

GIÁO TRÌNH HỆ CƠ SỞ DỮ LIỆU

- PHẦN 1: CƠ SỞ DỮ LIỆU
- PHẦN 2: SQL SERVER

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP/HCM
TRUNG TÂM CÔNG NGHỆ THÔNG TIN
-000-

PHẦN 1 CƠ SỞ DỮ LIỆU

MỤC LỤC

Chương 1	3
MÔ HÌNH QUAN HỆ	3
I NGUYÊN NHÂN RA ĐỜI CỦA MÔ HÌNH QUAN HỆ	3
II CƠ SỞ DỮ LIỆU VÀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU	3
1 CSDL là gì?	4
2 Hệ quản trị CSDL	4
3 Người dùng (User)	4
4 CSDLQH và Hệ tập tin theo lối cũ	5
III MÔ HÌNH QUAN HỆ	5
1 Mô hình quan hệ là gì ?	5
2 Các khái niệm cơ bản của mô hình quan hệ	6
3 Các phép toán tập hợp	10
4 Các phép toán quan hệ	11
5 Các tính chất của đại số quan hệ	12
IV MÔ HÌNH THỰC THỂ KẾT HỢP	14
1 Giới thiệu mô hình thực thể kết hợp	14
2 Chuyển từ mô hình thực thể kết hợp sang lược đồ CSDL	17
V BÀI TẬP	19
1 Phép toán tập hợp và phép toán quan hệ	19
2 Mô hình thực thể kết hợp	19
Chương 2	22
NGÔN NGỮ TRUY VẤN SQL	22
I CÁCH TẠO QUAN HỆ BẢNG ACCESS	22
II CÂU LỆNH TRUY VẤN	22
1 BIỂU THỨC (EXPRESSION)	22
2 CÂU LỆNH SQL	25
III BÀI TẬP	28
Chương 3	31
RÀNG BUỘC TOÀN VỊN QUAN HỆ	31
I RÀNG BUỘC TOÀN VỊN - CÁC YẾU TỐ CỦA RÀNG BUỘC TOÀN VỊN	31
1 Ràng Buộc Toàn Vịn	31
2 Các Yếu Tố Của Ràng Buộc Toàn Vịn	31
II PHÂN LOẠI RÀNG BUỘC TOÀN VỊN	32
1 Ràng buộc toàn vịn liên bộ	33
2 Ràng buộc toàn vịn về phụ thuộc tồn tại:	33
3 Ràng buộc toàn vịn về miền giá trị	33
4 Ràng buộc toàn vịn liên thuộc tính	34
5 Ràng buộc toàn vịn liên thuộc tính liên quan hệ	34
6 Ràng buộc toàn vịn về thuộc tính tổng hợp	34
III BÀI TẬP	34
Chương 4	36
PHỤ THUỘC HÀM	36
I KHÁI NIỆM PHỤ THUỘC HÀM	36

1	Định nghĩa phụ thuộc hàm	37
2	Phụ thuộc hàm hiển nhiên	37
3	Thuật toán Satifies.....	37
4	Các phụ thuộc hàm có thể có.....	38
II	HỆ LUẬT DẪN ARMSTRONG.....	41
1	Phụ thuộc hàm được suy diễn logic từ F	41
2	Hệ luật dẫn Amstrong	42
3	Hệ luật dẫn Armstrong là đầy đủ	45
III	THUẬT TOÁN TÌM F^+	46
1	Thuật toán cơ bản	46
2	Thuật toán cải tiến.....	47
IV	BÀI TẬP	47
Chương 5		49
PHỦ CỦA TẬP PHỤ THUỘC HÀM.....		49
I	ĐỊNH NGHĨA	49
II	PHỦ TỐI THIỂU CỦA MỘT TẬP PHỤ THUỘC HÀM	49
1	Phụ thuộc hàm có vế trái dư thừa	49
2	Tập phụ thuộc hàm có vế phải một thuộc tính	50
3	Tập phụ thuộc hàm không dư thừa	50
4	Tập phụ thuộc hàm tối thiểu.....	50
III	KHÓA CỦA LƯỢC ĐỒ QUAN HỆ.....	51
1	Định Nghĩa	51
2	Thuật toán tìm tất cả khóa	52
IV	BÀI TẬP	54
Chương 6		56
CHUẨN HÓA CƠ SỞ DỮ LIỆU.....		56
I	DẠNG CHUẨN CỦA LƯỢC ĐỒ QUAN HỆ	56
1	Định nghĩa các dạng chuẩn	56
II	PHÉP TÁCH KẾT NỐI BẢO TOÀN.....	61
1	Phép tách kết nối bảo toàn thông tin.....	61
2	Phép tách bảo toàn phụ thuộc hàm	66
III	THIẾT KẾ CSDL BẰNG CÁCH PHÂN RÃ.....	69
1	Phân rã thành dạng chuẩn BC (hay chuẩn 3) bảo toàn thông tin.....	69
2	Phân rã thành dạng chuẩn 3 vừa bảo toàn thông tin vừa bảo toàn phụ thuộc hàm.....	75
IV	BÀI TẬP	78

---oOo---

Chương 1 .

MÔ HÌNH QUAN HỆ

I NGUYÊN NHÂN RA ĐỜI CỦA MÔ HÌNH QUAN HỆ (RELATIONAL MODEL)

Trong nhiều năm, công nghệ tính toán và thông tin phát triển từ những hệ thống lớn, đắt tiền, độc quyền đến các hệ thống mở mạnh và không đắt tiền. Sự phát triển này mang lại lợi ích to lớn cho người dùng cuối bởi sự phát triển của các gói ứng dụng số như xử lý văn bản, bảng tính điện tử, văn phòng xuất bản, hệ quản lý cơ sở dữ liệu, máy tính trợ giúp công nghệ phần mềm....

Trước khi máy tính hóa cơ sở dữ liệu được giới thiệu, dữ liệu được lưu trữ theo kiểu điện tử thành nhiều tập tin riêng biệt sử dụng **hệ tập tin** (từ đây về sau ta gọi **hệ tập tin theo lối cũ**). Những tập tin này được xử lý bằng các ngôn ngữ thế hệ thứ ba như COBOL, FORTRAN, PASCAL và ngay cả BASIC để tạo ra các giải pháp cho các vấn đề của doanh nghiệp. Mỗi ứng dụng, chẳng hạn như hệ tính lương, hệ kho hay hệ thống kế toán sẽ có một tập các tập tin riêng chứa dữ liệu riêng. Các ứng dụng như vậy tạo ra ba vấn đề sau:

- Có sự liên kết chặt chẽ giữa cấu trúc luận lý và cấu trúc vật lý của các tập tin và chương trình ứng dụng khai thác chúng. Điều này khiến việc tạo nên các ứng dụng này rất khó khăn, tốn nhiều thời gian và do vậy mà tốn kém trong bảo trì hệ thống.
- Có sự dư thừa dữ liệu rất lớn qua việc trùng lặp các tập tin trong các ứng dụng khác nhau. Điều này tạo ra những vấn đề như: dữ liệu thiếu nhất quán, không gian đĩa bị lãng phí, thời gian bảo trì và lưu phòng hồ các tập tin gia tăng, vấn đề về quản trị như không chú trọng bảo mật và tổ chức dữ liệu thiếu thống nhất. Một ví dụ điển hình về sự trùng lặp dữ liệu là: Hệ quản lý nguồn nhân lực bao gồm ba hệ chính:
 1. Hệ lương, hệ này duy trì ngày công và lương cho tất cả nhân viên.
 2. Hệ nhân sự, hệ này duy trì lý lịch cá nhân, dữ liệu về tổ chức, công việc đào tạo và vị trí thăng tiến.
 3. Hệ hưu, hệ này quản trị các qui tắc liên quan đến nghỉ hưu, loại nghỉ hưu. Chi tiết về hưu của từng nhân viên.

Vấn đề phức tạp là Hệ lương thông thường được quản lý bởi phòng tài chính, trong khi Hệ nhân sự và Hệ hưu được quản lý bởi phòng tổ chức. Rõ ràng, có nhiều dữ liệu về nhân viên là chung cho cả ba hệ. Thường những hệ này thực hiện và giữ gìn riêng biệt và chúng tạo sự trùng dữ liệu nhân viên mà chúng dùng.

- Người sử dụng có ít khả năng khai thác trực tiếp dữ liệu.

II CƠ SỞ DỮ LIỆU VÀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU (DATABASE AND DATABASE MANAGEMENT SYSTEM)

Khởi đầu, sự giới thiệu CSDL và HQTCSDL nhằm giải quyết các vấn đề của hệ thông tin dựa trên các tập tin theo lối cũ (C1.I). Điều này tạo ra việc phát triển trên hai mươi lăm năm qua một hệ CSDL quan hệ thương mại xuất hiện cuối những năm thập niên 70 và các năm đầu của thập niên 80. Trước khi xem xét CSDL và hệ QTCSDLQH giải quyết một vài vấn đề của hệ thông tin theo lối cũ như thế nào chúng ta cần làm rõ vài khái niệm.

1 CSDL là gì?

Một cơ sở dữ liệu có thể định nghĩa tạm như sau: một chỗ chứa có tổ chức tập hợp các tập tin dữ liệu có tương quan, các mẫu tin và các cột.

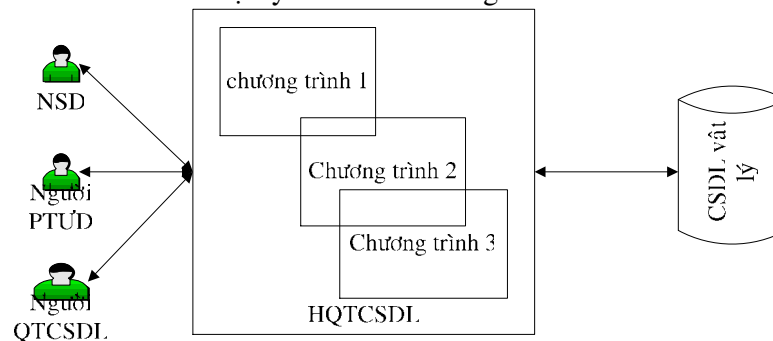
Ngày nay CSDL tồn tại trong mỗi ứng dụng thông dụng, ví dụ:

- Hệ kho và kiểm kê.
- Hệ đặt chỗ máy bay
- Hệ nguồn nhân lực.
- hệ dịch vụ công cộng như cấp nước, điện, khí đốt
- Điều khiển quá trình chế tạo và sản xuất

2 Hệ quản trị CSDL

Một hệ quản trị CSDL (HQTCSDL) là:

- một tập các phần mềm quản lý CSDL và cung cấp các dịch vụ xử lý CSDL cho các những người phát triển ứng dụng và người dùng cuối.
- HQTCSDL cung cấp một giao diện giữa người sử dụng và dữ liệu.
- HQTCSDL biến đổi CSDL vật lý thành CSDL logic.



Hình 1.2.1 - HQTCSDL cung cấp phần giao diện giữa NSD và CSDL

Dựa vào cách tổ chức dữ liệu, HQTCSDL được chia thành năm loại:

- loại phân cấp như hệ IMS của IBM
- loại mạng như IDMS của Cullinet Software
- Loại tập tin đảo như ADABAS của Software AG
- Loại quan hệ như ORACLE của Oracle, DB2 của IBM, ACCESS của Microsoft Access
- Loại đối tượng là một tiếp cận khá mới trong thiết kế HQTCSDL và việc sử dụng loại này sớm trở nên phổ biến

Hiện tại, loại HQTCSDL chính được sử dụng trong công nghệ là loại HQTCSDL quan hệ (RDBMS). Loại này đã chiếm lĩnh trong công nghệ trên 10-15 năm cuối cùng khi đánh bật loại HQTCSDL phân cấp và gần đây là HQTCSDL mạng.

3 Người dùng (User)

Người dùng khai thác CSDL thông qua HQTCSDL có thể phân thành ba loại: người quản trị CSDL, người phát triển ứng dụng và lập trình, người dùng cuối.

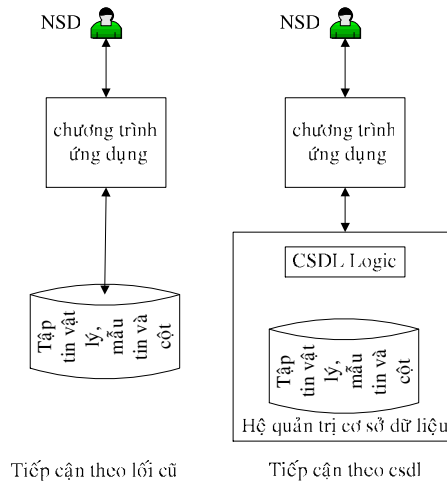
- Người quản trị CSDL, hàng ngày, chịu trách nhiệm quản lý và bảo trì CSDL như:
 - + sự chính xác và toàn vẹn của dữ liệu và ứng dụng trong CSDL, sự bảo mật của CSDL
 - + lưu phòng hồ và phục hồi CSDL
 - + giữ liên lạc với người phát triển ứng dụng, người lập trình và người dùng cuối.
 - + bảo đảm sự hoạt động trôi chảy và hiệu quả của CSDL và HQTCSDL
- Người phát triển và lập trình ứng dụng là những người chuyên nghiệp về máy tính có trách nhiệm thiết kế, tạo dựng và bảo trì hệ thông tin cho người dùng cuối.

- Người dùng cuối là những người không chuyên về máy tính nhưng họ là các chuyên gia trong các lãnh vực khác có trách nhiệm cụ thể trong tổ chức. Họ khai thác CSDL thông qua hệ được phát triển bởi người phát triển ứng dụng hay các công cụ truy vấn của HQTCSDL.

4 CSDLQH và Hệ tập tin theo lối cũ

Tiếp cận CSDL đã giải quyết 3 vấn đề của hệ tập tin theo lối cũ:

i Vấn đề 1: cấu trúc logic và cấu trúc vật lý



Hình 1.2.2 - Tiếp cận theo esdl tạo ra một lớp phần mềm giữa dữ liệu vật lý và chương trình ứng dụng

Kiến trúc bên trong HQTCSDL quan hệ tách biệt rõ ràng giữa:

- cấu trúc luận lý của tất cả tập tin và chương trình ứng dụng khai thác tập tin này và
- cấu trúc vật lý của esdl và phần lưu trữ các tập tin.

Tiếp cận này tạo cho người quản trị CSDL có thể thay đổi cấu trúc vật lý hay nơi lưu trữ của tập tin mà không ảnh hưởng đến chương trình ứng dụng.

ii Vấn đề 2: dư thừa dữ liệu

Khi HQTCSDLQH được giới thiệu, nhiều tổ chức mong tích hợp các tập tin đã phân tán khắp trong tổ chức vào một CSDL tập trung. Dữ liệu có thể chia sẻ cho nhiều ứng dụng khác nhau và người sử dụng có thể khai thác đồng thời các tập con dữ liệu liên quan đến họ. Điều này làm hạn chế sự dư thừa dữ liệu.

iii Vấn đề 3: Sự khai thác dữ liệu của người sử dụng

Trong hệ QTCSDLQH người dùng có thể trực tiếp khai thác dữ liệu thông qua việc sử dụng các câu truy vấn hay các công cụ báo cáo được cung cấp bởi hệ QTCSDL.

III MÔ HÌNH QUAN HỆ (RELATIONAL MODEL)

1 Mô hình quan hệ là gì ?

Mô hình Cơ sở dữ liệu Quan hệ (gọi tắt là Mô hình Quan hệ) do E.F Codd đề xuất năm 1971. Mô hình này bao gồm:

- Một hệ thống các ký hiệu để mô tả dữ liệu dưới dạng dòng và cột như **quan hệ, bộ, thuộc tính, khóa chính, khoá ngoại, ...**
- Một tập hợp các phép toán thao tác trên dữ liệu như phép toán tập hợp, phép toán quan hệ.
- ràng buộc toàn vẹn quan hệ.

Các hệ HQTCSDLQH ngày nay được xây dựng dựa vào lý thuyết của mô hình quan hệ.

Mục đích của môn học này giúp cho sinh viên nắm được kiến trúc tổng quát về mô hình quan hệ và áp dụng nó để lập mô hình dữ liệu quan hệ có hiệu quả trong *lưu trữ* và *khai thác*.

MASV	HOTENH	MONHOC	TENKHOA	DIEMTHI
99001	TRAN DAN THU	CO SO DU LIEU	CÔNG NGHỆ THÔNG TIN	3.0
99002	NGUYEN HA DA THAO	CO SO DU LIEU	CÔNG NGHỆ THÔNG TIN	8.0
99001	TRAN DAN THU	TIN HOC VAN PHONG	CÔNG NGHỆ THÔNG TIN	6.0
99005	LE THANH TRUNG	TIN HOC VAN PHONG	ANH VAN	5.0

Chẳng hạn với bài toán quản lý điểm thi của sinh viên; nếu lưu trữ dữ liệu theo dạng bảng với các cột MASV, HOTEN, MONHOC, TENKHOA, DIEMTHI thì các giá trị của các cột HOTEN, MONHOC, TENKHOA sẽ bị trùng lặp. Sự trùng lặp này gây nên một số vấn đề:

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

Việc lưu trữ dữ liệu như trên không đúng với mô hình quan hệ. Để lưu trữ đúng với mô hình quan hệ ta phải thay MONHOC bằng MAMH, thay TENKHOA bằng MAKHOA, tách một bảng dữ liệu lớn đó ra thành nhiều bảng con, như mô hình dưới.

MASV	MAMH	MAKHOA	DIEMTHI	MASV	HOTEN
99001	CSDL	CNTT	3.0	99001	TRAN DAN THU
99002	CSDL	CNTT	8.0	99002	NGUYEN HA DA THAO
99001	THVP	CNTT	6.0	99005	LE THANH TRUNG
99005	THVP	AV	5.0		

MAMH	TENMH	SOTIET
CSDL	CO SO DU LIEU	90
THVP	TIN HOC VAN PHONG	90

MAKHOA	TENKHOA
CNTT	CONG NGHE THONG TIN
AV	ANH VAN

2 Các khái niệm cơ bản của mô hình quan hệ

i Thuộc tính (Attribute, Arity)

Chẳng hạn với bài toán quản lý điểm thi của sinh viên; với đối tượng sinh viên ta cần phải chú ý đến các đặc trưng riêng như họ tên, ngày sinh, nữ (giới tính), lĩnh thường trú, học bổng, lớp mà sinh viên theo học, . . . các đặc trưng này gọi là **thuộc tính**.

Các thuộc tính được phân biệt qua tên gọi và phải thuộc vào một kiểu dữ liệu nhất định (số, chuỗi, ngày tháng, logic, hình ảnh,...). Kiểu dữ liệu ở đây là kiểu đơn. Trong cùng một đối tượng không được có hai thuộc tính cùng tên.

Thông thường mỗi thuộc tính chỉ chọn lấy giá trị trong một tập con của kiểu dữ liệu và tập hợp con đó gọi là **miền giá trị** của thuộc tính đó. Thuộc tính **ngày trong tháng** thì có kiểu dữ liệu là số nguyên, miền giá trị của nó là 1 đến (tối đa là) 31. Hoặc **điểm thi** của sinh viên chỉ là các số nguyên từ 0 đến 10.

Thường người ta dùng các chữ cái hoa A,B,C,... để biểu diễn các thuộc tính, hoặc A_1, A_2, \dots, A_n để biểu diễn một số lượng lớn các thuộc tính.

ii Lược Đồ Quan Hệ (Relation Schema)

Tập tất cả các thuộc tính cần quản lý của một đối tượng cùng với mối liên hệ giữa chúng được gọi là **lược đồ quan hệ**. Lược đồ quan hệ Q với tập thuộc tính $\{A_1, A_2, \dots, A_n\}$ được viết là $Q(A_1, A_2, \dots, A_n)$. Tập các thuộc tính của Q được ký hiệu là Q^+ . Chẳng hạn lược đồ quan hệ sinh viên (Đặt tên là S_v) với các thuộc tính như trên là:

$S_v(MASV, HOSV, TENS\bar{V}, NU, NGAYSINH, MALOP, HOCBONG, TINH)$

Thường khi thành lập một lược đồ, người thiết kế luôn gán cho nó một ý nghĩa nhất định, ý nghĩa đó gọi là **tên từ** của lược đồ quan hệ đó. Dựa vào tên từ người ta xác định được **tập thuộc tính khóa** của lược đồ quan hệ (khái niệm khoá sẽ được trình bày ở phần sau).

Khi phát biểu tên từ cho một lược đồ quan hệ, người thiết kế cần phải mô tả đầy đủ ý nghĩa để người khác tránh hiểu nhầm. Chẳng hạn tên từ của lược đồ quan hệ trên là: "mỗi sinh viên có một mã sinh viên (MASV) duy nhất, mỗi mã sinh viên xác định tất cả các thuộc tính của sinh viên đó như họ tên (HOTEN), nữ (NU), ngày sinh (NGAYSINH), lớp theo học (MALOP), học bổng (HOCBONG), tỉnh cư trú (TINH).

Nhiều lược đồ quan hệ cùng nằm trong một hệ thống quản lý được gọi là một **lược đồ cơ sở dữ liệu**.

Ví dụ lược đồ cơ sở dữ liệu để quản lý điểm sinh viên có thể gồm các lược đồ quan hệ sau:

$S_v(MASV, HOSV, TENS\bar{V}, NU, NGAYSINH, MALOP, TINH, HOCBONG)$

$Lop(MALOP, TENLOP, SISO, MAKHOA)$

$Kh(MAKHOA, TENKHOA, SOCBGD)$

$Mh(MAMH, TENMH, SOTIET)$

$Kq(MASV, MAMH, DIEMTHI)$

Phần giải thích các thuộc tính:

MASV	Mã sinh viên
HOTEN	Tên sinh viên
NU	Nữ
NGAYSINH	ngày sinh
LOP	lớp
TENLOP	tên lớp
SISO	sĩ số lớp
MAKHOA	mã khoa
HOCBONG	học bổng
TINH	tỉnh
TENKHOA	tên khoa
SOCBGD	số cán bộ giảng dạy
MAMH	mã môn học
TENMH	tên môn học
SOTIET	số tiết
DIEMTHI	điểm thi

iii Quan Hệ (Relation)

Sự thể hiện của lược đồ quan hệ Q ở một thời điểm nào đó được gọi là **quan hệ**, rõ ràng là trên một lược đồ quan hệ có thể định nghĩa rất nhiều quan hệ. Thường ta dùng các ký hiệu như R, S, Q để chỉ các lược đồ quan hệ, còn quan hệ được định nghĩa trên nó tương ứng được ký hiệu là r, s, q .

iv Bộ (Tuple)

Bộ là tập mỗi giá trị liên quan của tất cả các thuộc tính của một lược đồ quan hệ.

Chẳng hạn quan hệ sau có 2 bộ.

MASV	HOTEN	NU	NGAYSINH	MALOP	TINH	HOCBONG
99001	TRAN DAN THU	TRUE	15-03-1977	CĐTH2B	TIEN GIANG	120000
99002	NGUYEN HA DA THAO	TRUE	25-04-1986	TCTH29C	TPHCM	120000

Thường người ta dùng các chữ cái thường (như t, p, q,...) để biểu diễn các bộ. Chẳng hạn để nói bộ t thuộc quan hệ r ta viết: $t \in r$.

Về trực quan thì mỗi quan hệ xem như một bảng, trong đó mỗi cột là thông tin về một thuộc tính, mỗi dòng là thông tin về một bộ. Chẳng hạn sau đây là các thể hiện của các quan hệ định nghĩa trên lược đồ cơ sở dữ liệu quản lý sinh viên ở trên:

rSv

MASV	HOTEN	NU	NGAYSINH	MALOP	TINH	HOCBONG
99001	TRAN DAN THU	TRUE	15-03-1977	CĐTH2B	TIEN GIANG	120000
99002	NGUYEN HA DA THAO	TRUE	25-04-1986	CĐTH2B	TPHCM	120000
99003	PHAM ANH HUY	FALSE	16-08-1977	CĐTH2B	BAC LIEU	
99004	NGUYEN NGOC THUAN	FALSE	24-12-1980	CĐTH2B	CA MAU	
99005	LE THANH TRUNG	FALSE	20-11-1978	CĐAV1	CA MAU	120000
99006	NGUYEN HONG VAN	FALSE	19-09-1979	CĐAV1	SOC TRANG	
99007	VU THI LOAN	TRUE	15-03-1975	CĐAV1	CA MAU	
99008	TRUONG KIM QUANG	FALSE	15-05-1975	CĐTH2B	HA NOI	
99009	TON THAT QUYEN	FALSE	26-06-1976	CĐTH2B	VUNG TAU	60000
99010	HA VAN LONG	FALSE	14-04-1973	CĐAV1	BAC LIEU	
99011	BUI VAN ANH	FALSE	22-12-1972	CĐAV1	AN GIANG	
99012	LE HUU CHI	FALSE	28-08-1977	CĐĐT2	CAN THO	60000
99013	VU THANH CONG	FALSE	29-03-1979	CĐTH2B	KIEN GIANG	60000
99014	TRAN QUANG CUONG	FALSE	30-05-1981	CĐĐT2	DONG THAP	120000
99015	PHAM VAN HAI	FALSE	30-06-1976	CĐĐT2	CA MAU	
99016	HUYNH THANH HOANG	FALSE	29-07-1982	CĐĐT2	TPHCM	80000
99017	TRAN MINH LAM	FALSE	21-08-1977	CĐTH2B	TRA VINH	
99018	PHAN VAN SANG	FALSE	19-05-1979	CĐDL1	DONG THAP	120000
99019	PHAM THI HUYEN	FALSE	16-06-1982	CĐDL1	CAN THO	120000
99020	NGUYEN THI NGAN	TRUE	11-11-1981	CĐTH2B	CA MAU	120000
99021	PHAM TAN QUANG	FALSE	01-01-1980	CĐDL1	CA MAU	
99022	TRAN PHUOC QUYEN	FALSE	12-12-1979	CĐTH2B	BAC LIEU	60000
99023	LE THI THANH VAN	TRUE	11-11-1980	CĐDL1	TPHCM	120000

rKh

MAKHOA	TENKHOA	SOCBGD
CNTT	CONG NGHE THONG TIN	60
AV	ANH VAN	60
HOA	HOA CHAT	20
MÁY TÍNH	MOI TRUONG	10
XD	XAY DUNG	10
DL	DU LICH	5
TTIN	TOAN - TIN HOC	30
SINH	CONG NGHE SINH HOC	30
VL	VAT LY	20

rMh

MAMH	TENMH	SOTIET
CSDL	CO SO DU LIEU	90
FOX	FOXPRO	120
THVP	TIN HOC VAN PHONG	90
AVTH	ANH VAN TIN HOC	60
KTS	KY THUAT SO	60
CTDL	CAU TRUC DU LIEU	60
TTNT	TRI TUE NHAN TAO	45
MANG	MANG MAY TINH CB	45
VB	VI SUAL BASIC	90

ĐT	DIỆN TỬ	20
----	---------	----

AC	ACCESS	180
LR	LAP RAP MAY TINH	60
INTER	CAC DỊCH VỤ INTERNET	45
HDH	HE DIEU HANH	60
KTLT	KY THUAT LAP TRINH	45
VIFOX	VISUAL FOXPRO	60

rKq

MASV	MAMH	DIEMTHI
99001	CSDL	3.0
99002	CSDL	8.0
99003	CSDL	8.0
99004	CSDL	3.0
99005	CSDL	2.0
99001	THVP	6.0
99002	THVP	3.0
99003	THVP	8.0
99004	THVP	9.0
99005	THVP	5.0

99006	MANG	6.0
99007	MANG	2.0
99008	MANG	7.0
99009	MANG	3.0
99010	TTNT	5.0
99011	FOX	4.0
99012	FOX	5.0
99013	FOX	7.0
99014	VB	7.0
99015	VB	3.0

99016	KTS	7.0
99017	KTS	7.0
99017	FOX	4.0
99017	MANG	8.0
99017	CSDL	8.0
99017	TTNT	6.0
99002	MANG	8.0
99004	MANG	4.0
99018	TTNT	7.0
99019	CSDL	8.0

99020	THVP	7.0
99021	MANG	7.0
99022	MANG	6.0
99023	CSDL	8.0
99023	MANG	6.0

99023	TTNT	3.0
99023	THVP	6.0
99023	FOX	8.0
99023	VB	9.0
99023	KTS	6.0

99021	CSDL	8.0
99021	THVP	9.0
99022	FOX	5.0
99022	TTNT	6.0

rLop

MALOP	TENLOP	SISO	MAKHOA
CĐTH2B	CAO ĐẲNG TIN HỌC KHOÁ 2000B	60	cntt
TCTH29C	TRUNG CẤP TIN HỌC KHOÁ 29 C	121	cntt
CĐAV1	CAO ĐẲNG ANH VĂN 1	120	av
CĐĐT2	CAO ĐẲNG ĐIỆN TỬ 2	80	đt
CĐDL1	CAO ĐẲNG DU LỊCH 1	42	dl

v Khóa (Key, Candidate Key):

Cho lược đồ quan hệ R , $S \subseteq R^+$. S được gọi là một **siêu khóa** (*superkey*) của lược đồ quan hệ R nếu với hai bộ tùy ý trong quan hệ R thì giá trị của các thuộc tính trong S là khác nhau.

Một lược đồ quan hệ có thể có nhiều siêu khóa. Siêu khóa chứa ít thuộc tính nhất được gọi là **khóa chỉ định**, trong trường hợp lược đồ quan hệ có nhiều khóa chỉ định, thì khóa được chọn để cài đặt gọi là **khóa chính** (*Primary key*) (trong các phần sau khóa chính được gọi tắt là khóa)

Các thuộc tính tham gia vào một khóa được gọi là **thuộc tính khóa** (*prime key*), ngược lại được gọi là **thuộc tính không khóa** (*non prime key*).

Một thuộc tính được gọi là **khóa ngoại** nếu nó là thuộc tính của một lược đồ quan hệ này nhưng lại là khóa chính của lược đồ quan hệ khác.

Ví dụ: Ta hãy xem lược đồ quan hệ sau:

Xe (SODANGBO, QUICACH, INHDANG, MAUSAC, SOSUON, SOMAY, MAXE, QUOCGIA)

Siêu khóa: (SOSUON, QUICACH),...

Khóa chỉ định: (SODANGBO, QUOCGIA), (SOSUON), (SOMAY), (MAXE)

Khóa chính: MAXE

Thuộc tính khóa: SODANGBO, QUOCGIA, SOSUON, SOMAY, MAXE

Thuộc tính không khóa: QUICACH, HINH DANG, MAUSAC

Khóa của Sv là (MASV), Khóa của Mh là (MAMH), khóa của Kh là (MAKHOA), khóa của Kq là (MASV, MAMH) khóa của Lop là MALOP, trong Lop thuộc tính MAKHOA là khóa ngoại

3 Các phép toán tập hợp (set operation)

i Phép hợp (Union operation)

Cho hai lược đồ quan hệ Q_1 và Q_2 có cùng tập thuộc tính $\{A_1, A_2, \dots, A_n\}$. r_1 và r_2 lần lượt là hai quan hệ trên Q_1 và Q_2 . Phép hợp của hai lược đồ quan hệ Q_1 và Q_2 sẽ tạo thành một lược đồ quan hệ Q_3 . Q_3 được xác định như sau:

$$Q_3^+ = \{A_1, A_2, \dots, A_n\}$$

$$r_3 = r_1 + r_2 = \{t \mid t \in r_1 \text{ hay } t \in r_2\}$$

Ví dụ:

r_1			r_2			$r_3 = r_1 + r_2$		
MASV	MAMH	DIEMTHI	MASV	MAMH	DIEMTHI	MASV	MAMH	DIEMTHI
99001	CSDL	5.0	99002	CTDL	2.0	99001	CSDL	5.0
99002	CTDL	2.0	99001	TTNT	5.0	99002	CTDL	2.0
99003	MANG	8.0	99003	CSDL	6.0	99003	MANG	8.0
						99001	TTNT	5.0
						99003	CSDL	6.0

ii Phép Giao (Intersection):

Cho hai lược đồ quan hệ Q_1 và Q_2 có cùng tập thuộc tính $\{A_1, A_2, \dots, A_n\}$. r_1 và r_2 lần lượt là hai quan hệ trên Q_1 và Q_2 . Phép giao của hai lược đồ quan hệ Q_1 và Q_2 sẽ tạo thành một lược đồ quan hệ Q_3 như sau:

$$Q_3^+ = \{A_1, A_2, \dots, A_n\}$$

$$r_3 = r_1 * r_2 = \{t \mid t \in r_1 \text{ và } t \in r_2\}$$

Ví dụ:

r_1			r_2			$r_3 = r_1 * r_2$		
MASV	MAMH	DIEMTHI	MASV	MAMH	DIEMTHI	MASV	MAMH	DIEMTHI
99001	CSDL	5.0	99002	CTDL	2.0	99002	CTDL	2.0
99002	CTDL	2.0	99001	TTNT	5.0			
99003	MANG	8.0	99003	CSDL	6.0			

iii Phép Trừ (Minus, difference)

Cho hai lược đồ quan hệ Q_1 và Q_2 có cùng tập thuộc tính $\{A_1, A_2, \dots, A_n\}$. r_1 và r_2 lần lượt là hai quan hệ trên Q_1 và Q_2 . Phép trừ lược đồ quan hệ Q_1 cho Q_2 sẽ tạo thành một lược đồ quan hệ Q_3 như sau:

$$Q_3^+ = \{A_1, A_2, \dots, A_n\}$$

$$r_3 = r_1 - r_2 = \{t \mid t \in r_1 \text{ và } t \notin r_2\}$$

Ví dụ:

r_1			r_2			$r_3 = r_1 - r_2$		
MASV	MAMH	DIEMTHI	MASV	MAMH	DIEMTHI	MASV	MAMH	DIEMTHI
99001	CSDL	5.0	99002	CTDL	2.0	99001	CSDL	5.0
99002	CTDL	2.0	99001	TTNT	5.0			
99003	MANG	8.0	99003	CSDL	6.0			

99001	CSDL	5.0	99002	CTDL	2.0	99001	CSDL	5.0
99002	CTDL	2.0	99001	TTNT	5.0	99003	MANG	8.0
99003	MANG	8.0	99003	CSDL	6.0			

iv Tích Descartes (Cartesian Product, product)

Cho hai lược đồ quan hệ $Q_1(A_1, A_2, \dots, A_n)$, $Q_2(B_1, B_2, \dots, B_m)$. r_1 và r_2 lần lượt là hai quan hệ trên Q_1 và Q_2 . Tích Descartes của hai lược đồ quan hệ Q_1 và Q_2 sẽ tạo thành một lược đồ quan hệ Q_3 như sau:

$$Q_3^+ = Q_1^+ \cup Q_2^+ = \{A_1, \dots, B_1, \dots\}$$

$$r_3 = r_1 \times r_2 = \{(t_1, t_2) \mid t_1 \in r_1 \text{ và } t_2 \in r_2\}$$

Ví dụ:

r_1			$r_3 = r_1 \times r_2$				
MASV	MAMH	DIEMTHI	MASV	MAMH	DIEMTHI	MAMH	TENMH
99001	CSDL	5.0	99001	CSDL	5.0	CSDL	CO SO DU LIEU
99002	CTDL	2.0	99001	CSDL	5.0	FOX	FOXPRO
99003	MANG	8.0	99002	CTDL	2.0	CSDL	CO SO DU LIEU
			99002	CTDL	2.0	FOX	FOXPRO
			99003	MANG	8.0	CSDL	CO SO DU LIEU
			99003	MANG	8.0	FOX	FOXPRO

r_2	
MAMH	TENMH
CSDL	CO SO DU LIEU
FOX	FOXPRO

4 Các phép toán quan hệ

i Phép Chiếu (Projection)

Cho một lược đồ quan hệ $Q(A_1, A_2, \dots, A_n)$. r là quan hệ trên Q . $X \subseteq Q^+$.

Phép chiếu của Q lên tập thuộc tính X sẽ tạo thành lược đồ quan hệ $Q' = Q[X]$, trong đó Q'^+ chính là X và r' chính là r nhưng chỉ lấy các thuộc tính của X .

$$Q'^+ = X$$

$$r' = r[X] = r.X = \{t' \mid \exists t \in r \text{ và } t.X = t'[X] = t'\}$$

phép chiếu chính là phép rút trích dữ liệu theo cột (chiều dọc)

Ví dụ:

r			$r' = r.\{MAMH\}$
MASV	MAMH	DIEMTHI	MAMH
99001	CSDL	5.0	CSDL
99002	CTDL	2.0	CTDL
99003	MANG	8.0	MANG

ii Phép Chọn (Selection)

Cho lược đồ quan hệ $Q(A_1, A_2, \dots, A_n)$, r là một quan hệ trên Q . $X \subset Q^+$ và E là một mệnh đề logic được phát biểu trên tập X . Phần tử $t \in r$ thỏa mãn điều kiện E ký hiệu là $t(E)$.

Phép chọn từ r theo điều kiện E sẽ tạo thành một lược đồ quan hệ Q' như sau:

$$Q'^+ = Q^+$$

$$r' = r(E) = r:E = \{t \mid t \in r \text{ và } t(E)\}$$

phép chọn chính là phép rút trích dữ liệu theo dòng (chiều ngang)

Ví dụ:

r			$r' = r:DIEMTHI \geq 5$		
MASV	MAMH	DIEMTHI	MASV	MAMH	DIEMTHI
99001	CSDL	5.0	99001	CSDL	5.0
99002	CTDL	2.0			
99003	MANG	8.0	99003	MANG	8.0

iii Phép kết, Phép Kết Tự Nhiên (join, natural join):

Cho hai lược đồ quan hệ $Q_1 (A_1, A_2, \dots, A_n), Q_2 (B_1, B_2, \dots, B_m)$.

r_1 và r_2 lần lượt là hai quan hệ trên Q_1 và Q_2 .

A_i và B_j lần lượt là các thuộc tính của Q_1 và Q_2 sao cho $MGT(A_i) = MGT(B_j)$ (MGT: miền giá trị).

θ là một phép so sánh trên $MGT(A_i)$.

Phép kết giữa Q_1 và Q_2 sẽ tạo thành một lược đồ quan hệ Q_3 như sau:

$$Q_3^+ = Q_1^+ \cup Q_2^+$$

$$r_3 = r_1 \bowtie_{A_i \theta B_j} r_2 = \{t_{12} \mid \exists t_1 \in r_1, \exists t_2 \in r_2 \text{ sao cho}$$

$$t_{12} \cdot Q_1^+ = t_1$$

$$t_{12} \cdot Q_2^+ = t_2$$

$$t_1 \cdot A_i \theta t_2 \cdot B_j\}$$

Ta rút ra các bước cụ thể để thực hiện phép kết như sau:

- Tạo tích descartes
- Thực hiện phép chọn theo điều kiện $E=A_i \theta B_j$

Ví dụ:

A_i là thuộc tính B, B_j là thuộc tính F và θ là phép so sánh " \geq ". Ta được kết quả là quan hệ sau:

r_1			r_2			$r_3 = r_1 \bowtie_{B \geq F} r_2$					
A	B	C	E	F	H	A	B	C	E	F	H
6	5	4	1	5	9	6	5	4	1	5	9
7	5	5	4	6	8	6	5	4	7	5	3
4	2	6	7	5	3	7	5	5	1	5	9
						7	5	5	7	5	3

Nếu θ được sử dụng trong phép kết là phép so sánh bằng ($=$) thì ta gọi là phép kết bằng. Hơn nữa nếu $A_i = B_j$ thì phép kết bằng này được gọi là **phép kết tự nhiên**. Phép kết tự nhiên là một phép kết thường dùng nhất trong thực tế.

Ví dụ: Với $A_i = B_j = \text{MAMH}$

r_1			r_2		$r_3 = r_1 \bowtie_{MAMH} r_2$			
MASV	MAMH	DIEMTHI	MAMH	TENMH	MASV	MAMH	DIEMTHI	TENMH
99001	CSDL	5.0	CSDL	CO SO DU LIEU	99001	CSDL	5.0	CO SO DU LIEU
99002	CTDL	2.0	CTDL	CAU TRUC DU LIEU	99002	CTDL	2.0	CAU TRUC DU LIEU
99003	MANG	8.0						

iv Phép chia (division):

Cho hai lược đồ quan hệ $Q_1 (A_1, A_2, \dots, A_n), Q_2 (B_1, B_2, \dots, B_m)$.

r_1 và r_2 lần lượt là hai quan hệ trên Q_1 và Q_2 .

A_i và B_j lần lượt là các thuộc tính của Q_1 và Q_2 sao cho $n > m$.

Phép chia Q_1 và Q_2 sẽ tạo thành một lược đồ quan hệ Q_3 như sau:

$$Q_3^+ = \{A_1, \dots, A_{n-m}\}$$

$$r_3 = r_1 \div r_2 = \{t_3 \mid \forall t_2 \in r_2, \exists t_1 \in r_1 \text{ } t_3 = t_1 \cdot \{A_1, \dots, A_{n-m}\}$$

$$t_2 = t_1 \cdot \{A_{n-m+1}, \dots, A_n\}\}$$

Ví dụ:

r_1					r_2		$r_3 = r_1 \div r_2$		
A_1	A_2	A_3	A_4	A_5	B_1	B_2	A_1	A_2	A_3
a	b	d	c	g	c	g	a	b	d

a	b	d	e	f	e	f	e	g	c
b	c	e	e	f					
e	g	c	c	g					
e	g	c	e	f					
a	b	e	g	e					

5 Các tính chất của đại số quan hệ

Q là lược đồ quan hệ

q,r,s là quan hệ trên Q,

E,E₁,E₂ là mệnh đề logic trên Q⁺

X₁ ⊆ X₂ ⊆ Q⁺

Hãy chứng minh các tính chất sau:

$$(r:E_1):E_2 = (r:E_2):E_1$$

Chứng minh:

$$\begin{aligned} (r:E_1):E_2 &= \{t' \mid t' \in (r:E_1) \text{ và } t'(E_2)\} \\ &= \{t' \mid t' \in \{t \mid t \in r \text{ và } t(E_1)\} \text{ và } t'(E_2)\} \\ &= \{t' \in r \mid t'(E_1) \text{ và } t'(E_2)\} \\ &= \{t' \mid t' \in \{t \mid t \in r \text{ và } t(E_2)\} \text{ và } t'(E_1)\} \\ &= \{t' \mid t' \in (r:E_2) \text{ và } t'(E_1)\} \\ &= (r:E_2):E_1 \end{aligned}$$

$$(r+s):E = (r:E)+(s:E)$$

Chứng minh:

$$\begin{aligned} (r+s):E &= \{t \mid t \in (r+s) \text{ và } t(E)\} \\ &= \{t \mid t \in \{t' \mid t' \in r \text{ hoặc } t' \in s\} \text{ và } t(E)\} \\ &= \{t \mid (t \in r \text{ hoặc } t \in s) \text{ và } t(E)\} \\ &= \{t \mid (t \in r \text{ và } t(E)) \text{ hoặc } (t \in s \text{ và } t(E))\} \\ &= \{t \mid t \in \{t' \mid t' \in r \text{ và } t'(E)\} \text{ hoặc } t \in \{t' \mid t' \in s \text{ và } t'(E)\}\} \\ &= \{t \mid t \in (r:E) \text{ hoặc } t \in (s:E)\} \\ &= (r:E)+(s:E) \end{aligned}$$

$$(r*s):E = (r:E)*(s:E)$$

Chứng minh:

$$\begin{aligned} (r*s):E &= \{t \mid t \in (r*s) \text{ và } t(E)\} \\ &= \{t \mid t \in \{t' \mid t' \in r \text{ và } t' \in s\} \text{ và } t(E)\} \\ &= \{t \mid t \in r \text{ và } t \in s \text{ và } t(E)\} \\ &= \{t \mid (t \in r \text{ và } t(E)) \text{ và } (t \in s \text{ và } t(E))\} \\ &= \{t \mid t \in \{t' \mid t' \in r \text{ và } t'(E)\} \text{ và } t \in \{t' \mid t' \in s \text{ và } t'(E)\}\} \\ &= \{t \mid t \in (r:E) \text{ và } t \in (s:E)\} \\ &= (r:E)*(s:E) \end{aligned}$$

$$(r-s):E = (r:E)-(s:E)$$

Chứng minh:

$$\begin{aligned} (r-s):E &= \{t \mid t \in (r-s) \text{ và } t(E)\} \\ &= \{t \mid t \in \{t' \mid t' \in r \text{ và } t' \notin s\} \text{ và } t(E)\} \\ &= \{t \mid t \in r \text{ và } t \notin s \text{ và } t(E)\} \end{aligned}$$

$$\begin{aligned}
&= \{t \mid (t \in r \text{ và } t(E)) \text{ và } (t \notin s \text{ và } t(E))\} \\
&= \{t \mid t \in \{t' \mid t' \in r \text{ và } t'(E)\} \text{ và } t \notin \{t' \mid t' \in s \text{ và } t'(E)\}\} \\
&= \{t \mid t \in (r:E) \text{ và } t \notin (s:E)\} \\
&= (r:E) * (s:E)
\end{aligned}$$

$$\text{Với } X_2 \supseteq X_1 \Rightarrow (r.X_2).X_1 = r.X_1$$

Chứng minh:

$$\begin{aligned}
(r.X_2).X_1 &= \{t.X_1 \mid t \in (r.X_2)\} \\
&= \{t.X_1 \mid t \in \{t'.X_2 \mid t' \in r\}\} \\
&= \{(t'.X_2).X_1 \mid t' \in r\} \\
&= \{t'.X_1 \mid t' \in r\} \text{ vì } X_1 \subseteq X_2 \\
&= r.X_1
\end{aligned}$$

$$E \text{ phát biểu trên } X \Rightarrow (r:E).X = (r.X):E$$

Chứng minh:

$$\begin{aligned}
(r:E).X &= \{t.X \mid t \in (r:E)\} \\
&= \{t.X \mid t \in \{t' \mid t' \in r \text{ và } t'(E)\}\} \\
&= \{t.X \mid t \in r \text{ và } t(E)\} \\
&= \{t' \mid t' \in \{t.X \mid t \in r\} \text{ và } t'(E)\} \\
&= \{t' \mid t' \in (r.X) \text{ và } t'(E)\} \\
&= (r.X):E
\end{aligned}$$

$$q \mid \gg \mid r = r \mid \gg \mid q$$

Chứng minh:

$$\begin{aligned}
(q \mid \gg \mid r) &= \{t_{12} \mid \exists t_1 \in q, \exists t_2 \in r \ t_{12}.Q^+ = t_1, \ t_{12}.R^+ = t_2 \ t_{12}.A_i \theta_{12}.B_j\} \\
&= r \mid \gg \mid q
\end{aligned}$$

$$A_i \in Q, B_j \in S, C_k \in Q, D_1 \in R \Rightarrow (q \mid \gg \mid r) \mid \gg \mid s = q \mid \gg \mid (r \mid \gg \mid s)$$

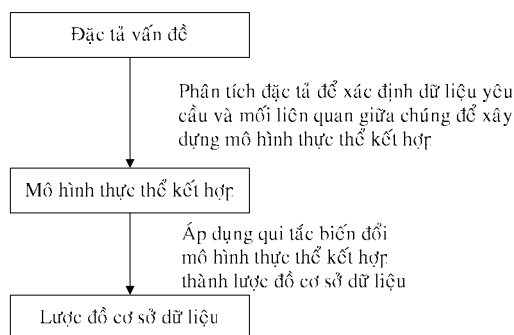
Chứng minh:

$$\begin{aligned}
(q \mid \gg \mid r) \mid \gg \mid s &= \{t_{12} \mid \exists t_1 \in (q \mid \gg \mid r), \exists t_2 \in s \ t_{12}.Q^+ \cup R^+ = t_1 \\
&\quad t_{12}.S^+ = t_2 \ t_{12}.A_i \theta_{12}.B_j\} \\
&= \{t_{12} \mid \exists t_1 \in \{u_{12} \mid \exists u_1 \in q, \exists u_2 \in r \ u_{12}.Q^+ = u_1 \ u_{12}.R^+ = u_2 \ u_1.C_k \theta_{12}.D_1\}, \\
&\quad \exists t_2 \in s \ t_{12}.Q^+ \cup R^+ = t_1, \ t_{12}.S^+ = t_2 \ t_1.A_i \theta_{12}.B_j\} \\
&= \{t_{123} \mid \exists t_1 \in q, \exists t_2 \in r, \exists t_3 \in s \ t_{123}.Q^+ = t_1, \ t_{123}.R^+ = t_2 \ t_{123}.S^+ = t_3 \\
&\quad t_{123}.A_i \theta_{123}.B_j \ t_{123}.C_k \theta_{123}.D_1\} \\
&= \{t_{12} \mid \exists t_1 \in q, \exists t_2 \in \{u_{12} \mid \exists u_1 \in r, \exists u_2 \in s \ u_{12}.R^+ = u_1 \\
&\quad u_{12}.S^+ = u_2 \ u_1.C_k \theta_{12}.D_1\}, \ t_{12}.Q^+ = t_1 \ t_{12}.R^+ \cup S^+ = t_2 \ t_{12}.A_i \theta_{12}.B_j\} \\
&= \{t_{12} \mid \exists t_1 \in q, \exists t_2 \in (r \mid \gg \mid s), \ t_{12}.Q^+ = t_1 \\
&\quad t_{12}.R^+ \cup S^+ = t_2 \ t_{12}.A_i \theta_{12}.B_j\} \\
&= q \mid \gg \mid (r \mid \gg \mid s)
\end{aligned}$$

IV MÔ HÌNH THỰC THỂ KẾT HỢP

1 Giới thiệu mô hình thực thể kết hợp

Các nhà phân tích thiết kế hệ thống thông tin thường xây dựng *lược đồ cơ sở dữ liệu* (C1.III.2.ii) từ mô hình thực thể kết hợp và mô hình này lại được xây dựng từ phần đặc tả vấn đề của một bài toán thực tế.



Hình 1.4.1 - Lược đồ CSDL xây dựng theo hướng Phân tích thiết kế

Lược đồ cơ sở dữ liệu xây dựng theo hướng này thông thường đạt tối thiểu dạng chuẩn 3 (3NF: third normal form) nghĩa là ở dạng có sự dư thừa dữ liệu ở mức tối thiểu, còn môn CSDL xây dựng lược đồ CSDL đạt dạng chuẩn 3 từ lược đồ cơ sở dữ liệu chưa đạt dạng chuẩn có kèm các *tân từ* (C1.III.2.ii). Ta hãy xem ví dụ sau:

i Ví dụ – Mối quan hệ một-nhiều

(a) Đặc tả vấn đề

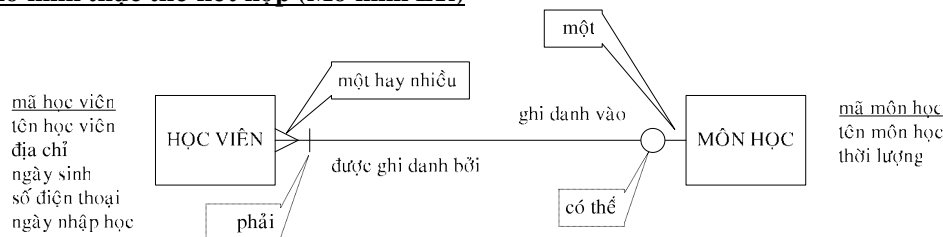
Những người phụ trách đào tạo của *Trường cao đẳng cộng đồng núi Ayers* mong muốn tạo lập một CSDL về các môn đào tạo của trường (như: chứng chỉ leo núi, công nghệ bay) và học viên ghi danh vào những môn học này. Trường cũng có qui định là cùng một lúc, học viên chỉ có thể ghi danh vào một môn học. Họ chỉ quan tâm về dữ liệu của đợt ghi danh hiện tại. Một khi học viên kết thúc môn học thì nhà trường sẽ không còn quan tâm đến họ và những học viên này phải được xóa khỏi CSDL. Thông tin cần lưu trữ về một học viên bao gồm: mã học viên, tên học viên, địa chỉ, ngày sinh, số điện thoại, ngày nhập học

Thông tin về môn học gồm mã môn học, tên môn học, thời lượng

Phân tích:

- phần đặc tả vấn đề chứa đựng các qui tắc quản lý và dữ liệu yêu cầu của vấn đề.
- dữ liệu của vấn đề là: chi tiết về *học viên có mã học viên, tên học viên, địa chỉ, ngày sinh, số điện thoại* và *ngày nhập học* chi tiết về *môn học có mã môn học, tên môn học và thời lượng*.
- qui tắc quản lý gồm:
 - + Cùng một lúc, một học viên chỉ có thể ghi danh vào một môn học.
 - + Nhiều học viên có thể ghi danh vào một môn học.
 - + Nhà trường chỉ quan tâm đến những học viên của môn học hiện tại.

(b) Mô hình thực thể kết hợp (Mô hình ER)



Hình 1.4.2- Mô hình ER của vấn đề học viên/môn học Trường cao đẳng cộng đồng núi Ayers

Các tính chất trong mô hình thực thể kết hợp:

- Hình chữ nhật được gọi là *tập thực thể*. Tên của tập thực thể được ghi bên trong hình chữ nhật và dùng danh từ để đặt tên cho tập thực thể.

- Đường nối giữa hai tập thực thể được gọi là **mối quan hệ (mối kết hợp)**. Mối quan hệ trong vấn đề trên là mối quan hệ một-nhiều (1:M). Nội dung của mối quan hệ được diễn tả theo hai chiều: “ghi danh vào”, “được ghi danh bởi” và chúng diễn tả hai nội dung sau:
 - + Mỗi HỌC VIÊN có thể ghi danh vào một MÔN HỌC
 - + Mỗi MÔN HỌC phải được ghi danh bởi một hay nhiều HỌC VIÊN
- Các dữ liệu ghi bên cạnh tập thực thể được gọi là thuộc tính. Chúng cung cấp thông tin chi tiết về tập thực thể. Có hai loại thuộc tính:
 - Thuộc tính nhận diện là thuộc tính để phân biệt thực thể này với thực thể kia trong tập thực thể.
 - Thuộc tính mô tả là thuộc tính cung cấp thông tin chi tiết hơn về thực thể trong tập thực thể.
- Mối quan hệ của vấn đề trên là mối quan hệ một-nhiều. Tính chất này của mối quan hệ gọi là **tính kết nối** của mối quan hệ. Tính kết nối một-nhiều rất phổ biến trong mô hình thực thể kết hợp. Hai loại kết nối còn lại ít phổ biến hơn nhưng không kém phần quan trọng là mối quan hệ một-một và mối quan hệ nhiều-nhiều.

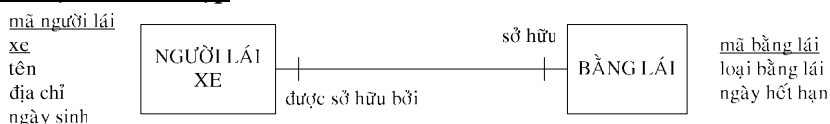
ii Ví dụ – mối quan hệ một-một

(a) Đặc tả vấn đề

Phòng cảnh sát mong muốn quản lý lý lịch cá nhân những người lái xe và bằng lái của họ. Một người chỉ lấy được một bằng lái và một bằng lái chỉ thuộc về một người. Thông tin về lái xe mà phòng cảnh sát quan tâm là: mã người lái xe, tên, địa chỉ, ngày sinh

Thông tin về bằng lái cần lưu trữ là: mã bằng lái, loại bằng lái, ngày hết hạn

(b) Mô hình thực thể kết hợp



Hình 1.4.3 - Mô hình ER của vấn đề người lái xe và bằng lái

- mỗi NGƯỜI LÁI XE phải sở hữu một BẰNG LÁI
- mỗi BẰNG LÁI phải được sở hữu bởi một NGƯỜI LÁI XE

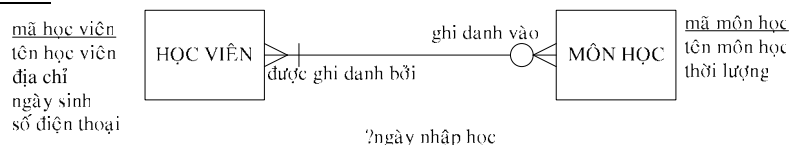
iii Ví dụ – mối quan hệ nhiều-nhiều

(a) Đặc tả vấn đề

Người phụ trách đào tạo Trường cao đẳng cộng đồng núi xanh mong muốn thiết lập một csdl về các môn học mà họ cung cấp (như chứng chỉ leo núi, cử nhân công nghệ bay) và các học viên ghi danh vào các môn học này. Nhà trường qui định là một học viên được ghi danh học tối đa ba môn học trong cùng một lúc. Họ chỉ quan tâm đến dữ liệu của môn học hiện tại. Một khi học viên kết thúc môn học, họ sẽ không còn thuộc diện quản lý của nhà trường và phải được xóa khỏi csdl trừ khi học viên này ghi danh học tiếp môn mới. Thông tin về một học viên gồm: mã học viên, tên học viên, địa chỉ, ngày sinh, số điện thoại, ngày nhập học

Thông tin về môn học gồm: mã môn học, tên môn học, thời lượng

(b) Mô hình ER



Hình 1.4.4 - Mô hình ER của vấn đề Trường cao đẳng cộng đồng núi xanh

- + **Mỗi** HỌC VIÊN có thể ghi danh vào **một hay nhiều** MÔN HỌC

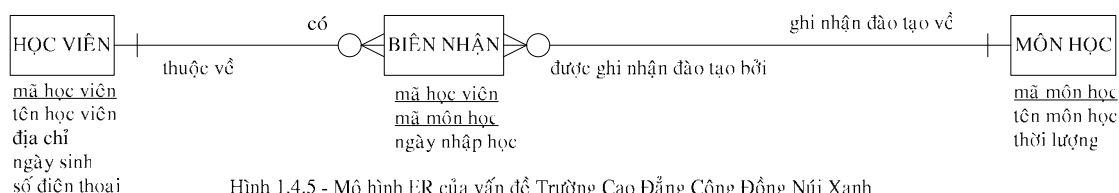
+ **Mỗi** MÔN HỌC phải được ghi danh bởi **một hay nhiều** HỌC VIÊN

Mô hình ER trên có mối quan hệ nhiều-nhiều.

(c)Loại bỏ tính kết nối nhiều-nhiều (nếu được)

Mô hình trên gặp phải khuyết điểm sau:

- **Ngày nhập học** là thuộc tính gắn liền với tập thực thể HỌC VIÊN sẽ không hợp lý vì không diễn tả được trường hợp học viên học cùng lúc nhiều môn học.
- Còn nếu **ngày nhập học** là thuộc tính của MÔN HỌC thì không diễn tả được tình trạng cùng môn học nhưng có các ngày nhập học khác nhau.



Hình 1.4.5 - Mô hình ER của vấn đề Trường Cao Đẳng Cộng Đồng Núi Xanh

Để giải quyết vấn đề này ta phải đưa vào:

- một tập thực thể làm trung gian giữa HỌC VIÊN và MÔN HỌC gọi là tập kết hợp PHIẾU GHI DANH.
- Thuộc tính nhận diện của tập kết hợp là sự kết hợp giữa thuộc tính nhận diện của tập thực thể HỌC VIÊN và MÔN HỌC
- thuộc tính mô tả của tập kết hợp PHIẾU GHI DANH là ngày nhập học
- tính kết nối của tập kết hợp với tập thực thể là một-nhiều

Nội dung của mối quan hệ giữa các tập thực thể là:

- mỗi HỌC VIÊN có thể có một hay nhiều PHIẾU GHI DANH
- mỗi PHIẾU GHI DANH phải thuộc về một HỌC VIÊN
- mỗi PHIẾU GHI DANH phải ghi nhận đào tạo về một MÔN HỌC
- mỗi MÔN HỌC có thể được ghi nhận đào tạo bởi một hay nhiều PHIẾU GHI DANH

Các qui tắc phải tuân thủ khi thêm tập kết hợp làm trung gian để loại bỏ tính kết nối nhiều-nhiều:

- Phải nhận diện được thuộc tính mô tả của tập kết hợp.
- Nếu có thuộc tính mô tả thì tạo tập kết hợp làm trung gian giữa hai tập thực thể.
- Nếu không có thuộc tính mô tả thì vẫn giữ nguyên mô hình như hình 1.4.4

2 Chuyển từ mô hình thực thể kết hợp sang lược đồ CSDL.

i Qui tắc chung

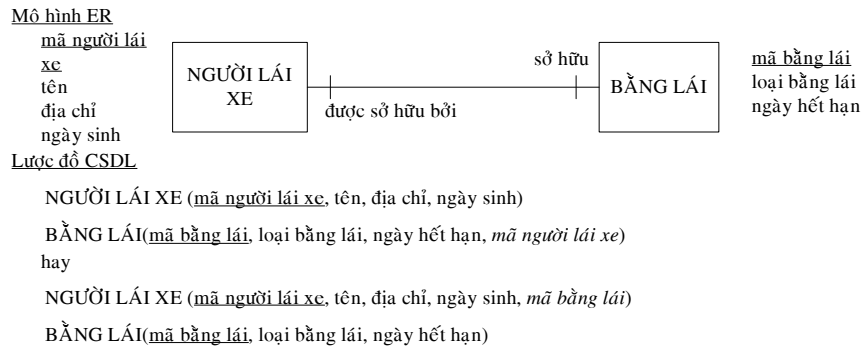
Khi biến đổi mô hình ER thành các mô hình quan hệ ta áp dụng các qui tắc sau:

- Mỗi tập thực thể trong mô hình ER được chuyển thành một lược đồ quan hệ.
- Mỗi thuộc tính trong mô hình ER được chuyển thành thuộc tính trong lược đồ quan hệ tương ứng
- Mỗi thuộc tính nhận diện trong mô hình ER được chuyển thành khóa chính trong lược đồ quan hệ tương ứng.
- Mỗi mối quan hệ trong ER được chuyển thành khóa ngoại theo qui tắc sau

ii Qui tắc chuyển mối quan hệ thành khóa ngoại

(a) Mối quan hệ một-một

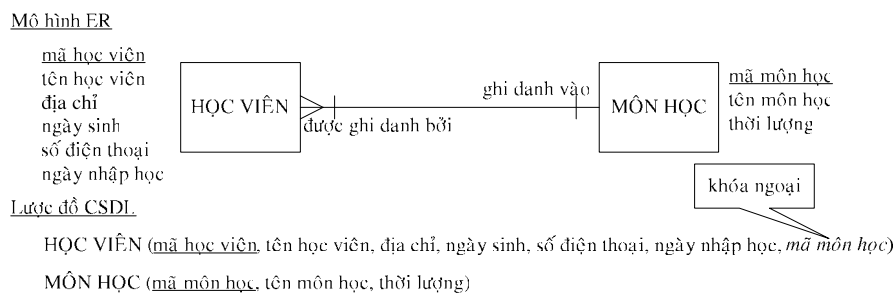
Chuyển khóa chính từ quan hệ 1 sang quan hệ 2 hay ngược lại. Ví dụ vấn đề người lái xe và bằng lái sẽ có mô hình quan hệ là một trong hai mô hình quan hệ sau



Hình 1.4.6 - Mô hình ER và mô hình quan hệ của vấn đề Người lái xe và bằng lái

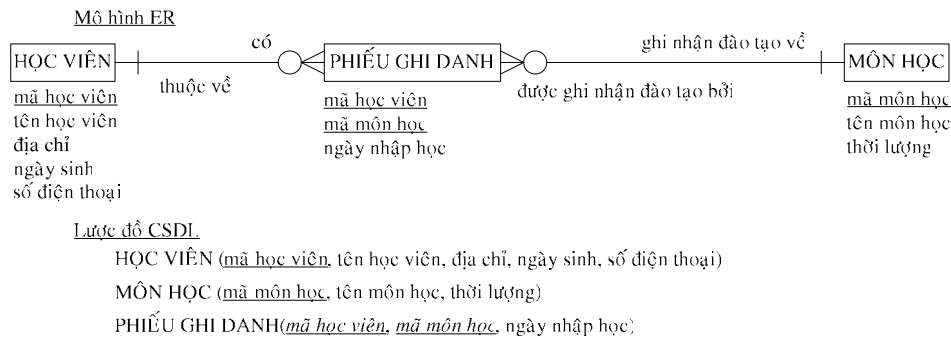
(b)Mối quan hệ một-nhiều

Chuyển khóa chính từ bên một sang bên nhiều.



Hình 1.4.7 - Mô hình ER và mô hình quan hệ của vấn đề Trường cao đẳng cộng đồng núi Ayers

(c)Mối quan hệ nhiều-nhiều đến tập kết hợp



Hình 1.4.8 - Mô hình ER và mô hình quan hệ của vấn đề Trường Cao Đẳng Cộng Đồng Núi Xanh

Trong quan hệ PHIẾU GHI DANH có các khóa chính khóa ngoại như sau:

- + mã học viên là khóa ngoại
- + mã môn học là khóa ngoại
- + mã học viên và mã môn học là khóa chính

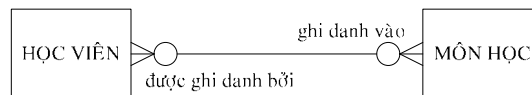
(d)Mối quan hệ nhiều-nhiều

Tạo một quan hệ mới có khóa chính là sự kết hợp các khóa chính của hai quan hệ có tính kết nối nhiều-nhiều.

Ví dụ giả sử Trường Cao Đẳng Cộng Đồng Núi Xanh không quan tâm đến ngày nhập học của học viên thì mô hình ER sẽ có mối quan hệ nhiều-nhiều như sau:

Mô hình ER

mã học viên
tên học viên
địa chỉ
ngày sinh
số điện thoại



mã môn học
tên môn học
thời lượng

Mô hình quan hệ

HOC VIÊN(mã học viên, tên học viên, địa chỉ, ngày sinh, số điện thoại)

MÔN HỌC(mã môn học, tên môn học, thời lượng)

HOC VIÊN MÔN HỌC(mã học viên, mã môn học)

Hình 1.4.9 - Mô hình ER và mô hình quan hệ

của vấn đề Trường cao đẳng cộng đồng núi xanh không có thuộc tính ngày nhập học

V BÀI TẬP1 Phép toán tập hợp và phép toán quan hệ

Cho lược đồ cơ sở dữ liệu dùng để quản lý hồ sơ sinh viên bao gồm các quan hệ Sv(sinh viên), Lop(Lớp), kh(khoa), Mh(môn học), Kq(kết quả) được mô tả bởi các lược đồ quan hệ như sau:

Sv (MASV, HOTEN, NU, NGAYSINH, MALOP, TINH, HOCBONG)

Tân từ: Mỗi sinh viên có mỗi MASV duy nhất. Mỗi MASV xác định tất cả các thuộc tính còn lại của sinh viên đó.

Lop (MALOP, TENLOP, SISO, MAKHOA)

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

Kh (MAKHOA, TENKHOA, SOCBGD)

Tân từ: Mỗi khoa có mỗi MAKHOA duy nhất. Mỗi MAKHOA xác định tất cả các thuộc tính còn lại của khoa đó.

Mh (MAMH, TENMH, SOTIET)

Tân từ: Mỗi Môn học có một MAMH duy nhất. Mỗi MAMH xác định tất cả các thuộc tính còn lại của môn học đó.

Kq (MASV, MAMH, DIEMTHI)

Tân từ: Mỗi sinh viên cùng với một môn học xác định duy nhất một điểm thi

YÊU CẦU:

1. Tìm khóa cho mỗi lược đồ quan hệ trên.
2. Hãy thực hiện các câu hỏi sau bằng ngôn ngữ đại số quan hệ
 - a. Lập danh sách sinh viên gồm MASV, HOTEN, HOCBONG
 - b. Lập danh sách sinh viên nữ khoa 'CNTT', danh sách cần MASV, HOTEN, HOCBONG
 - c. Lập bảng điểm cho tất cả sinh viên khoa 'CNTT', bảng điểm gồm các cột MASV, HOTEN, TENMH, DIEMTHI
 - d. Lập phiếu điểm cho sinh viên có MASV="99001"
 - e. Lập danh sách sinh viên gồm MASV, HOTEN, TENLOP, TENKHOA
 - f. Lập bảng điểm môn học có mã môn học là CSDL cho tất cả sinh viên có mã lớp là "CĐTH2B"
 - g. Lập danh sách sinh viên của lớp có mã lớp là "CĐTH2B" và có điểm thi môn học lớn hơn hay bằng 8.

2 Mô hình thực thể kết hợp

Dựa vào các phân tích sơ bộ dưới đây, hãy lập mô hình thực thể kết hợp cho mỗi bài toán quản lý sau:

i QUẢN LÝ LAO ĐỘNG

Để quản lý việc phân công các nhân viên tham gia vào xây dựng các công trình. Công ty xây dựng ABC tổ chức quản lý như sau:

Cùng lúc công ty có thể tham gia xây dựng nhiều công trình, mỗi công trình có một mã số công trình duy nhất (MACT), mỗi mã số công trình xác định các thông tin như: tên gọi công trình (TENCT), địa điểm(ĐIADIEM), ngày công trình được cấp giấy phép xây dựng (NGAYCAPGP), ngày khởi công (NGAYKC), ngày hoàn thành (NGAYHT).

Mỗi nhân viên của công ty ABC có một mã số nhân viên(MANV) duy nhất, một mã số nhân viên xác định các thông tin như: Họ tên (HOTEN), ngày sinh (NGAYSINH), phái (PHAI), địa chỉ (ĐIACHI). Mỗi nhân viên phải chịu sự quản lý hành chính bởi một phòng ban. Tất nhiên một phòng ban quản lý hành chính nhiều nhân viên. Công ty có nhiều phòng ban (Phòng kế toán, phòng kinh doanh, phòng kỹ thuật, phòng tổ chức, phòng chuyên môn, Phòng phục vụ....). Mỗi phòng ban có một mã số phòng ban(MAPB) duy nhất, mã phòng ban xác định tên phòng ban (TENPB).

Công ty phân công các nhân viên tham gia vào các công trình, mỗi công trình có thể được phân cho nhiều nhân viên và mỗi nhân viên cùng lúc cũng có thể tham gia vào nhiều công trình. Với mỗi công trình một nhân viên có một số lượng ngày công (SLNGAYCONG) đã tham gia vào công trình đó.

ii QUẢN LÝ THƯ VIỆN

Một thư viện tổ chức việc cho mượn sách như sau:

Mỗi quyển sách được đánh một mã sách (MASH) dùng để phân biệt với các quyển sách khác (giả sử nếu một tác phẩm có nhiều bản giống nhau hoặc có nhiều tập thì cũng xem là có mã sách khác nhau), mỗi mã sách xác định các thông tin khác như : tên sách (TENSACH), tên tác giả (TACGIA), nhà xuất bản (NHAXB), năm xuất bản (NAMXB).

Mỗi đọc giả được thư viện cấp cho một thẻ thư viện, trong đó có ghi rõ mã đọc giả (MADG), cùng với các thông tin khác như : họ tên (HOTEN), ngày sinh (NGAYSINH), địa chỉ (ĐIACHI), nghề nghiệp(NGHENGHIEP).

Cứ mỗi lượt mượn sách, đọc giả phải ghi các quyển sách cần mượn vào một phiếu mượn, mỗi phiếu mượn có một số phiếu mượn (SOPM) duy nhất, mỗi phiếu mượn xác định các thông tin như: ngày mượn (NGAYMUON), đọc giả mượn, các quyển sách mượn và ngày trả (NGAYTRA). Các các quyển sách trong cùng một phiếu mượn không nhất thiết phải trả trong cùng một ngày.

iii QUẢN LÝ BÁN HÀNG

Mỗi khách hàng có một mã khách hàng (MAKH) duy nhất, mỗi MAKH xác định được các thông tin về khách hàng như : họ tên khách hàng (HOTEN), địa chỉ (ĐIACHI), số điện thoại (ĐIENTHOAI). Các mặt hàng được phân loại theo từng nhóm hàng, mỗi nhóm hàng có một mã nhóm (MANHOM) duy nhất, mỗi mã nhóm hàng xác định tên nhóm hàng (TENNHOM), tất nhiên một nhóm hàng có thể có nhiều mặt hàng. Mỗi mặt hàng được đánh một mã số (MAHANG) duy nhất, mỗi mã số này xác định các thông tin về mặt hàng đó như : tên hàng (TENHANG), đơn giá bán (ĐONGIA), đơn vị tính (ĐVT). Mỗi hóa đơn bán hàng có một số hóa đơn (SOHD) duy nhất, mỗi hóa đơn xác định được khách hàng và ngày lập hóa đơn (NGAYLAPHĐ), ngày bán hàng (NGAYBAN). Với mỗi mặt hàng trong một hóa đơn cho biết số lượng bán (SLBAN) của mặt hàng đó.

iv QUẢN LÝ LỊCH DẠY - HỌC

Để quản lý lịch dạy của các giáo viên và lịch học của các lớp, một trường tổ chức như sau:

Mỗi giáo viên có một mã số giáo viên (MAGV) duy nhất, mỗi MAGV xác định các thông tin như: họ và tên giáo viên (HOTEN), số điện thoại (DTGV). Mỗi giáo viên có thể dạy nhiều môn cho nhiều khoa nhưng chỉ thuộc sự quản lý hành chính của một khoa nào đó.

Mỗi môn học có một mã số môn học (MAMH) duy nhất, mỗi môn học xác định tên môn học (TENMH). Ứng với mỗi lớp thì mỗi môn học chỉ được phân cho một giáo viên.

Mỗi phòng học có một số phòng học (SOPHONG) duy nhất, mỗi phòng có một chức năng (CHUCNANG); chẳng hạn như phòng lý thuyết, phòng thực hành máy tính, phòng nghe nhìn, xưởng thực tập cơ khí,...

Mỗi khoa có một mã khoa (MAKHOA) duy nhất, mỗi khoa xác định các thông tin như: tên khoa (TENKHOA), điện thoại khoa (DTKHOA).

Mỗi lớp có một mã lớp (MALOP) duy nhất, mỗi lớp có một tên lớp (TENLOP), sĩ số lớp (SISO). Mỗi lớp có thể học nhiều môn của nhiều khoa nhưng chỉ thuộc sự quản lý hành chính của một khoa nào đó.

Hàng tuần, mỗi giáo viên phải lập lịch báo giảng cho biết giáo viên đó sẽ dạy những lớp nào, ngày nào (NGAYDAY), môn gì?, tại phòng nào, từ tiết nào (TUTIET) đến tiết nào (DENTIET), tựa đề bài dạy (BAIDAY), ghi chú (GHICHU) về các tiết dạy này, đây là giờ dạy lý thuyết (LYTHUYET) hay thực hành - giả sử nếu LYTHUYET=1 thì đó là giờ dạy thực hành và nếu LYTHUYET=2 thì đó là giờ lý thuyết, một ngày có 16 tiết, sáng từ tiết 1 đến tiết 6, chiều từ tiết 7 đến tiết 12, tối từ tiết 13 đến 16.

-----oOo-----

Chương 2 .

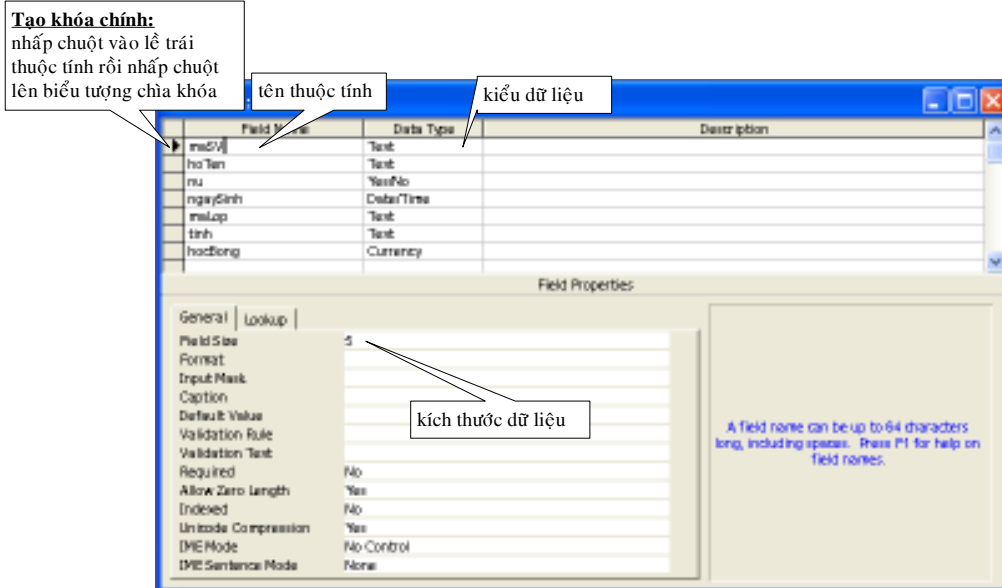
NGÔN NGỮ TRUY VẤN SQL

I CÁCH TẠO QUAN HỆ BẰNG ACCESS

Microsoft Access là một hệ quản trị cơ sở dữ liệu quan hệ. Ta có thể dùng HQTCSDL Access hay gọi tắt là Access để thực hành mô hình quan hệ.

Sau đây ta sẽ tạo lược đồ csdl quản lý sinh viên như ở trang 6 của tài liệu.

- + Khởi động Access: **Start-> Programs-> Microsoft Access.**
- + Tạo lược đồ csdl rỗng có tên là qLSV: **Blank Database->OK->qLSV->Create**
- + Tạo quan hệ bằng cách: **Tables->New->Design View->OK**



- Data type là Text (kiểu chuỗi), Yes/No (Kiểu luận lý), Date/Time (kiểu ngày tháng), Currency (kiểu số)
- Field size là kích thước kiểu dữ liệu
- + Khi tạo xong các thuộc tính và khóa chính ta lưu lại và đặt tên cho quan hệ Sv bằng cách : **File->Close->Yes->Sv->OK.**
- + Tạo các quan hệ còn lại theo bước 3 và 4
- + Nhập dữ liệu cho quan hệ Sv bằng cách: **Tables->Sv->Open**

II CÂU LỆNH TRUY VẤN

1 BIỂU THỨC (EXPRESSION)

Các thành phần tạo nên biểu thức bao gồm:

Literal value

Là các dữ liệu có giá trị đúng như văn bản thể hiện.

Dữ liệu chuỗi có dạng: “New York”

Dữ liệu số có dạng: 1056; 1056.25

Dữ liệu ngày có dạng: #1-Jan-94#; #12/2/2001#

Constant

Là một tên đại diện cho một giá trị không thay đổi như :

Const	Giá trị đại diện
-------	------------------

True	-1
False	0
Null	0

Toán tử số học:

Toán tử	Ý nghĩa	Ví dụ	Kết quả
+	Cộng số học	5+2	7
	Cộng ngày	#28/08/01# + 4	#01/09/01#
-	Trừ số học	5-2	3
	Ngày trừ số	#02/09/01# - 3	#30/08/01#
*	Ngày trừ ngày	#29/9/01#-#24/3/84#	16.597
	Phép nhân	5*2	10
/	Phép chia	5/2	2.5
\	Chia nguyên	5\2	2
^	Luỹ thừa	5^2	25
<i>Mod</i>	Lấy số dư của phép chia	5 Mod 2	1

Toán tử luận lý

Toán tử	Ý nghĩa	Ví dụ	Kết Quả
<i>Not</i>	Luật phủ định	<i>Not</i> True	False
		<i>Not</i> False	True
<i>And</i>	Luật và	True <i>And</i> True	True
		True <i>And</i> False	False
		False <i>And</i> True	False
		False <i>And</i> False	False
<i>Or</i>	Luật hay	True <i>Or</i> True	True
		True <i>Or</i> False	True
		False <i>Or</i> True	True
		False <i>Or</i> False	False

Toán tử so sánh

Toán tử	Ý nghĩa	Ví dụ	Kết quả
<	Nhỏ hơn	2 < 5	True
<=	Nhỏ hơn hay bằng	2 <= 5	True
>	Lớn hơn	2 > 5	False
>=	Lớn hơn hay bằng	2 >= 5	False
=	Bằng nhau	2 = 5	False
<>	Khác nhau	2 <> 5	True

Các toán tử khác

Toán tử	Ý nghĩa	Ví dụ	Kết quả
<i>Between ... And ...</i>	Giữa hai giá trị. Dùng trong query	2 <i>Between</i> 1 <i>And</i> 5	True
<i>Like</i>	Giống như	“Hung” <i>Like</i> “Hu*”	True
<i>&</i>	Nối chuỗi	“Nguyễn Văn” & “Hùng”	“Nguyễn Văn Hùng”

Với toán tử like ta có thể dùng các ký tự đại diện sau:

Wildcard characters		
Ký tự	Ý nghĩa	Ví dụ
?	Đại diện cho một ký tự bất kỳ	A? -> AN, AM, AC

*	Đại diện cho một chuỗi ký tự bất kỳ.	A* -> AI, ANH,
#	Đại diện cho một ký số	A## ->A13, A24, A35
[]	Đại diện cho các ký tự nằm trong	A[IN] -> AI, AN
-	Đại diện cho các ký tự nằm trong khoảng	A[M-O] -> AM, AN, AO
!	Đại diện cho ký tự không phải ký tự nằm sau !	A[!C] -> AA, AB, AD, AE, ...

Hàm:

Hàm có dạng *tenHam(danhSachDoiSo)*. Hàm luôn luôn đại diện cho một trị gọi là trị trả về.

IIf(điều kiện, trị 1, trị 2)

Kiểm tra điều kiện, nếu điều kiện đúng trả trị 1 ngược lại trả trị 2

Ví dụ: IIf(namNu = 1, “Nam”, ”Nu”)

Date()

Trả về ngày tháng năm của hệ thống.

Now(biểu thức ngày)

Trả về giờ, phút, giây, ngày tháng năm của hệ thống.

Time(biểu thức ngày)

Trả về giờ phút giây của hệ thống.

Day(biểu thức ngày)

Trả về một số từ 1 đến 31 là ngày của Date.

Month(biểu thức ngày)

Trả về một số từ 1 đến 12 là tháng của Date

Year(biểu thức ngày)

Trả về năm của ngày

Len(biểu thức chuỗi)

Trả về chiều dài của chuỗi.

Chr(mã Ascii)

Trả về ký tự có mã ASCII tương ứng.

InStr(Start, s1, s2)

Trả về vị trí chuỗi s2 nằm trong s1

LCASE(s), UCASE(s)

Đổi chuỗi s thành chuỗi gồm các ký tự thường (hoa)

Left(s, n), Right(s, n)

Trả về chuỗi gồm n ký tự bên trái (phải) của chuỗi s

Mid(s, i, n)

Trả về chuỗi con của chuỗi s, gồm n ký tự kể từ ký tự thứ i

Nz(v1, v2)

Nếu v1 = Null thì Trả về v2, ngược lại trả về v1

Các hàm tính toán trên nhóm:

SUM (thuộc tính)

Tính tổng giá trị của thuộc tính của các bộ trong bảng

MAX(thuộc tính)

tính giá trị lớn nhất của thuộc tính của các bộ trong bảng

MIN(thuộc tính)

tính giá trị nhỏ nhất của thuộc tính của các bộ trong bảng

AVG(thuộc tính>)

tính giá trị trung bình của thuộc tính của các bộ trong bảng

COUNT(thuộc tính)

chỉ đếm những bộ mà giá trị của thuộc tính là khác NULL

Biểu thức

Biểu thức là tổ hợp các toán tử, literal value, hằng, tên hàm, tên thuộc tính.

Biểu thức được lượng giá thành một giá trị.

