ chế hoạt động của Trigger After/for

*Triggers Instead Of*: kiểm tra trứơc khi thực hiện các thao tác Insert/Delete trên bảng, gọi là proactive. Trigger Instead Of sẽ bỏ qua các thao tác cập nhật CSDL, thay vào đó thực hiện việc lưu dữ liệu vào bảng Inserted khi có thao tác Insert, lưu dữ liệu vào bảng Deleted đối với thao tác Delete. Vì vậy, dữ liệu mới được thêm vào sẽ chỉ chứa trong bảng Inserted, dữ liệu cũ bị xóa đi sẽ chứa trong bảng Deleted nhưng vẫn còn tồn tại trong Database ở bảng mà Trigger Instead Of áp dụng. Trigger Instead Of có thể cài đặt cho cả Table và View (khung nhìn) nhưng thường được dùng để cập nhật View.



Hình 7. 2. Cơ chế hoạt động của Trigger instead of

## 7.2. Quản lý Trigger:

### 7.2.1. Tạo Trigger:

Để tạo Trigger, ta có thể click chuột phải vào Database đang làm việc, chọn mục Programmability, chọn Database Trigger, tiến hành tạo mới New Database Trigger theo đoạn script mẫu. Hoặc ta có thể tự xây dựng nội dung Trigger theo cú pháp sau:

**CREATE TRIGGER *Trigger\_Name***

**ON *Table|View* [WITH ENCRYPTION]**

**{ FOR | AFTER| INSTEAD OF }**

**{ [INSERT] [,] [UPDATE] [,] [DELETE] }**

**AS *Sql\_Statement* [ . . . *n* ]**

Trong đó:

**Trigger\_Name**: Tên của Trigger được lưu trong Database để quản lý.

***Table|View***: Tên Table, View mà Trigger sẽ áp dụng.

**WITH ENCRYPTION**: Lựa chọn cho phép mã hóa nội dung Trigger.

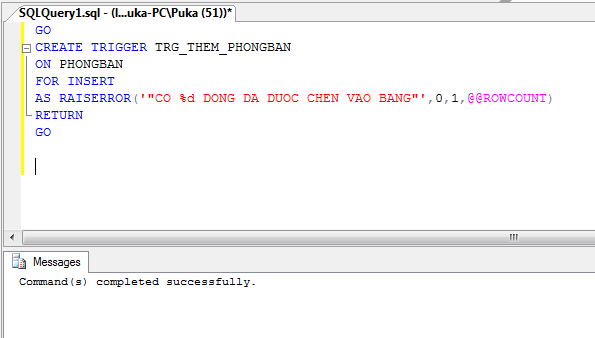
**FOR| AFTER| INSTEAD OF**: Loại của Trigger.(Xem lại mục 1.2 của chương này.)

**INSERT, UPDATE, DELETE**: Liệt kê các thao tác mà Trigger sẽ kiểm tra để thực thi.

***Sql\_Statement***: Nội dung của Trigger.

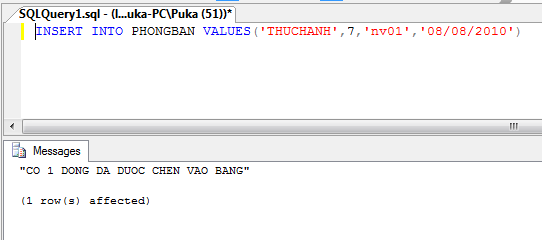
**Lưu ý**, câu lệnh tạo Trigger phải đặt trong một bó lệnh.

**Ví dụ:** Tạo Trigger cho thao tác Insert dữ liện trên bảng PHONGBAN



Hình 7. 3. Minh họa trigger cho thao tác Insert

Tiến hành thêm một dòng dữ liệu mới



Hình 7. 4. Thêm một dòng dữ liệu

### 7.2.2. Nguyên tắc hoạt động và các bước cài đặt Trigger

#### 7.2.2.1. Nguyên tắc hoạt động:

Cả hai loại Trigger For/After và Trigger Instead Of về bản chất đều là một thủ tục đặc biệt, được tự động thực thi khi có các thao tác cập nhật dữ liệu, tuy nhiên có những điểm phân biệt cần lưu ý giữa hai loại Trigger trên như sau:

|  |  |
| --- | --- |
| ***For/After*** | ***Instead of*** |
| Chỉ áp dụng cho Table | Áp dụng cho Table, View |
| Có thể định nghĩa nhiều Trigger trên một thao tácInsert/Delete/Update | Chỉ định nghĩa một Trigger trên một thao tác Insert/Delete/Update |
| Trigger được thực thi sau khi kiểm tra điều kiện ràng buộc toàn vẹn. Thao tác Insert/Delete/Update được thực thi xong rồi nội dung Trigger mới được thực thi. | Trigger được thực thi trước khi kiểm tra điều kiện ràng buộc toàn vẹn. Nội dung của Trigger sẽ được thực thi thay cho thao tác Insert/Delete/Update. |
|  | Không xây dựng được trên Table có áp dụng Cascade Delete/ Update |

#### 7.2.2.2. Các bước cài đặt Trigger

Để tiến hành cài đặt Trigger, ta thực hiện theo các bước sau:

**Bước 1:** Xác định ràng buộc toàn vẹn. Khi xác định ràng buộc toàn vẹn, ta cần xác định ba thành phần: nội dung, bối cảnh và bảng tầm ảnh hưởng (xem chương 3).

**Bước 2:** Tiến hành cài đặt Trigger. Lựa chọn loại Trigger cho phù hợp.

**Bước 3:** Kiểm tra Trigger. Tiến hành một số thao tác cập nhật có thể để kiểm tra tính đúng đắn của Trigger đã cài đặt.

**Một số điểm cần lưu ý khi cài đặt Trigger:**

* Phải hiểu rõ nội dung ràng buộc toàn vẹn để xây dựng nội dung xử lý của Trigger cho chính xác. Nắm vững bối cảnh của ràng buộc toàn vẹn giúp xác định Trigger cần cài đặt trên Table/View nào. Nắm vững bảng tầm ảnh hưởng giúp xác định Trigger cần cài đặt cho thao tác cập nhật dữ liệu nào, Insert, Delete hay Update.
* Trigger After/For sẽ thực thi sau khi thao tác cập nhật dữ liệu được thực thi, vì vậy nếu thao tác cập nhật dữ liệu vi phạm ràng buộc toàn vẹn ta phải hủy thao tác đó trong nội dung của Trigger. Nếu thao tác cập nhật dữ liệu xảy ra trên nhiều bộ dữ liệu và chỉ một số bộ dữ liệu trong đó làm vi phạm ràng buộc toàn vẹn thì việc hủy thao tác sẽ khiến các nội dung cập nhật thòa ràng buộc toàn vẹn cũng bị hủy theo, đây là nhược điểm của Trigger After/For. Trong khi đó Trigger Instead Of được thực thi thay thế cho thao tác cập nhật dữ liệu, dữ liệu cập nhật sẽ được quản lý trên Table Inserted và Table Deleted, ta có thể kiểm tra và chọn ra những bộ dữ liệu không vi phạm ràng buộc toàn vẹn và cập nhật trên database, tuy nhiên việc nội dung Trigger Instead Of thường phức tạp hơn Trigger After/For. Ngoài ra Trigger Instead Of còn được sử dụng để cài đặt cho cả Table và View trong khi Trigger After/For chỉ được cài đặt cho Table.
* Lưu ý trước khi thực hiện các thao tác cập nhật để kiểm tra tính đúng đắn của Trigger cần Back up Database. Nếu phát hiện có sai sót trong nội dung của Trigger, cần tiến hành cập nhật và chỉnh sửa.

### 7.2.3. Sửa, Xóa một Trigger

Để sửa nội dung một Trigger ta thực hiện cú pháp giống như cú pháp tạo một Trigger, tuy nhiên đổi từ khóa CREATE thành ALTER.

**ALTER TRIGGER *Trigger\_Name***

**ON *Table|View* [WITH ENCRYPTION]**

**{ FOR | AFTER| INSTEAD OF }**

**{ [INSERT] [,] [UPDATE] [,] [DELETE] }**

**AS *Sql\_Statement* [ . . . *n* ]**

Để xóa một Trigger ta sử dụng cú pháp

**DROP TRIGGER Trigger\_Name**

Trong một số trường hợp, khi ta thực hiện thao tác cập nhật dữ liệu trên nhiều record và ta có thể kiểm soát được việc cập nhật không vi phạm ràng buộc toàn vẹn, Trigger có thể làm tăng khối lượng xử lý của hệ thống, ví dụ ta muốn import 100 record mới vào Table thì tốt nhất trong trường hợp là nên vô hiệu hoá các Trigger và kiểm tra lại ràng buộc toàn vẹn sau.

Câu lệnh ALTER TABLE cho phép enable và disable Trigger:

**ALTER TABLE *table\_name***

**{ ENABLE | DISABLE } TRIGGER**

**{ ALL | *trigger\_name* [ ,...*n* ] }**

**Ví dụ: ALTER TABLE PHONGBAN DISABLE TRIGGER ALL**

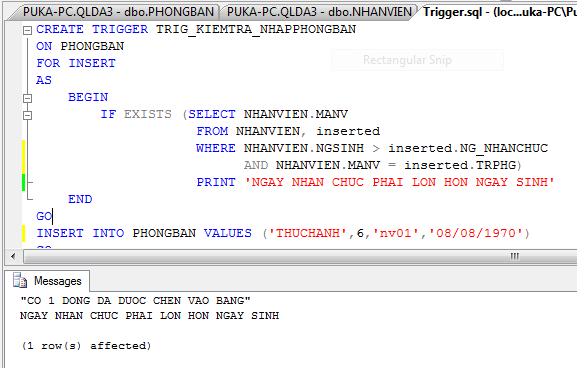
## 7.3. Một số ví dụ về Trigger

**Ví dụ 1:** Xây dựng Trigger kiểm tra “Ngày nhận chức của Trưởng phòng phải lớn hơn Ngày sinh của Trưởng phòng đó” cho thao tác Insert Table PHONGBAN.

Trước tiên, ta phân tích yêu cầu để cài đặt Trigger. Với nội dung Trigger được nêu ra trong ví dụ là thao tác Insert Table PHONGBAN, bối cảnh ta cần cài đặt ràng buộc toàn vẹn bằng Trigger ở đây chính là ở hai quan hệ PHONGBAN và NHANVIEN. Bảng tầm ảnh hưởng như sau:

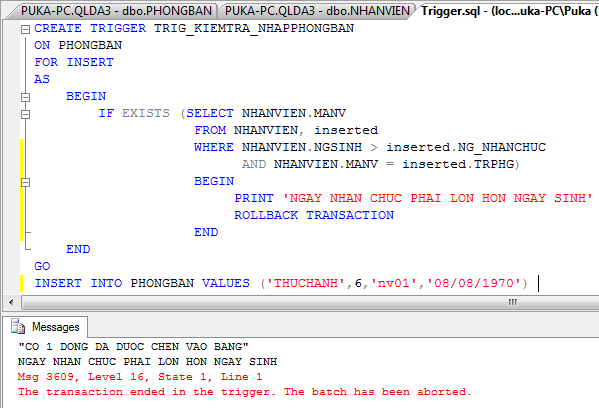
|  |  |  |  |
| --- | --- | --- | --- |
|  | INSERT | DELETE | UPDATE |
| **NHANVIEN** | - | - | + |
| **PHONGBAN** | + | - | + |

Như vậy để cài đặt thỏa ràng buộc toàn vẹn ở trên, ta cần cài đặt Trigger cho Table NHANVIEN ở thao tác DELETE, UPDATE và table PHONGBAN ở thao tác INSERT, UPDATE. Tuy nhiên yêu cầu của ví dụ đã nêu rõ cài đặt Trigger cho thao tác thêm mới một phòng ban nên nội dung Trigger như bên dưới.



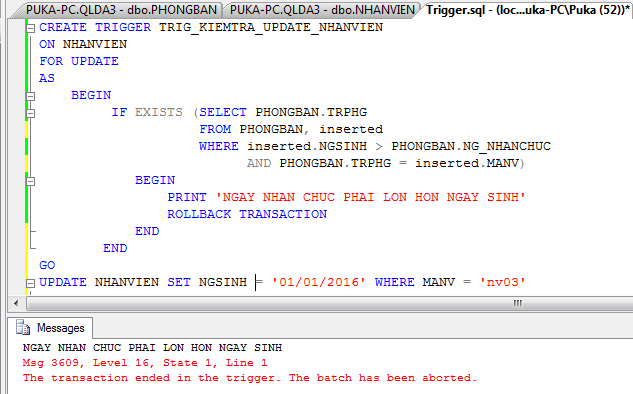
Hình 7. 5. Cài đặt trigger minh họa thêm 1 nhân viên

Xét nội dung Trigger ở trên, thao tác Insert đã thêm một phòng ban mới do ‘nv01’ làm trưởng phòng. Khi đó, các Trigger tương ứng đã được gọi lên thực thi. Vì đây là Trigger After/For nên thao tác Insert đã được thực hiện trước khi Trigger được thực thi, vì vậy để ngăn thao tác Insert diễn ra ta thêm dòng lệnh Rollback Transaction vào nội dung Trigger như sau:



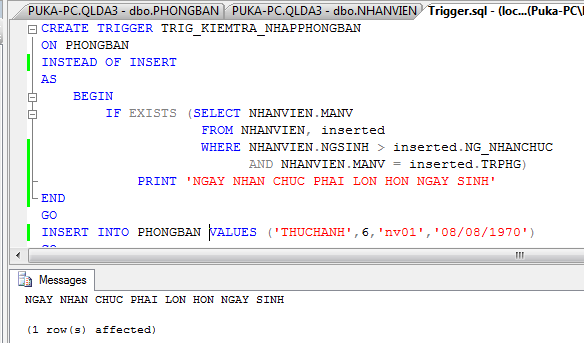
Hình 7. 6. Cài đặt trigger có dùng lệnh ROLLBACK TRANSACTION

Khi đó thao tác Insert sẽ bị hủy, nội dung dữ liệu mới sẽ không được ghi nhận vào Database. Nếu cần cài đặt luôn cho thao tác Update trên Table PHONGBAN thì ta thay đổi cụm từ FOR INSERT thành FOR INSERT, UPDATE. Vì thao tác Update về bản chất cũng chính là bao gồm thao tác Insert - thêm dữ liệu mới vào bảng Inserted và thao tác Delete - thêm dữ liệu cũ vào bảng Deleted. Ngoài ra nếu muốn cài đặt ràng buộc toàn vẹn trên hoàn chỉnh ta sẽ cần cài đặt tiếp nội dung ràng buộc đó trên quan hệ NHANVIEN với thao tác Update cùng nội dung Trigger tương tự.



