

SQL SERVER 2000



GV. Phạm Thị Hoàng Nhung
Bộ môn Công nghệ phần mềm
Đại học Thủy lợi

NỘI DUNG



- Tổng quan SQL Server 2000
- Thiết kế và thực thi cơ sở dữ liệu
- T-SQL Programming
- Giao dịch và Khoá
- Bảo mật và Quản lý người dùng
- T-SQL và SQL nâng cao
- Ràng buộc dữ liệu và Chỉ số
- Khung nhìn và Con trỏ
- Thủ tục
- Trigger
- Sao lưu và Phục hồi
- Chuyển đổi giữa các loại cơ sở dữ liệu
- Kiến trúc nhân bản

1	CHƯƠNG 1. TỔNG QUAN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU SQL SERVER 2000	10
1.1	Giới thiệu SQL Server 2000	10
1.1.1	Các đặc trưng của SQL Server 2000:	10
1.1.2	Các phiên bản-edition của SQL Server	11
1.1.3	Các Version của SQL Server	12
1.2	Các thành phần quan trọng trong SQL Server 2000	12
1.2.1	Relational Database Engine	12
1.2.2	Replication - Cơ chế tạo bản sao	12
1.2.3	Data Transformation Service (DTS) - Dịch vụ chuyển dịch dữ liệu	13
1.2.4	Analysis Service - Dịch vụ phân tích dữ liệu	13
1.2.5	English Query – Truy vấn dữ liệu sử dụng tiếng Anh	14
1.2.6	Meta Data Service	14
1.2.7	SQL Server Books Online – Sách dạy SQL Server trực tuyến	14
1.3	SQL Server Tools	14
1.3.1	Enterprise Manager	15
1.3.2	Query Analyzer	15
1.3.3	SQL Profiler	15
1.4	Kiến trúc của SQL Server	16
1.4.1	Client/Server Database system	16
1.4.2	Desktop Database system	16
1.5	SQL Server Database	16
1.6	Database Objects-Các đối tượng trong cơ sở dữ liệu	17
1.7	Câu hỏi trắc nghiệm	18
2	CHƯƠNG 2. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU	19
2.1	Cấu trúc của SQL Server	19
2.2	Cấu trúc vật lý của một SQL Server Database	19
2.3	Nguyên tắc hoạt động của transaction log trong SQL Server	20
2.4	Cấu trúc logic của một SQL Server Database	22
2.5	Các kiểu dữ liệu trong SQL Server (data types)	22
2.5.1	Integers	22
2.5.2	Decimal and Numeric	23
2.5.3	Money and Smallmoney	23
2.5.4	Approximate Numerics	23
2.5.5	Datetime and Smalldatetime	24
2.5.6	Character Strings	24
2.5.7	Unicode Character Strings	24
2.5.8	Binary Strings	25
2.5.9	Các kiểu dữ liệu khác	25
2.6	Câu hỏi trắc nghiệm	26
3	CHƯƠNG 3. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU- PHẦN THỰC HÀNH	27

3.1	Tạo cơ sở dữ liệu sử dụng Enterprise Manager	27
3.2	Sửa cơ sở dữ liệu	30
3.3	Xoá cơ sở dữ liệu	30
3.4	Nghiên cứu cơ sở dữ liệu Flight_Information	30
3.4.1	Cấu trúc bảng.....	30
3.4.2	Dữ liệu trên các bảng.....	34
3.5	Bài tập	39
4	CHƯƠNG 4. T-SQL PROGRAMING	40
4.1	Giới thiệu SQL Batch Processing	40
4.1.1	Cách thực Thi một nhóm lệnh (Batches).....	40
4.1.2	Lệnh GO	40
4.1.3	Ví dụ về Batch:	41
4.1.4	Chú thích (comment) trong batch:.....	41
4.2	Câu lệnh điều khiển	41
4.2.1	Begin..End	41
4.2.2	If..Else.....	42
4.2.3	Vòng lặp While.....	42
4.2.4	Từ khoá GOTO.....	43
4.2.5	Từ khoá Return.....	43
4.2.6	Câu lệnh CASE.....	43
4.3	Biến(Variables)	44
4.3.1	Grobal variables.....	44
4.3.2	Local variables.....	45
4.4	Hàm (Functions)	46
4.4.1	Hàm Conversion.....	46
4.4.2	Hàm Data Parts.....	46
4.4.3	Hàm ngày tháng và hàm toán học	47
4.4.4	Hàm hệ thống (System Function).....	47
4.4.5	Hàm nhóm	48
4.5	Câu hỏi trắc nghiệm	49
5	CHƯƠNG 5. TRANSACTIONS VÀ LOCKS	51
5.1	Giới thiệu Transactions-Giao dịch	51
5.2	Các tính chất của Transaction	51
5.2.1	Phân loại transaction.....	51
5.3	Các mức cô lập của Transaction	52
5.3.1	Giới thiệu Dirty Read (Đọc các dữ liệu bẩn).....	52
5.3.2	Các mức cô lập.....	54
5.4	Locks	55
5.4.1	Khái niệm	55
5.4.2	Phân loại	55
5.5	Câu hỏi trắc nghiệm	57
6	CHƯƠNG 6. BẢO MẬT VÀ QUẢN LÝ NGƯỜI DÙNG (USER AND SERCURITY)	59

6.1	Giới thiệu về SQL Server Security	59
6.2	Quản lý đăng nhập (Login)	59
6.2.1	Xác thực đăng nhập	59
6.2.2	Kiểm tra quyền (Permission).....	60
6.2.3	Tạo Login	60
6.3	Quản lý người dùng	61
6.3.1	SQL Server Users	61
6.3.2	Quản lý Username và Login name	62
6.4	Quản lý Role	62
6.4.1	Database Roles	62
6.4.2	Server Roles.....	63
6.4.3	Thêm thành viên cho Role.....	64
6.5	Đối tượng và quyền trên đối tượng (Database Objects and Object Permission)	64
6.5.1	Đối tượng.....	64
6.5.2	Quyền.....	65
6.5.3	Cho phép và huỷ bỏ quyền trên đối tượng	65
6.6	Câu hỏi trắc nghiệm	67
7	CHƯƠNG 7. T-SQL PROGRAMMING, TRANSACTIONS, MANAGING SECURITY – PHẦN THỰC HÀNH	69
7.1	Hướng dẫn trực tiếp	69
7.1.1	Transactions.....	69
7.1.2	Biến địa phương (local) và biến toàn cục(Global)	73
7.1.3	SQL Server Security	74
7.2	Bài tập	74
8	CHƯƠNG 8. T-SQL VÀ SQL NÂNG CAO	76
8.1	Giới thiệu sơ lược về T- SQL (Transact -SQL)	76
8.1.1	Data Definition Language (DDL)	76
8.1.2	Data Control Language (DCL):.....	77
8.2	Data Manipulation Language (DML):	77
8.3	Các câu lệnh truy vấn dữ liệu	78
8.3.1	Thực hiện Join để kết nối các bảng	78
8.3.2	Mệnh đề Top <i>n</i> :.....	81
8.3.3	Mệnh đề INTO.....	81
8.3.4	Từ khoá UNION(Hợp)	82
8.3.5	Từ khoá CUBE và ROLL UP.....	82
8.3.6	Mệnh đề COMPUTE và COMPUTE BY	85
8.4	Câu hỏi trắc nghiệm	87
9	CHƯƠNG 9. T-SQL VÀ SQL NÂNG CAO- PHẦN THỰC HÀNH	88
10	CHƯƠNG 10. RÀNG BUỘC DỮ LIỆU VÀ CHỈ SỐ	90
10.1	Ràng buộc dữ liệu	90
10.1.1	Giới thiệu.....	90
10.1.2	Ràng buộc thực thể.....	90

Chương 1. TỔNG QUAN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU SQL SERVER 2000

10.1.3	Ràng buộc miền dữ liệu.....	91
10.1.4	Ràng buộc tham chiếu.....	91
10.1.5	Ràng buộc NSD tự định nghĩa.....	91
10.2	Thực hiện các ràng buộc bằng T-SQL.....	91
10.2.1	PRIMARY KEY Constraint.....	91
10.2.2	UNIQUE Constraint.....	92
10.2.3	IDENTITY Property.....	92
10.2.4	DEFAULT Definition.....	92
10.2.5	FOREIGN Key Constraint.....	93
10.2.6	CHECK Constraint.....	93
10.2.7	NOT NULL Constraint.....	93
10.2.8	Rules.....	94
10.3	Indexes.....	94
10.3.1	Giới thiệu.....	94
10.3.2	Lời khuyên khi sử dụng indexes.....	95
10.3.3	Tạo Indexes.....	95
10.3.4	Các kiểu Indexes.....	96
10.3.5	Tính chất của Indexes.....	96
10.3.6	Hiển thị Indexes.....	97
10.3.7	Cách sử dụng Indexes.....	97
10.3.8	Xóa Indexes.....	97
10.3.9	Full-text Searches.....	98
10.3.10	Full-text Catalogs.....	98
10.3.11	Sử dụng Full-text Indexes.....	98
10.4	Câu hỏi trắc nghiệm.....	99
11	CHƯƠNG 11. DATA INTEGRITY AND INDEXES.....	101
	PHẦN THỰC HÀNH.....	101
11.1	Hướng dẫn trực tiếp.....	101
11.1.1	Tạo ràng buộc PRIMARY KEY.....	101
11.1.2	Tạo ràng buộc Unique.....	102
11.1.3	Sử dụng thuộc tính IDENTITY.....	103
11.1.4	Tạo ràng buộc Default.....	103
11.1.5	Tạo ràng buộc FOREIGN KEY.....	104
11.1.6	Tạo ràng buộc Check Constraint.....	106
11.1.7	Tạo ràng buộc Not Null.....	107
11.1.8	Tạo Rules.....	108
11.2	Indexes.....	109
11.2.1	Tạo indexes.....	109
11.2.2	Xem và sửa Indexes.....	111
11.2.3	Sử dụng Indexes.....	112
11.3	Bài tập.....	114
12	CHƯƠNG 12. KHUNG NHÌN & CON TRỎ.....	115
	(VIEWS & CURSORS).....	115

12.1	View	115
12.1.1	Giới thiệu	115
12.1.2	Tạo View	116
12.1.3	Lợi ích của View đối với người sử dụng	117
12.1.4	Một số hướng dẫn khi tạo View	117
12.1.5	Sửa dữ liệu thông qua Views	117
12.1.6	Indexed Views	118
12.1.7	Distributed Partitioned Views- Khung nhìn phân tán	120
12.1.8	Sử dụng View để cập nhật dữ liệu	123
12.1.9	Sửa cấu trúc Views	123
12.1.10	Xoá Views	124
12.2	Con trỏ_Cursors	124
12.2.1	Giới thiệu	124
12.2.2	Tạo con trỏ	124
12.2.3	Các bước trong sử dụng Cursor	125
12.2.4	Truy cập dữ liệu bằng cursor	125
12.2.5	Ví dụ	126
12.3	Câu hỏi trắc nghiệm	127
13	CHƯƠNG 13. KHUNG NHÌN VÀ CON TRỎ - PHẦN THỰC HÀNH	129
13.1	Tạo View	129
13.1.1	Sử dụng Create View Wizard	129
13.1.2	Tạo View bằng T-SQL	130
13.2	Sửa View	130
13.3	Con trỏ	131
13.3.1	Khai báo con trỏ (Cursor)	131
13.3.2	Mở con trỏ	132
13.3.3	Truy vấn dữ liệu	132
13.3.4	Truy vấn dòng đầu tiên	132
13.3.5	Truy vấn dòng tiếp theo	133
13.3.6	Truy vấn dòng cuối cùng	133
13.3.7	Truy vấn đến một dòng có vị trí xác định	133
13.3.8	Truy vấn đến dòng liên quan	133
13.3.9	Đóng và xoá vùng nhớ (Deallocating) của con trỏ	134
13.4	Bài tập	135
14	CHƯƠNG 14. THỦ TỤC- STORED PROCEDURES(SPS)	136
14.1	Định nghĩa	136
14.2	Lợi ích khi quản lý dữ liệu bằng SPs	137
14.3	Các kiểu SPs	137
14.3.1	System stored procedures	138
14.3.2	User-defined Stored Procedures	139
14.4	Thông báo lỗi	142
14.4.1	Return Codes	142