2 CÂU LỆNH SQL

SQL là ngôn ngữ truy vấn dựa trên đại số quan hệ. Câu lệnh của SQL dùng để rút trích dữ liệu của một một hay nhiều quan hệ. Kết quả của một câu lệnh SQL (truy vấn) là một quan hệ. Để đơn giản trong cách trình bày, ta xem quan hệ mà câu truy vấn sử dụng để tạo ra quan hệ khác gọi là quan hệ nguồn, quan hệ kết quả của truy vấn là quan hệ đích.

i Truy vấn định nghĩa dữ liệu (data definition query)

Tạo lược đồ quan hệ rSV:

```
+ Create Table rSV (MASV Text (10) CONSTRAINT khoaChinh PRIMARY KEY, HOTEN Text (30), NU YesNo, NGAYSINH Date, MALOP Text (10), TINH Text (50), HOCBONG Double )
```

Tạo lược đồ quan hệ rKQ:

```
+ Create Table rKQ (MASV text (10), MAMH Text (10), DIEMTHI Double, CONSTRAINT khoaChinh PRIMARY KEY (MASV, MAMH) )
```

Chú ý: Hai cách tạo trên cho ta thấy hai cách tạo khóa: khóa chỉ gồm một thuộc tính và khóa có nhiều thuộc tính

Thêm cột DTHOAI có kiểu dữ liệu text 20 ký tự vào lược đồ quan hệ rSV:

```
+ Alter Table rSV Add Column DTHOAI Text (20)
```

Sửa kiểu dữ liệu của một cột:

```
+ Alter Table rSV Alter Column DTHOAI double
```

Xóa cột của một lược đồ đã có:

```
+ Alter Table rSV Drop Column DTHOAI
```

Xóa ràng buộc khóa chính:

```
+ Alter Table rSV Drop Constraint khoaChinh
```

Thêm ràng buộc khóa chính:

```
+ Alter Table rSV Add Constraint khoaChinh Primary Key (MASV)
```

Thêm ràng buộc miền giá trị:

```
+ ALTER TABLE doc_exd WITH NOCHECK ADD CONSTRAINT exd_check CHECK (column_a > 1)
```

ii Truy vấn chọn (select query)

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

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

```
Select [Distinct|Top n[%]] field1 [As alias1][,field2 [As alias2][,...]]
From table1 [Inner Join table2 On table1.field1 θ table2.field2] ...
[Where dieuKien]
[Order By field1[Asc|Desc][,field2[Asc|Desc]][,...]];
```

Distinct: loại bỏ các bộ trùng trong quan hệ đích

Top n[%]: Chọn n hay n% mẫu tin đầu tiên.

table: Tên table hay query chứa dữ liệu.

field: Tên field hay một biểu thức.

Alias: Trường hợp field là một biểu thức thì <Alias> là một tên mới của biểu thức.

Inner Join: mỗi mẫu tin của table1 nối với bất kỳ mẫu tin nào của table2 có dữ liệu của field1 thỏa mãn điều kiện so sánh với dữ liệu của field2 tạo thành mẫu tin của query.

điều kiện: Biểu thức mà dữ liệu mẫu tin phải thỏa mãn

Chú ý:

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

Ví dụ: Lập danh sách sinh viên gồm MASV,HOTEN,HOCBONG

```
SELECT MASV ,HOTEN ,HOCBONG
FROM Sv ;
```

Ví dụ: Lập danh sách sinh viên nữ , có học bổng thuộc có mã lớp là “CĐTH2B”.

```
SELECT MASV ,HOTEN ,HOCBONG
FROM Sv
```

WHERE NU And MALOP='CĐTH2B' And HOCBONG>0;

Ví dụ: Lập danh sách sinh viên gồm MASV,HOTEN,HOCBONG trong đó học bổng được sắp giảm dần

```
SELECT MASV ,HOTEN ,HOCBONG
FROM Sv
ORDER BY HOCBONG DESC ;
```

Ví dụ : Lập danh sách sinh viên gồm MASV,HOTEN,TENLOP

```
SELECT MASV ,HOTEN ,TENLOP
FROM Sv Inner Join Lop On Sv.MALOP=Kh.LOP ;
```

Ví dụ: Lập danh sách sinh viên có MASV,HOTEN,NU,NGAYSINH,DIEMTHI với điểm thi môn ‘CSDL’ >= 8

```
SELECT Kq.MASV ,HOTEN ,NU ,NGAYSINH ,DIEMTHI
FROM Kq Inner Sv On Kq.MASV = Sv.MaSV
WHERE MAMH='CSDL' AND DIEMTHI>=8 ;
```

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

```
SELECT *
FROM Sv
WHERE HOTEN Like "NGUYEN*";
```

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

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

Để truy vấn cộng nhóm ta sử dụng câu lệnh SQL sau:

```
Select [Distinct|Top n[%]] field1 [As alias1][,field2 [As alias2][,...]]
From table1 [Inner Join table2 On table1.field1 θ table2.field2] ...
[Where dieuKienLocMauTinNguon]
[Group By fieldGroupBy[,fieldGroupBy[,...]]]
[Having dieuKienLocMauTinTongHop]
[Order By field1[Asc|Desc][,field2[Asc|Desc]][,...]];
```

điềuKiệnLocMẫuTinNguồn: điều kiện mà các mẫu tin nguồn phải thỏa mãn (phép chọn)
fieldGroupBy: tên field mà các mẫu tin có dữ liệu giống nhau trên ấy được xếp vào cùng nhóm.
điềuKiệnLocMẫuTinTổngHợp: điều kiện mà các mẫu tin tổng hợp phải thỏa mãn (phép chọn)

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

```
SELECT Kq.MASV, HOTEN, NU, NGAYSINH, COUNT (MAMH) AS SLMH
FROM Sv Inner Join Kq On Sv.MASV = Kq.MASV
GROUP BY Kq.MASV, HOTEN, NU, NGAYSINH
HAVING COUNT (MAMH) >=3;
```

Ví dụ: Đếm số lượng sinh viên nữ của mỗi khoa

```
SELECT Kh.MAKHOA, TENKHOA, COUNT (Kh.MAKHOA) AS SOLUONG
FROM (Sv Inner Join Lop On Sv.MALOP = LOP.MALOP)
INNER JOIN Kh On Lop.MAKHOA=Kh.MAKHOA
WHERE NU
GROUP BY Kh.MAKHOA, TENKHOA;
```

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

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

- o **biethuc toanTuSoSanh [ANY | ALL | SOME] (cauLenhSQL)**

ANY, SOME là bất kỳ, ALL là tất cả

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

- o **biethuc [NOT] IN (cauLenhSQL)**

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

- o **[NOT] EXISTS (cauLenhSQL).**

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

Ví dụ: Lập danh sách sinh viên có học bổng cao nhất

```
SELECT *
FROM Sv
WHERE HOCBONG >= ALL (SELECT HOCBONG FROM SV);
```

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

```
SELECT SV.MASV, HOTEN, NU, NGAYSINH, DIEMTHI
FROM sv Inner Join kq On Sv.MASV = Kq.MASV
WHERE MAMH='CSDL' AND DIEMTHI >= ALL
(SELECT DIEMTHI FROM KQ WHERE MAMH='CSDL');
```

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

v Truy vấn cập nhật dữ liệu (action query, data modification query)

Cú pháp: Update table Set field1 = biểuThức1, field2 = biểuThức2 Where điềuKiện

Cú pháp: Delete From table Where điều kiện

vi Truy vấn hợp (union query)

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

```
Select .....Union Select .....
```

Ví dụ: Lệnh sau đây gộp đôi danh sách sinh viên

```
SELECT MASV ,HOTEN ,NGAYSINH UNION SELECT MASV ,HOTEN ,NGAYSINH
```

III BÀI TẬP

1/ Cho lược đồ CSDL quản lý sinh viên. Hãy thực hiện các câu truy vấn sau

- Lập danh sách những sinh viên nam của tỉnh “LONG AN” học khoa “CNTT”, danh sách cần tất cả các thuộc tính của quan hệ Sv.
- Lập danh sách những sinh viên có điểm thi < 5 (thi lại), danh sách cần MASV,HOTEN,TENMH, DIEMTHI và được sắp tăng dần theo cột MASV.
- Lập danh sách các sinh viên có điểm thi trung bình các môn < 5, danh sách cần MASV,HOTEN, DIEMTRUNGBINH và được sắp tăng dần theo cột MASV.
- Tổng số tiền học bổng của mỗi khoa
- Những sinh viên nào đăng ký học nhiều hơn 3 môn học, danh sách cần MASV,HOTEN,SOLAN_DANGKY
- Lập danh sách sinh viên có điểm trung bình cao nhất, danh sách cần MASV, HOTEN, NGAYSINH, DIEMTRUNGBINH

2/ Cho lược đồ CSDL dùng để quản lý lao động bao gồm các lược đồ quan hệ sau:

Nhanvien (MANV ,HOTEN ,NGAYSINH , PHAI , DIACHI , MAPB)

Tên từ: Mỗi nhân viên có một mã số nhân viên (MANV) duy nhất. Một mã số nhân viên xác định các thông tin như họ tên (HOTEN), ngày sinh (NGAYSINH), phái (PHAI), địa chỉ (DIACHI) và phòng ban (MAPB) nơi quản lý nhân viên.

Phongban (MAPB , TENPB)

Tên từ: Mỗi phòng ban có một mã phòng ban (MAPB) duy nhất, mã phòng ban xác định tên phòng ban (TENPB)

Cong (MACT ,MANV , SLNGAYCONG)

Tên từ: Lược đồ quan hệ Cong ghi nhận số lượng ngày công (SLNGAYCONG) của một nhân viên (MANV) tham gia vào công trình (MACT).

Congtrinh (MACT , TENCT , DIADIEM , NGAYCAPGP , NGAYKC , NGAYHT)

Tên từ: Mỗi công trình có một mã số công trình (MACT) duy nhất. Mã số công trình xác định các thông tin như tên gọi công trình (TENCT), địa điểm (DIADIEM), ngày công trình được cấp giấy phép xây dựng (NGAYCAPGP), ngày khởi công (NGAYKC), ngày hoàn thành (NGAYHT).

Hãy thực hiện các câu hỏi sau bằng SQL

- Danh sách những nhân viên có tham gia vào công trình có mã công trình (MACT) là X. Yêu cầu các thông tin: MANV,HOTEN, SLNGAYCONG, trong đó MANV được sắp tăng dần.
- Đếm số lượng ngày công của mỗi công trình. Yêu cầu các thông tin: MACT, TENCT, TONGNGAYCONG (TONGNGAYCONG là thuộc tính tự đặt)

- c) Danh sách những nhân viên có sinh nhật trong tháng 8. yêu cầu các thông tin: MANV, TENNV, NGAYSINH, DIACHI, TENPB, sắp xếp quan hệ kết quả theo thứ tự tuổi giảm dần.
- d) Đếm số lượng nhân viên của mỗi phòng ban. Yêu cầu các thông tin: MAPB, TENPB, SOLUONG. (SOLUONG là thuộc tính tự đặt.)

3/ Cho các quan hệ sau:

Monhoc (MSMH , TENMH , SOTINCHI , TINHCHAT)

MSMH mã số môn học,

TENMH tên môn học

SOTINCHI số lượng tín chỉ,

TÍNH CHẤT bằng 1 nếu đó là môn học bắt buộc, bằng 0 nếu đó là môn học không bắt buộc

Sinhvien (MSSV , HOTEN , NGAYSINH , LOP)

MSSV mã số sinh viên,

HOTEN họ tên sinh viên

NGAYSINH ngày sinh,

LOP(C,4,0) lớp

Diem (MSSV , MSMH , DIEMTHI)

DIEMTHI điểm thi

Hãy dùng lệnh SQL để thực hiện các câu lệnh sau:

- a) Hãy cho biết những môn học bắt buộc có SOTINCHI cao nhất.
- b) Hãy liệt kê danh sách gồm MSSV, HOTEN, LOP, DIEMTHI của những sinh viên thi môn học CSDL, theo thứ tự LOP, DIEMTHI
- c) Hãy cho biết các sinh viên có điểm thi cao nhất về môn học có mã là CSDL
- d) Hãy cho biết phiếu điểm của sinh viên có mã số là 9900277
- e) Hãy liệt kê danh sách gồm MSSV, HOTEN., LOP, ĐIỂM TRUNG BÌNH của những sinh viên có điểm trung bình các môn dưới 5, theo thứ tự LOP, HOTEN.
- f) Hãy liệt kê danh sách điểm trung bình của sinh viên theo thứ tự , lớp, tên.
- g) Hãy cho biết điểm của sinh viên theo từng môn.

4/ Dựa vào lược đồ cơ sở dữ liệu

Docgia (MADG, HOTEN, NGAYSINH, DIACHI, NGHENGHIEP)

Phieumuon (SOPM, NGAYMUON, MADG)

Chitietmuon (SOPM, MAD AUSACH, NGAYTRA)

Dausach (MAD AUSACH, BAN, TAP, MASH)

Sach (MASH, TENSACH, TACGIA, NHAXB, NAMXB)

Hãy thực hiện các câu hỏi sau đây bằng SQL

- a) Danh sách các đọc giả đã đăng ký mượn sách trong ngày d. Yêu cầu các thông tin: MADG, HOTEN, DIACHI.
- b) Các quyển sách của phiếu mượn có SOPM là x. Yêu cầu các thông tin MASH, TENSACH, TACGIA, NGAYMUON, NGAYTRA.
- c) Tổng số lượt mà mỗi đọc giả đến mượn sách trong năm 2001. Yêu cầu thông tin MADG, HOTEN, SOLANMUON (SOLANMUON là thuộc tính tự đặt)
- d) Danh sách các đọc giả cao tuổi nhất đã mượn sách trong ngày d. Yêu cầu các thông tin MADG, HOTEN, NGAYSINH, DIACHI, NGHENGHIEP.

5/ Dựa vào lược đồ cơ sở dữ liệu

Khach (MAKH, HOTEN, DIACHI, DIENTHOAI)
 Hoadon (SOHD, NGAYLAPHD, NGAYBAN, MAKH)
 DongHoaDon (SOHD, MAHANG, SLBAN)
 Hang (MAHANG, TENHANG, DONGIA, DVT, MANHOM)
 Nhom (MANHOM, TENNHOM)

Hãy thực hiện các câu hỏi sau bằng SQL

- Danh sách các khách hàng đã mua hàng trong ngày d. Yêu cầu các thông tin MAKH, HOTEN, DIACHI, DIENTHOAI.
- Danh sách các mặt hàng trong số hóa đơn (SOHD) là x. Yêu cầu các thông tin MAHANG, TENHANG, SLBAN, ĐONGIA, THANHTIEN (THANHTIEN= SLBAN*ĐONGIA; THANHTIEN là thuộc tính tự đặt). Yêu cầu sắp xếp tăng dần theo cột TENHANG
- Danh sách các mặt hàng thuộc mã nhóm hàng là A có đơn giá cao nhất. Yêu cầu các thông tin : MAHANG, TENHANG, ĐONGIA
- Đếm số lượng mặt hàng của mỗi nhóm hàng. Yêu cầu các thông tin : MANHOM, TENNHOM, SOLUONG. (trong đó SOLUONG là thuộc tính tự đặt) (0,75đ)
- Danh sách các khách hàng đã mua các mặt hàng có mã nhóm hàng là A trong ngày d. Yêu cầu các thông tin MAKH, HOTEN, DIACHI, DIENTHOAI, TENHANG.
- Thống kê việc mua hàng trong năm 2002 của khách hàng có mã khách hàng là Kh01 (theo từng hóa đơn). Yêu cầu các thông tin MAKH, HOTEN, SOHD, TRIGIAHĐ trong đó TRIGIAHĐ là tổng số tiền trong một hóa đơn (TRIGIAHĐ là thuộc tính tự đặt)

6/ Dựa vào lược đồ cơ sở dữ liệu

Giaovien (MAGV, HOTEN, DTGV, MAKHOA)
 Khoa (MAKHOA, TENKHOA, DTKHOA)
 Lop (MALOP, TENLOP, SISO, MAKHOA)
 Monhoc (MAMH, TENMH)
 Phonghoc (SOPHONG, CHUCNANG)
 Lichbaogiang (MALICH, NGAYDAY, MAGV)
 Dongbaogiang (MALICH, TUTIET, DENTIET, BAIDAY, GHICHU, LYTHUYET, MAMH, MALOP, SOPHONG)

Hãy thực hiện các câu hỏi sau bằng SQL

- Xem lịch báo giảng tuần từ ngày 16/09/2002 đến ngày 23/09/2002 của giáo viên có MAGV (mã giáo viên) là TH3A040. Yêu cầu: MAGV, HOTEN, TENLOP, TENMH, SOPHONG, NGAYDAY, TUTIET, DENTIET, BAIDAY, GHICHU
- Xem lịch báo giảng ngày 23/09/2002 của các giáo viên có mã khoa là CNTT. Yêu cầu: MAGV, HOTEN, TENLOP, TENMH, PHONG, NGAYDAY, TUTIET, DENTIET, BAIDAY, GHICHU)
- Cho biết số lượng giáo viên (SOLUONGGV) của mỗi khoa, kết quả cần sắp xếp tăng dần theo cột tên khoa. yêu cầu: TENKHOA, SOLUONGGV (SOLUONGGV là thuộc tính tự đặt)

----oOo----

Chương 3 .

RÀNG BUỘC TOÀN VỆY QUAN HỆ (**entegrity constraint**)

I RÀNG BUỘC TOÀN VỆY - CÁC YẾU TỐ CỦA RÀNG BUỘC TOÀN VỆY

1 Ràng Buộc Toàn Vệy

Trong mỗi CSDL luôn tồn tại nhiều mối liên hệ giữa các thuộc tính, giữa các bộ. Sự liên hệ này có thể xảy ra trong một lược đồ quan hệ hoặc trong các lược đồ quan hệ của một cơ sở dữ liệu. Các mối liên hệ này là những điều kiện bất biến mà tất cả các bộ của những quan hệ có liên quan trong CSDL đều phải thỏa mãn ở mọi thời điểm. Những điều kiện bất biến đó được gọi là ràng buộc toàn vệy. Trong thực tế ràng buộc toàn vệy là các quy tắc quản lý được áp đặt trên các đối tượng của thế giới thực.

Nhiệm vụ của người phân tích thiết kế là phải phát hiện càng đầy đủ và chính xác các ràng buộc toàn vệy càng tốt và mô tả chúng một cách chính xác trong hồ sơ phân tích thiết kế - đó là một việc làm rất quan trọng và rất cần thiết.

Trong một cơ sở dữ liệu, ràng buộc toàn vệy được xem như là một công cụ để diễn đạt ngữ nghĩa của CSDL. Một CSDL được thiết kế công kênh nhưng nó thể hiện được đầy đủ ngữ nghĩa của thực tế vẫn có giá trị cao hơn rất nhiều so với một cách thiết kế gọn nhẹ nhưng nghèo nàn về ngữ nghĩa vì thiếu các ràng buộc toàn vệy của cơ sở dữ liệu.

Công việc kiểm tra ràng buộc toàn vệy thường được tiến hành vào thời điểm cập nhật dữ liệu (thêm, sửa, xóa). Những ràng buộc toàn vệy phát sinh cần phải được ghi nhận và xử lý một cách tường minh (thường là bởi một hàm chuẩn hoặc một đoạn chương trình).

2 Các Yếu Tố Của Ràng Buộc Toàn Vệy

Mỗi ràng buộc toàn vệy có 3 yếu tố: điều kiện, bối cảnh và tầm ảnh hưởng.

i Điều kiện

Điều kiện của một ràng buộc toàn vệy R có thể được biểu diễn bằng ngôn ngữ tự nhiên, thuật giải, ngôn ngữ đại số tập hợp, đại số quan hệ,... ngoài ra điều kiện của ràng buộc toàn vệy cũng có thể được biểu diễn bằng phụ thuộc hàm. Chẳng hạn, với lược đồ quan hệ SV thì có một ràng buộc toàn vệy như sau:

Với r là một quan hệ của Sv ta có ràng buộc toàn vệy sau

$$\forall t_1, t_2 \in r \\ t_1.MASV \neq t_2.MASV$$

cuối \forall

ii Bối cảnh

Bối cảnh của một ràng buộc toàn vệy là những quan hệ mà ràng buộc đó có hiệu lực hay nói một cách khác, đó là những quan hệ cần phải được kiểm tra ràng buộc toàn vệy. Bối cảnh của một ràng buộc toàn vệy có thể là một hoặc nhiều quan hệ. Chẳng hạn với ràng buộc toàn vệy trên thì bối cảnh là một quan hệ Sv

iii **Tâm ảnh hưởng**

Trong quá trình phân tích thiết kế một CSDL, người phân tích cần lập bảng tâm ảnh hưởng cho một ràng buộc toàn vẹn nhằm xác định thời điểm cần phải tiến hành kiểm tra các ràng buộc toàn vẹn đó. Các thời điểm cần phải kiểm tra RBTV chính là những thời điểm cập nhật dữ liệu (thêm/sửa/xóa)

Một bảng tâm ảnh hưởng của một RBTV có dạng sau:

(Tên RBTV)	Thêm(T)	Sửa(S)	Xóa(X)
r_1	+	-	-
r_2			
...
...
r_n			

Bảng này chứa toàn các ký hiệu + hoặc -

Chẳng hạn + tại ô tương ứng với dòng r_1 , cột thêm thì có nghĩa là khi thêm một bộ vào quan hệ r_1 thì cần phải kiểm tra RBTV

Dấu - Tại ô tương ứng với dòng r_1 , cột sửa thì có nghĩa là khi sửa một bộ trên quan hệ r_1 thì không cần phải kiểm tra RBTV này,...

II **PHÂN LOẠI RÀNG BUỘC TOÀN VẸN**

Trong quá trình phân tích thiết kế cơ sở dữ liệu, người phân tích phải phát hiện tất cả các ràng buộc toàn vẹn tiềm ẩn trong CSDL đó. Việc phân loại các ràng buộc toàn vẹn là rất có ích, nó nhằm giúp cho người phân tích có được một định hướng, tránh bỏ sót những ràng buộc toàn vẹn. Các ràng buộc toàn vẹn có thể được chia làm hai loại chính như sau:

- + Ràng buộc toàn vẹn trên phạm vi là một quan hệ bao gồm :Ràng buộc toàn vẹn miễn giá trị, ràng buộc toàn vẹn liên thuộc tính, ràng buộc toàn vẹn liên bộ.
- + Ràng buộc toàn vẹn trên phạm vi nhiều quan hệ bao gồm :Ràng buộc toàn vẹn phụ thuộc tồn tại, ràng buộc toàn vẹn liên bộ - liên quan hệ, ràng buộc toàn vẹn liên thuộc tính - liên quan hệ.

Để minh họa cho phần lý thuyết của chương này, ta nêu ví dụ sau đây

Ví dụ

Cho một CSDL C dùng để quản lý việc đặt hàng và giao hàng của một công ty. Lược đồ CSDL C gồm các lược đồ quan hệ như sau:

Q₁: Khách (MAKH, TENKH, DCKH, DT)

Tên từ: Mỗi khách hàng có một mã khách hàng (MAKH) duy nhất, mỗi MAKH xác định một tên khách hàng (TENKH), một địa chỉ (DCKH), một số điện thoại (DT).

Q₂: Hàng (MAHANG, TENHANG, QUYCACH, DVTINH)

Tên từ: Mỗi mặt hàng có một mã hàng (MAHANG) duy nhất, mỗi MAHANG xác định một tên hàng (TENHANG), quy cách hàng (QUYCACH), đơn vị tính (DVTINH).

Q₃: Dathang (SODH, MAHANG, SLDAT, NGAYDH, MAKH)

Tên từ: Mỗi lần đặt hàng có số đặt hàng (SODH) xác định một ngày đặt hàng (NGAYDH) và mã khách hàng tương ứng (MAKH). Biết mã số đặt hàng và mã mặt hàng thì biết được số lượng đặt hàng(SLDAT). Mỗi khách hàng trong một ngày có thể có nhiều lần đặt hàng

Q4: Hoadon (SOHD, NGAYLAP, SODH, TRIGIAHD, NGAYXUAT)

Tân từ: Mỗi hóa đơn có một mã số duy nhất là SOHD, mỗi hóa đơn bán hàng có thể gồm nhiều mặt hàng. Mỗi hóa đơn xác định ngày lập hóa đơn (NGAYLAP), ứng với số đặt hàng nào (SODH). Giả sử rằng hóa đơn bán hàng theo yêu cầu của chỉ một đơn đặt hàng có mã số là SODH và ngược lại, mỗi đơn đặt hàng chỉ được giải quyết chỉ trong một hóa đơn. **Do điều kiện khách quan có thể công ty không giao đầy đủ các mặt hàng cũng như số lượng từng mặt hàng như yêu cầu trong đơn đặt hàng nhưng không bao giờ giao vượt ngoài yêu cầu.** Mỗi hóa đơn xác định một trị giá của các mặt hàng trong hóa đơn (TRIGIAHD) và một ngày xuất kho giao hàng cho khách (NGAYXUAT)

Q5: Chitiethd (SOHD, MAHANG, GIABAN, SLBAN)

Tân từ: Mỗi SOHD, MAHANG xác định giá bán (GIABAN) và số lượng bán (SLBAN) của một mặt hàng trong một hóa đơn.

Q6: Phieuthu (SOPT, NGAYTHU, MAKH, SOTIEN)

Tân từ: Mỗi phiếu thu có một số phiếu thu (SOPT) duy nhất, mỗi SOPT xác định một ngày thu (NGAYTHU) của một khách hàng có mã khách hàng là MAKH và số tiền thu là SOTIEN. Mỗi khách hàng trong một ngày có thể có nhiều số phiếu thu.

1 Ràng buộc toàn vẹn liên bộ

Ràng buộc toàn vẹn liên bộ là sự ràng buộc toàn vẹn giữa các bộ trong cùng một quan hệ .

Ràng buộc toàn vẹn liên bộ hay còn gọi là ràng buộc toàn vẹn về khóa. Đây là loại ràng buộc toàn vẹn rất phổ biến, nó có mặt trong mọi lược đồ quan hệ của CSDL và thường được các hệ quản trị CSDL tự động kiểm tra.

Ví dụ: Với r là một quan hệ của Khách ta có ràng buộc toàn vẹn sau

$$R_1: \forall t_1, t_2 \in r$$

$$t_1.MAKH \neq t_2.MAKH$$

Cuối \forall

R ₁	Thêm	Sửa	Xóa
r	+	+	-

2 Ràng buộc toàn vẹn về phụ thuộc tồn tại:

Ràng buộc toàn vẹn về phụ thuộc tồn tại còn được gọi là ràng buộc toàn vẹn về khóa ngoại. Cũng giống như ràng buộc toàn vẹn về khóa chính, ràng buộc toàn vẹn về phụ thuộc tồn tại rất phổ biến trong CSDL

Ví dụ: Với r, s lần lượt là một quan hệ của Dathang, Khách ta có ràng buộc toàn vẹn sau

$$R_2: r[MAKH] \subseteq s[MAKH]$$

R ₂	Thêm	Sửa	Xóa
r	+	+	-
s	-	+	+

3 Ràng buộc toàn vẹn về miền giá trị

Ràng buộc toàn vẹn có liên quan đến miền giá trị của các thuộc tính trong một quan hệ. Ràng buộc này thường gặp. Một số hệ quản trị CSDL đã tự động kiểm tra một số ràng buộc loại này.

Ví dụ: Với r là một quan hệ của Hoadon ta có ràng buộc toàn vẹn sau

$R_3: \forall t \in r$

$t.TRIGIAHD > 0$

Cuối \forall

R_3	Thêm	Sửa	Xóa
r	+	+	-

4 Ràng buộc toàn vẹn liên thuộc tính

Ràng buộc toàn vẹn liên thuộc tính là mối liên hệ giữa các thuộc tính trong một lược đồ quan hệ.

Ví dụ: Với r là một quan hệ của Hoadon ta có ràng buộc toàn vẹn sau

$R_4: \forall t \in r$

$t.NGAYLAP \leq t.NGAYXUAT$

Cuối \forall

R_4	Thêm	Sửa	Xóa
r	+	+	-

5 Ràng buộc toàn vẹn liên thuộc tính liên quan hệ

Ràng buộc loại này là mối liên hệ giữa các thuộc tính trong nhiều lược đồ quan hệ.

Ví dụ: Với r, s lần lượt là quan hệ của Dathang, Hoadon ta có ràng buộc toàn vẹn sau

$R_5: \forall t_1 \in r, t_2 \in s$

Nếu $t_1.SODH = t_2.SODH$ thì

$t_1.NGAYDH \leq t_2.NGAYXUAT$

Cuối \forall

R_5	Thêm	Sửa	Xóa
r	+	+	+
s	+	+	+

6 Ràng buộc toàn vẹn về thuộc tính tổng hợp

Ràng buộc toàn vẹn về thuộc tính tổng hợp được xác định trong trường hợp mỗi thuộc tính A của một lược đồ quan hệ Q được tính toán giá trị từ các thuộc tính của các lược đồ quan hệ khác.

III BÀI TẬP

1/ Hãy tìm các ràng buộc toàn vẹn có trong CSDL cho các bài tập được liệt kê trong chương 3.

2/ QUẢN LÝ THI TỐT NGHIỆP PTCS

Một phòng giáo dục huyện muốn lập một hệ thống thông tin để quản lý việc làm thi tốt nghiệp phổ thông cơ sở. Công việc làm thi được tổ chức như sau:

Lãnh đạo phòng giáo dục thành lập nhiều hội đồng thi (mỗi hội đồng thi gồm một trường hoặc một số trường gần nhau). Mỗi hội đồng thi có một mã số duy nhất (MAHĐT), một mã số hội đồng thi xác định tên hội đồng thi (TENHĐT), họ tên chủ tịch hội đồng (TENCT), địa chỉ (ĐCHĐT), điện thoại (ĐTHĐT).

Mỗi hội đồng thi được bố trí cho một số phòng thi, mỗi phòng thi có một số hiệu phòng (SOPT) duy nhất, một phòng thi xác định địa chỉ phòng thi (ĐCPT). Số hiệu phòng thi được đánh số khác nhau ở tất cả các hội đồng thi.

Giáo viên của các trường trực thuộc phòng được điều động đến các hội đồng để coi thi, mỗi trường có thể có hoặc không có thí sinh dự thi, mỗi trường có một mã trường duy nhất (MATR), mỗi mã trường xác định một tên trường(TENTR), địa chỉ (ĐCTR), loại hình đào tạo (LHĐT) (Công lập, chuyên, bán công, dân lập, nội trú,...). Giáo viên của một trường có thể làm việc tại nhiều hội đồng thi. Một giáo viên có một mã giáo viên(MAGV), một mã giáo viên xác định tên giáo viên (TENGV), chuyên môn giảng dạy (CHUYENMON), chức danh trong hội đồng thi(CHUCDANH)

Các thí sinh dự thi có một số báo danh duy nhất(SOBD), mỗi số báo danh xác định tên thí sinh(TENTS), ngày sinh (NGSINH), giới tính (PHAI), mỗi thí sinh được xếp thi tại một phòng thi nhất định cho tất cả các môn, mỗi thí sinh có thể có chứng chỉ nghề (CCNGHE) hoặc không (thuộc tính CCNGHE kiểu chuỗi, CCNGHE="x" nếu thí sinh có chứng chỉ nghề và CCNGHE bằng rỗng nếu thí sinh không có chứng chỉ nghề).Thí sinh của cùng một trường chỉ dự thi tại một hội đồng thi. Mỗi môn thi có một mã môn thi duy nhất(MAMT), mỗi mã môn thi xác định tên môn thi(TENMT). Giả sử toàn bộ các thí sinh đều thi chung một số môn do sở giáo dục quy định. Mỗi môn thi được tổ chức trong một buổi của một ngày nào đó.

Ứng với mỗi môn thi một thí sinh có một điểm thi duy nhất(ĐIEMTHI)

Dựa vào phân tích ở trên, giả sử ta có lược đồ CSDL sau:

- Q1: HĐ(MAHĐT,TENHĐT, TENCT, ĐCHĐT,ĐTHĐT)
- Q2: PT(SOPT,ĐCPT,MAHĐT)
- Q3: TS(SOBD, TENTS,NGSINH,PHAI,CCNGHE, MATR,SOPT)
- Q4: MT(MAMT,TENMT,BUOI,NGAY)
- Q5: GV(MAGV,TENGV,CHUYENMON,CHUCDANH,MAHĐT,MATR)
- Q6: TR(MATR,TENTR,ĐCTR,LHĐT)
- Q7: KQ(SOBD,MAMT,ĐIEMTHI)

Yêu cầu:

- a) Hãy xác định khóa cho từng lược đồ quan hệ.
- b) Tìm tất cả các ràng buộc toàn vẹn có trong CSDL trên.
- c) Dựa vào lược đồ CSDL đã thành lập, hãy thực hiện các câu hỏi sau đây bằng ngôn ngữ đại số quan hệ.
 1. Danh sách các thí sinh thi tại phòng thi có số hiệu phòng thi (SOPT) là "100". Yêu cầu các thông tin:SOBD,TENTS,NGSINH,TENTR
 2. Kết quả của môn thi có mã môn thi (MAMT) là "T" của tất cả các thí sinh có mã trường(MATR) là "NTMK", kết quả được sắp theo chiều giảm dần của điểm thi(ĐIEMTHI). Yêu cầu các thông tin:SOBD,TENTS, ĐIEMTHI
 3. Kết quả thi của một học sinh có SOBD là MK01. Yêu cầu : TENMT,ĐIEMTHI
 4. Tổng số thí sinh có chứng chỉ nghề(CCNGHE) của mỗi trường, thông tin cần được sắp theo chiều tăng dần của TENTR. Yêu cầu các thông tin: MATR, TENTR, SOLUONGCC

----oOo----

Chương 4 .**PHỤ THUỘC HÀM****(functional dependency)**

Phụ thuộc hàm (functional dependency) là một công cụ dùng để biểu diễn một cách hình thức các ràng buộc toàn vẹn (vấn tắt: ràng buộc). Phương pháp biểu diễn này có rất nhiều ưu điểm, và đây là một công cụ cực kỳ quan trọng, gắn chặt với lý thuyết thiết kế cơ sở dữ liệu.

Phụ thuộc hàm được ứng dụng trong việc giải quyết các bài toán tìm khóa, tìm phủ tối thiểu và chuẩn hóa cơ sở dữ liệu.

I KHÁI NIỆM PHỤ THUỘC HÀM

Cho quan hệ **phanCong** sau:

phanCong (PHICONG, MAYBAY, NGÀYKH, GIOKH)

Cushing	83	9/8	10:15a
Cushing	116	10/8	1:25p
Clark	281	8/8	5:50a
Clark	301	12/8	6:35p
Clark	83	11/8	10:15a
Chin	83	13/8	10:15a
Chin	116	12/8	1:25p
Copely	281	9/8	5:50a
Copely	281	13/8	5:50a
Copely	412	15/8	1:25p

Quan hệ **phanCong** diễn tả phi công nào lái máy bay nào và máy bay khởi hành vào thời gian nào. Không phải sự phối hợp bất kỳ nào giữa phi công, máy bay và ngày giờ khởi hành cũng đều được chấp nhận mà chúng có các điều kiện ràng buộc qui định sau:

- + Mỗi máy bay có một giờ khởi hành duy nhất.
- + Nếu biết phi công, biết ngày giờ khởi hành thì biết được máy bay do phi công ấy lái.
- + Nếu biết máy bay, biết ngày khởi hành thì biết phi công lái chuyến bay ấy.

Các ràng buộc này là các ví dụ về phụ thuộc hàm và được phát biểu lại như sau:

- + MAYBAY xác định GIOKH
- + {PHICONG, NGÀYKH, GIOKH} xác định MABAY
- + {MAYBAY, NGÀYKH} xác định PHICONG

hay

- + GIOKH phụ thuộc hàm vào MAYBAY
- + MABAY phụ thuộc hàm vào {PHICONG, NGÀYKH, GIOKH}
- + PHICONG phụ thuộc hàm vào {MAYBAY, NGÀYKH}

và được ký hiệu như sau:

- + {MAYBAY} → GIOKH
- + {PHICONG, NGÀYKH, GIOKH} → MABAY
- + {MAYBAY, NGÀYKH} → PHICONG

Trong ký hiệu trên ta đã ký hiệu **MAYBAY** thay cho {**MAYBAY**} .

Một cách tổng quát:

1 Định nghĩa phụ thuộc hàm

$Q(A_1, A_2, \dots, A_n)$ là lược đồ quan hệ.

X, Y là hai tập con của $Q^+ = \{A_1, A_2, \dots, A_n\}$.

r là quan hệ trên Q .

t_1, t_2 là hai bộ bất kỳ của r .

$$X \rightarrow Y \Leftrightarrow (t_1.X = t_2.X \Rightarrow t_1.Y = t_2.Y)$$

(Ta nói X xác định Y hay Y phụ thuộc hàm vào X (X functional determines Y , Y functional dependent on X))

Tính chất:

+ phụ thuộc hàm $X \rightarrow \emptyset$ đúng với mọi quan hệ r

+ phụ thuộc hàm $\emptyset \rightarrow Y$ chỉ đúng trên quan hệ r có cùng giá trị trên Y .

Ví dụ: Quan hệ sau thỏa mãn phụ thuộc hàm $\emptyset \rightarrow \text{GIOKH}$

phanCong	(PHICONG,	MAYBAY,	NGAYKH,	GIOKH)
Cushing	83	9/8	10:15a	
Cushing	116	10/8	10:15a	
Clark	281	8/8	10:15a	
Clark	301	12/8	10:15a	
Clark	83	11/8	10:15a	
Chin	83	13/8	10:15a	
Chin	116	12/8	10:15a	
Copely	281	9/8	10:15a	
Copely	281	13/8	10:15a	
Copely	412	15/8	10:15a	

trên thực tế không có quan hệ r nào thỏa tính chất trên nên từ đây về sau nếu không nói rõ thì với một quan hệ r bất kỳ ta luôn xem phụ thuộc hàm $\emptyset \rightarrow Y$ luôn luôn không thỏa trên r .

2 Phụ thuộc hàm hiển nhiên (Trivial Dependencies)

Hệ quả: Nếu $X \supseteq Y$ thì $X \rightarrow Y$.

Chứng minh:

Giả sử $t_1.X = t_2.X$ do $X \supseteq Y$ nên $t_1.Y = t_2.Y$ theo định nghĩa suy ra $X \rightarrow Y$

Trong trường hợp này $X \rightarrow Y$ được gọi là **phụ thuộc hàm hiển nhiên**.

Ví dụ phụ thuộc hàm $X \rightarrow X$ là phụ thuộc hàm hiển nhiên.

Vậy với r là quan hệ bất kỳ, F là tập phụ thuộc hàm thỏa trên r , ta luôn có $F \supseteq \{\text{các phụ thuộc hàm hiển nhiên}\}$

3 Thuật toán Satisfies

Cho quan hệ r và X, Y là hai tập con của Q^+ . Thuật toán SATIFIES sẽ trả về trị true nếu $X \rightarrow Y$ ngược lại là false

SATIFIES

Vào: quan hệ r và hai tập con X, Y

ra: true nếu $X \rightarrow Y$, ngược lại là false

SATIFIES(r, X, Y)

1. Sắp các bộ của quan hệ r theo X để các giá trị giống nhau trên X nhóm lại với nhau
2. Nếu tập các bộ cùng giá trị trên X cho các giá trị trên Y giống nhau thì trả về true ngược lại là False

Ví dụ 1: SATIFIES (phanCong, MAYBAY, GIOKH)

phanCong	(PHICONG, MAYBAY , NGAYKH, GIOKH)
Cushing	83 9/8 10:15a
Clark	83 11/8 10:15a
Chin	83 13/8 10:15a
Cushing	116 10/8 1:25p
Chin	116 12/8 1:25p
Clark	281 8/8 5:50a
Copely	281 9/8 5:50a
Copely	281 13/8 5:50a
Clark	301 12/8 6:35p
Copely	412 15/8 1:25p

cho kết quả là true nghĩa là MAYBAY→GIOKH

Ví dụ 2: SATIFIES (phanCong, GIOKH, MAYBAY)

phanCong	(PHICONG, MAYBAY, NGAYKH, GIOKH)
Clark	281 8/8 5:50a
Copely	281 9/8 5:50a
Copely	281 13/8 5:50a
Cushing	83 9/8 10:15a
Clark	83 11/8 10:15a
Chin	83 13/8 10:15a
Cushing	116 10/8 1:25p
Chin	116 12/8 1:25p
Copely	412 15/8 1:25p
Clark	301 12/8 6:35p

cho kết quả là false nghĩa là không có phụ thuộc hàm GIOKH→MAYBAY

4 Các phụ thuộc hàm có thể có

i Cách tìm tất cả tập con của Q^+

Lược đồ quan hệ Phancong (PHICONG, MAYBAY, NGAYKH, GIOKH) có tập thuộc tính $Phancong^+ = \{PHICONG, MAYBAY, NGAYKH, GIOKH\}$ và tất cả các tập con có thể có của $Phancong^+$ được cho bởi bảng sau:

	PHICONG	MAYBAY	NGAYKH	GIOKH
\emptyset	{PHICONG}	{MAYBAY}	{NGAYKH}	{GIOKH}
		{PHICONG, MAYBAY}	{PHICONG, NGAYKH}	{PHICONG, GIOKH}
			{MAYBAY, NGAYKH}	{MAYBAY, GIOKH}
			{PHICONG, MAYBAY, NGAYKH}	{PHICONG, MAYBAY, GIOKH}
				{NGAYKH, GIOKH}
				{PHICONG, NGAYKH, GIOKH}
				{MAYBAY, NGAYKH, GIOKH}
				{PHICONG, MAYBAY, NGAYKH, GIOKH}

Số tập con có thể có của $Q^+ = \{A_1, A_2, \dots, A_n\}$ là 2^n

ii Cách tìm tất cả các phụ thuộc hàm có thể có của Q

Ứng với mỗi tập con của $Phancong^+$ cho $2^n = 2^4 = 16$ phụ thuộc hàm có thể có. Số phụ thuộc hàm có thể có là $2^4 * 2^4 = 16 * 16 = 256$

{ PHICONG, MAYBAY } → { MAYBAY, NGAYKH, GIOKH }	F ⁻
{ PHICONG, MAYBAY } → { MAYBAY, PHICONG, NGAYKH, GIOKH }	F ⁻
{ NGAYKH } → ∅	F ⁻
{ NGAYKH } → { PHICONG }	F ⁻
{ NGAYKH } → { MAYBAY }	F ⁻
{ NGAYKH } → { PHICONG, MAYBAY }	F ⁻
{ NGAYKH } → { NGAYKH }	PTHHN
{ NGAYKH } → { PHICONG, NGAYKH }	F ⁻
{ NGAYKH } → { MAYBAY, NGAYKH }	F ⁻
{ NGAYKH } → { MAYBAY, PHICONG, NGAYKH }	F ⁻
{ NGAYKH } → { GIOKH }	F ⁻
{ NGAYKH } → { PHICONG, GIOKH }	F ⁻
{ NGAYKH } → { MAYBAY, GIOKH }	F ⁻
{ NGAYKH } → { MAYBAY, PHICONG, GIOKH }	F ⁻
{ NGAYKH } → { NGAYKH, GIOKH }	F ⁻
{ NGAYKH } → { PHICONG, NGAYKH, GIOKH }	F ⁻
{ NGAYKH } → { MAYBAY, NGAYKH, GIOKH }	F ⁻
{ NGAYKH } → { MAYBAY, PHICONG, NGAYKH, GIOKH }	F ⁻
{ PHICONG, NGAYKH } → ∅	PTHHN
{ PHICONG, NGAYKH } → { PHICONG }	PTHHN
{ PHICONG, NGAYKH } → { MAYBAY }	F ⁻
{ PHICONG, NGAYKH } → { PHICONG, MAYBAY }	F ⁻
{ PHICONG, NGAYKH } → { NGAYKH }	PTHHN
{ PHICONG, NGAYKH } → { PHICONG, NGAYKH }	PTHHN
{ PHICONG, NGAYKH } → { MAYBAY, NGAYKH }	F ⁻
{ PHICONG, NGAYKH } → { MAYBAY, PHICONG, NGAYKH }	F ⁻
{ PHICONG, NGAYKH } → { GIOKH }	F ⁻
{ PHICONG, NGAYKH } → { PHICONG, GIOKH }	F ⁻
{ PHICONG, NGAYKH } → { MAYBAY, GIOKH }	F ⁻
{ PHICONG, NGAYKH } → { MAYBAY, PHICONG, GIOKH }	F ⁻
{ PHICONG, NGAYKH } → { NGAYKH, GIOKH }	F ⁻
{ PHICONG, NGAYKH } → { PHICONG, NGAYKH, GIOKH }	F ⁻
{ PHICONG, NGAYKH } → { MAYBAY, NGAYKH, GIOKH }	F ⁻
{ PHICONG, NGAYKH } → { MAYBAY, PHICONG, NGAYKH, GIOKH }	F ⁻
{ MAYBAY, NGAYKH } → ∅	PTHHN
{ MAYBAY, NGAYKH } → { PHICONG }	F
{ MAYBAY, NGAYKH } → { MAYBAY }	PTHHN
{ MAYBAY, NGAYKH } → { PHICONG, MAYBAY }	F ⁺
{ MAYBAY, NGAYKH } → { NGAYKH }	PTHHN
{ MAYBAY, NGAYKH } → { PHICONG, NGAYKH }	F ⁺
{ MAYBAY, NGAYKH } → { MAYBAY, NGAYKH }	PTHHN
{ MAYBAY, NGAYKH } → { MAYBAY, PHICONG, NGAYKH }	F ⁺
{ MAYBAY, NGAYKH } → { GIOKH }	F ⁺
{ MAYBAY, NGAYKH } → { PHICONG, GIOKH }	F ⁺
{ MAYBAY, NGAYKH } → { MAYBAY, GIOKH }	F ⁺
{ MAYBAY, NGAYKH } → { MAYBAY, PHICONG, GIOKH }	F ⁺
{ MAYBAY, NGAYKH } → { NGAYKH, GIOKH }	F ⁺
{ MAYBAY, NGAYKH } → { PHICONG, NGAYKH, GIOKH }	F ⁺
{ MAYBAY, NGAYKH } → { MAYBAY, NGAYKH, GIOKH }	F ⁺
{ MAYBAY, NGAYKH } → { MAYBAY, PHICONG, NGAYKH, GIOKH }	F ⁺
{ PHICONG, MAYBAY, NGAYKH } → ∅	PTHHN
{ PHICONG, MAYBAY, NGAYKH } → { PHICONG }	PTHHN
{ PHICONG, MAYBAY, NGAYKH } → { MAYBAY }	PTHHN
{ PHICONG, MAYBAY, NGAYKH } → { PHICONG, MAYBAY }	PTHHN
{ PHICONG, MAYBAY, NGAYKH } → { NGAYKH }	PTHHN
{ PHICONG, MAYBAY, NGAYKH } → { PHICONG, NGAYKH }	PTHHN
{ PHICONG, MAYBAY, NGAYKH } → { MAYBAY, NGAYKH }	PTHHN
{ PHICONG, MAYBAY, NGAYKH } → { PHICONG, MAYBAY, NGAYKH }	PTHHN
{ PHICONG, MAYBAY, NGAYKH } → { GIOKH }	F ⁺
{ PHICONG, MAYBAY, NGAYKH } → { PHICONG, GIOKH }	F ⁺
{ PHICONG, MAYBAY, NGAYKH } → { MAYBAY, GIOKH }	F ⁺

{ PHICONG, MAYBAY, NGAYKH } → { MAYBAY, PHICONG, GIOKH }	F ⁺
{ PHICONG, MAYBAY, NGAYKH } → { NGAYKH, GIOKH }	F ⁺
{ PHICONG, MAYBAY, NGAYKH } → { PHICONG, NGAYKH, GIOKH }	F ⁺
{ PHICONG, MAYBAY, NGAYKH } → { MAYBAY, NGAYKH, GIOKH }	F ⁺
{ PHICONG, MAYBAY, NGAYKH } → { MAYBAY, PHICONG, NGAYKH, GIOKH }	F ⁺
.....	
{ PHICONG, NGAYKH, GIOKH } → ∅	PTHHN
{ PHICONG, NGAYKH, GIOKH } → { PHICONG }	PTHHN
{ PHICONG, NGAYKH, GIOKH } → { MAYBAY }	F
{ PHICONG, NGAYKH, GIOKH } → { PHICONG, MAYBAY }	F ⁺
{ PHICONG, NGAYKH, GIOKH } → { NGAYKH }	PTHHN
{ PHICONG, NGAYKH, GIOKH } → { PHICONG, NGAYKH }	PTHHN
{ PHICONG, NGAYKH, GIOKH } → { MAYBAY, NGAYKH }	F ⁺
{ PHICONG, NGAYKH, GIOKH } → { MAYBAY, PHICONG, NGAYKH }	F ⁺
{ PHICONG, NGAYKH, GIOKH } → { GIOKH }	PTHHN
{ PHICONG, NGAYKH, GIOKH } → { PHICONG, GIOKH }	PTHHN
{ PHICONG, NGAYKH, GIOKH } → { MAYBAY, GIOKH }	F ⁺
{ PHICONG, NGAYKH, GIOKH } → { MAYBAY, PHICONG, GIOKH }	F ⁺
{ PHICONG, NGAYKH, GIOKH } → { NGAYKH, GIOKH }	PTHHN
{ PHICONG, NGAYKH, GIOKH } → { PHICONG, NGAYKH, GIOKH }	PTHHN
{ PHICONG, NGAYKH, GIOKH } → { MAYBAY, NGAYKH, GIOKH }	F ⁺
{ PHICONG, NGAYKH, GIOKH } → { MAYBAY, PHICONG, NGAYKH, GIOKH }	F ⁺
{ MAYBAY, NGAYKH, GIOKH } → ∅	PTHHN
{ MAYBAY, NGAYKH, GIOKH } → { PHICONG }	F ⁺
{ MAYBAY, NGAYKH, GIOKH } → { MAYBAY }	PTHHN
{ MAYBAY, NGAYKH, GIOKH } → { PHICONG, MAYBAY }	F ⁺
{ MAYBAY, NGAYKH, GIOKH } → { NGAYKH }	PTHHN
{ MAYBAY, NGAYKH, GIOKH } → { PHICONG, NGAYKH }	F ⁺
{ MAYBAY, NGAYKH, GIOKH } → { MAYBAY, NGAYKH }	PTHHN
{ MAYBAY, NGAYKH, GIOKH } → { MAYBAY, PHICONG, NGAYKH }	F ⁺
{ MAYBAY, NGAYKH, GIOKH } → { GIOKH }	PTHHN
{ MAYBAY, NGAYKH, GIOKH } → { PHICONG, GIOKH }	F ⁺
{ MAYBAY, NGAYKH, GIOKH } → { MAYBAY, GIOKH }	PTHHN
{ MAYBAY, NGAYKH, GIOKH } → { MAYBAY, PHICONG, GIOKH }	F ⁺
{ MAYBAY, NGAYKH, GIOKH } → { NGAYKH, GIOKH }	PTHHN
{ MAYBAY, NGAYKH, GIOKH } → { PHICONG, NGAYKH, GIOKH }	F ⁺
{ MAYBAY, NGAYKH, GIOKH } → { MAYBAY, NGAYKH, GIOKH }	PTHHN
{ MAYBAY, NGAYKH, GIOKH } → { MAYBAY, PHICONG, NGAYKH, GIOKH }	F ⁺
.....

Số phụ thuộc hàm có thể có của Q (A₁, A₂, ..., A_n) là 2ⁿ x 2ⁿ = 2²ⁿ

II HỆ LUẬT DẪN ARMSTRONG (Armstrong inference rule)

Người ta thường dùng F để chỉ tập các phụ thuộc hàm của lược đồ quan hệ Q. Ta có thể đánh số các phụ thuộc hàm của F là f₁, f₂, ..., f_m. Quy ước rằng chỉ cần mô tả các phụ thuộc hàm không hiển nhiên trong tập F (các phụ thuộc hàm hiển nhiên được ngầm hiểu là đã có trong F).

1 Phụ thuộc hàm được suy diễn logic từ F

Nói rằng phụ thuộc hàm X → Y được suy diễn logic từ F nếu một quan hệ r thỏa mãn tất cả các phụ thuộc hàm của F thì cũng thỏa phụ thuộc hàm X → Y. Ký hiệu F⁺ = X → Y.

Bao đóng của F ký hiệu F⁺ là tập tất cả các phụ thuộc hàm được suy diễn logic từ F.

Các tính chất của tập F⁺

1. **Tính phản xạ:** Với mọi tập phụ thuộc hàm F⁺ ta luôn luôn có F ⊆ F⁺
2. **Tính đơn điệu:** Nếu F ⊆ G thì F⁺ ⊆ G⁺
3. **Tính lũy đẳng:** Với mọi tập phụ thuộc hàm F ta luôn luôn có (F⁺)⁺ = F⁺.

Gọi G là tập tất cả các phụ thuộc hàm có thể có của r, phần phụ của F ký hiệu F⁻ = G - F⁺

Chứng minh

1. $X \rightarrow Y \in F \Rightarrow r$ thỏa $X \rightarrow Y \Rightarrow X \rightarrow Y \in F^+$
2. Nếu $X \rightarrow Y$ là phụ thuộc hàm thuộc F^+ ta phải chứng minh $X \rightarrow Y$ thuộc G^+
 Giả sử r thỏa tất cả các phụ thuộc hàm của G (1)
 $\Rightarrow r$ thỏa tất cả phụ thuộc hàm của F vì $F \subseteq G$
 $\Rightarrow r$ thỏa phụ thuộc hàm $X \rightarrow Y$ (2) vì $X \rightarrow Y \in F^+$
 (1) và (2) $\Rightarrow X \rightarrow Y \in G^+ \Rightarrow F^+ \subseteq G^+$
3. $F \subseteq F^+$ (tính phản xạ) $\Rightarrow F^+ \subseteq (F^+)^+$ (1)
 Nếu $X \rightarrow Y \in (F^+)^+$ (2) $\Rightarrow X \rightarrow Y \in F^+$ thật vậy: (3)
 Giả sử r thỏa tất cả các phụ thuộc hàm của F (4)
 $\Rightarrow r$ thỏa tất cả các phụ thuộc hàm của F^+ (theo định nghĩa)
 $\Rightarrow r$ thỏa tất cả các phụ thuộc hàm của $(F^+)^+$ (theo định nghĩa)
 $\Rightarrow r$ thỏa $X \rightarrow Y$ (vì (2)) $\Rightarrow X \rightarrow Y \in F^+$
 (1) và (3) $\Rightarrow (F^+)^+ = F^+$

2 Hệ luật dẫn Armstrong

Hệ luật dẫn là một phát biểu cho biết nếu một quan hệ r thỏa mãn một vài phụ thuộc hàm thì nó phải thỏa mãn phụ thuộc hàm khác.

Với X, Y, Z, W là tập con của Q^+ . r là quan hệ bất kỳ của Q . Ta có 6 luật dẫn sau:

1. **Luật phản xạ (reflexive rule):** $X \rightarrow X$
2. **Luật thêm vào (augmentation rule):** Cho $X \rightarrow Y \Rightarrow XZ \rightarrow Y$
3. **Luật hợp (union rule):** Cho $X \rightarrow Y, X \rightarrow Z \Rightarrow X \rightarrow YZ$
4. **Luật phân rã (decomposition rule):** Cho $X \rightarrow YZ \Rightarrow X \rightarrow Y$
5. **Luật bắc cầu (transitive rule):** Cho $X \rightarrow Y, Y \rightarrow Z \Rightarrow X \rightarrow Z$
6. **Luật bắc cầu giả (pseudo transitive rule):** Cho $X \rightarrow Y, YZ \rightarrow W \Rightarrow XZ \rightarrow W$

Hệ tiên đề Armstrong (Armstrong's Axioms) gồm 3 luật: (1), (2) và (5)

Chứng minh

Với t_1, t_2 là hai bộ bất kỳ của quan hệ r .

Luật phản xạ: Ta có $(t_1.X = t_2.X \Rightarrow t_1.X = t_2.X)$ theo định nghĩa suy ra $X \rightarrow X$

Luật thêm vào: giả sử có $t_1.XZ = t_2.XZ$ (1)
 $\Rightarrow t_1.X = t_2.X$
 $\Rightarrow t_1.Y = t_2.Y$ (do $X \rightarrow Y$) (2)
 $\Rightarrow XZ \rightarrow Y$ (do (1) \Rightarrow (2))

Luật hợp: giả sử có $t_1.X = t_2.X$ (1)
 $\Rightarrow t_1.X = t_2.X$ và $t_1.Z = t_2.Z$
 $\Rightarrow t_1.XZ = t_2.XZ$ (2)
 $\Rightarrow X \rightarrow YZ$ (do (1) \Rightarrow (2))

Luật phân rã: giả sử có: $t_1.X = t_2.X$ (1)
 $\Rightarrow t_1.YZ = t_2.YZ$ (do $X \rightarrow YZ$)
 $\Rightarrow t_1.Y = t_2.Y$ (2)
 $\Rightarrow X \rightarrow Y$ (do (1) \Rightarrow (2))

Luật bắc cầu: giả sử có $t_1.X = t_2.X$ (1)
 $\Rightarrow t_1.Y = t_2.Y$
 $\Rightarrow t_1.Z = t_2.Z$ (2)
 $\Rightarrow X \rightarrow Z$ (do (1) \Rightarrow (2))

Luật bắc cầu giả: giả sử có: $t_1.XZ = t_2.XZ$ (1)

$$\begin{aligned} \Rightarrow t_1.X &= t_2.X \text{ và } t_1.Z = t_2.Z \quad (2) \\ \Rightarrow t_1.Y &= t_2.Y \text{ (do } X \rightarrow Y \text{)} \quad (3) \\ \Rightarrow t_1.YZ &= t_2.YZ \text{ (Kết hợp (2) và (3))} \\ \Rightarrow t_1.W &= t_2.W \text{ (do } YZ \rightarrow W \text{)} \quad (4) \\ \Rightarrow XZ &\rightarrow W \end{aligned}$$

Trong 6 luật trên, chỉ cần 3 luật 1, 2 và 6 là đủ, nghĩa là các luật còn lại có thể suy dẫn từ ba luật này.

i Hệ luật dẫn Armstrong là đúng

Nói rằng $X \rightarrow Y$ là phụ thuộc hàm được suy diễn nhờ vào luật dẫn Armstrong nếu tồn tại các tập phụ thuộc hàm $F_0 \subset F_1 \subset \dots \subset F_n$ sao cho $X \rightarrow Y \in F_n$ với F_0, F_1, \dots, F_n lần lượt được hình thành thỏa phương pháp sau:

Bước 1: $F_0 = F$

Bước 2: chọn một số phụ thuộc hàm trong F_i áp dụng hệ luật dẫn Armstrong để thu được một số phụ thuộc hàm mới. Đặt $F_{i+1} = F_i \cup \{\text{các phụ thuộc hàm mới}\}$

Ví dụ: Cho $F = \{AB \rightarrow C, C \rightarrow B, BC \rightarrow A\}$ thì có $F_0 \subset F_1 \subset F_2$ sao cho $C \rightarrow A \in F_2$

$F_0 = \{AB \rightarrow C, C \rightarrow B, BC \rightarrow A\}$ áp dụng luật hợp cho $C \rightarrow B$ và $C \rightarrow C$

$F_1 = \{AB \rightarrow C, C \rightarrow B, BC \rightarrow A, C \rightarrow BC\}$ áp dụng luật bắc cầu.

$F_2 = \{AB \rightarrow C, C \rightarrow B, BC \rightarrow A, C \rightarrow BC, C \rightarrow A\}$

Hệ quả: Hệ luật dẫn Armstrong là đúng nghĩa là nếu F là tập các phụ thuộc hàm đúng trên quan hệ r và $X \rightarrow Y$ là một phụ thuộc hàm được suy diễn từ F nhờ hệ luật dẫn Armstrong thì $X \rightarrow Y$ đúng trên quan hệ r . Vậy $X \rightarrow Y$ là phụ thuộc hàm được suy diễn logic từ F

Phần tiếp theo chúng ta sẽ chứng minh hệ luật dẫn Armstrong là đầy đủ, nghĩa là mọi phụ thuộc hàm $X \rightarrow Y$ được suy diễn logic từ F sẽ được suy diễn từ F nhờ hệ luật dẫn Armstrong.

ii Bao đóng của tập thuộc tính X (closures of attribute sets)

(a) Định nghĩa

Q là lược đồ quan hệ, r là quan hệ tương ứng, F là tập các phụ thuộc hàm trong Q . X, A_i là các tập con của Q^+ .

Bao đóng của tập thuộc tính X đối với F ký hiệu là X^+ (hoặc X_F^+) được định nghĩa như sau:

$X^+ = \cup A_i$ với $X \rightarrow A_i$ là phụ thuộc hàm được suy diễn từ F nhờ hệ tiên đề Armstrong

Tính chất:

+ bao đóng của Q là Q^+

(b) Các tính chất của bao đóng

Nếu X, Y là các tập con của tập thuộc tính Q^+ thì ta có các tính chất sau đây:

1. **Tính phản xạ:** $X \subseteq X^+$
2. **Tính đơn điệu:** Nếu $X \subseteq Y$ thì $X^+ \subseteq Y^+$
3. **Tính lũy đẳng:** $X^{++} = X^+$
4. $(XY)^+ \supseteq X^+Y^+$
5. $(X^+Y)^+ = (XY^+)^+ = (X^+Y^+)^+$
6. $X \rightarrow Y \Leftrightarrow Y^+ \subseteq X^+$
7. $X \rightarrow X^+$ và $X^+ \rightarrow X$
8. $X^+ = Y^+ \Leftrightarrow X \rightarrow Y$ và $Y \rightarrow X$

Chứng minh:

1. $X \rightarrow X \Rightarrow X^+ \supseteq X$
 2. $A \in X^+ \Rightarrow X \rightarrow A \Rightarrow Y \rightarrow A \Rightarrow A \in Y^+$
 3. $A \in X^{++} \Rightarrow X^+ \rightarrow A$ và $X \rightarrow X^+$ (áp dụng 8) $\Rightarrow X \rightarrow A \Rightarrow A \in X^+$
 $\Rightarrow X^{++} \subseteq X^+$. Áp dụng 1 $\Rightarrow X^{++} \supseteq X^+$
-
7. $X \rightarrow A_1$ và $X \rightarrow A_2 \Rightarrow X \rightarrow A_1 \cup A_2 \dots X \rightarrow \cup A_i = X^+$
 $X^+ \supseteq X \Rightarrow X^+ \rightarrow X$ (Phụ thuộc hàm hiển nhiên)
-

(c) Thuật toán tìm bao đóng

Tính liên tiếp tập các tập thuộc tính X_0, X_1, X_2, \dots theo phương pháp sau:

Bước 1: $X_0 = X$

Bước 2: lần lượt xét các phụ thuộc hàm của F

Nếu $Y \rightarrow Z$ có $Y \subseteq X_i$ thì $X_{i+1} = X_i \cup Z$

Loại phụ thuộc hàm $Y \rightarrow Z$ khỏi F

Bước 3: Nếu ở bước 2 không tính được X_{i+1} thì X_i chính là bao đóng của X

Ngược lại lặp lại bước 2

Ví dụ 1:

Cho lược đồ quan hệ $Q(ABCDEFGHIH)$ và tập phụ thuộc hàm F

$F = \{$

$f_1:$	B	\rightarrow	A
$f_2:$	DA	\rightarrow	CE
$f_3:$	D	\rightarrow	H
$f_4:$	GH	\rightarrow	C
$f_5:$	AC	\rightarrow	D

$\}$

Tìm bao đóng của các tập $X = \{AC\}$ dựa trên F .

Giải:

Bước 1: $X_0 = AC$

Bước 2: Do f_1, f_2, f_3, f_4 không thỏa. f_5 thỏa vì $X^+ \supseteq AC$
 $X_1 = AC \cup D = ACD$

Lặp lại bước 2: f_1 không thỏa, f_2 thỏa vì $X_1 \supseteq AD$:

$$X_2 = ACD \cup CE = ACDE$$

f_3 thỏa vì $X_2 \supseteq D$

$$X_3 = ACDE \cup H = ACDEH$$

f_4 không thỏa, f_5 không xét vì đã thỏa

Lặp lại bước 2: f_2, f_3 không xét vì đã thỏa, f_1, f_4 không thỏa, f_5 không xét vì đã thỏa. Trong bước này X_3 không thay đổi $\Rightarrow X^+ = X_3 = \{ACDEH\}$ là bao đóng của X

Ví dụ 2:

$Q(A, B, C, D, E, G)$

$F = \{$

$f_1:$	A	\rightarrow	C
$f_2:$	A	\rightarrow	EG
$f_3:$	B	\rightarrow	D
$f_4:$	G	\rightarrow	E

$\}$

$X = \{A, B\}; Y = \{C, G, D\}$

Kết quả:

$X^+ = \{ABCDEG\}$

$Y^+ = \{CGDE\}$

(d) Định lý

Thuật toán tìm bao đóng cho kết quả $X_i = X^+$

Chứng minh

1. Ta chứng minh $X_{i-1} \subseteq X^+$ bằng phương pháp qui nạp.

Bước cơ sở chứng minh $X \rightarrow X_0$

Theo tính phản xạ của hệ luật dẫn thì $X \rightarrow X$ theo thuật toán thì $X_0 = X \Rightarrow X \rightarrow X_0$

Vậy $X_0 \subseteq X^+$

Bước qui nạp giả sử có $X \rightarrow X_{i-1}$ (1) ta phải chứng minh $X \rightarrow X_i$

Theo thuật toán tìm bao đóng thì có $f_j = X_j \rightarrow Y_j$ để $X_{i-1} \supseteq X_j$ và $X_i = X_{i-1} \cup Y_j$

$\Rightarrow X_{i-1} \rightarrow Y_j$ (2). (1) và (2) $\Rightarrow X \rightarrow Y_j$ (3)

(1) và (3) $\Rightarrow X \rightarrow X_{i-1}Y_j = X_i \Rightarrow X \rightarrow X_i$

Vậy $X_i \subseteq X^+$

2. Ta chứng minh $A \subseteq X^+ \Rightarrow A \subseteq X_i$ để kết luận $X_i \supseteq X^+$

$A \subseteq X^+$ nên có một phụ thuộc hàm $X \rightarrow A$. Theo thuật toán tìm bao đóng thì $X \subseteq X_i \Rightarrow A \subseteq X_i$

(e) Hệ quả

1. Q là lược đồ quan hệ. F là tập phụ thuộc hàm, A là thuộc tính chỉ xuất hiện ở vế phải của các phụ thuộc hàm trong F thì $X^+ = (X - A)^+ \cup A$
2. Q là lược đồ quan hệ. F là tập phụ thuộc hàm, X là tập con của Q^+ và $Y = \{\text{các thuộc tính xuất hiện ở vế phải của các phụ thuộc hàm trong } F\}$ thì $X^+ \subseteq X \cup Y$.

Chứng minh

1. Theo thuật toán tìm bao đóng thì bao đóng X^+ hay $(X-A)^+$ được hình thành qua một số bước. Ta chứng minh biểu thức $X^+ = (X - A)^+ \cup A$ theo qui nạp.

Bước cơ sở: $X_0 = X$, $(X-A)_0 = X - A \Rightarrow X_0 = (X - A)_0 \cup A$ đúng

Bước qui nạp: giả sử ta có $X_{i-1} = (X - A)_{i-1} \cup A$. Bao đóng X_i được hình thành do có $f_j = X_j \rightarrow Y_j$ để:

$X_{i-1} \supseteq X_j$ và $X_i = X_{i-1} \cup Y_j = (X - A)_{i-1} \cup A \cup Y_j$ (1).

Sự hình thành X_i luôn kéo theo sự hình thành $(X-A)_i$ vì:

$X_{i-1} = (X - A)_{i-1} \cup A \supseteq X_j$ do X_j không chứa A nên:

$(X - A)_{i-1} \supseteq X_j$ vậy $(X - A)_i = (X - A)_{i-1} \cup Y_j$ (2)

(1) và (2) cho:

$X_i = (X - A)_i \cup A$ là điều phải chứng minh

2. **Bước cơ sở:** $X_0 = X \Rightarrow X_0 \subseteq X \cup Y$

Bước qui nạp: giả sử có $X_{i-1} \subseteq X \cup Y$ ta chứng minh $X_i \subseteq X \cup Y$.

Bao đóng X_i được hình thành do có $f_j = X_j \rightarrow Y_j$ để:

$X_{i-1} \supseteq X_j$ và $X_i = X_{i-1} \cup Y_j \subseteq X \cup Y \cup Y_j$ do Y_j là vế phải của phụ thuộc hàm nên $Y \cup Y_j = Y$ vậy $X_i \subseteq X \cup Y$

3 Hệ luật dẫn Armstrong là đầy đủ**i Định lý**

Hệ luật dẫn Armstrong là đầy đủ nghĩa là mọi phụ thuộc hàm $X \rightarrow Y$ được suy diễn logic từ F sẽ được suy diễn từ F nhờ hệ luật dẫn Armstrong.

Chứng minh:

Để chứng minh $X \rightarrow Y$ được suy diễn từ F nhờ hệ luật dẫn Armstrong ta chứng minh bằng phương pháp phản chứng nghĩa là nếu $X \rightarrow Y$ không suy diễn được từ hệ luật dẫn Armstrong thì có quan

hệ r thỏa các phụ thuộc hàm F nhưng không thỏa phụ thuộc hàm $X \rightarrow Y$ (điều này nghịch lý với giả thuyết là mọi quan hệ r thỏa các phụ thuộc hàm trong F thì r cũng thỏa phụ thuộc hàm $X \rightarrow Y$).

Thật vậy giả sử $Q(A_1, A_2, \dots, A_n)$ là lược đồ quan hệ, a_i, b_i là các giá trị khác nhau trên miền giá trị A_i . r là quan hệ trên Q có hai bộ t và t' được xác định như sau:

$$t = (a_1, a_2, \dots, a_n)$$

$$t'.A_i = \begin{cases} a_i & \text{Nếu } A_i \in X^+ \\ b_i & \text{Ngược lại} \end{cases}$$

Vậy quan hệ r có $t.X = t'.X$ nhưng $t.Y \neq t'.Y$ ($t.Y$ gồm các giá trị a_i còn $t'.Y$ phải có ít nhất một b_i nếu không $Y \subseteq X^+ \Rightarrow X \rightarrow Y$ được suy dẫn từ hệ luật dẫn Armstrong). Như vậy r không thỏa phụ thuộc hàm $X \rightarrow Y$.

Bây giờ ta chứng minh quan hệ r thỏa mọi phụ thuộc hàm trong F . Gọi $W \rightarrow Z$ là phụ thuộc hàm trong F .

Nếu $W \not\subseteq X^+ \Rightarrow t.W \neq t'.W \Rightarrow$ mệnh đề $(t.W = t'.W \Rightarrow t.Z = t'.Z)$ đúng

Nếu $W \subseteq X^+ \Rightarrow t.Z = t'.Z =$ bộ các a_i

\Rightarrow mệnh đề $(t.W = t'.W \Rightarrow t.Z = t'.Z)$ đúng

ii Hệ quả:

Bao đóng của tập thuộc tính X đối với F là:

$X^+ = \cup A_i$ với $X \rightarrow A_i$ là phụ thuộc hàm được suy diễn logic từ F

Tính chất

$$X \rightarrow Y \in F^+ \Leftrightarrow Y \subseteq X^+$$

Chứng minh

$X \rightarrow Y \Rightarrow$ có k sao cho $Y = A_k \subseteq \cup A_i = X^+$

$Y \subseteq X^+ \Rightarrow X^+ = Y \cup (X^+ - Y) \Rightarrow X \rightarrow Y \cup (X^+ - Y) \Rightarrow X \rightarrow Y$

Bài toán thành viên

Nói rằng $X \rightarrow Y$ là thành viên của F nếu $X \rightarrow Y \in F^+$

Một vấn đề quan trọng khi nghiên cứu lý thuyết CSDL là khi cho trước tập các phụ thuộc hàm F và một phụ thuộc hàm $X \rightarrow Y$, làm thế nào để biết $X \rightarrow Y$ có thuộc F^+ hay không bài toán này được gọi là bài toán thành viên. Để trả lời câu hỏi này ta có thể tính F^+ rồi xác định xem $X \rightarrow Y$ có thuộc F^+ hay không. Việc tính F^+ là một công việc đòi hỏi thời gian và công sức. Tuy nhiên, thay vì tính F^+ chúng ta có thể dùng thuật toán sau để xác định $X \rightarrow Y$ có là thành viên của F hay không. Thuật toán này sử dụng tính chất vừa chứng minh trên.

Thuật toán xác định $F = X \rightarrow Y$ có là thành viên của F hay không

Bước 1: tính X^+

Bước 2: so sánh X^+ với Y nếu $X^+ \supseteq Y$ thì ta khẳng định $X \rightarrow Y$ là thành viên của F

Bạn đọc hãy nắm thật kỹ thuật toán này – nó mở đầu cho một loạt ứng dụng về sau.

III THUẬT TOÁN TÌM F^+

1 Thuật toán cơ bản

Để tính bao đóng F^+ của tập các phụ thuộc hàm F ta thực hiện các bước sau:

Bước 1: Tìm tất cả tập con của Q^+

Bước 2: Tìm tất cả các phụ thuộc hàm có thể có của Q .

Bước 3: Tìm bao đóng của tất cả tập con của Q .

Bước 4: Dựa vào bao đóng của tất cả các tập con đã tìm để xác định phụ thuộc hàm nào thuộc F^+

Ví dụ 3:

Cho lược đồ quan hệ $Q(A, B, C)$ $F = \{AB \rightarrow C, C \rightarrow B\}$ là tập phụ thuộc hàm trên Q . F^+ được tính lần lượt theo các bước trên là như sau:

Bước 1: Các tập con của Q^+

\emptyset	A	B	C
\emptyset	{A}	{B}	{C}
		{A, B}	{A, C}
			{B, C}
			{A, B, C}

Bước 2: các phụ thuộc hàm có thể có của F (không kê các phụ thuộc hàm hiển nhiên)

$A \rightarrow B$	$A \rightarrow BC$	$B \rightarrow C$	$AB \rightarrow C \in F$	$C \rightarrow A$	$C \rightarrow BC \in F^+$	$AC \rightarrow BC \in F^+$	$BC \rightarrow AC$
$A \rightarrow AB$	$A \rightarrow ABC$	$B \rightarrow AC$	$AB \rightarrow AC \in F^+$	$C \rightarrow B \in F$	$C \rightarrow ABC$	$AC \rightarrow ABC \in F^+$	$BC \rightarrow ABC$
$A \rightarrow C$	$B \rightarrow A$	$B \rightarrow BC$	$AB \rightarrow BC \in F^+$	$C \rightarrow AB$	$AC \rightarrow B \in F^+$	$BC \rightarrow A$	
$A \rightarrow AC$	$B \rightarrow AB$	$B \rightarrow ABC$	$AB \rightarrow ABC \in F^+$	$C \rightarrow AC$	$AC \rightarrow AB \in F^+$	$BC \rightarrow AB$	

Bước 3: bao đóng của các tập con của Q đối với F

$$\begin{aligned} \emptyset^+ &= \emptyset & A^+ &= A & C^+ &= BC \\ ABC^+ &= ABC & B^+ &= B & AC^+ &= ABC \\ & & AB^+ &= ABC & BC^+ &= BC \end{aligned}$$

Bước 4: $F = \{AB \rightarrow C, C \rightarrow B\}$ suy ra:

$$F^+ = \{AB \rightarrow C, AB \rightarrow AC, AB \rightarrow BC, AB \rightarrow ABC, C \rightarrow B, C \rightarrow BC, AC \rightarrow B, AC \rightarrow AB, AC \rightarrow BC, AC \rightarrow ABC\}$$

2 Thuật toán cải tiến

Dựa vào thuật toán cơ bản trên, ta nhận thấy có thể tính F^+ theo các bước sau:

Bước 1: Tìm tất cả tập con của Q^+

Bước 2: Tìm bao đóng của tất cả tập con của Q^+ .

Bước 4: Dựa vào bao đóng của các tập con đã tìm để suy ra các phụ thuộc hàm thuộc F^+ .

Ví dụ bao đóng $A^+ = A$ chỉ gồm các phụ thuộc hàm hiển nhiên

bao đóng $\{AB\}^+ = ABC$ cho các phụ thuộc hàm không hiển nhiên sau

$$AB \rightarrow C, AB \rightarrow AC, AB \rightarrow BC, AB \rightarrow ABC$$

(Tìm tất cả các tập con của $\{ABC\}$ rồi bỏ các tập con của $\{AB\}$)

Các tập con của $\{ABC\}$ là: $\emptyset, \{A\}, \{B\}, \{AB\}, \{C\}, \{AC\}, \{BC\}, \{ABC\}$

Bỏ các tập con của $\{AB\}$ là: $\emptyset, \{A\}, \{B\}, \{AB\}, \{C\}, \{AC\}, \{BC\}, \{ABC\}$

Các tập còn lại chính là vế phải của phụ thuộc hàm có vế trái là AB

IV BÀI TẬP

1/ Cho quan hệ sau:

r	A	B	C	D	E
a_1	b_1	c_1	d_1	e_1	
a_1	b_2	c_2	d_2	d_1	
a_2	b_1	c_3	d_3	e_1	
a_2	b_1	c_4	d_3	e_1	
a_3	b_2	c_5	d_1	e_1	

Phụ thuộc hàm nào sau đây thỏa r:

$$A \rightarrow D, AB \rightarrow D, C \rightarrow BDE, E \rightarrow A, A \rightarrow E$$

2/ Cho $Q^+ = \{ABCD\}$

- a) Tìm tất cả các tập con của Q
- b) Tìm tất cả các phụ thuộc hàm có thể có của Q (không liệt kê phụ thuộc hàm hiển nhiên)
- 3/ Tìm bao đóng F^+ của quan hệ $phanCong$ (PHICONG, MAYBAY, NGAYKH, GIOKH)
- 4/ Cho $F = \{AB \rightarrow C, B \rightarrow D, CD \rightarrow E, CE \rightarrow GH, G \rightarrow A\}$
- a) Hãy chứng tỏ phụ thuộc hàm $AB \rightarrow E, AB \rightarrow G$ được suy diễn từ F nhờ luật dẫn Armstrong
- b) Tìm bao đóng của AB (với bài toán không nói gì về lược đồ quan hệ Q ta ngầm hiểu Q^+ là tập thuộc tính có trong F nghĩa là $Q^+ = \{ABCDEFGHI\}$)
- 5/ Cho $F = \{A \rightarrow D, AB \rightarrow DE, CE \rightarrow G, E \rightarrow H\}$. Hãy tìm bao đóng của AB .
- 6/ Cho $F = \{AB \rightarrow E, AG \rightarrow I, BE \rightarrow I, E \rightarrow G, GI \rightarrow H\}$.
- a) Hãy chứng tỏ phụ thuộc hàm $AB \rightarrow GH$ được suy diễn từ F nhờ luật dẫn Armstrong
- b) Tìm bao đóng của $\{AB\}$
- 7/ Cho $F = \{A \rightarrow D, AB \rightarrow E, BI \rightarrow E, CD \rightarrow I, E \rightarrow C\}$ tìm bao đóng của $\{AE\}^+ = \{ACDEI\}$

-----oOo-----

Chương 5 .

PHỦ CỦA TẬP PHỤ THUỘC HÀM

I ĐỊNH NGHĨA

Nói rằng hai tập phụ thuộc hàm F và G là tương đương (Equivalent) nếu $F^+ = G^+$ ký hiệu $F \equiv G$.
Ta nói F phủ G nếu $F^+ \supseteq G^+$

Thuật toán xác định F và G có tương đương không

Bước 1: Với mỗi phụ thuộc hàm $X \rightarrow Y$ của F ta xác định xem $X \rightarrow Y$ có là thành viên của G không

Bước 2: Với mỗi phụ thuộc hàm $X \rightarrow Y$ của G ta xác định xem $X \rightarrow Y$ có là thành viên của F không

Nếu cả hai bước trên đều đúng thì $F \equiv G$

Ví dụ 1: Cho lược đồ quan hệ Q (ABCDE) hai tập phụ thuộc hàm:

$$F = \{A \rightarrow BC, A \rightarrow D, CD \rightarrow E\} \text{ và } G = \{A \rightarrow BCE, A \rightarrow ABD, CD \rightarrow E\}$$

a) F có tương đương với G không?

b) F có tương đương với $G' = \{A \rightarrow BCDE\}$ không?

Giải:

a) Ta có $A_G^+ = ABCDE \Rightarrow$ trong G^+ có $A \rightarrow BC$ và $A \rightarrow D \Rightarrow F \subseteq G^+ \Rightarrow F^+ \subseteq G^+$ (1).

$A_F^+ = ABCDE \Rightarrow$ trong F^+ có $A \rightarrow BCE$ và $A \rightarrow ABD \Rightarrow F^+ \supseteq G \Rightarrow F^+ \supseteq G^+$ (2)

(1) và (2) $\Rightarrow F^+ = G^+ \Rightarrow F \equiv G$.

b) Do $(CD)_{G'}^+ = CD \Rightarrow G'^+$ không chứa phụ thuộc hàm $CD \rightarrow E \Rightarrow F$ không tương đương với G'

II PHỦ TỐI THIỂU CỦA MỘT TẬP PHỤ THUỘC HÀM (minimal cover)

1 Phụ thuộc hàm có vế trái dư thừa

F là tập các phụ thuộc hàm trên lược đồ quan hệ Q , Z là tập thuộc tính, $Z \rightarrow Y \in F$. Nói rằng phụ thuộc hàm $Z \rightarrow Y$ có vế trái dư thừa (phụ thuộc không đầy đủ) nếu có một $A \in Z$ sao cho:

$$F \equiv F - \{Z \rightarrow Y\} \cup \{(Z-A) \rightarrow Y\}$$

Ngược lại $Z \rightarrow Y$ là phụ thuộc hàm có vế trái không dư thừa hay Y phụ thuộc hàm đầy đủ vào Z hay phụ thuộc hàm đầy đủ.

Ví dụ 2: $Q(A, B, C)$ $F = \{AB \rightarrow C; B \rightarrow C\}$

$$F \equiv F - \{AB \rightarrow C\} \cup \{(AB-A) \rightarrow C\} = \{B \rightarrow C\}$$

$AB \rightarrow C$ là phụ thuộc hàm không đầy đủ

$B \rightarrow C$ là phụ thuộc hàm đầy đủ

Chú ý: phụ thuộc hàm có vế trái chứa một thuộc tính là phụ thuộc hàm đầy đủ.

Ví dụ 3: cho tập phụ thuộc hàm $F = \{A \rightarrow BC, B \rightarrow C, AB \rightarrow D\}$ thì phụ thuộc hàm $AB \rightarrow D$ có vế trái dư thừa B vì:

$$F \equiv F - \{AB \rightarrow D\} \cup \{A \rightarrow D\}$$

$$\equiv \{A \rightarrow BC, B \rightarrow C, A \rightarrow D\}$$

Ta nói F là tập phụ thuộc hàm có vế trái không dư thừa nếu F không chứa phụ thuộc hàm có vế trái dư thừa.

Thuật toán loại khỏi F các phụ thuộc hàm có vế trái dư thừa.

Bước 1: lần lượt thực hiện bước 2 cho các phụ thuộc hàm $X \rightarrow Y$ của F

Bước 2: Với mọi tập con thật sự $X' \neq \emptyset$ của X .