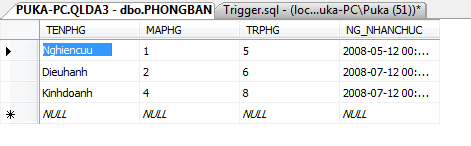
Hình 7. 7. Minh họa cài đặt trigger cho thao tác Update

Ngoài ra, ta có thể sử dụng Trigger Instead Of để cài đặt ràng buộc toàn vẹn trên cho thao tác Insert trên Table PHONGBAN như sau:



Hình 7. 8. Minh họa cài đặt trigger instead of cho hành động insert

Với Trigger Instead Of, thao tác Insert sẽ vẫn được thực thi nhưng dữ liệu mới sẽ chỉ được Insert vào bảng Inserted, dữ liệu trên Database không có bộ dữ liệu về phòng ban mới, do đó Trigger thêm một bộ dữ liệu mới vào Table PHONGBAN không được thực thi.



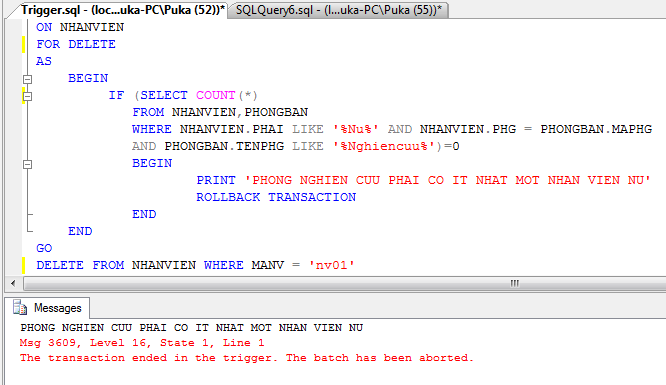
**Ví dụ 2:** Xây dựng Trigger để kiểm tra ràng buộc toàn vẹn: “Phòng ban Nghiencuu phải có ít nhất một nhân viên Nữ” cho thao tác Delete trên Table NHANVIEN. (Giả sử lấy bối cảnh nhân viên ‘nv01’ đang là nhân viên Nữ duy nhất của phòng Nghiencuu)

Bối cảnh để ta cài đặt ràng buộc toàn vẹn ở đây là hai quan hệ PHONGBAN và NHANVIEN với bảng tầm ảnh hưởng cụ thể như sau:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **INSERT** | **DELETE** | **UPDATE** |
| **NHANVIEN** | - | + | + |
| **PHONGBAN** | - | - | + |

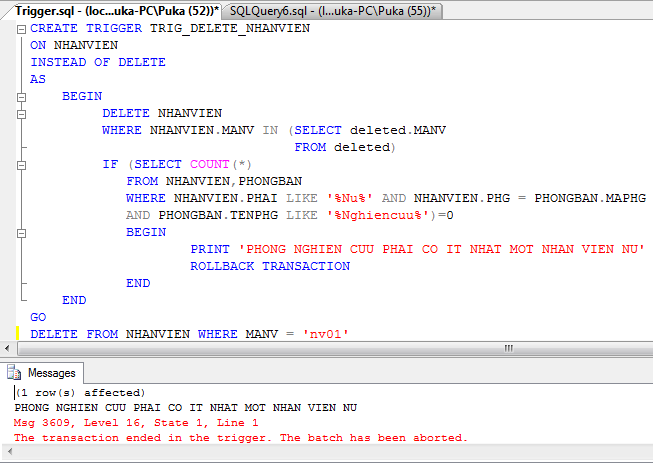
Trên quan hệ NHANVIEN, thao tác cần cài quan tâm đến khi cài đặt ràng buộc toàn vẹn ở ví dụ 2 chính là thao tác Delete và thao tác Update. Quan hệ PHONGBAN, thao tác cần quan tâm khi cài đặt ràng buộc toàn vẹn chính là thao tác Update. Ở đây ví dụ 2 chỉ xét ta cài đặt Trigger cho thao tác Delete trên quan hệ NHANVIEN. Nếu muốn cài đặt hoàn chỉnh ràng buộc trên, ta cần phân tích và cài đặt luôn cho cả thao tác Update trên Table NHANVIEN, và thao tác Update trên Table PHONGBAN như trong ví dụ 1.

Nội dung Trigger bên dưới chỉ xét việc cài đặt trên thao tác Delete của Table NHANVIEN, ta có thể thay cụm từ FOR DELETE thành FOR DELETE, UPDATE để bổ sung thêm việc cài đặt cho thao tác Update (riêng thao tác Update trên Table PHONGBAN ta cần cài đặt Trigger khác).



Hình 7. 9. Minh họa cài đặt trigger cho hành động Delete

Lưu ý: Nếu ta xây dựng Trigger Instead Of để kiểm tra ràng buộc được nêu trong ví dụ trên, khi đó nội dung của Trigger sẽ được thực hiện thay thế cho thao tác Delete trên Table NHANVIEN, nghĩa là điều kiện kiểm tra số nhân viên Nữ trên Table NHANVIEN sẽ trở thành: “Chỉ còn có một nhân viên Nữ của phòng ban Nghiên cứu và nhân viên được thêm mới vào bảng Deleted chính là nhân viên đó”, điều kiện kiểm tra này khó cài đặt. Khi đó để cài đặt Trigger Instead Of cho trường hợp này ta có thể thực hiện như sau:



Hình 7. 10. Minh họa cài đặt trigger instead of cho hành động Delete

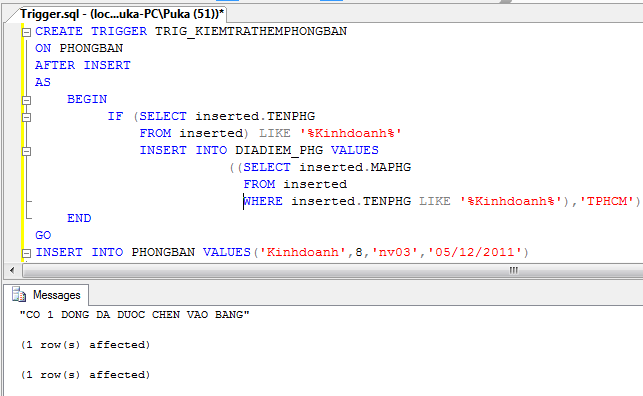
Khi thao tác Delete được thực hiện trên Table NHANVIEN, Trigger TRIG\_DELETE\_NHANVIEN sẽ được gọi thực thi thay thế cho thao tác Delete, lúc này câu lệnh Delete của Trigger sẽ được thực thi, sau đó nội dung ràng buộc sẽ được kiểm tra bên dưới, khi vi phạm ràng buộc thì Transaction sẽ bị hủy.

**Ví dụ 3:** Xây dựng Trigger kiểm tra ràng buộc toàn vẹn sau: “Địa điểm phòng ban Kinh doanh phải ở thành phố Hồ Chí Minh” đối với thao tác thêm mới một phòng ban.

Bối cảnh cần cài đặt ràng buộc ở đây chính là quan hệ PHONGBAN và quan hệ DIADIEM\_PHG với bảng tầm ảnh hưởng như sau:

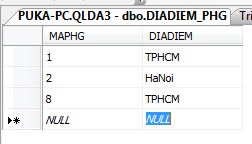
|  |  |  |  |
| --- | --- | --- | --- |
|  | **INSERT** | **DELETE** | **UPDATE** |
| **PHONGBAN** | + | - | + |
| **DIADIEM\_PHONG** | - | - | + |

Với yêu cầu của ví dụ 3, ta sẽ tiến hành cài đặt Trigger cho thao tác Insert trên Table PHONGBAN với nguyên tắc hoạt động như sau: Khi ta thêm mới một phòng ban, nếu phòng ban đó là phòng ban Kinh doanh, Trigger sẽ thực hiện luôn thao tác thêm địa điểm phòng ban là ‘TPHCM’ cho phòng ban vừa thêm. Với nguyên tắc hoạt động như vậy, thao tác Update trên Table PHONGBAN cũng có thể được cài đặt chung với Trigger này vì bản chất thao tác Update là bao gồm thao tác Insert - thêm dữ liệu mới vào bảng Inserted và thao tác Delete - thêm dữ liệu cũ vào bảng Deleted (riêng thao tác Update trên Table DIADIEM\_PHONG ta cần cài đặt Trigger khác).



Hình 7. 11. Minh họa cài đặt trigger after cho hành động Insert

Khi Insert một bộ dữ liệu mới phòng ban Kinh doanh trong Table PHONGBAN, Trigger TRIG\_KIEMTRATHEMPHONGBAN đã được thực thi và thêm mới địa điểm cho phòng ban Kinhdoanh trong Table DIADIEM\_PHG

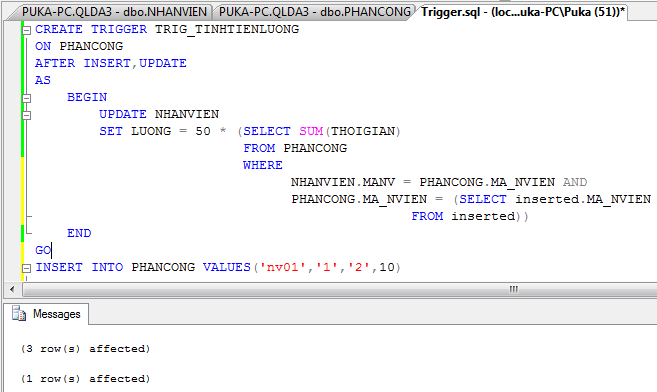


**Ví dụ 4:** Xây dựng Trigger để đảm bảo ràng buộc toàn vẹn như sau: “Lương của nhân viên được tính theo công thức 50 x thời gian phân công làm việc, biết rằng thời gian phân công làm việc được ghi nhận trong Table PHANCONG”.

Bối cảnh của ràng buộc toàn vẹn chính là quan hệ PHANCONG với bảng tầm ảnh hưởng như sau:

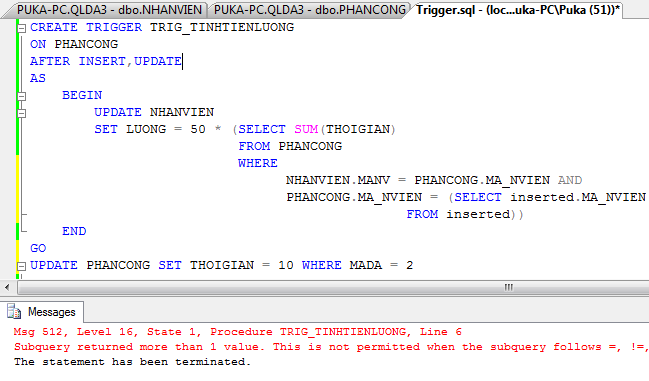
|  |  |  |  |
| --- | --- | --- | --- |
|  | **INSERT** | **DELETE** | **UPDATE** |
| **PHANCONG** | + | + | + |

Đầu tiên ta xây dựng Trigger cho thao tác Insert và Update trên Table PHANCONG



Hình 7. 12. Minh họa cài đặt trigger after cho hành động Update

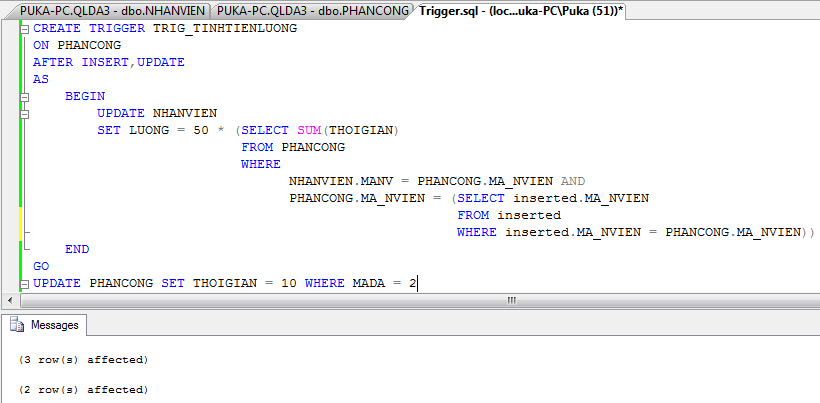
Tuy nhiên khi tiến hành thao tác Update trên Table PHANCONG sẽ phát sinh lỗi



Hình 7. 13. Minh họa cài đặt trigger after cho hành động Insert, Update

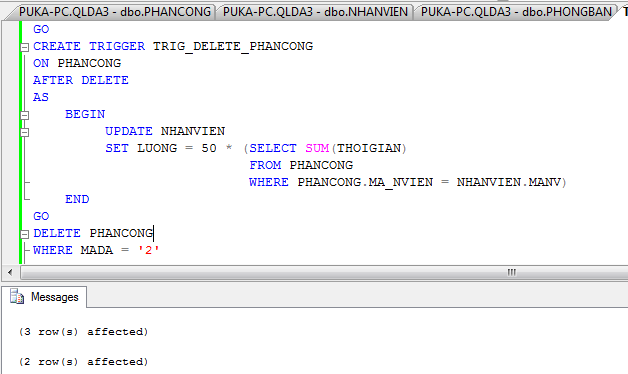
Trong trường hợp này, sau khi cài đặt Trigger xong, khi thao tác Update được thực hiện đã làm cập nhật dữ liệu đến hai dòng dữ liệu trong Table PHANCONG, từ đó làm cho số dòng dữ liệu trong Table Inserted là hai dòng. Vì vậy Sub Querry (SELECT inserted.MA\_NVIEN FROM inserted) sẽ trả về kết quả là một bảng hai dòng dữ liệu, phép so sánh bằng báo lỗi. Đây cũng là một lỗi thường gặp khi lập trình T-SQL. Lưu ý những Trigger ở các ví dụ trên cũng có thể gặp lỗi này nếu thao tác cập nhật dữ liệu xảy ra trên nhiều bộ dữ liệu, khiến cho Table Inserted và Table Deleted cũng xuất hiện các bộ dữ liệu tương ứng, tuy nhiên trong yêu cầu của ví dụ 1,2,3 chỉ xét đến thao tác cập nhật trên một bộ dữ liệu, nên các ví dụ trên ta không xét đến trường hợp này.