14.4.2	Câu lệnh RAISERROR	143
14.5	Câu hỏi trắc nghiệm.....	145
15	CHƯƠNG 15. STORED PROCEDURE- PHẦN THỰC HÀNH.....	146
15.1	Tạo SP bằng EM.	146
15.2	Thực thi SP	147
15.3	Bài tập	147
16	CHƯƠNG 16. TRIGGER.....	149
16.1	Định nghĩa.....	149
16.2	Đặc điểm của Trigger	150
16.3	Tạo Trigger.....	151
16.3.1	Tạo Trigger.....	151
16.3.2	Hướng dẫn khi tạo Trigger	151
16.4	Các kiểu Trigger	152
16.4.1	INSERT trigger.....	152
16.4.2	UPDATE trigger.....	153
16.4.3	DELETE trigger	155
16.5	Các câu lệnh không thể sử dụng trong Triggers.....	156
16.6	Triggers dây chuyền - Cascading Triggers.....	156
16.7	Triggers lồng nhau - Nested Triggers	157
16.8	INSTEAD OF Triggers.....	157
16.9	Câu hỏi trắc nghiệm.....	159
17	CHƯƠNG 17. TRIGGER – PHẦN THỰC HÀNH	160
17.1	Tạo INSERT trigger.	160
17.2	Tạo DELETE Trigger.....	160
17.3	Tạo UPDATE Trigger.	161
17.3.1	Tạo Table Level UPDATE Trigger.....	161
17.3.2	Tạo Column Level Update Trigger	162
17.4	Tạo Trigger có lựa chọn Encryption.....	163
17.5	Hiển thị danh sách các trigger trong Database.....	163
17.6	Sử dụng Triggers để tạo ràng buộc tham chiếu (Enforce Referential Interguity)	164
17.7	Cascade Delete sử dụng Nested trigger.....	164
17.8	Tạo INSTEAD OF Trigger	166
17.9	Bài tập	167
18	CHƯƠNG 18. SAO LƯU & PHỤC HỒI.....	169
	(BACKUP & RESTORE).....	169
18.1	Giới thiệu	169
18.2	Sao lưu cơ sở dữ liệu	169
18.3	Phục hồi cơ sở dữ liệu	170
18.4	Các loại Backup và Restore	173
18.4.1	Các loại sao lưu-Backups	173
18.4.2	Các mô hình khôi phục- Recovery Models	173
18.5	Full Database backup	174

18.5.1	Cách tạo Full database backup bằng EM	175
18.5.2	Khôi phục Full database backup bằng EM.....	177
18.6	Transaction log backup	178
18.6.1	Giới thiệu.....	178
18.6.2	Cắt (truncate) transaction log	179
18.6.3	Điều kiện transaction log backups.....	180
18.6.4	Cách tạo transaction log backup bằng EM.....	180
18.6.5	Khôi phục transaction log backup bằng EM	180
18.7	Differential backup	182
18.8	File hoặc Filegroup backup.....	183
19	CHƯƠNG 19. CHUYỂN ĐỔI GIỮA CÁC LOẠI CƠ SỞ DỮ LIỆU (DTS- DATA TRANSFORMATION SERVICE)	185
19.1	Import cơ sở dữ liệu.....	185
19.1.1	Import cơ sở dữ liệu từ SQL Server 2000	185
19.1.2	Import từ cơ sở dữ liệu Access	191
19.1.3	Import từ tập tin Excel.....	192
19.1.4	Import từ tập tin dạng Text.....	192
19.2	Export cơ sở dữ liệu	194
19.3	Xây dựng lịch trình Import và Export cơ sở dữ liệu.....	194
19.4	Những điều cần giải quyết sau khi Import hay Export.....	195
20	CHƯƠNG 20. KIẾN TRÚC NHÂN BẢN (REPLICATION)	196
20.1	Mục tiêu chính của nhân bản.....	196
20.1.1	Nhất quán dữ liệu (Data consistency)	196
20.1.2	Độc lập site (site autonomy).....	197
20.2	Kiến trúc nhân bản	197
20.2.1	Các thành phần chính của nhân bản:	197
20.2.2	Chiều di chuyển dữ liệu.....	198
20.3	Tác nhân (Agent).....	199
20.4	Các loại nhân bản.....	200
20.5	Nhân bản snapshot(Snapshot replication).....	200
20.5.1	Giới thiệu.....	200
20.5.2	Tác nhân (agent).....	201
20.6	Nhân bản giao dịch (transactional replication).....	202
20.6.1	Giới thiệu.....	202
20.6.2	Tác nhân (agent).....	202
20.6.3	Thu dọn trong nhân bản transaction	204
20.7	Các dạng nhân bản giao dịch.....	204
20.7.1	Cập nhật Subscriber lập tức(Immediate_Updating Subscriber).....	204
20.7.2	Nhân bản những thực thi của Stored procedure	207
20.8	Nhân bản kết hợp (Merge replication)	207
20.8.1	Giới thiệu.....	207
20.8.2	Tác nhân (agent).....	208
20.8.3	Giải quyết tranh chấp trong nhân bản kết hợp	209

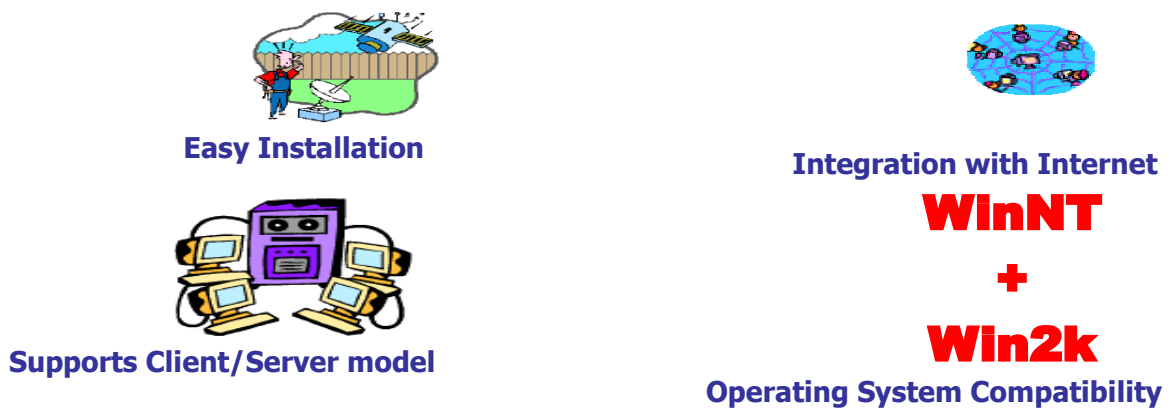
20.9 Giải quyết tranh chấp..... 209

1 Chương 1. TỔNG QUAN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU SQL SERVER 2000

1.1 Giới thiệu SQL Server 2000

SQL Server 2000 bao gồm một số đặc trưng tạo nên một Hệ quản trị cơ sở dữ liệu đáp ứng được yêu cầu rất cao trong thực thi cơ sở dữ liệu.

1.1.1 Các đặc trưng của SQL Server 2000:



Hình 1.1. Các đặc trưng của SQL Server

Đễ cài đặt (Easy Installation): SQL Server cung cấp các công cụ quản trị và phát triển để cho người sử dụng dễ dàng cài đặt, sử dụng và quản lý hệ thống.

Tích hợp với Internet(Integration with Internet): SQL Server 2000 database engine hỗ trợ XML. Nó được tối ưu để có thể chạy trên môi trường cơ sở dữ liệu rất lớn (Very Large Database Environment) lên đến Tera-Byte và có thể phục vụ cùng lúc cho hàng ngàn user. Mô hình lập trình (programming model) SQL Server 2000 được tích hợp với kiến trúc Windows DNA trợ giúp cho phát triển ứng dụng Web. Nó cũng hỗ trợ một số đặc tính khác như English Query để người phát triển hệ thống có thể truy vấn dữ liệu thân thiện hơn. Và Microsoft Search Services cung cấp khả năng tìm kiếm rất mạnh, đặc biệt thích hợp cho phát triển ứng dụng Web.

Hỗ trợ kiến trúc Client/Server(Supports Client/Server model): Ứng dụng có thể chạy trên Client, truy cập dữ liệu được lưu trữ trên Server. Server có nhiệm vụ xử lý các yêu cầu và trả lại kết quả cho Client.

Tương thích với nhiều hệ điều hành(Operating System Compatibility): Có thể cài đặt trên hầu hết các hệ điều hành của Microsoft (danh sách chi tiết kèm theo). Chú ý khi cài đặt trên Windows NT Server 4, bạn phải chạy thêm Service Pack 5(SP5).

Chương 1. TỔNG QUAN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU SQL SERVER 2000

Operating System	Enterprise Edition	Standard Edition	Personal Edition	Developer Edition	Desktop Engine	SQL Server CE	Enterprise Evaluation Edition
Microsoft Windows® 2000 DataCenter	Supported	Supported	Supported	Supported	Supported	N/A	Supported
Windows 2000 Advanced Server	Supported	Supported	Supported	Supported	Supported	N/A	Supported
Windows 2000 Server	Supported	Supported	Supported	Supported	Supported	N/A	Supported
Windows 2000 Professional	N/A	N/A	Supported	Supported	Supported	N/A	Supported
Microsoft Windows NT® 4.0 Server, Enterprise Edition	Supported	Supported	Supported	Supported	Supported	N/A	Supported
Windows NT 4.0 Server	Supported	Supported	Supported	Supported	Supported	N/A	Supported
Windows NT 4.0 Workstation	N/A	N/A	Supported	Supported	Supported	N/A	Supported
Microsoft Windows 98	N/A	N/A	Supported	N/A	Supported	N/A	N/A
Microsoft Windows CE	N/A	N/A	N/A	N/A	N/A	Supported	N/A

1.1.2 Các phiên bản-edition của SQL Server

Enterprise: Chứa đầy đủ các đặc trưng của SQL Server và có thể chạy tốt trên hệ thống lên đến 32 CPUs và 64 GB RAM. Thêm vào đó nó có các dịch vụ giúp cho việc phân tích dữ liệu rất hiệu quả (Analysis Services).

Standard: Rất thích hợp cho các công ty vừa và nhỏ vì giá thành rẻ hơn nhiều so với Enterprise Edition, nhưng lại bị giới hạn một số chức năng cao cấp khác, edition này có thể chạy tốt trên hệ thống lên đến 4 CPU và 2 GB RAM.

Personal: được tối ưu hóa để chạy trên PC nên có thể cài đặt trên hầu hết các phiên bản của windows, kể cả Windows 98.

Developer: Có đầy đủ các tính năng của Enterprise Edition nhưng được chế tạo đặc biệt như giới hạn số lượng người kết nối vào Server cùng một lúc.... Đây là edition mà các bạn muốn học SQL Server cần có. Edition này có thể cài trên Windows 2000 Professional hay Win NT Workstation.

Desktop Engine (MSDE): Đây chỉ là một engine chạy trên desktop và không có user interface (giao diện). Thích hợp cho việc triển khai ứng dụng ở máy client. Kích thước cơ sở dữ liệu bị giới hạn khoảng 2 GB.

Win CE : Dùng cho các ứng dụng chạy trên Windows CE

Trial: Có các tính năng của Enterprise Edition, download free, nhưng giới hạn thời gian sử dụng.

1.1.3 Các Version của SQL Server

SQL Server của Microsoft được thị trường chấp nhận rộng rãi kể từ version 6.5. Sau đó Microsoft đã cải tiến và hầu như viết lại một engine mới cho SQL Server 7.0. Cho nên có thể nói từ version 6.5 lên version 7.0 là một bước nhảy vọt. Có một số đặc tính của SQL Server 7.0 không tương thích với version 6.5. Trong khi đó từ Version 7.0 lên SQL Server 2000 thì những cải tiến chủ yếu là mở rộng các tính năng về Web và làm cho SQL Server 2000 đáng tin cậy hơn.

1.2 Các thành phần quan trọng trong SQL Server 2000

SQL Server 2000 được cấu tạo bởi nhiều thành phần như Relational Database Engine, Analysis Service và English Query.... Các thành phần này khi phối hợp với nhau tạo thành một giải pháp hoàn chỉnh giúp cho việc lưu trữ và phân tích dữ liệu một cách dễ dàng.