Nếu $X' \rightarrow Y \in F^+$ thì thay $X \rightarrow Y$ trong F bằng $X' \rightarrow Y$ thực hiện lại bước 2

Ví dụ: Ở ví dụ 3 phụ thuộc hàm $AB \rightarrow D$ có $A^+ = ABCD \Rightarrow A \rightarrow D \in F^+$. Trong F ta thay $AB \rightarrow D$ bằng $A \rightarrow D \Rightarrow F \equiv \{A \rightarrow BC, B \rightarrow C, A \rightarrow D\}$

2 Tập phụ thuộc hàm có vế phải một thuộc tính (the right sides of dependencies has a single attribute)

Mỗi tập phụ thuộc hàm F đều tương đương với một tập phụ thuộc hàm G mà vế phải của các phụ thuộc hàm trong G chỉ gồm một thuộc tính.

Ví dụ 4: cho $F = \{A \rightarrow BC, B \rightarrow C, AB \rightarrow D\}$ ta suy ra

$$F \equiv \{A \rightarrow B, A \rightarrow C, B \rightarrow C, AB \rightarrow D\} = G$$

G được gọi là **tập phụ thuộc hàm có vế phải một thuộc tính**.

3 Tập phụ thuộc hàm không dư thừa

Nói rằng F là tập phụ thuộc hàm không dư thừa nếu không tồn tại $F' \subset F$ sao cho $F' \equiv F$. Ngược lại F là tập phụ thuộc hàm dư thừa.

Ví dụ: cho $F = \{A \rightarrow BC, B \rightarrow D, AB \rightarrow D\}$ thì F dư thừa vì

$$F \equiv F' = \{A \rightarrow BC, B \rightarrow D\}$$

Thuật toán loại khỏi F các phụ thuộc hàm dư thừa:

Bước 1: Lần lượt xét các phụ thuộc hàm $X \rightarrow Y$ của F

Bước 2: nếu $X \rightarrow Y$ là thành viên của $F - \{X \rightarrow Y\}$ thì loại $X \rightarrow Y$ khỏi F

Bước 3: thực hiện bước 2 cho các phụ thuộc hàm tiếp theo của F

4 Tập phụ thuộc hàm tối thiểu (minimal cover)

F được gọi là một tập phụ thuộc hàm tối thiểu (hay phủ tối thiểu) nếu F thỏa đồng thời ba điều kiện sau:

1. F là tập phụ thuộc hàm có vế trái không dư thừa
2. F là tập phụ thuộc hàm có vế phải một thuộc tính.
3. F là tập phụ thuộc hàm không dư thừa

Thuật toán tìm phủ tối thiểu của một tập phụ thuộc hàm

Bước 1: loại khỏi F các phụ thuộc hàm có vế trái dư thừa.

Bước 2: Tách các phụ thuộc hàm có vế phải trên một thuộc tính thành các phụ thuộc hàm có vế phải một thuộc tính.

Bước 3: loại khỏi F các phụ thuộc hàm dư thừa.

Chú ý: Theo thuật toán trên, từ một tập phụ thuộc hàm F luôn tìm được ít nhất một phủ tối thiểu F_{tt} để $F \equiv F_{tt}$ và nếu thứ tự loại các phụ thuộc hàm trong tập F là khác nhau thì có thể sẽ thu được những phủ tối thiểu khác nhau.

Ví dụ 5: Cho lược đồ quan hệ $Q(A, B, C, D)$ và tập phụ thuộc F như sau:

$$F = \{AB \rightarrow CD, B \rightarrow C, C \rightarrow D\}$$

Hãy tính phủ tối thiểu của F .

Giải:

Bước 1: $AB \rightarrow CD$ là phụ thuộc hàm có vế trái dư thừa?

$$B \rightarrow CD \in F^+? \text{ trả lời: } B^+ = BCD \Rightarrow B \rightarrow CD \in F^+$$

Vậy $AB \rightarrow CD$ là phụ thuộc hàm có vế trái dư thừa $A \Rightarrow$ kết quả của bước 1 là:

$$F \equiv \{B \rightarrow CD; B \rightarrow C; C \rightarrow D\}$$

Bước 2: kết quả của bước 2 là:

$$F \equiv \{B \rightarrow D; B \rightarrow C; C \rightarrow D\} = F_{1tt}$$

Bước 3: trong F_{1tt} , $B \rightarrow C$ là phụ thuộc hàm dư thừa?

$$B \rightarrow C \in G^+? \text{ với } G = F_{1tt} - \{B \rightarrow C\} = \{B \rightarrow D; C \rightarrow D\}$$

$$B_G^+ = BD \Rightarrow B \rightarrow C \notin G^+ \Rightarrow \text{trong } F_{1tt} \text{ } B \rightarrow C \text{ không dư thừa.}$$

trong F_{1tt} , $B \rightarrow D$ là phụ thuộc hàm dư thừa?

$$B \rightarrow D \in G^+? \text{ với } G = F_{1tt} - \{B \rightarrow D\} = \{B \rightarrow C; C \rightarrow D\}$$

$$B_G^+ = BCD \Rightarrow B \rightarrow D \in G^+ \Rightarrow \text{trong } F_{1tt}, B \rightarrow D \text{ dư thừa.}$$

kết quả của bước 3 cho phủ tối thiểu:

$$F \equiv \{B \rightarrow C; C \rightarrow D\} = F_{tt}$$

Ví dụ 6: Cho lược đồ quan hệ Q (MSCD, MSSV, CD, HG) và tập phụ thuộc F như sau:

$$F = \{ \begin{array}{l} \text{MSCD} \rightarrow \text{CD}; \\ \text{CD} \rightarrow \text{MSCD}; \\ \text{CD, MSSV} \rightarrow \text{HG}; \\ \text{MSCD, HG} \rightarrow \text{MSSV}; \\ \text{CD, HG} \rightarrow \text{MSSV}; \\ \text{MSCD, MSSV} \rightarrow \text{HG} \end{array} \}$$

Hãy tìm phủ tối thiểu của F

kết quả:

$$F_{tt} = \{ \begin{array}{l} \text{MSCD} \rightarrow \text{CD}; \\ \text{CD} \rightarrow \text{MSCD}; \\ \text{CD, HG} \rightarrow \text{MSSV}; \\ \text{MSCD, MSSV} \rightarrow \text{HG} \end{array} \}$$

III KHÓA CỦA LƯỢC ĐỒ QUAN HỆ (Key)

1 Định Nghĩa

$Q(A_1, A_2, \dots, A_n)$ là lược đồ quan hệ.

Q^+ là tập thuộc tính của Q .

F là tập phụ thuộc hàm trên Q .

K là tập con của Q^+ .

Nói rằng K là một khóa của Q nếu:

1. $K^+ = Q^+$ và
2. Không tồn tại $K' \subset K$ sao cho $K'^+ = Q^+$

Tập thuộc tính S được gọi là **siêu khóa** nếu $S \supseteq K$

Thuộc tính A được gọi là **thuộc tính khóa** nếu $A \in K$ với K là khóa bất kỳ của Q . Ngược lại A được gọi là **thuộc tính không khóa**.

Một lược đồ quan hệ có thể có nhiều khóa và tập thuộc tính không khóa cũng có thể bằng rỗng.

(Khi thiết kế một hệ thống thông tin, thì việc lập lược đồ cơ sở dữ liệu đạt đến một tiêu chuẩn nào đó là một việc làm quan trọng. Việc xác định chuẩn cho một lược đồ quan hệ có liên quan mật thiết với thuật toán tìm khóa).

Thuật toán tìm một khóa của một lược đồ quan hệ Q

Bước 1: gán $K = Q^+$

Bước 2: A là một thuộc tính của K, đặt $K' = K - A$. Nếu $K'^+ = Q^+$ thì gán $K = K'$ thực hiện lại bước 2

Nếu muốn tìm các khóa khác (nếu có) của lược đồ quan hệ, ta có thể thay đổi thứ tự loại bỏ các phần tử của K.

Ví dụ 7:

$Q(A, B, C, D, E, G, H, I) F = \{AC \rightarrow B; BI \rightarrow ACD; ABC \rightarrow D; H \rightarrow I; ACE \rightarrow BCG; CG \rightarrow AE\}$

Tìm K

Lần lượt loại các thuộc tính trong K theo thứ tự sau:

A, B, D, E, I

Ta được một khóa là của lược đồ quan hệ là $\{C, G, H\}$

(Lưu ý là thuật toán này chỉ nên sử dụng trong trường hợp chỉ cần tìm một khóa).

2 Thuật toán tìm tất cả khóa

i Thuật toán cơ bản

Bước 1: Xác định tất cả các tập con khác rỗng của Q^+ . Kết quả tìm được giả sử là các tập thuộc tính $X_1, X_2, \dots, X_{2^n-1}$

Bước 2: Tìm bao đóng của các X_i

Bước 3: Siêu khóa là các X_i có bao đóng đúng bằng Q^+ . Giả sử ta đã có các siêu khóa là $S = \{S_1, S_2, \dots, S_m\}$

Bước 4: Xây dựng tập chứa tất cả các khóa của Q từ tập S bằng cách xét mọi S_i, S_j con của S ($i \neq j$), nếu $S_i \subset S_j$ thì ta loại S_j ($i, j = 1..n$), kết quả còn lại của S chính là tập tất cả các khóa cần tìm.

Ví dụ 8: Tìm tất cả các khóa của lược đồ quan hệ và tập phụ thuộc hàm như sau:

$Q(C, S, Z); F = \{f_1: CS \rightarrow Z; f_2: Z \rightarrow C\}$

X_i	X_i^+	Siêu khóa	khóa
C	C		
S	S		
CS	CSZ	CS	CS
Z	ZC		
CZ	CZ		
SZ	SZC	SZ	SZ
CSZ	CSZ	CSZ	

Vậy lược đồ quan hệ Q có hai khóa là: $\{C, S\}$ và $\{S, Z\}$

Thuật toán trên thì dễ hiểu, dễ cài đặt, tuy nhiên nếu với n khá lớn thì phép duyệt để tìm ra tập tất cả các tập con của tập Q^+ là không hiệu quả. Do vậy cần thu hẹp không gian duyệt. Chúng ta sẽ nghiên cứu thuật toán cải tiến theo hướng giảm số thuộc tính của tập cần duyệt tất cả các tập con.

ii Thuật toán cải tiến

Trước khi đi vào thuật toán cải tiến, ta cần quan tâm một số khái niệm sau:

- + Tập thuộc tính nguồn (TN) chứa tất cả các thuộc tính có xuất hiện ở vế trái và không xuất hiện ở vế phải của các phụ thuộc hàm và các thuộc tính không xuất hiện ở cả vế trái lẫn vế phải của các phụ thuộc hàm.
- + Tập thuộc tính đích (TD) chứa tất cả các thuộc tính có xuất hiện ở vế phải và không xuất hiện ở vế trái của các phụ thuộc hàm.

- + Tập thuộc tính trung gian (TG) chứa tất cả các thuộc tính xuất hiện ở cả vế trái lẫn vế phải của các phụ thuộc hàm.

Hệ quả: Nếu K là khóa của Q thì $TN \subseteq K$ và $TD \cap K = \emptyset$

Chứng minh $TN \subseteq K$

Theo hệ quả 2 của thuật toán tìm bao đóng ta có $K^+ \subseteq K \cup TD \cup TG$

Ta chứng minh $A \in TN \Rightarrow A \in K$. Thật vậy:

Nếu $A \notin K \Rightarrow K^+ \subseteq K \cup TD \cup TG \subseteq Q^+ - A \Rightarrow K$ không là khóa \Rightarrow mâu thuẫn

Chứng minh $TD \cap K = \emptyset$

Giả sử có thuộc tính $A \in TD \cap K$ ta sẽ dẫn đến điều mâu thuẫn. Thật vậy:

Theo hệ quả 1 của thuật toán tìm bao đóng thì $K^+ = (K-A)^+ \cup A$

$A \in TD \Rightarrow$ có X là vế trái của một phụ thuộc hàm trong F sao cho $X \rightarrow A$ (1) và $A \notin X \Rightarrow X \subseteq K^+ = (K-A)^+ \cup A$ vì $A \notin X \Rightarrow X \subseteq (K-A)^+ \Rightarrow (K-A) \rightarrow X$ (2)

(1) và (2) cho $(K-A) \rightarrow A \Rightarrow A \in (K-A)^+ \Rightarrow (K-A)^+ \cup A = (K-A)^+ \Rightarrow K^+ = (K-A)^+$ mâu thuẫn với điều K là khóa.

Dựa vào hệ quả trên ta có thuật toán tìm tất cả khóa sau:

Dữ liệu vào: Lược đồ quan hệ Q và tập phụ thuộc hàm F

Dữ liệu ra: Tất cả các khóa của quan hệ

Thuật toán tìm tất cả khóa của một lược đồ quan hệ

Bước 1: tạo tập thuộc tính nguồn TN , tập thuộc tính trung gian TG

Bước 2: if $TG = \emptyset$ then lược đồ quan hệ chỉ có một khóa K

$$K = TN$$

kết thúc

Ngược lại

Qua bước 3

Bước 3: tìm tất cả các tập con X_i của tập trung gian TG

Bước 4: tìm các siêu khóa S_i bằng cách $\forall X_i$

if $(TN \cup X_i)^+ = Q^+$ then

$$S_i = TN \cup X_i$$

Bước 5: tìm khóa bằng cách loại bỏ các siêu khóa không tối thiểu

$$\forall S_i, S_j \in S$$

if $S_i \subset S_j$ then Loại S_j ra khỏi Tập siêu khóa S

S còn lại chính là tập khóa cần tìm.

Ví dụ 9: Giải lại bài tập ví dụ 8

Áp dụng thuật toán cải tiến ta có lời giải như sau:

$$TN = \{S\}; TG = \{C, Z\}$$

Gọi X_i là các tập con của tập TG :

X_i	$(TN \cup X_i)$	$(TN \cup X_i)^+$	Siêu khóa	khóa
\emptyset	S	S		
C	SC	Q^+	SC	SC
Z	SZ	Q^+	SZ	SZ
CZ	SCZ	Q^+	SCZ	

Kết quả quan hệ trên có hai khóa là : $\{S, C\}$ và $\{S, Z\}$

IV BÀI TẬP

1/ Chứng minh các tính chất sau:

- a) Tính cộng đầy đủ $X \rightarrow Y$ và $Z \rightarrow W \Rightarrow XZ \rightarrow YW$
 b) Tính tích lũy $X \rightarrow Y$ và $Y \rightarrow ZW \Rightarrow X \rightarrow YZW$

2/ Cho $G = \{AB \rightarrow C, A \rightarrow B, B \rightarrow C, A \rightarrow C\}$. $F = \{AB \rightarrow C, A \rightarrow B, B \rightarrow C\}$ có tương đương với G không?

3/ Cho lược đồ CSDL

Kehoach (NGAY, GIO, PHONG, MONHOC, GIAOVIEN)

$F = \{ \text{NGAY, GIO, PHONG} \rightarrow \text{MONHOC}$
 $\text{MONHOC, NGAY} \rightarrow \text{GIAOVIEN}$
 $\text{NGAY, GIO, PHONG} \rightarrow \text{GIAOVIEN}$
 $\text{MONHOC} \rightarrow \text{GIAOVIEN} \}$

- a) Tính $\{ \text{NGAY, GIO, PHONG} \}^+$; $\{ \text{MONHOC} \}^+$
 b) Tìm phủ tối thiểu của F
 c) Tìm tất cả các khóa của Kehoach

4/ Cho lược đồ CSDL

Q (TENTAU, LOAITAU, MACHUYEN, LUONGHANG, BENCANG, NGAY)

$F = \{ \text{TENTAU} \rightarrow \text{LOAITAU}$
 $\text{MACHUYEN} \rightarrow \text{TENTAU, LUONGHANG}$
 $\text{TENTAU, NGAY} \rightarrow \text{BENCANG, MACHUYEN} \}$

- a) Hãy tìm tập phủ tối thiểu của F
 b) Tìm tất cả các khóa của Q

5/ $Q(A, B, C, D, E, G)$

Cho $F = \{AB \rightarrow C; C \rightarrow A; BC \rightarrow D; ACD \rightarrow B; D \rightarrow EG; BE \rightarrow C; CG \rightarrow BD; CE \rightarrow AG\}$

$X = \{B, D\}$, $X^+ = ?$

$Y = \{C, G\}$, $Y^+ = ?$

6/ cho lược đồ quan hệ Q và tập phụ thuộc hàm F

a) $F = \{AB \rightarrow E; AG \rightarrow I; BE \rightarrow I; E \rightarrow G; GI \rightarrow H\}$ chứng minh rằng $AB \rightarrow GH$.

b) $F = \{AB \rightarrow C; B \rightarrow D; CD \rightarrow E; CE \rightarrow GH; G \rightarrow A\}$ chứng minh rằng $AB \rightarrow E; AB \rightarrow G$

7/ Cho quan hệ r

A	B	C	D
x	u	x	Y
y	x	z	x
z	y	y	y
y	z	w	z

Trong các phụ thuộc hàm sau đây, PTH nào không thỏa

$A \rightarrow B; A \rightarrow C; B \rightarrow A; C \rightarrow D; D \rightarrow C; D \rightarrow A$

8/ Hãy tìm tất cả các khóa cho lược đồ quan hệ sau:

Q (BROKER, OFFICE, STOCK, QUANTITY, INVESTOR, DIVIDENT)

$F = \{ \text{STOCK} \rightarrow \text{DIVIDENT}$
 $\text{INVESTOR} \rightarrow \text{BROKER}$
 $\text{INVESTOR, STOCK} \rightarrow \text{QUANTITY}$
 $\text{BROKER} \rightarrow \text{OFFICE} \}$

9/ Xét lược đồ quan hệ và tập phụ thuộc dữ liệu:

$Q(C, T, H, R, S, G)$

$f = \{ f_1: C \rightarrow T;$

$f_2: HR \rightarrow C;$

$f_3: HT \rightarrow R;$

$f_4: CS \rightarrow G;$

$f_5: HS \rightarrow R \}$

Tìm phủ tối thiểu của F

10/ $Q(A, B, C, D, E, H)$

$F = \{A \rightarrow E; C \rightarrow D; E \rightarrow DH\}$

Chứng minh $K = \{A, B, C\}$ là khóa duy nhất của Q

11/ $Q(A, B, C, D)$

$F = \{AB \rightarrow C; D \rightarrow B; C \rightarrow ABD\}$

Hãy tìm tất cả các khóa của Q

12/ $Q(A, B, C, D, E, G)$

$F = \{AB \rightarrow C; C \rightarrow A; BC \rightarrow D; ACD \rightarrow B; D \rightarrow EG; BE \rightarrow C; CG \rightarrow BD; CE \rightarrow G\}$

Hãy tìm tất cả các khóa của Q.

13/ Xác định phủ tối thiểu của tập phụ thuộc hàm sau:

a) $Q(A, B, C, D, E, G)$,

$F = \{AB \rightarrow C; C \rightarrow A; BC \rightarrow D; ACD \rightarrow B; D \rightarrow EG; BE \rightarrow C; CG \rightarrow BD; CE \rightarrow AG\}$

b) $Q(A, B, C)$

$F = \{A \rightarrow B, A \rightarrow C, B \rightarrow A, C \rightarrow A, B \rightarrow C\}$

14/ Xác định phủ tối thiểu của các tập phụ thuộc hàm sau:

a) $Q_1(ABCDEFGH)$

$F_1 = \{A \rightarrow H, AB \rightarrow C, BC \rightarrow D; G \rightarrow B\}$

b) $Q_2(ABCSXYZ)$

$F_2 = \{S \rightarrow A; AX \rightarrow B; S \rightarrow B; BY \rightarrow C; CZ \rightarrow X\}$

c) $Q_3(ABCDEFGH IJ)$

$F_3 = \{BG \rightarrow D; G \rightarrow J; AI \rightarrow C; CE \rightarrow H; BD \rightarrow G; JH \rightarrow A; D \rightarrow I\}$

d) $Q_4(ABCDEFGH IJ)$

$F_4 = \{BH \rightarrow I; GC \rightarrow A; I \rightarrow J; AE \rightarrow G; D \rightarrow B; I \rightarrow H\}$

-----oOo-----

Chương 6 .**CHUẨN HÓA CƠ SỞ DỮ LIỆU*****I DANG CHUẨN CỦA LƯỢC ĐỒ QUAN HỆ (normal forms for relation schemes)***

Trong thực tế, một ứng dụng cụ thể có thể được thiết kế thành nhiều lược đồ cơ sở dữ liệu khác nhau, và tất nhiên chất lượng thiết kế của các lược đồ CSDL này cũng khác nhau. Chất lượng thiết kế của một lược đồ CSDL có thể được đánh giá dựa trên nhiều tiêu chuẩn trong đó sự trùng lặp thông tin và chi phí kiểm tra các ràng buộc toàn vẹn là hai tiêu chuẩn quan trọng.

Sau đây là một số tiêu chuẩn để đánh giá độ tốt/xấu của một lược đồ quan hệ. Trước tiên ta tìm hiểu một số khái niệm liên quan:

1 Định nghĩa các dạng chuẩn**i Dạng Chuẩn Một (First Normal Form)**

Một lược đồ quan hệ Q ở dạng chuẩn 1 nếu toàn bộ các thuộc tính của mọi bộ đều mang giá trị đơn.

Ví dụ 1: Xét quan hệ

MASV	HOVATEN	KHOA	TENMONHOC	DIEMTHI
99023	NGUYENTHITHU	CONG NGHE THONG TIN	KY THUAT LAP TRINH	6
			TOAN ROI RAC	8
			CO SO DU LIEU	4
99030	LE VAN THANH	DIEN TU	VI XULY	4

Quan hệ này không đạt chuẩn 1 vì các thuộc tính TENMONHOC, DIEMTHI của bộ thứ nhất không mang giá trị đơn (chẳng hạn sinh viên NGUYEN THI THU có thuộc tính TENMONHOC là KY THUAT LAP TRINH, TOAN ROI RAC, CO SO DU LIEU).

Ta hoàn toàn có thể đưa quan hệ trên về dạng chuẩn 1 như sau:

MASV	HOVATEN	KHOA	TENMONHOC	DIEMTHI
99023	NGUYENTHITHU	CONG NGHE THONG TIN	KY THUAT LAP TRINH	6
99023	NGUYENTHITHU	CONG NGHE THONG TIN	TOAN ROI RAC	8
99023	NGUYENTHITHU	CONG NGHE THONG TIN	CO SO DU LIEU	4
99030	LE VAN THANH	DIEN TU	VI XULY	4

Chú ý rằng khi xét các dạng chuẩn, nếu ta không nói gì thêm, ta hiểu dạng chuẩn đang xét ít nhất là đạt dạng chuẩn 1.

ii Dạng Chuẩn 2 (Second Normal Form)

Một lược đồ quan hệ Q ở dạng chuẩn 2 nếu Q đạt chuẩn 1 và mọi thuộc tính không khóa của Q đều phụ thuộc đầy đủ vào khóa.

Thuật toán kiểm tra dạng chuẩn 2

Vào: lược đồ quan hệ Q , tập phụ thuộc hàm F

Ra: khẳng định Q đạt chuẩn 2 hay không đạt chuẩn 2.

Bước 1: Tìm tất cả khóa của Q

Bước 2: Với mỗi khóa K , tìm bao đóng của tất cả tập con thật sự S của K .

Bước 3: Nếu có bao đóng S^+ chứa thuộc tính không khóa thì Q không đạt chuẩn 2

Ngược lại thì Q đạt chuẩn 2

Ví dụ 2: Cho lược đồ quan hệ $Q(A, B, C, D)$ và tập phụ thuộc hàm

$F = \{AB \rightarrow C; B \rightarrow D; BC \rightarrow A\}$. Hỏi Q có đạt chuẩn 2 không?

Giải:

$$TN = \{B\}, TG = \{AC\}$$

X_i	$(TN \cup X_i)$	$(TN \cup X_i)^+$	Siêu khóa	khóa
ϕ	B	BD		
A	AB	ABCD	AB	AB
C	BC	ABCD	BC	BC
AC	ABC	ABCD	ABC	

Khóa là $K_1=AB$ và $K_2=BC$. Ta thấy $B \subset K_1$, $B \rightarrow D$, D là thuộc tính không khóa \Rightarrow thuộc tính không khóa không phụ thuộc đầy đủ vào khóa $\Rightarrow Q$ không đạt chuẩn 2.

Ví dụ 3: Quan hệ sau đạt chuẩn 2.

$Q(G, M, V, N, H, P)$ $F = \{G \rightarrow M; G \rightarrow N; G \rightarrow H; G \rightarrow P; M \rightarrow V; NHP \rightarrow M\}$

Giải:

$$TN = \{G\} \quad TG = \{M, N, H, P\}$$

X_i	$(TN \cup X_i)$	$(TN \cup X_i)^+$	Siêu khóa	khóa
ϕ	G	Q^+	G	G
M	GM	Q^+	GM	
N	GN	Q^+	GN	
MN	GMN	Q^+	GMN	
H	GH	Q^+	GH	
MH	GMH	Q^+	GMH	
NH	GNH	Q^+	GNH	
MNH	GMNH	Q^+	GMNH	
P	GP	Q^+	GP	
MP	GMP	Q^+	GMP	
NP	GNP	Q^+	GNP	
MNP	GMNP	Q^+	GMNP	
HP	GHP	Q^+	GHP	
MHP	GMHP	Q^+	GMHP	
NHP	GNHP	Q^+	GNHP	
MNHP	GMNHP	Q^+	GMNHP	

Lược đồ quan hệ Q chỉ có một khóa và khóa chỉ có một thuộc tính nên mọi thuộc tính đều phụ thuộc đầy đủ vào khóa $\Rightarrow Q$ đạt chuẩn 2

Hệ quả:

- Nếu Q đạt chuẩn 1 và tập thuộc tính không khóa của Q bằng rỗng thì Q đạt chuẩn 2
- Nếu tất cả khóa của quan hệ chỉ gồm một thuộc tính thì quan hệ đó ít nhất đạt chuẩn 2.

Ví dụ 4: $Q(A, B, C, D, E, H)$ $F = \{A \rightarrow E; C \rightarrow D; E \rightarrow DH\}$

Giải:

$$TN = \{ACB\} \quad TG = \{E\}$$

X_i	$(TN \cup X_i)$	$(TN \cup X_i)^+$	Siêu khóa	khóa
ϕ	ACB	ABCDEH	ACB	ACB
E	ACBE	ABCDEH	ACBE	

\Rightarrow khóa của Q là $K = \{ABC\}$. $C \subset K$, $C \rightarrow D$, D là thuộc tính không khóa $\Rightarrow D$ phụ thuộc không đầy đủ vào khóa nên Q không đạt chuẩn 2.

iii Dạng Chuẩn 3 (Third Normal Form)

Thuộc tính phụ thuộc bậc cao

Q là lược đồ quan hệ, X, Y là hai tập con của Q^+ , A là một thuộc tính.

Nói rằng A phụ thuộc bậc cao vào X nếu cả ba điều sau thỏa:

- + $X \rightarrow Y, Y \rightarrow A$
- + $Y \not\rightarrow X$
- + $A \notin XY$

Định nghĩa 1:

Lược đồ quan hệ Q ở dạng chuẩn 3 nếu mọi phụ thuộc hàm $X \rightarrow A \in F^+$ với $A \notin X$ đều có:

- **Hoặc X là siêu khóa**
- **Hoặc A là thuộc tính khóa**

Định nghĩa 2:

Lược đồ quan hệ Q ở dạng chuẩn 3 nếu mọi thuộc tính không khóa của Q đều không phụ thuộc bậc cao vào một khóa bất kỳ của Q

Hai định nghĩa trên là tương đương, tuy nhiên việc cài đặt thuật toán kiểm tra dạng chuẩn 3 theo định nghĩa 1 thì hiệu quả hơn nhiều vì không phải kiểm tra tính phụ thuộc bậc cao.

Ta chứng minh hai định nghĩa tương đương bằng cách:

Từ định nghĩa 1 \Rightarrow không có phụ thuộc bậc cao vào một khóa bất kỳ của Q . Thật vậy:

Giả sử có phụ thuộc bậc cao vào khóa nghĩa là có $K \rightarrow Y, Y \rightarrow A, Y \not\rightarrow K$ và $A \notin KY$. $Y \rightarrow A$ là một phụ thuộc hàm nên theo định nghĩa 1 có hai trường hợp xảy ra cho Y :

- + Y là siêu khóa $\Rightarrow Y \rightarrow K$ điều này mâu thuẫn với $Y \not\rightarrow K$.
- + Y không là siêu khóa $\Rightarrow A$ là thuộc tính khóa \Rightarrow điều này trái với giả thiết $A \notin KY$

Từ định nghĩa 2 \Rightarrow nếu $X \rightarrow A \in F^+$ với $A \notin X$ thì X là siêu khóa hoặc A là thuộc tính khóa

Nếu $X \rightarrow A \in F^+$ với $A \notin X$ có X không là siêu khóa và A không là thuộc tính khóa thì dẫn đến một số điều sau:

A không là thuộc tính khóa $\Rightarrow A \notin K$

X không là siêu khóa $\Rightarrow X \not\rightarrow K$

Tóm lại ta có $K \rightarrow X, X \rightarrow A, X \not\rightarrow K$ và $A \notin KX \Rightarrow$

A phụ thuộc bậc cao vào K điều này mâu thuẫn với định nghĩa 2.

Hệ quả 1: Nếu Q đạt chuẩn 3 thì Q đạt chuẩn 2

Hệ quả 2: Nếu Q không có thuộc tính không khóa thì Q đạt chuẩn 3.

Chứng minh:

Hệ quả 1: Giả sử Q đạt dạng chuẩn 3 và có thuộc tính không khóa A không phụ thuộc hàm đầy đủ vào khóa $K \Rightarrow K' \subset K$ sao cho $K' \rightarrow A$ như vậy ta có $K \rightarrow K', K' \rightarrow A, K' \not\rightarrow K, A \notin KK' \Rightarrow Q$ có phụ thuộc bậc cao.

Hệ quả 2: mọi phụ thuộc hàm trong Q đều có vế phải là thuộc tính khóa $\Rightarrow Q$ đạt dạng chuẩn 3

Định lý:

Q là lược đồ quan hệ

F là tập các phụ thuộc hàm có vế phải một thuộc tính.

Q đạt chuẩn 3 nếu và chỉ nếu mọi phụ thuộc hàm $X \rightarrow A \in F$ với $A \notin X$ đều có X là siêu khóa hay A là thuộc tính khóa

Chứng minh:

Q đạt dạng chuẩn 3 theo định nghĩa ta suy ra mọi phụ thuộc hàm $X \rightarrow A \in F$ với $A \notin X$ có X là siêu khóa hoặc A là thuộc tính khóa.

Ngược lại ta phải chứng minh nếu mọi phụ thuộc hàm $X \rightarrow A \in F$ với $A \notin X$ có X là siêu khóa hoặc A là thuộc tính khóa thì mọi phụ thuộc hàm $X \rightarrow A \in F^+$ với $A \notin X$ cũng có X là siêu khóa hoặc A là thuộc tính khóa

Giả sử có phụ thuộc hàm $X \rightarrow A \in F^+$ với $A \notin X$ sao cho X không là siêu khóa và A không là thuộc tính khóa sẽ dẫn đến $A \in X^+ \subseteq X \cup \{\text{các thuộc tính khóa}\}$ điều này mâu thuẫn với $A \notin K$. Trước khi chứng minh $A \in X^+ \subseteq X \cup \{\text{các thuộc tính khóa}\}$ ta có nhận xét sau:

X không là siêu khóa $\Rightarrow X^+$ cũng không là siêu khóa. Theo thuật toán tìm bao đóng, X^+ được hình thành từ các $X_i \Rightarrow$ ở mỗi bước X_i cũng không là siêu khóa.

Bước cơ sở: $X_0 = X \Rightarrow X_0 \subseteq X \cup \{\text{các thuộc tính khóa}\}$

Bước qui nạp: giả sử có $X_{i-1} \subseteq X \cup \{\text{các thuộc tính khóa}\}$. Bao đóng X_i được hình thành do có $f_j = X_j \rightarrow Y_j$ để $X_{i-1} \supseteq X_j$ và $X_i = X_{i-1} \cup Y_j \Rightarrow f_j = X_j \rightarrow Y_j$ là phụ thuộc hàm có X_j không là siêu khóa $\Rightarrow f_j = X_j \rightarrow Y_j$ là phụ thuộc hàm có Y_j là thuộc tính khóa $\Rightarrow X_i = X_{i-1} \cup Y_j \subseteq X \cup \{\text{các thuộc tính khóa}\}$

Qua chứng minh trên $\Rightarrow A \in X^+ \subseteq X \cup \{\text{các thuộc tính khóa}\} \Rightarrow A \in X \cup \{\text{các thuộc tính khóa}\} \Rightarrow A \in \{\text{các thuộc tính khóa}\}$ điều này nghịch lý với điều $A \notin K$.

Thuật toán kiểm tra dạng chuẩn 3

Vào: lược đồ quan hệ Q , tập phụ thuộc hàm F

Ra: khẳng định Q đạt chuẩn 3 hay không đạt chuẩn 3.

Bước 1: Tìm tất cả khóa của Q

Bước 2: Từ F tạo tập phụ thuộc hàm tương đương F_{tt} có vẻ phải một thuộc tính.

Bước 3: Nếu mọi phụ thuộc hàm $X \rightarrow A \in F_{tt}$ với $A \notin X$ đều có X là siêu khóa hoặc A là thuộc tính khóa thì Q đạt chuẩn 3 ngược lại Q không đạt chuẩn 3

Ví dụ 5: Cho lược đồ quan hệ $Q(A, B, C, D)$ $F = \{AB \rightarrow C; D \rightarrow B; C \rightarrow ABD\}$. Hỏi Q có đạt chuẩn 3 không?

Giải:

$TN = \emptyset$ $TG = \{ABCD\}$

X_i	$(TN \cup X_i)$	$(TN \cup X_i)^+$	Siêu khóa	khóa
ϕ	ϕ	ϕ		
A	A	A		
B	B	B		
AB	AB	ABCD	AB	AB
C	C	ABCD	C	C
AC	AC	ABCD	AC	
BC	BC	ABCD	BC	
ABC	ABC	ABCD	ABC	
D	D	BD		
AD	AD	ABCD	AD	AD
BD	BD	BD		
ABD	ABD	ABCD	ABD	
CD	CD	ABCD	CD	
ACD	ACD	ABCD	ACD	

BCD	BCD	ABCD	BCD	
ABCD	ABCD	ABCD	ABCD	

$K_1 = \{AB\}; K_2 = \{AD\}; K_3 = \{C\}$ là các khóa \Rightarrow mọi phụ thuộc hàm $X \rightarrow A \in F$ đều có A là thuộc tính khóa. Vậy Q đạt chuẩn 3

Ví dụ 6: Quan hệ sau đạt chuẩn 3. $Q(N, G, P, M) \quad F = \{NGP \rightarrow M, M \rightarrow P\}$

iv Dạng Chuẩn BC (Boyce-Codd Normal Form)

Một quan hệ Q ở dạng chuẩn BC nếu mọi phụ thuộc hàm $X \rightarrow A \in F^+$ với $A \notin X$ đều có X là siêu khóa.

Hệ quả 1: Nếu Q đạt chuẩn BC thì Q đạt chuẩn 3 (hiển nhiên do định nghĩa)

Hệ quả 2: Mỗi lược đồ có hai thuộc tính đều đạt chuẩn BC (xét phụ thuộc hàm có thể có của Q)

Định lý:

Q là lược đồ quan hệ

F là tập các phụ thuộc hàm có vẻ phải một thuộc tính.

Q đạt chuẩn BC nếu và chỉ nếu mọi phụ thuộc hàm $X \rightarrow A \in F$ với $A \notin X$ đều có X là siêu khóa

Chứng minh:

Q đạt dạng chuẩn BC theo định nghĩa ta suy ra mọi phụ thuộc hàm $X \rightarrow A \in F$ với $A \notin X$ có X là siêu khóa.

Ngược lại ta phải chứng minh nếu mọi phụ thuộc hàm $X \rightarrow A \in F$ với $A \notin X$ có X là siêu khóa thì mọi phụ thuộc hàm $Z \rightarrow B \in F^+$ với $B \notin Z$ cũng có Z là siêu khóa. Thật vậy, do $Z \rightarrow B$ không là phụ thuộc hàm hiển nhiên nên theo thuật toán tìm bao đóng phải có $X \rightarrow A \in F$ sao cho $Z \supseteq X$ (X là siêu khóa) $\Rightarrow Z$ là siêu khóa.

Thuật toán kiểm tra dạng chuẩn BC

Vào: lược đồ quan hệ Q, tập phụ thuộc hàm F

Ra: khẳng định Q đạt chuẩn BC hay không đạt chuẩn BC.

Bước 1: Tìm tất cả khóa của Q

Bước 2: Từ F tạo tập phụ thuộc hàm tương đương F_{tt} có vẻ phải một thuộc tính

Bước 3: Nếu mọi phụ thuộc hàm $X \rightarrow A \in F_{tt}$ với $A \notin X$ đều có X là siêu khóa thì Q đạt chuẩn BC ngược lại Q không đạt chuẩn BC

Ví dụ 7: $Q(A, B, C, D, E, I) \quad F = \{ACD \rightarrow EBI; CE \rightarrow AD\}$. Hỏi Q có đạt chuẩn BC không?

Giải: $TN = \{C\} \quad TG = \{ADE\}$

X_i	$(TN \cup X_i)$	$(TN \cup X_i)^+$	Siêu khóa	khóa
ϕ	C	C		
A	AC	AC		
D	CD	CD		
AD	ACD	ABCDEI	ACD	ACD
E	CE	ABCDEI	CE	CE
AE	ACE	ABCDEI	ACE	
DE	CDE	ABCDEI	CDE	
ADE	ACDE	ABCDEI	ACDE	

$F \equiv F_{1tt} = \{ACD \rightarrow E, ACD \rightarrow B, ACD \rightarrow I, CE \rightarrow A, CE \rightarrow D\}$

Mọi phụ thuộc hàm của F_{1tt} đều có vẻ trái là siêu khóa \Rightarrow Q đạt dạng chuẩn BC

Ví dụ 8: $Q(SV, MH, THAY) F = \{SV, MH \rightarrow THAY; THAY \rightarrow MH\}$

Quan hệ trên đạt chuẩn 3 nhưng không đạt chuẩn BC..

Ví dụ 9:

Chẳng hạn cho $Q(A, B, C, D)$ và $F = \{AB \rightarrow C; D \rightarrow B; C \rightarrow ABD\}$

thì Q là 3NF nhưng không là BCNF

Nếu $F = \{B \rightarrow D, A \rightarrow C, C \rightarrow ABD\}$ là 2NF nhưng không là 3NF

Thuật toán kiểm tra dạng chuẩn của một lược đồ quan hệ.

Vào: lược đồ quan hệ Q , tập phụ thuộc hàm F

Ra: khẳng định Q đạt chuẩn gì?

Bước 1: Tìm tất cả khóa của Q

Bước 2: Kiểm tra chuẩn BC nếu đúng thì Q đạt chuẩn BC, kết thúc thuật toán ngược lại qua bước 3

Bước 3: Kiểm tra chuẩn 3 nếu đúng thì Q đạt chuẩn 3, kết thúc thuật toán ngược lại qua bước 4

Bước 4: Kiểm tra chuẩn 2 nếu đúng thì Q đạt chuẩn 2, kết thúc thuật toán. ngược lại Q đạt chuẩn 1

Định nghĩa: Dạng chuẩn của một lược đồ cơ sở dữ liệu là dạng chuẩn thấp nhất trong các dạng chuẩn của các lược đồ quan hệ con.

II PHÉP TÁCH KẾT NỐI BẢO TOÀN

1 Phép tách kết nối bảo toàn thông tin (lossless-join decomposition)

Cho lược đồ quan hệ $Q(TENNCC, DIACHI, SANPHAM, DONGIA)$ có quan hệ tương ứng là r

Đặt r_1 là quan hệ có được bằng cách chiếu r lên $Q_1(TENNCC, SANPHAM, DONGIA)$,

Đặt r_2 là quan hệ có được bằng cách chiếu r lên $Q_2(TENNCC, DIACHI)$

Đặt r' là quan hệ có được bằng cách kết tự nhiên giữa r_1 và r_2 qua $TENNCC$.

chẳng hạn:

r			
TENNCC	DIACHI	SANPHAM	DONGIA
Hung	12 Nguyễn Kiệm	Gạch ống	200
Hung	12 Nguyễn Kiệm	Gạch thẻ	250
Hung	40 Nguyễn Oanh	Gạch ống	200

$r_2 = r \cdot Q_2^+$		$r_1 = r \cdot Q_1^+$		
TENNCC	DIACHI	TENNCC	SANPHAM	DONGIA
Hung	12 Nguyễn Kiệm	Hung	Gạch ống	200
Hung	40 Nguyễn Oanh	Hung	Gạch thẻ	250

$r' = r_1 \bowtie r_2$			
TENNCC	DIACHI	SANPHAM	DONGIA
Hung	12 Nguyễn Kiệm	Gạch ống	200
Hung	12 Nguyễn Kiệm	Gạch thẻ	250
Hung	40 Nguyễn Oanh	Gạch ống	200
Hung	40 Nguyễn Oanh	Gạch thẻ	250

Kết quả là $r \neq r'$ hay $r \neq r.Q_1 | \gg | r.Q_2$.

Với kết quả trên, ta nói phép tách $\rho(Q_1, Q_2)$ tách Q thành Q_1, Q_2 là tách-kết nối (phân rã) mất mát thông tin.

Nếu $r = r.Q_1 | \gg | r.Q_2$ ta nói phép tách $\rho(Q_1, Q_2)$ là tách-kết nối không mất mát thông tin (tách kết nối bảo toàn thông tin hay phân rã bảo toàn thông tin).

Vậy với điều kiện nào thì phép tách trở thành tách-kết nối không mất mát thông tin?

i Định nghĩa phép tách Q thành 2 lược đồ con

Q là lược đồ quan hệ, Q_1, Q_2 hai lược đồ con có:

$$Q_1^+ \cap Q_2^+ = X$$

$$Q_1^+ \cup Q_2^+ = Q^+$$

Nói rằng lược đồ quan hệ Q được tách thành hai lược đồ con Q_1, Q_2 theo phép tách $\rho(Q_1, Q_2)$ là phép tách kết nối không mất (hay phép tách bảo toàn thông tin) nếu với r là quan hệ bất kỳ của Q ta có:

$$r = r.Q_1 | \gg | r.Q_2$$

Tức là r được tạo nên từ phép kết nối tự nhiên của các hình chiếu của nó trên các Q_1, Q_2

ii Tính chất

Nếu Q là một lược đồ quan hệ, Q_1, Q_2 là hai lược đồ quan hệ con có

$$Q_1^+ \cap Q_2^+ = X$$

$$Q_1^+ \cup Q_2^+ = Q^+$$

$$X \rightarrow Q_2^+$$

Thì $r = r.Q_1 | \gg | r.Q_2$

Chứng minh:

$$\underline{t \in r \Rightarrow t \in r.Q_1 | \gg | r.Q_2}$$

$$t \in r \Rightarrow \exists t_1 \in r_1 \ t_1 = t.Q_1 \ \exists t_2 \in r_2 \ t_2 = t.Q_2 \ t_1.X = t_2.X = t.X$$

$$\Rightarrow t \in r.Q_1 | \gg | r.Q_2 \text{ (Theo định nghĩa)}$$

$$\underline{t \in r.Q_1 | \gg | r.Q_2 \Rightarrow t \in r}$$

$$t \in r.Q_1 | \gg | r.Q_2 \Rightarrow \exists t_1 \in r_1 \ t_1 = t.Q_1 \quad (1)$$

$$\text{mà } t_1 \in r_1 = r.Q_1 \text{ nên theo định nghĩa phép chiếu ta lại có } \exists t' \in r \ t_1 = t'.Q_1 \quad (2)$$

$$(1) \text{ và } (2) \Rightarrow t'.Q_1 = t.Q_1 \Rightarrow t'.X = t.X \Rightarrow t'.Q_2 = t.Q_2 \text{ (do } X \rightarrow Q_2)$$

$$\Rightarrow t' = t \Rightarrow t \in r$$

Ví dụ 10: cho $Q(SAIP), Q_1 = (SA), Q_2 = (SIP) \ F = \{S \rightarrow A, SI \rightarrow P\}$. Hỏi việc tách Q thành Q_1 và Q_2 có gây ra mất mát thông tin không?

Áp dụng tính chất trên, ta có

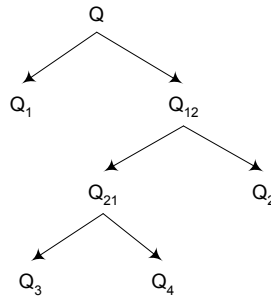
$$Q_1^+ \cap Q_2^+ = S$$

$$Q_1^+ \cup Q_2^+ = SAIP = Q^+$$

$$S \rightarrow SA = Q_1^+$$

Theo tính chất trên, với mọi quan hệ r của Q ta luôn có $r = r.Q_1 | \gg | r.Q_2$. Suy ra phép tách trên là phép tách kết nối bảo toàn thông tin.

iii Phép tách Q thành n lược đồ con



Q là một lược đồ quan hệ, F là tập phụ thuộc hàm. Q được tách thành các lược đồ con $Q_1, Q_2, Q_3, \dots, Q_n$ theo từng bước mà ở mỗi bước một lược đồ được tách thành hai lược đồ con và thỏa mãn điều kiện của tính chất bảo toàn thông tin thì với r là quan hệ bất kỳ của Q ta luôn có:

$$r = r.Q_1 | \gg | r.Q_2 | \gg | r.Q_3 \dots | \gg | r.Q_n$$

Chứng minh:

Ta chứng minh bằng phương pháp qui nạp.

Ở bước $i = 1$ thì $r = r.Q_1 | \gg | r.Q_{1m}$ đúng theo định lý bảo toàn thông tin

Giả sử biểu thức trên đúng ở bước $i = k$ nghĩa là ta có:

$$r = r.Q_1 | \gg | r.Q_2 | \gg | r.Q_3 \dots | \gg | r.Q_k | \gg | r.Q_{km} \quad (1)$$

ta phải chứng minh $r = r.Q_1 | \gg | r.Q_2 | \gg | r.Q_3 \dots | \gg | r.Q_k | \gg | r.Q_{k+1} | \gg | r.Q_{k+1m}$

Với Q_{km} được tách thành hai lược đồ con Q_{k+1} và Q_{k+1m} theo đúng điều kiện của tính chất bảo toàn thông tin nghĩa nếu s là quan hệ của Q_{km} thì $s = s.Q_{k+1} | \gg | s.Q_{k+1m} \Rightarrow$

$$r.Q_{km} = (r.Q_{km}).Q_{k+1} | \gg | (r.Q_{km}).Q_{k+1m} = r.Q_{k+1} | \gg | r.Q_{k+1m} \Rightarrow$$

$$r = r.Q_1 | \gg | r.Q_2 | \gg | r.Q_3 \dots | \gg | r.Q_k | \gg | r.Q_{k+1} | \gg | r.Q_{k+1m}$$

iv Thuật toán kiểm tra phép tách kết nối bảo toàn thông tin

(a) Thuật toán

Dữ liệu vào: lược đồ quan hệ $Q(A_1, A_2, \dots, A_n)$, tập phụ thuộc hàm F , phép tách $\rho = (Q_1, Q_2, \dots, Q_k)$.

Dữ liệu ra: kết luận phép tách ρ có phải là phép tách bảo toàn thông tin ?

1. Thiết lập bảng với $k+1$ dòng, $n+1$ cột. Cột j ứng với thuộc tính A_j ($j=1 \dots n$), hàng i ứng với lược đồ quan hệ Q_i ($i=1 \dots k$). Tại vị trí hàng i , cột j ta điền ký hiệu A_j nếu $A_j \in Q_i$, nếu không ta đặt ký hiệu b_t vào vị trí đó. (với t đầu tiên bằng 1) và sau đó tăng t lên một đơn vị.
2. Xét lần lượt các phụ thuộc hàm trong F , áp dụng cho bảng vừa mới thành lập ở trên. Giả sử xét $(X \rightarrow Y) \in F$, chúng ta tìm những hàng giống nhau ở tất cả các thuộc tính của X , nếu thấy những hàng như vậy ta sẽ làm cho các ký hiệu của hai hàng này bằng nhau ở tất cả các thuộc tính của Y . Khi làm cho 2 ký hiệu này bằng nhau, nếu một trong hai ký hiệu là a_j thì cho ký hiệu kia trở thành a_j , nếu hai ký hiệu là b_k hoặc b_l thì có thể cho chúng trở thành b_t hoặc b_l (với $t = \min(k, l)$). Bước này được tiếp tục cho các phụ thuộc hàm còn lại của F cho đến khi không còn áp dụng được nữa.
3. Xét bảng kết quả, nếu thấy trong bảng này có một hàng chứa toàn a_j ($i=1 \dots n$) thì kết luận đó là phép kết nối bảo toàn thông tin, ngược lại là phép kết nối mất mát thông tin.

Chú ý: một điều quan trọng cần phải nhớ là khi cho hai ký hiệu bằng nhau thì phải cho bằng nhau ở tất cả các xuất hiện của chúng trong bảng chứ không phải chỉ cho bằng nhau ở những ký hiệu trong phạm vi các phụ thuộc $X \rightarrow Y \in F$.

Ví dụ 11: Với Q (ABCDE)

$$Q_1 = (AD), Q_2 = (AB), Q_3 = (BE), Q_4 = (CDE), Q_5 = (AE)$$

$$F = \{A \rightarrow C, B \rightarrow C, A \rightarrow D, DE \rightarrow C, CE \rightarrow A\}$$

Kiểm tra tính bảo toàn thông tin của phép phân rã Q thành Q_1, Q_2, Q_3, Q_4, Q_5 .

Bước 1: $a_1 \ a_2 \ a_3 \ a_4 \ a_5$

	A	B	C	D	E
Q₁ (AD)	a_1			a_4	
Q₂ (AB)	a_1	a_2			
Q₃ (BE)		a_2			a_5
Q₄ (CDE)			a_3	a_4	a_5
Q₅ (AE)	a_1				a_5

Bước 2: Điền b_1, b_2, b_3, \dots

	A	B	C	D	E
Q₁ (AD)	a_1	b_1	b_2	a_4	b_3
Q₂ (AB)	a_1	a_2	b_4	b_5	b_6
Q₃ (BE)	b_7	a_2	b_8	b_9	a_5
Q₄ (CDE)	b_{10}	b_{11}	a_3	a_4	a_5
Q₅ (AE)	a_1	b_{12}	b_{13}	b_{14}	a_5

Sửa bảng giá trị để nó thỏa $A \rightarrow C$

Sửa b_4, b_{13} thành b_2

	A	B	C	D	E
Q₁ (AD)	a_1	b_1	b_2	a_4	b_3
Q₂ (AB)	a_1	a_2	b_2	b_5	b_6
Q₃ (BE)	b_7	a_2	b_8	b_9	a_5
Q₄ (CDE)	b_{10}	b_{11}	a_3	a_4	a_5
Q₅ (AE)	a_1	b_{12}	b_2	b_{14}	a_5

Sửa bảng giá trị để nó thỏa $B \rightarrow C$

Sửa b_8 thành b_2

	A	B	C	D	E
Q₁ (AD)	a_1	b_1	b_2	a_4	b_3
Q₂ (AB)	a_1	a_2	b_2	b_5	b_6
Q₃ (BE)	b_7	a_2	b_2	b_9	a_5
Q₄ (CDE)	b_{10}	b_{11}	a_3	a_4	a_5
Q₅ (AE)	a_1	b_{12}	b_2	b_{14}	a_5

Sửa bảng giá trị để nó thỏa $A \rightarrow D$

Sửa b_5, b_{14} thành a_4

	A	B	C	D	E
Q₁ (AD)	a_1	b_1	b_2	a_4	b_3
Q₂ (AB)	a_1	a_2	b_2	a_4	b_6
Q₃ (BE)	b_7	a_2	b_2	b_9	a_5
Q₄ (CDE)	b_{10}	b_{11}	a_3	a_4	a_5
Q₅ (AE)	a_1	b_{12}	b_2	a_4	a_5

Sửa bảng giá trị để nó thỏa $DE \rightarrow C$

sửa b_2 thành $a_3 \Rightarrow$ sửa tất cả b_2

thành a_3

	A	B	C	D	E
Q₁ (AD)	a_1	b_1	a_3	a_4	b_3
Q₂ (AB)	a_1	a_2	a_3	a_4	b_6
Q₃ (BE)	b_7	a_2	a_3	b_9	a_5
Q₄ (CDE)	b_{10}	b_{11}	a_3	a_4	a_5
Q₅ (AE)	a_1	b_{12}	a_3	a_4	a_5

Sửa bảng giá trị để nó thỏa $CE \rightarrow A$

Sửa b_7, b_{10} thành a_1 .

	A	B	C	D	E
Q₁ (AD)	a_1	b_1	a_3	a_4	b_3
Q₂ (AB)	a_1	a_2	a_3	a_4	b_6
Q₃ (BE)	a_1	a_2	a_3	b_9	a_5
Q₄ (CDE)	a_1	b_{11}	a_3	a_4	a_5
Q₅ (AE)	a_1	b_{12}	a_3	a_4	a_5

Lần lượt xét lại các phụ thuộc hàm trong F, nếu bảng giá trị chưa thỏa phụ thuộc hàm nào thì tiếp tục làm cho nó thỏa.

Sửa bảng giá trị để nó thỏa $A \rightarrow D$

	A	B	C	D	E
Q₁ (AD)	a_1	b_1	a_3	a_4	b_3
Q₂ (AB)	a_1	a_2	a_3	a_4	b_6
Q₃ (BE)	a_1	a_2	a_3	a_4	a_5
Q₄ (CDE)	a_1	b_{11}	a_3	a_4	a_5
Q₅ (AE)	a_1	b_{12}	a_3	a_4	a_5

Dòng thứ Q_3 (BE) của bảng chứa toàn giá trị a_j ($j=1..n$) nên phép phân rã trên là bảo toàn thông tin.

(b) Định lý

Bảng kết quả của thuật toán trên cho phép ta kết luận được tính bảo toàn hay không bảo toàn thông tin của phép tách.

Chứng minh:

Ta chứng minh nếu bảng kết quả thuật toán không có hàng chỉ chứa toàn giá trị a thì phép tách không bảo toàn thông tin. Thật vậy:

Ta xây dựng một quan hệ r có các giá trị như bảng kết quả của thuật toán, các hàng là các bộ. Quan hệ r thỏa tập phụ thuộc F vì thuật toán đã sửa các giá trị của r để nó khỏi vi phạm các phụ thuộc hàm trong F \Rightarrow r là một quan hệ của lược đồ Q. Ta tách quan hệ r thành các quan hệ r_i với $r_i = r.Q_i$ và dùng phép kết tự nhiên để kết chúng lại. Nếu:

- + $\exists k Q_k^+ \cap Q_i^+ = \emptyset \forall i \Rightarrow r_1 | \gg | r_2 \dots | \gg | r_k$ không tồn tại \Rightarrow phép tách không bảo toàn thông tin.
- + $\forall i, \exists k Q_i^+ \cap Q_k^+ = X_{ik} \neq \emptyset$ mà mỗi r_i đều có một bộ t_i chứa toàn a \Rightarrow các t_i nối được với nhau vì có cùng giá trị trên $X_{ik} \Rightarrow$ có một bộ $t \in r_1 | \gg | r_2 \dots | \gg | r_k$ có toàn giá trị a, bộ này lại không có trong r $\Rightarrow r \neq r_1 | \gg | r_2 \dots | \gg | r_k \Rightarrow$ phép tách không bảo toàn thông tin.

Ta chứng minh nếu bảng kết quả thuật toán có hàng chỉ chứa toàn giá trị a thì phép tách bảo toàn thông tin. Ta chứng minh điều này qua 2 bước:

+ **Bước 1:** chứng minh nếu $t \in r \Rightarrow t \in r_1 | \gg | r_2 \dots | \gg | r_k$. Suy ra $r \subseteq r_1 | \gg | r_2 \dots | \gg | r_k$. Giả sử $t = (a_1, \dots, a_n) \in r$. Ta tách quan hệ r thành các $r_i = r.Q_i$ với $t_i = t.Q_i$. Có hai trường hợp:

- o $\forall i, \exists k Q_i^+ \cap Q_k^+ = X_{ik} \neq \emptyset \Rightarrow$ các t_i nối được với nhau vì có cùng giá trị trên $X_{ik} \Rightarrow$ bộ $t \in r_1 | \gg | r_2 \dots | \gg | r_k \Rightarrow r \subseteq r_1 | \gg | r_2 \dots | \gg | r_k$.
- o $\exists k Q_k^+ \cap Q_i^+ = \emptyset \forall i$. Suy ra bảng kiểm tra bảo toàn thông tin ở giai đoạn chưa thỏa các phụ thuộc hàm, có dạng:

	A₁	A₂	...	A_k	A_{k+1}	...
Q₁				b _{k1}	b _{k2}	b..
Q₂				b..
...				b..
Q_k (A_k, A_{k+1}, ...)	b..	b..	b..	a _k	a _{k+1}	...

Với mọi $X \subseteq Q^+ t_k.X \neq t_i.X$ với $i \neq k$ nên khi làm bằng các giá trị theo các phụ thuộc hàm $X \rightarrow Y$ thì các giá trị b ở dòng Q_k không thay đổi còn các giá trị b ở các cột A_k, A_{k+1}, \dots không đổi thành a được. Suy ra bảng kết quả của thuật toán không bao giờ chứa dòng có toàn giá trị a. Vậy trường hợp $\exists k Q_k^+ \cap Q_i^+ = \emptyset \forall i$ không xảy ra khi bảng kiểm tra bảo toàn thông tin có một dòng toàn a.

+ **Bước 2:** chứng minh nếu $t \in r_1 | \gg | r_2 \dots | \gg | r_k \Rightarrow t \in r$. Suy ra $r_1 | \gg | r_2 \dots | \gg | r_k \subseteq r$. Giả sử $t = (a_1, \dots, a_n) \in r_1 | \gg | r_2 \dots | \gg | r_k$ theo định nghĩa suy ra $\forall i \exists t_i \in r_i$ sao cho $t.Q_i^+ = t_i$. Nhưng $r_i = r.Q_i^+ \Rightarrow \exists t_{i'} \in r$ sao cho $t_{i'}.Q_i^+ = t_i = t.Q_i^+ \forall i$. Trường hợp xấu nhất là các $t_{i'}$ là các dòng khác nhau. Trong trường hợp này, ta có thể xem $t_{i'}$ là dòng Q_i của bảng kiểm tra bảo toàn thông tin với các giá trị b xem như chưa biết. Nhưng các dòng Q_i phải thỏa các phụ thuộc hàm trong F, phép làm bằng các giá trị theo các phụ thuộc hàm đã dần dần xác định được tất cả các giá trị b của một dòng $t_{i'}$ nào đó, là dòng có toàn giá trị a. Vậy có một i' để $t_{i'} = t \Rightarrow t \in r \Rightarrow r \supseteq r_1 | \gg | r_2 \dots | \gg | r_n$ (2)

(1) và (2) $\Rightarrow r = r_1 | \gg | r_2 \dots | \gg | r_n$. Nói cách khác phép tách bảo toàn thông tin.

2 Phép tách bảo toàn phụ thuộc hàm (decompositions that preserve dependencies)

i Tập phụ thuộc hàm F_i của Q_i

Phần trên chỉ đề cập vấn đề tách một lược đồ quan hệ $Q(A_1, A_2, \dots, A_n)$ thành các lược đồ con Q_1, Q_2, \dots, Q_k còn không đề cập đến tập phụ thuộc hàm của các lược đồ con này. Nếu $Q(A_1, A_2, \dots, A_n)$ là lược đồ quan hệ, F phụ thuộc hàm, $\rho = (Q_1, Q_2, \dots, Q_k)$ là phép phân rã bảo toàn thông tin, r_i là quan hệ của Q_i thì tính chất sau thỏa:

$$+ r_i \text{ chỉ thỏa các phụ thuộc hàm } X \rightarrow Y \in F^+ \text{ với } XY \subseteq Q_i^+$$

Nói cách khác, tập phụ thuộc hàm của Q_i chính là F_i có $F_i^+ = \{X \rightarrow Y \in F^+ \mid XY \subseteq Q_i^+\}$. Ta có thể hiểu F được phân rã thành các F_1, \dots, F_k

Chứng minh tính chất trên:

Do r_i được tách từ r mà r thỏa $F^+ \Rightarrow r_i$ thỏa các phụ thuộc hàm $X \rightarrow Y \in F^+$ với $XY \subseteq Q_i^+$. Theo định nghĩa phụ thuộc hàm, đương nhiên r_i không thỏa các phụ thuộc hàm $X \rightarrow Y \in F^+$ với $XY \not\subseteq Q_i^+$. Ngoài ra r_i không thỏa bất kỳ một phụ thuộc hàm nào $X \rightarrow Y \notin F^+$. Thật vậy nếu có $X \rightarrow Y$ như vậy thì $r = r_1 \bowtie \dots \bowtie r_n$ cũng phải thỏa $X \rightarrow Y \notin F^+$. Điều này mâu thuẫn với định nghĩa của tập F^+ .

ii Định nghĩa:

Cho phân rã $\rho = (Q_1, Q_2, \dots, Q_k)$ của một lược đồ quan hệ, và một tập phụ thuộc hàm F . Hình chiếu của F trên một tập các thuộc tính Q_i^+ ký hiệu $\Pi_{Q_i}(F)$ là tập các phụ thuộc hàm $X \rightarrow Y \in F^+$ sao cho $XY \subseteq Z$.

$$\Pi_{Q_i}(F) = F_i^+ = \{X \rightarrow Y \mid X \rightarrow Y \in F^+ \text{ và } XY \subseteq Q_i\}$$

Ta nói phân rã ρ bảo toàn tập phụ thuộc hàm F nếu

$$F \equiv \cup \Pi_{Q_i}(F) \Leftrightarrow F^+ = (\cup \Pi_{Q_i}(F))^+ \text{ với } i=1..k$$

Hệ quả: $F^+ \supseteq (\cup \Pi_{Q_i}(F))^+$ với $i=1..k$

Nhận xét: từ hệ quả trên ta suy ra: để xác định phép phân rã $\rho = (Q_1, Q_2, \dots, Q_k)$ có bảo toàn phụ thuộc hàm hay không, với mỗi phụ thuộc hàm $X \rightarrow Y \in F$ ta xác định xem nó có là thành viên của tập phụ thuộc hàm $G = \cup \Pi_{Q_i}(F)$ hay không. Ta không cần xác định chiều ngược lại.

Ví dụ 12: Cho lược đồ quan hệ $Q(A, B, C)$ và $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$. Phép phân rã $\rho = (Q_1, Q_2)$ tách Q thành hai lược đồ quan hệ $Q_1(A, B)$ và $Q_2(B, C)$. Hãy tính hình chiếu của F trên Q_1^+ và Q_2^+ . Phép phân rã có bảo toàn phụ thuộc hàm F không?

Giải: về nguyên tắc ta có thể giải bài toán theo các bước dưới đây

Bước 1: Kê tất cả tập con của Q^+

	A	B	C
\emptyset	A	B	C
		AB	AC
			BC
			ABC

Bước 2: Tính bao đóng của các tập con của Q^+

$\emptyset^+ = \emptyset$	$A^+ = ABC$	$B^+ = ABC$	$C^+ = ABC$
	$AB^+ = ABC$	$AC^+ = ABC$	
		$BC^+ = ABC$	
		$ABC^+ = ABC$	

Bước 3: Tính F^+

A→B	B→A	C→A	AB→ABC	AC→B	BC→A
A→AB	B→AB	C→B	AB→C	AC→AB	BC→AB
A→C	B→C	C→AB	AB→BC	AC→BC	BC→AC
A→AC	B→AC	C→AC	AB→ABC	AC→ABC	BC→ABC
A→BC	B→BC	C→BC			
A→ABC	B→ABC	C→ABC			

Bước 4: Tính $\Pi_{Q_1}(F)$, $\Pi_{Q_2}(F)$

$$\Pi_{Q_1}(F) = F_1^+ = \{A \rightarrow B, A \rightarrow AB, B \rightarrow A, B \rightarrow AB\} \equiv \{A \rightarrow B, B \rightarrow A\} \text{ (chỉ lấy pth có vế phải 1 tt)}$$

$$\Pi_{Q_2}(F) = F_2^+ = \{B \rightarrow C, B \rightarrow BC, C \rightarrow B, C \rightarrow BC\} \equiv \{B \rightarrow C, C \rightarrow B\} \text{ (chỉ lấy pth có vế phải 1 tt)}$$

Bước 5:

$$G = \Pi_{Q_1}(F) \cup \Pi_{Q_2}(F) = \{A \rightarrow B, A \rightarrow AB, B \rightarrow A, B \rightarrow AB, B \rightarrow C, B \rightarrow BC, C \rightarrow B, C \rightarrow BC\}$$

$F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$ có $A \rightarrow B$, $B \rightarrow C$ đều là thành viên của G , còn $C \rightarrow A$ có là thành viên của G hay không ta tính C_G^+ . $C_G^+ = ABC \Rightarrow C \rightarrow A$ cũng là thành viên của G . Vậy phép phân rã trên bảo toàn phụ thuộc hàm.

Bài toán trên có thể được giải theo các bước đơn giản sau cho từng lược đồ quan hệ con:

Tính cho Q_1

Bước 1: Kê tất cả tập con của Q_1^+

	A	B
\emptyset	A	B
		AB

Bước 2: Tính bao đóng của các tập con của Q_1^+

$$\emptyset^+ = \emptyset \quad A^+ = ABC \quad B^+ = ABC$$

$$AB^+ = ABC$$

Bước 3: Tính $F_1^+ = \Pi_{Q_1}(F)$

A→B	B→A
A→AB	B→AB

Tính cho Q_2

Bước 4: Kê tất cả tập con của Q_2^+

	B	C
\emptyset	B	C
		BC

Bước 5: Tính bao đóng của các tập con của Q_2^+

$$\emptyset^+ = \emptyset \quad B^+ = ABC \quad C^+ = ABC$$

$$BC^+ = ABC$$

Bước 6: Tính $F_2^+ = \Pi_{Q_2}(F)$

B→C	C→B
B→BC	C→BC

Bước 7:

$$G = \Pi_{Q_1}(F) \cup \Pi_{Q_2}(F) = \{A \rightarrow B, A \rightarrow AB, B \rightarrow A, B \rightarrow AB, B \rightarrow C, B \rightarrow BC, C \rightarrow B, C \rightarrow BC\}$$

$F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$ có $A \rightarrow B$, $B \rightarrow C$ đều là thành viên của G còn $C \rightarrow A$ có là thành viên của G hay không ta tính C_G^+ . $C_G^+ = ABC \Rightarrow C \rightarrow A$ cũng là thành viên của G . Vậy phép phân rã trên bảo toàn phụ thuộc hàm.

iii Ý nghĩa của phân rã có bảo toàn phụ thuộc hàm

Ví dụ 13: Cho lược đồ quan hệ $Q(C, S, Z)$ và $F = \{CS \rightarrow Z, Z \rightarrow C\}$. Phép tách $\rho = (Q_1, Q_2)$ tách Q thành hai lược đồ $Q_1(S, Z)$ và $Q_2(C, Z)$. Hỏi phép tách có bảo toàn phụ thuộc hàm không?

Giải:

Q_1 có các tập thuộc tính con:

	S	Z
\emptyset	S	Z
		SZ

Bao đóng của các tập thuộc tính con Q_1^+

$$\emptyset^+ = \emptyset \quad S^+ = S \quad Z^+ = ZC$$

$$SZ^+ = CSZ$$

F_1^+ chỉ gồm các phụ thuộc hàm hiển nhiên vì tất cả các phụ thuộc hàm sau đều không thỏa:

$$Z \rightarrow C \quad SZ \rightarrow C$$

$$Z \rightarrow ZC \quad SZ \rightarrow CS$$

$$SZ \rightarrow CZ$$

$$SZ \rightarrow CSZ$$

Q_2 có các tập thuộc tính con:

	C	Z
\emptyset	C	Z
		CZ

Bao đóng của các tập thuộc tính con Q_2^+

$$\emptyset^+ = \emptyset \quad C^+ = C \quad Z^+ = ZC$$

$$CZ^+ = CZ$$

F_2^+ gồm các phụ thuộc:

$$Z \rightarrow C \quad Z \rightarrow ZC$$

$\Pi_{Q_1}(F) \cup \Pi_{Q_2}(F) = \{Z \rightarrow C, Z \rightarrow ZC\} \equiv \{Z \rightarrow C\}$ không tương đương với $F = \{CS \rightarrow Z, Z \rightarrow C\}$

Vậy phép phân rã trên không bảo toàn phụ thuộc hàm, điều này có nghĩa khi ta đưa dữ liệu vào Q_1 và Q_2 sao cho không vi phạm phụ thuộc hàm hình chiếu của nó, nhưng khi kết nối chúng lại thì dữ liệu kết quả của lược đồ quan hệ Q lại vi phạm phụ thuộc hàm $CS \rightarrow Z$

$\Pi_{Q_1}(F) = \{PTHHN\}$	$\Pi_{Q_2}(F) = \{Z \rightarrow C, Z \rightarrow ZC\}$	$F = \{CS \rightarrow Z, Z \rightarrow C\}$
$Q_1 \quad (S \quad Z)$	$Q_2 \quad (C \quad Z)$	$Q \quad (C \quad S \quad Z)$
$s_1 \quad z_1$	$c_1 \quad z_1$	$c_1 \quad s_1 \quad z_1$
$s_1 \quad z_2$	$c_1 \quad z_2$	$c_1 \quad s_1 \quad z_2$

iv Thuật toán kiểm tra bảo toàn phụ thuộc hàm

Thuật toán tìm bao đóng của tập thuộc tính X đối với $G = \cup \Pi_{Q_i}(F)$

Vào: $\rho = (Q_1, Q_2, \dots, Q_k), F, X$

Ra: X_G^+

Bước 1: Với mỗi phụ thuộc hàm $X \rightarrow Y \in F$ ta thực hiện từ bước 2 đến bước 4

Bước 2: đặt $Z' = X$

Bước 3: thế $Z' = Z' \cup ((Z' \cap Q_i^+)^+ \cap Q_i^+)$

Bước 4: nếu ở Q_i, Z' thay đổi thì thực hiện lại bước 3 cho Q_i đầu tiên

Ngược lại kết thúc thuật toán và trả về Z' (là bao đóng X_G^+)

Thuật toán kiểm tra bảo toàn phụ thuộc hàm

Vào: $\rho = (Q_1, Q_2, \dots, Q_k), F$

Ra: kết luận phép tách ρ bảo toàn hay không bảo toàn phụ thuộc hàm

Bước 1: Với mỗi phụ thuộc hàm $X \rightarrow Y \in F$ ta thực hiện từ bước 2 đến bước 3:

Bước 2: Tìm bao đóng X_G^+ với $G = \cup \Pi_{Q_i}(F)$

Bước 3: Nếu $Y \subseteq X_G^+$ thì $X \rightarrow Y \in \cup \Pi_{Q_i}(F)^+$

Bước 4: Nếu tất cả phụ thuộc $X \rightarrow Y \in F$ đều thuộc $\cup \Pi_{Q_i}(F)^+$ thì ta kết luận phân rã ρ bảo toàn phụ thuộc hàm ngược lại ρ không bảo toàn phụ thuộc hàm

Ví dụ 14: thực hiện lại ví dụ 13, nghĩa là kiểm tra phép tách có bảo toàn phụ thuộc hàm không?

Vào: $Q(C, S, Z), F = \{CS \rightarrow Z, Z \rightarrow C\}, Q_1(S, Z)$ và $Q_2(C, Z)$

Đương nhiên $Z \rightarrow C \in G = \Pi_{Q_1}(F) \cup \Pi_{Q_2}(F) \Rightarrow Z \rightarrow C \in (\Pi_{Q_1}(F) \cup \Pi_{Q_2}(F))^+$

1. $Z' = CS$

2. gán $Z' = Z' \cup ((Z' \cap Q_1^+)^+ \cap Q_1^+)$: $Z' = CS \cup (S \cap SZ) = CS$

Bước 1 và 2 có Z' không thay đổi, ta sang lược đồ Q_2 và tính tiếp Z'

3. gán $Z' = Z' \cup ((Z' \cap Q_2^+)^+ \cap Q_2^+)$: $Z' = CS \cup (C \cap CZ) = CS$

Z' không thay đổi và hết lược đồ quan hệ \Rightarrow ngưng không tính tiếp Z'

4. Vậy $CS_G^+ = CS \Rightarrow CS \rightarrow Z \notin (\Pi_{Q_1}(F) \cup \Pi_{Q_2}(F))^+$ phép phân rã không bảo toàn phụ thuộc hàm.

Ví dụ 15: thực hiện lại ví dụ 12 với nội dung kết luận phép tách ρ có bảo toàn phụ thuộc hàm không (không tính F^+)

Vào: $Q(A, B, C), F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}, Q_1(A, B)$ và $Q_2(B, C)$

Hiển nhiên $G = \Pi_{Q_1}(F) \cup \Pi_{Q_2}(F) \supseteq \{A \rightarrow B, B \rightarrow C\}$

Ta xác định $C \rightarrow A$ có thuộc $(\Pi_{Q_1}(F) \cup \Pi_{Q_2}(F))^+$

1. $Z' = C$

2. gán $Z' = Z' \cup ((Z' \cap Q_1^+)^+ \cap Q_1^+)$: $Z' = C \cup (\emptyset \cap AB) = C$

Bước 1 và 2 có Z' không thay đổi, ta sang lược đồ Q_2 và tính tiếp Z'

3. gán $Z' = Z' \cup ((Z' \cap Q_2^+)^+ \cap Q_2^+)$: $Z' = C \cup (ABC \cap BC) = BC$

Z' thay đổi \Rightarrow tính tiếp Z' bắt đầu từ lược đồ Q_1

4. gán $Z' = Z' \cup ((Z' \cap Q_1^+)^+ \cap Q_1^+)$: $Z' = BC \cup (ABC \cap AB) = ABC$

do $Z' = Q^+ \Rightarrow Z'$ sẽ không bao giờ thay đổi.

5. vậy $C_G^+ = ABC \Rightarrow C \rightarrow A \in (\Pi_{Q_1}(F) \cup \Pi_{Q_2}(F))^+$ phép phân rã bảo toàn phụ thuộc hàm.

III THIẾT KẾ CSDL BẰNG CÁCH PHÂN RÃ

1 Phân rã thành dạng chuẩn BC (hay chuẩn 3) bảo toàn thông tin

i Cách thông thường

Thuật toán phân rã Q, F thành dạng chuẩn BC (hay chuẩn 3) bảo toàn thông tin

Bước 1: Tìm tất cả khóa của Q

Bước 2: Tìm phụ thuộc hàm $X \rightarrow Y \in F$ có X không là siêu khóa và Y không chứa thuộc tính khóa.

Nếu tìm thấy thì tách Q thành Q_1 và Q_2 theo quy tắc sau:

$Q_1 = Q[XY]; F_1 \equiv \Pi_{Q_1}(F)$ tìm bao đóng của tất cả tập con của XY để suy ra $\Pi_{Q_1}(F) \Rightarrow F_1$

$Q_2 = Q[Q^+ - Y]; F_2 \equiv \Pi_{Q_2}(F)$ tìm bao đóng của tất cả tập con của $Q^+ - Y$ để suy ra $\Pi_{Q_2}(F) \Rightarrow F_2$

thực hiện thuật toán phân rã (Q_1, F_1)

thực hiện thuật toán phân rã (Q_2, F_2)

Ngược lại nếu không tìm thấy thì có hai trường hợp:

Trường hợp 1: mọi phụ thuộc hàm trong F_i đều có vế trái là siêu khóa thì Q_i đạt chuẩn BC

Trường hợp 2: nếu có phụ thuộc hàm có vế trái không là siêu khóa và vế phải là thuộc tính khóa thì Q_i đạt chuẩn 3.

Ví dụ 16: cho $Q(S, D, I, M)$ $F = \{SI \rightarrow D; SD \rightarrow M\}$ hãy phân rã Q thành các lược đồ con đạt chuẩn BC bảo toàn thông tin

Giải:

Bước 1: tìm tất cả khóa của Q

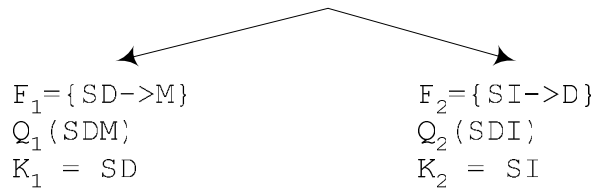
X_i	$TN \cup X_i$	$(TN \cup X_i)^+$	Siêu khóa	Khóa
\emptyset	SI	SDIM	SI	SI
D	SID	SDIM	SID	

Bước 2: phụ thuộc hàm $SD \rightarrow M \in F$ có SD không là siêu khóa.

$$F = \{SI \rightarrow D; SD \rightarrow M\}$$

$$Q(S, D, I, M)$$

$$K = SI$$



Chú ý: để tính được F_1, F_2, K_1, K_2 như hình trên, ta phải tính bao đóng của tất cả tập con của $\{SDM\}$ và $\{SDI\} \Rightarrow F_1, F_2$ rồi tìm tất cả khóa của Q_1 và Q_2 .

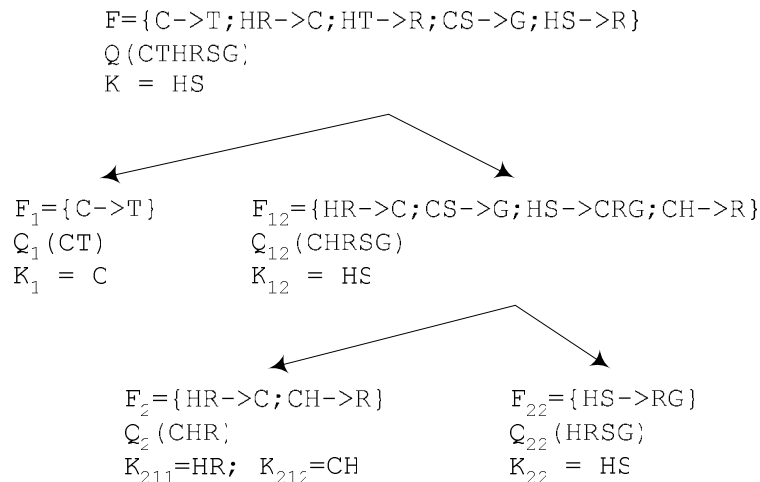
$$\begin{array}{ccccccc}
 S^+ = S & D^+ = D & M^+ = M & S^+ = S & D^+ = D & I^+ = I & \\
 SD^+ = SDM & SM^+ = SM & DM^+ = DM & SD^+ = SDM & SI^+ = SDIM & DI^+ = DI & \\
 & & SDM^+ = SDM & & SDI^+ = SDIM & &
 \end{array}$$

$$F_1^+ = \Pi_{Q_1}(F) = \{SD \rightarrow M, SD \rightarrow SM, SD \rightarrow DM, SD \rightarrow SDM\} \equiv \{SD \rightarrow M\} = F_1$$

$$F_2^+ = \Pi_{Q_2}(F) = \{SI \rightarrow D, SI \rightarrow SD, SI \rightarrow DI, SI \rightarrow SDI\} \equiv \{SI \rightarrow D\} = F_2$$

Q_1 và Q_2 đều đạt dạng chuẩn BC vì trong Q_i chỉ có phụ thuộc hàm có vế trái là khóa. F_1 được tạo thành bằng cách lấy các phụ thuộc hàm của $\Pi_{Q_1}(F)$ có vế phải một thuộc tính. Tương tự cho F_2

Ví dụ 17: cho $Q(CTHRSG)$, $F = \{C \rightarrow T; HR \rightarrow C; HT \rightarrow R; CS \rightarrow G; HS \rightarrow R\}$ hãy phân rã Q thành các lược đồ con đạt chuẩn BC bảo toàn thông tin. (giải như ví dụ trên)



Tính chất: Theo thuật toán trên, khi phân rã Q thành $Q_1 (XY)$ với $X \rightarrow Y$ và Q_2 thì tập khóa S_{Q_1} của Q_1 luôn luôn bằng với tập khóa S_{Q_2} của Q_2 .

Chứng minh

Thật vậy, K là một khóa của $Q \Rightarrow K$ là một siêu khóa của Q_2 . Giả sử có $K' \subset K$ và K' là khóa của $Q_2 \Rightarrow K' \rightarrow (Q^+ - Y)$ mà $X \rightarrow Y \Rightarrow K' \rightarrow Q^+$. Điều này mâu thuẫn với K là khóa của $Q \Rightarrow K$ là khóa của Q_2 . Ngược lại cũng đúng.

Dựa vào tính chất trên, ta cải tiến thuật toán phân rã nhằm giảm bớt khối lượng tính các phụ thuộc hàm của tập F^+

Thuật toán phân rã Q, F thành dạng chuẩn BC (hay chuẩn 3) bảo toàn thông tin

Bước 1: Tìm tập tất cả khóa S_K của Q

Bước 2: Tìm phụ thuộc hàm $X \rightarrow Y \in F$ có X không là siêu khóa và Y không chứa thuộc tính khóa. Nếu tìm thấy thì tách Q thành Q_1 và Q_2 theo quy tắc sau:

$Q_1 = Q[X \rightarrow Y]$; Tính F_1 bằng cách tính bao đóng tất cả tập con của XY

$Q_2 = Q[Q^+ - Y]$ S_K cũng là tập khóa của Q_2

thực hiện bước 1 cho Q_1

thực hiện bước 2 cho Q_2

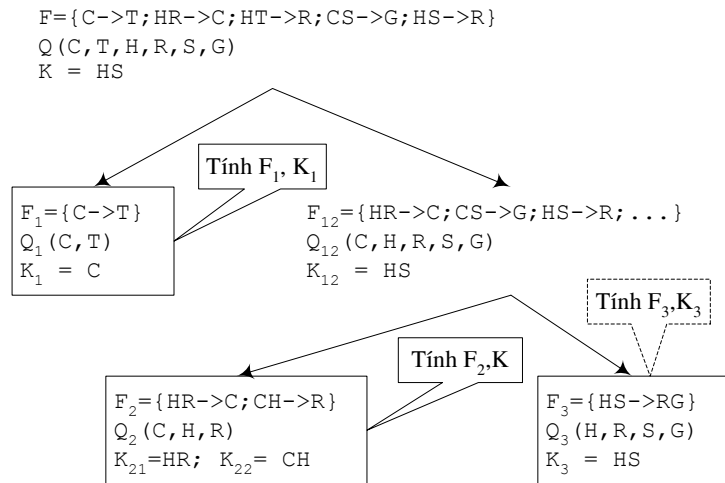
Ngược lại nếu không tìm thấy thì có hai trường hợp:

Trường hợp 1: mọi phụ thuộc hàm trong F_i đều có vế trái là siêu khóa thì Q_i đạt chuẩn BC

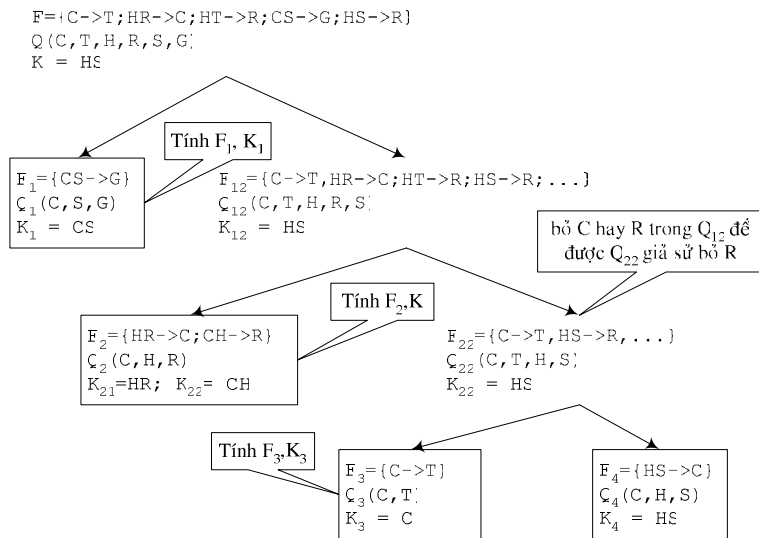
Trường hợp 2: nếu có phụ thuộc hàm có vế trái không là siêu khóa và vế phải là thuộc tính khóa thì Q_i đạt chuẩn 3.