Để khắc phục lỗi trên, chúng ta chỉnh sửa nội dung Trigger như sau để có thể áp dụng chính xác cho cả trường hợp có thao tác Update trên nhiều bộ dữ liệu.



Hình 7. 14. Minh họa hiệu chỉnh cài đặt trigger after cho hành động Insert

Để cài đặt hoàn chỉnh ràng buộc ở ví dụ 4, ta phải cài đặt thêm Trigger cho thao tác Delete trên Table PHANCONG.



Hình 7. 15. Minh họa cài đặt trigger after cho hành động Delete

Với cách cài đặt Trigger như trên, ta sẽ không kiểm tra bộ dữ liệu nào đã bị xóa trong Table PHANCONG và thêm vào Table Deleted, từ đó trừ lại tiền lương của nhân viên tương ứng. Mỗi lần có thao tác Delete diễn ra trên Table PHANCONG, toàn bộ Luong của Table NHANVIEN sẽ được tính toán lại. Cách cài đặt sẽ đơn giản hơn và tốc độ thực thi của Trigger cũng nhanh hơn do ít phép kết bảng hơn. Ta cũng có thể áp dụng cho cả hai trường hợp thao tác Insert và Update ở trên bằng cách thay cụm từ AFTER DELETE thành AFTER INSERT,DELETE, UPDATE. Như vậy cùng một nội dung ràng buộc, nếu ta phân tích tốt hơn thì việc cài đặt Trigger sẽ tốt hơn và hiệu quả hơn.

## 7.4. Trigger lồng

Thao tác của một Trigger có thể kéo theo việc thi hành một Trigger khác, các Trigger này đựơc gọi là Trigger lồng nhau. Trigger có thể lồng tối đa 32 cấp. Các Trigger được xem như một đơn vị thi hành Transaction. Do vậy, một Trigger trong dãy Trigger lồng nhau bị lỗi, thi SQL Server sẽ rollback tất cả các action đã thực hiện bởi các Trigger.

Trigger 1

Trigger 2

Gọi

Thực thi

Gọi

Khi Trigger gọi lại chính nó được gọi là Recursive Trigger, để cài đặt được Trigger phải bật option của Database qua câu lệnh:

*sp\_dboption database\_name, ‘rescusive triggers’, True*

**Sửa data**

**trên table 1**

**Gọi**

**Thực hiện**

**Sửa data**

**trên table 1**

**Sửa data**

**trên table 2**

**Gọi**

**Gọi**

**Thực hiện**

**Thực hiện**

# BÀI TẬP CHƯƠNG 7

## CÂU HỎI

**Câu hỏi 1:** Trigger là gì? Phân biệt hai loại Trigger: Trigger After-For và Trigger Instead Of?

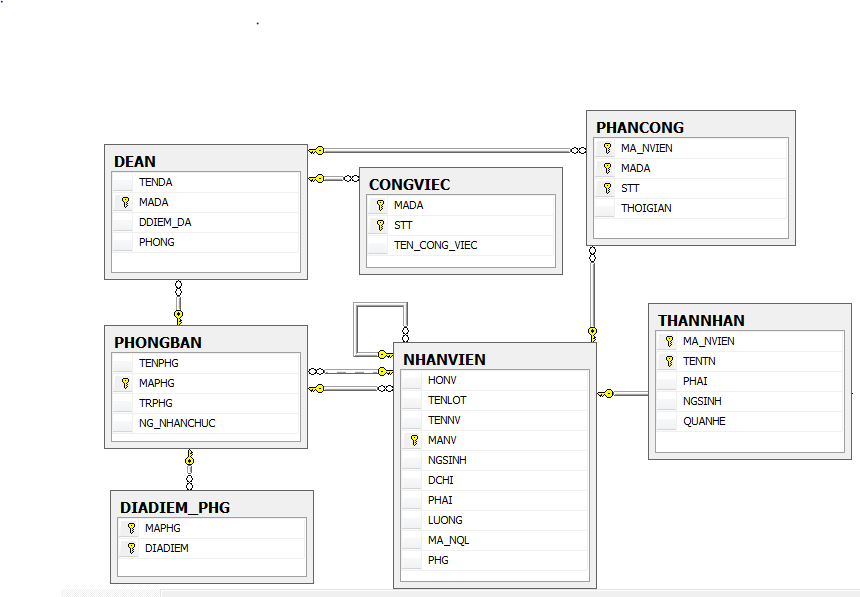
**Câu hỏi 2:** Phân biệt bảng Inserted và bảng Deleted

**Câu hỏi 3:** Trình bày các bước tiến hành cài đặt một Trigger

**Câu hỏi 4**: Trình bày khái niệm về Trigger lồng.

## BÀI TẬP

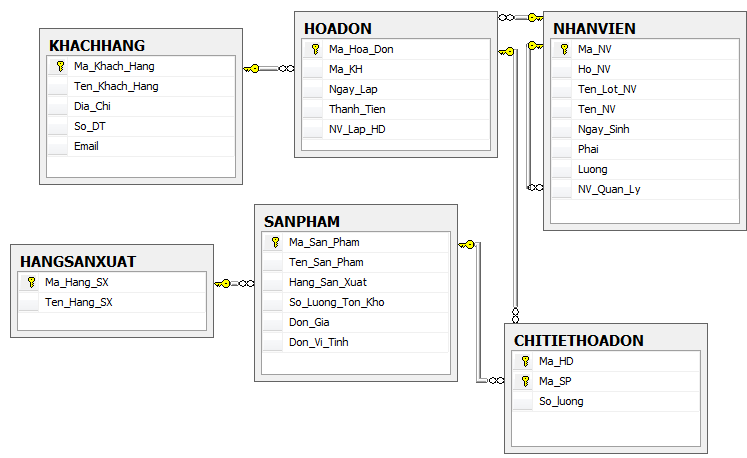
**Sử dụng lược đồ sau để thực hiện bài tập 1 (mô tả chi tiết về lược đồ quan hệ xem lại ở bài tập chương 1)**



**Bài tập 1: Tiến hành cài đặt các Trigger sau:**

1. Thông báo số dòng dữ liệu đã được thêm vào khi có thao tác Insert trong bảng NHANVIEN.
2. Thông báo số dòng dữ liệu đã được xóa khi có thao tác Delete trên bảng PHANCONG.
3. Lương của nhân viên không được vượt quá lương của người quản lý.
4. Ngày nhận chức của trưởng phòng phải lớn hơn ngày sinh.
5. Một nhân viên được phân công tối đa 3 đề án.
6. Một nhân viên được phân công tối đa 30 giờ.
7. Một nhân viên được phân công tối đa 20 giờ cho mỗi đề án
8. Một nhân viên được phân công tối đa 20 giờ cho mỗi đề án, tối đa 3 đề án.
9. Phòng ban chỉ được chủ trì đề án có cùng địa điểm với địa điểm của phòng ban
10. Cập nhật lương cho nhân viên khi có sự cập nhật dữ liệu trong bảng phân công công theo công thức: Lương = Tổng thời gian phân công \* 20 + phụ cấp trưởng phòng, biết phụ cấp trưởng phòng = 100.

**Sử dụng lược đồ sau để thực hiện các bài tập 2 (mô tả chi tiết về lược đồ quan hệ xem lại ở bài tập chương 1)**



**Bài tập 2: Tiến hành cài đặt các Trigger như sau:**

1. Thông báo số dòng dữ liệu được thêm vào khi có thao tác Insert trong bảng SANPHAM
2. Thông báo số dòng dữ liệu được cập nhật vào khi có thao tác Update trong bảng CHITIETHOADON
3. Khi thêm mới một hóa đơn, thông báo cho biết tổng số hóa đơn mà nhân viên vừa mới lập hóa đơn đó đã lập trong ngày.
4. Khi thêm mới một chi tiết hóa đơn, thông báo nếu số lượng tồn kho của sản phẩm còn dưới 30.
5. Tự động cập nhật số lượng tồn kho khi có một chi tiết hóa đơn mới được thêm vào
6. Tự động tính thành tiền cho hóa đơn dựa theo công thức

Thành tiền hóa đơn = Tổng ( Đơn giá \* Số lượng ) với Đơn giá: đơn giá sản phẩm, Số lượng: số lượng sản phẩm trong chi tiết hóa đơn.

1. Một Khách Hàng không được mua một sản phẩm với số lượng quá 50 trong một tuần.
2. Thêm cột Thuong trong bảng NHANVIEN dùng để tính tiền thưởng hoa hồng cuối mỗi năm cho nhân viên, gia tri cot Thuong được tính như sau: Thuong = 30% giá trị các hóa đơn mà nhân viên đó lập trong năm.

# CHƯƠNG 8 CON TRỎ (CURSOR), KHUNG NHÌN (VIEW), CHỈ MỤC (INDEX), GIAO TÁC (TRANSACTION) VÀ KHÓA (LOCK).

# MỤC ĐÍCH VÀ YÊU CẦU

**Mục đích:**

Chương này nhằm cung cấp cho sinh viên những kiến thức chung về:

* Những kiến thức cơ bản về Con trỏ, Khung nhìn, Chỉ mục, Giao tác và Khóa.
* Phương pháp giải quyết các vấn đề về quản trị cơ sở dữ liệu bằng Con trỏ, Khung nhìn, Chỉ mục, Giao tác và Khóa.

**Yêu cầu:**

Sau khi học xong chương này sinh viên có khả năng:

* Sử dụng thành thạo các lệnh tạo Con trỏ, Khung nhìn và Chỉ mục.
* Thao tác với Chỉ mục, Khung nhìn trong các tình huống cụ thề nhằm tăng tốc độ truy vấn trên cơ sở dữ liệu

# NỘI DUNG

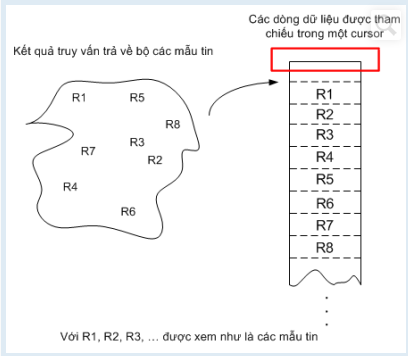
## 8.1. Con trỏ (Cursor)

### 8.1.1. Khái niệm về Cursor

Phần lớn các cơ sở dữ liệu 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 có 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 con trỏ (Cursor).

Kiểu dữ liệu Cursor giống như một cuốn sổ danh bạ chứa thông tin liên lạc của các khách hàng giao dịch 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, ta chỉ đang theo dõi thông tin một khách hàng. 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 8. 1. Cơ chế duyệt mẫu tin của một cursor

Chi tiết các bước thực hiện khi sử dụng biến kiểu dữ liệu cursor trong Transaction -SQL. Để làm việc với biến có kiểu Cursor, ta phải thực hiện từng bước như sau:

* Định nghĩa biến kiểu cursor bằng lệnh DECLARE.
* Sử dụng lệnh OPEN để mở ra cursor đã định nghĩa trước đó.
* Đọc và xử lý trên từng dòng dữ liệu bên trong Cursor.
* Đóng cursor lại bằng lệnh CLOSE và DEALLOCATE.

### 8.1.2. Định nghĩa Cursor

Việc định nghĩa biến có kiểu Cursor trong Transaction-SQL chính là việc chỉ định đến những dòng dữ liệu có bên trong các bảng dữ liệu mà biến Cursor đó sẽ tham chiếu đến. Thông thường trong lệnh định nghĩa biến có kiểu Cursor sẽ chỉ định ra loại của Cursor cho các mục đích sử dụng về sau thuận tiện hơn.