1.2.1 Relational Database Engine

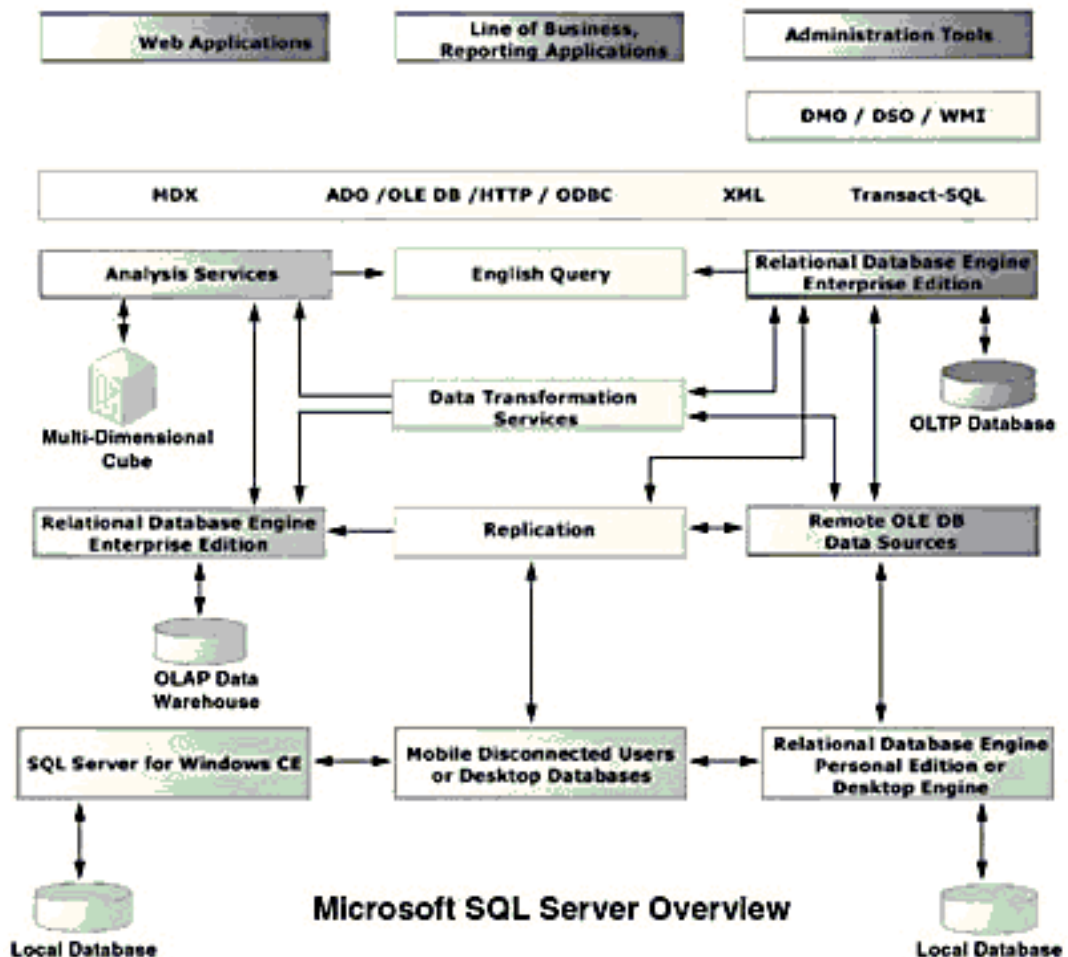
Đây là một engine có khả năng chứa dữ liệu ở các quy mô khác nhau dưới dạng table và hỗ trợ tất cả các kiểu kết nối (data connection) thông dụng của Microsoft như ActiveX Data Objects (ADO), OLE DB, and Open Database Connectivity (ODBC). Ngoài ra nó còn có khả năng tự điều chỉnh (tune up), ví dụ như sử dụng thêm các tài nguyên (resource) của máy khi cần và trả lại tài nguyên cho hệ điều hành khi một user log off.

1.2.2 Replication - Cơ chế tạo bản sao

Giả sử bạn có một cơ sở dữ liệu dùng để chứa dữ liệu được các ứng dụng thường xuyên cập nhật. Bạn muốn có một cái cơ sở dữ liệu giống y hệt như thế trên một server khác để chạy báo cáo (report Database) (cách làm này thường dùng để tránh ảnh hưởng đến hiệu năng của server chính). Vấn đề là report server của bạn cũng cần phải được cập nhật thường xuyên để đảm bảo tính chính xác của các báo cáo. **Bạn không thể dùng cơ chế back up and restore trong trường hợp này.** Lúc

Chương 1. TỔNG QUAN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU SQL SERVER 2000

đó cơ chế replication của SQL Server sẽ được sử dụng để bảo đảm cho dữ liệu ở 2 cơ sở dữ liệu được đồng bộ.



Hình 1.2. Tổng quan Microsoft SQL Server

1.2.3 Data Transformation Service (DTS) - Dịch vụ chuyển dịch dữ liệu

Nếu bạn có dữ liệu ở các dạng khác nhau cụ thể như chứa trong Oracle, DB2 (của IBM), SQL Server, Microsoft Access..., bạn muốn chuyển toàn bộ dữ liệu này sang SQL Server. Công việc này được thực hiện dễ dàng bằng cách sử dụng dịch vụ DTS.

1.2.4 Analysis Service - Dịch vụ phân tích dữ liệu

Ta nhận thấy thực tế rằng, dữ liệu được lưu trữ rất nhiều, hết năm này đến năm khác, nhưng khi cần biết một “tri thức” nào từ đó thì không có. Do đó

Chương 1. TỔNG QUAN HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU SQL SERVER 2000

Microsoft cung cấp cho bạn một công cụ rất mạnh giúp cho việc phân tích dữ liệu trở nên dễ dàng và hiệu quả bằng cách dùng khái niệm hình khối nhiều chiều (multi-dimension cubes) và kỹ thuật "khai phá dữ liệu" -data mining.

1.2.5 English Query – Truy vấn dữ liệu sử dụng tiếng Anh

Đây là một dịch vụ giúp cho việc truy vấn dữ liệu bằng tiếng Anh "trơn" (plain English).

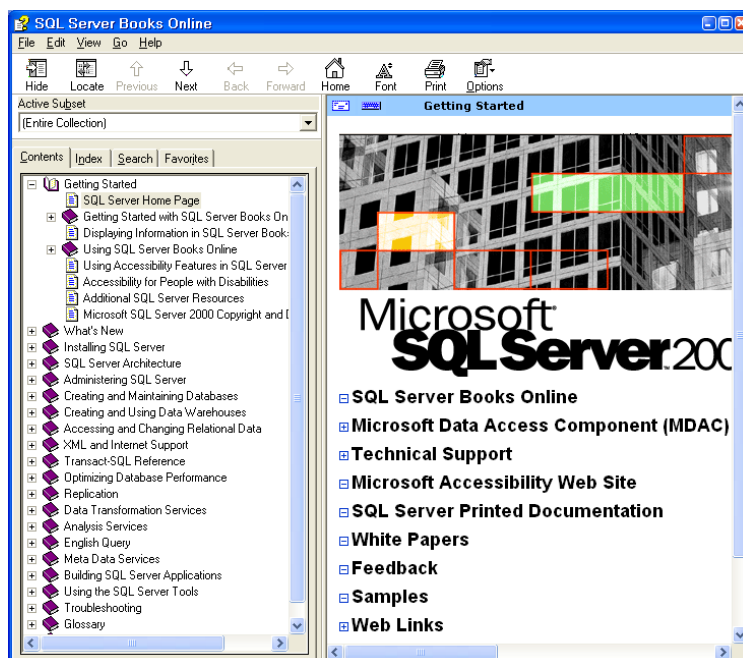
1.2.6 Meta Data Service

Dịch vụ này giúp cho việc chứa đựng và thao tác với **Meta Data** dễ dàng hơn.

Meta data là những thông tin mô tả về cấu trúc của dữ liệu trong cơ sở dữ liệu như dữ liệu. Bởi vì những thông tin này cũng được chứa trong cơ sở dữ liệu nên cũng là một dạng dữ liệu nhưng để phân biệt với dữ liệu "chính thống" người ta gọi nó là Meta Data.

1.2.7 SQL Server Books Online – Sách dạy SQL Server trực tuyến

Đây là cuốn sách trực tuyến được đính kèm khi cài đặt SQL Server. Nó là tài liệu không thể thiếu đối với những người muốn làm việc thực sự với hệ quản trị cơ sở dữ liệu này.



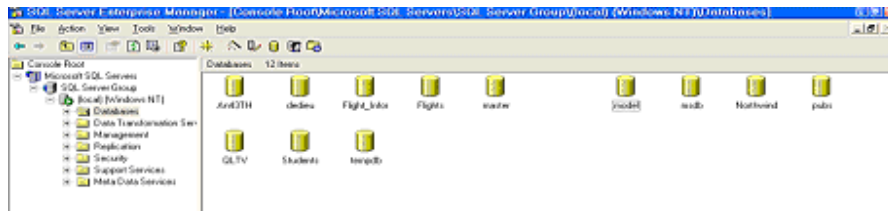
Hình 1.3. Cửa sổ hướng dẫn SQL Server (Book Online)

1.3 SQL Server Tools

Công cụ để quản trị SQL Server.

1.3.1 Enterprise Manager

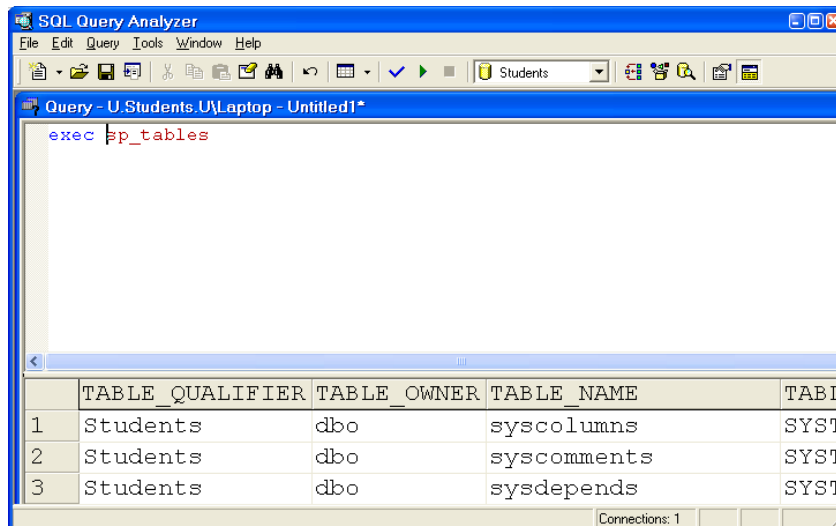
Đây là một công cụ cho ta thấy toàn cảnh hệ thống cơ sở dữ liệu một cách rất trực quan. Nó rất hữu ích đặc biệt cho người mới học và không thông thạo lắm về SQL.



Hình 1.4. Cửa sổ Enterprise Manager

1.3.2 Query Analyzer

Tiếp theo là Query Analyzer. Đối với một DBA giỏi thì hầu như chỉ cần công cụ này là có thể quản lý cả một hệ thống cơ sở dữ liệu mà không cần đến những thứ khác. Đây là một môi trường làm việc khá tốt vì ta có thể đánh bất kỳ câu lệnh SQL nào và chạy lập tức.



Hình 1.5. Cửa sổ Query Analyzer

1.3.3 SQL Profiler

Công cụ thứ ba cần phải kể đến là SQL Profiler. Nó có khả năng "chụp" (capture) tất cả các sự kiện hay hoạt động diễn ra trên một SQL server và lưu lại dưới dạng text file rất hữu dụng trong việc kiểm soát hoạt động của SQL Server.

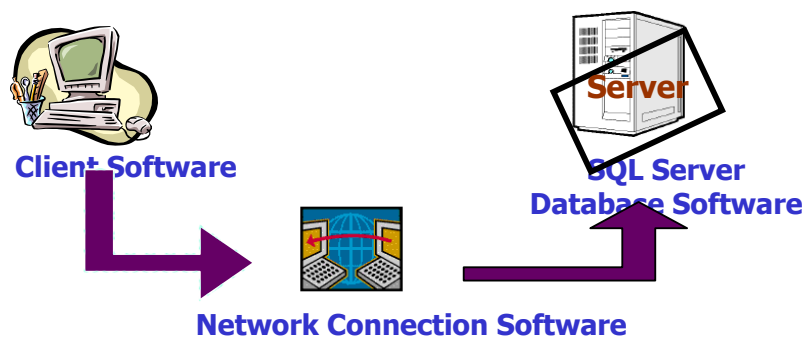
Ngoài một số công cụ trực quan như trên chúng ta cũng thường hay dùng osql và bcp (bulk copy) trong command prompt.