Chú ý: Thuật toán này chỉ tiện trong trường hợp khối lượng tính toán trong việc tìm tất cả khóa của lược đồ quan hệ Q không lớn. Nói cách khác tập trung gian TG có ít thuộc tính. Ngược lại ta phải dùng thuật toán của phần tiếp theo.

Ví dụ 18: phân rã lược đồ ở ví dụ trên thành các lược đồ con ở dạng chuẩn BC bảo toàn thông tin.

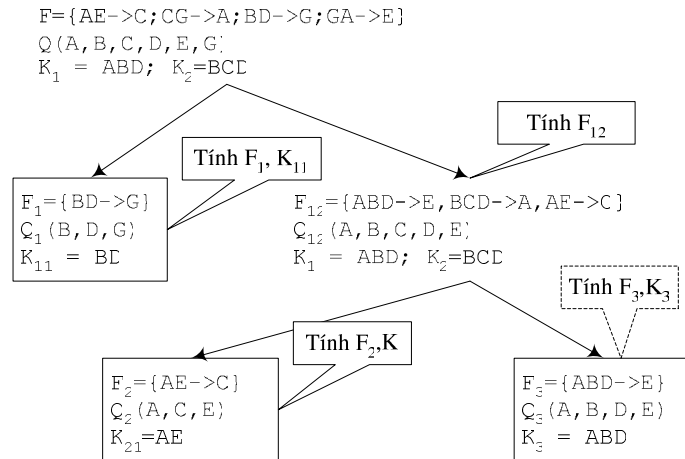


Trong F có 4 phụ thuộc hàm $C \rightarrow T, HR \rightarrow C, HT \rightarrow R, CS \rightarrow G$ làm Q không đạt dạng chuẩn 3 hay BC và phép phân rã trên đã chọn ngẫu nhiên phụ thuộc hàm $C \rightarrow T$ để phân rã thành Q_1 và tập thuộc tính của Q_{12} chính là tập thuộc tính của Q bỏ thuộc tính T . Tập phụ thuộc hàm F_{12} sẽ chứa các phụ thuộc hàm của F bỏ đi các phụ thuộc hàm có vế trái hay vế phải chứa thuộc tính T . Như vậy tùy theo cách chọn phụ thuộc hàm để phân rã thành Q_1 mà số lượng phụ thuộc hàm mang xuống Q_{12} khác nhau và chất lượng phân rã cũng khác nhau. Kết quả của phép phân rã trên chính là Q_1, Q_2, Q_3 của hình trên. Phép phân rã bảo toàn thông tin, và các lược đồ con đạt chuẩn BC nhưng phép phân rã không bảo toàn phụ thuộc hàm vì $G = F_1 \cup F_2 \cup F_3 = \{C \rightarrow T; HR \rightarrow C; CH \rightarrow R; HS \rightarrow RG\}$ không tương đương với F ($HT \rightarrow R \notin G^+$ và $\emptyset CS \rightarrow G \notin G^+$). Ta hãy xem phép phân rã sau sẽ cho kết quả tốt hơn.



Phép phân rã cũng cho kết quả phép phân rã bảo toàn thông tin, các lược đồ con Q_1, Q_2, Q_3, Q_4 đạt chuẩn BC và phép phân rã không bảo toàn phụ thuộc hàm vì $G = F_1 \cup F_2 \cup F_3 \cup F_4 = \{CS \rightarrow G; HR \rightarrow C; CH \rightarrow R; C \rightarrow T; HS \rightarrow C\}$ không tương đương với F ($HT \rightarrow R \notin G^+$). Phép phân rã này tốt hơn vì chỉ có một phụ thuộc hàm $HT \rightarrow R$ không thuộc G^+ trong khi phép phân rã trên có tới 2 phụ thuộc hàm $HT \rightarrow R$ và $CS \rightarrow G$ không thuộc G^+ . Số dĩ phép phân rã thứ 2 tốt hơn vì ở bước chọn phụ thuộc hàm để phân rã thành Q_1 phép phân rã đã chọn phụ thuộc hàm sao cho khi chiếu F xuống Q_{12} số phụ thuộc hàm mang xuống càng nhiều càng tốt.

Ví dụ 19: cho $Q(A, B, C, D, E, G)$, $F = \{AE \rightarrow C; CG \rightarrow A; BD \rightarrow G; GA \rightarrow E\}$ hãy phân rã Q thành các lược đồ con đạt chuẩn BC bảo toàn thông tin.



Nếu Q được phân rã thành:

$(Q_1(BDG), Q_2(A, B, C, D, E))$ lược đồ cơ sở dữ liệu đạt chuẩn 3

$(Q_1(BDG), Q_2(A, C, E), Q_3(A, B, D, E))$ lược đồ cơ sở dữ liệu đạt chuẩn BC

ii Bổ đề:

Nếu Q không ở dạng chuẩn BC thì có thuộc tính A, B thuộc Q^+ sao cho $(Q^+ - AB) \rightarrow A$

Chứng minh:

Q không ở dạng chuẩn BC \Rightarrow có $X \rightarrow A$ sao cho X không là siêu khóa \Rightarrow có thuộc tính B \notin XA (vì nếu không có B \notin XA thì X phải là siêu khóa) $\Rightarrow (Q^+ - AB) \supseteq X \Rightarrow (Q^+ - AB) \rightarrow A$

Nhận xét:

- + Một lược đồ Q ở dạng chuẩn BC vẫn có thể có AB sao cho $(Q^+ - AB) \rightarrow A$
- + Một lược đồ Q không có AB sao cho $(Q^+ - AB) \rightarrow A$ thì Q ở dạng chuẩn BC

iii Thuật toán

Thuật toán phân rã sau không cần tìm tất cả khóa của lược đồ quan hệ Q

Thuật Toán phân rã Q, F thành dạng chuẩn BC bảo toàn thông tin

Bước 1: $Z' = Q^+$

Bước 2: phân rã Z' theo thuật toán chi tiết để được 2 lược đồ $Z' - A$ và XA trong đó XA ở dạng chuẩn BC và $X \rightarrow A$

Nếu thuật toán chi tiết cho kết quả thì qua bước 3

Ngược lại kết thúc thuật toán

Bước 3: nhận XA là một lược đồ con của các lược đồ kết quả Q_1, \dots, Q_k

Bước 4: thực hiện phân rã $Z' - A, F$

Thuật toán chi tiết

Bước 1: nếu Z' không chứa AB sao cho $(Z' - AB) \rightarrow A$ thì báo không phân rã được.

Ngược lại qua bước 2

Bước 2: đặt $Y' = Z'$

Bước 3: nếu Y' chứa AB sao cho $(Y' - AB) \rightarrow A$ thì gán $Y' = Y' - B$ thực hiện lại bước 2

Bước 4: bước 3 cho kết quả $Y' = XA$ với XA ở dạng chuẩn BC và $X \rightarrow A$. Trả về XA

Nhận xét

Ở mỗi bước 2 của thuật toán phân rã Q, F ta thu được 2 lược đồ $Q_1^+ = Z' - A, Q_1^+ = XA$ với $Q_1^+ \cap Q_1^+ = (Z' - A) \cap XA = X$ và $X \rightarrow Q_1^+$ và Q_1 là lược đồ ở dạng chuẩn BC. Thuật toán lại tiếp tục phân rã Q_1 theo đúng cách đã làm \Rightarrow thuật toán phân rã bảo toàn thông tin và các lược đồ con Q_i đạt dạng chuẩn BC.

Thuật toán chi tiết tìm Q_1 đạt chuẩn BC sao cho Q_1^+ chứa nhiều thuộc tính nhất. Để tìm được Q_1 như vậy **thuật toán chi tiết** tìm hai thuộc tính $AB \in Q^+$ sao cho $(Q^+ - AB) \rightarrow A$. Nếu tìm thấy chứng tỏ Q chưa đạt chuẩn BC và thuật toán giảm B trong Q với hy vọng thu được lược đồ con Q_1 đạt chuẩn BC và thỏa phụ thuộc hàm $(Q^+ - AB) \rightarrow A$. Thuật toán chi tiết tiếp tục tìm và giảm cho tới khi thu được lược đồ con không có hai thuộc tính AB sao cho $(Q^+ - AB) \rightarrow A \Rightarrow Q_1$ là lược đồ con đạt chuẩn BC cần tìm.

Ví dụ 19: Cho quan hệ $Q(B, O, S, Q, I, D)$ và tập phụ thuộc hàm F

$$F = \{S \rightarrow D, \\ I \rightarrow B \\ IS \rightarrow Q \\ B \rightarrow O\}$$

Hãy phân rã Q thành các lược đồ con đạt dạng chuẩn BC và bảo toàn thông tin.

Giải

***Đặt $Z' = Q^+ = BOSQID$

Thực hiện thuật toán chi tiết

$Y' = BOSQID$

Chọn 2 thuộc tính. Tìm bao đóng của tập hợp thuộc tính còn lại. Nếu bao đóng chứa 1 trong 2 thuộc tính chọn chẳng hạn A, nghĩa là ta đã tìm được 2 thuộc tính AB sao cho $(Y' - AB) \rightarrow A$

Chọn BO: $(SQID)^+ \supset B$

Giảm O trong Y' ta được $Y' = BSQID$

Chọn BS: $(QID)^+ \supset B$

Giảm S trong Y' ta được $Y' = BQID$

Chọn BQ: $(ID)^+ \supset B$

Giảm Q trong Y' ta được $Y' = BID$

Chọn BD: $I^+ \supset B$

Giảm D trong Y' ta được $Y' = BI \Rightarrow Q_1 = (BI)$ và $F_1 = \{I \rightarrow B\}$

Để tính F_1 ta phải tính bao đóng của tất cả tập con của $\{BI\} \Rightarrow F_1$

***Giảm B trong Z' ta được $Z' = OSQID$

Đặt $Y' = OSQID$

Chọn OD: $(SQI)^+ \supset D$;

Giảm O trong Y' ta được $Y' = SQID$

chọn QD: $(SI)^+ \supset D$

giảm Q trong Y' ta được $Y' = SID$

chọn ID: $S^+ \supset D$;

giảm I trong Y' ta được $Y' = SD \Rightarrow Q_2 = (SD)$ và $F_2 = \{S \rightarrow D\}$

Để tính F_2 ta phải tính bao đóng của tất cả tập con của $\{SD\} \Rightarrow F_2$

***Giảm D trong Z' ta được $Z' = OSQI$

Đặt $Y' = OSQI$

chọn OQ: $(SI)^+ \supset Q$

giảm O trong Y' ta được $Y' = SQI \Rightarrow Q_3 = (SQI)$ và $F_3 = \{SI \rightarrow Q\}$

Ở bước trên không chọn AB để bao đóng tập hợp thuộc tính còn lại chứa A hay B

Để tính F_3 ta phải tính bao đóng của tất cả tập con của $\{SQI\} \Rightarrow F_3$

*** Giảm Q trong Z' ta được $Z' = OSI$

Đặt $Y' = OSI$

Chọn OS: $I^+ = IBO \supset O$

giảm S trong Y' ta được $Y' = OI \Rightarrow Q_4 = (OI)$ và $F_4 = \{I \rightarrow O\}$

*** Giảm O trong Z' ta được $Z' = SI \Rightarrow Q_5 = (SI)$ và $F_5 = \{P, THHN\}$

Ta có thể hiểu $Q_3 (SQI)$ là tổ hợp của 2 lược đồ con $Q_5 (SI)$ và $Q_4 (SQI)$

Vậy kết quả phân rã là:

1: $Q_1 (BI)$ $F_1 = \{I \rightarrow B\}$

2: $Q_2 (SD)$ $F_2 = \{S \rightarrow D\}$

3: $Q_3 (SQI)$ $F_3 = \{SI \rightarrow Q\}$

4: $Q_4 (OI)$ $F_4 = \{I \rightarrow O\}$

iv Chú ý

- + Nên tránh phân rã nếu lược đồ đã ở dạng chuẩn mong muốn.
- + Nên xem xét tổ hợp các lược đồ quan hệ con thành lược đồ lớn hơn nếu lược đồ lớn hơn vẫn đạt dạng chuẩn mong muốn.
- + Một kết quả phân rã bảo toàn phụ thuộc hàm sẽ có giá trị hơn kết quả phân rã không bảo toàn phụ thuộc hàm. Giữa hai kết quả phân rã đều không bảo toàn phụ thuộc hàm thì kết quả phân rã thỏa nhiều phụ thuộc hàm trong F sẽ có giá trị hơn.
- + Không có thuật toán phân rã lược đồ Q thành các lược đồ con ở dạng chuẩn BC vừa bảo toàn thông tin vừa bảo toàn phụ thuộc hàm.
- + Vẫn có lược đồ Q được phân rã thành các lược đồ con ở dạng chuẩn BC vừa bảo toàn thông tin vừa bảo toàn phụ thuộc hàm.

Ví dụ 20: cho lược đồ Q (CSZ) có $F = \{CS \rightarrow Z, Z \rightarrow C\}$. Q không thể phân rã thành các lược đồ con ở dạng chuẩn BC vừa bảo toàn thông tin vừa bảo toàn phụ thuộc hàm. Thật vậy:

TN={S} TG={CZ}

Tất cả khóa của Q là:

X_i	$TN \cup X_i$	$(TN \cup X_i)^+$	siêu khóa	khóa
\emptyset	S	S		
Z	SZ	SZC	SZ	SZ
C	SC	SZC	SC	SC
ZC	SZC	SZC	SZC	

Vậy Q đạt dạng chuẩn 3 nhưng không ở dạng chuẩn BC vì có $Z \rightarrow C$ có vế trái không là siêu khóa. Nhưng nếu ta phân rã Q thành các lược đồ con có ít hơn 3 thuộc tính thì phụ thuộc $CS \rightarrow Z$ không suy ra được từ các phụ thuộc hình chiếu.

2 Phân rã thành dạng chuẩn 3 vừa bảo toàn thông tin vừa bảo toàn phụ thuộc hàm

Thuật Toán phân rã Q, F thành dạng chuẩn 3, bảo toàn thông tin, bảo toàn phụ thuộc hàm

Dữ liệu vào: lược đồ quan hệ Q và tập phụ thuộc hàm F.

Dữ liệu ra: một phân rã sao cho mỗi lược đồ quan hệ con đều đạt chuẩn 3 vừa bảo toàn thông tin vừa bảo toàn phụ thuộc hàm.

Tìm phủ tối thiểu F_u của F

Nếu có một phụ thuộc hàm nào của F_u mà liên quan đến tất cả các thuộc tính của Q thì kết quả phân rã chính là Q (Q không thể phân rã)

Nếu có những thuộc tính của Q không nằm trong một phụ thuộc hàm nào của F_u - dù ở về phải hay về trái của F thì chúng tạo thành một lược đồ cần tìm.

Cứ mỗi phụ thuộc hàm $X \rightarrow A \in F_u$ thì XA là một lược đồ cần tìm

**Nếu có một lược đồ con chứa khóa K của Q thì kết thúc thuật toán
Ngược lại tạo một lược đồ con K**

Ví dụ 21: cho lược đồ Q (CTHRSG), $F = \{C \rightarrow T, HR \rightarrow C, TH \rightarrow R, CS \rightarrow G, HS \rightarrow R\}$. Hãy phân rã Q thành các lược đồ con đạt dạng chuẩn 3 vừa bảo toàn thông tin vừa bảo toàn phụ thuộc hàm.

Giai:

- + $F = F_{tt} = \{C \rightarrow T, HR \rightarrow C, TH \rightarrow R, CS \rightarrow G, HS \rightarrow R\}$ là phủ tối thiểu.
- + Áp dụng thuật toán trên Q được phân rã thành các lược đồ con Q_1 (CT), Q_2 (HRC), Q_3 (THR), Q_4 (CSG), Q_5 (HSR)
- + Khóa của Q

X_i	$TN \cup X_i$	$(TN \cup X_i)^+$	siêu khóa	khóa
\emptyset	HS	CTHRSG	HS	HS
C	HSC	CTHRSG	HSC	
T	HST	CTHRSG	HST	
CT	HSCT	CTHRSG	HSCT	
R	HSR	CTHRSG	HSR	
CR	HSCR	CTHRSG	HSCR	
TR	HSTR	CTHRSG	HSTR	
CTR	HSCTR	CTHRSG	HSCTR	

- + Q_5 chứa khóa của Q nên Q_1, Q_2, Q_3, Q_4, Q_5 là kết quả của phân rã.

Định lý: Thuật toán trên tạo ra một phân rã ở dạng chuẩn 3 vừa bảo toàn thông tin vừa bảo toàn phụ thuộc hàm

Chứng minh:

1. **Nếu F_{tt} có phụ thuộc hàm f_i liên quan đến tất cả thuộc tính thì Q đạt chuẩn 3.** Thật vậy:
 $f_i \in F_{tt} \Rightarrow f_i$ là phụ thuộc hàm có về phải 1 thuộc tính $\Rightarrow f_i$ có dạng $K \rightarrow A \Rightarrow K$ là siêu khóa.
 Nếu khóa của Q là $K' \subset K$ thì ta có $K' \rightarrow A \Rightarrow K \rightarrow A$ là phụ thuộc hàm có về trái dư thừa điều này mâu thuẫn với $f_i \in F_{tt}$. Vậy K là khóa của $Q \Rightarrow$ nếu Q có thuộc tính không khóa thì A là thuộc tính không khóa duy nhất của Q và mọi phụ thuộc hàm có về phải là A phải có về trái là $K \Rightarrow$ lược đồ quan hệ Q không có phụ thuộc hàm có về trái không là siêu khóa và về phải không là thuộc tính khóa $\Rightarrow Q$ đạt chuẩn 3.
2. **Nếu lược đồ Q' (W) gồm các thuộc tính không xuất hiện trong F_{tt} thì Q đạt chuẩn 3.** Thật vậy:
 V là tập con bất kỳ của W ta có $V^+ = V \Rightarrow F'$ của Q' chỉ gồm các phụ thuộc hàm hiển nhiên \Rightarrow trong F' không có phụ thuộc hàm có về trái không là siêu khóa và về phải là thuộc tính không khóa $\Rightarrow Q'$ đạt chuẩn 3.
3. **Ta chứng minh mỗi lược đồ con ở dạng chuẩn 3.** Thật vậy:

Theo thuật toán thì mỗi lược đồ con Q_i có dạng YB với $Y \rightarrow B \Rightarrow Y$ là siêu khóa. Giả sử trong Q_i có phụ thuộc hàm $X \rightarrow A$ có vế trái không là siêu khóa và vế phải không là thuộc tính khóa. Ta phân làm hai trường hợp:

Trường hợp 1: $A=B \Rightarrow X \rightarrow B \Rightarrow X \subset Y \Rightarrow Y \rightarrow B$ là phụ thuộc có vế trái dư thừa, điều này trái với $Y \rightarrow B$ là phụ thuộc hàm trong phủ tối thiểu.

Trường hợp 2: $A \neq B \Rightarrow A \in Y$ (1). Gọi K là khóa của $Q_i \Rightarrow K \subseteq Y$ (2). A là thuộc tính không khóa nên $A \notin K$ (3). (1) (2) (3) $\Rightarrow K \subset Y$ (4). K là khóa nên $K \rightarrow B \Rightarrow Y \rightarrow B$ là phụ thuộc hàm có vế trái dư thừa. Điều này trái với điều phụ thuộc hàm $Y \rightarrow B$ là phụ thuộc hàm của phủ tối thiểu F_{tt}

4. **Ta chứng minh phép phân rã bảo toàn phụ thuộc hàm.** Thật vậy:

Hiển nhiên $F_{tt} \subseteq G = \cup \Pi_{Q_i}(F_{tt}) \Rightarrow F_{tt}^+ \subseteq G^+$ (1)

Hơn nữa $F_{tt}^+ \supseteq G = \cup \Pi_{Q_i}(F_{tt}) \Rightarrow F_{tt}^{++} \supseteq G^+ \Rightarrow F_{tt}^+ \supseteq G^+$ (2)

(1) và (2) $\Rightarrow F_{tt}^+ = G^+$

5. **Ta chứng minh phép phân rã bảo toàn thông tin.** Thật vậy:

Lập bảng kiểm tra bảo toàn thông tin. Ta lần lượt đồng nhất các giá trị của bảng trên theo các phụ thuộc hàm được phát hiện ở mỗi bước của thuật toán tìm bao đóng của tập thuộc tính Q_i^+ với Q_i^+ chứa khóa K của lược đồ Q_i . Phụ thuộc hàm đầu tiên được phát hiện là $Y \rightarrow A_j \in F_{tt}$ sao cho $Q_i^+ \supseteq Y$ và $A_j \notin Q_i^+$. Ở dòng của lược đồ Q_i (YA_j) có giá trị a_j ở cột A_j nên khi làm bằng giá trị kết quả là ở cột A_j của dòng có lược đồ Q_i có thêm giá trị a_j . Tiếp tục cho các phụ thuộc hàm phát hiện tiếp theo ta sẽ có thêm các giá trị a ở các cột khác của dòng Q_i . Do Q_i chứa khóa nên các giá trị a mới thêm vào của dòng Q_i sẽ xuất hiện ở tất cả các thuộc tính của lược đồ Q_i . Suy ra hàng của lược đồ Q_i sẽ chứa toàn a là điều phải chứng minh. Để làm sáng tỏ ý tưởng của phần chứng minh này ta xét trường hợp cụ thể của ví dụ 21 :

Bước 1: ta lập bảng kiểm tra bảo toàn thông tin:

	C	T	H	R	S	G
Q1 (CT)	a_1	a_2				
Q2 (HRC)	a_1		a_3	a_4		
Q3 (THR)		a_2	a_3	a_4		
Q4 (CSG)	a_1				a_5	a_6
Q5 (HSR)			a_3	a_4	a_5	

Bước 2: Ta chứng minh dòng Q_5 của bảng trên sẽ chứa toàn giá trị a . Thật vậy: ta lần lượt đồng nhất các giá trị của bảng trên theo các phụ thuộc hàm được phát hiện theo thuật toán tìm bao đóng của $X = \{HSR\} \supseteq K$; $F = \{C \rightarrow T, HR \rightarrow C, TH \rightarrow R, CS \rightarrow G, HS \rightarrow R\}$

$X_0 = HSR$

$X_1 = HSRC$

$X_2 = HSRCT$

$X_3 = HSRCTG$

do $HR \rightarrow C$. Đồng nhất các giá trị theo phụ thuộc hàm này. Trên dòng Q_2 ở cột C chứa giá trị a nên trên dòng Q_5 sẽ có thêm giá trị a ở cột C
do $C \rightarrow T$. Đồng nhất các giá trị theo phụ thuộc hàm này.
do $CS \rightarrow G$ đồng nhất các giá trị theo phụ thuộc hàm này.

	C	T	H	R	S	G
Q1 (CT)	a_1	a_2				
Q2 (HRC)	a_1	a_2	a_3	a_4		
Q3 (THR)	a_1	a_2	a_3	a_4		
Q4 (CSG)	a_1	a_2			a_5	a_6
Q5 (HSR)	a_1	a_2	a_3	a_4	a_5	a_6

Do $X^+ = Q^+$ nên dòng Q_5 chứa toàn giá trị a

Ví dụ 22: Cho $Q(ABCDEFGHI)$, $F = \{AB \rightarrow D; EH \rightarrow G; G \rightarrow C; D \rightarrow C\}$ hãy phân rã Q thành các lược đồ con ở dạng chuẩn 3 vừa bảo toàn thông tin vừa bảo toàn phụ thuộc.

Giải:

Tìm phủ tối thiểu F_{tt} của F

$$F_{tt} = F = \{AB \rightarrow D; EH \rightarrow G; G \rightarrow C; D \rightarrow C\}$$

Áp dụng thuật toán, Q được phân rã thành lược đồ CSDL sau:

$$Q_1(ABD), Q_2(EHG), Q_3(GC), Q_4(DC)$$

Tìm khóa của Q

$$TN = \{ABEH\} \quad TG = \{GD\}$$

X_i	$TN \cup X_i$	$(TN \cup X_i)^+$	Siêu khóa	Khóa
\emptyset	ABEH	ABCDEGH	ABEH	ABEH
G	ABEHG	ABCDEGH	ABEHG	
D	ABEHD	ABCDEGH	ABEHD	
GD	ABEHGD	ABCDEGH	ABEHGD	

Q_1, Q_2, Q_3, Q_4 không chứa khóa \Rightarrow để bảo toàn thông tin ta cần có $Q_5(A, B, E, H)$. Vậy kết quả của phân rã là Q_1, Q_2, Q_3, Q_4, Q_5

IV BÀI TẬP

1/ Cho biết dạng chuẩn của các lược đồ quan hệ sau:

- a) $Q(ABCDEG)$; $F = \{A \rightarrow BC, C \rightarrow DE, E \rightarrow G\}$
 b) $Q(ABCDEFGH)$; $F = \{C \rightarrow AB, D \rightarrow E, B \rightarrow G\}$
 c) $Q(ABCDEFGH)$; $F = \{A \rightarrow BC, D \rightarrow E, H \rightarrow G\}$
 d) $Q(ABCDEG)$; $F = \{AB \rightarrow C, C \rightarrow B, ABD \rightarrow E, G \rightarrow A\}$
 e) $Q(ABCDEFGHI)$; $F = \{AC \rightarrow B, BI \rightarrow ACD, ABC \rightarrow D, H \rightarrow I, ACE \rightarrow BCG, CG \rightarrow AE\}$

2/ Kiểm tra sự bảo toàn thông tin ?

$Q(ABCDE)$ $R_1(AD)$; $R_2(AB)$; $R_3(BE)$; $R_4(CDE)$; $R_5(AE)$

$F = \{A \rightarrow C; B \rightarrow C; C \rightarrow D; DE \rightarrow C; CE \rightarrow A\}$

3/ Cho lược đồ quan hệ $Q(A, B, C, D)$ và tập phụ thuộc hàm $F = \{A \rightarrow B; B \rightarrow C; A \rightarrow D; D \rightarrow C\}$

Và một lược đồ CSDL như sau: $C = \{Q_1(AB); Q_2(AC); Q_3(BD)\}$

- a) C có bảo toàn thông tin đối với F
 b) C có bảo toàn phụ thuộc hàm ?

4/ Kiểm tra dạng chuẩn $Q(C, S, Z)$ $F = \{CS \rightarrow Z; Z \rightarrow C\}$

5/ Phân rã $Q(G, H, A, B, C, D)$ $F = \{GH \rightarrow AD; AG \rightarrow B; CD \rightarrow GH; C \rightarrow A; BH \rightarrow C\}$

6/ Cho lược đồ CSDL

Kehoach (NGAY, GIO, PHONG, MONHOC, GIAOVIEN)

$F = \{NGAY, GIO, PHONG \rightarrow MONHOC$

$MONHOC, NGAY \rightarrow GIAOVIEN$

$NGAY, GIO, PHONG \rightarrow GIAOVIEN$

$MONHOC \rightarrow GIAOVIEN\}$

- a) Xác định dạng chuẩn cao nhất của Kehoach
- b) Nếu Kehoach chưa đạt dạng chuẩn 3, hãy phân rã Kehoach thành lược đồ CSDL dạng chuẩn 3 vừa bảo toàn phụ thuộc hàm vừa bảo toàn thông tin.
- c) Nếu Kehoach chưa đạt dạng chuẩn BC, hãy phân rã KeHoach thành lược đồ CSDL dạng BC

7/ Cho lược đồ quan hệ $Q(A, B, C, D)$ và tập phụ thuộc hàm F

$$F = \{A \rightarrow B; B \rightarrow C; D \rightarrow B\} \quad C = \{Q_1(A, C, D); Q_2(B, D)\}$$

- a) Xác định các F_i (những phụ thuộc hàm F được bao trong Q_i)
- b) Lược đồ CSDL C có đạt dạng chuẩn BC? Nếu không có thể phân rã tiếp các Q_i của C để biến C thành dạng chuẩn BC?

8/ Giả sử ta có lược đồ quan hệ $Q(C, D, E, G, H, K)$ và tập phụ thuộc hàm F như sau;

$$F = \{CK \rightarrow H; C \rightarrow D; E \rightarrow C; E \rightarrow G; CK \rightarrow E\}$$

- a) Từ tập F , hãy chứng minh $EK \rightarrow DH$
- b) Tìm tất cả các khóa của Q .
- c) Xác định dạng chuẩn của Q .
- d) Hãy tìm cách phân rã Q thành một lược đồ CSDL đạt dạng chuẩn BC (hoặc dạng chuẩn 3). tìm tập phụ thuộc hàm và khóa cho mỗi lược đồ quan hệ con.

9/ Cho lược đồ quan hệ $Q(S, I, D, M)$

$$F = \{f_1: SI \rightarrow DM; f_2: SD \rightarrow M; f_3: D \rightarrow M\}$$

- a) Tính bao đóng D^+, SD^+, SI^+
- b) Tìm tất cả các khóa của Q
- c) Tìm phủ tối thiểu của F
- d) Xác định dạng chuẩn cao nhất của Q
- e) Nếu Q chưa đạt dạng chuẩn 3, hãy phân rã Q thành lược đồ CSDL dạng chuẩn 3 vừa bảo toàn phụ thuộc hàm vừa bảo toàn thông tin.
- f) Nếu Q chưa đạt dạng chuẩn BCNF, hãy phân rã Q thành lược đồ CSDL dạng BCNF
- g) Kiểm tra phép tách Q thành các lược đồ con (SID, SIM) có bảo toàn thông tin?
- h) Kiểm tra phép tách Q thành các lược đồ con (SID, SIM) có bảo toàn phụ thuộc hàm?

10/ Cho lược đồ quan hệ

$$R(W, A, Z, Y, Q, P)$$

$$R_1(A, Z);$$

$$R_2(W, Y, Q, P)$$

$$R_3(Y, Q, P, A)$$

$$F = \{W \rightarrow AYQP, A \rightarrow Z, YQP \rightarrow A\}$$

Hãy kiểm tra tính kết nối không mất thông tin.

11/ Cho lược đồ quan hệ Q(Môn, GiảngViên, Giờ giảng, Phòng, SinhViên, Hạng) với

$$F = \{M \rightarrow GV; G, P \rightarrow M; G, GV \rightarrow P; M, SV \rightarrow H; G, SV \rightarrow P\}$$

$$C = \{Q_1(M, G, P); Q_2(M, GV); Q_3(M, SV, H)\}$$

Kiểm tra xem lược đồ cơ sở dữ liệu sau đây có bảo toàn thông tin đối với F ?

12/ Kiểm Tra Dạng Chuẩn

a) Q(A, B, C, D) F={CA→D; A→B}

b) Q(S, D, I, M) F={SI→D; SD→M}

c) Q(N, G, P, M, GV) F={N, G, P→M; M→GV}

d) Q(S, N, D, T, X) F={S→N; S→D; S→T; S→X}

13/ Phân rã lược đồ thành dạng BCK

a) Q(S, D, I, M) F={S, I→D; S, D→M}

b) Q(A, B, C, D) F={A→B; B→C; D→B}

c) Q(C, S, Z) F={C, S→Z; Z→C}

14/ Phân rã lược đồ thành dạng 3NF vừa bảo toàn phụ thuộc hàm vừa bảo toàn thông tin

a) Q(A, B, C), F={A→B; A→C; B→A; C→A; B→C}

b) Q(MSCD, MSSV, CD, HG)

$$F = \{MSCD \rightarrow CD;$$

$$CD \rightarrow MSCD;$$

$$CD, MSSV \rightarrow HG;$$

$$MSCD, HG \rightarrow MSSV;$$

$$CD, HG \rightarrow MSSV;$$

$$MSCD, MSSV \rightarrow HG\}$$

c) Q(A, B, C, D) F={ AB→C; C→B}

-----oOo-----

ĐỀ THI MẪU MÔN CƠ SỞ DỮ LIỆU

(thời gian 60 phút)

ĐỀ 1**BÀI 1: (6 điểm)**

Để quản lý lịch dạy của các giáo viên và lịch học của các lớp, một trường tổ chức như sau:

Mỗi giáo viên có một mã số giáo viên (MAGV) duy nhất, mỗi MAGV xác định các thông tin như: họ và tên giáo viên (HOTEN), số điện thoại (DTGV). Mỗi giáo viên có thể dạy nhiều môn cho nhiều khoa nhưng chỉ thuộc sự quản lý hành chính của một khoa nào đó.

Mỗi môn học có một mã số môn học (MAMH) duy nhất, mỗi môn học xác định tên môn học (TENMH). Ứng với mỗi lớp thì mỗi môn học chỉ được phân cho một giáo viên.

Mỗi phòng học có một số phòng học (PHONG) duy nhất, mỗi phòng có một chức năng (CHUCNANG); chẳng hạn như phòng lý thuyết, phòng thực hành máy tính, phòng nghe nhìn, xưởng thực tập cơ khí...

Mỗi khoa có một mã khoa (MAKHOA) duy nhất, mỗi khoa xác định các thông tin như: tên khoa (TENKHOA), điện thoại khoa (DTKHOA).

Mỗi lớp có một mã lớp (MALOP) duy nhất, mỗi lớp có một tên lớp (TENLOP), sĩ số lớp (SISO). Mỗi lớp có thể học nhiều môn của nhiều khoa nhưng chỉ thuộc sự quản lý hành chính của một khoa nào đó.

Hàng tuần, mỗi giáo viên phải lập lịch báo giảng cho biết giáo viên đó sẽ dạy những lớp nào, ngày nào (NGAYDAY), môn gì, tại phòng nào, từ tiết (TUTIET) nào đến tiết (DENTIET) nào, tựa đề bài dạy (BAIDAY), những ghi chú (GHICHU) về các tiết dạy này, đây là giờ dạy lý thuyết (LYTHUYET) hay thực hành - giả sử nếu LYTHUYET=1 thì đó là giờ dạy thực hành và nếu LYTHUYET=2 thì đó là giờ lý thuyết, một ngày có 16 tiết, sáng từ tiết 1 đến tiết 6, chiều từ tiết 7 đến tiết 12, tối từ tiết 13 đến 16. Giả sử ta có lược đồ cơ sở dữ liệu để quản lý bài toán trên như sau:

Giaovien (MAGV, HOTEN, DTGV, MAKHOA)

Monhoc (MAMH, TENMH)

Phonghoc (PHONG, CHUCNANG)

Khoa (MAKHOA, TENKHOA, DTKHOA)

Lop (MALOP, TENLOP, SISO, MAKHOA)

Lichday (MAGV, MAMH, PHONG, MALOP, NGAYDAY, TUTIET, DENTIET, BAIDAY, LYTHUYET, GHICHU)

1. Hãy xác định khóa cho mỗi lược đồ quan hệ trên. (2,0 đ)

2. Phát biểu các ràng buộc toàn vẹn miền giá trị, ràng buộc toàn vẹn liên thuộc tính (1.0 đ)

3. Dựa vào lược đồ CSDL trên, hãy thực hiện các câu hỏi sau bằng SQL (3,0 đ)

a. Xem lịch báo giảng tuần từ ngày 16/09/2002 đến ngày 23/09/2002 của giáo viên có MAGV (mã giáo viên) là TH3A040. Yêu cầu: MAGV, HOTEN, TENLOP, TENMH, PHONG, NGAYDAY, TUTIET, DENTIET, BAIDAY, GHICHU

b. Xem lịch báo giảng ngày 23/09/2002 của các giáo viên có mã khoa là CNTT. Yêu cầu: MAGV, HOTEN, TENLOP, TENMH, PHONG, NGAYDAY, TUTIET, DENTIET, BAIDAY, GHICHU

c. Cho biết số lượng giáo viên (SOLUONGGV) của mỗi khoa, kết quả cần sắp xếp tăng dần theo cột tên khoa. yêu cầu: TENKHOA, SOLUONGGV (SOLUONGGV là thuộc tính tự đặt)

BÀI 2: (4 điểm)

Cho lược đồ quan hệ $Q(A, B, C, D, E, G, H, K)$ và tập phụ thuộc hàm F như sau;

$$F = \{C \rightarrow AD; E \rightarrow BH; B \rightarrow K; CE \rightarrow G\}$$

1. Kiểm tra xem các phụ thuộc hàm $E \rightarrow K; E \rightarrow G$ có thuộc tập F^+ ? (1,0đ)
2. Tìm tất cả các khóa của Q . (1,0đ)
3. Xác định dạng chuẩn của Q . (1,0đ)
4. Nếu Q chưa đạt chuẩn BC. Hãy phân rã Q thành lược đồ CSDL đạt chuẩn BC (1,0đ)

ĐÁP ÁN**BÀI 1: Câu 1:**

Giaovien (MAGV, HOTEN, DTGV, MAKHOA)

Monhoc (MAMH, TENMH)

Phonghoc (PHONG, CHUCNANG)

Khoa (MAKHOA, TENKHOA, DTKHOA)

Lop (MALOP, TENLOP, SISO, MAKHOA)

Lichday (MAGV, MAMH, PHONG, MALOP, NGAYDAY, TUTTIET, DENTIET, BAIDAY, LYTHUYET, GHICHU)

Câu 2:

$\forall t \in r_{\text{Giaovien}}$

$t.HOTEN \neq \text{NULL}$

RBTV miễn giá trị

cuối \forall

$\forall t \in r_{\text{Monhoc}}$

$t.TENMH \neq \text{NULL}$

RBTV miễn giá trị

cuối \forall

$\forall t \in r_{\text{Khoa}}$

$t.TENKHOA \neq \text{NULL}$

RBTV miễn giá trị

cuối \forall

$\forall t \in r_{\text{LOP}}$

$t.TENLOP \neq \text{NULL}$ và $t.SISO > 0$

RBTV miễn giá trị

cuối \forall

$\forall t \in r_{\text{Lichday}}$

$t.TUTTIET < t.DENTIET$ và

RBTV liên thuộc tính

$t.NGAYDAY \neq \text{NULL}$ và

RBTV miễn giá trị

$t.BAIDAY \neq \text{NULL}$ và

RBTV miễn giá trị

$(t.LYTHUYET = 1 \text{ Or } t.LYTHUYET = 2)$ và

RBTV miễn giá trị

$(t.TUTTIET \geq 1 \text{ và } t.TUTTIET \leq 16)$ và

RBTV miễn giá trị

$(t.DENTIET \geq 1 \text{ và } t.DENTIET \leq 16)$

RBTV miễn giá trị

cuối \forall

Câu 3:

- a. **SELECT** giaovien.magv,hoten,tenlop,tenmh,phong,ngayday,tutiet,đentiet,baiday,ghichu
FROM ((lichday **INNER JOIN** giaovien **ON** lichday.magv = giaovien.magv)
INNER JOIN lop **ON** Lichday.malop = lop.malop)
INNER JOIN monhoc **ON** lichday.mamh = monhoc.mamh
WHERE ngayday >=#16/09/2002# **AND** ngayday <=#23/09/2002# **AND** magv= "TH3A040"
- b. **SELECT** giaovien.magv,hoten,tenlop,tenmh,phong,ngayday,tutiet,đentiet,baiday,ghichu
FROM ((lichday **INNER JOIN** giaovien **ON** lichday.magv = giaovien.magv)
INNER JOIN lop **ON** Lichday.malop = lop.malop)
INNER JOIN monhoc **on** lichday.mamh = monhoc.mamh

WHERE ngayday = #23/09/2002# AND makh= "CNTT"

- c. **SELECT** tenkhoa,COUNT(giaovien.makhoa) **AS** soluonggv
FROM giaovien **INNER JOIN** khoa **ON** giaovien.makhoa=khoa.makhoa
GROUP BY giaovien.makhoa,tenkhoa

BÀI 2.

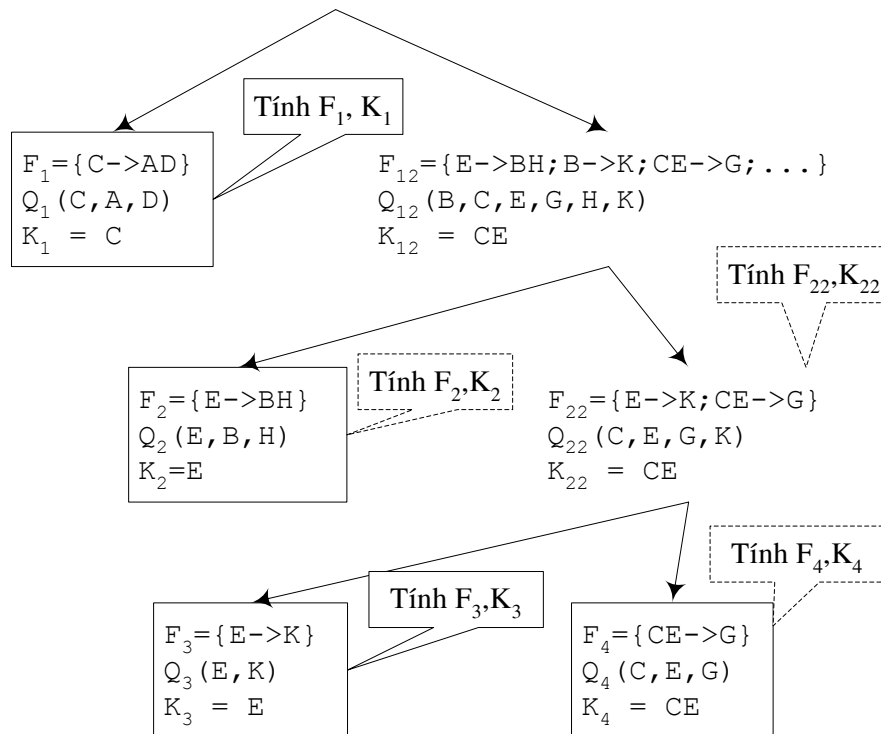
1. $E^+ = E, B, H, K \supseteq K$ nên $E \rightarrow K \in F^+$
 $E^+ = E, B, H, K \not\supseteq G$ nên $E \rightarrow G \notin F^+$
2. $TN=\{CE\}; \quad TG=\{B\}$

X_i	$TN \cup X_i$	$(TN \cup X_i)^+$	Siêu khóa	Khóa
\emptyset	CE	Q^+	CE	CE
B	CEB	Q^+	CEB	

vậy Q có khóa duy nhất là $K=\{C, E\}$

3. $C \subset K$ và $C^+=CAD \supseteq$ thuộc tính không khóa A \Rightarrow Q không đạt chuẩn 2 \Rightarrow Q đạt chuẩn 1

$F=\{C \rightarrow AD; E \rightarrow BH; B \rightarrow K; CE \rightarrow G\}$
 $Q(A, B, C, D, E, G, H, K)$
 $K = CE$



4. $Q(A, B, C, D, E, G, H, K); F=\{C \rightarrow AD; E \rightarrow BH; B \rightarrow K; CE \rightarrow G\}$
 Vậy lược đồ quan hệ Q được tách thành lược đồ cơ sở dữ liệu đạt chuẩn BC sau:

- $Q_1(C, A, D) \quad F_1=\{C \rightarrow AD\}$ đạt chuẩn BC
 $Q_2(E, B, H) \quad F_2=\{E \rightarrow BH\}$ đạt chuẩn BC
 $Q_3(E, K) \quad F_3=\{E \rightarrow K\}$ đạt chuẩn BC
 $Q_4(C, E, G) \quad F_4=\{CE \rightarrow G\}$ đạt chuẩn BC



Đề 2

Cho một lược đồ cơ sở dữ liệu C dùng để quản lý hoạt động sửa chữa, bảo trì xe của một gara xe hơi. Lược đồ cơ sở dữ liệu C gồm các lược đồ quan hệ như sau:

Q₁: Tho (MATHO, TENTHO, NHOM, NHOM_TRUONG)

Tân từ: Mỗi người thợ đều có mã số là MATHO để nhận diện. Mỗi thợ chỉ có một tên (TENTHO) và chỉ thuộc một nhóm (NHOM). Nhóm trưởng (NHOM_TRUONG) của mỗi nhóm là một trong số những người thợ của nhóm đó.
MGT (MATHO) = MGT (NHOM_TRUONG)

Q₂: Cong_viec (MACV, NOIDUNGCV)

Tân từ: Dịch vụ sửa chữa xe được chia nhỏ thành nhiều công việc để dễ dàng tính toán chi phí với khách hàng. Mỗi công việc đều có mã riêng (MACV) và nội dung của công việc được mô tả qua NOIDUNGCV.

Q₃: Hop_dong (SOHD, NGAYHD, MAKH, TENKH, DCHI, SOXE, TRIGIAHD, NG_GIAO_DK, NG_NGTHU)

Tân từ: Mỗi hợp đồng sửa xe ký kết với khách hàng đều có mã số (SOHD) để phân biệt. NGAYHD là ngày ký hợp đồng. Mỗi khách hàng có một mã số (MAKH), một tên (TENKH) và một địa chỉ (DCHI) để theo dõi công nợ. SOXE là số đăng bộ của xe đem đến sửa chữa, số này do phòng CSGT đường bộ cấp (nếu xe đổi chủ thì xem như một xe khác). Khách hàng ký hợp đồng chính là chủ xe sửa chữa. Một khách hàng có thể ký nhiều hợp đồng sửa chữa nhiều xe khác nhau hoặc hợp đồng sửa chữa nhiều lần của cùng một xe nhưng trong cùng một ngày. Những công việc sửa chữa cho một đầu xe chỉ ký hợp đồng một lần. TRIGIAHD là tổng trị giá của hợp đồng. NG_GIAO_DK là ngày dự kiến phải giao trả xe cho khách. NG_NGTHU là ngày nghiệm thu thật sự sau khi đã sửa chữa xong để thanh lý hợp đồng.

Q₄: Chitiet_HD (SOHD, MACV, TRIGIA_CV, MATHO, KHOANTHO)

Tân từ: Mỗi hợp đồng sửa xe có thể gồm nhiều công việc. MACV là mã số của từng công việc. TRIGIA_CV là chi phí về vật tư, phụ tùng, thiết bị, công thợ ... đã tính toán với khách. Mỗi công việc của hợp đồng sẽ giao cho một người thợ phụ trách (MATHO) và một người thợ có thể tham gia vào nhiều công việc của một hay nhiều hợp đồng khác nhau. KHOANTHO là số tiền giao khoán lại cho người thợ sửa chữa.

Q₅: Phieu_thu (SOPH, NGAYPH, SOHD, MAKH, HOTEN, SOTIENTHU)

Tân từ: Khách hàng (MAKH) có thể thanh toán tiền của một hợp đồng (SOHD) làm nhiều lần trước hoặc sau khi nghiệm thu (trong cùng ngày hoặc khác ngày). Mỗi lần thanh toán đều có số phiếu để phân biệt (SOPH), NGAYPH là ngày phát hành phiếu và SOTIENTHU là số tiền thanh toán. HOTEN là họ tên của người mang tiền đến thanh toán (có thể khác với tên của khách hàng đứng ra ký hợp đồng)

Câu hỏi:

- 1/ Xác định tập hợp F gồm tất cả các phụ thuộc hàm suy ra từ tân từ của các lược đồ quan hệ (không cần liệt kê các phụ thuộc hàm hiển nhiên). Xác định khóa cho từng lược đồ quan hệ.
- 2/ Mô tả tất cả các ràng buộc toàn vẹn của lược đồ cơ sở dữ liệu C. Lập bảng tầm ảnh hưởng tổng hợp.

3/ Dùng ngôn ngữ SQL để thực hiện những yêu cầu sau:

- Cho biết danh sách những người thợ hiện không tham gia vào một hợp đồng sửa chữa nào.
- Cho biết danh sách những hợp đồng hiện đã thanh lý (đã giao tra xe cho khách) nhưng chưa được thanh toán đầy đủ.
- Giả sử hôm nay là ngày 21/12/95 cho biết danh sách những hợp đồng cần phải hoàn tất trước ngày 31/12/95.
- Cho biết người thợ nào thực hiện nhiều công việc nhất.
- Cho biết người thợ nào thực hiện tổng giá trị công việc (tổng số tiền) cao nhất.

4/ Lược đồ cơ sở dữ liệu C ở dạng chuẩn mấy (cao nhất). Hãy dùng thuật toán phân rã để nâng cấp lược đồ cơ sở dữ liệu trên.

Lưu ý: Các thuộc tính đều được xem như thuộc tính đơn.

Đáp án:

Câu 1:

$F_1 = \{MATHO \rightarrow TENTHO, NHOM, NHOM_TRUONG\}$

$Q_1: Tho (MATHO, TENTHO, NHOM, NHOM_TRUONG)$

$F_2 = \{MACV \rightarrow NOIDUNGCV\}$

$Q_2: Congviec (MACV, NOIDUNGCV)$

$F_3 = \{SOHD \rightarrow NGAYHD, MAKH, SOXE, TRIGIAHD, NG_GIAO_DK, NG_NGTHU;$
 $MAKH \rightarrow TENKH, DCHI\}$

$Q_3: Hopdong (SOHD, NGAYHD, MAKH, TENKH, DCHI, SOXE, TRIGIAHD, NG_GIAO_DK, NG_NGTHU)$

$F_4 = \{SOHD, MACV \rightarrow TRIGIA_CV, MATHO, KHOANTHO\}$

$Q_4: ChiTiet_hd (SOHD, MACV, TRIGIA_CV, MATHO, KHOANTHO)$

$F_5 = \{SOPH \rightarrow NGAYPH, SOHD, HOTEN, SOTIENTHU; SOHD \rightarrow MAKH\}$

$Q_5: Phieu_thu (SOPH, NGAYPH, SOHD, MAKH, HOTEN, SOTIENTHU)$

Câu 2: mô tả tất cả các ràng buộc toàn vẹn:

$R_{11} \quad \forall t_1, t_2 \in rTho$ RBTV khóa chính
 $t_1.MATHO \neq t_2.MATHO$
 cuối \forall

$R_{12} \quad rTho[NHOM_TRUONG] \subseteq rTho[MATHO]$ RBTV khóa ngoại

$R_{13} \quad \forall t \in rTho$
 $t.TENTHO \neq NULL$ RBTV miền giá trị
 $t.NHOM \neq NULL$ RBTV miền giá trị
 cuối \forall

$R_{21} \quad \forall t_1, t_2 \in rCongviec$ RBTV khóa chính
 $t_1.MACV \neq t_2.MACV$
 cuối \forall

$R_{22} \quad \forall t \in rCongviec$ RBTV miền giá trị.
 $t.NOIDUNGCV \neq NULL$
 cuối \forall

- R₃₁ $\forall t_1, t_2 \in rHopdong$ RBTV khóa chính.
 $t_1.SOHD \neq t_2.SOHD$
- cuối \forall
- R₃₂ $\forall t \in rHopdong$ RBTV miền giá trị.
 $t.NGAYHD \neq NULL$ RBTV miền giá trị.
 $t.MAKH \neq NULL$ RBTV miền giá trị.
 $t.TENKH \neq NULL$ RBTV miền giá trị.
 $t.SOXE \neq NULL$ RBTV miền giá trị.
 $t.TRIGIAHD > 0$ RBTV miền giá trị.
 $t.NGAYHD \leq t.NG_NGTHU$ RBTV liên thuộc tính.
 $t.NG_NGTHU \leq t.NG_GIAO_DK$ RBTV liên thuộc tính.
- cuối \forall
- R₄₁ $\forall t_1, t_2 \in rChiTiet_hd$ RBTV khóa chính.
 $t_1.\{SOHD, MACV\} \neq t_2.\{SOHD, MACV\}$
- cuối \forall
- R₄₂ $rChitiet_HD[MATHO] \subseteq rTho[MATHO]$ RBTV khóa ngoại
- R₄₃ $rChitiet_HD[SOHD] \subseteq rHopdong[SOHD]$ RBTV khóa ngoại
- R₄₄ $rChitiet_HD[MACV] \subseteq rCongviec[MACV]$ RBTV khóa ngoại
- R₄₅ $\forall t \in rChiTiet_hd$ RBTV liên thuộc tính.
 $t.TRIGIA_CV > t.KHOANTHO$ RBTV miền giá trị.
 $t.KHOANTHO > 0$
- cuối \forall
- R₅₁ $\forall t_1, t_2 \in rPhieu_thu$ RBTV khóa chính
 $t_1.SOPH \neq t_2.SOPH$
- cuối \forall
- R₅₂ $rPhieu_thu[SOHD] \subseteq rHopdong[SOHD]$ RBTV khóa ngoại
- R₅₃ $\forall t \in rPhieuthu$ RBTV miền giá trị
 $t.NGAYPH \neq NULL$ RBTV miền giá trị
 $t.MAKH \neq NULL$ RBTV miền giá trị
 $t.HOTEN \neq NULL$ RBTV miền giá trị
 $t.SOTIENTHU > 0$ RBTV miền giá trị
- cuối \forall
- R₅₄ $\forall t \in rPhieuthu \exists t' \in rHopdong$ RBTV liên thuộc tính liên quan hệ.
 $t.SOHD = t'.SOHD$ và
 $t.NGAYPH \leq t'.NGAYHD$
- cuối \forall

Bảng tầm ảnh hưởng tổng hợp:

	rTho			rCongviec			rHopdong			rChiTiet_hd			rPhieu_thu		
	T	S	X	T	S	X	T	S	X	T	S	X	T	S	X
R ₁₁	+	+	-												
R ₁₂	+	+	+												
R ₁₃	+	+	-												
R ₂₁				+	+	-									
R ₂₂				+	+	-									
R ₃₁							+	+	-						
R ₃₂							+	+	-						
R ₄₁										+	+	-			

R ₄₂	-	+	+							+	+	-			
R ₄₃							-	+	+	+	+	-			
R ₄₄				-	+	+				+	+	-			
R ₄₅										+	+	-			
R ₅₁													+	+	-
R ₅₂							-	+	+				+	+	-
R ₅₃													+	+	-
R ₅₄							-	+	+				+	+	-

Câu 3:

a) **SELECT** matho,tenthò

FROM tho

WHERE matho **NOT IN**

(**SELECT** matho

FROM hop_dong **INNER JOIN** chitiet_HD **ON** hop_dong.sohd = chitiet_HD.sohd

WHERE ng_ngthu > date() **OR ISNULL**(ng_ngthu))

b) **SELECT**

sohd, ngayhd, makh, tenkh, dchi, soxe, trigiahd, ng_giao_dk, ng_ngthu

FROM hop_dong

WHERE ng_giao_dk <= **DATE**() **AND** (TRIGIAHD >

(**SELECT SUM**(SOTIENTHU) **FROM** phieu_thu

WHERE phieu_thu.sohd = hop_dong.sohd) **OR** sohd Not In (Select sohd From phieu_thu))

c) **SELECT** sohd, ngayhd, makh, tenkh, dchi, soxe, trigiahd, ng_giao_dk, ng_ngthu

FROM hop_dong

WHERE ng_giao_dk > #12/21/95# **AND** ng_giao_dk <= #12/31/95#

d) **SELECT** chitiet_hd.matho,tenthò, **COUNT**(macv) **AS** soluongcv

FROM chiTiet_hd **INNER JOIN** tho **ON** chiTiet_hd.matho = tho.matho

GROUP BY chiTiet_hd.matho,tenthò

HAVING COUNT(macv) >= **ALL** (**SELECT COUNT**(macv) **FROM** chiTiet_hd **GROUP BY** matho)

e) **SELECT** chiTiet_hd.matho,tenthò, **SUM**(trigia_cv) **AS** congtrigia_cv

FROM chiTiet_hd **INNER JOIN** tho **ON** chiTiet_hd.matho = tho.matho

GROUP BY chiTiet_hd.matho,tenthò

HAVING SUM(trigia_cv) >= **ALL** (**SELECT SUM**(trigia_cv) **FROM** chiTiet_hd **GROUP BY** matho)

Câu 4:

$F_1 = \{MATHO \rightarrow TENTHO, NHOM, NHOM_TRUONG\}$

$Q_1: Tho (MATHO, TENTHO, NHOM, NHOM_TRUONG)$

$K_1 = MATHO$

$\Rightarrow Q_1$ ở dạng chuẩn BC

$F_2 = \{MACV \rightarrow NOIDUNGCV\}$

$Q_2: Congviec (MACV, NOIDUNGCV)$

$K_2 = MACV$

$\Rightarrow Q_2$ ở dạng chuẩn BC

$F_3 = \{ \text{SOHD} \rightarrow \text{NGAYHD}, \text{MAKH}, \text{SOXE}, \text{TRIGIAHD}, \text{NG_GIAO_DK}, \text{NG_NGTHU};$
 $\text{MAKH} \rightarrow \text{TENKH}, \text{DCHI} \}$

$Q_3: \text{Hopdong}(\underline{\text{SOHD}}, \text{NGAYHD}, \text{MAKH}, \text{TENKH}, \text{DCHI}, \text{SOXE}, \text{TRIGIAHD}, \text{NG_GIAO_DK}, \text{NG_NGTHU})$
 $K_3 = \text{SOHD}$

$\Rightarrow Q_3$ ở dạng chuẩn 2

$F_4 = \{ \text{SOHD}, \text{MACV} \rightarrow \text{TRIGIA_CV}, \text{MATHO}, \text{KHOANTHO} \}$

$Q_4: \text{ChiTiet_hd}(\underline{\text{SOHD}}, \underline{\text{MACV}}, \text{TRIGIA_CV}, \text{MATHO}, \text{KHOANTHO})$

$K_4 = \{ \text{SOHD}, \text{MACV} \}$

$\Rightarrow Q_4$ ở dạng chuẩn BC

$F_5 = \{ \text{SOPH} \rightarrow \text{NGAYPH}, \text{SOHD}, \text{HOTEN}, \text{SOTIENTHU}; \text{SOHD} \rightarrow \text{MAKH} \}$

$Q_5: \text{Phieu_thu}(\underline{\text{SOPH}}, \text{NGAYPH}, \text{SOHD}, \text{MAKH}, \text{HOTEN}, \text{SOTIENTHU})$

$K_5 = \text{SOPH}$

$\Rightarrow Q_5$ ở dạng chuẩn 2

Vậy lược đồ cơ sở dữ liệu C đạt dạng chuẩn 2.

Để nâng cấp lược đồ cơ sở dữ liệu trên ta phải phân rã Q_3 và Q_5 thành:

$F_{31} = \{ \text{SOHD} \rightarrow \text{NGAYHD}, \text{MAKH}, \text{SOXE}, \text{TRIGIAHD}, \text{NG_GIAO_DK}, \text{NG_NGTHU} \}$

$Q_{31}: \text{Hopdong}(\text{SOHD}, \text{NGAYHD}, \text{MAKH}, \text{SOXE}, \text{TRIGIAHD}, \text{NG_GIAO_DK}, \text{NG_NGTHU})$

$K_{31} = \text{SOHD}$

$\Rightarrow Q_{31}$ đạt chuẩn BC

$F_{32} = \{ \text{MAKH} \rightarrow \text{TENKH}, \text{DCHI} \}$

$Q_{32}: \text{Khachhang}(\text{MAKH}, \text{TENKH}, \text{DCHI})$

$K_{32} = \text{MAKH}$

$\Rightarrow Q_{32}$ đạt chuẩn BC

$F_{51} = \{ \text{SOPH} \rightarrow \text{NGAYPH}, \text{SOHD}, \text{HOTEN}, \text{SOTIENTHU} \}$

$Q_{51}: \text{Phieu_thu}(\underline{\text{SOPH}}, \text{NGAYPH}, \text{SOHD}, \text{HOTEN}, \text{SOTIENTHU})$

$K_{51} = \text{SOPH}$

$\Rightarrow Q_{51}$ đạt chuẩn BC

$F_{52} = \{ \text{SOHD} \rightarrow \text{MAKH} \}$

$Q_{52}: \text{Hopdong}(\text{SOHD}, \text{MAKH})$

$K_{52} = \text{SOHD} \Rightarrow Q_{52}$ đạt chuẩn BC

Lược đồ Q_{52} là lược đồ con của Q_{31} nên ta loại Q_{52} khỏi lược đồ cơ sở dữ liệu C. Vậy lược đồ cơ sở dữ liệu C được phân rã thành các lược đồ $Q_1, Q_2, Q_{31}, Q_{32}, Q_4, Q_{51}$

-----oOo-----

ĐỀ 3

Cho một lược sơ đồ cơ sở dữ liệu C dùng để quản lý hoạt động kinh doanh kiều hối của một đơn vị. Lược đồ cơ sở dữ liệu C gồm các lược đồ quan hệ như sau :

Q1: Nguyen_te (LOAINT, NGAY, TIGIA, TILE_HH)

Tên từ : Mỗi nguyên tệ được nhận diện duy nhất qua LOAINT. Các loại nguyên tệ có thể là: USD, FF, DM, YEN, ...Thuộc tính TIGIA chỉ tỉ giá so với tiền đồng VN của mỗi nguyên tệ trong ngày (do Ngân hàng Ngoại thương quyết định vào đầu mỗi ngày và cố định trong ngày). Thuộc tính TILE_HH là tỉ lệ % huê hồng mà công ty được hưởng trên giá trị chi trả của mỗi nguyên tệ (tỉ lệ này cố định trong năm).

Lưu ý : Quan hệ này chỉ chứa các nguyên tệ mà công ty có chi trả kiều hối. Mỗi khi cần chi trả một loại nguyên tệ mới, công ty bắt đầu lưu tỉ giá nguyên tệ mới kể từ ngày chi trả trở đi.

Q2: Don_vi (MADV, NUOC)

Tên từ : Công ty làm đại diện cho khoảng 50 đơn vị của nước ngoài để chi trả kiều hối cho khách hàng tại VN. Mỗi đơn vị có mã riêng để phân biệt (MADV) và đặt trụ sở chính tại 1 nước (NUOC) .

Q3: Danh_sach (MADV, SODS, NGAYDS)

Tên từ : Mỗi danh sách liên quan đến 1 đơn vị, có một số thứ tự (SODS) để phân biệt với các danh sách khác của cùng đơn vị. Thuộc tính NGAYDS là ngày công ty nhận được danh sách, cũng là ngày mà đơn vị tại nước ngoài gửi danh sách cho công ty. Trong một ngày, một đơn vị tại nước ngoài chỉ gửi tối đa một danh sách.

Q4: Ctiet_ds (MADV, SODS, HOTENKH, DIACHI, LOAINT, TIENNT)

Tên từ : Mỗi danh sách chi trả của một đơn vị có thể gồm nhiều khách hàng. Giả sử rằng thuộc tính HOTENKH có thể thêm một số thông tin phụ đủ để phân biệt với các khách hàng khác trong cùng danh sách. Mỗi khách hàng chỉ có một địa chỉ (DIACHI) và nhiều khách hàng có thể có chung một địa chỉ. Trong một danh sách, mỗi khách hàng chỉ nhận một loại nguyên tệ với số tiền nguyên tệ là TIENNT

Q5: Giay_bao (SOGB, MADV, SODS, HOTENKH, NGAYGB, LAN)

Tên từ : Sau khi nhận danh sách của một đơn vị, công ty gửi giấy báo cho các khách hàng tại VN. Mỗi giấy báo có số thứ tự là SOGB (đánh số tăng dần) để phân biệt với các giấy báo khác (của cùng đơn vị hoặc khác đơn vị). Nếu sau 3 lần gửi giấy báo (mỗi lần cách nhau 1 tuần) mà khách hàng không đến nhận tiền, công ty sẽ gửi trả cho đơn vị tại nước ngoài).

Q6: Chi_tra (SOPCHI, MADV, SODS, HOTENKH, NGAYCTRA, SOTIENVN)

Tên từ : Mỗi khách hàng trong danh sách của 1 đơn vị, sau khi nhận giấy báo, sẽ đến nhận tiền đồng VN 1 lần tại công ty và mỗi phiếu chi tiền cho khách sẽ có số là SOPCHI để phân biệt với bất kỳ phiếu chi khác. Thuộc tính SOTIENVN chỉ số tiền ĐVN mà khách hàng nhận được tương đương với TIENNT ghi trong danh sách. Tỉ giá qui đổi được tính vào ngày chi trả (NGAYCTRA). Số tiền huê hồng mà công ty được hưởng sẽ được tính toán dựa trên số tiền thực chi (SOTIENVN) và tỉ lệ huê hồng của nguyên tệ.

Câu hỏi :

1. Xác định tập F gồm tất cả các phụ thuộc hàm suy ra từ tập của các lược đồ quan hệ. Xác định khóa cho từng lược đồ quan hệ.
2. Mô tả tất cả các ràng buộc toàn vẹn của lược đồ cơ sở dữ liệu. Lập bảng tầm ảnh hưởng tổng hợp.
3. Dùng ngôn ngữ SQL để thực hiện những yêu cầu sau:
 - a) Cho biết tỉ giá của các nguyên tệ trong ngày 21/12/95
 - b) Cho biết những danh sách chi trả kiều hối của các đơn vị có trụ sở chính đặt tại nước Pháp.
 - c) Cho biết những khách hàng không đến nhận tiền.
 - d) Cho tổng số tiền huê hồng mà công ty được trong khoảng thời gian từ ngày d_1 đến ngày d_2 .
 - e) Cho biết đơn vị nước ngoài có tổng số tiền chi trả (tính theo tiền đồng VN) cao nhất.
4. Lược đồ cơ sở dữ liệu C ở dạng chuẩn mấy (cao nhất) ? Hãy dùng thuật toán phân rã để nâng cấp cơ sở dữ liệu trên.

Lưu ý : Các thuộc tính có miền giá trị là ngày dương lịch xem như thuộc tính đơn.

----oOo----

Đề 4

Cho một lược đồ cơ sở dữ liệu C dùng để quản lý việc thuê mượn phòng tại một khách sạn. Lược đồ cơ sở dữ liệu C gồm các lược đồ quan hệ như sau :

Q₁: Phong (MAPH, SO_NGUOI, DACDIEM, GIA_PHONG)

Tân từ: Các phòng của khách sạn được phân biệt với nhau qua MAPH. SO_NGUOI là khả năng chứa tối đa của phòng. DACDIEM mô tả số đặc điểm của phòng. GIA_PHONG là giá cả thuê phòng trong 1 ngày.

Q₂: Tien_nghi (LOAI_TN, TEN_TN)

Tân từ: Ngoài các vật dụng tối thiểu, khách sạn có thể trang bị thêm một số tiện nghi khác cho các phòng như : điện thoại, tivi, tủ lạnh, ... LOAI_TN là mã số để phân biệt từng loại tiện nghi. TEN_TN là tên gọi của loại tiện nghi.

Q₃: Tai_san (LOAI_TN, STT, MAPH, NGÀY_TB)

Tân từ : Mỗi loại tiện nghi, khách sạn có thể mua một số lượng lớn và STT dùng để phân biệt các vật dụng trong cùng loại tiện nghi. Một vật dụng có thể được sắp xếp trang bị cho nhiều phòng khác nhau nhưng trong một ngày vật dụng chỉ trang bị cho một phòng. MAPH là phòng được trang bị và NGÀY_TB là ngày bắt đầu trang bị.

Lưu ý : Mỗi khi một vật dụng được thay đổi phòng thì cập nhật lại MAPH và NGÀY_TB của vật dụng đó.

Q₄: Thue_phong (MAPH, HOTEN, NGÀYBD, NGÀYKT, NGÀYTRA, LOAIDV, NGÀYDV, TIENDV)

Tân từ : HOTEN là họ tên của khách thuê phòng MAPH. Giả sử rằng họ tên các khách thuê phòng trong cùng một phòng trong một ngày luôn luôn khác nhau. NGÀYBD và NGÀYKT là ngày bắt đầu và ngày kết thúc (dự kiến) thuê phòng. NGÀYTRA là ngày trả thật sự. Giả sử rằng không có trường hợp khách trả phòng và thuê lại chính phòng đó trong cùng một ngày. Số tiền thuê phòng được chia đều cho số khách thuê trong cùng phòng.

Khách thuê phòng có thể sử dụng thêm các dịch vụ (gọi điện thoại đường dài, thuê xe, thủ tục hành chính, ...) LOAI_DV là mã số của loại dịch vụ sử dụng. NGÀYDV ngày dịch vụ thực hiện. TIENDV là số tiền khách thuê phải trả cho dịch vụ.

Nếu trong cùng một ngày khách thuê phòng sử dụng 1 dịch vụ nhiều lần thì tiền dịch vụ được cộng dồn lại thành một lần và tạo thành một bộ (ví dụ trong ngày gọi điện thoại 3 cuộc với số tiền phải trả lần lượt là : 5000ĐVN, 4500ĐVN, 2000ĐVN thì sẽ được tính chung một lần là 11500ĐVN). Các dịch vụ được tính riêng đối với từng khách. Nếu là dịch vụ chung cho một số khách thì sẽ tính tiền cho một đơn vị khách đại diện nào đó.

Câu hỏi :

1. Xác định tập F gồm tất cả các phụ thuộc hàm suy ra từ tân từ của các lược đồ quan hệ. Xác định các khóa cho từng lược đồ quan hệ.
2. Mô tả tất cả các ràng buộc toàn vẹn của lược đồ cơ sở dữ liệu C. Lập bảng tầm ảnh hưởng tổng hợp của các ràng buộc toàn vẹn.
3. Dùng ngôn ngữ SQL để thực hiện những yêu cầu sau :
 - a) Cho biết các thông tin của các phòng có khả năng chứa trên 3 người.
 - b) Cho biết các thông tin của các phòng có trang bị máy lạnh (LOAITN='ML')
 - c) Cho biết các thông tin của các phòng hiện nay (02/01/96) có trang bị máy lạnh.

- d) Giả sử hôm nay là ngày 02/01/96. Tính tổng số tiền phải trả (tiền thuê phòng và tiền dịch vụ) của từng khách đã thuê phòng X từ ngày 21/12/95 và trả phòng vào hôm nay .
- e) Cho biết doanh số thu được của từng phòng (không tính tiền dịch vụ)
1. Lược đồ cơ sở dữ liệu C ở dạng chuẩn mấy (cao nhất) ?. Hãy dùng thuật toán phân rã để nâng cấp lược đồ cơ sở dữ liệu C.

Lưu ý : Các thuộc tính có miền giá trị là ngày dương lịch xem như thuộc tính đơn.

-----oOo-----

ĐỀ 5

Cho một lược đồ cơ sở dữ liệu C dùng để quản lý việc cho mượn sách tại một thư viện (xem tại chỗ hoặc mang về nhà). Lược đồ cơ sở dữ liệu C gồm các lược đồ quan hệ như sau :

Q₁ : The_loai (MATL, TENTL)

Tân từ : Sách của thư viện được phân chia theo thể loại để bạn đọc dễ dàng tra cứu. MATL là mã số của từng thể loại và dùng để phân biệt giữa các thể loại. TENTL là tên gọi của thể loại.

Q₂ : Sach (MASH, TENSH, NGUYEN_TAC, TAC_GIA, MATL)

Tân từ : MASH dùng để phân biệt các quyển sách. TENSH là tên (tựa) bằng tiếng Việt của sách và NGUYEN_TAC là tên nguyên tác (tiếng Việt hoặc tiếng nước ngoài). TAC_GIA là tên tác giả (hay nhóm các tác giả) của sách. Nếu sách có nhiều tập hay nhiều bản thì cũng xem như các đầu sách khác nhau và có mã số khác nhau. MATL là mã thể loại của sách.

Q₃ : phieu_muon (MADG, TENDG, DCHI, NGAYCAP, MASH, NGAYMUON, NGAYTRA, TAI_CHO)

Tân từ : Mỗi độc giả chỉ có một phiếu mượn sách với mã số là MADG để phân biệt với các độc giả khác. Các thuộc tính TENDG, DCHI là tên và địa chỉ của độc giả. NGAYCAP là ngày cấp thẻ cho độc giả. MASH là mã số của sách mượn. Giả sử không có trường hợp mượn rồi trả lại cùng 1 quyển sách trong cùng 1 ngày. Nếu sách mượn đọc tại chỗ thì thuộc tính TAI_CHO có giá trị True và NGAYMUON=NGAYTRA. Nếu sách mượn về nhà thì thuộc tính TAI_CHO có giá trị False và NGAYTRA sẽ có giá trị trống cho đến khi sách được mang trả lại cho thư viện. Mỗi độc giả chỉ được giữ tại nhà tối đa 3 quyển sách và mỗi quyển sách chỉ được giữ tại nhà tối đa 30 ngày (không cần lưu ý đến biện pháp xử lý nếu khách vi phạm nội qui)

Q₄ : Le_phi (MADG, NAM, NGAY_NOP, SOTIEN)

Tân từ: Độc giả phải đóng lệ phí hằng năm (NAM) để gia hạn thẻ mới được mượn sách. NGAY_NOP, SOTIEN là ngày và số tiền nộp lệ phí cho NAM.

Câu hỏi :

1. Xác định tập F gồm tất cả các phụ thuộc hàm suy ra từ tân từ của các lược đồ quan hệ. Xác định các khóa cho từng lược đồ quan hệ.
2. Mô tả tất cả các ràng buộc toàn vẹn của lược đồ cơ sở dữ liệu C. Lập bảng tầm ảnh hưởng tổng hợp của các ràng buộc toàn vẹn.
3. Dùng ngôn ngữ SQL để thực hiện những yêu cầu sau :
 - a) Cho biết danh sách độc giả và những quyển sách mượn quá 20 ngày (kể từ ngày 02/01/96).
 - b) Cho biết những quyển sách có tên thể loại là “Tin học” và có sự tham gia biên soạn của tác giả “X”.
 - c) Cho biết tổng số lần mượn của từng quyển sách.
 - d) Cho biết tổng số lần mượn của từng thể loại sách.
 - e) Cho biết thể loại sách nào được mượn nhiều nhất.
4. Lược đồ cơ sở dữ liệu C ở dạng chuẩn mấy (cao nhất) ? Hãy dùng thuật toán phân rã để nâng cấp lược đồ cơ sở dữ liệu C.

Lưu ý : Các thuộc tính có miền giá trị là ngày dương lịch xem như thuộc tính đơn.

TÀI LIỆU THAM KHẢO

- [1].Nhập môn cơ sở dữ liệu
Đồng Thị Bích Thủy - Nguyễn An Tế
- [2].Cơ sở dữ liệu – lý thuyết và thực hành
Nguyễn Bá Tường
- [3].Cơ sở dữ liệu
Đỗ Trung Tuấn
- [4].Mô hình dữ liệu và ngôn ngữ truy vấn
JEFFREY D. ULLMAN
- [5].Lý thuyết thiết kế cơ sở dữ liệu
JEFFREY D. ULLMAN
- [6].The Theory Of Relational Databases
DAVID MAIER



TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP/HCM
TRUNG TÂM CÔNG NGHỆ THÔNG TIN
-000-

PHẦN 2 SQL SERVER

MỤC LỤC

BÀI 1:	GIỚI THIỆU MICROSOFT SQL SERVER 2000.....	4
1.1	Tổng quan về SQL Server 2000.....	4
1.2	Các thành phần của SQL Server 2000.....	7
1.3	Kiến trúc của CSDL quan hệ (relational Database Architecture).....	7
1.4	Giới thiệu về Transact-Sql.....	11
BÀI 2:	CÀI ĐẶT SQL SERVER 2000.....	12
2.1	Lập kế hoạch cài đặt SQL SERVER 2000.....	12
2.2	Quyết định những chọn lựa cấu hình cài đặt SQL SERVER 2000.....	12
2.2.1	Xác định tài khoản người dùng (User account) thích hợp cho SQL Server Service và SQL Server Agent Service.....	12
2.2.2	Chọn một chế độ xác thực (Authentication Mode).....	13
2.2.3	Xác định Collation.....	13
2.2.4	Chọn Network Libraries.....	13
2.2.5	Xác định client Licensing Mode.....	13
2.3	Các bước cài đặt SQL Server 2000.....	13
2.4	Tạo tập tin cài đặt không tham dự (unattended) và cài đặt từ xa (Remote) SQL Server 2000.....	20
2.5	Kết quả của việc cài đặt SQL SERVER 2000.....	21
BÀI 3:	CÁC CÔNG CỤ CỦA SQL SERVER.....	22
3.1	SQL ENTERPRISE MANAGER.....	22
3.2	SQL SERVER SERVICES MANAGER.....	22
3.2.1	Các dịch vụ của SQL Server.....	23
3.2.2	Khởi động, tạm ngưng, dừng các dịch vụ của SQL Server.....	24
3.3	OSQL.....	24
3.4	SQL QUERY ANALYZER.....	25
3.4.1	Giới thiệu.....	25
3.4.2	Khởi động Query Analyzer.....	26
3.4.3	Thành phần chính của Query Analyzer.....	26
3.4.4	Một vài phím nóng dùng trong Query Analyzer.....	28
BÀI 4:	LÀM VIỆC VỚI CƠ SỞ DỮ LIỆU SQL SERVER.....	29
4.1	Thiết kế một cơ sở dữ liệu.....	29
4.2	Cơ sở dữ liệu của SQL SERVER 2000.....	31
4.3	Tạo, hiệu chỉnh cơ sở dữ liệu SQL SERVER.....	34
4.3.1	Giới thiệu.....	34
4.3.2	Tạo cơ sở dữ liệu.....	34
4.3.3	Thao tác trên cơ sở dữ liệu của SQL Server.....	37
BÀI 5:	Kiểu DỮ LIỆU – LÀM VIỆC VỚI BẢNG.....	41
5.1	Kiểu dữ liệu (data type).....	41
5.1.1	System-Supplied Datatype.....	41
5.1.2	User-defined datatype.....	42
5.2	Làm việc với bảng của SQL Server.....	43
5.2.1	Tạo một bảng mới.....	43
5.2.2	Hiệu chỉnh bảng.....	45
5.2.3	Xóa bảng khỏi cơ sở dữ liệu.....	46
5.3	Bảng tạm (Temporary Tables).....	46
BÀI 6:	TOÀN VỆN DỮ LIỆU.....	48
6.1	Giới thiệu toàn vẹn dữ liệu (data Integrity).....	48

6.2	Tìm hiểu các toàn vẹn dữ liệu.....	48
6.2.1	Định nghĩa NULL/NOT NULL.....	48
6.2.2	Giá trị mặc định (Default Values).....	49
6.2.3	Thuộc tính Identity:.....	52
6.2.4	Check.....	53
6.2.5	Primary key Constraint.....	54
6.2.6	Unique Constraints.....	57
6.2.7	Foreign Key Constraint.....	58
BÀI 7:	TRUY XUẤT CƠ SỞ DỮ LIỆU CỦA SQL SERVER.....	61
7.1	Câu lệnh SELECT.....	61
7.2	Sử dụng JOINS để truy xuất dữ liệu.....	66
7.3	Dùng Sub-Queries.....	68
7.4	Hiệu chỉnh dữ liệu trong cơ sở dữ liệu của SQL SERVER.....	69
7.4.1	Chèn (INSERT) dữ liệu vào CSDL.....	69
7.4.2	Cập nhật (UPDATE) dữ liệu vào CSDL.....	71
7.4.3	Xóa dữ liệu trong cơ sở dữ liệu.....	71
BÀI 8:	KHUNG NHÌN - VIEW.....	73
8.1	Giới thiệu về View.....	73
8.2	Tạo, hiệu chỉnh, xóa View.....	73
8.3	Tạo Partition view.....	74
8.4	Truy xuất dữ liệu thông qua View.....	75
8.4.1	Xem dữ liệu thông qua view.....	75
8.4.2	Hiệu chỉnh dữ liệu thông qua View.....	75
BÀI 9:	CHUYỂN ĐỔI DỮ LIỆU.....	77
9.1	Khái niệm chuyển đổi và biến đổi dữ liệu.....	77
9.1.1	Import/Export dữ liệu.....	77
9.1.2	Biến đổi dữ liệu (Data Transformations).....	77
9.1.3	Các công cụ chuyển đổi dữ liệu (Data transfer tools).....	77
9.2	Dịch vụ chuyển đổi dữ liệu DTS (Data Transformation Services - DTS).....	78
9.2.1	DTS Package.....	78
9.2.2	DTS Connections.....	78
9.2.3	DTS Tasks.....	78
9.2.4	DTS Package Workflow.....	80
9.2.5	DTS Package Storage.....	80
9.3	Thực hiện việc biến đổi và chuyển đổi dữ liệu bằng công cụ đồ họa DTS.....	80
9.3.1	DTS Import/Export Wizard.....	80
9.3.2	DTS Designer.....	81
9.4	Dùng BULK COPY (BCP) và BULK INSERT.....	88
BÀI 10:	CƠ BẢN VỀ LẬP TRÌNH BẰNG TRANSACT- SQL.....	91
10.1	Khái niệm cơ bản.....	91
10.1.1	Định danh -IDENTIFIERS.....	91
10.1.2	Tham chiếu đến các đối tượng trong SQL Server.....	91
10.1.3	Kiểu dữ liệu (DATA TYPE).....	92
10.1.4	Batch.....	92
10.1.5	Kịch bản - SCRIPT.....	92
10.2	Biến (VARIABLES).....	92
10.3	Cấu trúc điều khiển.....	97
10.3.1	Khối BEGIN ... END.....	97
10.3.2	Phát biểu PRINT.....	97
10.3.3	Cấu trúc điều khiển IF ... ELSE.....	97

10.3.4	Biểu thức CASE.....	99
10.3.5	Cấu trúc vòng lặp WHILE	100
10.3.6	Lệnh RETURN	101
10.3.7	Lệnh WAITFOR	101
10.3.8	Lệnh RAISERROR.....	101
BÀI 11:	PROCEDURES, FUNCTIONS.....	104
11.1	STORED PROCEDURES.....	104
11.1.1	Giới thiệu Stored procedures.....	104
11.1.2	Tạo, thực thi, hiệu chỉnh, xóa stored procedures.....	104
11.1.3	Tham số và biến trong Stored procedures.....	106
11.2	FUNCTIONS.....	110
11.2.1	Scalar Functions.....	111
11.2.2	Table-valued Functions.....	111
BÀI 12:	TRANSACTIONS – LOCK.....	114
12.1	TRANSACTIONS.....	114
12.2	LOCK	116
BÀI 13:	SỬ DỤNG CURSORS ĐỂ TRUY XUẤT DỮ LIỆU.....	118
13.1	Khái niệm	118
13.2	Làm việc với T-SQL server cursors	119
13.3	Ví dụ	121
BÀI 14:	BẦY LỖI - TRIGGER	123
14.1	Giới thiệu về trigger.....	123
14.2	Tạo và quản lý các trigger	124
14.2.1	Tạo trigger.....	124
14.2.2	Quản lý trigger	125
14.3	Vài ví dụ về trigger.....	125
BÀI 15:	BẢO MẬT TRONG SQL SERVER.....	128
15.1	Khái niệm về bảo mật.....	128
15.1.1	Mô hình truy cập bảo mật của SQL Server.....	128
15.1.2	Các chế độ bảo mật.....	128
15.1.3	Tìm hiểu các Server-Wide Permission.....	130
15.1.4	Tìm hiểu các quyền (Permission) chỉ định trên cơ sở dữ liệu.....	131
15.1.5	Fixed Database Roles.....	132
15.2	Tạo tài khoản đăng nhập (Login).....	133
15.2.1	Dùng Create Login Wizard.....	133
15.2.2	Dùng Enterprise Manager để tạo một Login.....	137
15.2.3	Tạo Login bằng T-SQL.....	141

BÀI 1:

GIỚI THIỆU MICROSOFT SQL SERVER 2000

1.1 Tổng quan về SQL Server 2000.

Microsoft SQL Server 2000 là một hệ quản trị cơ sở dữ liệu quan hệ (Relational database management system - RDBMS), nó cung cấp các dịch vụ quản lý và lưu trữ dữ liệu cho các tổ chức thương mại lớn, cùng với việc truy xuất dữ liệu hỗ trợ đối với các người dùng thông qua internet. Nó cũng hỗ trợ khả năng truy xuất một cách dễ dàng đối với các tổ chức nhỏ hơn và các người dùng riêng biệt. SQL Server 2000 chấp nhận và thực thi các yêu cầu của các khách hàng (Client) đối với việc hiệu chỉnh và xóa dữ liệu, cũng như các lệnh tạo các đối tượng như là các cơ sở dữ liệu (database) và các bảng (Table). SQL Server cho phép người dùng truy xuất và sắp xếp dữ liệu theo cách quan hệ, việc lưu trữ dữ liệu một cách hiệu quả theo dạng dòng và cột. Các lệnh của người dùng được gửi như là các câu lệnh truy vấn giao tác (Transact-SQL). Transact-SQL (T-SQL) là một ngôn ngữ dùng bởi SQL Server 2000 để truy vấn một cơ sở dữ liệu hoặc hiệu chỉnh nội dung của nó.