**Cú pháp:**

**DECLARE** CURSOR\_NAME **CURSOR**

**[LOCAL | GLOBAL]**

**[FORWARD ONLY | SCROLL]**

**[STATIC | DYNAMIC | KEYSET]**

**[READ ONLY | SCROLL\_LOCK]**

**FOR Câu\_lệnh\_Select**

**[FOR UPDATE [OF Danh\_sách\_cột]]**

Trong đó:

* **CURSOR\_NAME:** tên của biến kiểu Cursor.
* **Từ khóa LOCAL|GLOBAL:** dùng chỉ định phạm vi hoạt động của biến Cursor hoặc là cục bộ (Local) bên trong một thủ tục, lô (Batch) các lệnh, một Trigger hoặc là toàn cục (Global) bên trong một kết nối. Một biến cursor có tính toàn cục sẽ được phép tham chiếu trong bất kỳ thủ tục nào của kết nối tạo ra biến Cursor đó.
* **Từ khóa FORWARD\_ONLY:** dùng chỉ định việc đọc dữ liệu trong Cursor chỉ theo chiều đi tới mà thôi (duyệt từ mẫu tin đầu tiên đến mẫu tin cuối cùng).
* **Từ khóa SCROLL:** dùng chỉ định việc đọc dữ liệu trong cursor được phép di chuyển tới lui, qua lại các dòng mẫu tin bên trong Cursor tùy thích.
* **Từ khóa STATIC:** dùng chỉ định dữ liệu đọc bên trong Cursor là tĩnh. Khi đó nếu những người dùng khác có các thay đổi ở bên dưới dữ liệu gốc (Base Table) thì các thay đổi đó sẽ không được cập nhật tự động trong dữ liệu của cursor. Bởi vì khi đó dữ liệu trong Cursor chính là dữ liệu của một bảng tạm đã được hệ thống sao chép và lưu trữ trong cơ sở dữ liệu tempdb của hệ thống khi địng nghĩa Cursor.
* **Từ khóa DYNAMIC:** dùng chỉ định dữ liệu bên trong Cursor là động. Khi đó việc cập nhật dữ liệu trong bảng cơ sở (Base Table) bởi những người dùng khác sẽ được cập nhật tự động trong dữ liệu Cursor có kiểu là DYNAMIC.
* **Từ khóa KEYSET:** có hoạt động gần giống với kiểu DYNAMIC, các thay đổi dữ liệu trên các cột không là khóa chính trong bảng cơ sở bởi những người dùng khác sẽ được cập nhật trong dữ liệu Cursor. Tuy nhiên đối với các mẫu tin vừa thêm mới hoặc các mẫu tin đã bị hủy bỏ bởi những người dùng khác sẽ không được hiển thị trong dữ liệu Cursor có kiểu là KEYSET.
* **Từ khóa READ\_ONLY:** dùng chỉ định dữ liệu bên trong Cursor là chỉ đọc nhằm hạn chế việc sửa đổi dữ liệu bên trong Cursor. Khi khai báo Cursor với kiểu dữ liệu là tĩnh (STATIC) thì dữ liệu trong Cursor xem như là chỉ đọc.
* **Từ khóa SCROLL\_LOCK:** dùng chỉ định hệ thống Microsoft SQL Server tự động khóa các dòng mẫu tin cần phải thay đổi giá trị hoặc bị hủy bỏ bên trong bảng nhằm đảm bảo các hành động cập nhật luôn luôn thành công.
* **Câu lệnh SELECT:** dùng để chỉ định nội dung dữ liệu của Cursor. Câu lệnh SELECT trong Cursor không thể chứa các mệnh đề: INTO, COMPUTE, COMPUTE BY.
* **Danh sách các cột cập nhật:** chỉ định danh sách tên các cột sẽ được phép thay đổi giá trị trong Cursor. Mặc định tất cả các cột trong mệnh đề SELECT sẽ được phép thay đổi giá trị nếu dữ liệu Cursor không phải là chỉ đọc.

**Ví dụ 1:** Định nghĩa một biến cursor chứa toàn bộ các dòng dữ liệu bên trong bảng NHANVIEN, các dòng dữ liệu trong cursor cho phép được cập nhật. Sử dụng lệnh khai báo biến cursor như sau:

DECLARE Cur\_NhanVien CURSOR

FORWARD\_ONLY

DYNAMIC

FOR

SELECT \* FROM NHANVIEN

**Ví dụ 2:** Định nghĩa một biến cursor chứa toàn bộ các dòng dữ liệu bên trong bảng NHANVIEN, các dữ liệu trong cursor chỉ được phép đọc và việc đọc dữ liệu trong cursor chỉ theo một chiều tới. Sử dụng lệnh khai báo biến cursor như sau:

DECLARE Cur\_NhanVien CURSOR

FORWARD\_ONLY

STATIC

READ\_ONLY

FOR

SELECT \* FROM NHANVIEN

### 8.1.3. Mở Cursor

Để có thể đọc được các dòng dữ liệu bên trong Cursor, trước tiên cần phải mở Cursor bằng lệnh OPEN. Hoạt động bên trong của lệnh này thực ra là hệ thống sẽ thực hiện câu lệnh truy vấn SELECT đã được chỉ định trong lệnh định nghĩa biến Cursor trước đó. Trong trường hợp nếu Cursor được định nghĩa với kiểu STATIC hoặc KEYSET thì hệ thống sẽ tạo ra bảng tạm chứa các dữ liệu kết quả của lệnh SELECT nằm trong cơ sở dữ liệu tempdb. Cú pháp lệnh OPEN như sau:

**OPEN Tên\_cursor**

Trong đó Tên\_cursor là tên biến Cursor đã được định nghĩa qua câu lệnh DECLARE trước đó.

*Ví dụ:* Để mở các cursor đã định nghĩa ở các ví dụ trên, sử dụng lệnh OPEN như sau:

OPEN Cur\_NhanVien

### 8.1.4. Đọc và xử lý dữ liệu trong Cursor

**FETCH *[NEXT | PRIOR | FIRST | LAST | ABSLUTE n | RELATIVE n]***

**FROM TÊN\_CURSOR**

**[INTO DANH\_SÁCH\_BIẾN ]**

Trong đó:

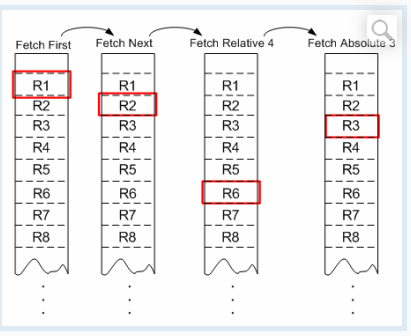
**• Các từ khóa NEXT, PRIOR, FIRST, LAST:** dùng để đọc dữ liệu của dòng dữ liệu kế tiếp (next), dòng dữ liệu phía trước (prior), dòng dữ liệu đầu tiên (first), dòng dữ liệu cuối cùng (last) so với dòng dữ liệu hiện hành bên trong Cursor. Sau khi đọc dữ liệu thành công, dòng dữ liệu hiện hành sẽ bị thay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

**• Tên Cursor:** tên của biến kiểu Cursor đã được định nghĩa trước đó bằng lệnh DECLARE.

**• Danh sách biến:** danh sách tên các biến cục bộ đã được định nghĩa trước đó. Các biến này sẽ lưu trữ các giá trị dữ liệu được đọc từ lệnh FETCH.



Hình 8. 2. Minh họa định hướng duyệt tin của cursor

**Ví dụ 1**: Để đọc dữ liệu cursor bảng NHANVIEN chỉ lọc các nhân viên ở phòng số 5. Sử dụng các lệnh như sau:

*Bước 1:* Khai báo biến cursor:

DECLARE Cur\_NhanVien

KEYSET

FOR

SELECT \*

FROM NHANVIEN

WHERE PHG = 5

*Bước 2:* Mở biến cursor:

OPEN Cur\_NhanVien

*Bước 3:* Đọc dữ liệu

FETCH NEXT FROM Cur\_NhanVien

WHILE @@FETCH\_STATUS = 0

/\*thực hiện đọc dòng kế tiếp trong khi việc đọc dữ liệu thành công\*/

BEGIN

FETCH NEXT FROM Cur\_NhanVien

END

**Ví dụ 2**: Đọc dữ liệu các NHANVIEN có Phái là ‘Nu’ trong bảng NHANVIEN

--Khai báo Cursor

DECLARE CURSOR\_NHANVIEN CURSOR

FOR

SELECT \* FROM NHANVIEN

WHERE PHAI LIKE ‘%Nu%’

--Mở Cursor

OPEN CURSOR\_NHANVIEN

--Đọc dữ liệu CURSOR

FETCH NEXT FROM CURSOR\_NHANVIEN

WHILE @@FETCH\_STATUS = 0

BEGIN

--Đọc những dữ liệu dòng kế tiếp

FETCH NEXT FROM CURSOR\_NHANVIEN

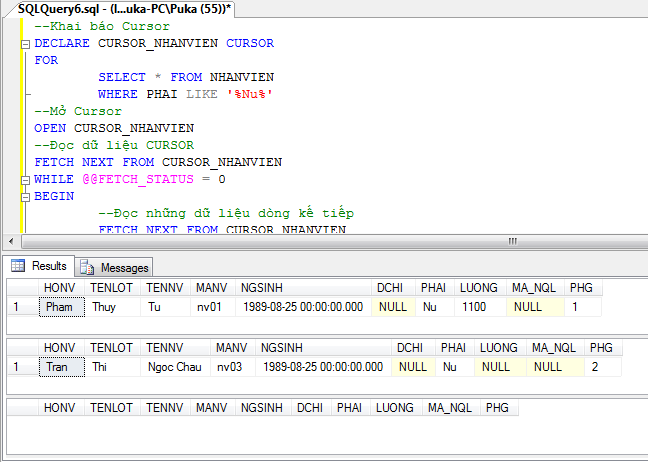
END

--Đóng Cursor

CLOSE CURSOR\_NHANVIEN

DEALLOCATE CURSOR\_NHANVIEN

Kết quả sẽ hiển thị như sau:



Hình 8. 3. Minh họa cài đặt biến Cursor

**Ví dụ 3**: Sử dụng biến Cursor xét duyệt từng dòng dữ liệu trong Table PHANCONG và cập nhật tiền lương cho nhân viên có mã ‘nv03’

--Gán lương = 0 ở những dòng dữ liệu NULL

UPDATE NHANVIEN

SET LUONG = 0

WHERE LUONG IS NULL

--Đoạn lệnh sử dụng Cursor

DECLARE @THOIGIAN INT

DECLARE CURSOR\_PHANCONG CURSOR

FOR

SELECT THOIGIAN

FROM PHANCONG

WHERE PHANCONG.MA\_NVIEN = ‘nv03’

OPEN CURSOR\_PHANCONG

FETCH NEXT FROM CURSOR\_PHANCONG INTO @THOIGIAN

UPDATE NHANVIEN

SET LUONG = LUONG + @THOIGIAN \* 50

WHERE MANV = ‘nv03’

WHILE FETCH\_STATUS = 0

BEGIN

FETCH NEXT FROM CURSOR\_PHANCONG INTO @THOIGIAN

UPDATE NHANVIEN

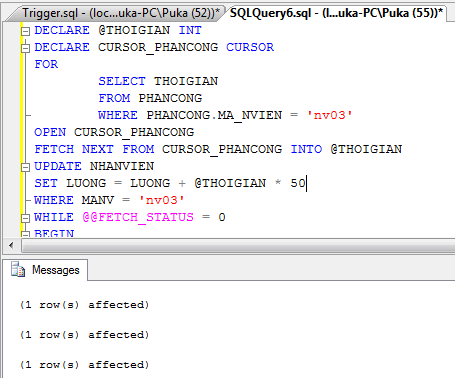
SET LUONG = LUONG + @THOIGIAN \* 50

WHERE MANV = ‘nv03’

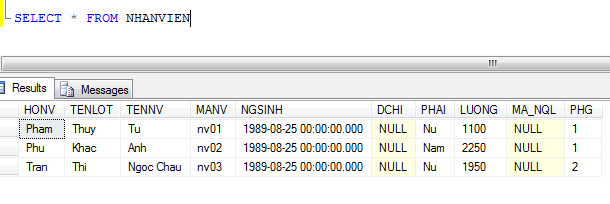
END

CLOSE CURSOR\_PHANCONG

DEALLOCATE CURSOR\_PHANCONG



Hình 8. 4. Minh họa thao tác duyệt mẫu tin và thêm dữ liệu bằng Cursor



Hình 8. 5. Kết quả sau khi thêm dòng dùng biến Cursor

### 8.1.5. Đóng Cursor

**CLOSE** CURSOR\_NAME

**DEALLOCATE** CURSOR\_NAME

Trong đó CURSOR\_NAME là tên biến Cursor đã được định nghĩa và mở ra trước đó. Lệnh CLOSE chỉ là thực hiện hành động giải phóng các dòng dữ liệu tham chiếu bên trong biến Cursor, ta có thể mở lại Cursor mà không cần thiết phải định nghĩa lại biến Cursor bằng lệnh DECLARE. Tuy nhiên việc đọc dữ liệu sẽ không còn là hợp lệ nữa sau khi ta đã ra lệnh CLOSE để đóng Cursor. Lệnh DEALLOCATE để giải phóng thật sự biến Cursor ra khỏi bộ nhớ. Sau khi thực hiện lệnh này, nếu có lệnh nào tham chiếu đến tên Cursor đều sẽ gây ra lỗi.

## 8.2. Khung nhìn (View)

### 8.2.1. Khái niệm

View là một Table ảo mà nội dung được định nghĩa bởi một câu lệnh Select, View cũng bao gồm các cột và các dòng dữ liệu, nhưng không là nơi lưu trữ dữ liệu, các dòng và cột dữ liệu được tham chiếu từ các Table khi định nghĩa View. Về logic, ta có thể coi View như một bảng bình thường, tức là có thể truy vấn, thay đổi dữ liệu, tạo Index trên các cột của View, v.v… Về mặt vật lý, View không chứa dữ liệu. Đúng như tên gọi của nó, View chỉ là một khung nhìn, cấu trúc của View trong CSDL chỉ là một câu lệnh SELECT tham chiếu đến các bảng khác.