1.4 Kiến trúc của SQL Server

SQL Server được thiết kế để làm việc hiệu quả trên 2 môi trường:

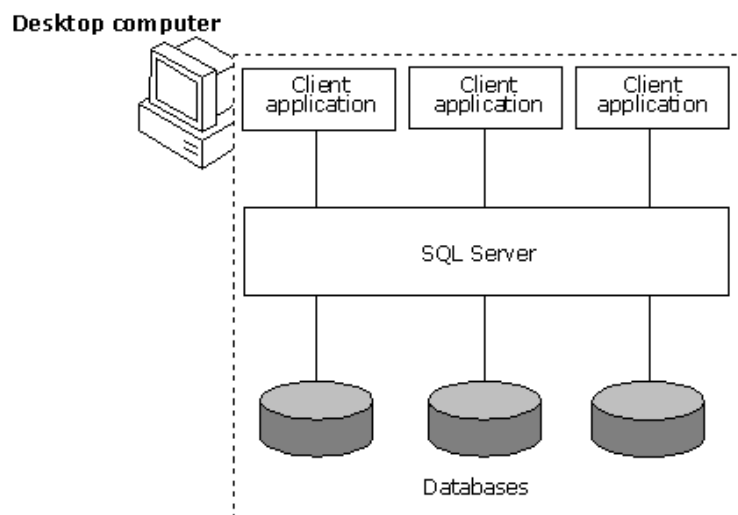
- Hệ thống cơ sở dữ liệu Client/Server (Client/Server Database system)
- Hệ thống cơ sở dữ liệu Desktop (Desktop Database system)

1.4.1 Hệ thống cơ sở dữ liệu Client/Server



Hình 1.6. Hệ thống cơ sở dữ liệu Client/Server

1.4.2 Hệ thống cơ sở dữ liệu Desktop

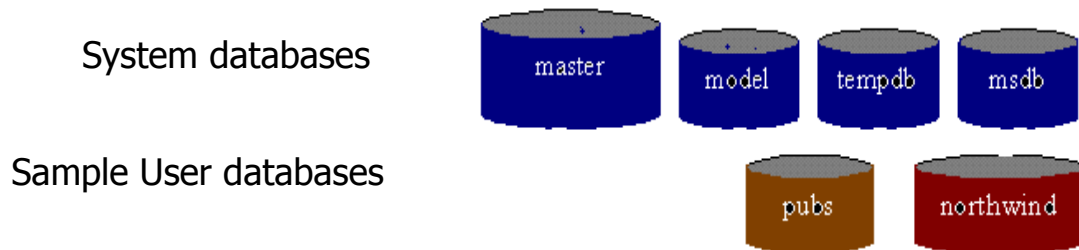


Hình 1.7. Hệ thống cơ sở dữ liệu Desktop

1.5 SQL Server Database

Ta đã biết, Database là tập hợp những dữ liệu được lưu trữ trong file nằm trên đĩa. Một Database sẽ chứa những file dữ liệu trong đó có những dữ liệu thực.

SQL Server có 2 kiểu Database:



Hình 1.7. Một số cơ sở dữ liệu hệ thống và cơ sở dữ liệu ví dụ

- System Databases
- User Databases

System Databases chứa những thông tin về SQL SERVER. SQL Server sử dụng System Databases để thực hiện và quản lý các Database người dùng (User Databases). System Databases và các 'sample user Databases' được tạo ra mặc định ngay khi cài đặt hệ thống.

1.6 Database Objects-Các đối tượng trong cơ sở dữ liệu

Tables: Các bảng chứa dữ liệu

Columns: Các cột trong bảng

Rows: Các hàng trong bảng

Data types: Các kiểu dữ liệu

Constraints: Các ràng buộc dữ liệu

Defaults: Giá trị mặc định của cột nào đó

Rules: Các luật được thiết đặt trên dữ liệu

Indexes: Chỉ số

Views: Các khung nhìn

Stored Procedures: Các thủ tục

Triggers

Các đối tượng sẽ được đề cập chi tiết trong những chương sau.

1.7 Câu hỏi trắc nghiệm

- 1. Thành phần nào sau đây cho phép người dùng quản lý các đối tượng trong SQL Server bằng đồ họa.**
 - A Service Manager
 - B Query Analyzer
 - C Enterprise Manager
 - D Book Online

- 2. Thành phần nào sau đây cung cấp giao diện đồ họa (GUI) cho phép người phát triển ứng dụng và người quản trị hệ thống có thể thực hiện những công việc hằng ngày như truy vấn tables, thao tác dữ liệu trong bảng một cách dễ dàng**
 - A Service Manager
 - B Query Analyzer
 - C Enterprise Manager
 - D Book Online

- 3. Có thể chuyển một cơ sở dữ liệu được xây dựng trên Microsoft Access sang SQL Server được không?**
 - A Có
 - B Không

- 4. Bạn có thể cài đặt phiên bản Enterprise trên hệ điều hành Windows 2000 Professional không?**
 - A Có
 - B Không

- 5. SQL Server 2000 có cho phép tạo bản sao của một cơ sở dữ liệu không?**
 - A Có
 - B Không

- 6. Khi cài đặt SQL Server 2000 trên WinNT Server 4, bạn cần cài đặt thêm ... để hỗ trợ?**

2 Chương 2. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU

2.1 Cấu trúc của SQL Server

Như đã trình bày ở các bài trước một trong những đặc điểm của SQL Server 2000 là **Multiple-Instance** nên khi nói đến một (SQL) Server nào đó là ta nói đến một Instance của SQL Server 2000, thông thường đó là Default Instance. Một Instance của SQL Server 2000 có 4 System Databases và một hay nhiều user Database. Các System Databases bao gồm:

Master: Chứa tất cả những thông tin cấp hệ thống (system-level information) bao gồm thông tin về các cơ sở dữ liệu khác trong hệ thống như vị trí của các data files, các login account và các thiết đặt cấu hình hệ thống của SQL Server (system configuration settings).

Tempdb: Chứa tất cả những table hay stored procedure được tạm thời tạo ra trong quá trình làm việc bởi user hay do bản thân SQL Server engine. Các table hay stored procedure này sẽ biến mất khi khởi động lại SQL Server hay khi ta disconnect.

Model: Cơ sở dữ liệu này đóng vai trò như một bảng tạm (template) cho các cơ sở dữ liệu khác. Nghĩa là khi một user Database được tạo ra thì SQL Server sẽ copy toàn bộ các system objects (tables, stored procedures...) từ Model Database sang Database mới vừa tạo.

Msdb: Cơ sở dữ liệu này được SQL Server Agent sử dụng để hoạch định các báo động và các công việc cần làm (schedule alerts and jobs).

2.2 Cấu trúc vật lý của một cơ sở dữ liệu SQL Server

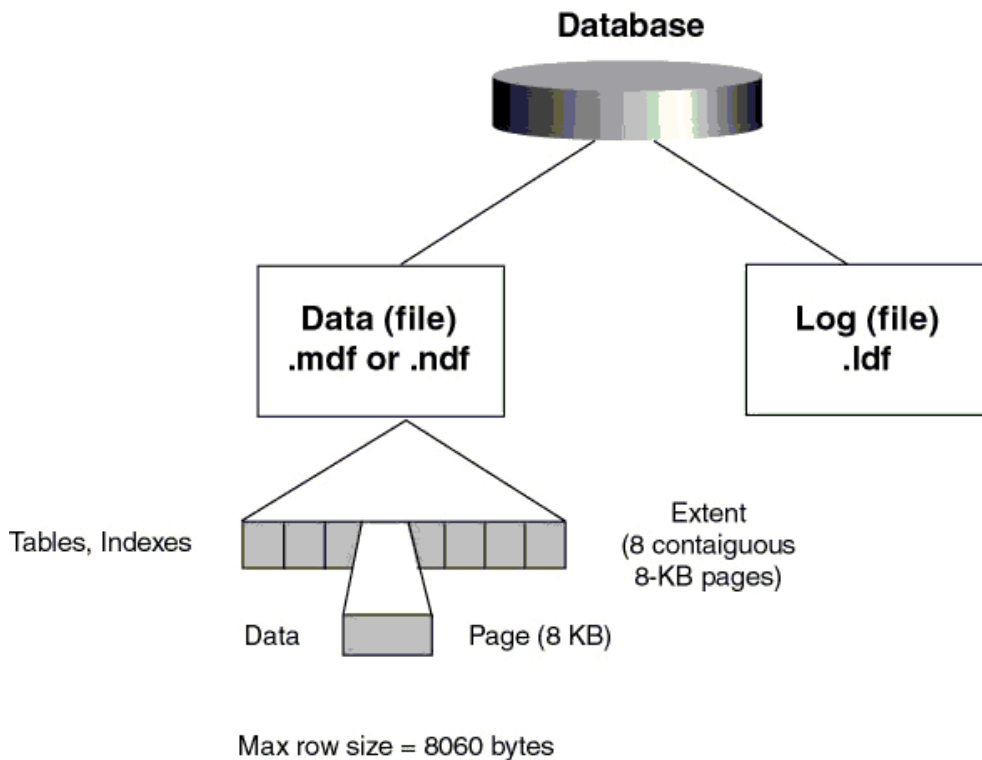
Mỗi một cơ sở dữ liệu trong SQL Server đều chứa ít nhất một data file chính (primary), có thể có thêm một hay nhiều data file phụ (Secondary) và một transaction log file.

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

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

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

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



Hình 2.1. Cấu trúc vật lý của một cơ sở dữ liệu SQL Server

Trước khi SQL Server muốn lưu dữ liệu vào một table nó cần phải dành riêng một khoảng trống trong data file cho table đó. Những khoảng trống đó chính là các extents.

Có 2 loại Extents:

Mixed Extents (loại hỗn hợp) dùng để chứa dữ liệu của nhiều tables trong cùng một Extent.

Uniform Extent (loại thuần nhất) dùng để chứa dữ liệu của một table. Đầu tiên SQL Server dành các Page trong Mixed Extent để chứa dữ liệu cho một table sau đó khi dữ liệu tăng trưởng thì SQL dành hẳn một Uniform Extent cho table đó.

2.3 Nguyên tắc hoạt động của transaction log trong SQL Server

Transaction log file trong SQL Server dùng để ghi lại các thay đổi xảy ra trong cơ sở dữ liệu.

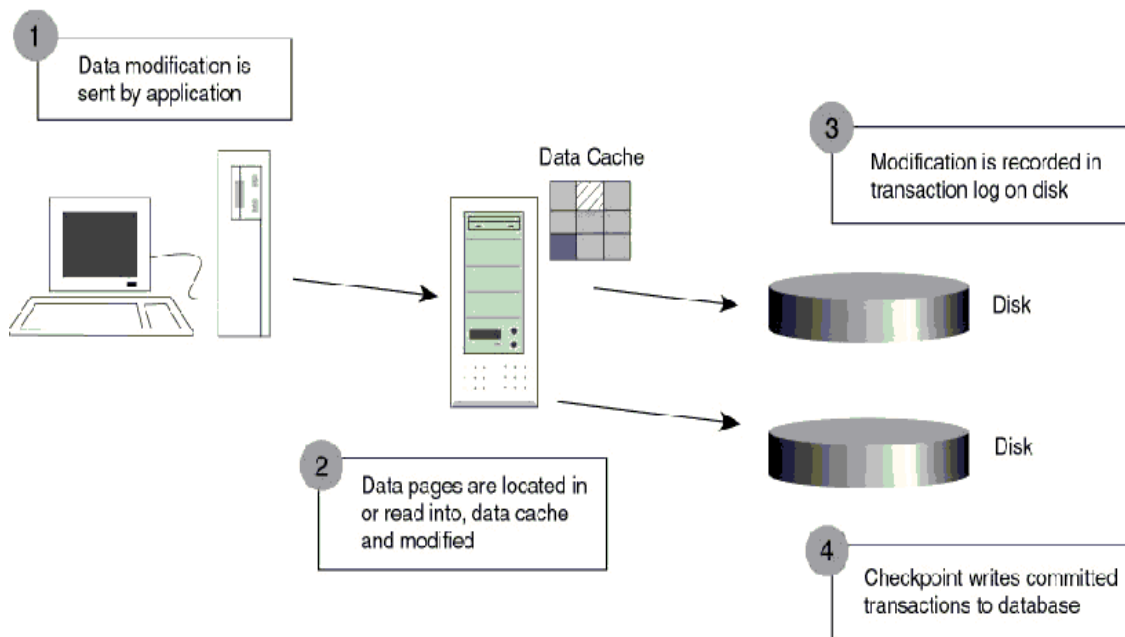
Quá trình này diễn ra như sau:

Đầu tiên khi có một sự thay đổi dữ liệu như Insert, Update, Delete được yêu cầu từ các ứng dụng, SQL Server sẽ tải (load) data page tương ứng lên memory

Chương 2. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU

(vùng bộ nhớ này gọi là data cache), sau đó dữ liệu trong data cache được thay đổi (những trang bị thay đổi còn gọi là *dirty-page*).