Một client từ một chương trình ứng dụng hoặc một người dùng gửi các câu lệnh T-SQL (các truy vấn hoặc hiệu chỉnh) thông qua mạng đến SQL Server để xử lý. Các chương trình ứng dụng mà gửi các câu lệnh có thể được viết bằng các ngôn ngữ như Visual Basic, Visual C++, hoặc Java. Chương này sẽ giúp bạn làm quen với Microsoft SQL Server cũng như thảo luận về các thành phần và thiết kết của nó.

Microsoft SQL Server phát triển một cách nhảy vọt trong nhiều năm. Phiên bản mới nhất, SQL Server 2000 hỗ trợ nổi bật hơn về tính bảo mật, lưu trữ và tính sẵn dùng của dữ liệu. SQL Server 2000 được tích hợp chặt chẽ với hệ điều hành Windows 2000, việc nâng cao bảo mật và các công cụ dựa trên giao diện đồ họa đã cung cấp trong SQL Server 2000 là cho các tác vụ quản trị trở nên dễ dàng hơn.

Các câu lệnh T-SQL dùng để hiệu chỉnh hoặc truy xuất dữ liệu có thể được chia thành ba nhóm khác nhau:

- **Ngôn ngữ định nghĩa dữ liệu (Data Definition Language - DDL)** được dùng để định nghĩa các đối tượng của cơ sở dữ liệu như các kiểu dữ liệu, bảng và các truy vấn.
- **Ngôn ngữ thao tác dữ liệu (Data Manipulation Language -DML)** được dùng để chọn, chèn, cập nhật và xóa dữ liệu trong các đối tượng của CSDL.
- **Ngôn ngữ điều khiển dữ liệu (Data Control Language - DCL)** được dùng để gán hoặc hủy bỏ các quyền từ các người dùng của CSDL.

SQL Server 2000 chạy trên hệ thống mạng Windows NT 4 hoặc Windows 2000 Server bằng cách dùng nhiều bộ xử lý và có thể được cài đặt như một hệ thống CSDL cá nhân (personal desktop) trên các máy trạm Windows NT 4, Windows 2000 Professional, Windows 98, và Windows Me. Bạn có thể dùng cùng một CD để cài đặt phiên bản Server SQL Server 2000 hoặc Personal SQL Server 2000. Ngoài ra bạn có thể cài đặt nhiều thể hiện (instance) của SQL Server 2000 trên cùng một máy tính, mỗi thể hiện có một người chủ (owner) và tập các người dùng (user) của nó.

Cơ sở dữ liệu (CSDL) - Databases

Một CSDL hay là một 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 dữ liệu 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 đựng 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, thì không có sự trùng lặp 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 vai 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, Tên, mã nhân viên, phòng ban, bậc lương, công việc ... Mỗi dòng trình bày một thể hiện của đối tượng thể hiện trong bảng. Ví dụ, một dòng trong bảng **NHANVIEN** trình bày thông tin về nhân viên có mã nhân viên là 12345.

Khi tổ chức dữ liệu vào các bảng, bạn 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 bạn định nghĩa sẽ được tổ chức dữ liệu một cách hiệu quả.

SQL

Để làm việc với dữ liệu trong một CSDL, bạn phải dùng một tập các lệnh và câu lệnh (một ngôn ngữ) được hỗ trợ bởi DBMS. Bạn có thể dùng vài ngôn ngữ khác nhau với CSDL quan hệ; thông thường nhất là SQL. Ngôn từ 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.

SQL Server 2000 bắt đầu hỗ trợ cho XML. Các đặc tính mới này bao gồm:

- Khả năng truy xuất đến SQL Server thông qua URL.

- Hỗ trợ cho lượt đồ dữ liệu XML (XML-Data schemas) và khả năng để chỉnh định các truy vấn XPath đối với các lượt đồ này.
- Khả năng truy xuất và ghi dữ liệu XML:
- Truy xuất dữ liệu XML bằng cách dùng câu lệnh SELECT và mệnh đề FOR
- Ghi dữ liệu XML bằng OpenXML rowset provider.
- Microsoft SQL Server 2000 OLEDB provider (SQLOLEDB) cho phép tải dữ liệu XML được gán như văn bản lệnh và trả về các tập kết quả như một luồng.

Các ấn bản của SQL Server 2000

Microsoft® SQL Server™ 2000 sẵn có các ấn bản sau:

SQL Server 2000 Enterprise Edition

Dùng như một sản phẩm database server. Hỗ trợ tất cả các đặc tính có sẵn trong SQL Server 2000, và phân chia các mức hiệu năng để hỗ trợ các trang Web lớn và các tiến trình giao tác trực tuyến của doanh nghiệp (online transaction processing OLTP) và các hệ thống kho dữ liệu.

SQL Server 2000 Standard Edition

Dùng như một database server cho các nhóm và bộ phận nhỏ.

SQL Server 2000 Personal Edition

Dành cho các người dùng lưu động - những người có thời gian không kết nối với network nhưng vẫn chạy ứng dụng mà đòi hỏi sự lưu trữ dữ liệu SQL Server. Cũng sử dụng khi đang chạy một ứng dụng stand-alone mà yêu cầu việc lưu trữ dữ liệu SQL Server trên máy tính client.

SQL Server 2000 Developer Edition

Dành cho các nhà lập trình phát triển ứng dụng mà dùng SQL Server 2000 như là nơi lưu trữ dữ liệu. Mặc dù Developer Edition hỗ trợ tất cả các đặc tính của Enterprise Edition cho phép người phát triển viết và kiểm tra các ứng dụng có thể sử dụng các đặc tính, Developer Edition chỉ cấp quyền sử dụng như là hệ thống phát triển hoặc kiểm tra ứng dụng, không phải là một sản phẩm server.

SQL Server 2000 Windows CE Edition

Microsoft® SQL Server 2000™ Windows® CE Edition (SQL Server CE) được dùng khi lưu trữ dữ liệu trên thiết bị Windows CE. Có khả năng đồng bộ hóa dữ liệu với bất kỳ ấn bản của SQL Server 2000 để sự đồng bộ hóa dữ liệu Windows CE với CSDL chính.

SQL Server 2000 Enterprise Evaluation Edition

Ấn bản với đầy đủ đặc tính, có thể tải về từ trang Web. Với mục tiêu chỉ cho sử dụng ước lượng các đặc tính của SQL Server; phiên bản này sẽ ngưng chạy sau 120 ngày tải về.

Ngoài các ấn bản này của SQL Server 2000, **SQL Server 2000 Desktop Engine** là một thành phần mà cho phép người phát triển ứng dụng phân tán một bản sao của SQL Server 2000 relational database engine với ứng dụng của họ. Trong khi các chức năng của database engine trong SQL Server 2000 Desktop Engine là tương tự như database engine trong các ấn bản SQL Server, kích cỡ của các Desktop Engine databases không vượt quá 2 GB.

1.2 Các thành phần của SQL Server 2000

Các thành phần của Server

- **SQL Server service**
Thực thi SQL Server database engine. Có một SQL Server service cho mỗi thể hiện (instance) của SQL Server đang chạy trên máy tính.
- **SQL Server Agent service**
Thực thi các tác nhân mà chạy các tác vụ quản trị SQL Server theo thời lịch. Chỉ có một SQL Server Agent service cho mỗi instance của SQL Server đang chạy trên máy tính. SQL Server Agent cho phép định nghĩa và lập lịch các tác vụ mà chạy dựa trên thời lịch hoặc tuần hoàn.
- **Microsoft Search service** (chỉ ở Windows NT và Windows 2000)
Thực thi bộ máy tìm kiếm full-text (full-text search engine). Chỉ có một dịch vụ bắt chấp số các instance SQL Server trên máy tính.
- **MSDTC service** (Chỉ ở Windows NT và Windows 2000)
Quản trị các giao tác phân tán. Chỉ có một service, bắt chấp số các instance SQL Server trên máy tính.
- **MSSQLServerOLAPService service** (chỉ ở Windows NT và Windows 2000)
Thực thi SQL Server 2000 Analysis Services. Chỉ có một service, bắt chấp số các instance SQL Server trên máy tính.

Các công cụ đồ họa (Graphical Tools)

- **SQL Server Enterprise Manager:** công cụ quản trị CSDL và server chính, nó 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 2000.
- **SQL Query Analyzer:** dùng để tạo và quản trị các đối tượng CSDL và kiểm tra các câu lệnh Transact-SQL, các batch, script một cách tương tác.
- **SQL Server Service Manager:** được dùng để start, stop, và pause các dịch vụ của SQL Server.
- **Client Network Utility:** dùng để quản trị các client Net-Libraries và định nghĩa các bí danh server bao gồm các tham số kết nối server tùy chọn nếu cần.
- **Server Network Utility:** dùng để quản trị các server Net-Libraries.
- **SQL Server Books online:** là một tài liệu trực tuyến hỗ trợ với Microsoft® SQL Server™ 2000. Bạn 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.

1.3 Kiến trúc của CSDL quan hệ (relational Database Architecture)

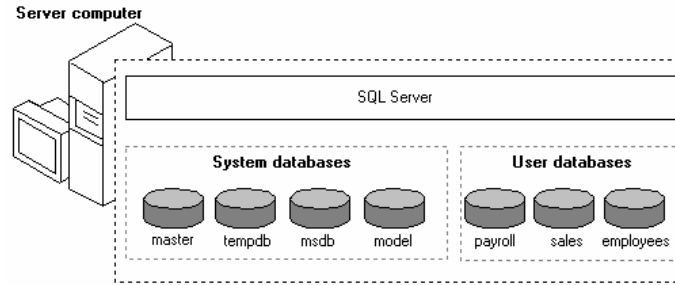
Trong SQL Server 2000, dữ liệu được lưu trong các CSDL. Về vật lý, một CSDL bao gồm hai hoặc nhiều hơn hai tập tin trên một hoặc nhiều đĩa, sự thể hiện vật lý này chỉ thấy được bởi nhà quản trị và nó trong suốt đối với người dùng. Sự chọn lựa vật lý của CSDL là trách nhiệm chính của người quản trị CSDL. Về mặt 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 procedure). 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 (user database). Ngoài ra, các hoạt động của SQL Server tùy thuộc vào bốn CSDL hệ thống (system Database): **master**, **model**, **Tempdb**, và **msdb**. Các CSDL này tồn tại trong mỗi thể hiện của SQL Server.

CSDL hệ thống và CSDL người dùng

- **CSDL Master:** được 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 cỡ ít nhất bằng kích cỡ CSDL model.
- **CSDL Tempdb:** được dùng như là vùng lưu chứa tạm thời đối với các bảng và các thủ tục tạm. SQL Server 2000 hỗ trợ hai loại bảng tạm: bảng tạm toàn cục (global temporary table) 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 cục 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*, và *restorehistory*. SQL Server có thể tự động tăng hoặc giảm kích cỡ 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
- Các CSDL khác là các CSDL do người dùng tạo ra.



Hình 1: Các CSDL hệ thống và CSDL của người dùng

Khái niệm về cơ sở dữ liệu quan hệ: Databases, Tables, Columns, Views, Datatypes, và Database Schemas

Dữ liệu trong một RDBMS như SQL Server 2000 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... Thành thạo 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.

Quản lý dữ liệu chính là lưu trữ với sự trợ giúp của một số đối tượng được cung cấp bởi SQL Server 2000. Bảng 2 liệt kê các đối tượng chính trong một database của SQL Server 2000.

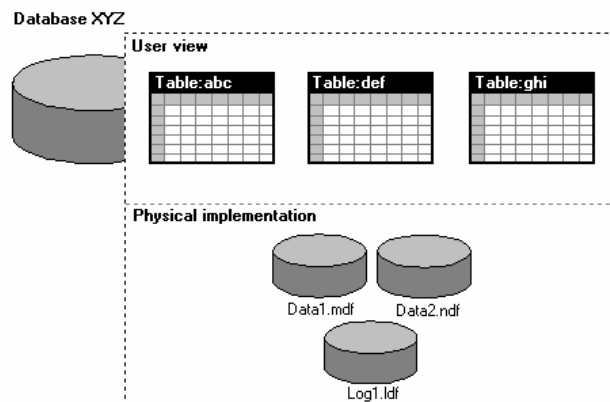
Bảng 1:

Đối tượng	Mô tả
Bảng - Table	Một table là một tập hợp của dữ liệu lưu trữ trong nhiều dòng nhiều cột.
Mặc định – Default	Một default là một cài đặt hệ thống, nghĩa là không được chỉ định bởi người dùng.
Cột - Column	Cột là các đối tượng dùng lưu trữ các phân tử riêng biệt bên trong một dòng dữ liệu.
Dòng – Row	Dòng là một mục đơn của dữ liệu bên trong một bảng, nó bao gồm tập các cột.
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. Các loại dữ liệu khác nhau là tinyInt, smallInt, int, real, float, smalldatetime, datetime, smallmoney, money, char và sql_variant.
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 2000 hỗ trợ các chỉ mục clustered và non-clustered. Chỉ mục Clustered (Clustered index) 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 cập nhập, 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.
Khung nhìn - View	View là một cách nhìn khác của table. Dùng để hiển thị các dữ liệu trong một hoặc nhiều table. Để sử dụng được view, người dùng phải có quyền trên view và tất cả các đối tượng phụ thuộc.

Cấu trúc vật lý của một CSDL

Mỗi CSDL bao gồm ít nhất một tập tin dữ liệu (data file) và một tập tin log (transaction log file). 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 (fault tolerance), tập tin dữ liệu và log được trải dài trên nhiều đĩa và thông thường dùng một Raid (Redundant array of independent disks)



Hình 2: Khung nhìn người dùng và triển khai vật lý của một CSDL

Nhân bản - Replication

Nhân bản cho phép nhiều thể hiện của SQL Server ở vị trí từ xa có cùng dữ liệu. Vị trí từ xa có thể bao gồm các người dùng di động hoặc các site kết nối thông qua internet, dial-up hay intranet. Sự phân chia vật lý của dữ liệu cải thiện hiệu năng của tổ chức khi dữ liệu cần được xử lý tại các nơi khác để trình bày ở site khác để tham chiếu.

Bảo mật trong SQL Server.

SQL Server 2000 dùng hai mức bảo mật khi kiểm tra sự hợp lệ của một người dùng. Đó là chứng thực (authentication) và authorization

- **Authentication**

Để kết nối đến một thể hiện của SQL Server 2000 thì người dùng phải chỉ định một định danh đăng nhập hợp lệ (ID). Một ID đăng nhập là một tài khoản định danh điều khiển sự truy xuất đến một thể hiện của SQL Server 2000. SQL Server kiểm tra ID đăng nhập có được phép kết nối đến thể hiện của SQL Server hay không. Sự kiểm tra của một ID đăng nhập được gọi là chứng thực (authentication). SQL Server 2000 dùng hai loại chứng thực: Windows authentication và SQL Server authentication. Hai chế độ chứng thực (Authentication Mode): Windows Authentication mode, mixed mode.

- **Authorization**

Sau khi một ID - tài khoản đăng nhập được chứng thực, SQL Server 2000 xác định ID có được ủy nhiệm để thi hành các thao tác trong CSDL hay không. Một ID đăng nhập tự nó không đưa ra các quyền (permission) truy xuất đến các đối tượng trong một CSDL. Một ID phải có một sự ủy nhiệm hoặc có quyền hợp lệ. Điều này ngăn chặn một đăng nhập tự động truy xuất đến các CSDL trong một thể hiện của SQL Server 2000.

Chúng ta sẽ tìm hiểu bảo mật trong SQL Server kỹ hơn ở trong bài bảo mật.

1.4 Giới thiệu về Transact-Sql

Transact_SQL là ngôn ngữ dùng chủ yếu trong SQL Server.

- ✓ Có đầy đủ tính chất của một ngôn ngữ lập trình.
- ✓ SQL là một chuẩn do IBM đề ra và tất cả ngôn ngữ lập trình bổ sung thêm một số tính năng riêng của ngôn ngữ lập trình đó.
- ✓ Đối với SQL Server thì các ứng dụng muốn truyền thông với SQL Server đều phải các câu lệnh T-SQL đến Server.
- ✓ Câu lệnh T-SQL là tập hợp các đoạn mã mà thực thi một vài hành động lên các đối tượng hoặc dữ liệu của một CSDL. SQL Server cung cấp 3 loại câu lệnh T-SQL: DDL, DCL, và DML

Ngôn ngữ định nghĩa dữ liệu (Data Definition Language – DDL): Dùng để định nghĩa và quản lý các đối tượng và đặc tính của đối tượng một CSDL. Lệnh DDL hỗ trợ việc định nghĩa, khai báo và chỉnh sửa các đối tượng của CSDL như Databases, Tables, Views. Mỗi lớp các đối tượng thì DDL có các lệnh như CREATE, ALTER, VÀ DROP (Ví dụ CREATE TABLE, ALTER TABLE, và DROP TABLE).

Ngôn ngữ điều khiển dữ liệu (Data Control Language – DCL): Dùng để hiệu chỉnh các quyền (Permission) trên các đối tượng của CSDL. Các permission được điều khiển bởi lệnh các lệnh GRANT (gán quyền), REVOKE (huỷ bỏ quyền), DENY (từ chối quyền)

Ngôn ngữ thao tác dữ liệu (Data Manipulation Language – DML): Các lệnh này được dùng để truy vấn (SELECT), chèn (INSERT), cập nhật (UPDATE), xóa (DELETE) các dữ liệu của các đối tượng được tạo bởi DDL.

BÀI 2: CÀI ĐẶT SQL SERVER 2000

2.1 Lập kế hoạch cài đặt SQL SERVER 2000

Các yêu cầu về phần cứng

Phần cứng	Yêu cầu tối thiểu
Máy tính	Intel hoặc tương thích. Pentium 166 MHz hoặc cao hơn.
Bộ nhớ (RAM)	Enterprise Edition: 64 MB; 128 MB hoặc hơn Standard Edition: 64 MB. Personal Edition: 64 MB đối với Windows 2000; 32 MB đối với các hệ điều hành khác. Developer Edition: 64 MB. Desktop Engine: 64 MB đối với Windows 2000; 32 MB đối với các hệ điều hành khác.
Không gian đĩa cứng	Các thành phần CSDL SQL Server: 95 đến 270 MB; 250 MB điển hình. Analysis Services: 50 MB; 130 MB điển hình. English Query: 80 MB. Desktop Engine only: 44 MB.
Màn hình	VGA hoặc độ phân giải cao hơn. 800 x 600 hoặc độ phân giải cao hơn SQL Server.
Thiết bị chuột	Microsoft Mouse hoặc tương thích.
CD	

2.2 Quyết định những chọn lựa cấu hình cài đặt SQL SERVER 2000

2.2.1 Xác định tài khoản người dùng (User account) thích hợp cho SQL Server Service và SQL Server Agent Service

Mỗi dịch vụ của SQL Server 2000 chạy trong một ngữ cảnh bảo mật của một user account. Bạn có thể chọn hoặc **local system account** hoặc **domain user account**. Thông thường các Service đều chạy chung một user account.

Local system account là một account hệ thống của Win NT hoặc Win 2000 với đầy đủ các quyền quản trị trên máy tính cục bộ. Tuy nhiên, Account này không có

quyền truy cập mạng. Tuy nhiên, trong môi trường Client/Server, bạn nên tạo và dùng một domain user account có tính chuyên biệt dành cho các dịch vụ. Sự lựa chọn một domain user account cho phép những dịch vụ của SQL Server truyền thông với những cài đặt SQL Server khác, tài nguyên mạng (như là chia sẻ tập tin) trên những máy tính khác trong môi trường domain.

Domain user account mà bạn chọn phải có đầy đủ các quyền truy xuất trên máy tính cục bộ, nhưng không cần phải là thành viên của nhóm Administrator cục bộ hoặc domain administrator. Những quyền được chỉ định này phải bao gồm quyền log on như là một dịch vụ, quyền truy xuất và thay đổi thư mục SQL Server, quyền truy xuất và thay đổi tập tin CSDL, đọc và ghi các khóa bất kỳ nào đó trong Registry của Windows. Bạn không cần lo lắng việc gán các quyền này, chương trình cài đặt SQL Server 2000 sẽ gán những quyền này một cách tự động cho domain user account mà bạn chỉ định.

2.2.2 Chọn một chế độ xác thực (Authentication Mode)

SQL server 2000 hỗ trợ 2 chế độ xác thực: Windows authentication Mode và Mixed mode. Mặc định là Authentication Mode.

Chế độ **Windows Authentication Mode** chỉ cho phép các user của hệ điều hành mới có thể kết nối với SQL Server. Chế độ **Mixed mode** cho phép tất cả các user của hệ điều hành hoặc SQL server đều có thể kết nối đến SQL Server.

Nếu bạn đang trong giai đoạn tìm hiểu và thử nghiệm SQL Server thì bạn nên chọn Mixed Mode. Nếu bạn triển khai ứng dụng thì bạn nên sử dụng Windows authentication mode, với chế độ này tính bảo mật được tăng cao nhờ tích hợp thêm các khả năng bảo mật của hệ điều hành.

2.2.3 Xác định Collation

Collation là tập hợp những qui tắc quản lý, nó sẽ ảnh hưởng đến cách lưu trữ dữ liệu, thứ tự sắp xếp... Ví dụ khi bạn cài đặt hệ điều hành, bạn chọn lựa ngôn ngữ, bàn phím được sử dụng. Mỗi ngôn ngữ sẽ có bộ ký tự khác nhau do đó có những code page khác nhau và dựa trên cơ sở này hệ điều hành sẽ cài đặt các numbers, currencies, Times, Dates khác nhau. Khi bạn cài đặt SQL Server, chương trình cài đặt sẽ xác định một collation mặc định cho SQL Server dựa trên các cài đặt của hệ điều hành.

2.2.4 Chọn Network Libraries

SQL Server sử dụng Network libraries để gửi các packet giữa SQL Server khách và chủ. Server và Client phải có ít nhất một Network Libraries chung.

Các Network Libraries: NEBEUI, TCP/IP, IPX/SPX, Apple Talk ADSP, Banyan Vines, VIA giginet San.

2.2.5 Xác định client Licensing Mode

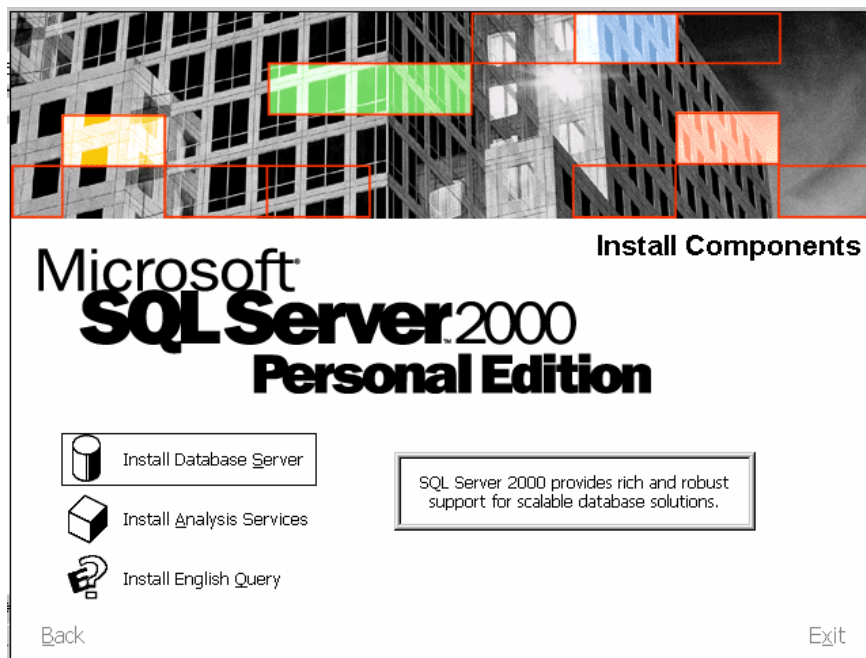
SQL Server cung cấp 2 kiểu client licensing: Per processor và Per seat.

2.3 Các bước cài đặt SQL Server 2000.

Bạn đưa đĩa CD chứa chương trình cài đặt SQL Server hoặc chạy tập tin Setup từ một vị trí nào đó trên mạng.



Hình 3: Màn hình đầu tiên khi đưa đĩa SQL Server Personal Edition vào
Chọn SQL Server 2000 Components

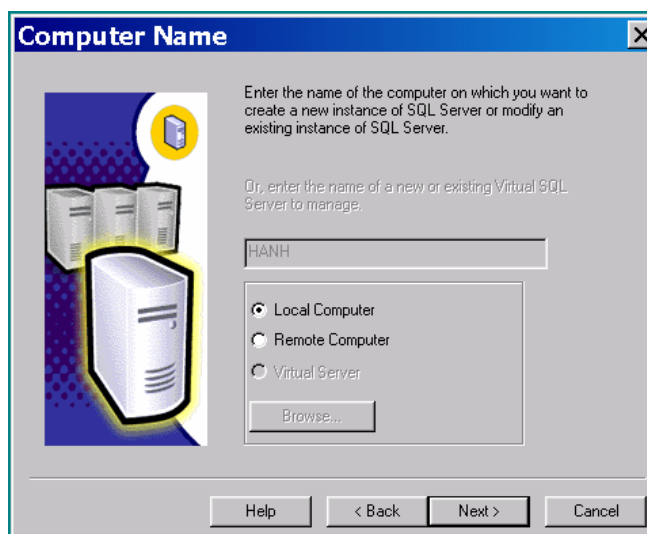


Hình 4: Chọn thành phần cần cài đặt
Chọn Install Database Server



Hình 5: Màn hình Welcome của MicroSoft SQL Server

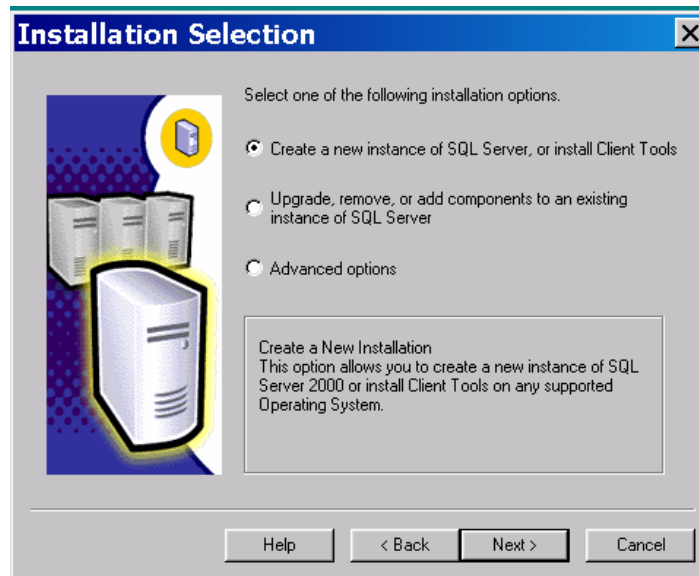
Chọn Next



Hình 6: Chọn tên máy tính nơi mà sẽ cài 1 instance mới hoặc hiệu chỉnh instance có sẵn

- Local Computer : Cài trên máy Local
- Remote Computer : Cài đặt từ xa
- Virtual Server : Cài trên Server ảo

Chọn Local Computer, chọn Next



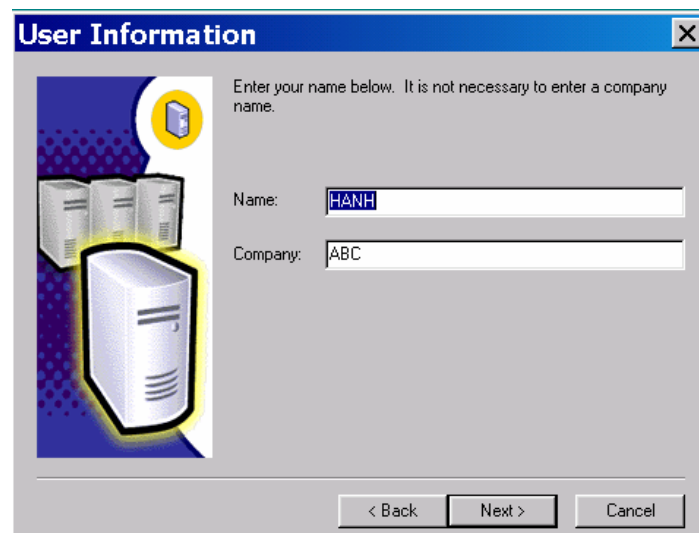
Hình 7: Chọn hình thức cài đặt

Create a new instance of SQL Server, or install client tools: Cài một instance mới trên server hoặc cài các client tools

Upgrade, remove, or add components to an existing instance of SQL Server: Nâng cấp, hủy bỏ, hoặc thêm các component vào instance có sẵn.

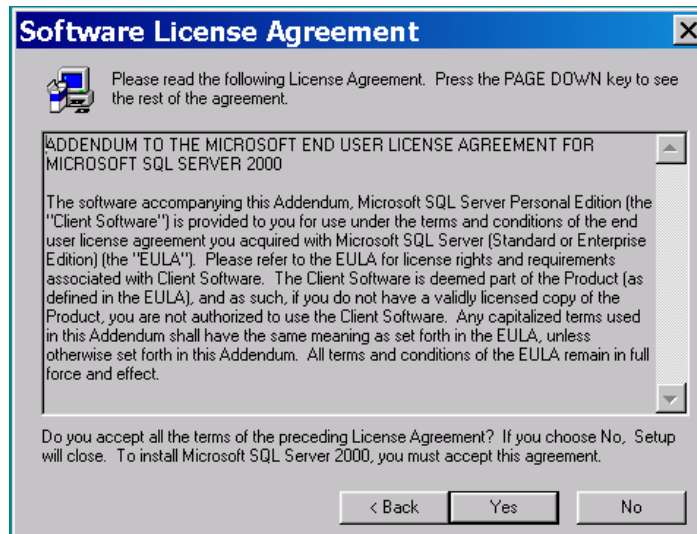
Advanced option: Một số chức năng nâng cao hoặc tạo tập tin cài đặt không tham dự.

Chọn Create a new instance of SQL Server, or install client tools, chọn Next



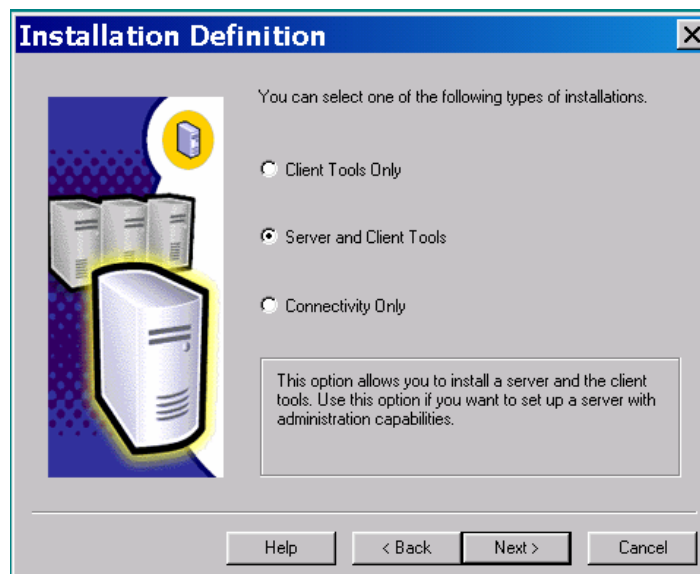
Hình 8: Thông tin của người dùng

Chọn Next



Hình 9: Thông báo License

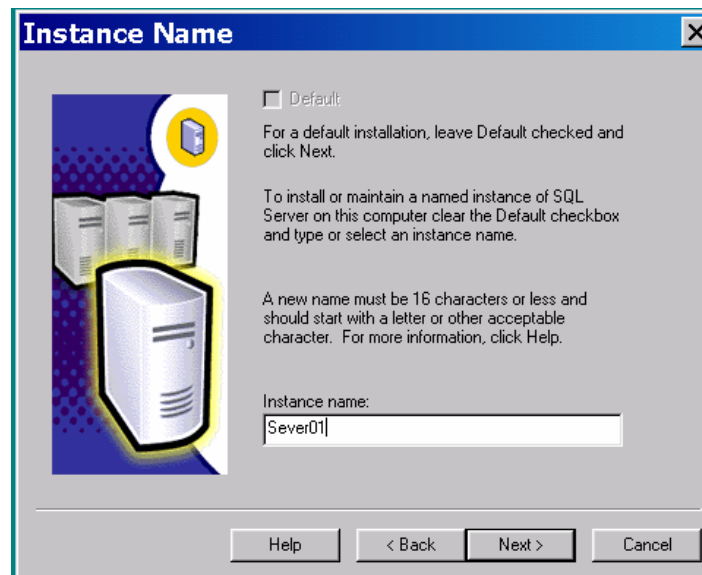
Chọn next



Hình 10: Xác định công cụ cài đặt

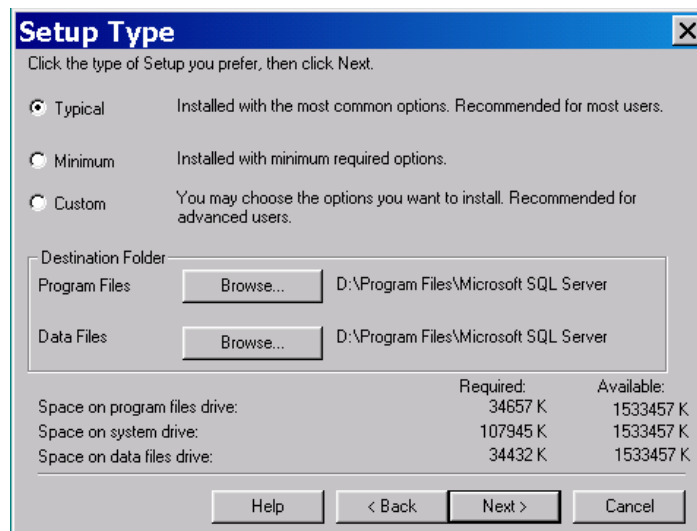
- | | |
|-------------------------|--|
| Client Tools Only | : Chỉ cài đặt các tools của client |
| Server and Client Tools | : Cài đặt các tools của Server và Client |
| Connectivity Only | : Cài đặt các thành phần truy xuất dữ liệu và các thư viện network |

Chọn Server and ClientTools



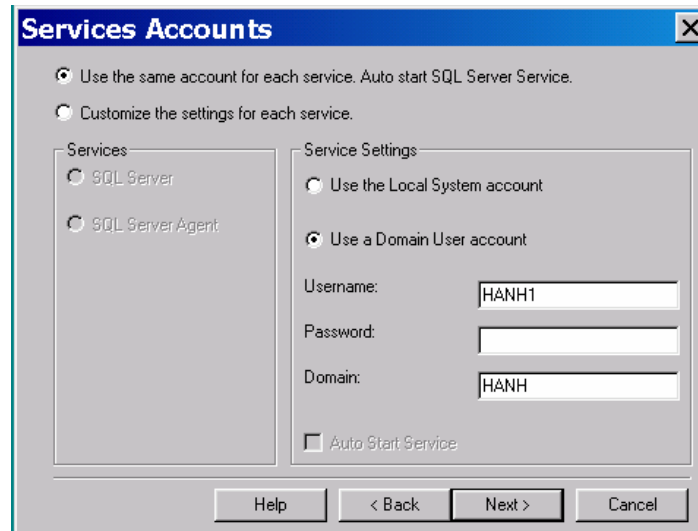
Hình 11: Quy định tên của instance

Gõ vào Server01, chọn Next



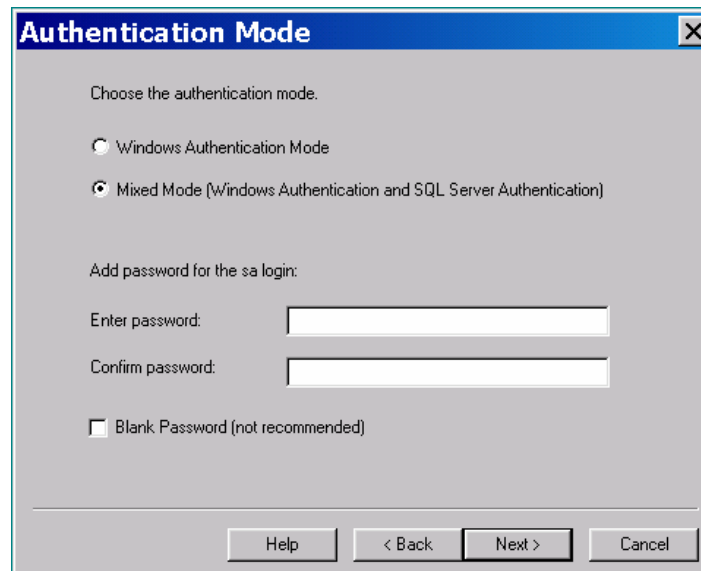
Hình 12: Chọn loại cài đặt

Chọn Typical, chọn Next



Hình 13: xác định Service Account để chạy các Service

Chọn Next

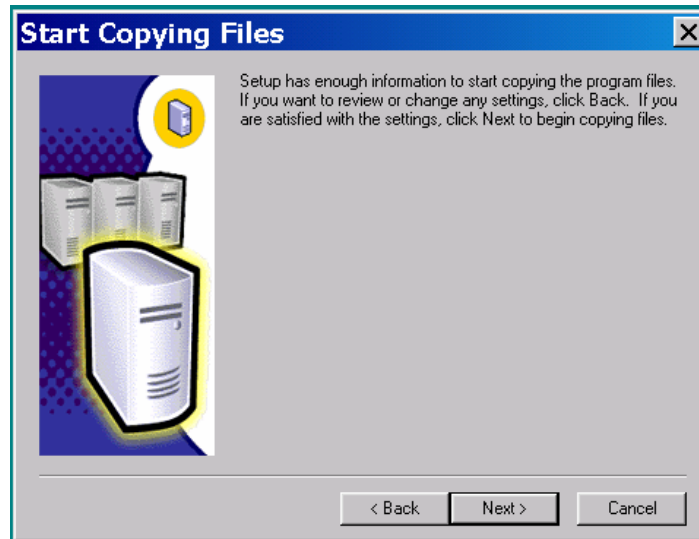


Hình 14: Xác định chế độ chứng thực

Windows Authentication Mode: chế độ xác thực của Windows

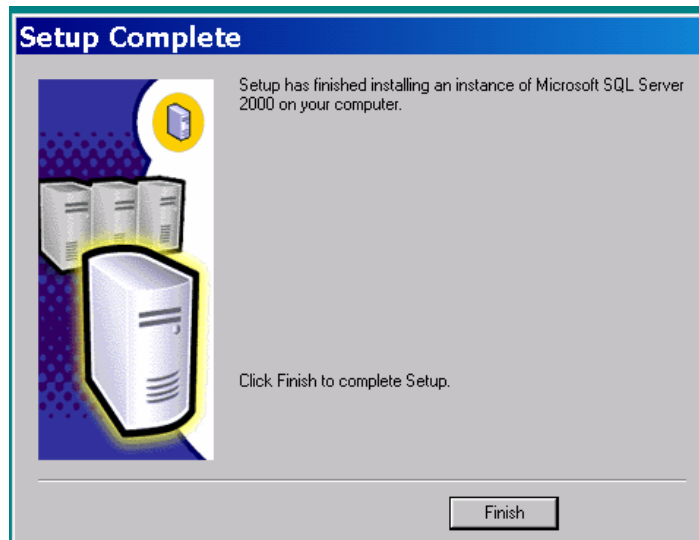
Mixed Mode: Chế độ xác thực vừa của Windows và vừa của SQL Server

Chọn Mixed Mode



Hình 15: Thông báo là SQL Server sẽ bắt đầu được cài đặt

Chọn Next, tiến trình cài đặt bắt đầu



Hình 16: Kết thúc cài đặt

2.4 Tạo tập tin cài đặt không tham dự (unattended) và cài đặt từ xa (Remote) SQL Server 2000

Có 3 cách để tạo ra tập tin khởi đầu cài đặt dành cho việc cài SQL Server 2000 không tham dự.

Cách 1: Chương trình Setup SQL Server 2000 cung cấp lựa chọn Advanced Option để ghi nhận lại tập tin không tham dự .ISS. Nếu lựa chọn này được chọn thì quá trình cài đặt sẽ được ghi nhận lại thành tập tin .ISS trong thư mục \Winnt, và SQL Server không được cài đặt thực sự trong tiến trình này.

Cách 2: tập tin .ISS được cung cấp trong đĩa SQL Server 2000, trong thư mục gốc. Bạn có thể dùng trực tiếp từ các tập tin này hoặc hiệu chỉnh bằng các trình soạn thảo văn bản. Tập tin đó là SQLLINS.ISS, SQLCLI.ISS; SQLCST.ISS

Cách thứ 3: Tập tin .ISS được tự động tạo ra ngay sau khi mỗi lần bạn cài đặt SQL Server, nó nằm trong \WinNt. Tuy nhiên, nếu dùng tập tin này thì bạn phải chỉnh sửa nó nằng một trình soạn thảo văn bản để thêm phần [SdFinish-0]

2.5 Kết quả của việc cài đặt SQL SERVER 2000

Trình cài đặt SQL server tạo ra một loạt các thư mục để lưu các tập tin thi hành và cấu hình khác nhau. Theo ngầm định, SQL Server được cài đặt trên ổ đĩa C: trong thư mục MSSQL. Dưới thư mục MSSQL là vài thư mục khác:

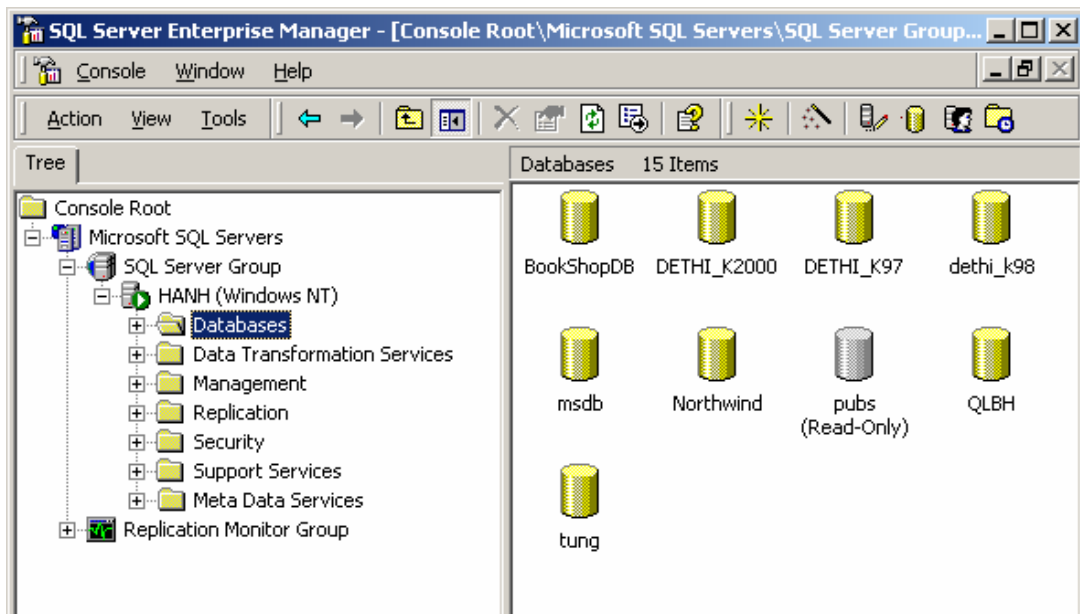
Backup	Thư mục vẫn còn trống ngay sau khi cài đặt. Thư mục này dùng để lưu trữ các tập tin dự phòng.
BIN	Các thư viện mạng phía client
BINN	Các tập thi hành và các tập tin kết hợp. Kể cả các tập tin thi hành chính của SQL Server, và mọi công cụ điều hành được nạp.
CHARSETS	Các bộ ký tự sắp xếp cho các kiểu cài đặt khác nhau.
DATA	Vị trí ngầm định cho các CSDL.
INSTALL	SQL Server Books Online và các chỉ mục.
LOG	Các sổ theo dõi lỗi SQL Server. Các tập tin văn bản tương đương với sổ theo dõi sự kiện Windows NT, nhưng chi tiết hơn.
REPLDATA	Vị trí ngầm định cho dữ liệu tạm thời được dùng trong khi sao lập.
SNMP	Các MIB (Management Information Bases = cơ sở thông tin quản trị) cho SQL Server.
SQLOLE	các mẫu về các dùng OLE automation của Visual Basic để quản lý SQL Server.
SYMBOLS	Gỡ rối các ký hiệu do các lập trình viên sử dụng.

BÀI 3: CÁC CÔNG CỤ CỦA SQL SERVER

3.1 SQL ENTERPRISE MANAGER

Enterprise Manager còn gọi tắt là EM, là một công cụ chính dành cho nhà quản trị server và CSDL. Enterprise Manager cho phép bạn dừng và khởi động một Server, cũng như cho phép bạn 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 mặc định (default), các thiết bị dự phòng (backup device), và các vết lỗi (error log).
- ✓ Quản lý các service khác.



Hình 17: Cửa sổ Enterprise Manager của SQL Server

3.2 SQL SERVER SERVICES MANAGER

SQL Server Manager là một công cụ cho phép khởi động, tạm dừng hoặc dừng các dịch vụ (service) trong SQL Server. SQL Server hoạt động được thông qua các dịch vụ mà nó tự cung cấp.

3.2.1 Các dịch vụ của SQL Server

Một dịch vụ (*service*) là một chương trình hoặc tiến trình thực thi một chức năng đặc biệt nào đó nhằm hỗ trợ cho các chương trình khác. SQL Server cung cấp các dịch vụ sau: *SQL Server Service*, *SQL Server Agent Service*, *Microsoft Search Service* và *Microsoft Distributed Transaction Coordinator*. Mỗi dịch vụ có chức năng và nhiệm vụ riêng, hỗ trợ cho các hoạt động của SQL Server. Nếu bạn cài đặt một hoặc nhiều hơn thể hiện (instance) của SQL Server thì tên của service cho mỗi instance của SQL Server là *MSSQL\$InstanceName*, *\$InstanceName* là một tên của instance mà bạn chỉ định ở lần cài đặt. Ứng với mỗi SQL Server Agent service cho mỗi instance được gọi là *SQLAGENT\$InstanceName*. Tuy nhiên, nhiều instances của SQL Server, sẽ chỉ có một Microsoft Distributed Transaction Coordinator và Microsoft Search.

SQL SERVER SERVICE

Khi bạn khởi động SQL Server có nghĩa là dịch vụ SQL Server service được khởi động ở Windows NT hoặc Windows 2000. Dịch vụ này quản lý các tập tin CSDL, xử lý các câu lệnh T-SQL, định vị tài nguyên giữa các kết nối của người dùng hiện hành, đảm bảo tính nhất quán dữ liệu, và nhiều hơn nữa.

SQL SERVER AGENT SERVICES

SQL Server Agent hỗ trợ việc lập biểu và thực thi các công việc (job), các cảnh báo (alert), thông báo, và kế hoạch duy trì CSDL. Không có service này, công việc quản trị của bạn sẽ trở nên khó khăn nhiều. SQL Server Agent cho phép bạn thực hiện tự động các thao tác nhằm duy trì CSDL.

Ví dụ: bạn có thể tạo một job để thực hiện tự động dự phòng (backup) dữ liệu mỗi đêm vào lúc 1 giờ sáng và một job khác thực hiện dự phòng transaction log mỗi 30 phút một lần.

Để kiểm tra hiệu năng hệ thống của bạn cảnh báo hiện trạng hiệu năng để báo cho bạn nếu server CPU đã hoạt động trên 90%. SQL Server Agent phải chạy để thực thi các tác vụ kiểu như vậy.

Dịch vụ này có thể được cấu hình khởi chạy một các tự động hoặc chạy một cách thủ công. Bạn nên cấu hình cho nó khởi động tự động để đảm bảo rằng các job, alerts, và notification sẽ có thể được thực thi.

MICROSOFT DISTRIBUTED TRANSACTION COORDINATOR.

Dùng quản lý các giao tác phân tán.

MICROSOFT SEARCH.

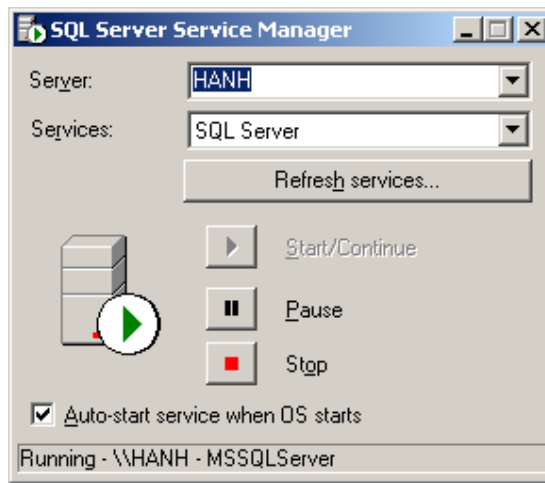
Dịch vụ Microsoft Search cho phép tạo một chỉ mục full-text và cơ chế tìm kiếm. Chuẩn SQL-92 định nghĩa các khả năng tìm kiếm với phép so sánh ký tự bằng, nhỏ hơn, lớn hơn một hàng số ký tự, giá trị ký tự có chứa mẫu chuỗi.

Dùng dịch vụ Microsoft Search cho phép Microsoft® SQL Server™ 2000 và SQL Server 7.0 được hỗ trợ nhiều tìm kiếm tinh vi trên cột chuỗi ký tự.

3.2.2 Khởi động, tạm ngưng, dừng các dịch vụ của SQL Server

Để khởi động hay dừng các dịch vụ SQL Server bằng cách dùng SQL Server Service Manager thì thực hiện các bước sau:

1. Click chọn **Start** → **Programs** → **Microsoft SQL Server**, và chọn **Service Manager** để hiển thị công cụ SQL Service Manager như hình



Hình 18: SQL Server Service Manager.

2. Tên server cục bộ xuất hiện trong mục **Server** và các dịch vụ của SQL Server xuất hiện trong mục **Service**. Trong danh sách xổ xuống, chọn tên của server và dịch vụ mà bạn muốn điều khiển.

Bạn có thể khởi động (start) và dừng (stop) các service đang được chọn bằng cách click vào ứng tương ứng. Nếu Service ở trạng thái dừng thì click vào Start/Continue để chạy tiếp.

3. **Auto-start service when OS Server:** cấu hình tự động chạy dịch vụ khi hệ điều hành khởi động.

Lưu ý:

Nếu SQL Server và SQL Server Agent service không được cấu hình chạy tự động thì bạn phải khởi động nó một cách thủ công

3.3 OSQL

OSQL là một dấu nhắc tiện ích dùng để truy vấn một instance của SQL Server 2000 một cách tương tác bằng T-SQL, thủ tục hệ thống, tập tin lệnh. Nó cũng được dùng để xem xét các công việc hoặc bó lệnh, kể cả các lệnh của hệ điều hành đối với SQL Server 2000. Bạn có thể dùng osql bằng cách kết nối vào server và

thực thi các lệnh trong chế độ tương tác hoặc bằng cách kết nối vào và thực thi các lệnh như một phần của cú pháp lệnh osql.

Cú pháp lệnh của osql như sau:

Osql -S servername - U login_id - P password (1)

Osql -S servername - E (2)

Chúng ta dùng (1) khi bạn kết nối đến server bằng một tài khoản đăng nhập của SQL server, dùng (2) khi bạn kết nối bằng tài chế độ chứng thực là Windows. Khi dùng Osq để kết nối đến SQL Server 2000, có nhiều tham số mà bạn có thể dùng như là một phần của chuỗi kết nối. Nên nhớ rằng, Osq phân biệt chữ thường, chữ hoa.

- S servername	Chỉ định tên của Server của SQL Server mà bạn muốn kết nối.
- U login id	Chỉ định tên của tài khoản đăng nhập
- P password	Chỉ định password của tài khoản đăng nhập nếu có
- E	Kết nối bằng tài khoản đăng nhập hiện hành của windows

Khi bạn dùng osql trong chế độ tương tác, nó hiện thị thứ tự các dòng. Chúng ta gõ các câu lệnh SQL, mỗi câu lệnh trên mỗi dòng. Osq không thực thi câu truy vấn của bạn cho đến khi bạn gõ từ khóa GO trên một dòng. Không làm việc bằng osql, gõ EXIT để tắt kết nối và đóng tiện ích osql.

Ví dụ:

```

C:\WINNT\System32\cmd.exe - osql -S hanh -U sa -P
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>osql -S hanh -U sa -P
1> USE Northwind
2> SELECT customerID, companyname
3> FROM Customers
4> WHERE Companyname LIKE 'W%'
5> GO
customerID  companyname
-----
WARTH      Wartian Herkku
WELLI      Wellington Importadora
WHITC      White Clover Markets
WILMK      Wilman Kala
WOLZA      Wolski Zajazd

<5 rows affected>
1> _

```

Hình 19: Minh họa tiện ích osql

3.4 SQL QUERY ANALYZER

3.4.1 Giới thiệu.

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.

3.4.2 Khởi động Query Analyzer

SQL Query Analyzer có thể được khởi động từ SQL Server Enterprise Manager hoặc từ Start Menu hoặc từ cửa sổ Command bằng cách thực thi tiện ích ISQLW.

Khi bạn khởi động SQL Query Analyzer, thì hộp thoại Connect to SQL Server xuất hiện. Khi đó bạn phải xác định chế độ chứng thực được dùng để kết nối tới SQL Server.



Hình 20: Hộp thoại kết nối

Bạn có thể mở cùng lúc nhiều cửa sổ Query Analyzer để cho phép bạn làm việc trong các CSDL khác nhau hoặc thực thi các script khác nhau trong cùng một thời điểm. Mỗi cửa sổ tạo kết nối riêng biệt đến server. Những kết nối duy trì các cài đặt khác nhau và mỗi cửa sổ có CSDL hiện hành. Nếu bạn cố gắng thực thi một thao tác dành riêng trên một CSDL từ một query analyzer trong khi một query analyzer khác đang dùng CSDL, thao tác sẽ bị lỗi.

3.4.3 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.

Cửa sổ Query Analyzer

- **Thanh tiêu đề** (Title bar): Hiển thị tên của Server, CSDL hiện hành, và tài khoản kết nối tới.
- Công cụ **Database** trên thanh công cụ, cho biết và cho phép bạn thay đổi CSDL được kết nối hiện hành.
- **Editor pane**: dùng để đưa vào các câu lệnh và thực thi các câu lệnh T-SQL

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

Màu của mã lệnh bạn có thể biết là bạn nhập câu lệnh vào là đúng hay sai. Bạn có thể đổi màu qui định bằng các chọn trang fonts trong hộp thoại Tools → Option.

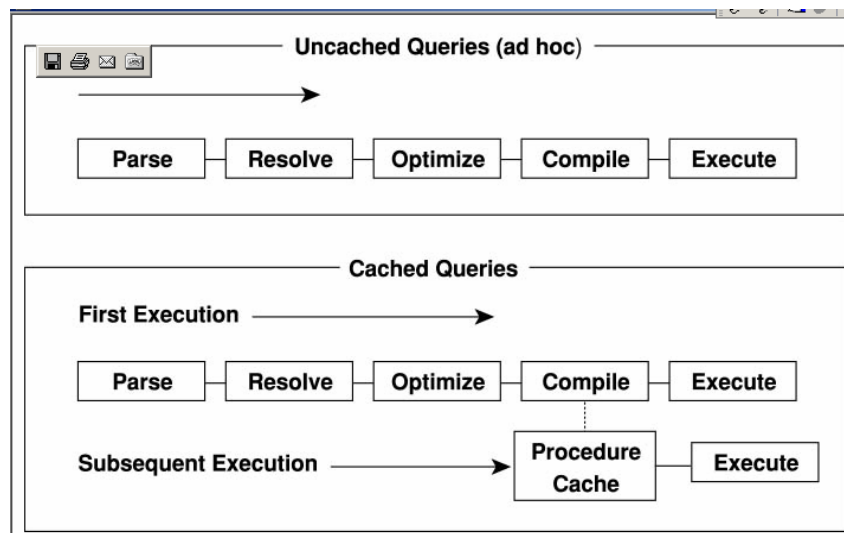
Results pane: hiển thị kết quả của truy vấn được thực thi. Một hoặc nhiều hơn một trang được hiển thị trong results pane.

Trang Messages: Hiển thị thông báo và các lỗi gửi trả từ server.

Trang Results: Hiển thị những kết quả như văn bản tự do.

Trang Results Grid: Hiển thị các kết quả trong bảng kẻ lưới. Dữ liệu trong lưới chỉ để xem, không thể hiệu chỉnh được. Một vài truy vấn yêu cầu server trả về nhiều hơn một tập kết quả thì sẽ có nhiều hơn một trang Results Grid sẽ được hiển thị. Nếu lưới kết quả trống sau khi thi hành một truy vấn thì truy vấn không được trả về một bảng kết quả

Trang Execution Plan: Hiển thị một biểu đồ của kế hoạch thực thi của truy vấn hiện hành. Bật tắt Execution Plan ta chọn từ thực đơn Query



Hình 21: Thành phần chính của Query Analyzer.

Lưu ý:

- Bạn có thể nhập vào chuỗi các câu lệnh mới hoặc mở một tập lệnh có sẵn. Khi bạn đã làm việc xong với tập lệnh bạn có thể lưu nó thành một tập tin để dàng dùng lại sau (tập tin có phần mở rộng là .SQL).
- Truy vấn có thể là một câu lệnh đơn hoặc nhiều câu lệnh. Những câu lệnh có thể không thể thực thi như là một phần của cùng một truy vấn với những câu lệnh khác. Trong trường hợp này được viết cách nhau bởi từ khóa GO.
- Những câu lệnh có thể gõ trên cùng một hàng hoặc trải dài qua nhiều hàng. Do câu lệnh T-SQL thì quá dài để đặt trên một dòng, nên ta gõ chúng trên nhiều dòng, điều này sẽ làm cho chúng ta dễ đọc các câu lệnh.
- Nếu không có lệnh được chọn thì khi bạn thi hành truy vấn thì toàn bộ nội dung của query pane sẽ được thực thi

3.4.4 Một vài phím nóng dùng trong Query Analyzer

Hành động	Phím nóng
Thực thi	CTRL+E or F5
Tim 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
Giúp đỡ về Query Analyzer	F1
Giúp đỡ một câu lệnh T-SQL được chọn	SHIFT+F1

BÀI 4: LÀM VIỆC VỚI CƠ SỞ DỮ LIỆU SQL SERVER

4.1 Thiết kế một cơ sở dữ liệu.

Thiết kế CSDL là một bước vô cùng quan trọng trong tiến trình phát triển ứng dụng. Trong quá trình thiết kế, bạn phải quyết định những table nào cần để lưu trữ các dữ liệu của bạn. Quá trình thiết kế một CSDL logic được thực hiện độc lập với các hệ quản trị CSDL. Một số lưu ý khi thiết kế một CSDL logic:

- Các bảng và tên của chúng (còn gọi là các thực thể).
- Tên các cột (các thuộc tính) của mỗi bảng.
- Các đặc tính của cột như là giá trị duy nhất, cho phép null hay không, và kiểu của dữ liệu mà cột sẽ lưu chứa.
- Khóa chính (Primary key) cho mỗi bảng. Đó là một cột hoặc tập các cột mà chứa các giá trị được định nghĩa không trùng lặp ở các dòng trong bảng. Mỗi bảng chỉ có thể có một khóa chính. Mặc dù khóa chính là không yêu cầu bắt buộc phải có nhưng các bảng nên luôn luôn có.
- Các mối quan hệ (Relationship) giữa các bảng. Các dòng trong một bảng phụ thuộc một hoặc nhiều dòng khác trong bảng khác. Những phụ thuộc trong những bảng này được gọi là mối quan hệ. Để định nghĩa mối quan hệ, một cột hoặc tập các cột trong một bảng được gọi là khóa ngoại (foreign key) nếu nó tham chiếu đến khóa chính của bảng khác.

Ví dụ: mỗi dòng trong một bảng đơn hàng (DONHANG) phụ thuộc vào một dòng trong bảng khách hàng (KHACHHANG) bởi vì mỗi đơn hàng phải được đặt bởi một khách hàng. Điều này chính là nối quan hệ giữa bảng DONHANG và bảng KHACHHANG. Bảng đơn hàng phải có một cột lưu giữ các giá trị được tham chiếu đến một dòng riêng lẻ trong bảng KHACHHANG. Các dòng trong bảng HOADON phải được đảm bảo chỉ tham chiếu đến một khách hàng vì vậy mỗi quan hệ nên dựa trên cơ sở khóa chính của bảng KHACHHANG. Cột của bảng đơn hàng mà tham chiếu đến khóa chính của bảng khách hàng được gọi là khóa ngoại.

Các kiểu mối quan hệ: Ba kiểu của mối quan hệ có thể giữa các bảng:

- **One-to-One.** Mỗi dòng trong bảng chính có quan hệ đến chỉ một dòng trong bảng quan hệ. Một mối quan hệ one-to-one là được thực hiện bởi định nghĩa khoá ngoại là duy nhất (không trùng).
- **One-to-Many.** Mỗi dòng trong bảng chính được liên quan đến một hoặc nhiều dòng trong bảng quan hệ. Ví dụ một khách hàng có thể đặt nhiều đơn hàng, nhưng một đơn hàng không thể được đặt bởi nhiều khách hàng.
- **Many-to-Many.** Nhiều dòng trong một bảng liên quan đến nhiều dòng trong bảng khác.

Ví dụ: một tác giả có thể viết nhiều quyển sách và một quyển sách có thể được viết bởi nhiều hơn một tác giả. Mối quan hệ many-to-many giữa 2 bảng là thực hiện bằng cách tạo một bảng thứ ba và tạo một mối quan hệ one-to-many đến bảng chức năng này từ mỗi bảng ban đầu.

Một CSDL sau khi thiết kế được đánh giá thông qua các dạng chuẩn. CSDL đạt chuẩn cao thì CSDL đó lưu trữ đầy đủ thông tin, không bị trùng lặp, có tính nhất quán cao.

Các dạng chuẩn của CSDL quan hệ

- **Dạng chuẩn 1:** Tất cả các thuộc tính đều được định nghĩa ở dạng 1 giá trị đơn hoặc không ở dạng repeating group.

Ví dụ: Các quan hệ sau không đạt dạng chuẩn 1

OrderID	CustomID	OrderDate	Items	OrderTotal
1111	101	1/1/02	4 apples	1000
1111	103	1/2/02	5 bananas	900

Không đạt chuẩn 1 vì thuộc tính Items không là giá trị đơn

OrderId	CustomId	Items	Quarter1	Items	Quarter2	Items	Quarter3	...

Không đạt chuẩn 1 vì thuộc tính Items và Quarter có dạng repeating group

- **Dạng chuẩn 2:** Tất cả các thuộc tính không khóa phụ thuộc đầy đủ vào khóa.

ProductName	SupplerName	CompanyName	SupplerPhoneNumber

Giả sử ProductName không trùng, ProductName là khóa.
Vi phạm chuẩn 2 vì SupplerPhoneNumber chỉ phụ thuộc vào SupplerName mà không phụ thuộc vào ProductName (khóa)

- **Dạng chuẩn 3:** Tất cả thuộc tính không khóa phụ thuộc đầy đủ và không phụ thuộc bắt cầu vào khóa.

CompanyName	Address	City	Region	Postcode

Giả sử CompanyName là không trùng
Vi phạm chuẩn 3 vì từ City và Region thì ta sẽ biết được Postcode, như vậy PostCode phụ thuộc vào City và Region. Trong đó CompanyName là một khóa dự tuyển (Candidate)

Lưu ý:

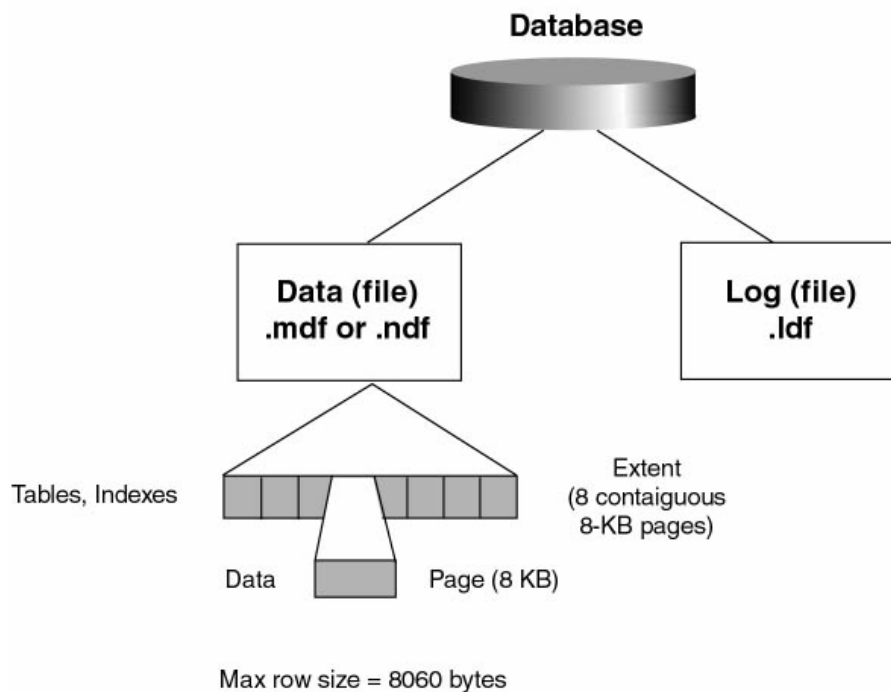
- ✓ Một cột (field) có giá trị lặp đi lặp lại một giá trị thì giá trị đó gọi là dữ liệu dư thừa.
- ✓ Các giá trị này người sử dụng có thể gõ sai chính tả khi đó nên suy nghĩ để quyết định có nên nảy sinh thêm 1 table hay không.
- ✓ Hoặc trong tương lai có thể nảy sinh ra các giá trị mới của field đó thì việc phát sinh Table là ưu việt.
- ✓ Khi xây dựng dữ liệu của khóa chính, có 2 trường phái: khóa có ngữ nghĩa và khóa không có ngữ nghĩa. Để giải quyết các lỗi phát sinh khi dùng giá trị khoá có ngữ nghĩa, các chuyên gia khuyên là ngữ nghĩa của khoá chỉ dùng để tham khảo và không nên dùng nó khi viết các ứng dụng.

4.2 Cơ sở dữ liệu của SQL SERVER 2000

Một CSDL của SQL Server là tập hợp các bảng (Table) dùng để chứa các dữ liệu và những đối tượng khác, chẳng hạn Views, indexes, store procedures, và triggers... Chúng được ấn định để hỗ trợ các hoạt động được thực hiện với dữ liệu. Dữ liệu được chứa trong một CSDL thường liên quan đến một chủ đề hay tiến trình đặc biệt. SQL Server có thể hỗ trợ nhiều CSDL. Mỗi một CSDL có thể liên quan hay không liên quan đến các CSDL khác. Ví dụ một SQL Server có thể có CSDL chứa dữ liệu về nhân sự, và một CSDL khác chứa các đơn hàng.

Khi bạn tạo một CSDL, bạn phải xây dựng cấu trúc lưu trữ dữ liệu. Các trúc này bao gồm ít nhất một tập tin dữ liệu (data file) và một tập tin vết giao tác (transaction log file). Bạn nên hiểu cách thức Microsoft SQL Server 2000 lưu trữ dữ liệu, cũng như chức năng của transaction log trước khi bạn làm việc với CSDL của SQL Server.

Cách dữ liệu được lưu trữ trong CSDL của SQL Server:



Hình 22: Cách lưu trữ dữ liệu

Các tập tin CSDL SQL Server

Một CSDL được lưu chứa trong các tập tin vật lý trên đĩa cứng, một CSDL trải dài trên ít nhất là hai tập tin. Một vài **tập tin dữ liệu (data file)**, và một **tập tin vết (transaction log file)**. Những tập tin này được chỉ định khi CSDL được tạo hay hiệu chỉnh. SQL Server 2000 cho phép ba loại tập tin CSDL:

- **Primary data files:** một CSDL có một primary data file dùng để ghi nhận lại tất cả những tập tin khác trong CSDL, và lưu trữ dữ liệu. Theo ngầm định, tên của primary data file có phần mở rộng là **.MDF**.
- **Secondary data files:** một CSDL có thể không có hoặc có nhiều secondary data files, dùng để lưu các đối tượng của CSDL. Theo ngầm định, tên của secondary data file có phần mở rộng là **.NDF**.

- **Log files:** một CSDL có ít nhất một log file dùng chứa những thông tin cần thiết cho việc phục hồi tất cả những giao tác (transaction) trong CSDL. Theo ngầm định, log file có phần mở rộng là **.LDF**.

Mỗi tập tin CSDL có năm thuộc tính: *tên tập tin logic*, *một tên tập tin vật lý*, *một kích thước (size) ban đầu*, *một kích thước tối đa (maximum size)*, và *gia số tăng kích thước (growth increment)*. Các thuộc tính của mỗi tập tin, theo cùng với những thông tin khác được ghi chú trong bảng hệ thống **sysfiles**, một dòng ứng với mỗi tập tin được dùng trong một CSDL.

Nhóm tập tin (Filegroups)

Các Filegroup cho phép bạn kết nhóm các tập tin nhằm mục đích quản trị và sắp xếp dữ liệu. Các Filegroup có thể cải thiện hiệu năng CSDL bằng cách cho phép một CSDL được tạo ngang qua nhiều đĩa hoặc hệ thống RAID. Bạn có thể tạo các bảng và các chỉ mục trên đĩa được chỉ rõ bằng cách dùng filegroup. Có 3 dạng filegroup.

- **Primary filegroup** bao gồm primary data file và tất cả những tập tin khác không đặt trong những filegroup khác. Các bảng hệ thống (System table) – Dùng định nghĩa các người dùng (user), các đối tượng (object), và các quyền (permission) đối với một CSDL – là được đặt trong primary filegroup của CSDL đó. SQL Server tự động tạo các bảng hệ thống của từng CSDL khi chúng ta tạo CSDL.
- **User-defined filegroups** là bất kỳ các filegroup xác định mặc định bởi người dùng trong suốt tiến trình tạo hoặc hiệu chỉnh CSDL. Một bảng hoặc một chỉ mục có thể được tạo và đặt trong một user-defined filegroup chỉ định.
- **Default filegroup:** Chứa tất cả đối tượng các trang (page) của các bảng và các chỉ mục mà không được chỉ định filegroup khi chúng được tạo ra. Theo mặc nhiên, Default filegroup là primary filegroup. Các thành viên của role db_owner database có thể chuyển đổi trạng thái default từ một filegroup này cho filegroup khác. Tại một thời điểm chỉ có thể có một default filegroup. Nếu default filegroup không được chỉ định thì primary filegroup là default filegroup một cách tự động. Lệnh hiệu chỉnh CSDL (ALTER DATABASE) được dùng để thay đổi default filegroup.

```
ALTER DATABASE database_name MODIFY FILEGROUP filegroup_name DEFAULT
```

Cách cấp phát khoảng không để lưu trữ

Dữ liệu được lưu trữ trong các khối 8Kb liền nhau của không gian của đĩa được gọi là trang (page). Có nghĩa là một CSDL có thể lưu trữ 128 page mỗi megabyte (MB).

Các dòng không thể trải dài trên các page và phải nằm gọn trong 1 page. Tổng số lớn nhất của dữ liệu trong một dòng đơn là 8060 bytes (8192, trừ overhead). Tổng số khoảng không lớn nhất mà có thể sử dụng bởi các dòng trên một trang là 8094 bytes.

Các bảng, các chỉ mục, ... được lưu trữ trong các extent. Một extent có 8 page kề nhau, hoặc 64 KB. Vì vậy, một CSDL có 16 extent mỗi megabyte. Có thể có đến 8 đối tượng nhỏ có thể chia sẻ trong một extent (Mixed Extent). Khi một bảng tăng lên 8 page, nó dùng extent đồng dạng (Uniform extent).

SQL Server dùng 2 kiểu đồ thị định vị (allocation map) để ghi nhận lại định vị của các extent:

- **Global Allocation Map (GAM)**

Các trang GAM ghi nhận các extent đã được cấp phát. Mỗi một GAM kiểm soát 64000 extent, hoặc gần 4 GB dữ liệu. GAM có một bit ứng với mỗi extent trong vùng mà có kiểm soát. Nếu bit đó mang giá trị là 1 thì extent là trống; nếu bit đó mang giá trị 0 thì extent đã được cấp phát.

- Shared Global Allocation Map (SGAM)

Các trang SGAM ghi nhận các extent được sử dụng như là các mixed extent và có ít nhất một trang chưa dùng. Mỗi SGAM kiểm soát 64000 extent, hay gần 4 GB dữ liệu. SGAM có một bit ứng với extent trong vùng mà nó kiểm soát. Nếu bit mang giá trị 1 thì extent được đang sử dụng như là một mixed extent và hiện có trang trống (free page); nếu mang giá trị 0, thì extent đã được dùng như là mixed extent hay nó là một mixed extent mà tất cả các trang đều được dùng.

Mỗi extent có một giá trị bit đặt trong GAM và SGAM dựa trên cơ sở có đang được dùng:

Sử dụng hiện thời của extent	Bit GAM	Bit SGAM
Trống, chưa được dùng	1	0
Uniform extent, hoặc mixed extent đầy	0	0
Mixed extent với các free page	0	1

Các extent được quản lý với một thuật toán đơn giản. Để định một extent đồng dạng, SQL Server tìm trên GAM một bit 1 và đặt nó thành bit 0. Để tìm một mixed extent với những free page, SQL Server tìm trên SGAM một bit 1. Để định một mixed extent, SQL Server tìm trên GAM một bit 1, đặt nó thành 0, và đặt vào bit tương ứng trên SGAM giá trị 1. Một extent trống, SQL Server đảm bảo bit trên GAM mang giá trị 1 và trên SGAM mang giá trị 0

Transaction Log làm việc như thế nào?

Transaction log ghi nhận sự hiệu chỉnh dữ liệu – các câu lệnh INSERT, UPDATE, và DELETE – khi chúng được thi hành. Tiến trình ghi vết ghi nhận lại:

- Một sự thay đổi dữ liệu được gửi từ ứng dụng.
- Khi một sự thay đổi được thực hiện thì các trang dữ liệu ảnh hưởng được tải lên từ tập tin dữ liệu trong bộ nhớ (gọi là *data cache*), nếu các trang không sẵn sàng trong data cache từ truy vấn trước đó.
- Mỗi câu lệnh hiệu chỉnh dữ liệu thì luôn luôn được ghi trong log như nó được tạo. Thay đổi thì luôn luôn ghi nhận lại thành vết và được ghi vào tập tin log (log file) trước khi thay đổi đó được tác động trong CSDL. Kiểu của log này gọi là *write-ahead log*.
- Khi các trang dữ liệu hiện nằm trong data cache, và những trang log được ghi nhận lại trên đĩa trong một tập tin transaction log thì tiến trình checkpoint ghi tất cả các transaction đã hoàn tất (committed transaction) vào CSDL trên đĩa.

Một transaction đơn có thể có nhiều hiệu chỉnh dữ liệu. Mỗi transaction bắt đầu với một lệnh BEGIN TRANSACTION. Nếu ứng dụng hoàn tất tất cả sự hiệu chỉnh dữ liệu một

cách thành công thì transaction kết thúc với lệnh COMMIT TRANSACTION (như là một transaction được nối là transactin hoàn tất- *committed transaction*).

Trong suốt quá trình hoạt động, tiến trình checkpoint đều đặn thường xuyên kiểm tra các transaction đã hoàn tất mà sự hiệu chỉnh dữ liệu chưa được ghi vào tập tin dữ liệu. Tiến trình checkpoint ghi những hiệu chỉnh này vào tập tin dữ liệu và checkpoint các transaction cho biết rằng nó đã được viết vào tập tin dữ liệu chưa.

Nếu hệ thống bị hỏng hóc, tiến trình phục hồi (recovery process) tự động chạy khi SQL Server được khởi động lại. Tiến trình này sử dụng transaction log để quay ngược lại đến các transaction hoàn tất mà chưa từng được "checkpointed" và xoá bỏ đến (roll back) các transaction chưa hoàn tất.

Cơ chế tự động ghi nhận vết trong SQL Server là không là một lựa chọn (option) (có nghĩa là bạn không thể tắt nó đi), tất cả các hiệu chỉnh dữ liệu đều phải đi qua transaction log (Có 2 phương pháp tải một lượng dữ liệu lớn mà có thể được thực hiện mà **không dùng transaction log**, đó là chương trình **bulk copy** và lệnh **SELECT INTO**). Dữ liệu rất quan trọng nên transaction log không bao giờ đầy (full) bởi điều này sẽ ngăn chặn hiệu chỉnh dữ liệu trong CSDL.

4.3 Tạo, hiệu chỉnh cơ sở dữ liệu SQL SERVER

4.3.1 Giới thiệu

Để tạo một CSDL, trước hết bạn phải định nghĩa một tên cho CSDL, kích cỡ của nó, và các tập tin primary data file, secondary data file và file group dùng để lưu trữ nó. Bạn nên xem xét vài nhân tố sau trước khi bạn tạo CSDL:

- Quyền để tạo một CSDL mặc nhiên phải là thành viên của sysadmin và DBCreator fixed server role, mặc dù quyền này có thể gán cho bất kỳ user nào.
- User - người tạo ra CSDL trở thành chủ (owner) của CSDL.
- Có thể có tối đa 32767 CSDL có thể tạo trong một server.
- Tên của CSDL phải đặt theo qui tắt định danh.

Khi tạo CSDL bạn nên chỉ định dung lượng lớn nhất có thể có của một CSDL, điều này sẽ ngăn chặn sự gia tăng không kiểm soát kích thước của CSDL. SQL Server tạo CSDL thông qua 2 bước:

- SQL Server sử dụng một bản sao của CSDL Model để khởi tạo CSDL mới và biến đổi nó.
- Sau đó SQL Server nhồi đầy phần còn lại của CSDL bởi các trang trống.

Các phương pháp tạo, hiệu chỉnh một CSDL của SQL Server.

Cách 1: dùng SQL Enterprise Manager

Cách 2: dùng Create Database Wizard.

Cách 3: dùng câu lệnh CREATE DATABASE.

4.3.2 Tạo cơ sở dữ liệu

Tạo bằng Database Wizard (ở Ô tại cửa sổ Enterprise Manager)

1. Mở rộng **server group**, sau đó nói rộng **server** nơi mà sẽ tạo CSDL.
2. Chọn thực đơn **Tools → Wizards**.

3. Mở rộng **Database**.
4. Nhấp phải chuột tại **Create Database Wizard**.
5. Hoàn tất các bước trong Wizard.

Tạo bằng Enterprise Manager (ở tại cửa sổ Enterprise Manager)

1. Mở rộng **server group**, sau đó mở rộng **server** nơi mà sẽ tạo CSDL.
2. Nhấp nút phải chuột tại nút **Database**, chọn **New DataBase**.
3. Khai báo các thông tin cần thiết, sau đó chọn OK:

Trang General

+ Name: <Tên logic của CSDL>

Trang Data Files

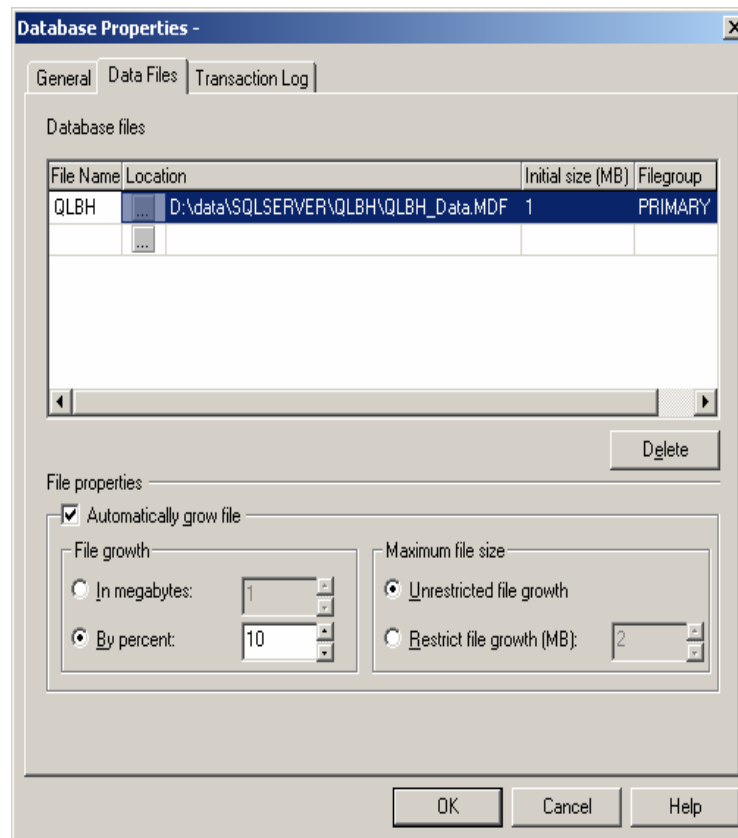
+ Location : <Tên của tập tin Datafile, vị trí lưu tập tin trên đĩa>

+ Initial size: <Kích cỡ khởi tạo CSDL>

+ File Group : <Tên của File Group>

+ File properties: Khai báo một số thuộc tính khác như tỉ lệ gia tăng (File Growth), kích cỡ tối đa (maximum size)

Trang Transaction log: Tương tự như trang Data files nhưng khai báo cho tập tin log.



Hình 23: Hộp thoại xem thuộc tính của SQL Server

Tạo bằng câu lệnh Create Database (gõ lệnh trong cửa sổ Query Analyzer).