Các chức năng chính của View:

* + Chỉ cho user xem những gì cần xem, che dấu nội dung của các Table.
  + Đơn giản hoá việc truy cập dữ liệu.
  + Chọn dữ liệu cần thiết ứng với mỗi user, đảm bảo an toàn dữ liệu.
  + Dùng để Import và Export.

### 8.2.2. Phân loại View

Trong thực tế các loại View mà ta gặp thường có dạng sau

* Là 1 tập các cột trong 1 bảng (SELECT không đủ cột).
* Là 1 tập các dòng trong 1 bảng (SELECT có mệnh đề WHERE).
* Là 1 tập các dòng, các cột trong 1 bảng (Kết hợp của hai trường hợp trên).
* Là 1 tập kết quả của các phép JOIN.
* Là 1 truy vấn tổng hợp, sử dụng các hàm trên một tập giá trị như SUM, COUNT, MIN, MAX, v.v…
* Là tổng hợp của các trường hợp trên.

Ngoài cách phân loại trên, View còn hai dạng đặc biệt ta cần lưu ý là Updatable View và Partition View.

* + ***Updatable View:***

Như đã trình bày ở 2.1 trong chương này, View cũng có thể coi như là một table. Vì vậy có thể thực hiện các thao tác cập nhật dữ liệu (Insert, Update, Delete) trên View. Và khi thay đổi dữ liệu trong View thì những Table liên quan đến View dữ liệu cũng sẽ thay đổi theo.

Vì thế khi làm việc với View, ta nên để ý xem View có phải Updatable View hay không. Nếu có thì nên GRANT quyền cho User chỉ được phép SELECT trên View để tránh trường hợp đáng tiếc xảy ra.

View không phải là Updatable trong một số tình huống sau:

* Trong View sử dụng các hàm như SUM, COUNT, MIN, MAX, v.v…
* Sử dụng các mệnh đề UNION, UNION ALL, CROSSJOIN, INTERSECT, EXCEPT, v.v….
  + ***Partition View:***

Nếu View có dạng sau thì được gọi là Partition View

SELECT <select\_list1>

FROM T1

UNION ALL

SELECT <select\_list2>

FROM T2

UNION ALL

...

SELECT <select\_listn>

FROM Tn

Trong đó T1, T2,…Tn là các bảng có thể ở nhiều Server khác nhau.

### 8.2.3. Tạo View

Cú pháp:

CREATE VIEW [<db\_name>.][<owner>.]view\_name [(column[ ,...n ])]

[WITH {ENCRYPTION|SCHEMABINDING|

VIEW\_METADATA}[ ,...n]]

AS select\_statement [ WITH CHECK OPTION ]

Trong đó:

* + Column: tên cột được dùng trong trường hợp cột được phát sinh từ một biểu thức, hàm.
  + Select\_statement: câu lệnh Select dùng định nghĩa View.

Câu lệnh Select trong View không được chứa:

* Mệnh đề ORDER BY.
* Mệnh đề COMPUTE và COMPUTE BY.
* Các table tạm không được tham chiếu trong View vì vậy không được dùng SELECT INTO trong View.

**Ví dụ:**

GO

CREATE VIEW ViewNhanvienNu

AS

SELECT NHANVIEN.MANV,NHANVIEN.HONV +' '

+ NHANVIEN.TENLOT +' '

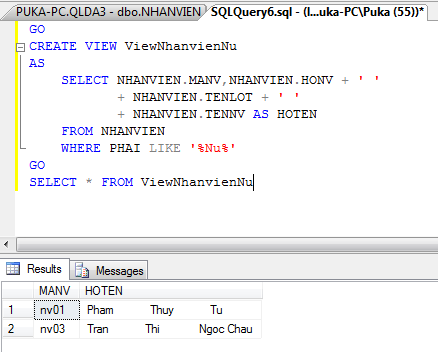
+ NHANVIEN.TENNV AS HOTEN

FROM NHANVIEN

WHERE PHAI LIKE'%Nu%'

GO

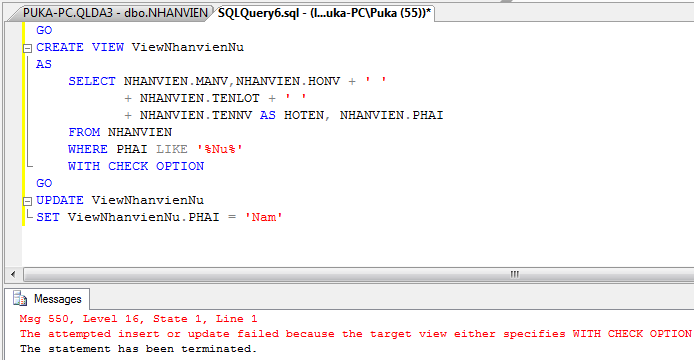
Kết quả như sau:



Hình 8. 6. Minh họa cài đặt View

* ***With check option:*** buộc các lệnh sửa đổi dữ liệu trên View phải thỏa tập điều kiện bên trong câu lệnh Select định nghĩa View.

Ví dụ:



Hình 8. 7. Minh họa cài đặt View có tùy chọn With check option

Câu lệnh Select định nghĩa View có điều kiện Phái là Nữ tuy nhiên câu lệnh Update bên dưới lại thay đổi Phái trong View thành Nam nên báo lỗi Check Option.

* ***With encryption:*** mã hoá cậu lệnh Create View trong bảng hệ thống Syscomments
* ***Thuộc tính shemabinding****:* khi một View được tạo với WITH SCHEMABINDING thì những table cơ bản không được xoá và hiệu chỉnh.

### 8.2.4. Quản lý View

*Xóa View:*

**DROP VIEW** VIEW\_NAME

*Đổi tên view:*

**sp\_rename** OLD**\_**VIEW\_NAME, NEW\_VIEW\_NAME

*Kiểm tra view:*

**sp\_helptext** VIEW\_NAME

*Chỉnh sửa view:*

**ALTER VIEW** VIEW\_NAME (COLUMN\_NAME)

**AS** SELECT\_STATEMENT

Để hiển thị thông tin các View trong Database ta dùng lện sau:

**SELECT** \*

**FROM** INFORMATION\_SCHEMA.TABLES

**WHERE** TABLE\_TYPE = ‘VIEW’

|  |  |  |
| --- | --- | --- |
| ***System View*** | ***Based  on system table*** | ***Công dụng*** |
| *information\_schema.tables* | sysobjects | Tên các view |
| *information\_schema.view\_table\_usage* | sysdepends | Tên các table nguồn dùng định nghĩaView. |
| *information\_schema.views* | syscomments | Định nghĩa View |
| *information\_schema.view\_column\_usage* | syscolumns | Tên các cột dùng trong View |

## 8.3. Chỉ mục (Index)

### 8.3.1. Khái niệm

Index, hay "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, như vậy cột nào được sắp Index thì các giá trị trong cột đó sẽ được sắp xếp trong một cấu trúc Table theo một trật tự dựa vào dữ liệu của Table đó, giúp cho việc truy xuất dữ liệu trở nên nhanh hơn. Điều này tương tự như khi tra từ điển, nếu từ điển không sắp xếp các từ theo thứ tự như: A--->AA--->AB.... thì sẽ mất rất nhiều thời gian để tra một từ.

Trong SQL Server, Index giúp tăng tốc truy xuất dữ liệu, giống như khi tra từ điển, hệ thống cần tìm kiếm thông tin trong hàng ngàn dữ liệu nên việc sắp xếp là cần thiết, mặt khác, do việc sắp xếp các dữ liệu là duy nhất (như trong từ điển không thể có hai từ như nhau vì như vậy rất khó sắp xếp) nên có thể nói rằng Index cũng tăng tính duy nhất trong CSDL. Cuối cùng, như đã biết các câu lệnh JOIN, ORDER BY, GROUP BY cũng yêu cầu việc truy xuất dữ liệu liên tục nên Index đồng thời cũng tăng tốc cho việc này.

Tuy nhiên, Index cũng có những nhược điểm.

* Nhược điểm đầu tiên là việc tạo Index rất tốn thời gian, đối với các project nhỏcó thể vấn đề thời gian chưa đáng kể, nhưng nếu xét trên một CSDL lớn thì việctạo Index tốn rất nhiều chi phí thời gian.
* Nhược điểm thứ hai, thao tác Insert trên Table có Index sẽ tốn thời gian hơn khi không có Index (việc này giống như khi ta thêm một từ mới vào từ điển, nếu từ điển không được sắp xếp theo thứ tự, ta chỉ việc thêm từ mới vào cuối từ điển, nhưng nếu từ điển đã được sắp xếp theo thứ tự, ta cần tìm đúng vị trí để thêm từ mới vào).Vì vậy nếu một Table có tần suất thao tác Insert thường xuyên, việc cài đặt Index sẽ tạo ra gánh nặng khi thực thi thao tác Insert trên Table đó.
* Nhược điểm cuối cùng, do Index là sắp xếp các dữ liệu nên nếu có sự thay đổi thì index cũng được update theo.

### 8.3.2. Phân loại:

Index trong CSDL có hai loại: Clustered Index và Non-Clustered Index.

Clustered Index: thường được tự tạo ra khi bảng có Primary Key. Do Primary Key đã duy trì độ duy nhất dữ liệu của cột, nên có thể nói Clustered Index chính là Unique Index. Trong Clustered Index, các dữ liệu trong Table được sắp xếp một cách vật lý, tức là trong Clustered Index, dữ liệu sắp xếp đúng theo bảng chữ cái. Cách làm việc của Clustered Index là dựa trên ROOT PAGE, khi cần truy xuất, hệ thống sẽ tìm đến địa chỉ cần tìm trong bảng SYSINDEXES. Ví dụ trong Root page có hai cột, một cột chứa dữ liệu trong cột Index, cột còn lại tham chiếu đến số dòng trong Table, giả sử cột Index có giá trị là A (tham chiếu đến dòngtrong Table là dòng 1) và D (tham chiếu đến dòng trong Table là dòng 6), (việc này do Index quyết định), khi ta cần tìm "C", như vậy theo hệ chữ cái, hệ thống sẽ biết rằng C trong khoảng giữa A và D, hệ thống từ đó dò từ dòng thứ 2 đến dòng thứ 5 trong Table, từ đó truy xuất được dữ liệu cần biết. Trong một bảng chỉ được có duy nhất một Clustered Index.

Non-Clustered Index: khác với Clustered Index, Non-Clustered Index không sắp xếp dữ liệu theo một trật tự vật lý như Clustered mà là ngẫu nhiên trong bảng thông tin, miễn sao nằm trong một logic do Index qui định. Trong một Table có thể chứa đến 249 Non-Clustered Index. Cách hoạt động tương tự như Clustered Index, tuy nhiên khi truy xuất đến bảng thông tin cuối thì thông tin không được sắp xếp theo trật tự.

### 8.3.3. Tạo Index

Cú pháp:

Tạo Clustered Index:

CREATE CLUSTERED INDEX index\_name ON dbo.Tblname(Colname1, Colname2...)

Tạo Non – Clustered Index:

CREATE NON - CLUSTERED INDEX index\_name ON dbo.Tblname(Colname1, Colname2...)

**Ví dụ**:

Cho đoạn lệnh SQL như sau:

SELECT \*

INTO NHANVIEN\_TAM

FROM NHANVIEN

WHERE LUONG + 2 > 500

GO

SELECT MANV, HONV, TENLOT, TENNV

FROM NHANVIEN

WHERE LUONG = 1950

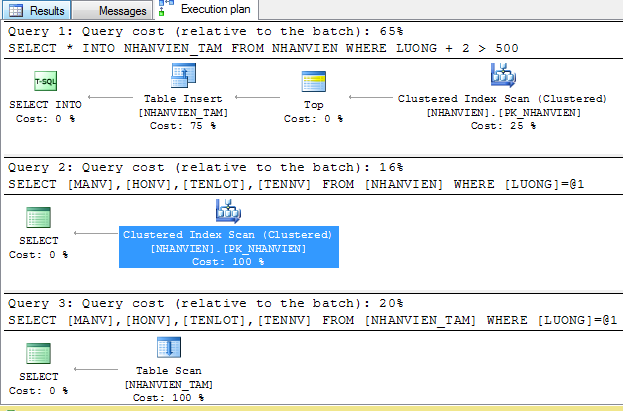
GO

SELECT MANV, HONV, TENLOT, TENNV

FROM NHANVIEN\_TAM

WHERE LUONG = 1950

Kết quả Execution Plan:



Hình 8. 8. Màn hình chi tiết tiến trinh Excution plan

Với hai Table có dữ liệu như nhau là Table NHANVIEN và Table NHANVIEN\_TAM, chi phí cho truy vấn trên Table NHANVIEN\_TAM không có Index là cao hơn. (20% > 12%).

## 8.4. Giao tác (Transaction) và Khóa (Lock)

### 8.4.1. Khái niệm về Transaction

Transaction (giao tác) được dùng để đảm bảo tính toàn vẹn dữ liệu khi xảy ra cập nhật dữ liệu với các thao tác Insert, Delete, Update. Khi một Transaction bao gồm nhiều lệnh cập nhật, nó đảm bảo tất cả các cập nhật đều được thực hiện thành công, hoặc trong trường hợp một lệnh gặp sự cố toàn bộ Transaction bị hủy bỏ. Khi đó dữ liệu trở về trạng thái như trước khi xảy ra Transaction. Nói cách khác Transaction ngăn chặn tình huống dữ liệu được cập nhật nửa chừng, trong đó một phần được cập nhật còn một phần bị bỏ qua.