Tiếp theo mọi sự thay đổi đều được ghi vào transaction log file cho nên người ta gọi là *write-ahead* log. Cuối cùng thì một quá trình gọi là **Check Point Process** sẽ kiểm tra và viết tất cả những transaction đã được committed (hoàn tất) vào đĩa cứng (flushing the page).



Hình 2.2. Quá trình hoạt động của Transaction

Ngoài **Check Point Process** những *dirty-page* còn được đưa vào đĩa bởi một **Lazy writer**. Đây là một thành phần làm nhiệm vụ quét qua phần data cache theo một chu kỳ nhất định sau đó lại dừng để chờ lần quét tới.

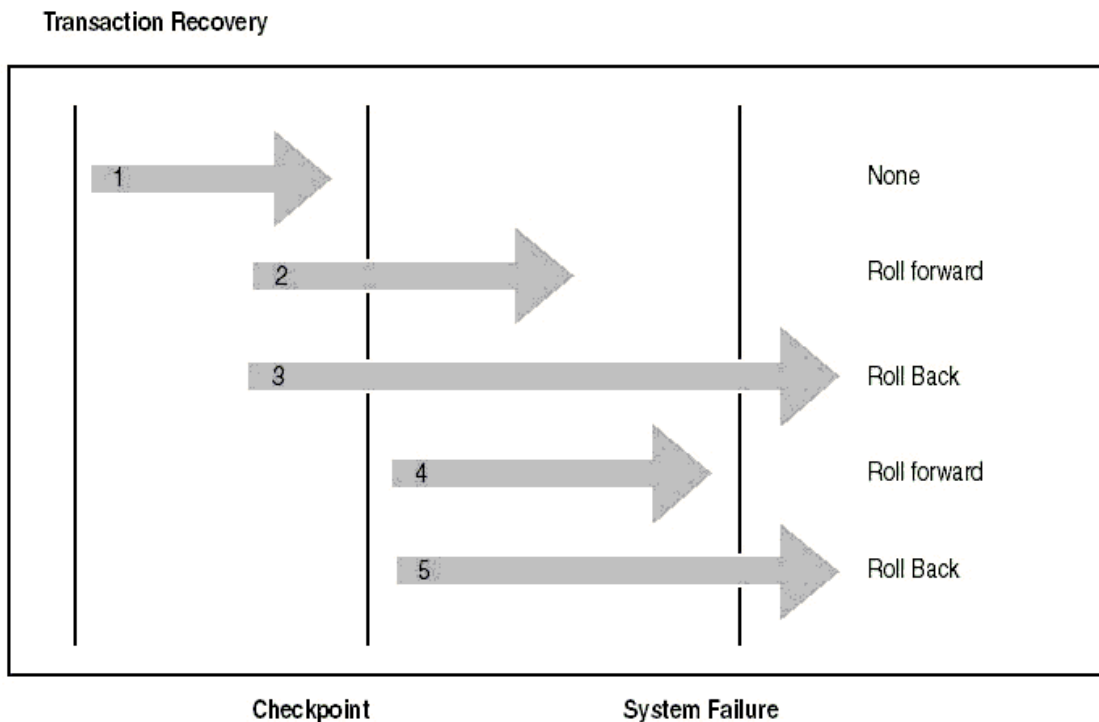
Check Point Process hoạt động như thế nào để có thể đảm bảo một transaction được thực thi mà không gây ra dữ liệu “bẩn”-dirty data.

Trong hình vẽ bên dưới (Transaction Recovery), một transaction được biểu diễn bằng một mũi tên. Trục nằm ngang là trục thời gian. Giả sử một Check Point được đánh dấu vào thời điểm giữa transaction 2 và 3 như hình vẽ và sau đó sự cố xảy ra trước khi gặp một Check point kế tiếp. Như vậy khi SQL Server được restart nó sẽ dựa trên những gì ghi trong transaction log file để phục hồi dữ liệu (xem hình vẽ).

Điều đó có nghĩa là SQL Server sẽ không cần làm gì cả đối với transaction 1 vì tại thời điểm Check point data đã được lưu vào đĩa rồi. Trong khi đó transaction 2 và 4 sẽ được Roll Forward vì tuy đã được committed nhưng do sự cố xảy ra trước

Chương 2. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU

thời điểm check point kế tiếp nên dữ liệu chưa kịp lưu vào đĩa. Tức là dựa trên những thông tin được ghi trên log file SQL Server hoàn toàn có đầy đủ cơ sở để viết vào đĩa cứng. Còn transaction 3 và 5 thì chưa được committed (do bị down bất ngờ) cho nên SQL Server sẽ Roll Back hai transaction này dựa trên những gì được ghi trên log file.



Hình 2.3. Khôi phục Transaction

2.4 Cấu trúc logic của một SQL Server Database

Hầu như mọi thứ trong SQL Server được tổ chức thành những objects ví dụ như tables, views, stored procedures, indexes, constraints.... Những system objects trong SQL Server thường có bắt đầu bằng chữ *sys* hay *sp*. Các objects trên sẽ được nghiên cứu lần lượt trong các bài sau.

2.5 Các kiểu dữ liệu trong SQL Server (data types)

2.5.1 Integers

2.5.1.1 Bigint

Là kiểu dữ liệu Integer có miền giá trị từ -2^{63} (-9223372036854775808) đến $2^{63}-1$ (9223372036854775807).

2.5.1.2 Int

Là kiểu dữ liệu Integer có miền giá trị từ -2^{31} (-2,147,483,648) đến $2^{31} - 1$ (2,147,483,647).

2.5.1.3 Smallint

Là kiểu dữ liệu Integer có miền giá trị từ 2^{15} (-32,768) đến $2^{15} - 1$ (32,767).

2.5.1.4 Tinyint

Là kiểu dữ liệu Integer có miền giá trị từ 0 đến 255.

2.5.1.5 Bit

Chỉ có một trong hai giá trị là 0 hoặc 1

2.5.2 Decimal and Numeric

2.5.2.1 Decimal

Độ chính xác được xác định và miền giá trị từ $-10^{38} + 1$ đến $10^{38} - 1$.

2.5.2.2 Numeric

Chức năng tương tự như decimal.

2.5.3 Money and Smallmoney

2.5.3.1 Money

Là kiểu dữ liệu tiền tệ có miền giá trị từ -2^{63} (-922,337,203,685,477.5808) đến $2^{63} - 1$ (+922,337,203,685,477.5807)

2.5.3.2 Smallmoney

Là kiểu dữ liệu tiền tệ có miền giá trị từ -214,748.3648 đến +214,748.3647

2.5.4 Approximate Numerics

2.5.4.1 Float

Độ chính xác (phần thập phân) thay đổi và miền giá trị từ $-1.79E + 308$ đến $1.79E + 308$.

2.5.4.2 Real

Độ chính xác (phần thập phân) thay đổi và miền giá trị từ $-3.40E + 38$ đến $3.40E + 38$.

2.5.5 Datetime and Smalldatetime

2.5.5.1 Datetime

Kiểu dữ liệu ngày/tháng từ January 1, 1753, tới December 31, 9999, với độ chính xác là 3/100 của second, hoặc 3.33 milliseconds.

2.5.5.2 Smalldatetime

Kiểu dữ liệu ngày/tháng từ January 1, 1900, tới June 6, 2079, với độ chính xác là 1 phút.

2.5.6 Character Strings

2.5.6.1 Char

Kiểu dữ liệu character có độ dài xác định và không theo mã Unicode, có khả năng lưu trữ tối đa 8,000 characters.

2.5.6.2 Varchar

Kiểu dữ liệu character có độ dài thay đổi và không theo mã Unicode, có khả năng lưu trữ tối đa 8,000 characters.

2.5.6.3 Text

Kiểu dữ liệu character có độ dài thay đổi và không theo mã Unicode, có khả năng lưu trữ tối đa $2^{31} - 1$ (2,147,483,647) characters.

2.5.7 Unicode Character Strings

2.5.7.1 Nchar

Kiểu dữ liệu character có độ dài xác định và theo mã Unicode, có khả năng lưu trữ tối đa 4,000 characters.

2.5.7.2 Nvarchar

Kiểu dữ liệu character có độ dài thay đổi và theo mã Unicode, có khả năng lưu trữ tối đa 4,000 characters.

2.5.7.3 Ntext

Kiểu dữ liệu character có độ dài thay đổi và theo mã Unicode, có khả năng lưu trữ tối đa $2^{30} - 1$ (1,073,741,823) characters.

2.5.8 Binary Strings

2.5.8.1 Binary

Kiểu dữ liệu Binary có độ dài xác định và khả năng lưu trữ tối đa 8,000 bytes.

2.5.8.2 Varbinary

Kiểu dữ liệu Binary có độ dài thay đổi và khả năng lưu trữ tối đa 8,000 bytes.

2.5.8.3 Image

Kiểu dữ liệu Binary có độ dài thay đổi và khả năng lưu trữ tối đa $2^{31} - 1$ (2,147,483,647) bytes.

2.5.9 Các kiểu dữ liệu khác

2.5.9.1 Cursor

Là một tham chiếu tới một con trỏ

2.5.9.2 Sql_variant

Là kiểu dữ liệu có khả năng lưu trữ rất nhiều kiểu dữ liệu khác nhau của SQL SERVER, ngoại trừ text, ntext, timestamp, and sql_variant.

2.5.9.3 Table

Là kiểu dữ liệu đặc biệt được sử dụng để lưu trữ tập kết quả của một quá trình xử lý.

2.5.9.4 Uniqueidentifier

Là kiểu dữ liệu có khả năng tự động cập nhật giá trị khi có 1 bản ghi được thêm mới (tương tự như kiểu dữ liệu Autounumber của Microsoft Access)

2.6 Câu hỏi trắc nghiệm

- 1. Cơ sở dữ liệu hệ thống ... được sử dụng như là template khi tạo tất cả các cơ sở dữ liệu mới.**
- 2. Một cơ sở dữ liệu trong SQL Server chứa ít nhất ...file?**
- 3. Các file chứa cơ sở dữ liệu thường được đặt ở đường dẫn ...?**
- 4. Một cơ sở dữ liệu muốn chuyển sang thực hiện trên máy tính khác có cài SQL Server, thông thường bạn phải copy đi những file ...?**
- 5. Những thay đổi trong một cơ sở dữ liệu được ghi lại ở file ...?**
- 6. Dữ liệu ở cột 1 được sử dụng để lưu trữ dữ liệu về sản phẩm của 1 cửa hàng, bạn hãy chọn kiểu dữ liệu tương ứng với nó ở cột 2.**

Cột 1	Đáp án	Cột 2
1. Tên sản phẩm	a.	Int
2. Hình ảnh mô tả	b.	Smallmoney
3. Giá sản phẩm	c.	Varchar
4. Hạn sử dụng	d.	Datetime
5. Số lượng tồn trong kho	e.	Image

3 Chương 3. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU- Phần thực hành

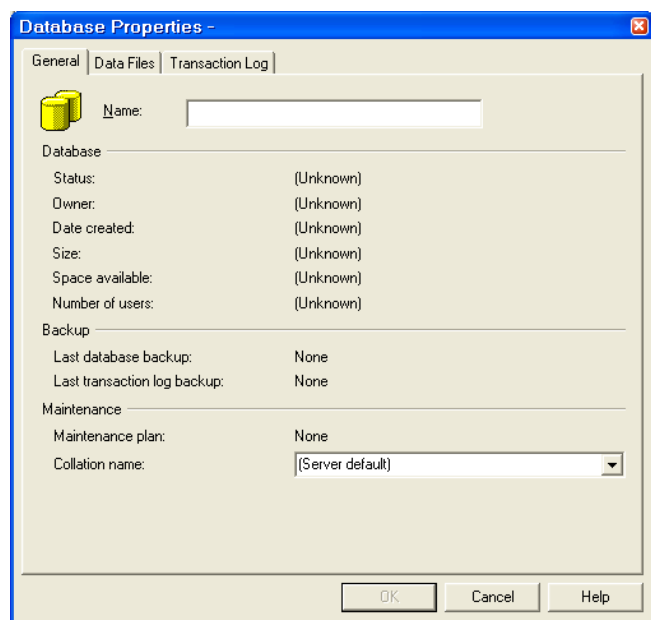
Mục đích:

- Sử dụng Enterprise Manager để tạo, sửa và xoá cơ sở dữ liệu (database)
- Tạo, sửa và xoá bảng (table)
- Thêm, sửa, xoá dữ liệu trong các bảng
- Xem thông tin của bảng

3.1 Tạo cơ sở dữ liệu sử dụng Enterprise Manager

Trong Enterprise Manager, chúng ta có thể tạo cơ sở dữ liệu trực tiếp hoặc sử dụng hỗ trợ Wizard. Sau đây là cách tạo trực tiếp:

1. Khởi động Service Manager (thao tác này để khởi động SQL SERVER).
2. Kích chọn Enterprise Manager trong thanh menu của Microsoft SQL Server.
3. Chọn Server chứa cơ sở dữ liệu
4. Kích chọn Action/New/Database từ menu Action
5. Nhập tên cơ sở dữ liệu (Ví dụ: Flights)

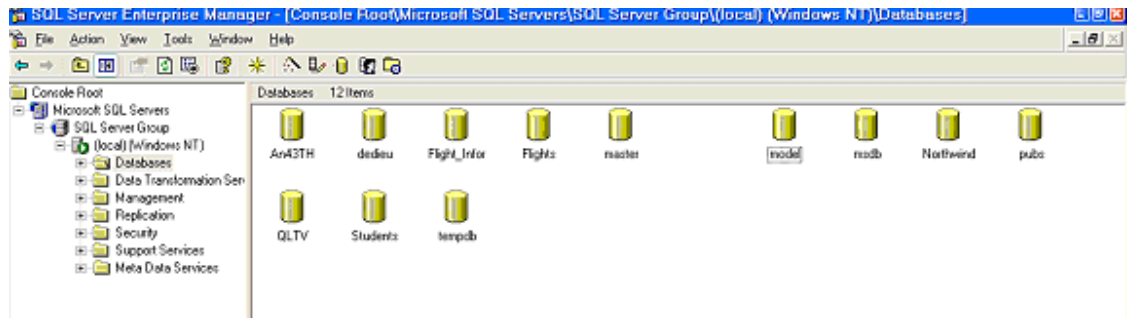


Hình 3.1

6. Kích OK

Chương 3. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU –Phần thực hành

7. Kích đúp vào đối tượng Database trong cửa sổ bên phải, chúng ta sẽ nhìn thấy cơ sở dữ liệu Flights vừa được tạo

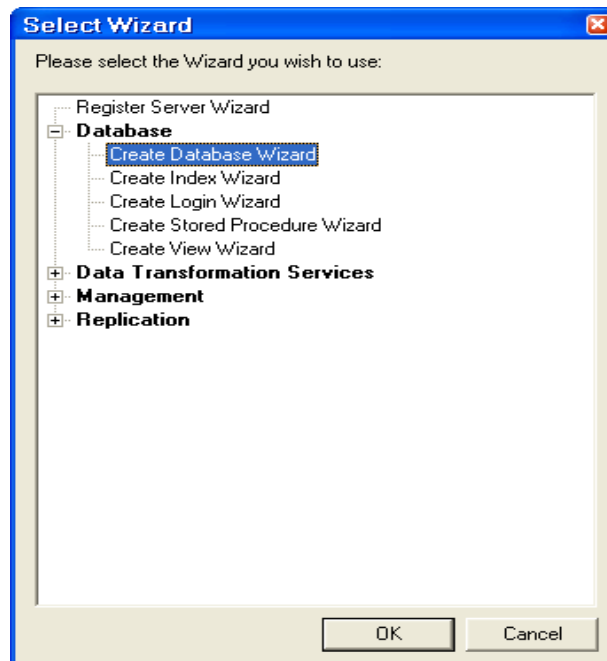


Hình 3.2

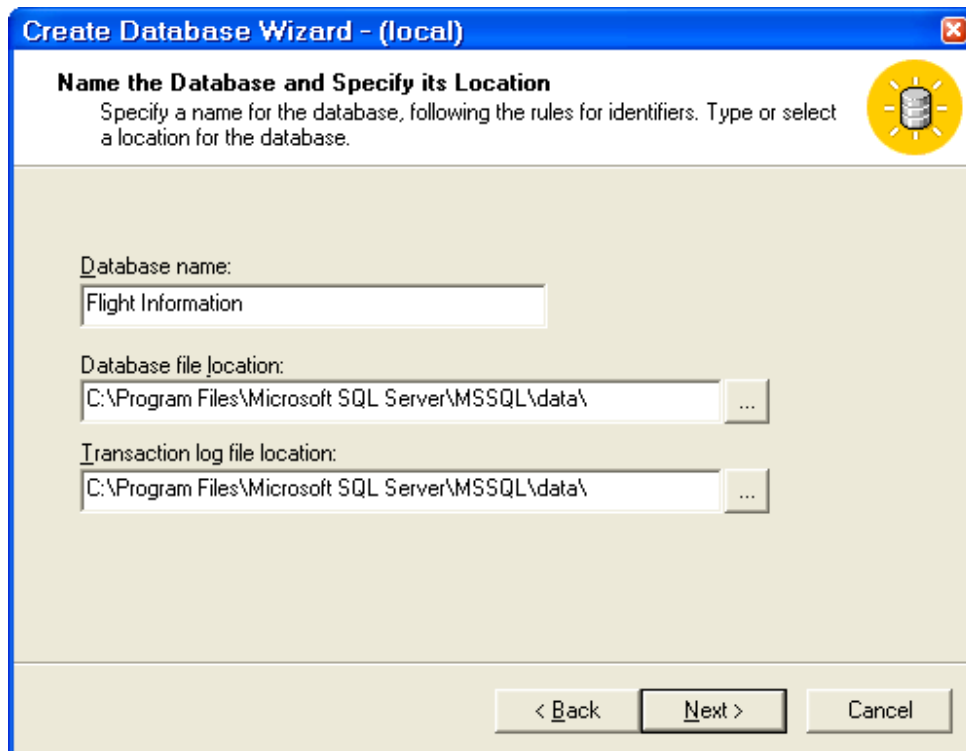
SQL Server hỗ trợ thực hiện Wizard cho một vài công việc chung.

Sau đây là cách tạo cơ sở dữ liệu thực hiện Wizard:

1. Kích chọn Tools/Wizard... từ menu Tool trên menu bar của Enterprise Manager.
2. Chọn Create Database Wizard (Hình 3.3)
3. Kích OK
4. Kích Next
5. Nhập tên cơ sở dữ liệu (ví dụ: Flight Information)
6. Kích Next

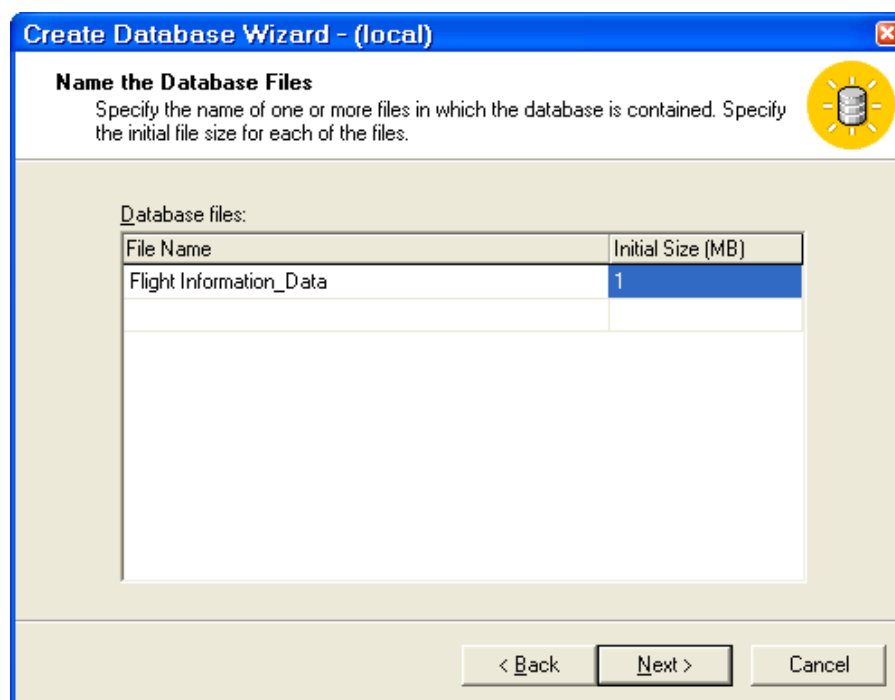


Hình 3.3



Hình 3.4

7. Xuất hiện cửa sổ cho phép nhập tên file chứa cơ sở dữ liệu và kích thước tối đa của file.



Hình 3.5

8. Kích Next. Xuất hiện cửa sổ cho phép tăng kích thước cơ sở dữ liệu khi nó quá lớn và giới hạn kích thước
9. Kích Next. Xuất hiện cửa sổ cho phép thay đổi tên và kích thước của file log (ghi lại lịch sử) của cơ sở dữ liệu
10. Kích Next. Xuất hiện cửa sổ cho phép tăng kích thước log file khi nó quá lớn và giới hạn kích thước.
11. Kích Next



Hình 3.6

12. Kích Finish để hoàn thành.

3.2 Sửa cơ sở dữ liệu

Sau khi tạo cơ sở dữ liệu, chúng ta có thể thay đổi định nghĩa ban đầu. Các loại thay đổi:

- Mở rộng, co hẹp kích thước data file hoặc log file.
- Tạo filegroups
- Thay đổi tên cơ sở dữ liệu

3.3 Xoá cơ sở dữ liệu

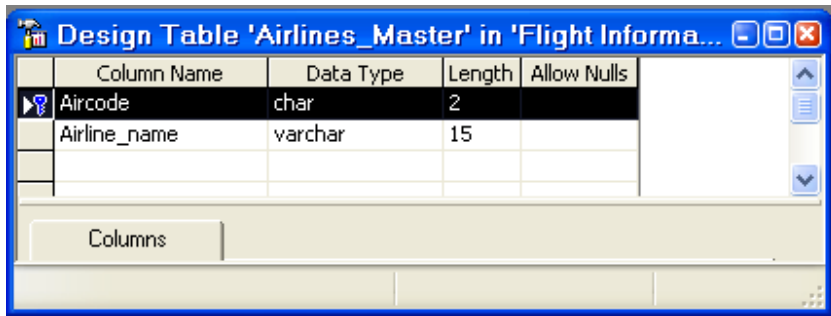
- Chọn cơ sở dữ liệu cần xoá
- Sau đó Delete.

3.4 Nghiên cứu cơ sở dữ liệu Flight_Information

3.4.1 Cấu trúc bảng

1. Bảng Airlines_Master: Chi tiết của các Airlines

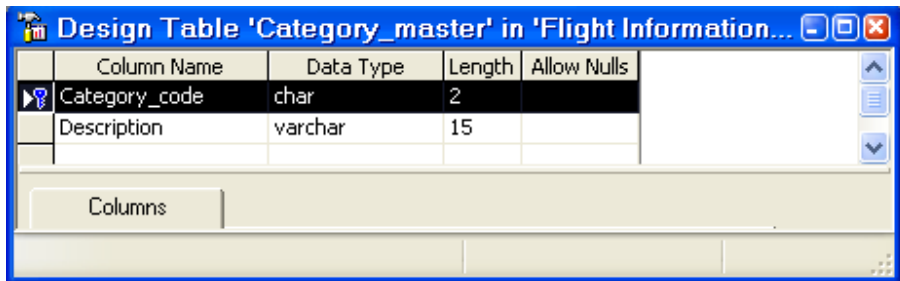
Chương 3. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU –Phần thực hành



Design Table 'Airlines_Master' in 'Flight Informa...'