Cú pháp

```
CREATE DATABASE database_name
[ ON
  [ < filespec > [...n]
  [, < filegroup > [...n]
]
[ LOG ON { < filespec > [...n] } ]
[ COLLATE collation_name ]
[ FOR LOAD | FOR ATTACH ]

< filespec > ::=
[ 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]
```

Thực hiện:

- (1) Gõ lệnh trong cửa sổ Query Analyzer
- (2) Gọi thực thi câu lệnh

Ví dụ: Tạo CSDL có tên là SalesDB, tập tin dữ liệu tên là SalesDB_dat.mdf đặt trong C:\Data, kích cỡ khởi tạo là 10MB, kích thước tối đa là 50MB, tỉ lệ gia tăng là 5MB, và tập tin vết tên là SalesDB_log.ldf đặt trong C:\Data, kích thước khởi tạo là 5MB, kích thước tối đa là 25MB, tỉ lệ gia tăng là 10%.

```
CREATE DATABASE SalesDb
ON
(NAME = SalesDb_dat,
  FILENAME = 'c:\data\salesDB_dat.mdf',
  SIZE = 10,
  MAXSIZE = 50,
  FILEGROWTH = 5)
LOG ON
(NAME = 'SalesDb_log',
  FILENAME = 'c:\data\salesDB_log.ldf',
  SIZE = 5MB,
  MAXSIZE = 25MB,
  FILEGROWTH = 10%)
GO
```

Lưu ý:

- Thư mục **Data** phải hiện hữu trong C:\

- Sao khi gõ câu lệnh, đánh dấu chọn khối lệnh và nhấn F5 để thực thi.

4.3.3 Thao tác trên cơ sở dữ liệu của SQL Server

Khi làm việc với CSDL, bạn có thể thực hiện trực tiếp trong cửa sổ Enterprise Manager hoặc dùng các câu lệnh T-SQL trong cửa sổ Query Analyzer.

4.3.3.1 Kiểm tra sự tồn tại của cơ sở dữ liệu

Cách 1: Tại cửa sổ EM, kiểm tra sự tồn tại của CSDL trong nhánh DataBase.

Cách 2: Tại cửa sổ QA, thực hiện câu lệnh Sp_helpdb <Tên CSDL>

Ví dụ: Sp_helpDB SalesDB

4.3.3.2 Xem, thay đổi thuộc tính của cơ sở dữ liệu.

Cách 1: Dùng Enterprise Manager

Mở nút **DataBase**, R_Click tại tên CSDL cần xem hoặc hiệu chỉnh

Chọn **Properties**

Thay đổi tùy ý

Cách 2: Dùng T_SQL

Cú pháp

```
ALTER DATABASE database
{ ADD FILE < filespec > [...n] [ TO FILEGROUP filegroup_name ]
| ADD LOG FILE < filespec > [...n]
| REMOVE FILE logical_file_name
| ADD FILEGROUP filegroup_name
| REMOVE FILEGROUP filegroup_name
| MODIFY FILE < filespec >
| MODIFY NAME = new_dbname
| MODIFY FILEGROUP filegroup_name {filegroup_property | NAME =
new_filegroup_name }
/SET < optionspec > [...n] [ WITH < termination > ]
| COLLATE < collation_name >
}
```

[< filespec > ::=](#)

```
(NAME = logical_file_name
[ , NEWNAME = new_logical_name ]
[ , FILENAME = 'os_file_name' ]
[ , SIZE = size ]
[ , MAXSIZE = { max_size | UNLIMITED } ]
[ , FILEGROWTH = growth_increment ])
```

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

```
ALTER DATABASE SalesDb
MODIFY FILE (NAME='salesdb_log', size=10MB)
```

Ví dụ 2: Bổ sung thêm một tập tin dữ liệu SalesDB_data2

```
ALTER DATABASE SalesDB
ADD FILE (NAME=SalesDB_data2,
FILENAME='C:\data\SalesDb2.mdf', SIZE=10 MB,
MAXSIZE=20MB)
```

4.3.3.3 Xóa cơ sở dữ liệu.

Cách 1: Dùng Enterprise Manager

Nhấn nút phải chuột tại tên CSDL, chọn Delete

Cách 2: Dùng câu lệnh T-SQL

```
DROP DATABASE database_name [...n]
```

Ví dụ: DROP DATABASE SalesDB

4.3.3.4 Đổi tên cơ sở dữ liệu.

Dùng hàm sp_renamedb theo cú pháp sau

```
sp_renamedb [ @dbname = ] 'old_name', [ @newname = ] 'new_name'
```

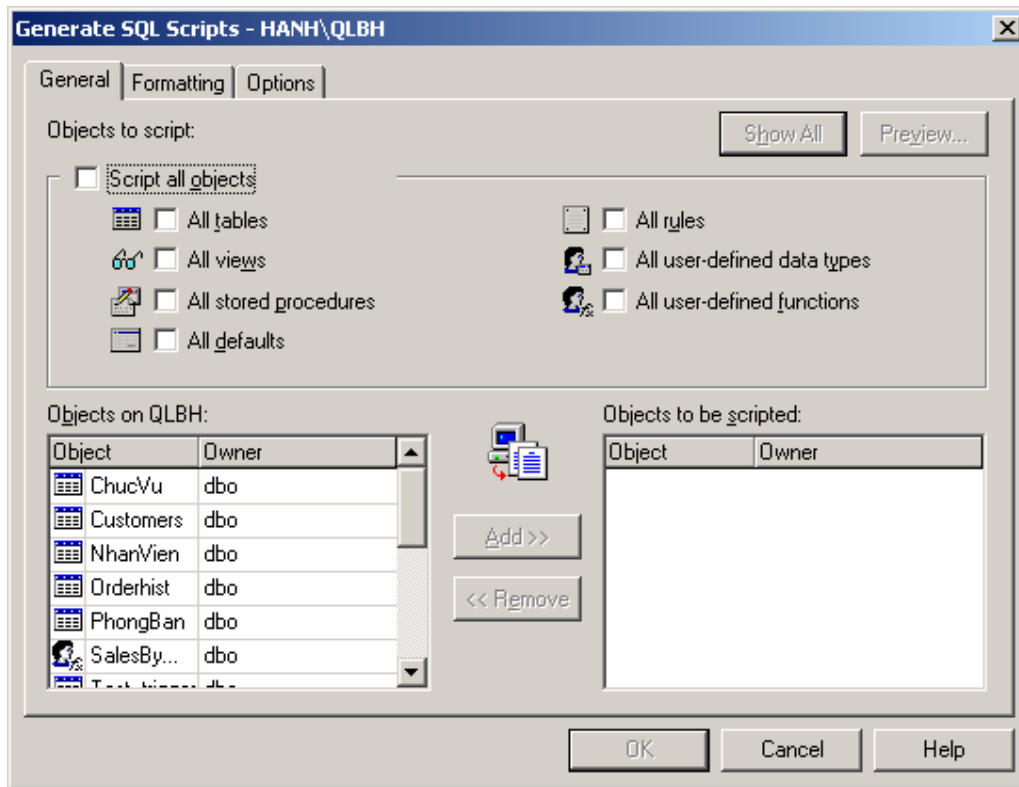
4.3.3.5 Tạo một script cho CSDL và các đối tượng của CSDL.

Đôi khi, bạn cần sao chép cấu trúc của CSDL hoặc cấu trúc các đối tượng của CSDL, thì bạn sẽ thực hiện tạo script cho chúng. Khi có script bạn sẽ mở và thực thi đoạn script tại cửa sổ Query Analyzer để tái tạo lại các đối tượng.

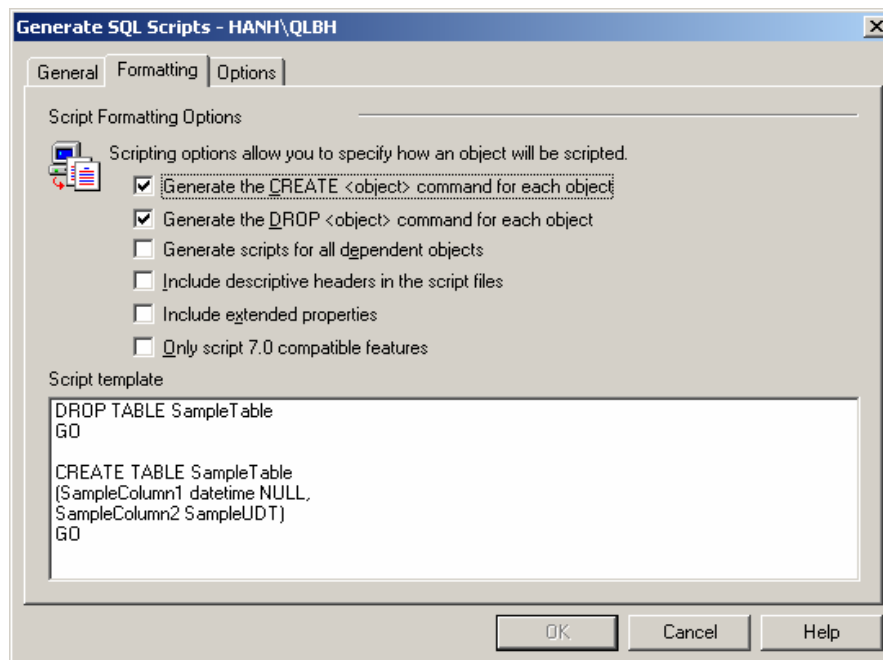
Các bước thực hiện:

- Mở rộng một server group; mở rộng một server.
- Mở rộng nhánh DataBase, nhấp phải tại CSDL muốn tạo script, trở đến **All Tasks**, nhấp **General SQL SQL Script...**
- Khai báo các lựa chọn thích hợp.
 - + Trang General: chọn đối tượng cần tạo script.
 - + Trang Formatting: chọn các tùy chọn định dạng script.
 - *Generate the CREATE <object> command for each object*: Tạo script theo cách sử dụng định nghĩa đang có của nó. Lựa chọn này được chọn theo mặc định.
 - *Generate the DROP <object> command for each object*: Bổ sung vào script cho mỗi đối tượng câu lệnh drop khi tạo script. Lựa chọn này được chọn theo mặc định.
 - *General scripts for all dependent objects*: Tự động tạo các script cho các đối tượng có liên quan với đối tượng đang tạo script.
 - *Include descriptive headers in the script files*: Thêm lời chú thích được bổ sung vào tập tin script cho mỗi đối tượng tạo script.

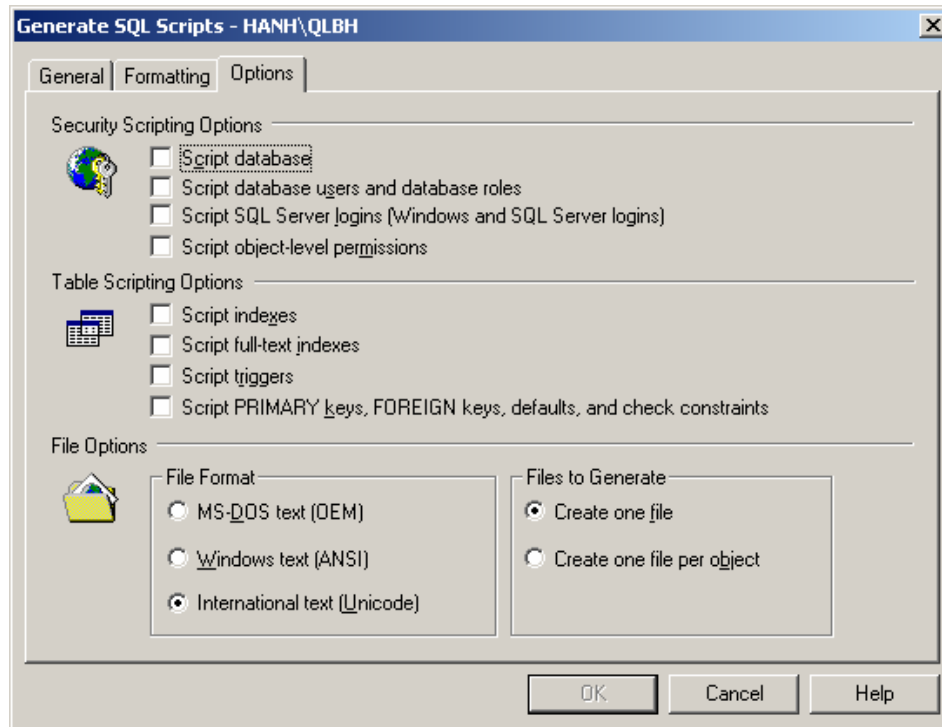
- + Trang options: hãy chọn các tùy chọn *security-related*, *table-related*, và *script file-related*
- Trên trang **General**, nhấp **PreView** để xem trước nội dung của script được tạo ra.



Hình 24: Trang General của hộp thoại phát sinh script các đối tượng của CSDL



Hình 25: Trang Formatting của hộp thoại phát sinh script các đối tượng của CSDL



Hình 26: Trang Option của hộp thoại phát sinh script các đối tượng của CSDL

Sử dụng Script vừa tạo:

- Chuyển đến vị trí mới cần tải tạo lại CSDL/ các đối tượng CSDL.
- Vào cửa sổ Query Analyzer, mở tập tin Script.
- Hiệu chỉnh các vị trí vật lý nếu cần.
- Cho thực thi đoạn Script
- Kiểm tra kết quả.

BÀI 5: KIỂU DỮ LIỆU – LÀM VIỆC VỚI BẢNG

5.1 Kiểu dữ liệu (data type)

Kiểu dữ liệu là một đặc tính của một cột (Column). Nó định rõ loại dữ liệu và dạng dữ liệu được nhập vào cột.

Có 2 nhóm:

- **System-Supplied datatype:** Các kiểu dữ liệu cơ bản được hỗ trợ bởi SQL Server.
- **User-defined datatype:** Các kiểu dữ liệu của người dùng tự định nghĩa dựa trên các kiểu dữ liệu cơ bản.

5.1.1 System-Supplied Datatype.

System-Supplied Datatype là kiểu dữ liệu cơ bản được hỗ trợ bởi SQL Server. Các đối tượng lưu chứa dữ liệu đều có một kiểu dữ liệu để lưu, các đối tượng đó có thể là

- Các cột (Column) trong các bảng.
- Các tham số (parameters) trong stored procedures.
- Các biến (Variables) trong stored procedure, function, script, batch.
- Các hàm T_SQL trả về một hoặc nhiều giá trị thuộc một kiểu dữ liệu nào đó.

Ta có thể dùng các kiểu dữ liệu để tạo các ràng buộc toàn vẹn dữ liệu. Ví dụ cột TENNV thì không thể được định nghĩa với kiểu dữ liệu là Date, vì cột Date chỉ chấp nhận giá trị ngày.

Khi ta gán kiểu dữ liệu cho một đối tượng nào đó thì ta cần quan tâm đến các tính chất sau:

- Loại dữ liệu được chứa đựng bởi đối tượng.
- Chiều dài lưu trữ giá trị hoặc là kích cỡ của nó.
- Tính đúng của số (đối với các kiểu số).

Các kiểu dữ liệu cơ bản:

Loại	Kiểu dữ liệu cơ sở	Kích cỡ	Vùng giá trị	Mô tả
Binary	Binary	8 KB	"0"... "9", "a"... "f", "A"... "F"	Chứa các bit thông tin
	Varbinary	8 KB	"0"... "9", "a"... "f", "A"... "F"	
	Image	2 ³¹ -1 bytes	2 ³¹ -1 bytes	Dữ liệu hình ảnh
Character	Char	255 bytes	1..8000 ký tự	Ký tự hoặc chuỗi
	Varchar	255 bytes	1..8000 ký tự	Ký tự hoặc chuỗi
	Text	2147483647 bytes	2 ³¹ -1 ký tự (2147483647)	Ký tự hoặc chuỗi
Date and Time	Datetime	8 bytes	01/01/1753->31/12/9999	Chuỗi biểu diễn ngày giờ
	Smalldatetime	4 bytes	1/1/1900 -> 6/6/2079	Chuỗi biểu diễn ngày giờ
Decimal	Decimal	17 bytes	-10 ³⁸ -1 -> 10 ³⁸ -1	Số thực
	Numeric	17 bytes	-10 ³⁸ -1 -> 10 ³⁸ -1	Số thực
Floating point	Float	8 bytes	-1.79E+308 -> 1.79E+308	Số thực
	Real	4 bytes	-3.40E+38 -> 3.40E+38	Số thực
Integer	Bigint	8 bytes	-2 ⁶³ -> 2 ⁶³	Số nguyên

	Int	4 bytes	$-2^{31} \rightarrow 2^{31}-1$	Số nguyên
	Smallint	2 bytes	$-2^{15} \rightarrow 2^{15}-1$	Số nguyên
	Tinyint	1 bytes	0..255	Số nguyên
Monetary	Money	8 bytes	$-2^{63} \rightarrow 2^{63}-1$	Dữ liệu tiền tệ
	Smalmoney	4 bytes	-214748.3648 \rightarrow 214748.3648	Dữ liệu tiền tệ
Special	Bit	1 bytes	0 hoặc 1	Dữ liệu có một trong hai trạng thái 0 hoặc 1
	Cursor	Kiểu DL cho biến hoặc giá trị trả về của procedure, tham chiếu đến 1 mẫu tin		
	Timestamp	8 bytes	Chuỗi có dạng: 0x000000100000a90	Theo dõi mẫu tin nào bị thay đổi dữ liệu
	Uniqueidentifier	16 bytes	Số thập lục phân	
	SQL_variant	Là kiểu dữ liệu có thể chứa bất kỳ loại dữ tùy ý của SQL Server ngoại trừ text , ntext , image , and the timestamp data type		
	Table			
Unicode	Nchar		4000 ký tự	Ký tự hoặc chuỗi
	Nvarchar		4000 ký tự	Ký tự hoặc chuỗi
	Ntext		$2^{30}-1$ ký tự	Ký tự hoặc chuỗi

5.1.2 User-defined datatype.

Người sử dụng có thể dựa yêu cầu cần lưu trữ và các kiểu dữ liệu cơ bản để định nghĩa ra một kiểu dữ liệu của người dùng dùng để lưu trữ một dữ liệu đặc biệt nào đó. SQL Sever cho phép bạn cải tiến các kiểu dữ liệu để đảm bảo tính nhất quán khi làm việc trong môi trường dữ liệu đa dạng trong các bảng hay các CSDL khác nhau.

- User-defined data type không cho phép bạn định nghĩa các kiểu dữ liệu phức hoặc có cấu trúc.
- Mỗi một User-defined data type có thể được định nghĩa riêng cho một CSDL hoặc cho toàn bộ các CSDL. Nếu User-defined data type được định nghĩa trong CSDL Master thì nó được dùng chung cho toàn bộ các CSDL.
- Các User-defined data type mà bạn tạo trong CSDL model thì sẽ có trong tất cả các CSDL mới tạo một cách tự động.
- Mỗi user-defined data type được lưu thành một mẫu tin trong bảng **systypes**.
- Bạn có thể tạo và xóa user-defined data type bằng các thủ tục hệ thống. Tên của kiểu dữ liệu phải tuân thủ qui tắc định danh và phải là duy nhất trong mỗi CSDL. Định nghĩa mỗi user-defined data type trong giới hạn của các kiểu dữ liệu cơ bản. Phải chỉ định mặc định là chấp nhận giá trị NULL hay NOT NULL khi đối tượng không có giá trị.

Tạo một User-Defined Data Type

Dùng thủ tục hệ thống **sp_addtype** để tạo một user-defined data type.

```
sp_addtype type, system_data_type ['NULL' | 'NOT NULL']
```

Ví dụ 1: Tạo kiểu dữ liệu tên là **isbn** với kiểu dữ liệu cơ bản là **smallint** và **không chấp nhận giá trị Null**

```
EXEC sp_addtype isbn, 'smallint', NOT NULL
```

Ví dụ 2: Tạo kiểu dữ liệu tên là **zipcode** với kiểu dữ liệu cơ bản là **char**, **độ dài tối đa là 10** và **chấp nhận giá trị Null**

```
EXEC sp_addtype zipcode, 'char(10)', NULL
```

Ví dụ 3: Tạo kiểu dữ liệu tên là **longstring** với kiểu dữ liệu cơ bản là **varchar**, **độ dài tối đa là 63** và **chấp nhận giá trị Null**

```
EXEC sp_addtype longstring, 'varchar(63)', NULL
```

Xoá một User-Defined Data Type: dùng thủ tục hệ thống *sp_droptype* để xoá một user-defined data type từ bảng systypes. Một user-defined data type không thể xoá được nếu nó được tham chiếu bởi các bảng và những đối tượng khác.

```
Sp_droptype type
```

Ví dụ:

```
EXEC sp_droptype isbn
```

Xem các user-defined data types trong CSDL hiện hành: dùng thủ tục **sp_help** hoặc truy vấn trong **information_schema.domains**

Ví dụ:

```
Use SalesDB
Sp_help
hoặc
SELECT domain_name, data_type,
character_maximum_length
FROM information_schema.domains
ORDER BY domain_name
```

5.2 Làm việc với bảng của SQL Server

Bảng là một đối tượng của CSDL và là nơi chứa đựng các dữ liệu về một thực thể nào đó ví dụ Khách hàng, đơn đặt hàng, tồn kho,... Một bảng là một tập hợp các cột (Column). Mỗi một cột đại diện cho một thuộc tính của dữ liệu trong bảng.

Khi bạn tạo một table, bạn phải chỉ định rõ tên của bảng, tên cột, kiểu dữ liệu của cột. Tên cột phải duy nhất trong một bảng, có thể dùng tên trùng nhau ở các bảng khác nhau trong cùng một CSDL. Phải chỉ rõ một kiểu dữ liệu cho mỗi cột và những lựa chọn khác nếu cần. Bạn có thể tạo:

- Tối đa 2 tỉ table cho mỗi CSDL.
- Tối đa 1024 cột trong mỗi bảng.
- 8060 bytes mỗi dòng (kiểu image và text dùng 16 bytes mỗi dòng)

Hai cách cơ bản làm việc trên bảng: **Enterprise Manager (tự nghiên cứu)**, **câu lệnh T-SQL**.

5.2.1 Tạo một bảng mới

Cú pháp lệnh

```
CREATE TABLE
[ database_name.[ owner ] . | owner. ] table_name
({ < column_definition >
```

```

    | column_name AS computed_column_expression
    | < table_constraint > ::= [ CONSTRAINT constraint_name ]
      | [ { PRIMARY KEY | UNIQUE } [,...n]
    )
[ ON { filegroup | DEFAULT } ]
[ TEXTIMAGE_ON { filegroup | DEFAULT } ]

< column_definition > ::= { column_name data_type }
[ COLLATE < collation_name > ]
[ [ DEFAULT constant_expression ]
  | [ IDENTITY [ (seed, increment) [ NOT FOR REPLICATION ] ] ]
]
[ ROWGUIDCOL ]
[ < column_constraint > ] [ ...n ]

< column_constraint > ::= [ CONSTRAINT constraint_name ]
{ [ [ NULL | NOT NULL ]
  | [ { PRIMARY KEY | UNIQUE }
    [ CLUSTERED | NONCLUSTERED ]
    [ WITH FILLFACTOR = fillfactor ]
    [ ON { filegroup | DEFAULT } ] ] ]
| [ [ FOREIGN KEY ]
  REFERENCES ref_table [ (ref_column) ]
  [ ON DELETE { CASCADE | NO ACTION } ]
  [ ON UPDATE { CASCADE | NO ACTION } ]
  [ NOT FOR REPLICATION ]
]
| CHECK [ NOT FOR REPLICATION ]
  (logical_expression)
}

< table_constraint > ::= [ CONSTRAINT constraint_name ]
{ [ { PRIMARY KEY | UNIQUE }
  [ CLUSTERED | NONCLUSTERED ]
  { (column [ ASC | DESC ] [,...n]) }
  [ WITH FILLFACTOR = fillfactor ]
  [ ON { filegroup | DEFAULT } ]
]
| FOREIGN KEY
  [ (column [,...n]) ]
  REFERENCES ref_table [ (ref_column [,...n]) ]
  [ ON DELETE { CASCADE | NO ACTION } ]
  [ ON UPDATE { CASCADE | NO ACTION } ]
  [ NOT FOR REPLICATION ]
| CHECK [ NOT FOR REPLICATION ]
  (search_conditions)
}

```

Ví dụ 1:

```

USE SalesDb
GO
CREATE TABLE Employees

```

```
(
    EmployeeID SMALLINT IDENTITY(1,1) NOT NULL,
    FirstName NVARCHAR(30) NOT NULL,
    LastName NVARCHAR(30) NOT NULL,
    Address1 NVARCHAR(60) NOT NULL,
    City NVARCHAR(15) NOT NULL,
    State CHAR(2) NOT NULL,
    Phone VARCHAR(24) NOT NULL,
    DOB DATETIME NOT NULL,
    HireDate DATETIME NOT NULL,
    PositionID TINYINT NOT NULL
)
```

5.2.2 Hiệu chỉnh bảng

Thao tác cơ bản hiệu chỉnh bảng gồm thêm cột, xóa cột, thay đổi thuộc tính của cột. Để thực hiện ta sẽ dùng câu lệnh ALTER TABLE¹.

Thêm các cột

```
ALTER TABLE <TableName>
ADD <Column Definition>[,...n]
```

Ví dụ: Thêm cột Address vào bảng Employees

```
ALTER TABLE Employees
ADD Address2 NVARCHAR(6) NOT NULL DEFAULT 'N/A'
```

Lưu ý: Nếu bảng đã có sẵn dữ liệu và cột thêm vào được định nghĩa là NOT NULL thì ta phải điền dữ liệu của các dòng ở cột mới thêm vào là một giá trị mặc định nào đó để tránh giá trị Null.

Xóa các cột

```
ALTER TABLE <TableName>
DROP COLUMN <Column name>[,...n]
```

Ví dụ:

```
ALTER TABLE Employees
DROP COLUMN Address2
```

Lưu ý: Lệnh trên sẽ không thực hiện được vì khi tạo cột Address2 ta đã khai báo giá trị mặc định nên SQL Server đã tạo ra một đối tượng 'Default Constraint'. Do đó, muốn xóa cột thì phải xóa tất cả các đối tượng liên quan đến cột cần xóa rồi mới xóa cột đó.

Ví dụ:

```
ALTER TABLE Employees
DROP CONSTRAINT DF_Employees_Addre_1372D2FE
ALTER TABLE Employees
DROP COLUMN Address2
```

¹ Bạn tham khảo cú pháp lệnh đầy đủ trong Books-Online

Lưu ý: `DF_Employees_Addre_1372D2FE` là tên của Default Constraint do SQL Server tự đặt.

Thay đổi kiểu dữ liệu cho cột

```
ALTER TABLE <TableName>
```

```
ALTER COLUMN <Column Name NewDatatype>
```

Ví dụ:

```
ALTER TABLE Employees
```

```
ALTER Address NVARCHAR(20)
```

Xoá toàn bộ dữ liệu trong Table

```
TRUNCATE TABLE <Name Table>
```

Ví dụ:

```
TRUNCATE TABLE Employees
```

Lưu ý: Nếu bảng muốn xóa là một bảng con (child table) thì bạn có thể xóa dữ liệu của nó bất kỳ lúc nào bạn thích, nhưng nếu nó là một bảng cha (Parent Table) thì bạn phải xóa dữ liệu ở bảng con trước, kế tiếp xóa khoá ngoại (Foreign key constraint) giữa 2 bảng, cuối cùng mới xóa dữ liệu ở bảng cha.

5.2.3 Xóa bảng khỏi cơ sở dữ liệu.

Xóa một bảng là gỡ bỏ định nghĩa bảng và tất cả dữ liệu, cũng như các quyền (permission) định cho bảng đó.

Trước khi xóa một bảng, bạn phải gỡ bỏ tất cả các phụ thuộc giữa bảng đó và những đối tượng khác.

```
DROP TABLE <Name Table>
```

Ví dụ:

```
DROP TABLE Employees
```

5.3 Bảng tạm (Temporary Tables).

Bạn cũng có thể tạo các bảng tạm. Các bảng tạm tương tự như các bảng bình thường, ngoại trừ việc các bảng tạm thời được chứa trong CSDL TempDb và được xóa một cách tự động khi không còn sử dụng nữa.

Có hai loại bảng tạm: bảng tạm cục bộ (Local) và bảng tạm tổng thể (global). Chúng khác nhau về tên, tính hiển thị, và tính có sẵn.

Bảng tạm cục bộ: bảng tạm cục bộ có một dấu # là ký tự đầu tiên trong tên của chúng; chúng chỉ hiển thị đối với nối kết hiện hành dành cho người sử dụng, và chúng được xóa khi người dùng ngắt nối kết với các thể hiện của SQL Server.

Ví dụ: Tạo bảng tạm tên là #MyLocalTempTable

```
CREATE TABLE #MyLocalTempTable
```

```
(
  ID INT PRIMARY KEY,
  ColA VARCHAR(30) NULL,
  ColB VARCHAR(30) NULL,
  ColC VARCHAR(30) NULL
)
```

Nếu một bảng tạm được tạo trong tiến trình của một thủ tục thì bảng tạm đó sẽ bị xóa khi thủ tục hoàn tất. Trong trường hợp một tiến trình A gọi thủ tục B mà B có tạo một bảng tạm thì chỉ có B mới dùng được bảng tạm đó còn tiến trình A không nhìn thấy bảng tạm đó. Trong trường hợp thủ tục B được gọi cùng một lúc bởi nhiều tiến trình khác nhau thì nó sẽ có nhiều bảng tạm giống nhau khi đó SQL Server sẽ giải pháp giải quyết bằng cách thêm một hậu tố (suffix) vào tên của các bảng tạm cùng tên sao cho các bảng này có tên là khác nhau (tên của bảng tạm tối đa là 116 ký tự)

Bảng tạm toàn cục: Các bảng tạm toàn cục thì có 2 dấu ## là ký tự đầu tiên trong tên của chúng; chúng hiển thị đối với bất kỳ người sử dụng nào sau khi chúng được tạo; và chúng được xóa khi tất cả những người sử dụng đang tham chiếu table ngắt kết nối với SQL Server.

Ví dụ:

```
CREATE TABLE ##MyGlobalTempTable
(
    ID INT PRIMARY KEY,
    ColA VARCHAR(30) NULL,
    ColB VARCHAR(30) NULL,
    ColC VARCHAR(30) NULL )
```


BÀI 6: TOÀN VỆN DỮ LIỆU

6.1 Giới thiệu toàn vẹn dữ liệu (data Integrity)

Toàn vẹn dữ liệu là đề cập đến trạng thái của tất cả các giá trị dữ liệu lưu trữ trong CSDL là đúng. Nếu dữ liệu không đúng mà đã được lưu trữ trong CSDL thì được gọi là vi phạm toàn vẹn dữ liệu.

Các bảng trong CSDL của SQL Server có một số loại toàn vẹn dữ liệu khác nhau. Ví dụ định nghĩa NOT NULL, định nghĩa DEFAULT, thuộc tính IDENTITY, CONSTRAINTS, RULES, TRIGGERS, INDEXES.

Xác định đúng kiểu dữ liệu của cột hoặc biến cũng là một cách thúc ép tính toàn vẹn dữ liệu. Ví dụ bạn không thể chấp nhận giá trị của cột CustomName là một giá trị dạng ngày giờ cũng như ngược lại. Để tạo hoặc thêm các ép thỏa toàn vẹn dữ liệu, chúng ta có thể thực hiện trong các câu lệnh.

Định nghĩa ràng buộc:

Create Table ... : Định nghĩa trong lúc thiết kế cấu trúc bảng.

Alter Table... : Định nghĩa trong khi hiệu chỉnh bảng.

Để kiểm tra hoặc xem các toàn vẹn dữ liệu

Sp_HelpConstraint <Tên Table>

Hoặc

Bật cửa sổ **Object Browser** của **Query Analyzer**, mở nhánh Constraint của từng bảng.

Xóa các toàn vẹn dữ liệu

ALTER TABLE <Tên Table>

DROP CONSTRAINT <Tên Constraint>

6.2 Tìm hiểu các toàn vẹn dữ liệu.

6.2.1 Định nghĩa NULL/NOT NULL

Một giá trị không biết, chưa xác định chúng ta quy là giá trị **NULL**. Khả năng null của một cột được xem là khả năng của cột chấp nhận hoặc không chấp nhận giá trị null.

- Bạn có thể định nghĩa giá trị của một cột không là null.
- Một giá trị null không đồng nhất với giá trị 0, khoảng trắng, chuỗi rỗng.
- Null có nghĩa là không có thao tác nhập nào thực hiện được.
- Sự tồn tại của Null thường cho biết rằng giá trị chưa được biết rõ hay chưa xác định. Chẳng hạn, một giá trị Null trong cột *price* của bảng *Items* không có nghĩa là mặt hàng này không có giá hoặc là giá bằng 0.

Nói chung, hãy tránh chấp nhận giá trị Null bởi vì chúng gây ra nhiều phức tạp hơn trong các truy vấn cũng như cập nhật dữ liệu. Việc chỉ định một cột không chấp nhận giá trị Null có thể giúp duy trì tính toàn vẹn dữ liệu.

Thông thường để khai báo một cột có thể chấp nhận giá trị null, chúng ta sẽ khai báo trong khi định nghĩa hoặc hiệu chỉnh cột. Tức là dùng trong câu lệnh Create Table/Alter Table.

Ví dụ:

```
USE SalesDb
GO
DROP TABLE Product_Info
GO
CREATE TABLE Product_Info
(
    Product_ID      smallint NOT NULL,
    Product_Name    char(20) NOT NULL,
    Description     char(30) NULL,
    Price           smallmoney NOT NULL
)
GO
```

6.2.2 Giá trị mặc định (Default Values)

Trong số các đặc tính của cột, chúng ta xét thấy giá trị có thể null và giá trị mặc định. Cả hai đặc tính này định ra giá trị chèn vào một cột khi nó không được chỉ định trong câu lệnh INSERT. Các trường hợp này có thể xảy ra khi giá trị cột không được đưa vào trong câu lệnh INSERT:

- Khi cột được định nghĩa như là chấp nhận giá trị NULL và không có giá trị mặc định, giá trị của cột là NULL.
- Khi cột được định nghĩa không chấp nhận giá trị NULL và không có giá trị mặc định, một lỗi sẽ xảy ra.
- Khi cột có một giá trị mặc định

Như vậy, mỗi một cột trong một mẫu tin của bảng đều phải chứa một giá trị, ngay cả khi giá trị đó là NULL. Có những trường hợp bạn cần phải tải một hàng dữ liệu vào một bảng nhưng bạn không biết giá trị dành cho cột hay giá trị này không tồn tại. Nếu cột chấp nhận các giá trị Null, bạn có thể tải hàng có giá trị Null. Thông thường, các cột chấp nhận giá trị Null có thể không phải là các cột cần thiết nên giải pháp tốt hơn hết là ấn định một giá trị mặc nhiên (không nhập giá trị vào thì cột sẽ chấp nhận giá trị mặc định). Việc đó chính là định nghĩa DEFAULT cho cột ở những nơi thích hợp. Chẳng hạn, người ta thường chỉ định 0 là giá trị mặc định cho các cột số, hoặc N/A là giá trị mặc định cho các cột chuỗi khi không có giá trị nào được chỉ định).

Khi bạn nhập vào một mẫu tin của bảng có một định nghĩa Default dành cho một cột bạn đang gián tiếp hướng dẫn SQL Server nhập một giá trị mặc định trong cột khi bạn không chỉ định một giá trị cho cột đó.

SQL Server 2000 có hai cách để triển khai các giá trị mặc định cho các cột: Default Constraint và Default Object.

6.2.2.1 Default Constraint

Default constraint có thể được tạo tại thời điểm tạo bảng, thêm sau khi bảng được tạo.

- Giá trị Default được dùng để gán giá trị hằng số cho một cột.
- Chỉ có một giá trị Default có thể được tạo cho một cột.

- Các cột **TIMESTAMP**, **IDENTITY** và **ROWGUIDCOL** không thể có default constraint, vì giá trị của chúng đã tự động xác định.
- Giá trị default có thể là một hằng số; một hàm hệ thống, chẳng hạn `Getdate()`; một biến toàn cục, như `@@trancount`; hoặc một hàm do người dùng định nghĩa.

Khai báo default constraint

- **Định nghĩa Default constraint trong khi tạo bảng**

```
CREATE TABLE tablename(
    columnname datatype [NULL | NOT NULL]
    [CONSTRAINT constraintname] DEFAULT expression
    [...])
```

- **Định nghĩa Default constraint đối với một bảng đã tồn tại.**

```
ALTER TABLE tablename
    ADD [ CONSTRAINT constraintname ] DEFAULT expression FOR columnname
```

Ví dụ 1: Tạo bảng Events với các default constraint

```
CREATE TABLE Events
( EventID int IDENTITY (1, 1) NOT NULL ,
  EventType nvarchar (10) NOT NULL,
  EventTitle nvarchar (100) NULL ,
  EventDescription ntext NULL ,
  EventLanguage nvarchar (2) NULL ,
  EventDate smalldatetime NULL DEFAULT GETDATE(),
  EventEndDate smalldatetime NULL DEFAULT DATEADD(day,
1,
GETDATE()), EventCreator nvarchar (50) NOT NULL
DEFAULT SYSTEM_USER
)
```

Ví dụ 2:

- Tạo bảng Events không có default constraint

```
CREATE TABLE Events
( EventID int IDENTITY (1, 1) NOT NULL,
  EventType nvarchar (10) NOT NULL,
  EventTitle nvarchar (100) NULL,
  EventDescription ntext NULL,
  EventLanguage nvarchar (2) NULL,
  EventDate smalldatetime NULL,
  EventEndDate smalldatetime NULL,
  EventCreator nvarchar (50) NOT NULL
)
```

- Thêm các default constraint cho bảng Events

```
ALTER TABLE Events
ADD DEFAULT 'Party' FOR EventType
-----
ALTER TABLE Events
ADD CONSTRAINT EVenDate_DF DEFAULT GETDATE() FOR
EventDate
```

Kiểm tra constraint

```
Sp_helpConstraint Events
--- chèn một mẫu tin trống vào bảng Events
INSERT Events DEFAULT VALUES
SELECT * FROM Events
```

Kết quả

	EventID	EventType	EventTitle	EventDescription	EventLanguage
1	1	Party	NULL	NULL	NULL

EventDate	EventEndDate	EventCreator
2004-02-15 01:28:00	2004-02-16 01:28:00	sa

Xoá default constraint

```
ALTER TABLE Events
DROP CONSTRAINT DF__Events__EventTyp__7E6CC920

ALTER TABLE Events
DROP CONSTRAINT EVenDate_DF
```

6.2.2.2 Default Object

Default object là một cách khác để định nghĩa một giá trị mặc định cho một cột. Các Default Object được gọi là “default” có đầu tiên trong phiên bản 2000 của SQL Server. Các Default không là một phần của toàn vẹn khai báo bởi vì chúng không là một của cấu trúc bảng; chúng thực sự là một phần của lược đồ CSDL.

Quá trình khai báo một Default như sau:

- Định nghĩa Default.
- Kết Default vào cột của bảng hoặc kiểu dữ liệu.
- Nếu muốn dùng Default thì sẽ gỡ bỏ khỏi cột hoặc kiểu dữ liệu.
- Không cần Default nữa thì xoá khỏi CSDL

Định nghĩa default

```
CREATE DEFAULT default AS constant_expression
```

Kết đính default với cột:

```
sp_binddefault defaultname, tablename.columnname
```

Kết dính default với user-defined datatype

`sp_bindefault defaultname, datatypename [, futureonly]`

Futureonly chỉ định rằng các cột đã tồn tại có liên quan đến kiểu dữ liệu sẽ không kế thừa giá trị mặc định mới. Cờ này chỉ có thể dùng khi kết giá trị mặc định cho kiểu dữ liệu.

Ví dụ:

```
CREATE DEFAULT CalifDef AS 'CA'
GO
sp_bindefault 'CalifDef', 'Orders.ShipRegion'
```

Gỡ bỏ kết dính một default với cột

`sp_unbindefault tablename.columnname`

Gỡ bỏ kết dính một default với User-defined datatype

`sp_unbindefault datatypename [, futureonly]`

Xóa một Default

`DROP DEFAULT defaultname`

Lưu ý: Chỉ xóa được những Default không được kết với cột hoặc kiểu dữ liệu.

6.2.3 Thuộc tính Identity:

- Identity là một thuộc tính của cột, nó không là một constraint. Tuy nhiên, Identity dùng để ràng buộc sự tồn tại dữ liệu.
- Một bảng chỉ có duy nhất một cột kiểu Identity.
- Kiểu dữ liệu của cột Identity phải là bigint, int, smallint, hoặc tinyint.
- Giá trị của cột Identity sẽ tự động tăng.

Một cột Identity được tạo khi bảng được tạo bằng lệnh Create Table, hoặc khi hiệu chỉnh cột trong bảng bằng lệnh Alter table.

Ví dụ:

```
CREATE TABLE Table1
( ID INT IDENTITY,
  FirstName VARCHAR(30) NOT NULL,
  LastName VARCHAR(30) NOT NULL
)
GO
---Chèn dữ liệu vào Table1, không cần đưa giá trị
cho cột ID
INSERT Table1 (FirstName, LastName)
VALUES ('Minh', 'Thu')
```

Tuy nhiên, đôi khi bạn cần chỉ định giá trị cho cột có định nghĩa Identity, bạn thực hiện tuần tự các bước sau: Bật chế độ chèn dữ liệu cho cột Identity cho bảng, chèn dữ liệu, tắt chế độ chèn cho cột Identity nếu cần.

Ví dụ:

```
SET IDENTITY_INSERT Table1 ON
INSERT Table1 (ID, FirstName, LastName) VALUES
(99, 'Thuy', 'Tien')
SELECT * FROM Table1
```

6.2.4 Check

Kiểu dữ liệu và giá trị default ép thỏa ràng buộc miền giá trị. Các Check giới hạn các giá trị có thể đưa vào cột. Chúng sẽ xác định các giá trị nào là hợp lệ.

- Một cột có thể có nhiều hơn một check constraint, chúng được lượng giá theo thứ tự được tạo.
- Các check constraint giới hạn các giá trị được phép bằng cách định nghĩa:
 - Một vùng hoặc nhiều vùng các giá trị có thể chấp nhận được.
 - Danh sách các giá trị.
 - Một mẫu định trước.

Bạn có thể qui định nhiều constraint check cho một cột đơn, chúng được lượng giá theo thứ tự được tạo.

6.2.4.1 Check Constraint

Check Constraint là một của định nghĩa bảng. Chúng có thể được định nghĩa trong khi tạo bảng, hiệu chỉnh bảng, và có thể xóa bất kỳ lúc nào. Chúng có thể được vô hiệu hoá (disabled) hoặc làm có hiệu lực (enabled) khi cần. Một cột có thể có nhiều hơn một check constraint. Chúng được lượng giá theo thứ tự được tạo. Check constraint:

- ✓ Lượng giá thành một biểu thức logic, như là biểu thức của mệnh đề WHERE.
 - ✓ Có thể tham chiếu đến các cột khác trong cùng một bảng.
- Định nghĩa Check Constraint khi tạo bảng
 - Định nghĩa ở mức cột


```
CREATE TABLE tablename (columnname datatype [ CONSTRAINT constraintname ]
CHECK [NOT FOR REPLICATION] (logical_expression)
```
 - Định nghĩa ở mức bảng


```
CREATE TABLE tablename (columnname datatype [...], [ CONSTRAINT
constraintname ] CHECK [NOT FOR REPLICATION] (logical_expression)
```
 - Định nghĩa CHECK CONSTRAINT với bảng đã tồn tại


```
ALTER TABLE tablename
[ WITH CHECK | WITH NOCHECK ] ADD
[ CONSTRAINT constraintname ]
CHECK [NOT FOR REPLICATION] (logical_expression)
```

Ví dụ:

```
ALTER TABLE Chucvu
ADD CONSTRAINT NV_HSPC_Chk CHECK
        (HSPC >= 0.1 AND HSPC < 0.5)
```

6.2.4.2 RULE

Rule là một tính năng tương thích ngược để định nghĩa các qui tắc hợp lệ mà có thể kết buộc vào các cột của bảng hoặc các kiểu dữ liệu do người dùng định nghĩa. Giống như đối tượng Default, Rule được tạo trên chính nó trước khi được kết buộc vào đối tượng khác. Để tạo một Rule, ta sẽ dùng lệnh CREATE RULE.

Một cột chỉ có thể có một Rule được kết buộc, bạn có thể kết buộc một Rule với cột đã có định nghĩa Check Constraint. Cả hai đều có giá trị nhưng sẽ ưu tiên Check Constraint.

Định nghĩa Rule

```
CREATE RULE rulename AS condition_expression
```

Kết buộc Rule vào một cột

```
sp_bindrule rulename, tablename.columnname
```

Kết buộc Rule vào user-defined datatype

```
sp_bindrule rulename, datatype_name [, futureonly]
```

Futureonly chỉ định rằng các cột tồn tại sẵn mà có dùng kiểu dữ liệu này thì không kế thừa rule mới. Cờ này chỉ sử dụng khi kết Rule với kiểu dữ liệu, đối với cột thì không.

Ví dụ: Tạo một rule kiểm tra ngày và kết nó vào cột OrderDate cho bảng Orders

```
CREATE RULE ActiveDate AS
@Date BETWEEN '01/01/70' AND GETDATE ()
AS
sp_bindrule ActiveDate, 'Orders.OrderDate'
```

Các biểu thức dùng trong Rule giống các nguyên tắc như các điều kiện của check constraint và cũng tương tự biểu thức trong mệnh đề Where, ngoài bạn không thể tham chiếu đến các cột CDSL khác trong các Rule. Nếu bạn so sánh cú pháp của lệnh Check và biểu thức rule, hai sự khác biệt chính là:

- Biểu thức Rule dùng một biến (bắt đầu với một ký hiệu @) mà sẽ được thay thế bởi giá trị cột khi được đánh vào cột.
- Một biểu thức Rule không thể tham chiếu đến các cột của bảng.

Điểm thứ hai là hành vi khác biệt lớn nhất giữa Check constraint và các Rule: Rule chỉ tương đương với Check constraint ở mức cột.

6.2.5 Primary key Constraint

Một bảng thường có một hay nhiều cột với các giá trị riêng để nhận biết mỗi hàng trong bảng. Các cột này được gọi là khóa chính (Primary key) của bảng và bảo đảm tính toàn vẹn thực thể của bảng.

- SQL Server tự động tạo một chỉ mục cho bảng ứng với các cột tham gia primary key constraint.
- Một bảng chỉ có một constraint Primary key.
- Một cột nằm trong constraint Primary key không thể chấp nhận giá trị Null, trùng lặp. Bởi vì Primary key constraint bảo đảm tính duy nhất của dữ liệu nên chúng thường được ấn định cho cột nhận dạng (identity column).
- Nếu một Primary 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 Primary key constraint phải là sự kết hợp duy nhất.

Tạo Primary Key Constraint

Có thể tạo constraint Primary key trong khi tạo table hoặc thêm constraint Primary key cho table có sẵn

Để hiệu chỉnh constraint Primary key của một table thì bạn phải xóa constraint Primary key và tạo lại.

- **Định nghĩa Primary Key Constraint khi tạo bảng:**

- Định nghĩa ở mức cột

```
CREATE TABLE tablename
(columnname datatype [ CONSTRAINT constraintname ]
PRIMARY KEY [ CLUSTERED | NONCLUSTERED ]
[ WITH FILLFACTOR = fillfactor ]
[ ON { filegroup | DEFAULT } ] [...]
```

- Định nghĩa ở mức bảng

```
CREATE TABLE tablename
(columnname datatype [...], [ CONSTRAINT constraintname ]
PRIMARY KEY [ CLUSTERED | NONCLUSTERED ]
{ ( column [ ASC | DESC ] [ ,...n ] ) }
[ WITH FILLFACTOR = fillfactor ]
[ ON { filegroup | DEFAULT } ]
```

Ví dụ 1: Vừa tạo bảng vừa định nghĩa Primary Key Constraint

Cách 1: Định nghĩa ở mức cột

```
CREATE TABLE Table1
( col1 INT NOT NULL PRIMARY KEY,
  col2 VARCHAR(30)
)
-- Kiểm tra constraint
EXEC Sp_helpconstraint Table1
```


Cách 2: Định nghĩa ở mức bảng

```
CREATE TABLE Table2
    (col1 INT NOT NULL,
     col2 VARCHAR(30)
     CONSTRAINT Table2_PK PRIMARY KEY(col1)
    )
EXEC Sp_helpconstraint Table1
```

▪ **Định nghĩa Primary Key Constraint với bảng đã tồn tại:**

- Định nghĩa ở mức cột

```
ALTER TABLE tablename ADD columnname datatype
[CONSTRAINT constraintname ] PRIMARY KEY
[ON {filegroup | DEFAULT}] [...]
```

- Định nghĩa ở mức bảng

```
ALTER TABLE tablename ADD[CONSTRAINT constraintname]
PRIMARY KEY { ( column [ ASC | DESC ] [ ,...n ] ) }
[ ON { filegroup | DEFAULT } ]
```

Ví dụ 2: Thêm một PRIMARY KEY constraint cho một bảng có sẵn

```
CREATE TABLE Table3
    ( col1 INT NOT NULL,
     col2 VARCHAR(30)
    )
ALTER TABLE Table3
ADD CONSTRAINT table3_PK PRIMARY KEY (col1)
EXEC Sp_helpconstraint Table3
```

Lưu ý: Khi một Primary Key Constraint được thêm vào một bảng với cột có sẵn trong bảng thì SQL Server sẽ kiểm tra dữ liệu hiện có có tuân theo các qui tắc của một Primary key hay không: **Không Null, Không trùng lặp**. Nếu không thỏa qui tắc thì sẽ không tạo được Primary Key Constraint.

Xóa một Primary Key Constraint

```
ALTER TABLE Table2
DROP CONSTRAINT table2_PK
```

Lưu ý: Không thể xóa một Primary Key constraint nếu nó được tham chiếu bởi một Foreign key Constraint của một bảng khác, muốn xóa thì phải xóa Foreign key Constraint trước.

6.2.6 Unique Constraints

Unique Constraints dùng để đảm bảo không có giá trị trùng ở các cột. Mặc dù cả Unique constraint và Primary key constraint đều tuân theo tính duy nhất, nhưng hãy sử dụng Unique constraint khi bạn 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.

Định nghĩa Unique Constraint:

- **Định nghĩa UNIQUE CONSTRAINT khi tạo bảng**

- Định nghĩa ở mức cột

```
CREATE TABLE tablename (columnname datatype
[ CONSTRAINT constraintname] UNIQUE [ CLUSTERED | NONCLUSTERED ]
[ ON { filegroup | DEFAULT } ] [...])
```

- Định nghĩa ở mức bảng

```
CREATE TABLE tablename
(columnname datatype [...], [ CONSTRAINT constraintname]
UNIQUE [ CLUSTERED | NONCLUSTERED ]
{ ( column [ ASC | DESC ] [ ,...n ] ) }
[ ON { filegroup | DEFAULT } ] )
```

- **Định nghĩa UNIQUE CONSTRAINT với bảng đã tồn tại**

- Định nghĩa ở mức cột

```
ALTER TABLE tablename
ADD columnname datatype [ CONSTRAINT constraintname]
UNIQUE [ CLUSTERED | NONCLUSTERED ]
[ ON { filegroup | DEFAULT } ] [...]
```

- Định nghĩa ở mức bảng

```
ALTER TABLE tablename
ADD [ CONSTRAINT constraintname]
UNIQUE [ CLUSTERED | NONCLUSTERED ]
{ ( column [ ASC | DESC ] [ ,...n ] ) }
[ ON { filegroup | DEFAULT } ]
```

Ví dụ:

```
ALTER TABLE Nhanvien
ADD Scmnd CHAR(15) CONSTRAINT NV_CMND_UNIQUE
UNIQUE
```

6.2.7 Foreign Key Constraint

Một khóa ngoại (Foreign key) là một cột hay sự kết hợp của nhiều cột được thiết lập và tuân theo một liên kết giữa các dữ liệu trong hai bảng. Một liên kết được tạo ra giữa hai bảng bằng cách bổ sung một hay nhiều cột có chứa giá trị khóa primay key của một bảng vào một bảng khác. Các cột này trở thành khóa ngoại của bảng thứ hai.

Ta có thể quy định khóa ngoại bằng các ấn định một constraint Foreign key khi bạn tạo hay thay đổi một bảng.

Một constraint Foreign key không bắt buộc phải liên kết với chỉ một constraint Foreign key trong một bảng khác, nó cũng có thể được ấn định để tham chiếu các cột của một constraint Foreign Unique trong một bảng khác.

Mặc dù mục đích chính của một constraint Foreign key là điều khiển dữ liệu có thể được chứa trong bảng khóa ngoại, nhưng nó cũng điều khiển các thay đổi đối với bảng khóa chính. Chẳng hạn, nếu một mẫu tin phòng ban bị xóa ra khỏi bảng Phongban, và mã phòng ban được sử dụng cho các nhân viên trong bảng NhanVien, tính toàn vẹn trong mối quan hệ giữa hai bảng này sẽ bị phá vỡ. Các dòng nhân viên có mã phòng ban bị xóa sẽ nằm mồ côi trong bảng NhanVien mà không có liên kết với dữ liệu trong bảng PhongBan.

Một constraint Foreign key sẽ ngăn chặn tình trạng mồ côi dữ liệu. Constraint Foreign key sẽ bảo đảm không cho phép bạn xóa dữ liệu trong bản chính nếu các dữ liệu này có sự liên kết với dữ liệu trong bảng khóa ngoại.

▪ Định nghĩa FOREIGN KEY CONSTRAINT khi tạo bảng

- Định nghĩa ở mức cột

```
CREATE TABLE tablename
(columnname datatype [ CONSTRAINT constraintname ]
[ FOREIGN KEY ] REFERENCES ref_table [ ( ref_column ) ]
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
[ NOT FOR REPLICATION ] [...]
```

- Định nghĩa ở mức bảng

```
CREATE TABLE tablename
(columnname datatype [...],
[ CONSTRAINT constraintname ]
FOREIGN KEY [ ( column [ ,...n ] ) ]
REFERENCES ref_table [ ( ref_column [ ,...n ] ) ]
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
[ NOT FOR REPLICATION ]
```

Ví dụ:

```

CREATE TABLE PhongBan
    ( MaPb INT,
      TenPb VARCHAR(30)
      CONSTRAINT Pb_Pk (MaPb) PRIMARY KEY (MaPb)
    )

GO

CREATE TABLE NhanVien
    ( MaNv INT,
      TenNV VARCHAR(30),
      MaPB int,
      MaCv int
      CONSTRAINT Nv_Pk PRIMARY KEY (MaNv)
      CONSTRAINT Nv_Fk FOREIGN KEY (MaPb) REFERENCES
      PhongBan (MaPb)
    )

GO

-- Xem các constraint
SP_HELPCONSTRAINT NHANVIEN
GO

```

- **Định nghĩa Foreign Key Constraint với bảng đã tồn tại:**

```

ALTER TABLE tablename
[ WITH CHECK | WITH NOCHECK ] ADD
[ CONSTRAINT constraintname ]
FOREIGN KEY [( column [ ,...n ] )]
REFERENCES ref_table [( ref_column [ ,...n ] )]
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
[ NOT FOR REPLICATION ]

```

Trong đó:

WITH CHECK: trước khi tạo ràng buộc, SQL Server sẽ kiểm tra dữ liệu hiện có có vi phạm ràng buộc hay không, nếu có sẽ không tạo constraint.

WITH NOCHECK: Tạo Constraint mà không cần kiểm tra dữ liệu hiện có có vi phạm ràng buộc hay không

Ví dụ:

```

CREATE TABLE ChucVu
    ( MaCv INT PRIMARY KEY,
      TenCv VARCHAR(30),
      HSPC Real
    )

```

```
        )  
GO  
ALTER TABLE Nhanvien  
ADD CONSTRAINT Nv_Cv_Pk FOREIGN KEY (Macv) REFERENCES  
ChucVu (MaCv)
```

Lưu ý:

```
ON DELETE CASCADE| NO ACTION  
ON UPDATE CASCADE| NO ACTION
```

BÀI 7:

TRUY XUẤT CƠ SỞ DỮ LIỆU CỦA SQL SERVER

Mục đích chính của CSDL trong SQL Server là lưu trữ dữ liệu sao cho dữ liệu dễ dàng được khai thác bởi người sử dụng. Bạn cũng có thể truy cập dữ liệu thông qua một ứng dụng hoặc các trình tiện ích để gửi yêu cầu nhập dữ liệu hoặc hiệu chỉnh dữ liệu đến SQL Server. Nhằm mục đích tìm hiểu ta dùng SQL Query Analyzer như là một công cụ chính để truy xuất và hiệu chỉnh dữ liệu trong một CSDL của SQL Server.

7.1 Câu lệnh SELECT

Câu lệnh Select được sử dụng một cách thường xuyên và là cách cơ bản để truy vấn dữ liệu.

```
SELECT [ALL | DISTINCT] [TOP n [WITH TIES]] select_list
[ INTO new_table ]
FROM table_source
[ WHERE search_condition ]
[ GROUP BY group_by_expression ]
[ HAVING search_condition ]
[ ORDER BY order_expression [ ASC | DESC ] ]
```

Products(ProductID, ProductName, SupplierID, CategoryID, UnitPrice, ...)

Customers(CustomerID, CompanyName, Address, City, Region, Country, ...)

Employees(EmployeeID, LastName, FirstName, BirthDate, City, ...)

Orders(OderID, CustomerID, EmployeeID, OrderDate, ...)

Order Details(OrderID, ProductID, UnitPrice, Quantity, Discount)

Mệnh đề SELECT

```
USE NorthWind
SELECT * FROM Products

SELECT ProductID, ProductName, UnitPrice
FROM Products
WHERE UnitPrice > 40
ORDER BY UnitPrice DESC
```

Từ khóa **DISTINCT**: Loại bỏ các mẫu tin trùng trong tập kết quả.

```
SELECT DISTINCT City, Region
FROM Customers
ORDER BY Region
```

Từ khóa **TOP n** : Lấy ra n mẫu tin đầu tiên

-- Chỉ lấy đúng 5 mẫu tin đầu tiên có số lượng bán cao nhất.

```
SELECT TOP 5 orderid, productid, quantity
FROM [order details]
ORDER BY quantity DESC
```

--- WITH TIES có nghĩa là các mẫu tin ngang bằng giá trị trong cột ORDER BY được liệt kê.

```
SELECT TOP 5 WITH TIES orderid, productid, quantity
FROM [order details]
ORDER BY quantity DESC
Từ khóa AS: Đặt bí danh cho bảng, cột
```

```
SELECT p.Productid, p.Productname
FROM Products As p
```

Mệnh đề SELECT INTO: Sinh thêm một bảng mới mà dữ liệu được lấy từ các bảng khác.

```
SELECT FirstName, LastName
INTO EmployeeNames
FROM Northwind.dbo.Employees
```

Dùng với các hàm SUM, MAX, MIN, AVG, COUNT

```
--AVG
USE Northwind
SELECT AVG(UnitPrice)
FROM dbo.Products

-- SUM
SELECT SUM(Quantity)
FROM dbo.[Order Details]

--- Phân biệt Count và Count(*); Count(*) đếm tất cả
các mẫu tin có trong bảng
SELECT COUNT(*)
FROM dbo.Employees

--- Count(Reportsto) đếm tất cả các giá trị có trong
cột ReportTo, nếu giá trị của Reportsto là NULL thì SQL
Server sẽ bỏ qua không đếm
USE Northwind
SELECT COUNT(ReportsTo)
FROM dbo.Employees
```

Mệnh đề WHERE, GROUP BY, và HAVING: Where và Having dùng để lọc dữ liệu trong câu Select. **Select với Where. Lọc dữ liệu thỏa điều kiện. Các phép toán có thể dùng trong where: >, >=, <, <=, =, AND, OR, BETWEEN ... AND, !=, IS NULL, NOT IS NULL**

```
Cho biết danh sách các mặt hàng đã bán trong hóa đơn số 2
SELECT OrderID, Quantity
FROM Orderhist
WHERE OrderID=2
```

Select với Group by: Kết nhóm

Cho biết tổng lượng hàng đã bán ứng với mỗi ProductId

```
SELECT productID, SUM(quantity) AS Total_quantity
FROM [Order details]
GROUP BY productID
```

Cho biết tổng lượng hàng đã bán ứng với ProductId=2

```
SELECT ProductID, SUM(Quantity) AS Total_quantity
FROM [Order Details]
WHERE ProductID=2
GROUP BY productID
```

Cho biết danh sách các mã hàng có tổng số lượng >=30

```
SELECT ProductID, SUM(Quantity) AS Total_quantity
FROM [Order details]
GROUP BY ProductID
HAVING SUM(quantity)>=30
```

Lưu ý: *Having để giới hạn kết quả do Group By sản sinh ở các hàm Where giới hạn trước khi Group By*

Cho biết tổng lượng hàng đã bán ứng với mỗi mã nhóm hàng

```
SELECT Categoryid, SUM(Quantity) AS Total_quantity
FROM [order details] JOIN products ON [order
details].productId=products.productId
GROUP BY Categoryid
```

Tổng số lượng bán ứng với mỗi hóa đơn

```
SELECT orderid, SUM(quantity) AS Total_quantity
FROM [order details]
GROUP BY orderid
HAVING SUM(quantity)>=250
```

Sử dụng Group By với toán tử ROLLUP.

ROLL UP Sẽ chèn thêm các dòng Total nằm trước các nhóm mẫu tin được phân theo GROUP BY.

Ví dụ:

```
/* Tổng số lượng đã được đặt hàng cho mỗi Product ứng với mỗi Order cho các Order
có OrderID<1025
```



```

*/
use northwind
SELECT      productID,orderid,      SUM(quantity)      AS
Total_quantity
FROM [Order details]
WHERE orderid<10250
GROUP BY OrderId,productId
ORDER BY OrderId, productID

```

Kết quả

Không có With Rollup		Có With Rollup	
ProductID	Orderid	ProductID	Orderid
Total_quantity		Total_quantity	
11	10248 12	NULL	NULL 76
42	10248 10	NULL	10248 27
72	10248 5	11	10248 12
14	10249 9	42	10248 10
51	10249 40	72	10248 5
		NULL	10249 49
		14	10249 9
		51	10249 40

Ví dụ khác: Danh sách Order ứng với tổng quantity của product có mã ProductID là 8 hoặc 9.

```

SELECT ProductID,Orderid, SUM(quantity) AS
Total_quantity
FROM [Order Details]
WHERE ProductID IN (8,9)
GROUP BY ProductID, OrderId
WITH ROLLUP
ORDER BY ProductID, OrderId

```

```

/* total quantity for: each product for each order. all
products for each order
*/

```

```

SELECT ProductID, Orderid, SUM(quantity) AS
Total_quantity
FROM [Order details]
WHERE orderid<10250
GROUP BY OrderId,productId
WITH ROLLUP
ORDER BY OrderId, productID

```

Sử dụng mệnh đề Group By với toán tử CUBE: Tương tự như Rollup nhưng thêm các dòng Total của mỗi sự kết hợp có thể có giữa các cột

Ví dụ:

```
SELECT ProductID, Orderid, SUM(quantity) AS
Total_quantity
FROM [Order details]
WHERE orderid<10250
GROUP BY OrderId,productId
WITH CUBE
ORDER BY OrderId, productId
```

Kết quả

```
NULL NULL 76
11 NULL 12
14 NULL 9
42 NULL 10
51 NULL 40
72 NULL 5
NULL 10248 27
11 10248 12
42 10248 10
72 10248 5
NULL 10249 49
14 10249 9
51 10249 40
```

Ví dụ khác:

```
SELECT ProductID, Orderid, SUM(quantity) AS
Total_quantity
FROM [Order Details]
GROUP BY OrderId, ProductId
WITH CUBE
ORDER BY ProductID, OrderId
```

Dùng hàm GROUPING: Hàm Grouping sẽ trả về một giá trị là 0 hoặc 1 để xác định dòng kết quả đó là dòng tổng (1) hay là dòng chi tiết (0)

Ví dụ:

```
SELECT ProductID, grouping(productID)as a, Orderid,
grouping(orderid) as b,SUM(quantity) AS Total_quantity
FROM [Order details]
WHERE orderid<10250
```

```
GROUP BY OrderId,productId
WITH CUBE
ORDER BY OrderId, productId
```

ProductID	a	Orderid	b	Total_quantity
NULL 1	NULL 1	76		
11 0	NULL 1	12		
14 0	NULL 1	9		
42 0	NULL 1	10		
51 0	NULL 1	40		
72 0	NULL 1	5		
NULL 1	10248 0	27		
11 0	10248 0	12		
42 0	10248 0	10		
72 0	10248 0	5		
NULL 1	10249 0	49		
14 0	10249 0	9		
51 0	10249 0	40		

Dùng toán tử COMPUTE và COMPUTE BY: Thông thường dùng để kiểm tra số liệu, dùng kèm với các hàm thống kê SUM, AVG, MAX, MIN.... COMPUTE ... BY ... :có kết nhóm

```
SELECT productID,orderid, quantity
FROM [Order Details]
ORDER BY productID,OrderId
COMPUTE SUM(quantity)
```

```
SELECT productID,orderid, quantity
FROM [Order Details]
ORDER BY productID,OrderId
COMPUTE SUM(quantity) BY productid
COMPUTE SUM(quantity)
```

7.2 Sử dụng JOINS để truy xuất dữ liệu

Bằng JOIN, chúng ta có thể lấy dữ liệu từ hai hoặc nhiều bảng dựa trên mối quan hệ giữa các bảng. Tuy nhiên nếu ta không thích dùng Join để lấy dữ liệu thì bạn cũng có thể viết các câu truy vấn bằng dạng truy vấn con (Subqueries). Dùng Joins thì tốc độ thực hiện của câu truy vấn nhanh hơn SubQueries nhưng lại khó hiểu hơn. Trong một truy vấn tham chiếu đến nhiều table, thì tất cả các cột phải được chỉ rõ một các tường minh là cột đó lấy từ table nào.

Thông thường thì điều kiện kết nối là dùng phép so sánh bằng, nhưng trong SQL Server ta có thể định nghĩa mối quan hệ giữa trên các toán tử khác như <>, >, >=, <, <= ...

Các cột được dùng để kết nối không nhất thiết phải cùng tên hay cùng kiểu dữ liệu. Nếu kiểu dữ liệu khác nhau thì có thể dùng các hàm để chuyển đổi kiểu dữ liệu.

Gồm hai loại Joins chính: INNER JOINS và OUTER JOINS. Tuy nhiên ta còn có thể tạo một dạng Join khác như CROSS-JOINS và SELF-JOINS.

Inner Joins: Liên kết Inner Joins chỉ trả về những mẫu tin đều hiển hữu ở cả hai bảng quan hệ và phải thỏa mãn điều kiện kết.

Ví dụ:

```
SELECT Em.EmployeeID, LastName+ ' '+FirstName AS
EmployeeName,
COUNT(OrderID) AS TotalOrders
FROM Employees AS Em INNER JOIN Orders AS O ON
Em.EmployeeID=O.EmployeeID
WHERE MONTH(OrderDate)=8 AND YEAR(Orderdate)=1996
GROUP BY Em.EmployeeID, LastName+ ' '+FirstName
ORDER BY Em.EmployeeID
```

Điều kiện kết nằm ở mệnh đề Where

```
SELECT P.ProductID, S.SupplierID, S.CompanyName
FROM Suppliers AS S, Products AS P
WHERE S.SupplierID = P.SupplierID
AND P.UnitPrice > $10
AND S.CompanyName LIKE N'F%'
```

Điều kiện kết nằm ở mệnh đề From

```
SELECT P.ProductID, S.SupplierID, S.CompanyName
FROM Suppliers AS S JOIN Products AS P
ON (S.SupplierID = P.SupplierID)
WHERE P.UnitPrice > $10
AND S.CompanyName LIKE N'F%'
```

Outer Joins: trả về tất cả những mẫu tin nằm ít nhất một bảng nào đó trong các bảng tham gia kết nối và cũng phải thỏa điều kiện kết. Outer Joins cung cấp 3 kiểu outer Join: Left, Right, và Full. Trả về tất cả các dòng từ bảng bên trái mà được tham chiếu từ Left Outer Joins. Trả về tất cả các dòng từ bảng bên phải mà được tham chiếu từ Right Outer Joins. Trả về tất cả các dòng từ cả 2 bảng mà được tham chiếu từ Full Outer Joins.

Ví dụ : Dùng câu truy vấn sau và lần lượt thay đổi kiểu Join (Inner, Left, Right, Full), cho thi hành và cho nhận xét kết quả của câu truy vấn. Tự rút ra kết luận

```
SELECT O.OrderID, O.CustomerID, c.ContactName, C.City
```

```
FROM Orders O LEFT JOIN Customers C
      ON O.CustomerID = C.CustomerID AND
      O.ShipCity=C.City
ORDER BY O.OrderID
```

Cross Joins: Mỗi dòng trong tất cả các dòng của bảng bên trái sẽ kết hợp với tất cả các dòng của bảng bên phải. Giả sử X, Y là số dòng của bảng bên trái và bên phải, thì tập kết quả sau khi Cross Join có X*Y dòng.

Self Joins: Tự liên kết

7.3 Dùng Sub-Queries

Subqueries là một câu lệnh Select mà nó trả về một giá trị đơn hoặc một tập các giá trị và nó được nằm trong các câu lệnh như Select, Insert, Update, or Delete. Một SubQuery có thể được dùng bất kỳ nơi nào trong biểu thức được cho phép. Một SubQuery cũng được gọi trong một truy vấn khác, hoặc một câu lệnh select có subquery cũng có thể được gọi từ một câu Select khác (32 cấp).

- ✓ Ngắt câu lệnh phức tạp thành những đoạn truy vấn đơn giản.
- ✓ Trả lời một truy vấn từ một truy vấn khác.
- ✓ Các Sub Query hầu như có thể viết bằng Join và SQL Server luôn luôn thi hành những câu lệnh Join nhanh hơn SubQueries.
- ✓ Không thể dùng SubQueries trên cột hình ảnh.

```
/* SELECT statement built using a subquery. */
```

Lọc ra những Product Name có UnitPrice bằng với Unitprice của ProductName là 'Sir Rodney' 's Scones'

```
SELECT ProductName
FROM Products
WHERE UnitPrice =
      (SELECT UnitPrice
       FROM Products
       WHERE ProductName = 'Sir Rodney' 's Scones')
```

```
/* SELECT statement built using a join that returns the same
result set. */
```

```
SELECT ProductName
FROM Products AS Prd1
      JOIN Products AS Prd2
      ON (Prd1.UnitPrice = Prd2.UnitPrice)
WHERE Prd2.ProductName = 'Sir Rodney' 's Scones'
```

Dùng từ khóa IN trong subquery.

Lọc ra những OrderId, EmployeeId do những Employee ở City Seattle Order.

```
SELECT OrderId, EmployeeID AS EmpID
FROM Orders
```

```

WHERE EmployeeID IN (
    SELECT EmployeeId
    FROM Employees
    WHERE City='Seattle'
)
ORDER BY OrderID

```

Dùng các toán tử so sánh trong Subquery: Các toán tử bao gồm =, >, >=, <, <=, <>, >=ALL, > ANY,

Lọc ra những OrderId, ProductId, UnitPrice mà có Unitprice lớn hơn như Unitprice được bán bởi EmployeeID=5

```

SELECT OrderId, [Order Details].ProductId, [Order
Details].UnitPrice
FROM [Order Details]
WHERE Unitprice > ALL ( SELECT [Order
Details].UnitPrice
                        FROM [Order Details] JOIN Orders
                        ON [Order Details].OrderId=
Orders.OrderId
                        where Orders.EmployeeID=5
                        )
ORDER BY [Order Details].UnitPrice, OrderID)
ORDER BY UnitPrice.OrderID

```

Dùng từ khóa EXISTS, NO EXISTS trong subquery:

Lọc ra những OrderId ứng với các CustomerId mà những Customer này ở London City.

```

SELECT OrderId, CustomerId
FROM Orders
WHERE EXISTS ( SELECT * FROM Customers
              WHERE
Customers.CustomerId=Orders.CustomerId
              AND City='London'
              )
ORDER BY OrderID

```

7.4 Hiệu chỉnh dữ liệu trong cơ sở dữ liệu của SQL SERVER

7.4.1 Chèn (INSERT) dữ liệu vào CSDL.

Câu lệnh Insert được dùng để thêm một hoặc nhiều dòng dữ liệu vào bảng hoặc một View. Một số lưu ý:

- ✓ Cố gắng chèn dữ liệu đúng kiểu dữ liệu của cột, tránh đưa giá trị Null vào cột không chấp nhận giá trị Null.
- ✓ Tuân thủ đúng các toàn vẹn dữ liệu, bẫy lỗi.
- ✓ Chỉ định một danh sách các giá trị (values) ứng với danh sách các cột, nếu không chỉ định thì insert sẽ theo thứ tự cấu trúc của table.
- ✓ Cột có thuộc tính Identity thì không cần chỉ định giá trị. Nếu cột có định nghĩa Default và chúng ta chấp nhận giá trị default thì ta có thể bỏ qua việc chỉ định giá trị của cột này.
- ✓ Khi chèn dữ liệu thì tại một thời điểm cũng như một câu lệnh chỉ có thể chèn vào một bảng duy nhất.

7.4.1.1 Chèn mẫu tin từ danh sách các giá trị được chỉ định (Insert ... Values)

```
INSERT <TableName> [(field1, field2, ...)] VALUES (Value1, Value2,...)
```

```
CREATE TABLE NewProducts
(
ProductID INT NOT NULL,
    ProductName NVARCHAR(40),
    CategoryID INT NULL,
    UnitPrice MONEY NULL,
    SupplierID INT NULL
)
GO
INSERT NewProducts (ProductID, ProductName)
VALUES (123, 'Ice Tea')
```

7.4.1.2 Chèn dữ liệu bằng các giá trị từ các bảng khác (Insert ... Select)

```
INSERT <TableName> [(field1, field2, ...)]
SELECT (Value1, Value2,...)
```

```
INSERT NewProducts (ProductID, ProductName)
SELECT ProductID, ProductName
FROM Products
WHERE CategoryID=2
INSERT NewProducts (Od.ProductID, P.ProductName)
SELECT OD.ProductID, ProductName
FROM Products as P INNER JOIN [Order Details]
AS Od
ON P. ProductID = Od. ProductID
WHERE CategoryID<>2
```

7.4.1.3 Chèn mẫu tin từ một stored Procedure (trình bày sau)

7.4.1.4 Xây dựng một Table bằng SELECT INTO

```
SELECT DISTINCT C.CustomerID, C.CompanyName
INTO NewCustomers
FROM Customers AS C INNER JOIN Orders
ON C.CustomerID= orders.CustomerID
AND orders.EmployeeID=2
```

7.4.2 Cập nhật (UPDATE) dữ liệu vào CSDL.

SQL Server cung cấp 3 phương pháp để thay đổi/cập nhật dữ liệu trong một bảng có sẵn. Đó là câu lệnh Update, giao diện lập trình ứng dụng (Application Programming Interfaces - APIs) và dùng con trỏ (Cursors). Trong phần bài học này chỉ trình bày câu lệnh Update.

```
UPDATE <table_name>
SET <column_name> = <expression>
[ FROM <table_list>]
[ WHERE <search_condition>]
WHERE CURRENT OF <Variable_Cursor>
```

Ví dụ:

```
USE Northwind
UPDATE dbo.Products
SET UnitPrice=UnitPrice*1.1
GO
UPDATE [Order Details]
SET Discount=Discount+0.05
WHERE Discount<>0 AND ProductId=2
GO
UPDATE [Order Details]
SET UnitPrice=
(SELECT UnitPrice+ UnitPrice*0.2
 FROM Products
 WHERE = ProductId=2
 )
WHERE ProductId=2
```

7.4.3 Xóa dữ liệu trong cơ sở dữ liệu.

7.4.3.1 Dùng câu lệnh DELETE.

Xóa một hoặc nhiều dòng trong một bảng hay một truy vấn.

```
DELETE table_or_view FROM table_sources WHERE search_condition
```

```
DELETE Orders
```



```
FROM Orders
WHERE EmployeeID IN ( SELECT EmployeeId
                      FROM Employees
                      WHERE City='Seattle'
                    )
```

7.4.3.2 Dùng APIs và cursors để xóa dữ liệu (trình bày sau).

7.4.3.3 Dùng câu lệnh TRUNCATE TABLE.

- Để xóa toàn bộ dữ liệu trong một bảng.
- Về phần chức năng, hoàn toàn giống như câu lệnh Delete.
- Nhanh hơn câu lệnh Delete
- Không bật các bẫy lỗi (trigger)

Cú pháp:

```
TRUNCATE TABLE Tên Table
```

Ví dụ:

```
TRUNCATE TABLE NewProducts
```

BÀI 8: KHUNG NHÌN - VIEW

8.1 Giới thiệu về View.

View là một bảng ảo mà nội dung được định nghĩa bởi một truy vấn (câu Select). Giống như một bảng thực, một view bao gồm một tập các cột và các dòng dữ liệu. Tuy nhiên, một view không là nơi lưu trữ dữ liệu. Các dòng và cột của dữ liệu được tham chiếu từ các bảng trong một truy vấn mà định nghĩa View và là kết quả động khi View được tham chiếu.

Dùng view để:

- Chỉ cho User xem những gì cần cho xem.
- Đơn giản hóa việc truy cập dữ liệu.
- Dùng để lựa chọn những dữ liệu cần thiết ứng với mỗi user.
- Dùng View để Import, Export.
- Kết hợp các dữ liệu khác nhau.

Hạn chế khi định nghĩa View:

- Không bao gồm các mệnh đề COMPUTER hoặc COMPUTER BY.
- Không bao gồm từ khóa INTO.
- Chỉ được dùng ORDER BY chỉ khi từ khóa TOP được dùng.
- Không thể tham chiếu quá 1024 cột.
- Không thể kết hợp với câu lệnh T-SQL khác trong một cùng một bó lệnh.
- Không thể định nghĩa chỉ mục full text trên View.

Partitioned Views: Một partition View kết nối theo chiều dọc các dữ liệu phân tán từ một tập các bảng ở một hay nhiều server, các dữ liệu sẽ hiện lên như thể là chúng được lấy từ một bảng. Có hai loại: Local partition view là view có tham chiếu các table và các view khác nằm trong cùng một server. Distributed partition view có ít nhất một bảng nằm ở server khác.

8.2 Tạo, hiệu chỉnh, xóa View

Tạo View

```
CREATE VIEW [ < database_name > . ] [ < owner > . ] view_name [ (column[,...n]) ]
[ WITH < view_attribute > [,...n] ]
AS
select_statement
[ WITH CHECK OPTION ]
< view_attribute > ::=
{ ENCRYPTION | SCHEMABINDING }
```

Giải thích

view_name: là tên của View. Tên View phải tuân thủ các qui tắc định danh.

Column: Tên được dùng cho cột trong view. Tên cột chỉ dùng trong trường hợp cột được phát sinh từ một biểu thức, hàm, một hằng, các cột trong các table trùng tên. Tuy nhiên tên cột cũng có thể được ấn định trong câu lệnh Select. Không chỉ định tên cột chính là tên các cột trong câu lệnh select

select_statement: Là câu lệnh select để định nghĩa View. Nó có thể tham chiếu một hoặc nhiều bảng hoặc các View khác với một câu Select phức tạp.

Lưu ý: Để tạo một view, bạn phải có quyền dành riêng trên các bảng hoặc view tham chiếu trong định nghĩa view.

WITH CHECK OPTION: Bắt buộc tất cả các câu lệnh hiệu chỉnh dữ liệu thực thi dựa vào View phải tuyệt đối tôn trọng triệt để đến tập tiêu chuẩn trong câu lệnh Select. Nếu bạn dùng từ khóa này, các dòng không thể được hiệu chỉnh trong cách mà tại sao chúng hiện trong view. Bất kỳ hiệu chỉnh nào mà sẽ gây ra tình trạng thay đổi đều bị hủy bỏ, và một lỗi được hiện ra.

WITH ENCRYPTION: Mã hóa câu lệnh Select tạo ra view

SCHEMABINDING: Kết view với một giản đồ. Khi SCHEMABINDING được chỉ định, câu lệnh Select phải chỉ rõ chủ quyền của các bảng, các view. Các hàm được tham chiếu View hay bảng tham gia trong view được tạo với schema không thể xóa trừ phi view đó bị xóa hoặc thay đổi cơ chế này. Câu lệnh Alter Table trên bảng tham gia trong view cũng bị lỗi.

Ví dụ 1: Xem danh sách các InvoiceNo và CustNo

```
CREATE VIEW vwSim
AS
SELECT InvoiceNo, CustNo
From tblSim
Where InvoiceDate = Getdate()
ORDER BY InvoiceNo
```

Xem nội dung View

```
Select * From vwSim
```

Ví dụ 2:

```
CREATE VIEW vwSales
AS
Select c.CustNo, c.CustName, sim.InvoiceNo,
sim.InvoiceDate, sid.ItemNo, sid.Quatity
From TblCustomer c INNER JOIN TblSim sim
ON c.CustNo=sim.CustNo INNER JOIN TblSid sid
ON sim.InvoiceDate = sid.InvoiceDate
```

Xem nội dung View

```
Select * From vwSales
```

8.3 Tạo Partition view

- Các bảng tham gia trong Partition view phải có cấu trúc giống nhau.
- Có một cột có check constraint, với phạm vi của Check constraint ở mỗi bảng là khác nhau.
- Tạo view bằng cách kết các dữ liệu bằng từ khóa UNION ALL

Ví dụ: Ta có 3 table tương ứng dùng để lưu trữ các khách hàng ở 3 miền Bắc, Trung, Nam có cấu trúc tạo như sau:

```
Create Table KH_BAC
```

```

        (Makh int primary key,
        TenKh Nchar(30),
        Khuvuc Nvarchar(30) CHECK (Khuvuc='Bac bo')
        )
Create Table KH_TRUNG
        (Makh int primary key,
        TenKh Nchar(30),
        Khuvuc Nvarchar(30) CHECK (Khuvuc='Trung bo')
        )
Create Table KH_NAM
        (Makh int primary key,
        TenKh Nchar(30),
        Khuvuc Nvarchar(30) CHECK (Khuvuc='Nam bo')
        )

```

Tạo một partition View gộp 3 bảng trên lại với nhau:

```

Create View Khachhang
AS
        Select * From KH_BAC
        UNION ALL
        Select * From KH_TRUNG
        UNION ALL
        Select * From KH_NAM

```

8.4 Truy xuất dữ liệu thông qua View.

(Tương tự như truy xuất dữ liệu trên bảng)

8.4.1 Xem dữ liệu thông qua view.

Dùng câu lệnh Select (Ví dụ ở trên)

8.4.2 Hiệu chỉnh dữ liệu thông qua View.

Thao tác hiệu chỉnh dữ liệu giống thao tác hiệu chỉnh dữ liệu trên một bảng. Tuy nhiên, view phải thỏa mãn điều kiện sau:

- View chỉ tham chiếu duy nhất một bảng.
- Thao tác Delete không bao giờ được phép thực hiện trên nhiều bảng trong View.
- Các hàm kết hợp Group By, Union, Distinct, Top không dùng trong danh sách chọn trong câu Select của View.
- Không có các cột tính toán.