**Ví dụ về Transaction:**

-- tạo bảng với ràng buộc cột i không được chứa giá trị 2

**CREATE TABLE #t1(i INT, CONSTRAINT ck1 CHECK (i<>2) )**

 -- dùng giá trị mặc định XACT\_ABORT = OFF

-- **SET XACT\_ABORT OFF**

**BEGIN TRAN**

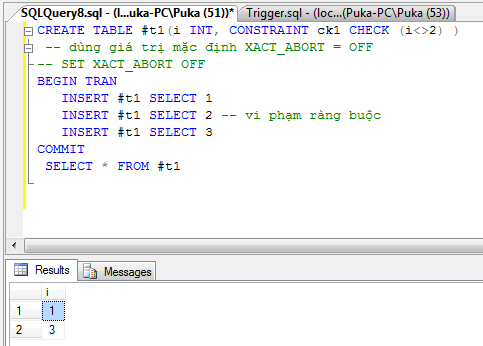
**INSERT #t1 SELECT 1**

**INSERT #t1 SELECT 2** -- vi phạm ràng buộc

**INSERT #t1 SELECT 3**

**COMMIT**

**SELECT \* FROM #t1**



Hình 8. 9. Minh họa cài đặt Transaction

Như vậy trong Transaction trên, lệnh Insert thứ hai gây ra lỗi nhưng lệnh thứ ba vẫn tiếp tục được thực hiện, và Transaction vẫn kết thúc thành công. Kết quả là bảng vẫn có hai bản ghi từ lệnh Insert thứ nhất và thứ ba. Nếu ta chọn SET XACT\_ABORT ON thì Table #t1 không có bản ghi nào vì toàn bộ Transaction đã bị hủy bỏ. Chính xác ra là các lệnh phía sau lệnh gây ra lỗi không được thực hiện tiếp, còn các lệnh thực hiện trước đó bị quay lui (ROLLBACK) trở lại.

### 8.4.2. Các lệnh thường sử dụng trong Transaction

Để xây dựng một Transaction, ta có thể sử dụng mẫu chuẩn như sau của SQL Server:

SET XACT\_ABORT ON

BEGIN TRAN

BEGIN TRY

-- lệnh 1

-- lệnh 2 …

COMMIT

END TRY

BEGIN CATCH

ROLLBACK

DECLARE @ErrorMessage VARCHAR(2000)

SELECT @ErrorMessage = 'Lỗi: ' + ERROR\_MESSAGE()

RAISERROR(@ErrorMessage, 16, 1)

END CATCH

Trong Transaction, thường xuất hiện các lệnh sau để quản lý tốt Transaction:

* **Lệnh Commit**: Lệnh COMMIT là lệnh điều khiển Transaction được sử dụng để lưu các thay đổi đã được triệu hồi bởi một transaction tới Database.

***Cú pháp***: COMMIT;

* **Lệnh ROLLBACK**: Lệnh ROLLBACK là lệnh điều khiển Transaction được sử dụng để trao trả transaction về trạng thái trước khi có các thay đổi mà chưa được lưu tới Database.

***Cú pháp***: ROLLBACK;

* **Lệnh SAVEPOINT**: Một SAVEPOINT là một điểm (point) trong một Transaction khi ta có thể lùi Transaction về một điểm cụ thể mà không cần lùi Transaction về trạng thái đầu trước khi có thay đổi đó.

***Cú pháp***: SAVEPOINT TEN\_CUA\_SAVEPOINT;

Lùi về một Save Point: ROLLBACK TO TEN\_CUA\_SAVEPOINT;

* **Lệnh RELEASE SAVEPOINT**: Lệnh RELEASE SAVEPOINT được sử dụng để xóa một SAVEPOINT đã tạo.

***Cú pháp***: RELEASE SAVEPOINT TEN\_CUA\_SAVEPOINT;

* **Lệnh SET TRANSACTION**: Lệnh SET TRANSACTION có thể được sử dụng để khởi tạo một Database Transaction. Lệnh này được sử dụng để xác định đặc trưng cho Transaction đó.

***Cú pháp***: SET TRANSACTION [ READ WRITE | READ ONLY ];

### 8.4.3. Khóa (Lock)

Locks là cơ cấu cho phép ngăn ngừa các hành động trên đối tượng có thể gây ra xung đột với những gì đã thực hiện và hoàn thành trên đối tượng trước đó. Khi làm việc trên cơ sở dữ liệu đa người dùng, xung đột giữa nhiều người sử dụng cùng thực hiện là thường xuyên xảy ra. Xử lý đụng độ hay tranh chấp trên đối tượng, chúng ta phải biết khi nào nên khoá (lock) khi nào không thể khoá, và những loại lock nào đang có.

Lock hướng giải quyết 4 vấn đề sau:

* Dirty reads (đọc dữ liệu sai)
* Unrepeatable reads (đọc hai lần bản ghi)
* Phantoms (Đọc các bản ghi nháp, không có)
* Lost updates (cập nhật, mất dữ liệu).

**Đọc dữ liệu sai (Dirty reads):** Đọc dữ liệu sai xảy ra khi giao dịch đọc một bản ghi mà một phần giao dịch khác chưa hoàn thành. Nếu giao dịch trước đó hoàn thành thì sẽ không xảy ra các vấn đề này, nhưng nếu giao dịch trước đó chưa hoàn thành hay đang thực hiện chế độ Rollback chúng ta sẽ phải đọc dữ liệu cũ, dữ liệu sai.

**Đọc bản ghi hai lần (Unrepeatable reads):**Khi đọc mẩu tin hai lần trong một giao dịch trong khi giao dịch khác chỉ thông báo về tình trạng dữ liệu trong một khoảng thời gian quy định.

**Phantoms (đọc các bản ghi không có):** Nghĩa là chúng ta đọc được những bản ghi không có. Vì những bản ghi đó xuất hiện không bị tác động bởi các lệnh UPDATE hoặc DELETE. Khác với hai vấn đề trên Phantoms là các vấn đề liên quan đến hệ điều hành, nó không yêu cầu bất kỳ sơ đồ nào để mô tả.

**Lost Update (cập nhật mất DL):** Xảy ra khi một giao dịch cập nhật dữ liệu vào cơ sở dữ liệu thành công, nhưng lại ghi đè lên dữ liệu của giao dịch khác. Xảy ra khi: 2 giao dịch đang đọc mẩu tin dữ liệu, sau đó giao dịch 1 ghi dữ liệu của bản ghi, giao dịch 2 cũng ghi kết quả chỉ có giao dịch 2 được cập nhật.

Các loại Lock được sử dụng trong SQL Server:

**Share locks (khoá chia sẻ):** Đây là loại căn bản nhất, lock chia sẻ tài nguyên cho phép đọc dữ liệu, không cho phép thay đổi bất kỳ thuộc tính nào của tài nguyên.

**Exclusive locks (khoá độc quyền):** Không tương thích với các loại khoá khác. Khoá này ngăn ngừa hai người sử dụng cùng cập nhật, xoá, thêm dữ liệu.

**Update locks (khoá cập nhật):** Kết hợp giữa Share Lock và Exclusive Lock. Với câu lệnh UPDATE trong nội dung có mệnh đề WHERE, trong khi chưa cần cập nhật  
thì Lock sẽ ở trạng thái Share Lock. Khi câu lệnh UPDATE thực hiện thì Lock sẽ ở chế độ Exclusive Lock.

# BÀI TẬP CHƯƠNG 8.

## CÂU HỎI

**Câu hỏi 1:** Cursor là gì ? Cursor thường được sử dụng với mục đích gì? Trình bày cách đọc và xử lý nội dung các dòng lệnh bên trong Cursor.

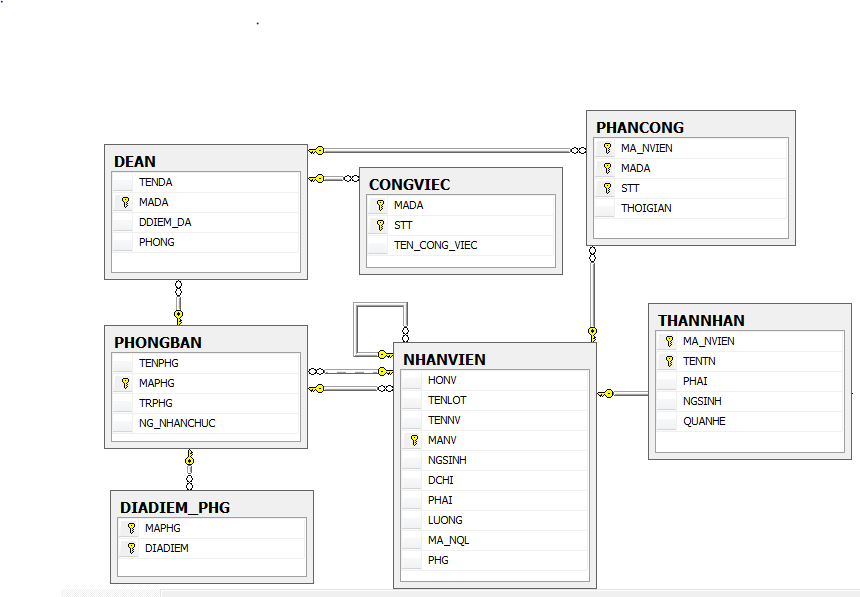
**Câu hỏi 2:** Trình bày khái niệm và vai trò của View trong một cơ sở dữ liệu.

**Câu hỏi 3:** Trình bày khái niệm, phân loại và vai trò của Index.

**Câu hỏi 4**: Trình bày khái niệm về Transaction và Lock.

## BÀI TẬP

**Sử dụng lược đồ sau để thực hiện bài tập 1 (mô tả chi tiết về lược đồ quan hệ xem lại ở bài tập chương 1)**



**Bài tập 1:**

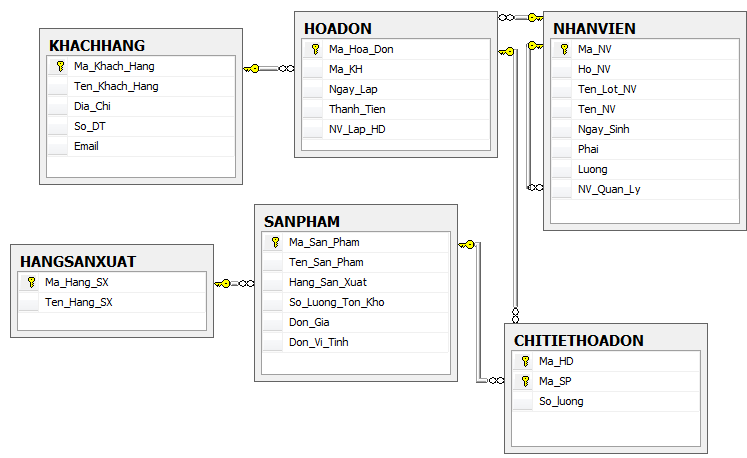
**Thực hiện các yêu cầu cài đặt Cursor như sau:**

1. Cài đặt biến Cursor đọc dữ liệu từng dòng của bảng NHANVIEN và truy vấn ra danh sách các nhân viên ở phòng ban Nghiên cứu.
2. Cài đặt biến Cursor đọc dữ liệu từng dòng của bảng PHONGBAN và truy vấn ra danh sách các phòng ban chủ trì từ 3 đề án trở lên
3. Cài đặt biến Cursor đọc dữ liệu từng dòng của bảng PHANCONG và cập nhật lương của nhân viên có mã ‘nv03’ tăng thêm 100\*thời gian làm việc trong bảng phân công.
4. Thêm thuộc tính SODEAN\_PC trong bảng NHANVIEN. Cài đặt biến Cursor đọc dữ liệu từng dòng của bảng PHANCONG và cập nhật giá trị SODENAN\_PC.

**Thực hiện tạo các View , xác định Updatable View hay Partion View trong các View được yêu cầu bên dưới :**

1. Tạo view cho xem danh sách mã, họ tên, ngày sinh, lương của các nhân viên ở phòng ban nghiên cứu
2. Tạo view cho xem danh sách mã, tên phòng ban và họ tên trưởng phòng của phòng ban đó
3. Tạo view cho xem danh sách mã, họ tên các nhân viên nữ và tên các đề án mà họ tham gia làm việc
4. Tạo view cho xem danh sách mã, tên đề án và họ tên các nhân viên tham gia vào đề án đó.
5. Tạo view cho xem danh sách mã, tên đề án và số lượng nhân viên tham gia vào đề án đó.
6. Tạo view cho xem danh sách mã, tên phòng ban và lương cao nhất của nhân viên trong phòng ban đó.
7. Tạo view cho xem danh sách mã đề án, tên đề án và thời gian làm việc tổng cộng của các nhân viên tham gia đề án đó.