Column Name	Data Type	Length	Allow Nulls
Aircode	char	2	
Airline_name	varchar	15	

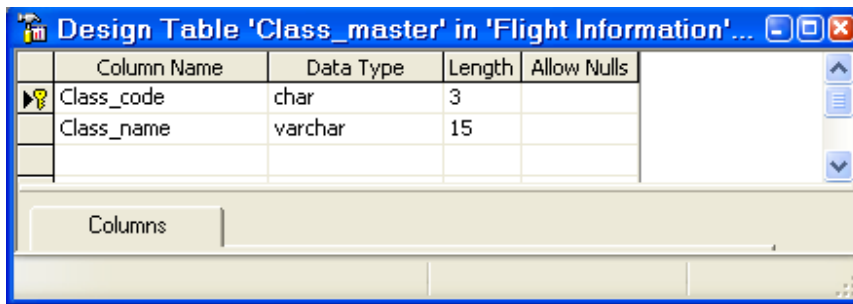
2. Bảng Category_master: Danh mục các Airlines



Design Table 'Category_master' in 'Flight Information...'

Column Name	Data Type	Length	Allow Nulls
Category_code	char	2	
Description	varchar	15	

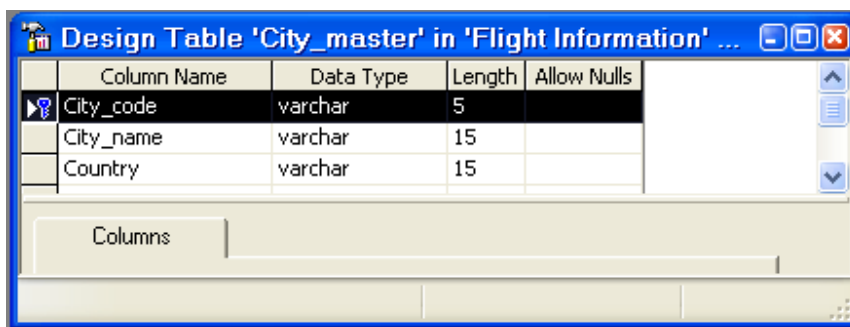
3. Bảng Class_Master: Các mức(hạng vé) airline cung cấp



Design Table 'Class_master' in 'Flight Information...'

Column Name	Data Type	Length	Allow Nulls
Class_code	char	3	
Class_name	varchar	15	

4. Bảng City_Master: Các thành phố đi và đến



Design Table 'City_master' in 'Flight Information...'

Column Name	Data Type	Length	Allow Nulls
City_code	varchar	5	
City_name	varchar	15	
Country	varchar	15	

5. Bảng Day_Master: Các ngày có thể phục vụ

Chương 3. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU –Phần thực hành

Design Table 'Day_master' in 'Flight Informati...'

Column Name	Data Type	Length	Allow Nulls
Day_code	int	4	
Day_name	varchar	15	

6. Bảng Meal: Các thức ăn có thể lựa chọn

Design Table 'Meal' in 'Flight Information' on '(lo...'

Column Name	Data Type	Length	Allow Nulls
Meal_code	varchar	5	
Meal_name	varchar	30	

7. Bảng Service: Các dịch vụ được cung cấp

Design Table 'Service' in 'Flight Information' ...

Column Name	Data Type	Length	Allow Nulls
Service_code	varchar	3	
Service_name	varchar	20	

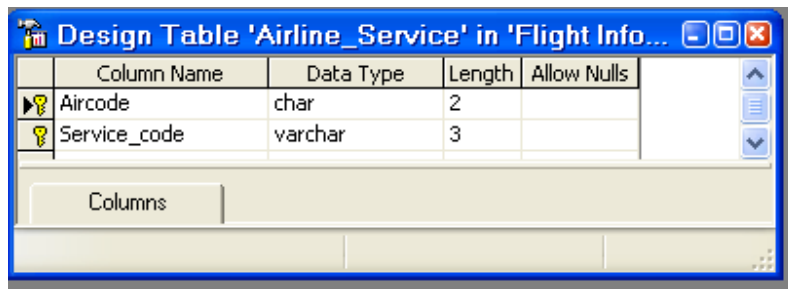
8. Bảng Status_Master: Tình trạng của vé máy bay

Design Table 'Status_master' in 'Flight Informati...'

Column Name	Data Type	Length	Allow Nulls
Status_code	varchar	3	
Description	varchar	15	

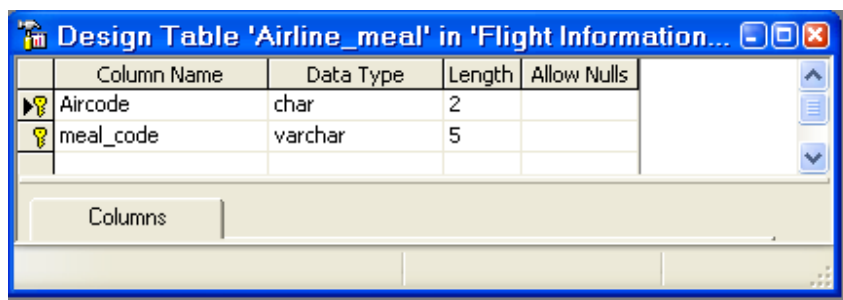
9. Airline_Service: Dịch vụ được cung cấp trên mỗi hãng

Chương 3. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU –Phần thực hành



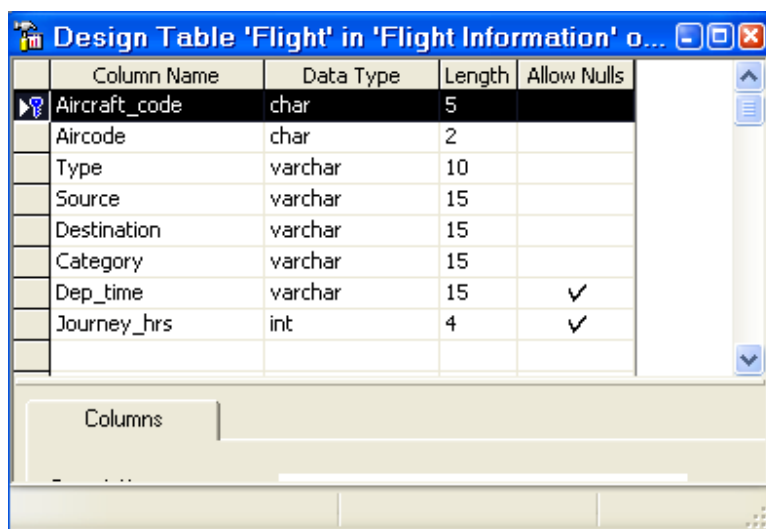
Column Name	Data Type	Length	Allow Nulls
Aircode	char	2	
Service_code	varchar	3	

10. Airline_Meal: Các thức ăn được phục vụ trên mỗi hãng



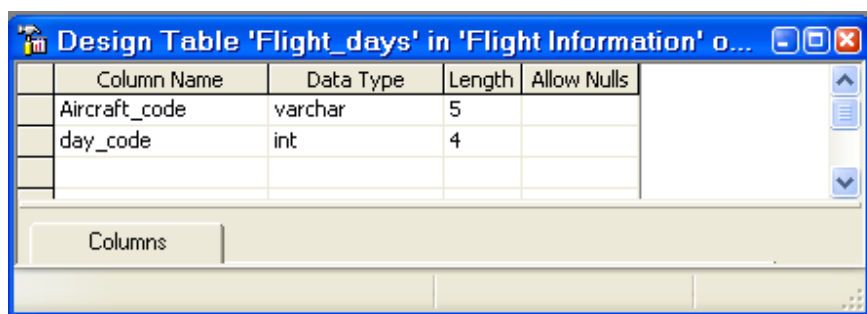
Column Name	Data Type	Length	Allow Nulls
Aircode	char	2	
meal_code	varchar	5	

11. Flight: Các chuyến bay của các hãng



Column Name	Data Type	Length	Allow Nulls
Aircraft_code	char	5	
Aircode	char	2	
Type	varchar	10	
Source	varchar	15	
Destination	varchar	15	
Category	varchar	15	
Dep_time	varchar	15	✓
Journey_hrs	int	4	✓

12. Flight_days: Ngày mỗi chuyến bay có thể phục vụ



Column Name	Data Type	Length	Allow Nulls
Aircraft_code	varchar	5	
day_code	int	4	

13. Flight_details: Chi tiết của mỗi chuyến bay

Column Name	Data Type	Length	Allow Nulls
Aircraft_code	varchar	5	
Class_code	char	2	
Fare	numeric	5	
Seats	numeric	5	

14. Passenger: Chi tiết về các khách hàng

Column Name	Data Type	Length	Allow Nulls
PNR_no	numeric	5	
Ticket_no	numeric	5	
Name	varchar	15	
Age	int	4	
Sex	char	10	
[PP No]	varchar	20	
[Meal Pref]	varchar	20	<input checked="" type="checkbox"/>

15. Reservation: Chi tiết về việc đặt vé máy bay

Column Name	Data Type	Length	Allow Nulls
PNR_no	numeric	5	
Aircraft_code	varchar	5	
Journey_date	datetime	8	
Class_code	char	3	
No_of_seats	bigint	8	
Address	varchar	50	
Contact_no	numeric	5	
Status	char	3	

3.4.2 Dữ liệu trên các bảng

1. Bảng Airlines_Master

Chương 3. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU –Phần thực hành

Aircode	Airline_name
BW	Jet Airways
AI	Air India
BA	British Airways
IA	Indian Airlines
*	

2. Bảng Category_Master

Category_code	Description
D	Domestic
I	International
*	

3. Bảng Class_Master

Class_code	Class_name
E	Economy
Ex	Executive
FC	First Class
*	

4. Bảng City_Master

City_code	City_name	Country
Bang	Bangalore	India
Cal	Calcutta	India
Che	Chennai	India
Del	Delhi	India
Lon	London	England
Mum	Mumbai	India
NY	New York	USA
*		

5. Bảng Day_Master

Day_code	Day_name
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday
*	

6. Bảng Meal

Meal_code	Meal_name
CNV	Chinese Non-Veget
CONNV	Continental Non-Ve
CONV	Continental Vegeta
CV	Chinese Vegetarian
NV	Non-Vegetarian
V	Vegetarian
*	

7. Bảng Service

Chương 3. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU –Phần thực hành

	Service_code	Service_name
▶	CC	Child Care
	N	Nurse
	WC	Wheel Chair
*		

8. Bảng Status_Master

	Status_code	Description
▶	C	Cancelled
	R	Reserved
	T	Travelled
	U	Untravelled
	W	Waitlist
*		

9. Bảng Airline_Service

	Aircode	Service_code
▶	9W	CC
	9W	N
	9W	WC
	AI	CC
	AI	N
	AI	WC
	BA	CC
	BA	N
	BA	WC
	IA	CC
	IA	N
	IA	WC
*		

10. Bảng Airline_Meal

	Aircode	meal_code
▶	9W	V
	9W	NV
	AI	V
	AI	NV
	AI	V
	AI	CV
	AI	CNV
	AI	CONV
	AI	CONNV
	BA	V
	BA	NV
	BA	CV
	BA	CNV
	BA	CONV
	BA	CONNV
	IA	V
	IA	NV
*		

11. Bảng Flight

Chương 3. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU –Phần thực hành

Aircraft_code	Aircode	Type	Source	Destination	Category	Dep_time	Journey_hrs
▶ 9W01	9W	Airbus	Cal	Che	D	15.30	3
9W02	9W	Airbus	Mum	Del	D	9.30	2
9W03	9W	Airbus	Che	Mum	D	10.45	2
AI01	AI	Boeing	Cal	NY	I	2.20	18
AI02	AI	Boeing	Mum	NY	I	1.30	20
AI03	AI	Boeing	Del	NY	I	3.00	17
BA01	BA	Boeing	Mum	Lon	I	23.30	9
BA02	BA	Boeing	Del	Lon	I	2.15	11
BA03	BA	Boeing	Mum	NY	I	3.30	20
IC01	IA	Airbus	Bang	Che	D	10.45	1
IC02	IA	Airbus	Bang	Mum	D	14.00	2
IC03	IA	Boeing	Del	Cal	D	17.45	1
IC04	IA	Boeing	Mum	Cal	D	18.30	2
*							