Ví dụ:

```

INSERT INTO Khachhang VALUES (11, 'dddd', 'Nam bo')

```

Hiệu chỉnh dữ liệu thông qua partitioned View

Khi dùng lệnh Insert và update phải tôn trọng các qui tắc sau:

- Tất cả các cột phải có giá trị ngay cả cột chấp nhận null và cột có giá trị default.
- Từ khóa Default không được sử dụng trong câu Insert, update.
- Phải có giá trị đúng của cột có check constraint.
- Câu lệnh insert không cho phép nếu bảng thành viên có cột có thuộc tính identity, cột timestamp.
- Không insert hoặc Update nếu có một kết self-join trong cùng view hay bảng thành viên.

Khi dùng lệnh delete, ta có thể xóa các mẫu tin trong bảng thành viên thông qua view. Lệnh Delete không thực thi nếu có liên kết Self-join

BÀI 9: CHUYÊN ĐỔI DỮ LIỆU

9.1 Khái niệm chuyên đổi và biến đổi dữ liệu.

Sau khi bạn tạo CSDL của bạn, bạn cần nhập các mẫu dữ liệu. Thông thường, bạn thường đưa dữ liệu vào (importing data) hoặc chuyển dữ liệu (transferring) có sẵn từ một hoặc nhiều nguồn dữ liệu khác đến hoặc đi từ SQL Server 2000. Trong bài này chúng ta nghiên cứu import dữ liệu từ nguồn dữ liệu khác, đồng thời cũng giới thiệu các công cụ (Tool) chính sử dụng để importing data và biến đổi dữ liệu (transforming data). Các công cụ đó là DTS, Bcp, và lệnh BULK INSERT.

9.1.1 Import/Export dữ liệu.

Import dữ liệu là quá trình đưa dữ liệu có sẵn từ nguồn dữ liệu khác hoặc chính SQL Server vào trong SQL Server. Export là quá trình ngược lại với import, đưa dữ liệu của SQL Server ra ngoài nguồn dữ liệu bên ngoài. Nguồn dữ liệu đó có thể là một CSDL hàng thứ ba, bảng tính, tập tin văn bản (Text). Tuy nhiên trước khi bạn import/export dữ liệu này vào, bạn phải thực hiện các tác vụ chuẩn bị để ước lượng dữ liệu bên ngoài và quyết định các bước sẽ phải thực hiện trong tiến trình import/export. Các bước chuẩn bị này cũng sẽ giúp bạn chọn công cụ thích hợp để dùng

- Quyết định tính nhất quán (consistency) của dữ liệu hiện đã có trong nguồn dữ liệu bên ngoài/bên trong.
- Quyết định những cột được đưa vào/đưa ra.
- Quyết định dạng dữ liệu (Format) của dữ liệu có sẵn nên hiệu chỉnh để nó nhất quán trong CSDL đích đến (Ví dụ: Cần đổi dạng ngày hoặc chuyển giá trị số sang giá trị chuỗi như 1, 2, 3 chuyển thành nghèo, trung bình, khá).
- Quyết định cột dữ liệu có sẵn nên hiệu chỉnh.
- Quyết định import/export dữ liệu sẽ là một tác vụ thực hiện một lần hay một tác vụ thực hiện định kỳ.
- Quyết định các truy xuất dữ liệu có sẵn là truy xuất trực tiếp hay gián tiếp.
-

9.1.2 Biến đổi dữ liệu (Data Transformations)

Sau khi bạn ước lượng dữ liệu trong nguồn dữ liệu bên ngoài/bên trong, bạn cần quyết định cách tiến hành. Đôi khi, những thay đổi dữ liệu có thể thực hiện ngay trong nguồn dữ liệu bên ngoài nhưng thông thường những thay đổi này không có thể thực hiện trong nguồn dữ liệu bên ngoài mà không hoặc dùng ứng dụng hiện có (ví dụ: thêm cột hoặc thay đổi định dạng cột) hoặc tiêu tốn quá nhiều thời gian (ví dụ: Cố gắng thúc ép nhất quán dữ liệu tại nơi sự nhất quán chưa có. Những thay đổi này có thể hoặc là sau khi dữ liệu được import vào SQL Server, sử dụng các bảng tạm và sử dụng câu lệnh Transact-SQL để lọc và tinh chế dữ liệu, hoặc có thể được thực hiện trong tiến trình import vào chính bảng. Những thay đổi đến dữ liệu tạo trong tiến trình import và export được nói đến như các biến đổi dữ liệu. Một biến đổi xảy ra khi một hoặc nhiều thao tác hoặc chức năng được áp dụng tương phản tới dữ liệu trước khi dữ liệu được chuyển đến đích đến. Dữ liệu tại nguồn thì không thay đổi. Biến đổi dữ liệu thực hiện một cách dễ dàng để thực thi tinh chế dữ liệu, chuyển đổi và xác nhận tính hợp lệ dữ liệu phức tạp trong suốt tiến trình import và export.

9.1.3 Các công cụ chuyển đổi dữ liệu (Data transfer tools)

SQL Server 2000 cung cấp số công cụ dành cho việc import và export dữ liệu. Các công cụ này có những khả năng khác nhau để trích lọc tập các dữ liệu từ nguồn dữ liệu có sẵn và chuyển đổi dữ liệu. Bảng dưới đây mô tả ngắn gọn các công cụ chính và khả năng của nó.

Các công cụ chuyển đổi dữ liệu và chức năng của chúng

Công cụ	Mô tả
DTS	DTS là một công cụ đồ họa dùng để import, export, và transform dữ liệu. DTS có thể làm việc trực tiếp các nguồn dữ liệu đa dạng. DTS tạo các gói (package) mà có thể lập biểu. DTS cũng có thể import và export các lược đồ đối tượng CSDL giữa các thể hiện (instance) của SQL Server.
Bcp	Bcp là một lệnh tiện ích tại dấu nhắc được dùng để sao chép dữ liệu từ một tập tin văn bản thành một bảng hoặc View của SQL Server thông qua ODBC. Khả năng biến đổi dữ liệu của Bcp bị giới hạn và qui định dạng dạng tập tin khó hiểu. Làm việc với CSDL của MicroSoft hoặc hàng thứ 3 là một tiến trình 2 bước.
Lệnh BULK INSERT Transact-SQL	BULK INSERT là một lệnh Transact-SQL dùng để sao chép dữ liệu từ một tập tin văn bản ASCII thành một bảng hoặc View của SQL Server thông qua OLEDB. Câu lệnh BULK INSERT cung cấp chức năng tương tự như Bcp (và cũng hạn chế) trong một câu lệnh Transact-SQL và có thể nhúng trong một gói DTS.

9.2 Dịch vụ chuyển đổi dữ liệu DTS (Data Transformation Services - DTS)

DTS là một tập các công cụ mạnh mà bạn có thể dùng để import, export, và transform dữ liệu đến và đi từ một nguồn và đích dữ liệu đa dạng.

9.2.1 DTS Package.

Một DTS package là một tập hợp có tổ chức của các DTS Connection, DTS Task, DTS Package Workflow, DTS transformation.

Mỗi package gồm có một hoặc nhiều bước mà được thực thi một cách tuần tự hoặc song song khi mà package được chạy.

9.2.2 DTS Connections.

DTS Connection là các kết nối đến dữ liệu nguồn hay đích đến.

Data source connection: Là một kết nối đến một CSDL chuẩn (như là SQL Server, Access, dBase,...), một kết nối OLE DB đến một nguồn dữ liệu ODBC,..

File connection: Là một kết nối đến một tập tin văn bản

Data link connection: Một kết nối đến một tập tin trung gian mà nó lưu trữ một chuỗi kết nối để tạo một kết nối OLE DB mà được thực hiện tại thời điểm chạy.

9.2.3 DTS Tasks.

Là một tập các chức năng rời rạc, được thực thi như là một bước đơn trong một package. Mỗi tác vụ (task) định rõ một mục công việc là:

- Thực thi một câu lệnh T-SQL
- Thực thi một script
- Khởi động một ứng dụng mở rộng.

- Sao chép các đối tượng SQL Server.
- Thực thi hoặc lấy các kết quả từ một DTS package.

Các tác vụ có thể trong DTS Designer

Loại	Tác vụ	Mô tả
Các tác vụ mà sao chép và quản lý dữ liệu và biến đổi dữ liệu	Bulk Insert Task	Dùng để chạy câu lệnh T-SQL BULK INSERT từ trong một DTS package. Tác vụ này hỗ trợ phương cách nhanh nhất để sao chép thông tin vào một bảng or view, nhưng nó không ghi nhận lại (log) các dòng gây lỗi. Nếu bạn cần giữ lại các dòng gây lỗi vào một tập tin, bạn nên sử dụng tác vụ Transform Data task để thay thế.
	Execute SQL task	Dùng để chạy các câu lệnh Transact-SQL trong suốt việc thực thi package. Bạn có thể thực hiện một số thao tác bao gồm việc xóa một bảng và chạy một thủ tục bằng Execute SQL task.
	Copy SQL Server Objects task	Dùng để sao chép các đối tượng của CSDL của SQL Server (siêu dữ liệu - meta data) từ một thể hiện của SQL Server đến thể hiện khác. Tác vụ này có thể chuyển các đối tượng từ một thể hiện của SQL Server 7.0 đến thể hiện khác; từ một thể hiện của SQL Server 7.0 đến SQL Server 2000; hoặc từ thể hiện của SQL Server 2000 đến thể hiện khác của SQL Server 2000.
Các tác vụ mà transform data	Transfer Database Objects tasks	Một tập hợp các task mà chép thông tin server-wide (Copy SQL Server Objects task chỉ sao chép thông tin đặc biệt của CSDL) từ một thể hiện của SQL Server đến thể hiện khác. Những tác vụ này bao gồm Transfer Database task, Transfer Error Messages task, Transfer Logins task, Transfer Jobs task, và Transfer Master Stored Procedures task. Các tác vụ này được sử dụng bởi Copy Database Wizard.
	Transform Data task	Sao chép, biến đổi và chèn dữ liệu từ một data source đến một data destination. Tác vụ này thì hầu hết thực hiện cơ bản của động cơ bơm dữ liệu (data pump engine) trong DTS.
	Data Driven Query task	Chọn, tùy chọn, và thực thi một của vài thao tác Transact-SQL (như là update hoặc delete) trên một dòng dựa trên dữ liệu trong dòng. Sử dụng tác vụ này nếu Transform Data task và Bulk Insert task không phù hợp với các yêu cầu của ứng dụng của bạn.
Các tác vụ mà chức năng như là các công việc	ActiveX Script task	Để chạy một ActiveX script. Bạn có thể dùng tác vụ này để viết code để thực hiện các chức năng mà không có sẵn trong DTS Designer.
	Dynamic Properties task	Truy xuất dữ liệu từ một nguồn bên ngoài và gán giá trị truy xuất được cho các thuộc tính package được chọn. External sources có thể là một tập tin .INI, tập tin

Loại	Tác vụ	Mô tả
		dữ liệu, truy vấn, biến toàn cục, biến môi trường, hoặc một hằng số.
	Execute Package task	Dùng chạy DTS package khác như là một phần của workflow. Không dùng tác vụ này một cách đệ quy bởi vì nó có thể sinh ra tràn stack, mà điều này có thể dẫn đến MMC bị shut down.
	Execute Process task	Dùng để chạy một chương trình hoặc tập tin bó lệnh có khả năng thực thi. Tác vụ này có thể được dùng để mở bất kỳ application chuẩn nào đó, như là Microsoft Excel, nhưng nó được sử dụng chính là để chạy các tập tin bó lệnh hoặc các ứng dụng thương mại mà nó làm việc với một data source.
	File Transfer	Dùng để Download dữ liệu từ một từ một server từ xa bởi Protocol task hoặc một Internet location sử dụng FTP. FTP task và Ftp.exe dùng cùng một phương pháp kết nối.
	Send Mail task	Dùng để gửi một thông báo e-mail message như là một tác vụ. Ví dụ, thông báo có thể được gửi đến nhà quản trị về sự thành công hay thất bại của một thao tác dự phòng (backup). Để sử dụng tác vụ này, bạn cần cài đặt một MAPI client ở thể hiện của SQL Server mà bạn đang chạy.

9.2.4 DTS Package Workflow.

Workflow là lưu đồ làm việc của DTS Package. Bạn có thể định rõ thứ tự thực thi của các bước trong một package với ràng buộc độ ưu tiên.

- **Unconditional:** Nếu Task 2 được kết nối với task 1 bởi một ràng buộc unconditional thì Task 2 sẽ phải đợi cho đến khi Task 1 hoàn tất và sau đó nó sẽ được thực thi, bất chấp sự thành công hay thất bại của Task 1
- **On Success:** Nếu Task 3 được kết nối với Task 2 bởi ràng buộc On Success thì Task 2 sẽ đợi cho đến khi Task 2 hoàn tất, và sau đó sẽ thực thi nếu Task 2 hoàn tất một cách thành công.
- **On Failure** Nếu Task 4 liên kết với Task 2 bởi một ràng buộc On Failure, Task 4 sẽ đợi cho đến khi Task 2 hoàn tất và sau đó sẽ chỉ thực thi nếu Task 2 failed thì hoàn tất một cách thành công.

Các Task không có các ràng buộc độ ưu tiên thực thi song song

9.2.5 DTS Package Storage.

Bạn có thể lưu một DTS package vào SQL Server 2000, SQL Server 2000 Meta Data Services, một tập tin Microsoft Visual Basic, hay một tập tin lưu trữ có cấu trúc. Khi bạn lưu một DTS package thì tất cả DTS connections, tasks, transformations, và workflow đều được lưu lại.

9.3 Thực hiện việc biến đổi và chuyển đổi dữ liệu bằng công cụ đồ họa DTS.

9.3.1 DTS Import/Export Wizard

Là một cách đơn giản để tạo một DTS packages để sao chép dữ liệu giữa các nguồn dữ liệu nhưng nó bị giới hạn đối với những biến đổi dữ liệu phức tạp, thêm nhiều task và các lưu đồ tác vụ phức tạp. DTS Import/Export Wizard được sẵn sàng trong Enterprise Manager.

9.3.2 DTS Designer

Cho phép bạn tạo mới hoặc hiệu chỉnh package bằng các đối tượng đồ họa để hỗ trợ xây dựng DTS package kể cả các lưu đồ phức tạp. DTS Designer sẵn dùng trong Data Transformation Services chứa trong Enterprise Manager.

Các bước tạo một package

- Mở mới một package.
- Khai báo các connection.
- Tạo các tác vụ (task) cần thực hiện.
- Qui định các Workflow giữa các tác vụ nếu cần.
- Lưu, thực hiện package nếu cần.

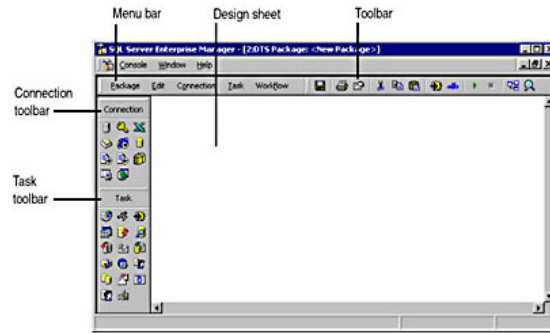
9.3.2.1 Thao tác mở một Package

Mở mới một Package: Trong **Enterprise Manager**, nhấn nút phải chuột tại **Data Transformation Services** và chọn **New Package**.

Mở một Package có sẵn: Khi mở (open) một package có sẵn trong Data Transformation Services tùy thuộc vào cách mà DTS package đó được lưu trữ như thế nào.

- Package lưu trong tập tin có cấu trúc:
 - R_Click tại Data Transformation Services
 - Chọn Open Package để mở tập tin từ đĩa.
- Package lưu trong SQL Server.
 - Click tại hộp Local Packages trong Data Transformation Service.
 - Double-click vào tên của DTS package cần mở.
- Package lưu trong Meta data Services Package.
 - Click tại nhánh Meta data Services Package trong Data Transformation Service.
 - Double-click vào tên của DTS package cần mở.

Trong cửa sổ DTS Designer cho phép bạn tạo các kết nối đồ họa đến nguồn và đích đến dữ liệu. Cấu hình các DTS tasks, thực hiện các DTS transformation, và chỉ định các ràng buộc ưu tiên. Bạn dùng phương pháp drag và drop và bạn tắt các hộp thoại cho các đối tượng để tạo các DTS package trong trang thiết kế. Hình bên dưới hiển thị giao diện người dùng ứng với DTS Designer

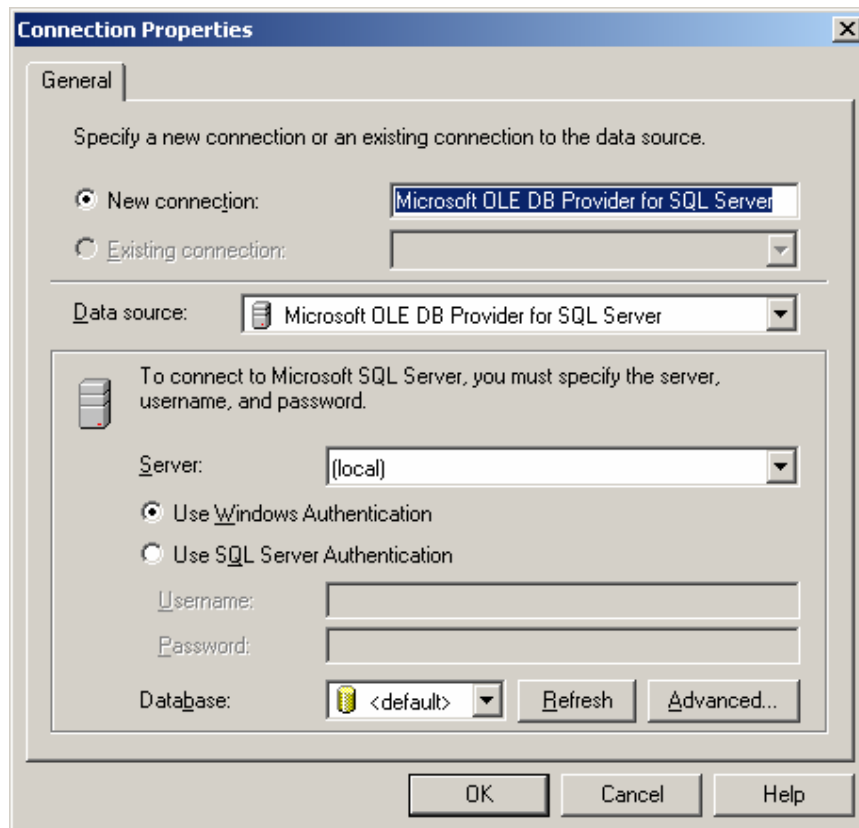


Hình 27: Giao diện thiết kế DTS

9.3.2.2 Khai báo connection

Bước 1: Chọn nguồn dữ liệu (Data Source)

- Drag một đối tượng data source từ thanh công cụ Connection vào cửa sổ thiết kế hoặc chọn Connection → loại data source
→ Hộp thoại Connection Properties xuất hiện khác nhau tùy vào sự chọn lựa data source.
- Hoàn tất hộp thoại để chỉ định data source.

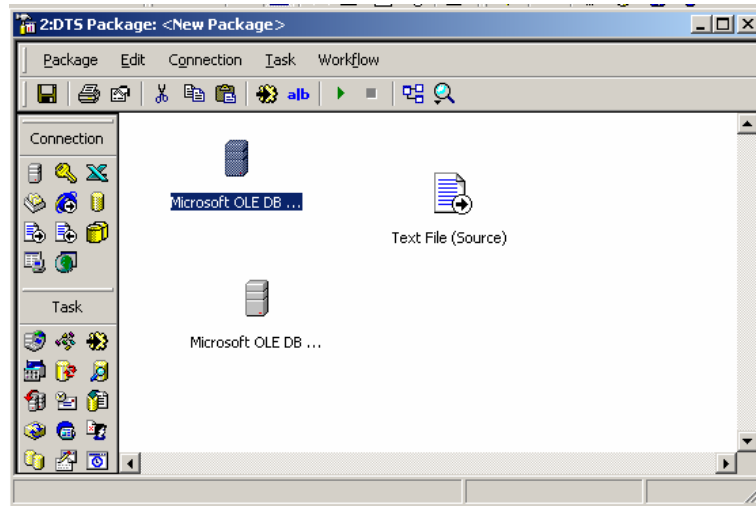


Hình 28: Kết nối đến SQL Server bằng Microsoft OLE DB Provider for SQL Server

Bước 2: Chọn lựa và cấu hình một data destination.

Tương tự như cấu hình một Data Source

Hình 3 hiển thị một bảng thiết kế gồm 3 data source: 2 kết nối đến Microsoft OLE DB Provider for SQL Server và một kết nối đến một tập tin văn bản (Source).



Hình 29: Cài đặt một data Destination.

Lưu ý:

Một Connection đến tập tin văn bản thì phải xác định rõ là một **Text file (Source)** hoặc **Text file (Destination)**.

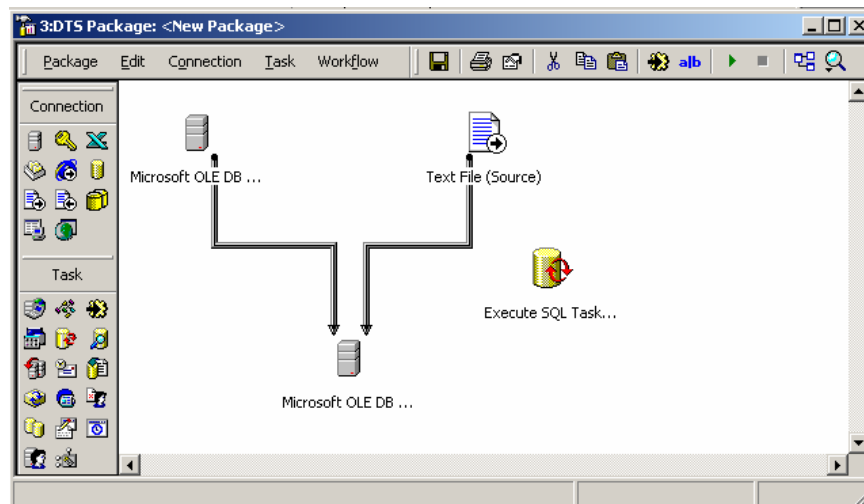
9.3.2.3 Định nghĩa các tác vụ (Task) mà bạn muốn thực hiện

Sử dụng hoặc thực đơn Task hoặc thanh công cụ Task.

Lưu ý:

- Nếu bạn chọn tác vụ Transform dữ liệu, bạn được nhắc nhở xác định data source và data destination. Khi đó một mũi tên màu xám xuất hiện trở từ data source đến data destination.
- Nếu bạn chọn bất kỳ tác vụ khác thì một hộp thoại sẽ xuất hiện để nhắc nhở bạn cấu hình thuộc tính của tác vụ. Tác vụ xuất hiện trong cửa sổ thiết kế như là một biểu tượng.

Hình bên dưới hiển thị 3 tác vụ Transform Data và một tác vụ SQL thực hiện (Execute) tạo một bảng.

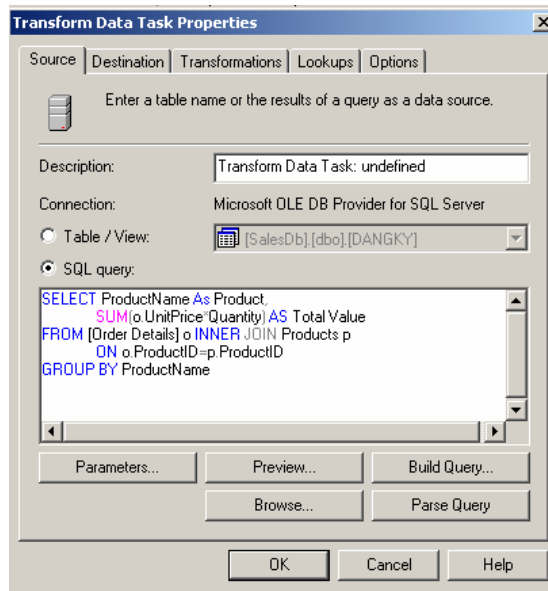


Hình 30: Một DTS với 3 tác vụ transform data và một tác vụ Execute SQL

9.3.2.4 Hiệu chỉnh tác vụ.

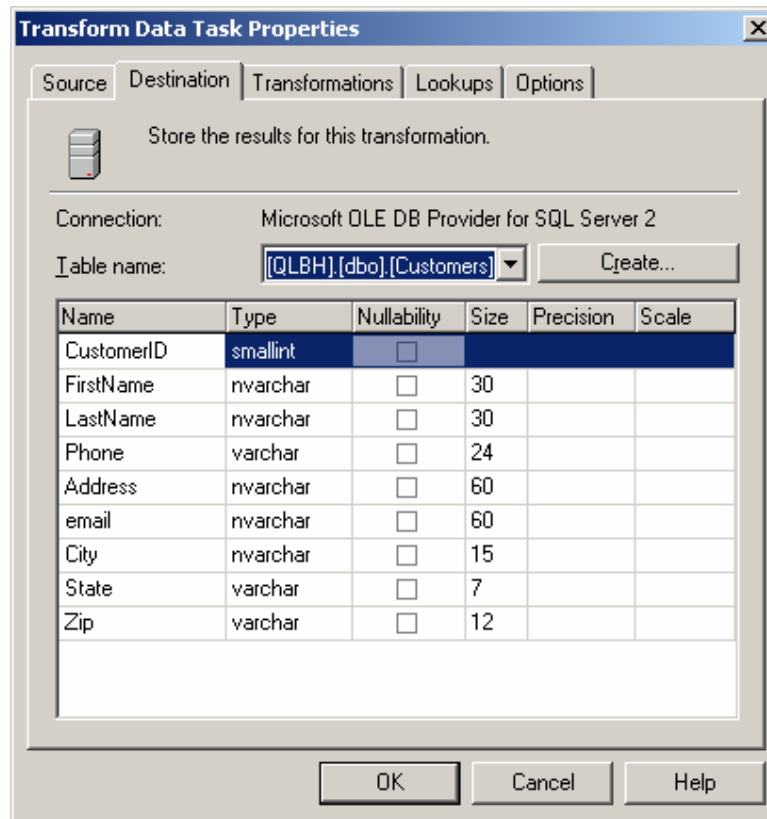
Để hiệu chỉnh và tùy chọn một tác vụ transform dữ liệu, double-click vào mũi tên xám giữa data source và data destination để mở hộp thoại cho tác vụ này.

Trang Source: nếu data source là một CSDL, bạn có thể lọc các dữ liệu được sao chép từ các bảng hoặc View cụ thể được chọn hoặc sử dụng một truy vấn Transact-SQL. Hình 5 hiển thị một truy vấn Transact-SQL được dùng để lọc dữ liệu được import.



Hình 31: Sử dụng một truy vấn để lọc dữ liệu import.

Trang Destination: bạn có thể xác định thông tin về dữ liệu được import (như là các cột đích đến). Những sự lựa chọn của bạn sẽ biến đổi tùy vào data destination. Nếu data destination là một CSDL, bạn có thể tạo và định nghĩa một bảng mới hoặc chọn một bảng có sẵn để import dữ liệu vào.



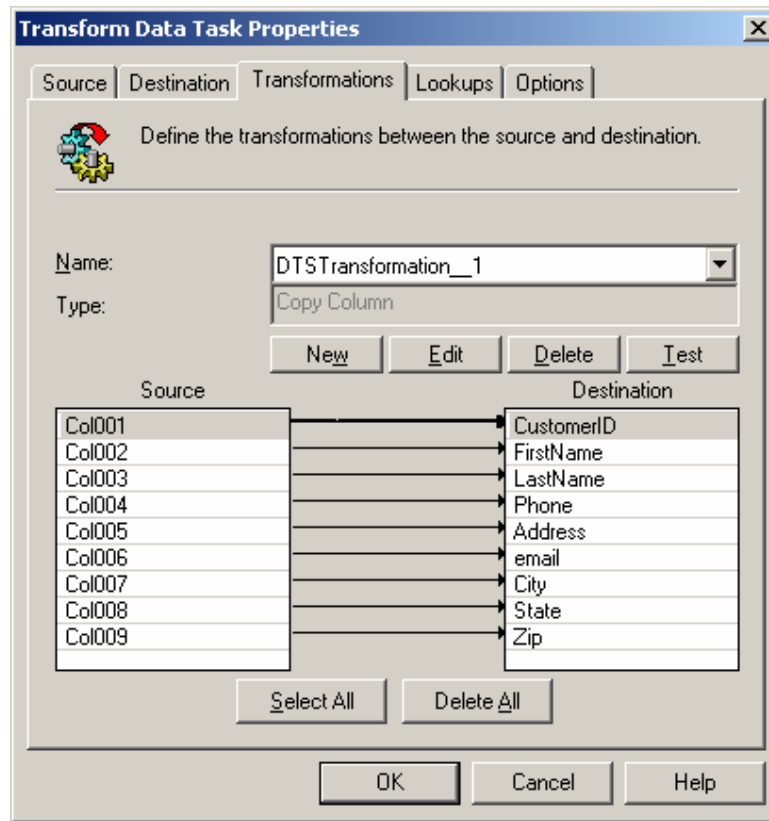
Hình 32: Trang Destination của hộp thoại thuộc tính của tác vụ transform data

Trang Transformations: bạn có thể thiết lập sự biến đổi dữ liệu theo ý người dùng. Theo mặc nhiên, các cột nguồn sẽ được sao chép đến các cột đích mà không cần hiệu chỉnh.

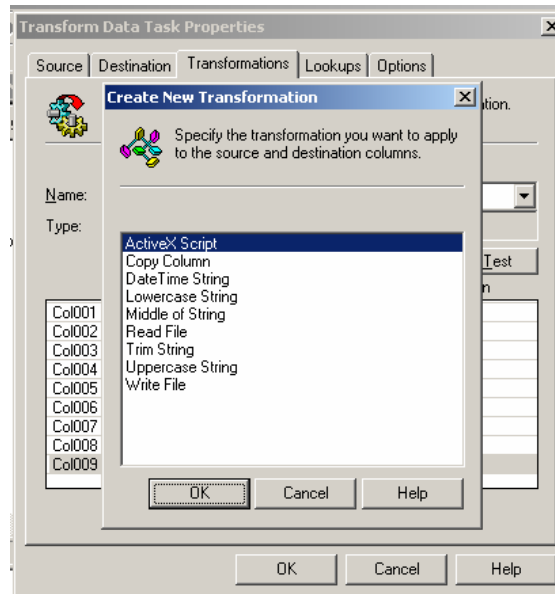
Tạo mới hoặc hiệu chỉnh sự chuyển dữ liệu giữa data source và data destination:

- Chọn cột mà bạn muốn có sự hiệu chỉnh trong danh sách tên đổ xuống hoặc bằng cách click vào mũi tên giữa source và destination.
- Click vào nút **New** hoặc **Edit** để tạo một transformation mới hoặc hiệu chỉnh một transform có sẵn. (double-click vào mũi tên màu đen để hiệu chỉnh một transform).

Nếu bạn click vào nút New, bạn có thể chọn kiểu transform mà bạn muốn từ danh sách các transform sẵn sàng trong việc tạo mới một transform.



Hình 33: Trang Transformations của hộp thoại thuộc tính của tác vụ transform data



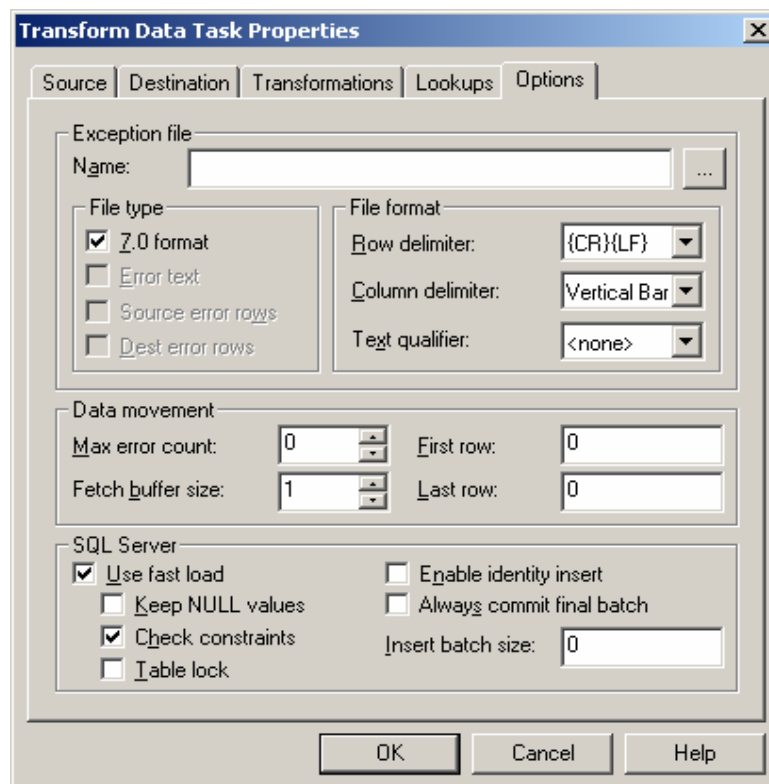
Hình 34: Tạo một transform mới

Nếu bạn chọn ActiveX Script từ hộp thoại Create New Transformation, bạn có thể tạo một transform script mới để thực thi các transform phức tạp hơn xem hình 10

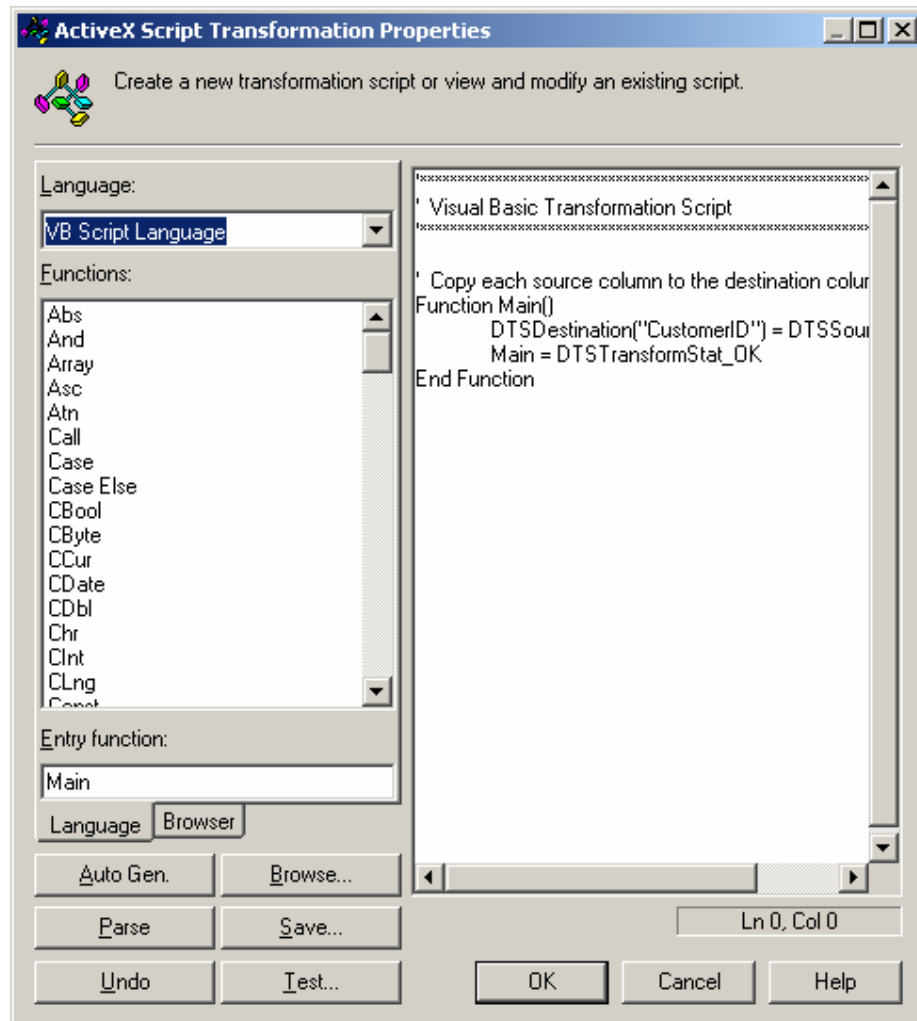
Trang Lookups: bạn có thể định nghĩa một truy vấn dò tìm. Một truy vấn dò tìm đòi hỏi một data connection để chạy các truy vấn và các thủ tục (stored procedure) tương ứng với một data source và data destination. Sử dụng một truy vấn dò tìm để tìm ra thông tin dạng

danh sách, thực thi các cập nhật song song trên hệ thống nhiều CSDL, xác nhận tính hợp lệ nhập vào trước khi tải dữ liệu, gọi các thủ tục trong sự trả lời đến điều kiện nhập dữ liệu và sử dụng các giá trị biến toàn cục như là các tham số của truy vấn.

Trang Options: bạn định nghĩa một số thuộc tính thêm vào cho transform. Bạn có thể định nghĩa một hoặc nhiều tập tin ngoại lệ (exception file) dùng để ghi nhận lại các exception record trong suốt quá trình thực thi package. Tập tin này có thể ở trên ổ đĩa cục bộ hoặc mạng. Bạn có thể tách các lỗi của source và destination trong các tập tin riêng lẻ. Bạn cũng có thể định số lỗi tối đa cho phép trước khi việc thực thi package bị dừng. Cuối cùng, bạn cũng có thể xác định các thuộc tính thực thi đặc biệt khi destination connection là Microsoft OLE DB Provider for SQL Server. Các thuộc tính này bao gồm việc chỉ định thực thi bulk-copy với tốc độ cao, kiểm tra ràng buộc trong suốt quá trình thực thi package, kiểu khóa, kích cỡ của khối, và nhận dạng các thuộc tính chèn....



Hình 35: Trang khai báo một số option



Hình 36: Tạo một transformation script mới

Một khi bạn có cài đặt một tác vụ Data Transformation task và bất kỳ các tác vụ DTS khác thì DTS package sẽ thực thi, bạn phải cấu hình các ràng buộc quyền ưu tiên. Trong ví dụ, chúng ta có 2 data source riêng lẻ để chép đến data destination. Chúng ta cũng có một tác vụ tạo bảng (Create Table task). Bạn dùng ràng buộc quyền ưu tiên để xác định thứ tự của việc thực thi mỗi tác vụ. Để thiết lập độ ưu tiên của khối công việc, hãy chọn 2 hay nhiều bảng theo thứ tự tác vụ sẽ được thực thi và sau đó chọn loại workflow từ thực đơn Workflow. Ví dụ: Nếu tác vụ Create Table phải thực thi trước khi sao chép dữ liệu đến data destination, Hãy chọn ràng buộc ưu tiên là On Success từ thực đơn Workflow. Xem hình.

Bạn có thể tạo một tác vụ Send Mail và cấu hình ràng buộc ưu tiên On Failure precedence giữa tác vụ tạo bảng (Create Table) và tác vụ gửi mail (Send Mail). Điều này có nghĩa là sẽ gửi một thông báo e-mail đến nhà quản trị nếu tác vụ Create Table bị fail. Khi bạn đang dùng tự động hoàn toàn hay các DTS packages được lập lịch để thực thi các quá trình hoạt động của CSDL thì thông báo lỗi là cần thiết

9.4 Dùng BULK COPY (BCP) và BULK INSERT

Bulk copy là một giao diện chương trình ứng dụng (Application Programming Interface – API) được xây dựng trong SQL Server mà cho phép xuất nhập dữ liệu một cách hiệu quả. Ta dùng BCP trong trường hợp trong một lần ta muốn chuyển một số lượng lớn dữ liệu từ

các hệ quản trị CSDL khác vào trong SQL Server. BCP là một phương pháp cơ bản chuyển đổi dữ liệu vào ra SQL Server.

BCP là một API được định nghĩa bởi SQL Server 2000. Hiện nay, giao diện này chỉ được đưa ra thông qua ODBC (Open Database Connectivity- Hệ thống kết nối CSDL mở) và DB-LIB (DataBase Library). SQL Server cung cấp một tiện ích (BCP.EXE) mà hầu hết người dùng quen nghĩ là BCP. Ta có 2 phương pháp để xuất nhập dữ liệu là CBP command-line utility, và BULK INSERT Transact-SQL statement.

Dạng dữ liệu

Dạng dữ liệu	bcp Switch	Lệnh Bulk Insert	Ý nghĩa
Native	-n	DataFileType='native'	Kiểu dữ liệu là tự nhiên, khi cập nhật phải tốn nhiều thời gian chuyển đổi kiểu dữ liệu
Character	-c	DataFileType='char'	Kiểu dữ liệu là CHAR cho tất cả các cột
Unicode character	-w	DataFileType='widechar'	Kiểu dữ liệu là Unicode char cho tất cả các dữ liệu, có thể một số dữ liệu sẽ bị mất nếu kiểu dữ liệu của Source và Destination không trùng nhau.
Unicode Native	-N	DataFileType='widenative'	Kiểu dữ liệu là Native cho những dữ liệu không là kiểu Char, và dùng Unicode cho những dữ liệu kiểu char (char, Varchar, nchar, text...)

BCP Command-line Utility: là một hàm đơn giản của Bulk Copy, thực thi tại dấu nhắc lệnh, dùng để sau chép dữ liệu vào SQL Server hoặc xuất dữ liệu ra thành một tập tin dữ liệu. Muốn dùng được bcp, thì trước tiên dữ liệu trong chương trình nguồn (một DBMS khác) phải được sang một tập tin dữ liệu vào sau đó mới chuyển dữ liệu từ tập tin data vào table của SQL. Ngược lại, bcp sẽ chuyển dữ liệu từ một table thành một tập tin dành cho chương trình khác chẳng hạn như Excel.

Data file: là một tập tin hệ thống hoạt động, dùng chứa dữ liệu để Import vào SQL Server (gọi là Source) hoặc tập tin chứa dữ liệu khi dùng Export (gọi Destination) .

```
bcp [database_name.][owner].table_name in | out data_file [-c | -n | w | N] [-T]
```

Ví dụ: bcp Northwind..customers out d:\data\kh.txt -c -T

-T: chỉ định là Connect trực tiếp với Server đang log on

```
SELECT * INTO Cust FROM Customers
```

```
TRUNCATE TABLE Cust
```

```
bcp Northwind..cust IN d:\data\kh.txt -c -T
```

CÂU LỆNH BULK INSERT: dùng để chuyển một dữ liệu vào bảng của SQL Server từ một tập tin dữ liệu khác. Câu lệnh Bulk insert không thể đưa dữ liệu từ bảng của SQL Server ra thành tập tin dữ liệu dành cho chương trình khác. Tuy vậy, nó lại là một câu lệnh trong T-SQL nên có thể được dùng một cách thuận tiện trong các hàm hoặc thủ tục trong SQL Sever.

```
BULK INSERT 'database_name' ['owner' .] 'table_name' FROM 'data_file'  
[ WITH (DATAFILETYPE = 'char' | 'native' | 'widechar' | 'widenative') ]
```

```
BULK INSERT dbo.Employees FROM 'C:\DATA\Employees.TXT'  
WITH (DATAFILETYPE = 'widechar')
```

```
BULK INSERT dbo.Customers FROM 'C:\DATA\Customers.TXT'  
WITH (DATAFILETYPE = 'widechar')
```

Lưu ý:

- Số cột hoặc số thứ tự của các cột của file text và của bảng không tương thích thì không chèn dữ liệu vào được. Cách giải quyết?
- Nếu trong file Text có giá trị của cột Identity thì sẽ chấp nhận giá trị đó, nếu không có thì nó tự phát sinh.
- Khi Export vào một tập tin, tập tin sẽ bị ghi đè nếu như tập tin đó hiện tồn tại.
- Khi BCP đang import dữ liệu vào bảng, người dùng khác vẫn có thể đang dùng bảng đó. Những dòng được Import chỉ hiện hữu khi tập lệnh import kết thúc.
- Khi BCP đang import dữ liệu, người dùng khác vẫn truy xuất dữ liệu bình thường.
- Khi đang import dữ liệu vào bảng, có 2 phương pháp có thể sử dụng: Logged và no-logged BCP.

BÀI 10:

CƠ BẢN VỀ LẬP TRÌNH BẢNG TRANSACT-SQL

10.1 Khái niệm cơ bản.

10.1.1 Định danh -IDENTIFIERS.

Tên của tất cả các đối tượng đều được gọi là định danh. Mọi thứ trong SQL Server đều có một định danh, trong đó bao gồm Servers, Databases, và các đối tượng (Object) của CSDL như bảng, Views, cột, chỉ mục, ràng buộc,... Có những đối tượng bắt buộc phải qui định một định danh, ngược lại có một số đối tượng không cần định danh (SQL Server sẽ tự động định danh)

Ví dụ:

```
CREATE TABLE Table1
(Keycol INT PRIMARY KEY, Description NVARCHAR(30))
```

Qui tắc định danh:

- Tối đa 128 ký tự.
- Bắt đầu là một ký tự thông thường A -> Z
- Bắt đầu là một ký hiệu (@, #) sẽ có một ý nghĩa khác.
- Những định danh nào có khoảng trắng ở giữa thì phải kẹp trong dấu [] hoặc “
- Đặt các định danh sao cho ngắn gọn, đầu đủ ý nghĩa, phân biệt giữa các đối tượng với nhau, không trùng lặp, không trùng với từ khóa của T-SQL.

10.1.2 Tham chiếu đến các đối tượng trong SQL Server.

- Tên đầy đủ: **Server.database.owner.object**
- Tên ngắn: Nếu là local server thì ta khỏi chỉ Server, nếu ở CSDL hiện hành thì không cần chỉ Database, Owner mặc định là user name trong Database.

Nếu Tham chiếu tường minh thì tăng tốc.

Ví dụ:

```
CREATE TABLE Northwind.dbo.orederhist
CREATE TABLE Northwind..orederhist
```

Database Owner (dbo)

Dbo là một người dùng mà có đầy đủ các quyền thao tác trong CSDL. Bất kỳ một thành viên thuộc nhóm **sysadmin** đóng vai trò người dùng CSDL được gọi là **dbo**. Cũng như bất kỳ một đối tượng nào được tạo bởi bất kỳ thành viên thuộc nhóm **sysadmin** thì mặc nhiên thuộc **dbo**.

Ví dụ: nếu người dùng **Andrew** là thành viên của nhóm **sysadmin** tạo bảng **T1**, thì **T1** thuộc chủ quyền **dbo** và nó thuộc **dbo (dbo.T1)**, không thuộc **Andrew** (không là **Andrew.T1**). Ngược lại, nếu **Andrew** không là thành viên của **sysadmin** nhưng chỉ là một thành viên của **db_owner** và tạo bảng **T1**, **T1** thuộc về **Andrew** và đủ tư cách gọi là **Andrew.T1**.

User **dbo** không bao giờ bị xóa và nó luôn luôn hiện hữu trong mỗi CSDL. Chỉ có những đối tượng được tạo bởi thành viên của sysadmin (hoặc bởi user dbo) thì thuộc về dbo.

10.1.3 Kiểu dữ liệu (DATA TYPE).

Kiểu dữ liệu là một định nghĩa để xác định loại dữ liệu mà đối tượng có thể chứa đựng. Cột, tham số, biến, giá trị trả về của hàm, thủ tục, tất cả đều phải có kiểu dữ liệu.

Có 2 loại kiểu dữ liệu: **System-Supplied datatype** và **User-defined data types**

(Đã được trình bày trong bài trước)

10.1.4 Batch

Batch là một tập các phát biểu T-SQL nằm liên tiếp và kết thúc bởi phát biểu GO, và được biên dịch đồng thời bởi SQL Server.

Ví dụ:

```
USE pubs
GO /* Signals the end of the batch */

CREATE VIEW auth_titles
AS
SELECT *
FROM authors
GO /* Signals the end of the batch */

SELECT *
FROM auth_titles
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, Create Index,...

10.1.5 Kịch bản - SCRIPT

Một Script là một tập của một hay nhiều bó lệnh được lưu lại thành một tập tin .SQL

10.2 Biến (VARIABLES)

Biến là một đối tượng trong tập lệnh T-SQL mà nó dùng để lưu trữ dữ liệu. Sau khi biến đã được 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 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
-

Các loại biến: có 2 loại biến là biến cục bộ (local) và biến toàn cục (Global). Biến kiểu Global được SQL Server đưa ra và bạn có thể dùng bất cứ khi nào và cũng không cần khai báo (được xem như là những hàm chuẩn của SQL Server).

Local variable

- 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 hoặc Function mà nó được khai báo.
- Tên của biến bắt đầu @

Khai báo:

```
DECLARE @var_name var_type
```

Gán giá trị cho biến:

Nếu biến vừa khai báo xong thì biến mặc nhiên được gán giá trị là NULL.

Để gán biến ta dùng lệnh SET hoặc dùng câu lệnh SELECT

```
SET @var_name = expression
SELECT { @var_name = expression } [,...n]
```

Ví dụ 1:

```
USE Northwind
DECLARE @EmpIDVar INT
SET @EmpIDVar=3
SELECT * FROM [Orders]
WHERE
```

Ví dụ 2:

```
DECLARE MyVariable INT
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 [Orders]
WHERE
```

Ví dụ 3:

```
USE Northwind
```

```
GO
-- Khai báo 2 biến.
DECLARE @FirstNameVariable NVARCHAR(20),
@RegionVariable NVARCHAR(30)

-- Gán giá trị cho 2 biến.
SET @FirstNameVariable = N'Anne'
SET @RegionVariable = N'WA'

-- Dùng chúng trong mệnh đề WHERE của lệnh SELECT.
SELECT LastName, FirstName, Title
FROM Employees
WHERE FirstName = @FirstNameVariable
      OR Region = @RegionVariable
GO
```

Ví dụ 4:

```
USE Northwind
GO
-- Khai báo 1 biến
DECLARE @EmpIDVariable INT
-- Gán giá trị biến bằng câu lệnh Select
SELECT @EmpIDVariable = MAX(EmployeeID)
FROM Employees
GO
-- Nếu câu Select trả về là một tập giá trị thì biến sẽ nhận giá trị sau cùng.
```

Ví dụ 5:

```
DECLARE @ProdIDVariable int
SELECT @ProdIDVariable = ProductID
FROM Northwind..Products
```

Ví dụ 6:

```
USE Northwind
GO
DECLARE @EmpIDVariable INT
SELECT @EmpIDVariable = EmployeeID
FROM Employees
ORDER BY EmployeeID DESC
SELECT @EmpIDVariable
GO
```

Global variable trong SQL Server 2000 gọi là System Function: 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 hàm hệ thống thường dùng:

@@VERSION	phiên bản của SQL Server và hệ điều hành
	SELECT @@VERSION
@@TRANCOUNT	Xem coi có bao nhiêu transaction đang mở
	<pre>IF (@@TRANCOUNT > 0) BEGIN RAISERROR('Task cannot be executed within a transaction.', 10, 1) RETURN END</pre>
@@ROWCOUNT	Trả về số dòng bị ảnh hưởng đối với lệnh thực thi gần nhất
	<p>Ví dụ 1:</p> <pre>USE Northwind UPDATE Employees SET LastName = 'Brooke' WHERE LastName = 'Brook' IF (@@ROWCOUNT = 0) BEGIN PRINT 'Warning: No rows were updated' RETURN END</pre> <p>Ví dụ 2:</p> <pre>UPDATE Customers SET Phone ='030' + Phone WHERE Country='Germany' PRINT @@ROWCOUNT</pre>
@@IDENTITY	trả về số Identity phát sinh sau cùng
	<pre>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)</pre>

	<pre> Insert into Table_CTHD values (@maso, 'sp002',10) -- Kiểm tra SELECT * FROM Table_HD SELECT * FROM Table_CTHD </pre>
@@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)

Một số hàm thường dùng:

GetDate()	Lấy ngày, giờ hiện hành của hệ thống
Month(Date); Year(Date)	Lấy tháng, năm của ngày Date
DateAdd(Datepart, Number, Date)	Cộng thêm Date một giá trị số
DATEDIFF (datepart, startdate, enddate)	Khoảng chênh lệch giữa startdate và enddate
DATEPART (datepart, date)	Trả về số nguyên biểu diễn datepart nào đó của ngày được chỉ định
CAST (expression AS data_type)	Dùng để chuyển đổi kiểu dữ liệu
CONVERT (data_type [(length)], expression [, style])	Dùng để chuyển đổi kiểu dữ liệu
LOWER (character_expression)	Chuyển sang chữ thường
UPPER (character_expression)	Chuyển sang chữ hoa
REPLACE ('string_expression1', 'string_expression2', 'string_expression3')	Thay thế chuỗi biểu thức
DIFFERENCE (character_expression, character_expression)	So sánh 2 biểu thức

Ví dụ:

```

USE pubs
SELECT 'The price is ' + CAST(price AS
varchar(12))
FROM titles
WHERE price > 10.00
GO

```

```
-- Use CONVERT.
```

```

USE pubs
SELECT SUBSTRING(title, 1, 30) AS Title, ytd_sales
FROM titles
WHERE CONVERT(char(20), ytd_sales) LIKE '3%'
GO

```

10.3 Cấu trúc điều khiển.

T-SQL cung cấp một số cấu trúc điều khiển cơ bản để bạn có thể thực thi một khối lệnh dựa trên kết quả của một phép so sánh. Nó cũng tương tự như một số ngôn ngữ lập trình khác.

10.3.1 Khối BEGIN ... END

Nếu bạn cần nhiều phát biểu được thực thi với nhau thì ta đặt các phát biểu trong cặp Begin ... End. Nó được hữu dụng trong các cấu trúc điều khiển.

10.3.2 Phát biểu PRINT

Phát biểu PRINT: Dùng để in thông tin ra màn hình kết quả của SQL

```
PRINT 'any ASCII text' | @local_variable | @@FUNCTION | string_expr
```

Ví dụ:

```
PRINT 'Hello!'  
PRINT N'Chào bạn'  
PRINT @@VERSION
```

10.3.3 Cấu trúc điều khiển IF ... ELSE

Là một cấu trúc điều kiện, cho phép thực thi một hoặc nhiều phát biểu tùy thuộc vào một điều kiện nào đó. câu lệnh để thực thi một khối các câu lệnh theo một điều kiện nào đó.

Cú pháp:

```
IF condition  
    {statements}  
[ ELSE [Condition 1]  
    {statements}]
```

Condition: là một biểu thức logic, có giá trị *True* hoặc *False*. Tùy thuộc vào *condition*, một trong hai khối lệnh sẽ được thực thi.

Ví dụ 1: Kiểm tra xem trong Customers của NorthWind có chứa các khách hàng đến từ Germany không?

```
USE NorthWind  
IF (SELECT COUNT(*) FROM Customers  
    WHERE Country='Germany') > 0  
    BEGIN  
        Print ' Có tồn tại các khách hàng từ  
        Germany ở trong cơ sở dữ liệu.'  
        Print ' statements to process German  
        customers '  
    END
```

```
ELSE
    BEGIN
        PRINT ' Không có khách hàng đến từ Germany
trong cơ sở dữ liệu.'
    END
```

Ví dụ 2:

```
USE pubs
GO
DECLARE @msg varchar(255)
IF ( SELECT COUNT(price) FROM titles
WHERE title_id LIKE 'TC%' AND price BETWEEN 10 AND
20) > 0
BEGIN
    SET NOCOUNT ON
    SET @msg = 'Có vài quyển sách có giá từ $10 đến
$20. Các sách đó là:'
    PRINT @msg
    SELECT title FROM titles
    WHERE title_id LIKE 'TC%' AND price BETWEEN 10
AND 20
END
ELSE
BEGIN
    SET NOCOUNT ON
    SET @msg = 'Không có quyển sách nào có giá từ
$10 đến $20. Bạn nên tham khảo các quyển sách có
giá nhỏ hơn $10 sau đây.'
    PRINT @msg
    SELECT title FROM titles
    WHERE title_id LIKE 'TC%' AND price < 10
END
```

Ví dụ 3:

```
USE pubs
IF (SELECT AVG(price) FROM titles WHERE type =
'mod_cook') < $15
BEGIN
    PRINT 'The following titles are excellent
mod_cook books:'
    PRINT ' '
    SELECT SUBSTRING(title, 1, 35) AS Title
    FROM titles WHERE type = 'mod_cook'
```

```

END
ELSE
    IF (SELECT AVG(price)
        FROM titles WHERE type = 'mod_cook') >
        $15
    BEGIN
        PRINT 'The following titles are expensive
mod_cook books:'
        PRINT ' '
        SELECT SUBSTRING(title, 1, 35) AS Title
        FROM titles WHERE type = 'mod_cook'
    END

```

10.3.4 Biểu thức 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 tùy thuộc vào điều kiện hoặc một điều khiển nào đó.

Dạng 1:

```

CASE input_expression
  WHEN when_expression THEN result_expression
  [...]
  [
    ELSE else_result_expression
  ]
END

```

Dạng 2:

```

CASE
  WHEN Boolean_expression THEN result_expression
  [...]
  [
    ELSE else_result_expression
  ]
END

```

Ví dụ 1: Cho 2 số a và b, so sánh 2 số đó, số nào lớn 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
```

Ví dụ 2: Dựa vào price của các title cho biết price đó như thế nào?

```
Use Pub
```

```
SELECT title, price,
'classification'=CASE
    WHEN price < 10.00 THEN 'Low Priced'
    WHEN price BETWEEN 10.00 AND 20.00 THEN 'Moderately
Priced'
    WHEN price > 20.00 THEN 'Expensive'
    ELSE 'Unknown'
END
FROM titles
```

Ví dụ 3: Cho biết ý nghĩa của đoạn lệnh sau:

```
USE NorthWind
```

```
SELECT ProductID, Quantity, UnitPrice, [discount%]=
    CASE
        WHEN Quantity <=5 THEN 0.05
        WHEN Quantity BETWEEN 6 and 10 THEN 0.07
        WHEN Quantity BETWEEN 11 and 20 THEN 0.09
    ELSE
        0.1
    END
FROM [Order Details]
ORDER BY Quantity, ProductId
```

Ví dụ 4: Ý nghĩa của đoạn lệnh sau?

```
SELECT title, pub_id,
    CASE WHEN price IS NULL THEN (SELECT MIN(price)
        FROM titles)
    ELSE price
    END
FROM titles
```

10.3.5 Cấu trúc vòng lặp WHILE ...

Là phát biểu điều khiển vòng lặp. Vòng lặp sẽ thực hiện cho đến khi biểu thức điều kiện (*Boolean_expression*) trong While mang giá trị False. Biểu thức điều kiện có thể là một câu SELECT.

```
WHILE Boolean_expression
{ sql_statement | statement_block }
[ BREAK ]
```

```
{ sql_statement | statement_block }
[ CONTINUE ]
```

[**BREAK**] : Dừng để kết thúc vòng lặp khi gặp một trường hợp cụ thể nào đó.

[**CONTINUE**] : Lặp lại đầu vòng lặp.

Thông thường 2 từ khóa Break và Continue phải nằm trong trong cấu trúc If ... Else..

Ví dụ 1: Trong khi đơn giá trung bình vẫn còn nhỏ hơn \$30 thì cập nhật các đơn giá tăng lên gấp đôi đơn giá cũ.

```
USE pubs
GO
WHILE (SELECT AVG(price) FROM titles) < $30
BEGIN
    UPDATE titles
        SET price = price * 2
    SELECT MAX(price) FROM titles
    IF (SELECT MAX(price) FROM titles) > $50
        BREAK
    ELSE
        CONTINUE
END
PRINT 'Too much for the market to bear'
```

10.3.6 Lệnh RETURN

```
RETURN [ integer_expression ]
```

Nếu gặp phát biểu Return, quá trình xử lý sẽ kết thúc

Đôi khi ta dùng RETURN trong thủ tục để thủ tục có thể trở thành hàm như các ngôn ngữ khác.

10.3.7 Lệnh WAITFOR

Là một chỉ thị cho SQL Server tạm dừng một thời gian trước khi xử lý tiếp các phát biểu sau đó.

```
WAITFOR { DELAY 'time' | TIME 'time' }
```

‘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 dừng đến thời gian ‘Time’ chỉ ra.

```
WAITFOR    ' 02:10'
```

```
WAITFOR    ' 02:10'
```

10.3.8 Lệnh RAISERROR

Phát sinh lỗi của 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 } {, severity, state }
           [, argument [...n] ])
           [ WITH option [...n] ]
```

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.

Đề 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

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

severity: Độ nghiêm khắc 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. Bạn 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, bạn có thể tiếp tục làm việc mặc dù bạn 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 (Ví dụ như lock hoặc không gian đĩa cho CSDL) 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, bạn dùng DBCC CHECKDB (Database) 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 RAISERROR, 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.

Thêm một lỗi mới của người dùng định nghĩa:

```
Sp_addMessage msg_ID, severity, 'msg' [, 'language' ] [, 'with_log' ] [, 'replace' ]
```

Xóa một lỗi của người dùng

```
sp_dropmessage Msg-ID
```

Giải thích

msg_id: là mã số của lỗi mới, là một số *int*, không được trùng các mã đã có sẵn, bắt đầu là 50001.

severity: là mức lỗi của lỗi, là một số *smallint*. Mức hợp lệ là từ 1 đến 25. Chỉ có người quản trị CSDL mới có thể phát sinh thêm một thông báo lỗi mới từ 19 đến 25.

'msg': là một chuỗi thông báo lỗi, tối đa 255 ký tự.

'language': là ngôn ngữ của thông báo lỗi, không chỉ định thì mặc định là ngôn ngữ của phiên kết nối.

'with_log': thông báo lỗi có được ghi nhận vào nhật ký của ứng dụng khi nó xảy ra hay không, mặc định là **FALSE**. Nếu là **true**, thì lỗi luôn luôn được ghi vào nhật ký ứng dụng. Chỉ có những thành viên thuộc **sysadmin** server role mới có thể sử dụng tham số này.

'replace': nếu được chỉ định chuỗi **REPLACE**, thì thông báo lỗi đã tồn tại được ghi đè bởi chuỗi thông báo mới và mức lỗi mới. Tham số này phải chỉ định nếu *msg_id* đã có.

Lưu ý: nếu trả về 0 tức là thêm vào thành công, 1 thất bại.

BÀI 11: PROCEDURES, FUNCTIONS

11.1 STORED PROCEDURES.

11.1.1 Giới thiệu Stored procedures.

Stored procedure (thủ tục) là một tập các lệnh T-SQL và một số cấu trúc điều kiện, 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.

- 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 đắn 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.

Phân loại:

- **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 xp.
- **Remote sp:** là một thủ tục được gọi thực thi từ một server từ xa.

11.1.2 Tạo, thực thi, hiệu chỉnh, xóa stored procedures.

- Khi thực thi thủ tục, bạn phải cung cấp các giá trị của tham số của các thủ tục nếu có. Một thủ tục có thể được gọi thực thi hoặc tự động thực thi khi SQL Server khởi động. Gọi thực thi bằng từ khóa EXECUTE.

- Khi cần thêm một tham số (parameter) hoặc thay đổi một vài phần trong đoạn mã thì ta dùng lệnh ALTER để hiệu chỉnh.
- Xóa một thủ tục dùng lệnh DROP
- Các thủ tục có thể được tạo trước khi các đối tượng mà thủ tục tham chiếu, đặt tính này gọi là tính trì hoãn.

Tạo thủ tục

Cách 1: Dùng Enterprise Manager

R-Click tại Store procedure trong CSDL, chọn New Store procedure

Cách 2: Tạo Stored procedure Wizard

Tool → Wizard, click vào DataBase, chọn Create Store Procedure Wizard

Cách 3: Bằng lệnh Create procedure

```
CREATE PROC [ EDURE ] procedure_name [ ; number ]
[ { @parameter data_type } [ VARYING ] [ = default ] [ OUTPUT ]
] [,...n] [ WITH { RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION } ] [
FOR REPLICATION ]
AS
sql_statement [ ...n ]
```

[VARYING]: Chỉ dùng với biến Cursor

Ví dụ 1: Tạo thủ tục để liệt kê các order có ngày giao hàng đã quá hạn theo yêu cầu

```
CREATE PROC dbo.overdueOrders - quá hạn
AS
SELECT *
FROM dbo.orders
WHERE RequiredDate < GETDATE() and shippeddate is
null
```

Kiểm tra sự tồn tại của Stored procedures

```
sp_helptext proc_name
```

Thực thi một Stored procedures

```
[[ EXEC [ UTE ] ]
{
[ @return_status = ]
{ procedure_name [ ; number ] | @procedure_name_var
}
[[ @parameter = ] { value | @variable [ OUTPUT ] } [[ DEFAULT ] ]
[,...n]
[ WITH RECOMPILE ]
```

Đơn giản hơn

```
EXECUTE ProductName [ ; number ] [<parameter>[, ...n] [ OUTPUT ]]
```

Ví dụ: EXECUTE dbo.overdueOrders

Thực thi sp ngay khi SQL Server khởi động: Store procedure phải nằm ở CSDL Master.

Cách 1: Dùng thủ tục **Sp_procoption** để gán thuộc tính tự động thực thi

```
USE Master
EXECUTE Sp_procoption [ @ProcName = ] 'procedure'
, [ @OptionName = ] Startup
, [ @OptionValue = ] True
```

Ví dụ:

```
EXECUTE Sp_procoption dbo.overdueOrders Startup, True
```

Cách 2: Dùng **Enterprise Manager**

R-Click tại tên thủ tục → Properties → Execute whenever SQL Server Start

Hiệu chỉnh một stored procedures

```
USE Northwind
GO
ALTER PROC dbo.overdueOrders
AS
    SELECT CONVERT (CHAR (8), RequiredDate, 1)
    RequiredDate, CONVERT (CHAR (8), orderDate, 1)
    orderDate, orderId, Customerid, EmployeeID
    FROM dbo.orders
    WHERE RequiredDate < GETDATE () and shippeddate is
    null
    ORDER BY RequiredDate
```

Xóa một stored procedures

```
DROP ProcedureName
```

11.1.3 Tham số và biến trong Stored procedures.

Tham số và biến là phần cơ bản để tạo nên sự uyển chuyển của thủ tục.

- **Input parameter:** tham số nhập, đưa giá trị của tham số để thông báo cho thủ tục nên làm gì trong CSDL
- **Output parameter:** tham số xuất chứa giá trị trả về của thủ tục.

Khi dùng tham số phải khai báo tham số (Tên tham số, kiểu dữ liệu, Giá trị mặc nhiên nếu có, có chỉ dẫn tham số OUT PUT không)

```
@parameter_name [AS] datatype
[=default | NULL] [VARYING] [OUTPUT]
```

Ví dụ 1: Tạo thủ tục dùng để chèn một mẫu tin vào bảng Customer, với các tham số dạng Input

```
USE SalesDB
GO
```

```
CREATE PROC Sp_InsertCust
@No_para VARCHAR(10),
@Name_para NVARCHAR(50),
@Address_para NVARCHAR(50),
@Phone_para VARCHAR(24),
@Fax_para VARCHAR(24),
@Mail_para VARCHAR(50)
AS
INSERT INTO tblCustomer
(CustNo, CustName, Address, Phone, Fdax, Mail)
VALUES (@No_para, @Name_para, @Address_para,
@Phone_para, @Fax_para,@Mail_para)
```

Thực thi thủ tục có tham số

```
USE SalesDB
GO
Sp_InsertCust 'CDS', 'Trường Tin học ABC','12
Nguyễn Văn Bảo', '2352344',234652','cds@yahoo.com'
```

Kiểm tra việc chèn dữ liệu

```
SELECT CustMo, CustName, Address
FROM tblCustomer
```

Ví dụ 2:

```
USE Northwind
GO
IF EXISTS (SELECT name
FROM sysobjects
WHERE name = 'GetUnitPrice' AND
type = 'P')
DROP PROCEDURE GetUnitPrice
GO

CREATE PROCEDURE GetUnitPrice @prod_id int, @unit_
price money OUTPUT
AS
SELECT @unit_price = UnitPrice
FROM Products
WHERE ProductID = @prod_id
GO
```

Thực thi: Bạn phải khai báo một biến trước khi gọi thủ tục thực thi

```
DECLARE @price money
EXECUTE GetUnitPrice 77, @unit_price = @price OUTPUT
PRINT CONVERT(varchar(6), @price)
GO
```

Ví dụ 3: Tạo thủ tục InsertRows như sau:

```
USE SalesDB
GO
CREATE TABLE mytable
    ( column1 int, column2 char(10) )
GO
CREATE PROCEDURE InsertRows @start_value int
AS
DECLARE @loop_counter int, @start_val int
SET @start_val = @start_value - 1
SET @loop_counter = 0
WHILE (@loop_counter < 5)
    BEGIN
        INSERT INTO mytable VALUES (@start_val + 1, "new row")
        PRINT (@start_val)
        SET @start_val = @start_val + 1
        SET @loop_counter = @loop_counter + 1
    END
GO
```

Thực thi: Hãy thực thi thủ tục với giá trị khởi tạo là 1

```
EXECUTE InsertRows 1
GO
```

Kiểm tra kết quả

```
SELECT * FROM mytable
```

Ví dụ 4:

--- Tạo thủ tục Count_tables có 2 tham số Output

```
USE Pubs
GO
CREATE PROC count_tables
    @authorcount INT OUTPUT,
    @titlecount INT OUTPUT
```

```

AS
SELECT * FROM authors
SET @authorcount=@@rowcount

SELECT * FROM Titles
SET @titlecount=@@rowcount

---- Thực thi thủ tục Count_tables
DECLARE @a_count INT, @t_count INT
EXECUTE count_tables @a_count OUTPUT, @t_count OUTPUT
SELECT authorcount=@a_count, titlecount=@t_count

```

Ví dụ 5: Viết một thủ tục tính giai thừa của một số nằm trong khoảng 0 đến 12
 Dùng T-SQL để tính giai thừa bằng đệ quy, của các số từ 0 đến 12. Các tham số
 có giá trị lớn hơn 12 sẽ không cho phép bởi vì kết quả trả về sẽ vượt quá phạm vi
 của dữ liệu kiểu int

```

CREATE PROC Giaithua @sol int
AS
DECLARE @SolGiam1 int, @Ketqua int
IF (@sol < 0 OR @sol > 12)
  BEGIN
    -- Giá trị tham số không hợp lệ.
    RETURN -1
  END

IF (@sol=0 or @sol=1)
  SELECT @Ketqua=1
ELSE
  BEGIN
    SET @ SolGiam1=@Sol - 1
    EXEC @Ketqua=Giaithua @ SolGiam1 - Đệ qui gọi
    lại chính nó
    IF (@Ketqua= -1)
      BEGIN
        RETURN -1
      END

    SET @Ketqua=@Ketqua * @sol
    IF (@@ERROR <> 0)
      RETURN -1
  END

```

```

END

RETURN (@Ketqua)

```

Thực thi: Tính giai thừa cho 3

```

DECLARE @kq INT
EXEC @kq = giaithua 3
PRINT @kq

```

Khi thủ tục Giathua đã tồn tại thì chúng ta có thể dùng bó lệnh sau để hiển thị giai thừa của tất cả các số từ 0 đến 12:

```

DECLARE @Ketqua int, @n int
SET @Ketqua =0
SET @n =0
WHILE (@n <= 12) BEGIN
    EXEC @Ketqua = Giaithua @n
    IF (@Ketqua = -1) BEGIN
        RAISERROR('Error executing factorial procedure.', 16, -1)
        RETURN
    END
    PRINT CONVERT(varchar, @n) + '! = ' + CONVERT(varchar(50), @Ketqua)
    SET @n=@n + 1
END

```

11.2 FUNCTIONS.

Hàm thực sự tương tự như Stored procedure của SQL Server, nội dung bao gồm các phát biểu T-SQL kết hợp tạo thành hàm, có thể gọi thực thi các hàm như là một đơn vị độc lập.

Hàm được dùng trong:

- Danh sách chọn của một câu lệnh Select để cho ra một giá trị.
- Một điều kiện tìm kiếm của mệnh đề Where trong các câu lệnh T-SQL

Buil-in functions: Những hàm này hoạt động như là một định nghĩa trong T-SQL và không thể hiệu chỉnh. Nó 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 và scalar (vô hướng).

Nên tìm hiểu và tận dụng tối đa các hàm Buil-in function của SQL Server

User-defined function hay còn gọi là UDFs: 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 UDF, 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)

Scalar Function. Một hàm vô hướng trả về một giá trị đơn và có thể được dùng bất cứ nơi nào biểu thức hay biến có thể được dùng (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 Function. Một hàm có giá trị bảng trả về một tập kết quả và có thể được dùng bất cứ nơi nào bảng hay view được dùng. Hàm giá trị bảng có thể được tham chiếu trong mệnh đề FROM của câu lệnh SELECT. Các hàm người dùng có thể có nhiều phức tạp hơn viêu và có thể có tham số.

Với các ngôn ngữ lập trình khác, nếu tham số không truyền vào thì xem như là hàm sẽ lấy giá trị default của các tham số, nhưng với SQL Server thì phải truyền giá trị Default vào.

11.2.1 Scalar Functions

```
CREATE FUNCTION [ owner_name. ] function_name
  ([ { @parameter_name [AS] scalar_parameter_data_type [= default] } [...n] ])
RETURNS scalar_return_data_type
[ WITH < function_option> [ [,] ...n ] ]
[ AS ]
BEGIN
  function_body
  RETURN scalar_expression
END
```

Ví dụ:

```
CREATE FUNCTION TotalAmount
  (@UnitPrice money, @Quantity smallint, @Discount
  real)
RETURNS money
AS
BEGIN
  RETURN (@UnitPrice*@Quantity)*(1-@discount)
END
```

Sử dụng:

```
SELECT ProductID, Total=dbo.TotalAmount(UnitPrice,
Quantity, Discount)
FROM [Order details]
WHERE OrderID=10250
```

11.2.2 Table-valued Functions

Được chia thành hai loại nhỏ: inline table-value và multistatement table-valued.

Một hàm inline table-valued: Nó có thể được xem như là một View có tham số. Chúng thực thi một câu lệnh Select như trong một view nhưng có thể bao gồm các tham số, giống như thủ tục.


```

CREATE FUNCTION [ owner_name. ] function_name
  ([ { @parameter_name [AS] scalar_parameter_data_type [= default] } [,...n] ])
RETURNS TABLE
[ WITH < function_option > [ [,] ...n ] ]
[ AS ]
RETURN [ ( [ select-stmt ] ) ]

```

Ví dụ 1:

```

CREATE FUNCTION SalesByCategory(@Categoryid Int)
RETURNS TABLE
AS
RETURN
  (SELECT c.CategoryName, P. ProductName,
        SUM(Quantity) AS TotalQty
   FROM Categories c
   INNER JOIN Products p ON c.CategoryID= p.
   CategoryID
   INNER JOIN [Order Details] od ON p.ProductID =
   od.ProductID
   WHERE c.CategoryID= @Categoryid
   GROUP BY c. CategoryName, p.ProductName)

```

Hàm Multistatement Table-valued là dạng phức tạp nhất. Loại hàm này xây dựng tập kết quả từ một hay nhiều câu lệnh SELECT.