**Sử dụng lược đồ sau để thực hiện các bài tập 2 (mô tả chi tiết về lược đồ quan hệ xem lại ở bài tập chương 1)**



**Bài tập 2:**

**Thực hiện các yêu cầu cài đặt Cursor như sau:**

1. Cài đặt Cursor duyệt từng dòng dữ liệu của bảng SANPHAM và truy vấn ra danh sách các sản phẩm có trong hóa đơn ‘HD001’
2. Cài đặt Cursor duyệt từng dòng dữ liệu của bảng NHANVIEN và truy vấn ra danh sách các nhân viên lập ít nhất 3 hóa đơn
3. Cài đặt Cursor duyệt từng dòng dữ liệu của bảng CHITIETHOADON và cập nhật thành tiền cho hóa đơn theo công thức :Thanh\_Tien = So\_Luong \* Don\_gia
4. Cài đặt Cursor duyệt từng dòng dữ liệu của bảng CHITIETHOADON và cập nhật So\_Luong\_Ton\_Kho cho các sản phẩm xuất hiện trong các CHITIETHOADON của hóa đơn ‘HD001’
5. Cài đặt Cursor duyệt từng dòng dữ liệu của bảng HOADON và cập nhật Lương của các nhân viên tăng thêm một khoản bằng 10% thành tiền của các hóa đơn do nhân viên đó lập.

**Thực hiện tạo các View , xác định Updatable View hay Partion View trong các View được yêu cầu bên dưới:**

1. Tạo view cho xem danh sách mã, tên sản phẩm và tên hãng sản xuất của các sản phẩm đó.
2. Tạo view cho xem danh sách mã, tên nhân viên và mã các hóa đơn do các nhân viên đó lập.
3. Tạo view cho xem danh sách mã, tên khách hàng và mã các hóa đơn của các khách hàng đó.
4. Tạo view cho xem danh sách mã, tên khách hàng và tên các sản phẩm mà các khách hàng đó đã mua.
5. Tạo view cho xem danh sách mã, tên của nhân viên và số lượng hóa đơn đã lập của từng nhân viên
6. Tạo view cho xem danh sách mã, tên của khách hàng và thành tiền của hóa đơn cao nhất mà khách hàng đã mua.
7. Tạo view cho xem danh sách mã, tên sản phẩm và số lượng bán ra của từng sản phẩm( dựa vào HOADON và CHITIETHOADON)
8. Tạo view cho xem danh sách mã, tên hãng sản xuất và tổng số tiền bán ra các sản phẩm của hãng sản xuất đó( dựa vào HOADON và CHITIETHOADON)

# CHƯƠNG 9 BẢO MẬT (SECURITY)

# MỤC ĐÍCH VÀ YÊU CẦU

**Mục đích:**

Chương này nhằm cung cấp cho sinh viên những kiến thức chung về:

* Những kiến thức cơ bản về cơ chế bảo mật trong SQL Server
* Phương thức đăng nhập và cấp quyền thao tác đối với cơ sở dữ liệu.

**Yêu cầu:**

Sau khi học xong chương này sinh viên có khả năng:

* Sử dụng thành thạo lệnh tạo tài khoản đăng nhập và công cụ cấp quyền thao tác trên cơ sở dữ liệu.
* Biết và hiểu cơ chế phân quyền trong SQL Server khi thao tác trên CSDL.

# NỘI DUNG

## 9.1. Cơ chế bảo mật trong SQL Server

### 9.1.1. Hai lớp bảo mật trong SQL Server

Kiến trúc Security trong SQL Server dựa vào 3 thành phần chính:

* Principals: các tài khỏan bảo mật
* Securables: các đối tựơng được bảo vệ
* Permissions: quyền sử dụng các Securables của Principals.

SQL Server có 2 cơ chế xác thực User.SQL Server có thể xác thực các đăng nhập dựa vào cơ chế bên trong của riêng nó hay dựa vào Windows để xác thực việc đăng nhập. Mỗi cơ chế đều có những thuận tiện và bất tiện riêng.

* SQL Server Authentication: Đây là cơ chế xác thực của SQL Server. Với chế độ này, SQL Server lưu trữ các tài khỏan đăng nhập và mật mã trong database Master. Điểm thuận tiện của chế độ này là SQL Server có thể xác thực bất kỳ phiên đăng nhập nào mà không quan tâm người dùng đã được xác thực như thế nào trong Window, miễn là họ có một tài khoản SQL Server. Chế độ này làm giảm tính bảo mật.
* Windows Authentication: cơ chế này dựa vào Windows. Windows thực hiện xác thực và SQL Server tin tưởng điều này và cung cấp quyền truy cập cho các tài khỏan Windows.

DBA có thể cấu hình những chế độ xác thực này bằng 2 cách:

* Mixed Security: Một login có thể thực hiện dựa vào kết nối SQL Server hay Windows.
* Windows Only: Chỉ những login có quyền trong Windows mới được đăng nhập.

Để dễ dàng quản lý các Permission trên Server, SQL Server cung cấp các Role. Role cũng tương tự như Groups trong hệ điều hành Windows.Server-Level Roles còn gọi là Fixed Server Roles. Nó cố định vì ta không thể tạo thêm Server-Level Roles.

Ví dụ về những Server-level roles tiêu biểu:

* Sysadmin: thành viên của sysadmin có thể thực hiện bất kỳ hành động nào trên Server.
* Serveradmin: thành viên của serveradmin có thể thay đổi cấu hình và shut down Server.
* Securityadmin: thành viên của Securityadmin có thể quản lý các login và thuộc tính của chúng.

Để làm việc với các Server-Level Roles ta có thể dùng các Commands, Views, Functions.

### 9.1.2. Các khái niệm về bảo mật trong SQL Server

#### 9.1.2.1. Login và User

SQL Server cho phép truy nhập vào hệ thống thông qua các login. Chỉ khi có quyền ở mức độ nhất định ta mới có thể tạo thêm login. Trong SSMS ta có thể xem các login này bằng cách mở rộng node Security/Logins. Các login này mới chỉ có quyền truy nhập vào server chứ chưa hẳn đã truy nhập được vào các database chứa trong đó.

Đến lượt mỗi database lại duy trì một danh sách các user, các user này luôn luôn đính (mapped) với một login ở mức server. Khi ta đăng nhập vào SQL Server thông qua login này, ta sẽ có quyền truy nhập vào database theo quyền hạn mà user tương ứng với nó được cung cấp. Ta xem các user này bằng cách mở rộng node Security/Users của database tương ứng.

#### 9.1.2.2. Server Role

Role thực chất là tập hợp một nhóm các quyền và đại diện bằng một tên để thuận tiện cho việc quản lý. Server role là nhóm các quyền ở mức server mà login khi được cấp sẽ có thể thực hiện một số thao tác nhất định ở mức server. Ví dụ role sysadmin có toàn quyền hoạt động trong server (tạo database, khởi động lại server…) và tự khắc nó cũng có quyền truy nhập vào tất cả database, diskadmin chỉ có quyền trong các thao tác về đĩa (như bổ sung thêm file vào database…). Login khi mới được tạo sẽ có role public, role này thực chất không có quyền gì ngoài quyền truy nhập vào server. Thông thường chỉ DBA mới nên có quyền sysadmin, còn các developer chỉ cần role public và bổ sung thêm khi cần.

#### 9.1.2.3. Database role

Cũng với mục đích như vậy, database role tập hợp các quyền truy nhập vào database thành từng nhóm để dễ tạo lập và sửa đổi. Ví dụ role db\_datareader có quyền đọc dữ liệu từ tất cả các bảng, role db\_datawriter có quyền ghi vào tất cả các bảng; role db\_owner có quyền cao nhất trong database (tạo bảng, tạo thủ tục, thực thi thủ tục…). Các user đều mặc định có role public. Khi user chỉ có role public, user chỉ có thể nhìn thấy tên database mà không có quyền gì khác.

#### 9.1.2.4. Authetication

SQL Server xác thực các đăng nhập bằng hai cơ chế, Windows Authentication và Sql Server Authentication. Việc này bắt nguồn từ khi tạo login trong SQL Server. Khi ta tạo một login mới, SQL Server sẽ yêu cầu ta chọn cơ chế xác thực Windows hay SQL Server cho login đó. Nếu chọn Windows, ta cần cung cấp một Windows Account và SQL Server sẽ chỉ lưu tên của Account đó trong danh sách login. Nếu chọn SQL Server Authentication, ta cần cung cấp Login Name và Password và cả hai đều được lưu trong SQL Server.

Khi đăng nhập vào SQL Server, ta cũng sẽ phải chọn một trong hai cơ chế xác thực. Nếu chọn Windows, chính Account hiện đang đăng nhập vào Windows được dùng. Ta không có quyền chọn login và thực tế, SQL Server sẽ vòng xuống Windows và kiểm tra xem Account nào đang đăng nhập vào Windows, và lấy Account đó, kiểm tra xem có nằm trong danh sách Login của nó hay không. Nếu có thì SQL Server sẽ chấp nhậnđăng nhập, ngược lại sẽ chặn lại.

Với SQL Server Authentication, cơ chế đăng nhập giống như bình thường – ta cung cấp Login ID và Password. SQL Server sẽ kiểm tra và quyết định có cho phép đăng nhập hay không.

Một điểm cần lưu ý là với Login sử dụng Windows Authentication, SQL Server chỉ lưu Login Name và không hề Lưu Password. Và khi đăng nhập nếu ta cố tình chọn sang SQL Server Authentication, hệ thống sẽ không xác thực cho ta.

#### 9.1.2.5. Group, phân quyền cho User và Group

Phân quyền là một trong những yếu tố đóng vai trò quan trọng đối với sự sống còn của cơ sở dữ liệu. Phân quyền người dùng bao gồm các thao tác sau:

* Cấp phát quyền truy cập cơsở dữ liệu cho người dùng và các nhóm người dùng, phát hiện và ngăn chặn những thao tác trái phép của người sử dụng trên cơ sở dữ liệu.
* Cấp phát quyền sử dụng các câu lệnh, các đối tượng cơ sở dữ liệu đối với người dùng.
* Thu hồi (huỷ bỏ) quyền của người dùng.

Một nhóm bao gồm các user có quyền giống nhau, thay vì ta phải gán quyền riêng cho từng user mỗi khi user đó được tạo, ta chỉ cần tạo nhóm và phân quyền cho nhóm đó, nếu có thêm một user mới, thì khi tạo sẽ gắn nó với nhóm đã phân quyền.

Cú pháp câu lệnh phân quyền như sau:

**GRANT *permission\_list***

**ON *object\_name***

**TO *name\_list***

Cú pháp lệnh thu hồi quyền như sau:

**REVOKE *permission\_list***

**ON *object\_name***

**FROM *name\_list***

Trong đó:

* ***permission\_list*** là danh sách các quyền hạn sẽ được gán hay tước bỏ. Nếu nhiều hơn một quyền thì các quyền hạn đó được cách nhau bởi dấu phẩy.
* ***object\_name*** là tên của table, view, hay stored procedure mà user sẽ được gán quyền hay bị tước quyền.
* ***name\_list*** là danh sách user hay group sẽ được gán quyền hay bị tước quyền. Giữa các tên user hoặc group ngăn cách bằng dấu phẩy. Nếu sử dụng từ khóa PUBLIC thì tác vụ gán chung cho toàn bộ các user.

**Ví dụ:** Cho phép User tên Anh được xem thông tin NGAYSINH,MANV trong Table NHANVIEN

GRANT SELECT

ON NHANVIEN(MANV,NGAYSINH)

TO Anh

Khi ta cấp phát quyền nào đó cho một người dùng trên một đối tượng cơ sở dữ liệu, người dùng đó có thể thực thi câu lệnh được cho phép trên đối tượng đã cấp phát. Tuy nhiên, người dùng đó không có quyền cấp phát những quyền mà mình được phép cho những người sử dụng khác. Trong một số trường hợp, khi ta cấp phát quyền cho một người dùng nào đó, ta có thể cho phép người đó chuyển tiếp quyền cho người dùng khác bằng cách chỉ định tuỳ chọn WITH GRANT OPTION trong câu lệnh GRANT.

**Ví dụ:** Cho phép User tên Anh được xem thông tin NGAYSINH,MANV trong Table NHANVIEN và cho phép User này được cấp quyền này cho User khác.

GRANT SELECT

ON NHANVIEN(MANV,NGAYSINH)

TO Anh

WITH GRANT OPTION

Khi ta sử dụng câu lệnh REVOKE để thu hồi quyền trên một đối tượng cơ sở dữ liệu từ một người dùng nào đó, chỉ những quyền mà ta đã cấp phát trước đómới được thu hồi, những quyền mà người dùng này được cho phép bởi những người dùng khác vẫn còn có hiệu lực. Nói cách khác, nếu hai người dùng khác nhau cấp phát cùng các quyền trên cùng một đối tượng cơ sở dữ liệu cho một người dùng khác, sau đó người thu nhất thu hồi lại quyền đã cấp phát thì những quyền mà người dùng thứ hai cấp phát vẫn có hiệulực.