```

CREATE FUNCTION [owner_name.]function_name
  ([{ @parameter_name [AS] data_type [=default] } [ ,...n ]])
RETURNS @return_variable
TABLE ({column_definition | table_constraint} [ ,...n ])
[WITH { ENCRYPTION | SCHEMABINDING } [ [,] ...n ] ]
[AS]
BEGIN
function_body
RETURN
END

```

Ví dụ:

```

CREATE FUNCTION Contacts(@suppliers bit=0)
RETURNS @Contacts TABLE (ContactName nvarchar(30),
Phone nvarchar(24), ContactType nvarchar(15))
AS
BEGIN

```

```
INSERT @Contacts
SELECT ContactName, Phone, 'Customer' FROM
Customers
INSERT @Contacts
SELECT FirstName + ' ' + LastName, HomePhone,
'Employee'
FROM Employees
IF @Suppliers=1
INSERT @Contacts
SELECT ContactName, Phone, 'Supplier'
FROM Suppliers
RETURN
END
```

Sử dụng

```
SELECT * FROM CONTACTS(1) ORDER BY ContactName
```

BÀI 12: TRANSACTIONS – LOCK

SQL Server dùng các transaction và các lock để đảm bảo tính vững chắc và toàn vẹn dữ liệu, không chấp nhận các lỗi xảy ra trong hệ thống.

12.1 TRANSACTIONS

Transaction: Một transactin có thể được định nghĩa như là một chuỗi các thao tác thực thi cùng với nhau như là một khối thống nhất đơn của công việc. Một khối thống nhất của công việc phải có bốn đặc điểm được gọi là *ACID* (*Atomicity, Consistency, Isolation, và Durability*).

- **Atomicity:** đặc tính này thể hiện rằng hoặc là tất cả các hiệu chỉnh dữ liệu được thực hiện hoặc là tất cả chúng đều không được thực hiện.
- **Consistency:** đây là một trạng thái mà tất cả các dữ liệu ở trong tình trạng nhất quán sau khi một transaction được hoàn tất một cách thành công. Tất cả các qui tắc (rules) trong một CSDL quan hệ phải được thoả mãn đối với các hiệu chỉnh trong một transaction nhằm duy trì toàn bộ các toàn vẹn dữ liệu.
- **Isolation:** đặc tính này thể hiện rằng bất kỳ sự hiệu chỉnh dữ liệu thực hiện bởi các transaction đồng thời phải độc lập với các hiệu chỉnh khác của các transaction đồng thời khác. Nói một cách đơn giản hơn, một transaction hoặc là truy xuất dữ liệu ở trạng thái mà trước khi transaction đồng thời thực hiện hiệu chỉnh hoặc là truy xuất dữ liệu sau khi transaction thứ hai được hoàn tất.
- **Durability:** Đặc tính này thể hiện rằng bất kỳ thay đổi trong dữ liệu bởi một transaction đã hoàn tất giữ nguyên ảnh hưởng trong hệ thống. Vì vậy, bất kỳ sự thay đổi bởi một transaction hoàn tất vẫn còn thậm chí trong sự kiện hệ thống bị fail. Đặc tính này được đảm bảo bởi sự dự phòng và phục hồi transaction log.

Ba loại Transaction:

- **Implicit Transactions (Transaction ngầm định):** Khi một connection đang mở trong chế độ implicit, SQL Server bắt đầu một transaction mới một cách tự động sau khi transaction hiện hành hoàn tất hoặc Roll back. Bạn không cần phát họa bắt đầu một transaction; bạn chỉ cần commit hoặc Rollback mỗi transaction. Chế độ Implicit transaction phát sinh một chuỗi các transaction liên tục

Sau khi chế độ transactin implicit đã được bật ON cho một kết nối, SQL Server tự động bắt đầu một transaction khi nó thực thi bất kỳ các lệnh sau: Alter Table, Creat, Delete, Drop, Fetch, Grant, Insert, Open, Revoke, Select, Truncate Table, Update.

- **Explicip Transactions: (Transaction tường minh):** Là một transaction mà chúng ta phải định nghĩa bắt đầu một transaction (Begin transaction) và kết thúc một transaction (Commit Transaction)

Bắt đầu một transaction

```
BEGIN TRAN [ SACTION ] [ transaction_name | @tran_name_variable ]
```

Hoàn Tất Transaction

```
COMMIT [ TRAN [ SACTION ] [ transaction_name | @tran_name_variable ] ]
```

Lưu vị trí Transaction

```
SAVE TRAN [ SACTION ] { savepoint_name | @savepoint_variable }
```

Hủy một Transaction

```
ROLLBACK [ TRAN [ SACTION ]
          [ transaction_name | @tran_name_variable
          [ savepoint_name | @savepoint_variable ] ]
```

@@TRANCOUNT: Trả về số thứ tự mà transaction được mở, tối đa lồng 32 cấp, không nên lồng.

- **Distributed Transaction:** là một loại explicit Transaction nhưng giao tác của nó liên quan nhiều server. Sự quản lý phải được kết hợp giữa các nhà quản lý tài nguyên của các server và điều này gọi là transaction manager. Các transaction trong một server những tham chiếu từ nhiều database, thực ra cũng là một distributed transaction.

Transaction log là một tranction dùng lock để ngăn chặn người dùng hiệu chỉnh dữ liệu ảnh hưởng từ các transaction chưa hoàn tất.

Công dụng transaction log

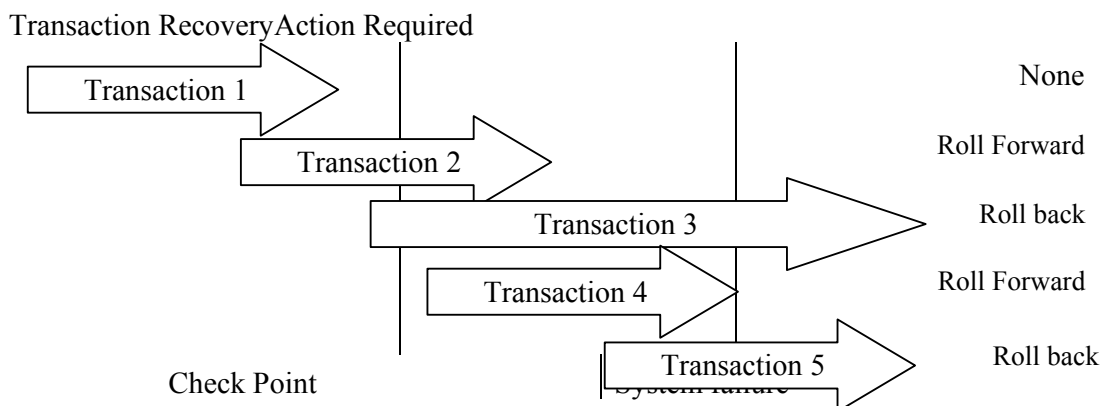
- **Phục hồi các transaction đặc biệt:** Khi một application đưa ra lệnh ROLL BACK hoặc SQL Server nhận ra một lỗi, thì bảng ghi log được dùng để roll back bất kỳ hiệu chỉnh trong suốt quá trình của transaction chưa hoàn tất.
- **Phục hồi tất cả các transaction chưa hoàn tất khi Sql server được bắt đầu:**
- **Hoàn trả lại database lại đến một thời điểm bị lỗi:** Nhằm đảm bảo không phát sinh mâu thuẫn sau khi có sự cố.

Write-ahead Transaction Log: Dùng Write – ahead log đảm bảo rằng không có dữ liệu hiệu chỉnh nào được cất vào đĩa trước khi cất trong log record.

Check point : Là một hành động thực hiện định kỳ trong CSDL, nó sẽ ghi lại tất cả các transaction liên quan đến data lên transaction log, nhằm phục vụ cho việc recovery data.

Check point xảy ra khi:

- Gặp câu lệnh Check point.
- Có sự hiệu chỉnh trên CSDL
- Trước ngay khi SQL Server Shutdown
- Áp định định kỳ.



12.2 LOCK

Lock: là cơ cấu ngăn chặn các xung đột do các user không thể đọc hoặc hiệu chỉnh các dữ liệu mà các dữ liệu này hiện đang trong một tiến trình xử lý khác. Tuy nhiên, bạn vẫn có thể thao tác trên những đối tượng còn phụ thuộc vào chuyển tác mà user khác đang thực hiện. Khi đó hệ thống sẽ kiểm soát tiến trình của bạn có tương thích với quá trình trước đó hay không.

Các vấn đề đồng thời (Concurrency problem)

- **Lost Updates (cập nhật mất dữ liệu):** Lost updates xảy ra khi 2 hoặc nhiều transaction chọn cùng một dữ liệu và sau đó cập nhật dòng dựa trên giá trị cũ. Mỗi transaction không biết những transaction khác. Thao tác cập nhật cuối cùng ghi đè lên những thao tác cập nhật khác mà kết quả dẫn đến mất dữ liệu.
- **Uncommitted Dependency (Dirty Read – đọc dữ liệu sai):** Uncommitted Dependency xảy ra khi transaction thứ 2 chọn một dòng mà đang sẵn sàng cập nhật bởi một transaction. Transaction thứ 2 đang sẵn sàng đọc dữ liệu mà chưa được hoàn tất và có thể bị thay đổi bởi transaction cập nhật
- **Inconsistent Analysis (Nonrepeatable Real – đọc hai lần mẫu tin):** Xảy ra khi transaction thứ hai truy xuất cùng một dữ liệu với vài lần và đọc lên những dữ liệu khác nhau ở mỗi lần đọc. Inconsistent Analysis tương tự với Uncommitted Dependency trong trường hợp transaction thứ nhất đang hiệu chỉnh dữ liệu thì một transaction thứ hai đọc dữ liệu. Tuy nhiên, dữ liệu đọc bởi transaction thứ hai đã được committed bởi transaction update
- **Phantom Reads (đọc các mẫu tin ma):** Xảy ra khi hành động insert hoặc delete được thi hành trên một dòng dữ liệu mà nó thuộc vùng dữ liệu đọc của một transaction khác.

Kiểu locks

- **Share locks:** được dùng cho những thao tác mà không làm thay đổi hay cập nhật dữ liệu (thao tác chỉ đọc), như là một câu select
- **Exclusive locks:** được dùng cho những thao tác hiệu chỉnh dữ liệu, như là Insert, Update, hay Delete. Đảm bảo rằng nhiều cập nhật không thể được thực hiện trong cùng tài nguyên tại cùng một thời điểm.
- **Update locks:** được dùng trên những tài nguyên mà có thể được cập nhật. Ngăn chặn một dạng thông thường của deadlock mà xảy ra khi nhiều session đang đọc, đang lock, và có khả năng cập nhật tài nguyên sau này.
- **Intent locks:** Dùng để thiết lập một lock kế thừa. Kiểu intent lock là : Intent shared (IS), Intent exclusive (IX), và share with intent exclusive (SIX)
- **Schema locks:** được dùng khi thao tác tùy thuộc vào gián đồ của table là đang thực thi. Kiểu schema locks là schema modification (Sch-M), schema stability (Sch-S)
- **Bulk Update (BU) locks:** Cho phép chia sẻ cho Bulk-copy thi hành.

Deadlock

Deadlock xảy ra khi có một sự phụ thuộc chu trình giữa hai hay nhiều luồng cho một tập hợp tài nguyên nguyên nào đó. SQL Server sẽ tự giải quyết trường hợp deadlock bằng cách RollBack một trong các transaction, và ưu tiên rollback những transaction có thời gian ít hơn. Để giảm bớt deadlock, bạn nên:

- ✓ Truy xuất các object theo thứ tự.
- ✓ Tránh sự tương tác người dùng trong thời gian transaction.
- ✓ Cố giữ transaction càng ngắn càng tốt.
- ✓ Dùng mức cô lập thấp nhất.
- ✓ Dùng giới hạn các connection.

BÀI 13:

SỬ DỤNG CURSORS ĐỂ TRUY XUẤT DỮ LIỆU

13.1 Khái niệm

Cursor là một đối tượng của CSDL mà nó hỗ trợ cho phép truy xuất và thao tác dữ liệu trong một tập kết quả (result set). Sau khi cursor được định vị trên một dòng, các hoạt động có thể thực hiện trên dòng đó hoặc khối các dòng bắt đầu từ vị trí đó.

Dùng Cursor để:

- Định vị một dòng đặc biệt trong tập kết quả.
- Truy xuất một hoặc khối dòng bắt đầu từ vị trí cursor trong tập kết quả.
- Cung cấp thao tác hiệu chỉnh dữ liệu cho các dòng tại vị trí của cursor trong tập kết quả.
- Cung cấp các mức độ khác nhau của tính tường minh trong dự thay đổi được tạo bởi những người dùng khác đến tập kết quả.
- Cung cấp việc truy cập dữ liệu trong tập kết quả cho các câu lệnh T-SQL trong scripts, Stored procedure, và triggers.

Các thao tác cần thực hiện trong khi sử dụng cursor trong SQL Server:

- Cursor cần phải khai báo và các thuộc tính của nó cũng cần được xác định.
- Mở cursor.
- Phải lấy (fetch) các dòng cần thiết từ cursor.
- Dữ liệu trong dòng hiện hành có thể được hiệu chỉnh nếu cần thiết.
- Tạm thời không dùng cursor thì phải đóng cursor lại.
- Cursor cần phải được giải phóng (deallocate) khi không cần dùng nữa.

Cursor:

- **T-SQL Server cursors:** Cursor này được dựa trên câu lệnh khai báo cursor, nó được dùng chủ yếu trong các script, store procedure, triggers. Nó được thi hành trên server và được quản lý bởi các câu lệnh TOSQL gửi từ client đến server. Khi làm việc với cursor thì phải khai báo, mở, truy xuất, xử lý, đóng, giải phóng.
- **API Server cursors (API-Application Program interface):** Nó được thực thi trên server và được quản lý bởi một hàm cursor API. API Server cursors được cung cấp bởi hàm cursor API trong OLE DB, ODBC, và DB-Library. Mỗi lần một ứng dụng của client gọi một hàm cursor API, SQL server OLE DB provider, ODBC driver, hoặc DB-Library DLL gửi các yêu cầu đến server cho hành động ứng với các hàm cursor API. Nó chỉ khác với T-SQL cursor ở syntax, còn về bản chất tương tự.
- **Client Cursors:** SQL server ODBC driver, DB-Library DLL, và ADO API DLL giúp thi hành cursor client một cách nội tại. cursor client được thi hành bằng cách nắm giữ tập kết quả dữ liệu của clients. Mỗi lần Application của client gọi hàm cursor API, thì SQL server OLE DB provider, ODBC driver, hoặc DB-Library DLL thực thi tính toán ngay trên tập kết quả dữ liệu được giữ trên client. Ta chỉ dùng client cursor để làm giảm bớt sự giới hạn mà server consor không được cung cấp các câu lệnh T-SQL. Nếu con trỏ static hoặc Scroll thì ta có thể dùng client cursor.

Loại cursor**Static**

- Tập kết quả của control loại Static được xây dựng trong tempdb khi con trỏ được mở.
- Một static con trỏ luôn luôn hiện tập kết quả giống như tập kết quả có được ngay sau khi con trỏ mở.
- Con trỏ không phản ánh được bất kỳ sự thay đổi nào trong database ngay cả khi những dòng dữ liệu có thay đổi, các dòng mới được insert bởi các transaction khác cũng vẫn không hiện lên mặc dù chúng thỏa điều kiện lọc dữ liệu.
- Thao tác Insert, Update, Delete đều không có tác dụng khi dùng static cursor

Keyset-driven:

- Thành viên và thứ tự của các dòng trong một keyset-driven cursor là cố định khi cursor được mở. Con trỏ được điều khiển bởi một tập giá trị nhận dạng gọi là Keyset. Keyset được xây dựng từ một tập các cột mà dùng để nhận dạng các dòng trong tập kết quả. Keyset được xây dựng trong Tempdb khi con trỏ được mở
- Cho phép hiệu chỉnh (update, delete) dữ liệu trên cột không là keyset (bởi chủ cursor hoặc từ user khác) khi user duyệt thông qua con trỏ.
- Có thể thêm (insert) vào bảng nếu như cursor thể chèn dữ liệu vào bảng.

Dynamic: Trái ngược với static cursor.

- Dynamic cursors phản ánh được toàn bộ sự thay đổi của các dòng dữ liệu trong tập kết quả khi duyệt con trỏ.
- Giá trị dữ liệu, thứ tự, và thành viên của các dòng trong tập kết quả có thể thay đổi ứng với mỗi lần duyệt con trỏ.
- Tất cả các lệnh Insert, Update, Delete của các user đều hữu hiệu thông qua con trỏ.
- Sự Update hữu hiệu ngay tức thời nếu chúng được update thông qua qua con trỏ ứng với tại mẫu tin hiện thời, còn nếu update bên ngoài con trỏ thì nó không hữu hiệu cho đến khi nó hoàn tất.

Fast Forward only: Tương tự như Dynamic cursor nhưng nó chỉ có thể duyệt con trỏ thào một chiều từ First đến Last

Cursor type	Membership	Order	Values
Forward-only	Dynamic	Dynamic	Dynamic
Static	Fixed	Fixed	Fixed
Dynamic	Dynamic	Dynamic	Dynamic
Keyset-driven	Fixed	Fixed	Dynamic

13.2 Làm việc với T-SQL server cursors**Khai báo cursor**

Bằng câu lệnh declare ở sau từ khóa AS trong stored procedures hoặc functions

```
DECLARE cursor_name CURSOR
[ LOCAL | GLOBAL ]
[ FORWARD_ONLY | SCROLL ]
```



```
[ STATIC | KEYSSET | DYNAMIC | FAST_FORWARD ]
[ READ_ONLY | SCROLL_LOCKS | OPTIMISTIC ]
[ TYPE_WARNING ]
FOR select_statement
[ FOR UPDATE [ OF column_name [,...n] ] ]
```

- *select_statement*: là câu lệnh truy vấn để định nghĩa tập kết quả của cursor. Từ khóa COMPUTE, COMPUTE BY, FOR BROWSE, and INTO Không cho phép trong *select_statement* này.
- READ ONLY Không cho phép Update trong cursor này.
- UPDATE [OF *column_name* [,...n]]: Quy định cột cho phép được update khi dùng cursor. Nếu OF *column_name* [,...n] được chỉ định rõ ràng thì chỉ có các cột được chỉ định mới được cho phép hiệu chỉnh, nếu không có column list, all columns có thể update.

Mở cursor

Mở cursor trước khi dùng (Cursor phải được khai báo rồi)

```
OPEN {cursor_name }
```

Lấy mẫu tin hoặc điều hướng cursor (FETCH)

Truy xuất từng dòng dữ liệu. Kiểm tra phạm vi con trỏ bằng @@Fetch_status

```
FETCH [[NEXT | PRIOR | FIRST | LAST | ABSOLUTE n | RELATIVE n]]
FROM cursor_name [ INTO @variable_name [,...n] ]
```

@@Fetch_status: Trả về giá trị 0 hoặc 1. Trả về 1 có nghĩa là con trỏ đã được dời đến quá cuối tập kết quả. Trả về 0 vẫn còn trong phạm vi của tập kết quả. Khi đó dùng vòng lặp While để duyệt cả tập kết quả của Cursor.

Xử lý dữ liệu:

Có thể dùng các câu lệnh Update hoặc Delete để hiệu chỉnh dữ liệu

- Sử dụng dữ liệu của mẫu tin hiện hành: dữ liệu được lấy lên và gán cho các biến tương ứng trong câu lệnh Fetch
- Cập nhật dữ liệu thông qua cursor: thực chất là dữ liệu được hiệu chỉnh trực tiếp vào trong bảng

Cập nhật giá trị cho cột

```
UPDATE <Ten Table>
SET <TenColumn>= <Value> [,...n]
WHERE CURRENT OF <Cur_Name>
```

Xoá dữ liệu thông qua cursor: Thực sự là dữ liệu xóa trên bảng

```
DELETE <Ten Table>
WHERE CURRENT OF <Cur_Name>
```

Đóng Cursor

Kết thúc hành động của cursor cho lần mở (open), nó vẫn hiện hữu cho đến khi gặp một lệnh Open khác hoặc gặp lệnh Close cursor.

```
CLOSE cursor_name
```

Giải phóng Cursor

Giải phóng cursor, hủy bỏ tham chiếu đến con trỏ từ session hiện hành. Tiến trình này làm cho tài nguyên trở về trạng thái sẵn sàng truy xuất.

```
DEALLOCATE cursor_name
```

13.3 Ví dụ.

Giả sử người quản lý cần một bảng báo cáo lịch sử khách hàng theo dạng sau:

Customer:ALFKI - Alfreds Futterkiste

Order:10643 (Aug 25 1997)

Order:10692 (Oct 3 1999)

Order:10702 (Oct 13 1999)

Customer:ANATR - Ana Trujillo Emparedados y helados

Order:10625 (Aug 8 1997)

Order:10759 (Nov 28 1999)

Customer:ANTON - Antonio Moreno Taquería

Order:10507 (Apr 15 1997)

Order:10535 (May 13 1999)

Order:10573 (Jun 19 1999)

Order:10677 (Sep 22 1999)

Order:10682 (Sep 25 1999)

Đoạn Batch thực hiện báo cáo như sau:

```
--- Khai báo contrỏ
DECLARE rpt CURSOR FOR
SELECT c.CustomerID, c.CompanyName, o.OrderID,
o.OrderDate
FROM Customers c, Orders o
WHERE c.CustomerID = o.CustomerID AND c.CustomerID LIKE
'A%' AND DatePart( year, o.OrderDate) = 1997

DECLARE @cid char( 8), @cname char( 40),
@ordid char( 8), @orddt datetime, @old char( 8)
--- Mở contrỏ
OPEN rpt
--- Lấy dữ liệu của mẫu tin đầu tiên vào các biến
FETCH NEXT FROM rpt INTO @cid, @cname, @ordid, @orddt
SELECT @old = ' '
```

```

WHILE @@fetch_status = 0
BEGIN
    IF @old = @cid
        BEGIN
            PRINT ' Order:' + rtrim( @ordid) + ' (' +
                cast(@orddt as CHAR( 10)) + ')'
        END
    ELSE
        BEGIN
            PRINT 'Customer:' + rtrim( @cid) + ' - ' +
                rtrim(@cname)
            PRINT ' Order:' + rtrim( @ordid) + ' (' +
                cast(@orddt as CHAR( 11)) + ')'
            SELECT @old = @cid
        END
    FETCH NEXT FROM rpt INTO @cid, @cname, @ordid,
    @orddt
END
--- Đóng con trỏ
CLOSE rpt
--- Giải phóng con trỏ
DEALLOCATE rpt

```

Ví dụ 2:

```

DECLARE MyCursor CURSOR FOR
SELECT c.CustomerID, c.Companyname, c.contactname,
       o.OrderID, o.OrderDate
FROM Customers c, Orders o WHERE c.CustomerID =
o.CustomerID
FOR UPDATE
OPEN MyCursor
DECLARE @cid VARCHAR( 8), @c VARCHAR( 80), @o INT,
        @od DATETIME, @cn VARCHAR( 80)
FETCH NEXT FROM MyCursor INTO @cid, @c, @cn, @o, @od
SELECT @cid
BEGIN TRANSACTION
UPDATE Customers SET CompanyName = 'q'
WHERE CURRENT OF Mycursor
DEALLOCATE MyCursor
SELECT * FROM Customers
ROLLBACK TRANSACTION

```

BÀI 14: BẦY LỖI - TRIGGER

14.1 Giới thiệu về trigger

Chất lượng của một CSDL được đánh giá một phần bởi tính nhất quán và độ chính xác của dữ liệu trong CSDL. Để đảm bảo tính toàn vẹn dữ liệu ta có nhiều phương pháp, trigger là một phương pháp hữu hiệu.

- Trigger là một loại stored procedure đặc biệt, nó được định nghĩa để tự động thực thi khi có một câu lệnh Update, Insert, hoặc Delete được phát ra trên bảng hoặc View.
- Trigger là một công cụ mạnh mà nó có thể dùng để ràng buộc các qui tắc quản lý một các tự động khi dữ liệu bị hiệu chỉnh.
- Trigger cũng có thể nói rộng các tính toàn vẹn kiểm soát logic của SQL Server.
- Trigger tự động thực thi, không thể gọi một trigger thi hành một cách trực tiếp.

Dùng trigger khi:

- Ràng buộc toàn vẹn dữ liệu cho phù hợp với mô hình quan hệ CSDL.
- Kiểm soát dữ liệu hiện tại khi có thay đổi đến giá trị trong mẫu tin trong bảng.
- Kiểm tra dữ liệu nhập vào phù hợp với mối quan hệ dữ liệu giữa các bảng.
- Định nghĩa thông báo lỗi của người dùng.
- So sánh trạng thái của dữ liệu trước và sau hiệu chỉnh

Đặc điểm và giới hạn của trigger

- Trigger (After trigger) là Reactive; constraints và instead of trigger là proactive. (Reactive: khi delete/insert một dòng vào table, thì sau khi insert thì trigger mới được tự động thực thi thì gọi là reactive, proactive là kiểm tra trước khi Insert/Delete)
- Các constraint được kiểm tra trước, sau đó mới tới trigger.
- Các bảng có thể có nhiều trigger cho bất kỳ hành động nào. Tuy nhiên không nên dùng quá nhiều trigger trong cùng một bảng. SQL Server chỉ cho phép chỉ định trigger nào thi hành đầu tiên, thi hành cuối cùng, còn các trigger khác thứ tự thi hành không xác định. Vì vậy, nếu có quá nhiều trigger trên 1 đối tượng có thể sẽ gặp nhiều rắc rối khi có nhiều trigger không xác định thứ tự.
- Không thể tạo trigger trên các đối tượng ở temporary.
- Nên thiết kế trigger không trả về tập kết quả nhằm đảm bảo tính chất chuyên tác giữa các user và lập trình.
- Các trigger có thể xử lý hành động trên nhiều dòng.
- Trigger không ngăn ngừa thay đổi cấu trúc, trigger chỉ quan tâm đến sự thay đổi dữ liệu trong bảng. Khi bạn xóa đối tượng trong CSDL với các bước hợp lý, SQL Server sẽ cho phép xóa đối tượng đó và trigger không thể kiểm soát được.
- **Trigger Events:** Có 3 biến cố mà trigger sẽ tự động thực thi khi biến cố xảy ra, đó là **Insert, Update, Delete**. Trigger không thể được gọi một cách trực tiếp.

Cơ cấu thực thi trigger:

Khi insert hoặc update dữ liệu của bảng bật trigger, trigger sẽ lưu trữ dòng dữ liệu mới hoặc dòng dữ liệu đã hiệu chỉnh vào một bảng có tên là **Inserted** trong bộ

nhớ cash, khi xóa dữ liệu của bảng bật trigger lên, trigger sẽ lưu trữ dòng dữ liệu bị xóa vào bảng có tên là **Deleted** trong bộ nhớ cash. Các bảng tồn tại trong bộ nhớ và được truy vấn bởi các lệnh T-SQL trong các trigger. Bạn có thể sử dụng thông tin trong bảng inserted và Deleted để so sánh, lưu trữ, rollback,... nếu cần, khi đó bạn không cần tạo ra các biến để lưu trữ thông tin và tốc độ truy xuất nhanh.

Hai loại trigger trong SQL Server 2000: INSTEAD OF và AFTER (nếu chỉ nói trigger có nghĩa là nói đến AFTER Trigger)

- **FOR triggers và AFTER triggers:** các trigger này chỉ được thực thi khi tất cả các thao tác Insert, Update hay Delete thực hiện xong. Tất cả các hành động tham chiếu và kiểm tra constraint cũng phải được thực hiện xong trước khi trigger thi hành. Loại trigger này chỉ cài đặt được trên bảng, không cài đặt được trên View.

Khi tạo trigger và không chỉ định rõ thì mặc định là AFTER, FOR chỉ là từ khóa tương thích ngược với các phiên bản trước của SQL Server.

- **INSTEAD OF triggers:** Trigger này chỉ có trong SQL Server 2000. Trigger này sẽ thi hành thay cho các câu lệnh Insert, Delete, Update. Như vậy khi tạo trigger kiểu này bạn phải viết lại các lệnh insert, Delete, Update đối với dữ liệu. Có thể áp dụng cho cả bảng và View, tuy nhiên nó không cho phép áp dụng với các view có lựa chọn WITH CHECK OPTION.

Nested trigger: có nghĩa là bảng Table1 có trigger, Table2 có trigger khác. Nếu ta thao tác trên Table1 thì trigger của nó sẽ thực thi, nếu thao tác này có liên quan đến Table2 thì trigger2 ở bảng Table2 thực thi. Gọi là lồng các trigger, bạn có thể lồng tối đa là 32 cấp.

14.2 Tạo và quản lý các trigger

Một trigger có thể tạo và quản lý bằng cách sử dụng Query Analyzer hoặc Enterprise Manager. Tạo một trigger trên đối tượng thì phải có quyền Owner đối với đối tượng.

14.2.1 Tạo trigger

```
CREATE TRIGGER trigger_name
ON { table | view }
[ WITH ENCRYPTION ]
{
  { { FOR | AFTER | INSTEAD OF } { [DELETE] [,] [ INSERT ] [,] [ UPDATE ] }
  [ WITH APPEND ]
  [ NOT FOR REPLICATION ]
  AS
  sql_statement [ ...n ]
}
}
```

Giải thích:

- *trigger_name* : Tên của trigger. Nếu trong 1 bảng có nhiều trigger thì tên của các trigger phải là duy nhất.
- ON { *table* | *view* } : Chỉ định table/View được áp dụng trigger, chỉ có instead of được áp dụng cho cả view và table.
- [WITH ENCRYPTION] : Trigger được mã hóa. Thông tin mã hóa nằm trong bảng syscomment
- { FOR | AFTER | INSTEAD OF } : Xác định loại trigger cho thao tác DELETE, INSERT, UPDATE.

Delete trigger: trigger sẽ được thực thi khi có mẫu tin bị xóa khỏi bảng, SQL Server tạo ra bảng mang tên DELETED để cất mẫu tin bị xóa, trong trigger ta có thể tham khảo đến mẫu tin này.

Insert trigger: trigger sẽ được thực thi khi có mẫu tin chèn vào bảng, SQL server tạo ra bảng mang tên INSERTED để cất mẫu tin chèn, trong trigger ta có thể tham khảo đến mẫu tin này.

Update trigger: Mỗi khi có mẫu tin nào đó được cập nhật, giá trị những cột có liên quan đến trigger sẽ được kiểm tra trước khi cập nhật. Mẫu tin bị cập nhật sẽ được sao lưu trong bảng Inserted (chứa giá trị mới) và Deleted (chứa giá trị cũ).

- [NOT FOR REPLICATION]: Trigger sẽ không thực hiện khi bảng có liên quan đến kỹ thuật sao chép nhân bản (relication)

Lưu ý:

- Một hành động (Insert hoặc Delete hoặc Update) có thể kích hoạt cùng lúc nhiều trigger khác nhau.
- Trigger sẽ thi hành cho dù thao tác của người sử dụng có tác động thực sự trên các mẫu tin hay không, do đó dùng @@ROWCOUNT để kiểm tra trước khi cho các hành động trong trigger thi hành.
- Các phát biểu sau không thể có trong trigger: các phát biểu Create, Drop, Alter Table, Alter DataBase, Truncate Table, Grant/Revoke, Reconfigure, Load DataBase, Transaction, Update statistics, Select Into, Disk.

14.2.2 Quản lý trigger

Alter Trigger² Hiệu chỉnh trigger

Drop Trigger : Xóa trigger

Sp_rename : đổi tên.

Sp_helptrigger, Sp_helptext: Xem code trigger

DISABLE TRIGGER/ ENABLE TRIGGER trong câu lệnh Alter Table

14.3 Vài ví dụ về trigger.

Ví dụ 1: Viết một trigger cho thao tác Insert, điểm kiểm tra ngày lập hoá đơn thì luôn lớn hơn ngày giao.

```
CREATE TRIGGER Trg_NgayLap_NgayGiaoHD
ON Hoadon AFTER INSERT
AS
DECLARE @NgayLapHD DateTime, @NgayGiao DateTime
SELECT @NgayLapHD=hd.NgayLapHD,NgayGiao=hd.NgayGiaoNhan
FROM HoaDon hd INNER JOIN Inserted i ON hd.MaHD=i.MaHD
If @NgayGiao<@NgayLapHD
BEGIN
        RAISERROR(500103,10,1)
```

² Tham thảo cú pháp lệnh trong Books-Online

```

ROLLBACK TRANSACTION
END
-----

```

```

INSERT HoaDon VALUES (1003, '1/1/2004', 'N', 'TP.
HCM', 111, '12/24/2003')

```

Ví dụ 2: Tạo một trigger có tên là Msg_InstUpd_Kh, trigger này thực hiện chi 2 thao tác Insert, Update của bảng KháchHang. Trigger sẽ thông báo “Có n dòng đã được hiệu chỉnh”.

```

Use SalesDB
GO
CREATE TRIGGER Msg_InstUpd_Kh
On KháchHang
FOR INSERT, UPDATE
AS
RAISEERROR('“Có %d dòng đã được hiệu chỉnh”',
0, 1, @@Rowcount)
RETURN

```

Kiểm chứng: bằng cách chèn hoặc cập nhật 1 hoặc nhiều mẫu tin vào bảng KháchHang.

```

INSERT INTO KháchHang ....

```

Ví dụ 3: Hai bảng HoaDon và CT_HoaDon có quan hệ 1-n, tức là khi thêm một chi tiết hóa đơn trong bảng chi tiết hoá đơn thì hóa đơn này phải đã được phát sinh trong bảng hóa đơn. Vì thế ta phải viết một trigger Insert cho bảng CT_HoaDon. Nếu ta có trigger này thì khi ta vi phạm SQL Server cũng sẽ báo lỗi vì vi phạm ràng buộc toàn vẹn khóa ngoại. Tuy nhiên ta hãy viết một trigger để thực hiện thông báo khi có lỗi này xuất hiện.

```

Use SalesDB
GO

CREATE TRIGGER Trigger_Ins_CT_HD
ON HoaDon
FOR INSERT
/* Insert trigger cho bảng hóa đơn*/
AS
IF NOT EXISTS
(select * From Inserted I inner join HoaDon hd ON
i.Mahd = hd.Mahd)
/* Nếu Mahd được chèn vào bảng CT_HoaDon không tồn tại
trong bảng hóa đơn thì không chèn được */
BEGIN

```

```
RAISERROR(60000,16,1,'MaHd', 'CT_HoaDon', 'Mahd', 'Hoa
Don')
        ROLLBACK
END
```

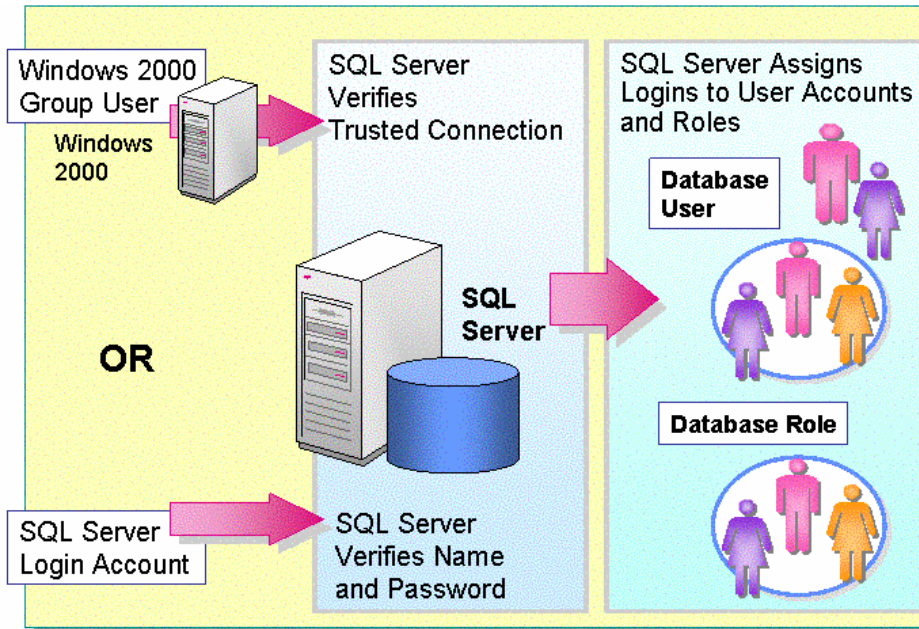
Ví dụ 4: Price trong CT_HoaDon luôn luôn lớn hơn hay bằng GiaGoc trong bảng sản phẩm. Để bắt ràng buộc này bạn thường dùng Check Constraints, tuy nhiên bạn cũng có thể bẫy lỗi bằng trigger

```
Create Trigger MGT_Gia
ON CT_HoaDon
FOR UPDATE
AS
IF EXISTS (Select * From INSERTED I where i.price <
tblItem.OriginalPrice)
        Begin
        RAISERROR('Khong thể cập nhật vì giá không hợp
lệ',16,1)
        ROLLBACK TRAN
        End
```


BÀI 15: BẢO MẬT TRONG SQL SERVER

15.1 Khái niệm về bảo mật.

15.1.1 Mô hình truy cập bảo mật của SQL Server.



Hình 37: Các lớp kiểm tra bảo mật chứng thực của SQL Server

Việc kết nối đến SQL Server 2000 thật đơn giản. Sự bảo mật được kiểm tra ở ba nơi khác nhau: có thể bị kiểm hợp lệ bởi Windows 2000, bản thân SQL Server, mức CSDL riêng lẻ. Ngay sau khi bạn kết nối vào SQL Server bạn chưa thật sự truy cập được bất kỳ một đối tượng CSDL nào, bạn cần phải được cấp quyền (permissions) truy cập đến đối tượng.

15.1.2 Các chế độ bảo mật.

SQL Server 2000 cung cấp hai chế độ bảo mật:

Hai loại chứng thực

Windows Authentication

SQL Server kiểm tra nhận dạng của user và sau đó cho phép hay từ chối đăng nhập truy xuất dựa trên cơ sở tên của User mà không cần tên đăng nhập và password riêng biệt. Điều này gọi là kết nối tin tưởng. Khi bạn kết nối đến SQL Server theo cách này thì có nghĩa là bạn trình bày với SQL Server một ủy nhiệm bảo mật của Windows (như là một thẻ bài truy cập của bạn). Bạn xây dựng các ủy nhiệm này trong quá trình đăng nhập vào mạng windows 2000. Các ủy nhiệm bảo mật này được truyền âm thầm cho bạn, vì thế bạn không cần làm bất cứ điều gì đặc biệt để vượt qua việc kiểm tra bảo mật.

SQL Server Authentication

Người quản trị CSDL có thể tạo ra các tài khoản và password đăng nhập SQL Server. Các tài khoản này hoàn toàn không tùy thuộc vào các tài khoản hay nhóm người dùng hệ điều hành. Nếu có một kết nối chỉ định chứng thực SQL Server thì

SQL Server 2000 thực thi chứng thực chính nó bằng cách kiểm tra xem tài khoản đang nhập có tồn tại hay không và mật khẩu chỉ định có khớp với một ghi nhận trước đây trong SQL Server 2000 không.

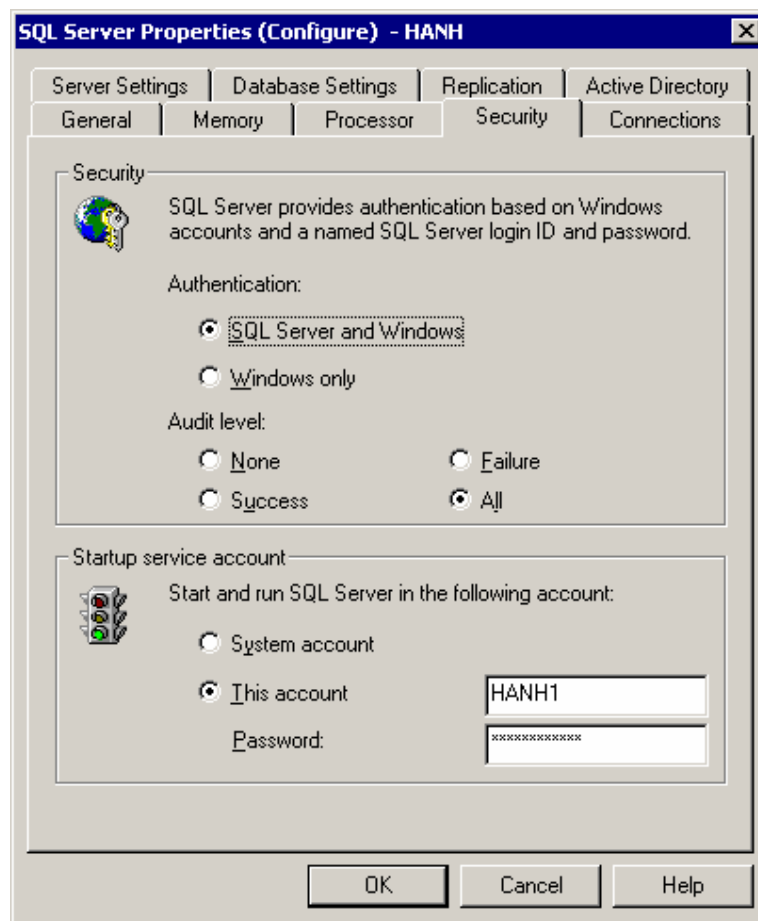
Chế độ chứng thực:

Windows Authentication mode: Người sử dụng chỉ có thể kết nối với SQL Server 200 bằng Windows Authentication (Kết nối tin tưởng)

Mixed mode: Người dùng có thể kết nối với SQL Server 200 bằng cách dùng cả Windows Authentication và SQL Server Authentication

Chuyển đổi chế độ chứng thực:

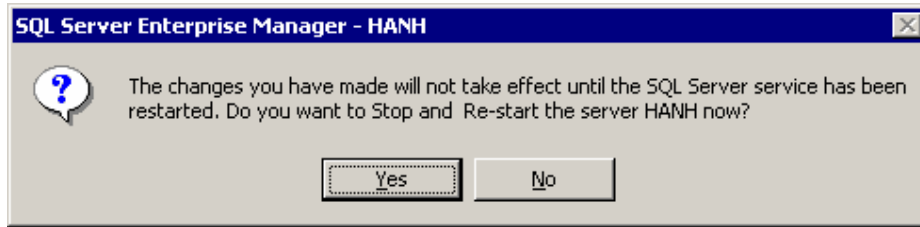
Sau khi cài đặt, bạn có thể sử dụng Enterprise Manager để chuyển đổi qua lại giữa các chế độ. Tại cửa sổ Enterprise Manager, nhấp nút phải chuột tại instance và chọn Properties



Hình 38: cấu hình Security cho SQL Server

Ở trang Security click chọn hoặc SQL Server And Windows hoặc Windows Only để đổi chế độ chứng thực.

Sau khi bạn chuyển chế độ, bạn phải stop và sau đó restart dịch vụ của SQL Server service để sự thay đổi này có tác dụng. Enterprise Manager sẽ hỏi ý kiến bạn.



Hình 39: Hộp thoại hỏi ý kiến người dùng có muốn Stop và Restart Server

15.1.3 Tìm hiểu các Server-Wide Permission.

SQL Server 2000 hỗ trợ một số server role được định nghĩa trước được kết hợp với các quyền quản trị. Các server role này cấp các quyền server-wide để thực hiện các tác vụ khác nhau và bao gồm các quyền mà bạn có thể cấp cho những người dùng thông qua việc sử dụng các server role này. Bạn không thể xoá các server role và không thể thay đổi các quyền của chúng. Để cấp các quyền cho một người dùng, thì bạn thêm đăng nhập của chúng server role. Với Transact-SQL, bạn có thể thêm các người dùng hay nhóm các người dùng vào một server role

Lưu ý: Các Server role về cơ bản giống như các group trong Windows 2000. Trong SQL Server 2000 có 8 server role

Server Role	Thành viên của Server Role có thể ...
sysadmin	Thực thi bất cứ thao tác nào trong một thể hiện SQL Server 2000 và trong bất kỳ CSDL. Mặc nhiên, tất cả các thành viên của nhóm Windows built-in Administrators, tài khoản người dùng sa thuộc vào server role này.
serveradmin	Cấu hình SQL Server 2000 bằng cách dùng thủ tục hệ thống sp_configure và có thể kết thúc các service. Các thành viên của nhóm điều hành viên built-in của Windows là rất tốt để nhận server role này.
setupadmin	Cài đặt và cấu hình linked server, remote server, và replication. Có thể chỉ định một stored procedure được thực thi lúc khởi động (startup), như là sp_serveroption. Các thành viên của nhóm điều hành viên built-in của windows là rất tốt nhận server role này.
securityadmin	Thực hiện tất cả các thao tác liên quan đến security trong SQL Server 2000, kể cả quản lý các quyền câu lệnh CREATE DATABASE, điều khiển server logins, và đọc error log. Giúp đỡ ở các nhân viên văn phòng. Thành viên của nhóm điều hành viên built-in của windows là rất tốt nhận server role này.
processadmin	Quản lý các tiến trình chạy các instance của SQL Server. Có thể ngắt (kill) tiến trình của các user, các truy vấn.
dbcreator	Có thể tạo, hiệu chỉnh, và xóa các CSDL. Những nhà quản trị CSDL lâu năm đảm trách server role này tốt.
Diskadmin	Có thể quản trị các tập và các thiết bị dự phòng. Nói chung Role này dùng để tương thích ngược với SQL Server 6.x.
bulkadmin	Có thể thực hiện các câu lệnh BULK INSERT. Cho phép các thành viên của sysadmin server role làm đại diện các tác vụ BULK INSERT mà không

Server Role	Thành viên của Server Role có thể ...
	cần gán các quyền sysadmin. Hãy cẩn thận bởi vì các thành viên cũng phải truy xuất đọc đến bất kỳ dữ liệu được chèn và quyền INSERT trên bất kỳ bảng mà dữ liệu sẽ được chèn.

Lưu ý: Một thành viên của bất kỳ server role nào đều có thể thêm các user khác có server role đó.

15.1.4 Tìm hiểu các quyền (Permission) chỉ định trên cơ sở dữ liệu.

Khi Truy xuất đến SQL Server 2000 thì bạn chưa có quyền truy xuất đến các CSDL. Ngoại trừ các thành viên trong sysadmin role, thành viên trong một server role có sẵn quyền truy xuất CSDL. Các quyền truy xuất CSDL phải được cấp một cách rõ ràng bởi một system administrator hoặc thành viên của administrator role trong CSDL. Các quyền có thể được cấp (grant), từ chối (deny), cời bỏ (revoke) và bao gồm các quyền tạo đối tượng, quản trị CSDL, thực thi các câu lệnh T-SQL, chèn dữ liệu vào bảng, xem dữ liệu bằng view. SQL Server 2000 có một số cơ chế để cấp các quyền cụ thể cho các user trong một CSDL.

Các quyền cụ thể trên CSDL

Quyền	Mô tả
Database owner	User có thể được chỉ định như là chủ (owner) của CSDL và có thể thực hiện bất kỳ hành động liên quan đến CSDL.
DBO role	Tất cả các thành viên của sysadmin server role thì tự động là thành viên của dbo role trong mỗi CSDL, và có thể thực hiện bất kỳ thao tác liên quan đến CSDL.
User	Các user và group có thể được cấp user truy xuất đến CSDL theo tài khoản bảo mật của Windows 2000 hoặc SQL Server 2000. Sau đó một giấy phép người dùng CSDL được cấp các quyền trong CSDL thông qua database role, role chung, và chỉ định cấp các quyền câu lệnh và đối tượng.
Guest user	Một user mà có thể truy xuất đến 1 instance của SQL Server 2000 (nhưng người này không có tài khoản truy người dùng để truy xuất đến CSDL cụ thể) có thể được cho phép truy xuất đến CSDL như là một người khách (guest user). Tài khoản guest có thể được cấp các quyền cụ thể trong CSDL (đọc dữ liệu). Theo mặc nhiên, một CSDL không có tài khoản guest user.
Public role	Tất cả các user được phép truy xuất đến CSDL trở thành thành viên của public role trong mỗi CSDL. Public role có thể được cấp các quyền cụ thể (Tổng quát các quyền cần thiết cho tất cả các user của CSDL).
Fixed database role	Cho phép các user có thể được thêm vào các fixed database role trong một CSDL. Các Fixed database role chứa các quyền định trước trong CSDL để thi hành các hoạt động của database-wide.
User-defined database role	Cho phép các user có thể thêm vào user-defined database role trong một CSDL. Các role này có thể được tạo bởi administrator và cấp một cách cụ thể các quyền đưa ra hoặc các quyền trong CSDL.

Quyền	Mô tả
Statement permissions	Quyền thực thi các câu lệnh quản trị (như CREATE PROCEDURE) có thể được cấp, huỷ bỏ, từ chối các users, groups, và roles.
Object permissions	Quyền truy xuất đến các đối tượng CSDL (như là bảng hay view) có thể được cấp, huỷ bỏ, từ chối các users, groups, và roles.
Application role	Quyền thực thi các hành động trong một CSDL có thể được cấp cho một application, thậm chí cấp cho user. Một application kết nối đến một CSDL và kích hoạt application role. Các User truy xuất đến một CSDL thông qua sự kết nối này để dành lấy các quyền kết hợp với application role trong suốt quá trình kết nối. Các quyền phát hành đến một user cụ thể thì không liên quan khi user đang truy xuất CSDL thông qua application role.

15.1.5 Fixed Database Roles.

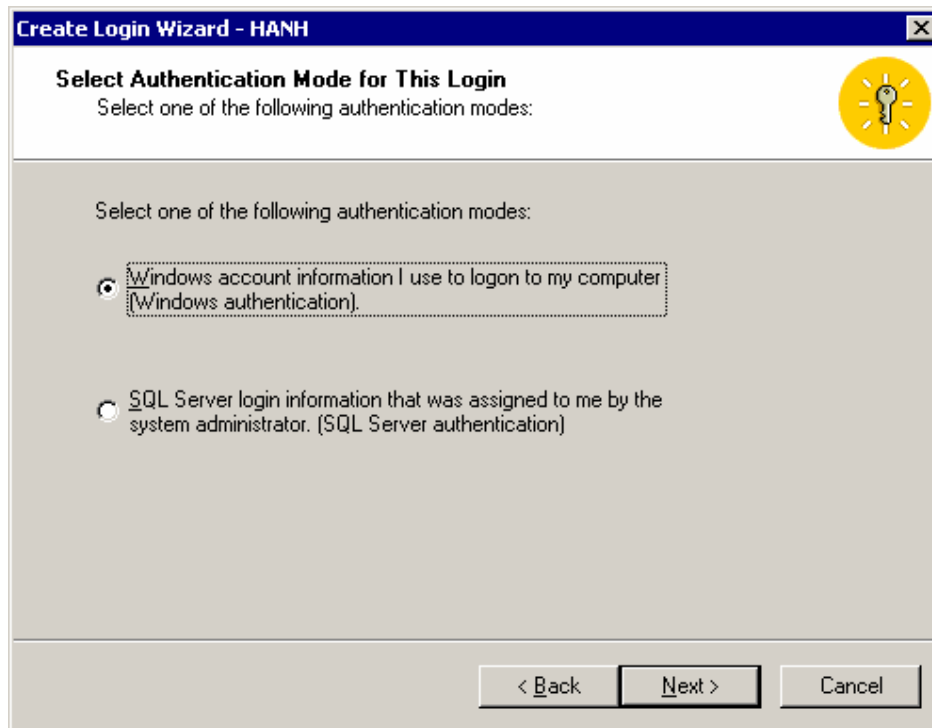
Mỗi CSDL có 9 role về CSDL được định nghĩa trước với các quyền kết hợp với database-wide để thực hiện các tác vụ khác nhau. Bạn không thể xóa những database role này và cũng không thể thay đổi các quyền của chúng. Để gán một người dùng các quyền này trong một CSDL, bạn thêm tài khoản của người dùng vào database role. Nếu những fixed database role này không gán tổng hợp các quyền mà bạn cần thì bạn có thể tạo các role với các quyền người dùng (thông thường nhiều quyền hạn chế).

9 fixed database role có thể có trong SQL Server 2000.

Database Role	Thành viên của CSDL này có thể ...
db_owner	Thực hiện bất kỳ tác vụ trong CSDL của SQL Server 2000. Các thành viên của role này có cùng quyền như là chủ của CSDL là các thành viên của dbo role.
db_accessadmin	Thêm hay xóa các user và group của Windows 2000 hoặc Win NT4.0 và các user trong một CSDL (dùng thủ tục hệ thống sp_grantdbaccess).
db_securityadmin	Quản lý tất cả các permission, role, role membership, và chuyển ower (ownership) trong một CSDL (sử dụng lệnh GRANT, REVOKE, và DENY).
db_ddladmin	Thêm, hiệu chỉnh, xóa các đối tượng trong CSDL (sử dụng lệnh CREATE, ALTER, và DROP).
db_backupoperator	Chạy các lệnh DBCC, phát hành checkpoint, và dự phòng CSDL (sử dụng các câu lệnh T-SQL: DBCC, CHECKPOINT, và BACKUP).
db_datareader	Đọc dữ liệu từ bất kỳ các bảng hoặc view của người dùng trong CSDL (bạn có quyền SELECT đối với tất cả table và view).
Db_datawriter	Hiệu chỉnh hoặc xóa dữ liệu từ các bảng hay view của người dùng trong CSDL (bạn phải có quyền INSERT, UPDATE, và DELETE đối với tất cả các table và view).
Db_denydatareader	Không đọc dữ liệu từ bất kỳ bảng trong CSDL (bạn không có quyền SELECT đối với bất kỳ đối tượng). Có thể permission on any objects). Có thể được sử dụng với role db_ddladmin để cho phép tạo các đối tượng làm chủ bằng dbo role, nhưng không có thể đọc nhạy cảm chứa trong các đối tượng đó.
Db_denydatawriter	Không hiệu chỉnh hay xóa dữ liệu từ các bảng của người dùng trong CSDL (bạn không có quyền INSERT, UPDATE, và DELETE đối với các đối tượng)

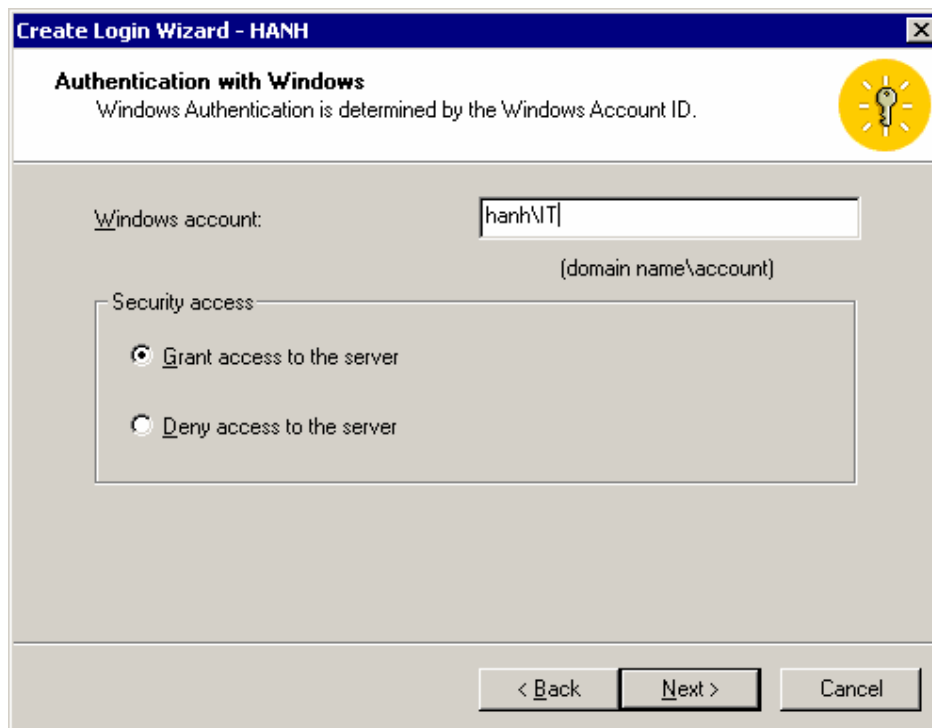
15.2 Tạo tài khoản đăng nhập (Login).

15.2.1 Dùng Create Login Wizard.



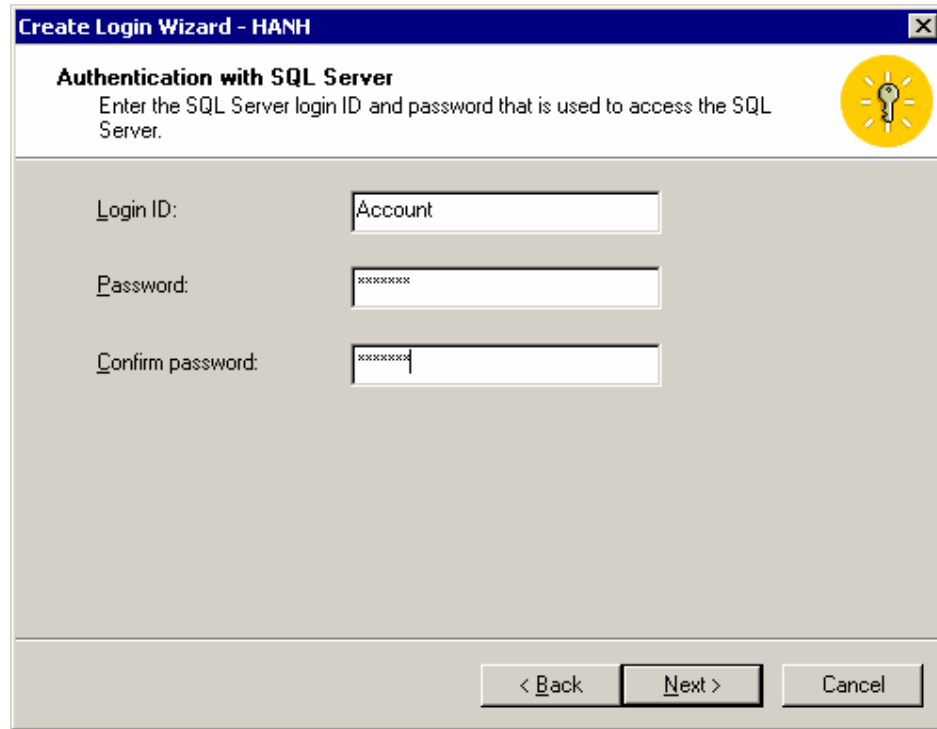
Hình 40: Chọn chế độ đăng nhập cho login đang tạo

Nếu bạn chọn Windows authentication thì bạn phải liên kết login ID này với một user hay group của Windows 2000 có sẵn. Khai báo như trong hình sau



Hình 41: Xác định 1 tài khoản của Windows 2000 và xác định cho phép hoặc từ chối login mới truy xuất đến Server

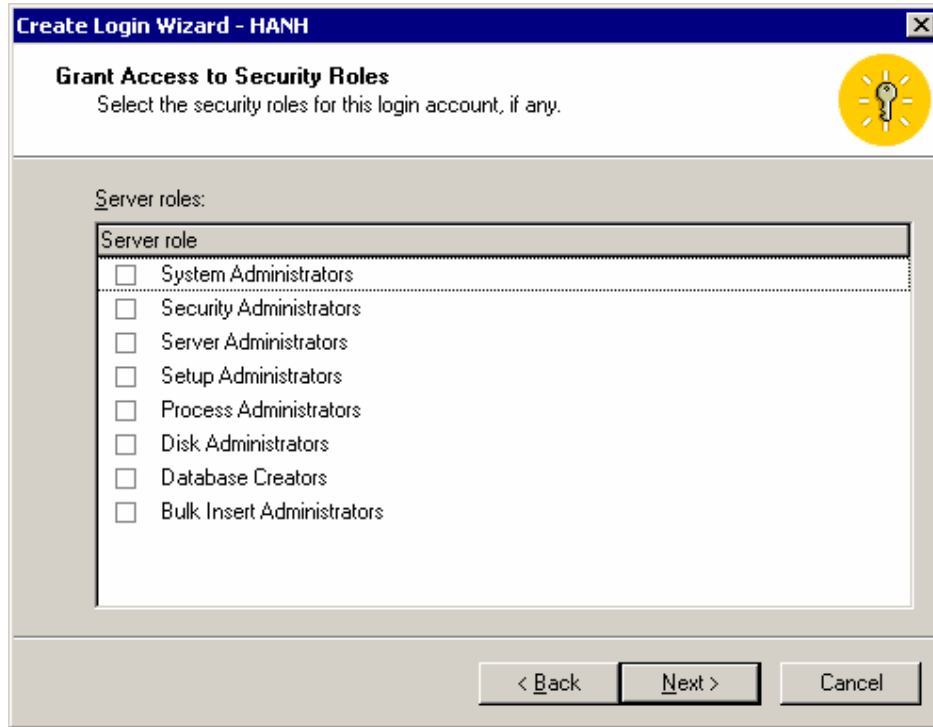
Nếu bạn chọn SQL Server authentication, bạn sẽ tạo một tài khoản bảo mật của SQL Server 2000 như trong hình kể.



Hình 42: Khai báo LogiID và Password

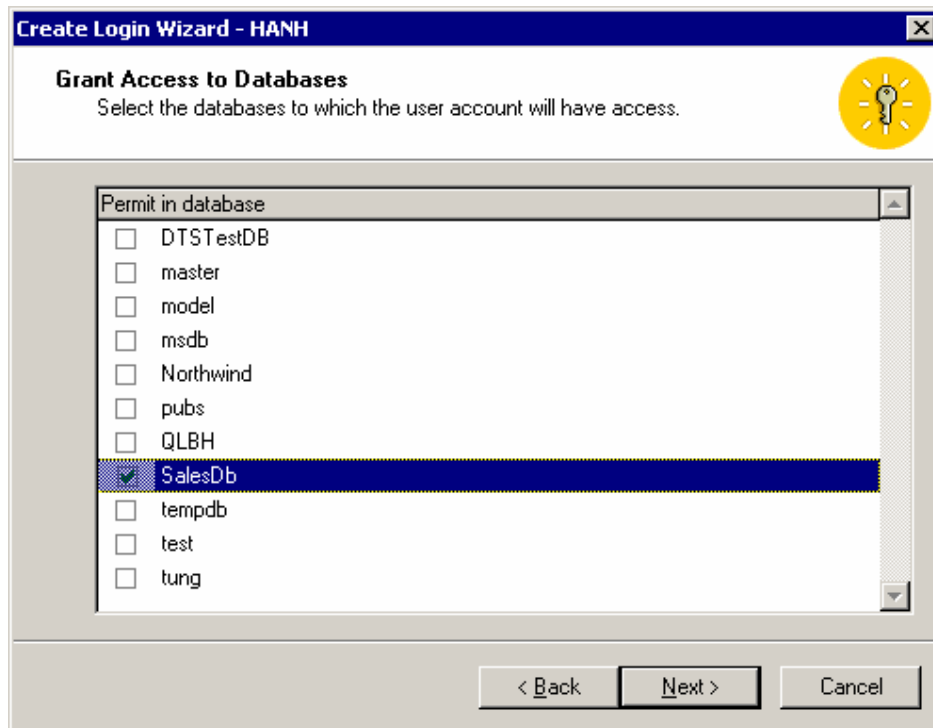
Phải chỉ định rõ tên login và Password. Để ngăn chặn (deny) một login, đơn giản bạn chỉ cần xóa bỏ login từ nơi chứa login trong Enterprise Manager (hoặc từ bảng sysxlogins trong CSDL master).

Sau khi bạn xác định kiểu đăng nhập và liên kết hay tạo tài khoản bảo mật, bạn chỉ định server role (nếu cần) cho login này trong hộp thoại kế. Nếu user sẽ không là một server-wide administrator, thì không cần chọn các server role.



Hình 43: Chỉ định server role cho login đang được tạo

Kế tiếp bạn chỉ định CSDL (nếu cần) để cho user này có thể truy xuất trong hộp thoại kế tiếp. Nhớ rằng hầu hết các server role không cung cấp CSDL truy xuất



Hình 44: Chỉ định 1 hay nhiều CSDL để login này được truy xuất

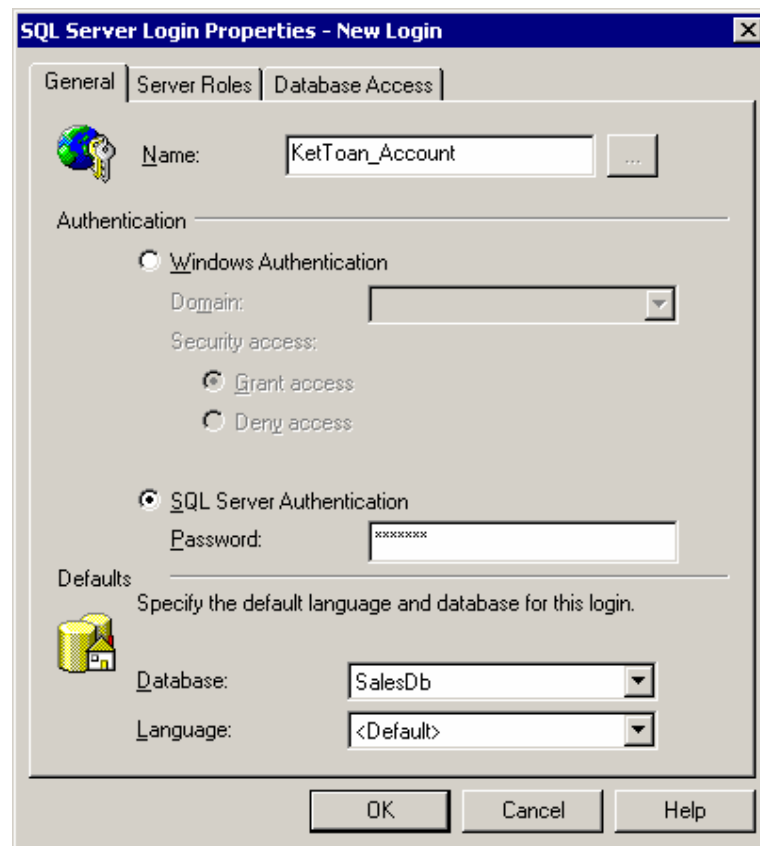
Cuối cùng, bạn được SQL Server cho xem trước các lựa chọn mà bạn đã thực hiện trước khi login mới được tạo thực sự. Click vào nút Finish để tạo login.



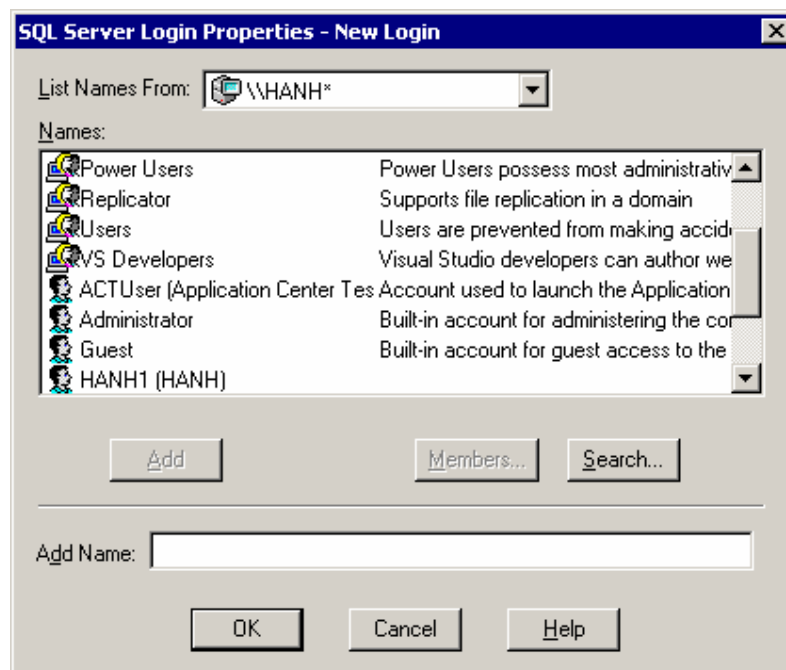
Hình 45: Chọn nút Finish để hoàn tất

15.2.2 Dùng Enterprise Manager để tạo một Login.

Để tạo một login bằng Enterprise Manager, nhấp nút phải chuột tại Security/Login của 1 instance, và chọn New Login. Thông thường, các trang Server Role, và Database Access trong hộp thoại Property của login



Hình 46: Chỉ định Name, chế độ chứng thực, chọn CSDL mặc định

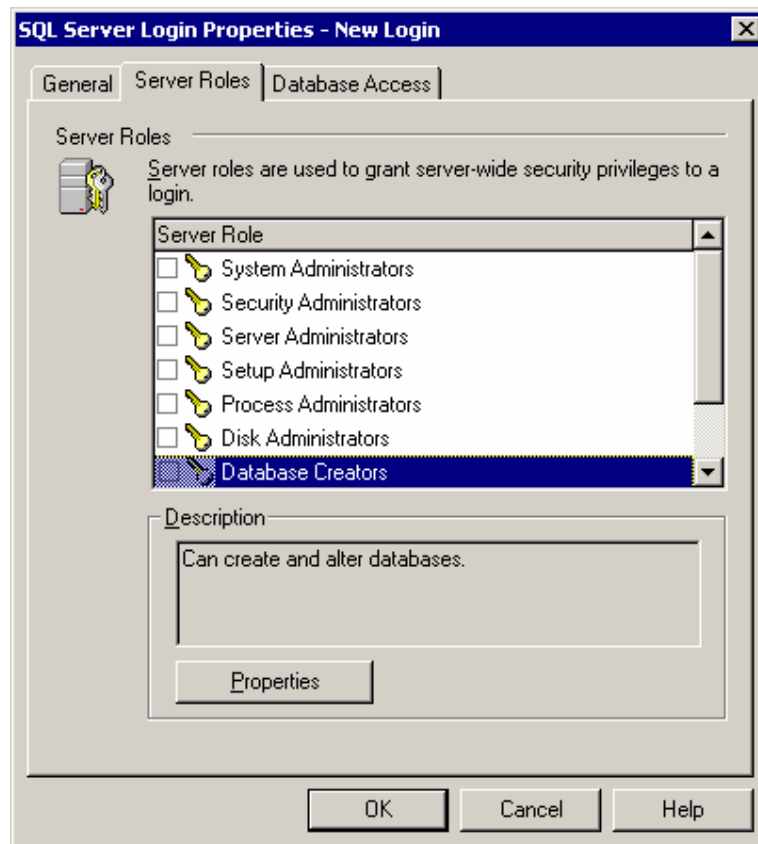


Hình 47: Chọn tài khoản người dùng của Windows

Bạn cũng chọn một CSDL mặc định và ngôn ngữ trong trang General. CSDL mặc định sẽ là CSDL hiện hành khi mà User đăng nhập vào. Mặc định không chọn là CSDL Master nhưng bạn nên chọn 1 CSDL khác làm mặc định. Ngôn ngữ mặc định là ngôn ngữ mặc định của instance hiện hành.

Lưu ý:

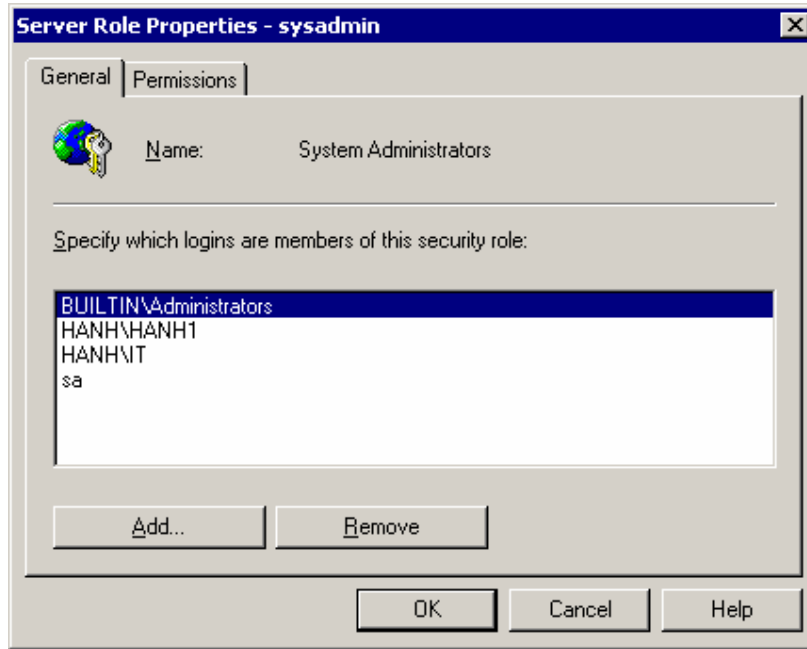
Bạn cũng có thể từ chối sự truy xuất của một User hay group của Windows. Điều này được áp đặt đến việc truy xuất khác của user hay group (kể cả thành viên của một nhóm khác mà có login khác).



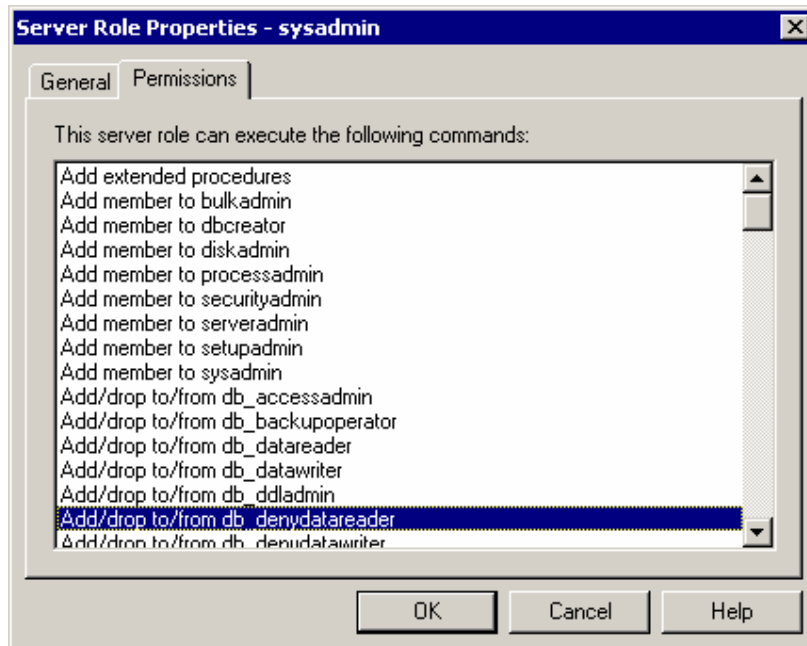
Hình 48: Gán login thuộc server role nào đó nếu cần

Lưu ý rằng nếu System Administrators server role được chọn, thì bạn có thể xem nhóm built-in Administrator group, và login sa của the SQL Server là những thành viên của sysadmin server role.

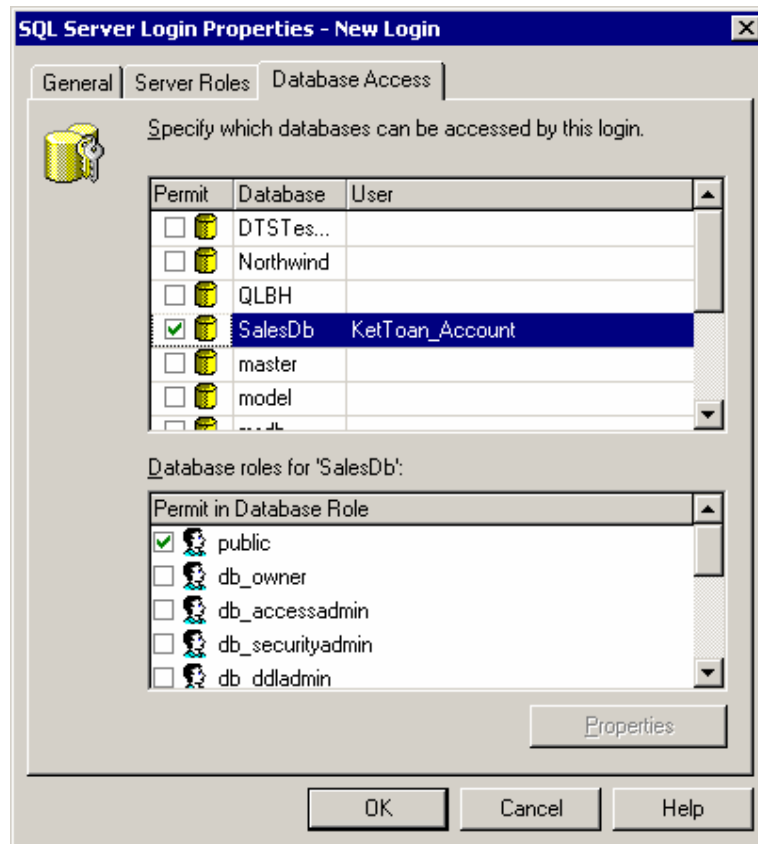
Chọn nút properties để xem một số properties của Server role



Hình 49: Thành viên của server role



Hình 50: Các quyền hạn cụ thể của Server role



Hình 51: Chỉ định CSDL được truy xuất tới và các quyền cụ thể cho login

Khi bạn chọn 1 CSDL thì login là thành viên của public role trong mỗi CSDL. Bạn không thể xóa (remove) một login ra khỏi public role.

15.2.3 Tạo Login bằng T-SQL.

Bạn cũng có thể tạo các Login bằng T-SQL

Windows Logins

Các thủ tục hệ thống dùng để cấp, hủy, từ chối, hiệu chỉnh một login cho một user hay group của Windows. Chỉ có những thành viên của sysadmin hoặc securityadmin server roles mới có thể thực thi các thủ tục hệ thống này

Thủ tục hệ thống	Mô tả
Sp_grantlogin 'login'	Tạo một login cho một user hay group của Windows 2000.
Sp_revokellogin 'login'	Hủy login từ SQL Server đối với các user hay group của Windows 2000 (hoặc Windows NT 4.0). Điều này không rõ ràng lắm vì nó vẫn có thể được cấp quyền truy xuất đến SQL Server 2000.
Sp_denylogin 'login'	Ngăn chặn một user hay thành viên của Windows 2000 (hoặc Windows NT 4.0) kết nối đến SQL Server 2000. Ngăn chặn các user hay group của thông qua một login khác kết với user hay group của Windows.
Sp_defaultdb 'login', 'database'	Thay đổi CSDL mặc định cho một login.

Thủ tục hệ thống	Mô tả
Sp_defaultlanguage 'login', 'language'	Thay đổi ngôn ngữ mặc định của một login.

Ví dụ:

```
Sp_grantlogin 'CDCN4\Bill'
```

SQL Server Logins

Các thủ tục hệ thống sau cho phép cấp, hủy, từ chối, hiệu chỉnh một login kết với một tài khoản người dùng SQL Server. Chỉ có các thành viên của sysadmin hoặc securityadmin server roles mới có thể thực thi các thủ tục hệ thống này

Thủ tục hệ thống	Mô Tả
Sp_addlogin 'login', ['password', 'database', 'language', 'sid', encryption_option']	Tạo một login SQL Server mới. Password là NULL nếu không chỉ định. CSDL mặc định là master nếu không chỉ định. Ngôn ngữ mặc định là ngôn ngữ của server hiện hành nếu không chỉ định. Mặc định, password ở trong csdl master.
Sp_droplogin 'login'	Xóa một SQL Server login.
Sp_password 'old_password', 'new_password', 'login'	Thêm hoặc thay đổi password cho SQL Server login.
Sp_defaultdb 'login', 'database'	Thay đổi CSDL mặc định
Sp_defaultlanguage 'login', 'language'	Thay đổi ngôn ngữ mặc định.

Ví dụ:

```
Sp_addlogin 'Joe', 'Joe123', 'Northwind'
```

Tạo một SQL Server login mới có tên là **Joe**, với password là **Joe123** và CSDL mặc định Northwind.

Server Roles

Các thủ tục hệ thống sau được dùng để thêm hay xóa một login vào một server role. Chỉ có thành viên của sysadmin server role mới có thể thêm các login vào bất kỳ server role. Các thành viên của server role cũng có thể thêm các login vào server role đó.

Thủ tục hệ thống	Mô tả
Sp_addsrvrolemember 'login', 'role'	Thêm login như là thành viên của server role.
Sp_dropsrvrolemember 'login', 'role'	Xóa login không là thành viên của một server role.

Ví dụ:

```
Sp_addsrvrolemember 'Joe', 'securityadmin'
```

Thêm login có tên là Joe vào server role Security Administrator.

Database Access

Các thủ tục sau đây được dùng để thêm hay xóa một login (Windows hoặc SQL Server) hiện hữu được quyền truy xuất trong CSDL hiện hành. Không giống như SQL Server Enterprise Manager, bạn có thể cấp một nhóm của Windows 2000 (hoặc Windows NT 4.0) group truy xuất đến CSDL mà không cần tạo login trước một cách tường minh trong

bảng sysxlogins. Chỉ có các thành viên của sysadmin server role, và db_accessadmin và db_owner fixed database role mới có thể thực thi các thủ tục hệ thống này.

Thủ tục hệ thống	Mô tả
Sp_grantdbaccess 'login', 'name_in_db'	Thêm một login như là một user trong CSDL hiện hành. Mặc dù tên user name trong CSDL có thể khác với tên login, điều này không khuyến cáo.
Sp_revokedbaccess 'name'	Bỏ một login như là một user trong CSDL hiện hành.

Ví dụ:

```
USE Northwind
EXEC Sp_grantdbaccess 'Joe'
```

Cho phép login tên là Joe truy xuất đến CSDL hiện hành, dùng user name là Joe trong CSDL Northwind.

Database Roles

Các thủ tục hệ thống sau đây được dùng để thay đổi database owner, thêm hoặc xóa một tài khoản bảo mật vào một database role có sẵn, hoặc tạo hoặc xóa một user-defined database role.

```
Use Northwind
```

```
EXEC Sp_addrolemember 'db_securityadmin', ' CDCN4\KeToan'
```

Thêm tài khoản CDCN4\KeToan vào db_securityadmin database role trong CSDL Northwind.

Thủ tục hệ thống	Mô tả
Sp_changedbowner 'login', remap_alias_flag	Thay đổi owner của một CSDL người dùng. Chỉ có những thành viên của sysadmin server role hoặc owner database hiện hành mới có thể thay đổi owner của CSDL.
Sp_addrolemember 'role', 'security_account'	Thêm một tài khoản vào một database role trong CSDL hiện hành. Bạn có thể thêm một user-defined database role vào fixed hoặc user-defined database role. Chỉ có những thành viên của sysadmin server role và db_owner and db_security fixed database roles mới có thể thêm thành viên vào database role. Thành viên của database role có thể thành viên vào cho database role đó.
Sp_droprolemember 'role', 'security_account'	Xóa một tài khoản từ một CSDL vào CSDL hiện hành. Chỉ có những thành viên của sysadmin server role và db_owner và db_security fixed database roles mới có thể xóa các thành viên ra khỏi database role. Các thành viên của database role mới có thể xóa các thành viên ra khỏi database role.
Sp_addrole 'role', 'owner'	Thêm một user-defined database role mới trong CSDL hiện hành. Mặc dù bạn có thể chỉ định một owner của role, sử dụng mặc định là dbo là không được khuyến cáo. Các thành viên của sysadmin server role và db_securityadmin và db_owner fixed database roles mới có thể tạo user-defined database roles.
Sp_droprole 'role'	Xóa một user-defined database role ở CSDL hiện hành. Các thành

Thủ tục hệ thống	Mô tả
	viên của sysadmin server role, db_securityadmin và db_owner fixed database roles mới có thể xóa user-defined database roles.

Để xem các thông tin về login, dùng các thủ tục hệ thống sau:

Thủ tục hệ thống	Mô tả
Sp_helplogins ['login']	Trả về các thông tin của tất cả các login hoặc một login được chỉ định, kể cả các CSDL mà login có truy xuất đến và các database roles mà login là thành viên.
Sp_helpsrvrolemember ['role']	Trả về thông tin về tất cả các server roles và những thành viên của chúng hoặc tất cả các thành viên trong một server role chỉ định.
Sp_helpuser ['security_account']	Trả về thông tin về tất cả các user hoặc user chỉ định trong CSDL hiện hành, kể cả tất cả các hội thành viên của database role.
Sp_helprolemember ['role']	Trả về các thông tin của tất cả các database role hoặc tất cả các thành viên trong database role chỉ định trong CSDL hiện hành.
Sp_helpntgroup ['name']	Trả về các thông tin về các nhóm hoặc 1 nhóm chỉ định của Windows 2000 (hoặc Windows NT 4.0) trong CSDL hiện hành.

Gán các quyền Database

Các view và các stored procedure có thể được tạo trên các table. Khi một user cố gắng lấy thông tin thông qua view hoặc procedure, thì SQL Server 2000 phải kiểm tra user có được phép lấy dữ liệu hay không. Nếu view hoặc procedure được làm chủ bởi một user và của các bảng cơ sở lại là của một user khác thì SQL Server 2000 phải kiểm tra các quyền trên mỗi object trong dây chuyền đó. Khi một chuỗi các ownership kéo dài thì điều này sẽ ảnh hưởng đến việc thực hiện. Nhưng có lẽ quan trọng hơn là nó có thể bị gắt gỏi cho nhà quản trị vạch ra và gỡ lỗi các từ việc đưa ra bảo mật

Các quyền về lệnh

Câu lệnh Transact-SQL	Quyền để thực thi câu lệnh Transact-SQL
CREATE DATABASE	Thừa kế bởi thành viên của sysadmin và dbcreator server roles. Mặc dù sysadmin và securityadmin server roles có thể cấp quyền một cách trực tiếp cho các tài khoản để thực hiện câu lệnh này, Tóm lại các tài khoản bảo mật sử dụng dbcreator server role nếu system administrator đại diện quyền. Quyền này chỉ tồn tại trong CSDL master.
BACKUP DATABASE BACKUP LOG	Kế thừa bởi các thành viên của sysadmin server role và db_owner và db_backupoperator fixed database roles. Mặc dù bạn có thể cấp quyền để chạy những câu lệnh này một cách trực tiếp đến các tài khoản bảo mật, một cách tổng quát bạn sẽ sử dụng db_backupoperator fixed database role.

Câu lệnh Transact-SQL	Quyền để thực thi câu lệnh Transact-SQL
CREATE TABLE CREATE VIEW CREATE PROCEDURE CREATE DEFAULT CREATE RULE CREATE FUNCTION	Kế thừa các thành viên của sysadmin server role và db_owner và db_ddladmin fixed database roles. Quyền cho phép tạo những đối tượng thì được cấp trực tiếp đến nhà lập trình trong suốt thời gian triển khai. Theo mặc định các object được làm chủ bởi người tạo ra đối tượng (mặc dù các đối tượng tạo bởi các thành viên của sysadmin server role thì chủ sẽ là dbo role). Các thành viên của db_owner hoặc db_ddladmin fixed database roles có thể được chỉ định dbo role như là owner của đối tượng được tạo. Ngoài ra, các thành viên của sysadmin server role hoặc db_owner hoặc db_ddladmin fixed database role có thể chỉ định bất kỳ user như là chủ của object mà chúng tạo ra. Tuy nhiên, các user mà không là thành viên của một trong các role này thì không thể chỉ định user khác hoặc dbo role làm chủ của object chúng tạo ra.
CREATE TRIGGER	Kế thừa bởi chủ của table mà của sysadmin server role, và db_owner and db_ddladmin fixed database roles. Những thành viên này không thể cấp quyền để chạy câu lệnh này cho những tài khoản bảo mật khác..

Ví dụ:

```
CREATE TABLE Northwind.dbo.CustomerTable
    (CustID nchar (5), CustomerName nvarchar (40))
```

Cho phép tạo một bảng, cấp ownership cho dbo role. Chỉ có những thành viên sysadmin server role và db_owner hoặc db_ddladmin fixed database roles có thể thực hiện một cách thành công câu lệnh này.

Thay đổi Ownership của Object

Một thành viên của db_owner, db_ddladmin, hoặc db_securityadmin fixed database role, hoặc a member of the sysadmin server role có thể thay đổi ownership của bất kỳ object trong bằng cách chạy thủ tục sp_changeobjectowner.

```
sp_changeobjectowner 'CDCN4\KeToan.Customer', 'dbo'
```

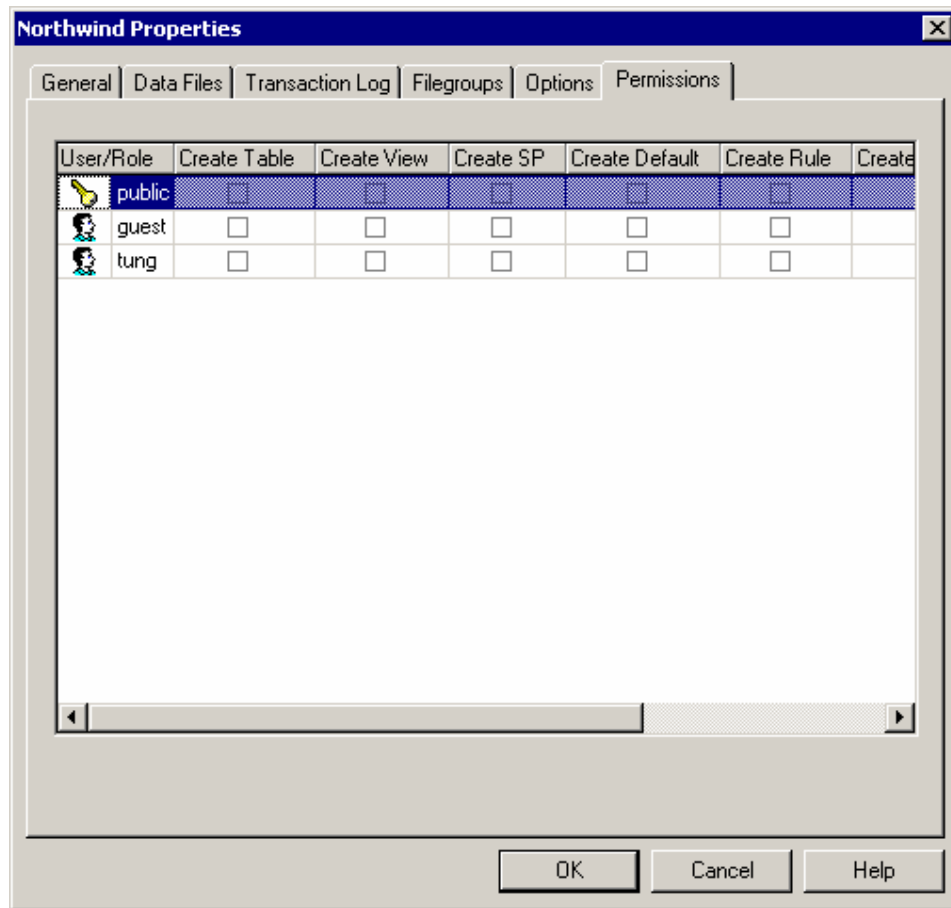
Đổi ownership của table Customer từ CDCN4\KeToanBill cho dbo role.

Lưu ý

Thay đổi owner của một object thì sẽ xóa tất cả các quyền hiện có trên đối tượng. Nếu bạn cần giữ lại các quyền thì nên tạo Scrip trước khi đổi owner.

Dùng Enterprise Manager để Grant, Deny, or Revoke quyền lệnh

Nhấp nút phải chuột tại CSDL cần thực hiện cấp/xóa/từ chối/xem các quyền



Hình 52: Trang Permission của hộp thoại thuộc tính của CSDL

Dùng Transact-SQL để Grant, Deny, or Revoke các quyền

Dùng câu lệnh GRANT CREATE TABLE TO

```
GRANT CREATE TABLE TO Joe, SalesManagers,
    [CDCN4\SQLServerAdmins]
```

Cấp quyền lệnh CREATE TABLE cho Joe (một SQL Server login), SalesManagers (a user-defined database role), và CDCN4\SQLServerAdmins (một Windows group).

Xem các quyền bằng Transact-SQL : Dùng sp_helpprotect

```
EXEC sp_helpprotect NULL, NULL, NULL, 's'
```

Xem tất cả các quyền câu lệnh trong CSDL hiện hành.