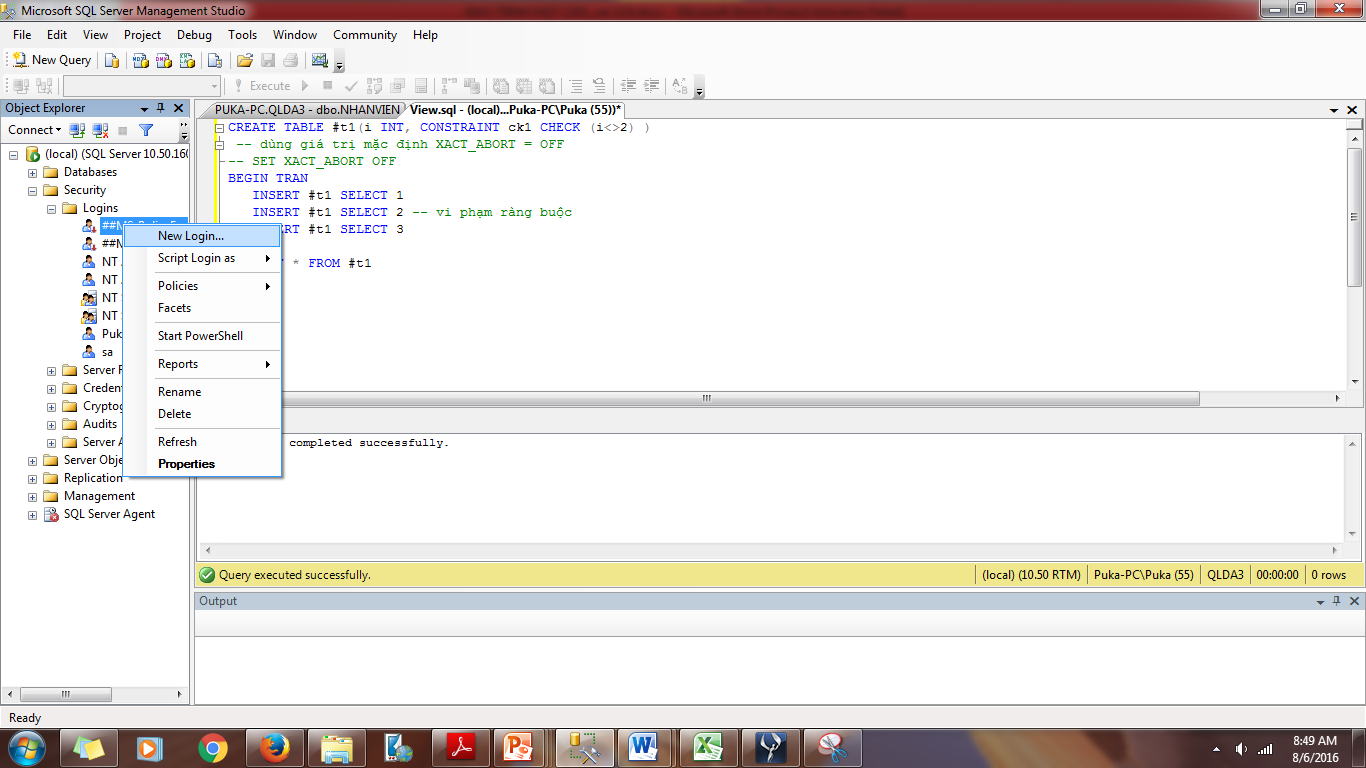
Nếu ta đã cấp phát quyền cho người dùng nào đó bằng câu lệnh GRANT với tuỳ chọn WITH GRANT OPTION thì khi thu hồi quyền bằng câu lệnh REVOKE phải chỉ định tuỳ chọn CASCADE.Trong trường hợp này, các quyền được chuyển tiếp cho những người dùng khác cũng đồng thời được thu hồi.

## 9.2. Tạo các tài khoản đăng nhập

**Bước 1:** Nhấn vào biểu tượng SQL Server Management Studio ở màn hình desktop hoặc Start 🡪Program 🡪 SQL Server 2008 R2.

**Bước 2:** Truy cập vào phần quản trị SQL Server sử dụng tài khoản Windows Authentication (Yêu cầu đăng nhập windows với tài khoản Administrator)

**Bước 3:** Mở rộng cây bên trái chọn mục Security 🡪 Logins như hình bên dưới, nhấn chuột phải vào Users muốn sửa thông tin, chọn Properties.

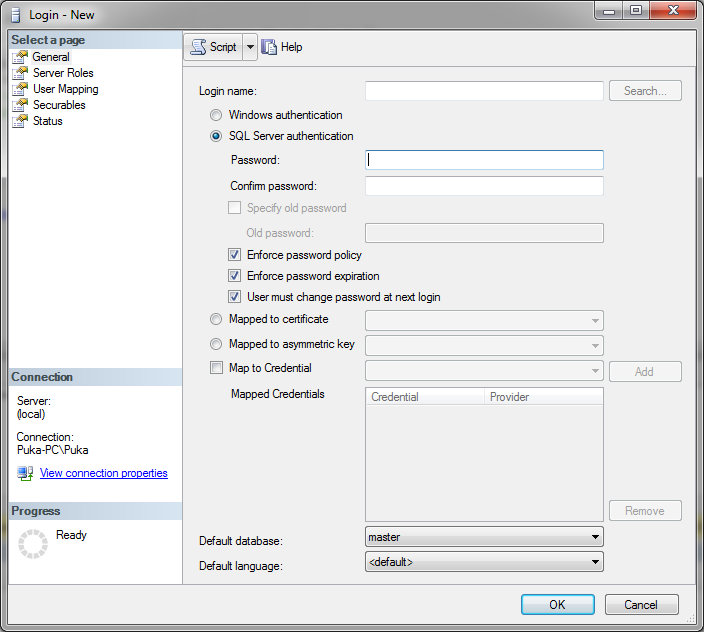


Hình 9. 1. Thao tác tạo tài khoản đăng nhập

**Bước 4:** Sau khi chọn bước 3 sẽ hiện lên cửa sổ Login Properties như hình bên dưới, ta có thể thay đổi Password, cho phép đặt chế độ cho password như:

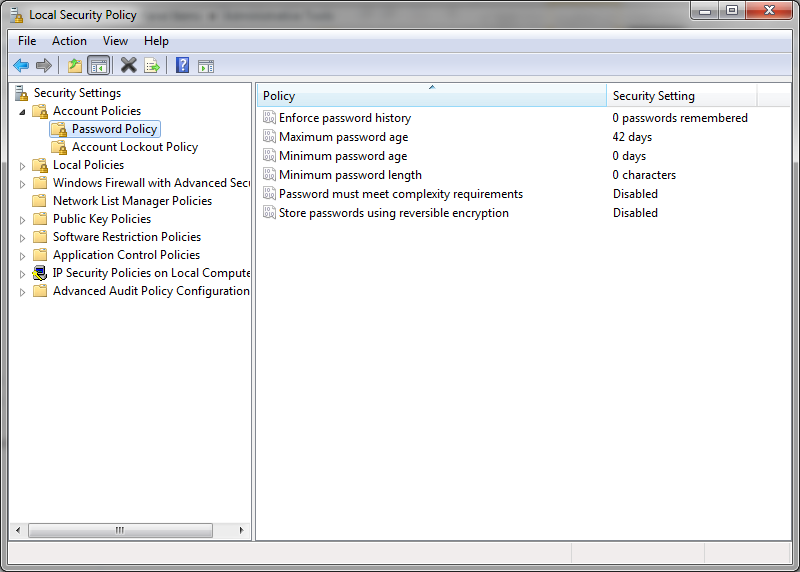
* Enforce password policy:  Yêu cầu đặt Password đầy đủ các ký tự vừa ký tự đặc biệt, text, số.
* Enforce password expiration: Cho phép Password sẽ hết hạn trong vòng 42 ngày (xem thêm hình minh họa bên dưới nữa), bỏ chọn nếu không xác định password hết hạn.
* User must change password at next login: Yêu cầu người dùng đăng nhập lần đầu tiên phải đổi password.

Nhấn OK để hoàn thành thao tác.



Hình 9. 2. Màn hình thiết lập tài khoản đăng nhập

Kiểm tra thời gian hết hạn của password: vào Start | All Programs | Administrative Tools | Local Security Policy, xem hình bên dưới.



Hình 9. 3. Kiểm tra thời hạn hiệu lực của password đăng nhập

# BÀI TẬP CHƯƠNG 9

## CÂU HỎI

**Câu hỏi 1:** Trình bày cơ chế bảo mật trong SQL Server.

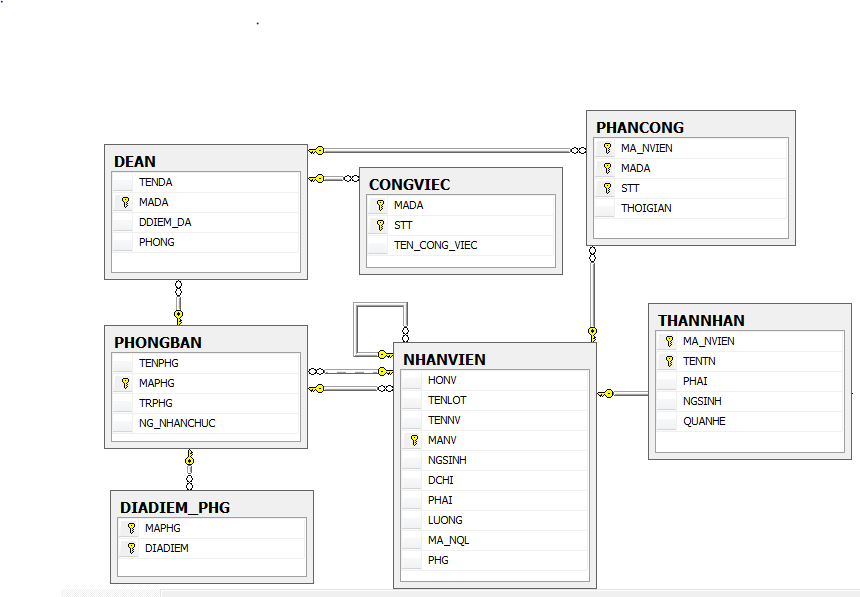
**Câu hỏi 2:** Trình bày khái niệm user, role trong SQL Server

**Câu hỏi 3:** Trình bày khái niệm, cú pháp câu lệnh phần quyền và thu hồi quyền trong SQL Server.

**Câu hỏi 4**: Trình bày các bước tạo một login mới trong SQL Server.

## BÀI TẬP

**Sử dụng lược đồ sau để thực hiện bài tập 1 (mô tả chi tiết về lược đồ quan hệ xem lại ở bài tập chương 1)**



**Bài tập 1. Thực hiện các yêu cầu theo trình tự sau:**

**Tạo các Login và User:**

1. Tạo các login sau: KHACH\_LOGIN, NHANVIEN\_LOGIN, QUANLY\_LOGIN
2. Tạo các user cho login KHACH\_LOGIN: KHACH01
3. Tạo các user cho NHANVIEN\_LOGIN: NHANVIEN01
4. Tạo các user cho QUANLY\_LOGIN: QUANLY01

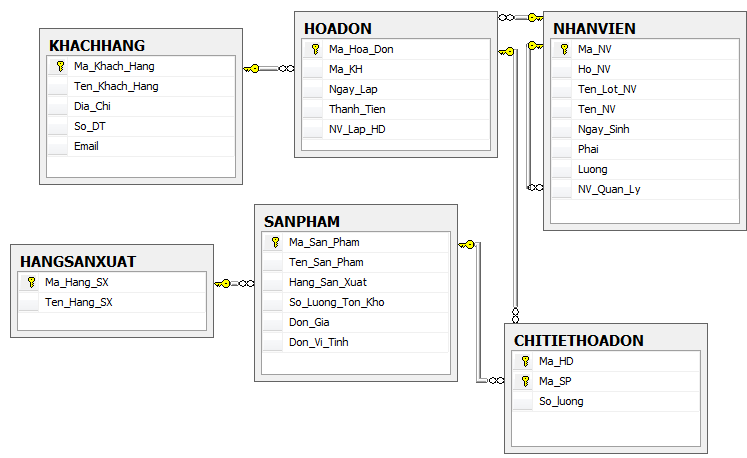
**Phân quyền:**

1. Đăng nhập SQL Server với login sa, phân toàn quyền quản trị database cho user QUANLY01 và user QUANLY01 được cấp quyền lại cho user khác.
2. Đăng nhập SQL Server với login QUANLY, phân quyền cho phép user NHANVIEN01 có thể xem thông tin của các bảng DEAN, PHANCONG, CONGVIEC. Sau đó đăng nhập SQL Server với login NHANVIEN, tiến hành xem danh sách bảng PHONGBAN.
3. Đăng nhập SQL Server với login QUANLY, cấp quyền cho phép user NHANVIEN01 xem danh sách bảng PHONGBAN.
4. Đăng nhập SQL Server với login QUANLY, cấp quyền cho phép user KHACH01 có thể xem danh sách tên các đề án và địa điểm.

**Thu hồi quyền:**

Thu hồi các quyền đã cấp cho user QUANLY01.

**Sử dụng lược đồ sau để thực hiện các bài tập 2 (mô tả chi tiết về lược đồ quan hệ xem lại ở bài tập chương 1)**



**Bài tập 2. Thực hiện các yêu cầu theo trình tự sau:**

**Tạo các Login và User:**

1. Tạo các login sau: KHACH\_LOGIN, NHANVIEN\_LOGIN, QUANLY\_LOGIN
2. Tạo các user cho login KHACH\_LOGIN: KHACH02
3. Tạo các user cho NHANVIEN\_LOGIN: NHANVIEN02
4. Tạo các user cho QUANLY\_LOGIN: QUANLY02

**Phân quyền:**

1. Đăng nhập SQL Server với login sa, phân toàn quyền quản trị database cho user QUANLY02 và user QUANLY02 được cấp quyền xem và sửa trên bảng HOADON và CHITIETHOADON lại cho user khác.
2. Đăng nhập SQL Server với login QUANLY, phân quyền cho phép user NHANVIEN01 có thể xem thông tin của các bảng HOADON, CHITIETHOADON. Sau đó đăng nhập SQL Server với login NHANVIEN, tiến hành chỉnh sửa nội dung trong bảng CHITIETHOADON.
3. Đăng nhập SQL Server với login QUANLY, cấp quyền cho phép user NHANVIEN02 chỉnh sửa nội dung trên bảng CHITIETHOADON.
4. Đăng nhập SQL Server với login QUANLY, cấp quyền cho phép user KHACH02 có thể xem danh sách tên và ngày lập của HOADON.

**Thu hồi quyền:**

Thu hồi các quyền đã cấp cho user QUANLY02.