12. Bảng Flight_days

Aircraft_code	day_code
▶ 9W01	1
9W02	2
AI01	2
AI02	1
AI03	2
BA01	2
BA02	1
BA03	2
IC01	1
IC02	3
IC03	1
IC04	2
9W01	3
9W02	4
9W03	4
AI01	4
AI02	3
AI03	4
BA01	4
BA02	3
BA03	4
IC01	2
IC02	5
IC03	3
IC04	4
9W01	5
9W03	6
AI01	7
AI02	5
AI03	6
BA01	7
BA02	5
BA03	7
IC01	5
IC02	6
IC03	5
IC04	6
9W02	6
AI02	6
AI03	7
IC03	6
IC05	1

Chương 3. THIẾT KẾ VÀ THỰC THI CƠ SỞ DỮ LIỆU –Phần thực hành

13. Bảng Flight_details

	Aircraft_code	Class_code	Fare	Seats
▶	9W01	E	6100	300
	9W01	Ex	6895	150
	9W02	E	5900	300
	9W02	Ex	6850	150
	9W03	E	4175	300
	9W03	Ex	5400	150
	AI01	E	72215	350
	AI01	Ex	77145	175
	AI01	FC	81215	100
	AI02	E	60100	350
	AI02	Ex	70150	175
	AI02	FC	75540	100
	AI03	E	6100	300
	AI03	Ex	6100	300
	AI03	FC	6100	300
	BA01	E	63215	350
	BA01	Ex	69145	175
	BA01	FC	73215	100
	BA02	E	66130	350
	BA02	Ex	72190	175
	BA02	FC	78540	100
	BA03	E	62350	350
	BA03	Ex	63100	175
	BA03	FC	64150	100
	IC01	E	3409	300
	IC01	Ex	5117	150
	IC02	E	4000	300
	IC02	Ex	4600	150
	IC03	E	5048	425
	IC03	Ex	7574	125
	IC04	E	5817	425
	IC04	Ex	5861	125
*				

14. Bảng Passenger

	PNR_no	Ticket_no	Name	Age	Sex	PP No	Meal Pref
▶	1	1	Allen Smith	Use 'Group By'	Male	117889	CONNV
	1	2	Stella Smith	23	Female	322901	V
	1	3	Pam Smith	26	Female	279011	NV
	2	4	Peter Jones	27	Male	293211	NV
	2	5	Lily Jones	30	Female	347821	V
	3	6	James Crawford	44	Male	123111	NV
	3	7	Kitty Crawford	30	Female	1237112	CONNV
	4	8	Alex Lee	22	Male	40103	V
	4	9	Greta Lee	20	Female	41312	NV
	4	10	Shania Lee	30	Female	34784	NV
	4	10	Christopher Lee	30	Female	12453	NV
	3	7	Stella Morris	27	Female	31567	V
*							

15. Bảng Reservation

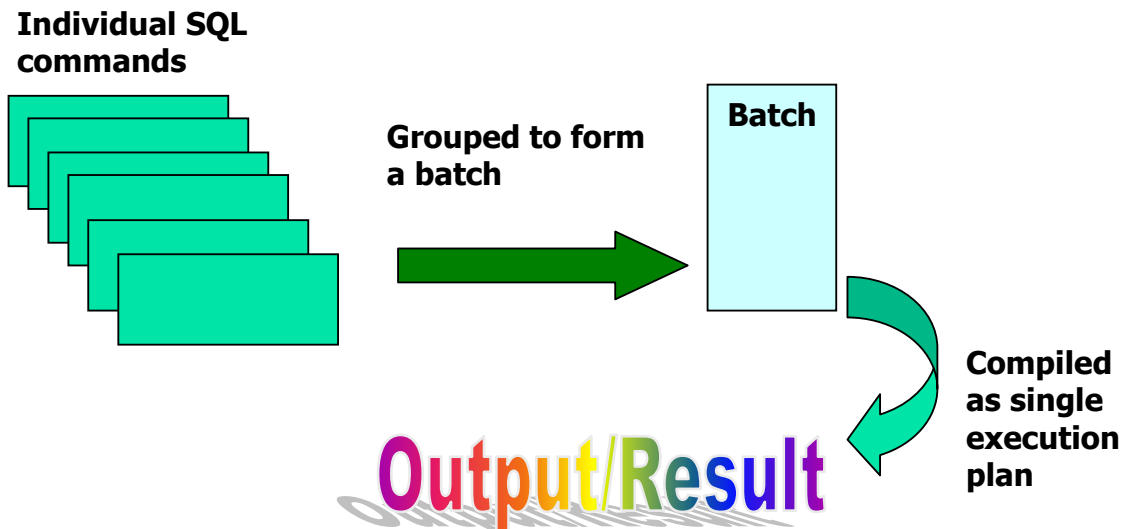
	PNR_no	Aircraft_code	Journey_date	Class_code	No_of_seats	Address	Contact_no	Status
▶	1	IC01	3/15/2001	Ex	2	25, Temple street,	3317575	C
	2	BA02	5/15/2001	FC	3	Pushpa Vihar, 6131	6453892	C
	3	IC01	3/15/2001	Ex	2	11A, Ellite Auto hot	3214344	C
	4	BA02	3/21/2001	E	4	Videocon Bldg, And	9430243	R
	5	IC01	5/15/2001	FC	1	Nariman Point, Mun	2122122	C
*								

3.5 Bài tập

1. Tạo cơ sở dữ liệu **Flight Informtion**
2. Thêm dữ liệu vào các bảng (như trên)
3. Chọn Query Analyzer để thực hiện một số yêu cầu sau:
 - a. Tăng giá tất cả các chuyến bay của hãng 'IC01' lên 10%
 - b. Hãng AI (Air India) quyết định không phục vụ thức ăn có meal_code= 'CONV'. Hãy xoá thông tin liên quan.
 - c. Xoá cột Type của bảng Flight
 - d. Xóa bảng Status_Master trong cơ sở dữ liệu
 - e. Cập nhật lại cột Sex trong bảng Passanger: Nếu là F thì sửa lại Females, nếu là M thì sửa lại là Males.

4 Chương 4. T-SQL PROGRAMING

4.1 Giới thiệu SQL Batch Processing



Hình 4.1. Cách thực hiện của nhóm lệnh (Batches)

4.1.1 Cách thực Thi một nhóm lệnh (Batches)

Khi thực thi một nhóm lệnh SQL Server sẽ phân tích và tìm biện pháp tối ưu cho các câu lệnh như một câu lệnh đơn và chứa execution plan đã được biên dịch (compiled) trong bộ nhớ sau đó nếu nhóm lệnh trên được gọi lại lần nữa thì SQL Server không cần biên dịch mà có thể thực thi ngay điều này giúp cho một batch chạy nhanh hơn.

Những câu lệnh nằm trong 1 batch sẽ được xử lý cùng lúc. Nếu một trong những câu lệnh của Batch không thực hiện được thì Batch cũng không được thực hiện

4.1.2 Lệnh GO

Lệnh này chỉ dùng để gửi một tín hiệu cho SQL Server biết đã kết thúc một batch job và yêu cầu thực thi. Nó vốn không phải là một lệnh trong T-SQL.

4.1.3 Ví dụ về Batch:

```
Use Pubs
Select * from authors
Update authors
set phone= '890 451-7366'
where au_lname= 'White'
GO
```

4.1.4 Chú thích (comment) trong batch:

- Là chuỗi ký tự trong chương trình được compiler bỏ qua.
- Sử dụng để chú thích hoặc tạm thời bỏ qua những phần trong chương trình đang được cân nhắc
- Thực hiện bảo trì chương trình nguồn dễ dàng
- Chú thích thường được dùng để ghi lại tên chương trình, tên tác giả, ngày của những thay đổi chính trong chương trình
- Chú thích có thể sử dụng biểu diễn những tính toán phức tạp và những giải thích trong phương pháp lập trình
- Các kiểu của Comment: SQL dùng dấu `--` để đánh dấu phần chú thích cho câu lệnh đơn và dùng `/*...*/` để chú thích cho một nhóm.

Ví dụ:

```
USE Northwind
GO
-- First line of a multiple-line comment.
SELECT * FROM Employees /*This is a comment*/
```

4.2 Câu lệnh điều khiển

Chúng ta có thể sử dụng các câu lệnh điều khiển khi lập trình cùng với T-SQL.

4.2.1 Begin..End

Cú pháp:

```
BEGIN
{
    statement | statement_block
}
END
```

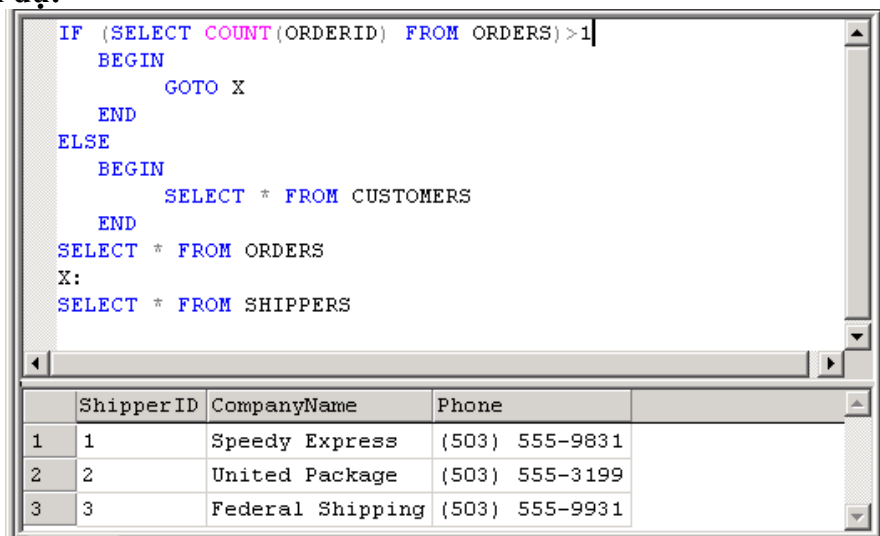
4.2.2 If..Else

IF...ELSE: Rẽ nhánh theo kết quả của biểu thức Logic.

Cú pháp:

```
IF Boolean_expression
{ sql_statement | statement_block }
[ ELSE
{ SQL SERVER_statement | statement_block } ]
```

Ví dụ:



```
IF (SELECT COUNT(ORDERID) FROM ORDERS)>1
BEGIN
    GOTO X
END
ELSE
BEGIN
    SELECT * FROM CUSTOMERS
END
SELECT * FROM ORDERS
X:
SELECT * FROM SHIPPERS
```

	ShipperID	CompanyName	Phone
1	1	Speedy Express	(503) 555-9831
2	2	United Package	(503) 555-3199
3	3	Federal Shipping	(503) 555-9931

Hình 4.1

4.2.3 Vòng lặp While

WHILE: Chúng ta có thể thực hiện câu lệnh SQL hoặc khối các lệnh dựa trên một số điều kiện. Các câu lệnh trong While được thực hiện lặp đi lại trong khi biểu thức Logic còn đúng.

Cú pháp:

```
WHILE Boolean_expression
{ statement | statement_block }
[ BREAK ]
{ statement | statement_block }
[ CONTINUE ]
```

Sử dụng BREAK và CONTINUE:

Chúng ta có thể sử dụng từ khóa CONTINUE hoặc BREAK trong vòng lặp WHILE để điều khiển việc thực hiện các câu lệnh.

Ví dụ:

Chương 4. T-SQL PROGRAMING

```
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'
```

4.2.4 Từ khoá GOTO

Chúng ta có thể thay đổi được thứ tự thực hiện bằng việc xác định vị trí (Label).

Cú pháp:

```
GOTO label
```

4.2.5 Từ khoá Return

Chúng ta có thể sử dụng RETURN ở bất kỳ vị trí nào để thoát ra khỏi khối, thủ tục. Những câu lệnh sau lệnh Return không được thực hiện

Cú pháp:

```
RETURN [ integer expression ]
```

4.2.6 Câu lệnh CASE

Từ khóa Case trả lại giá trị dựa trên kết quả của biểu thức logic.

Nó có thể được sử dụng ở bất kỳ vị trí nào phép toán được phép.

Cú pháp:

```
CASE expression
    WHEN expression1 THEN expression1
    [[WHEN expression2 THEN expression2] [...]]
    [ELSE expression]
END
```

Ví dụ 1:

```
SELECT au_fname, au_lname, CASE state
    WHEN 'OR' THEN 'Oregon'
END AS StateName FROM authors
```

Ví dụ 2:

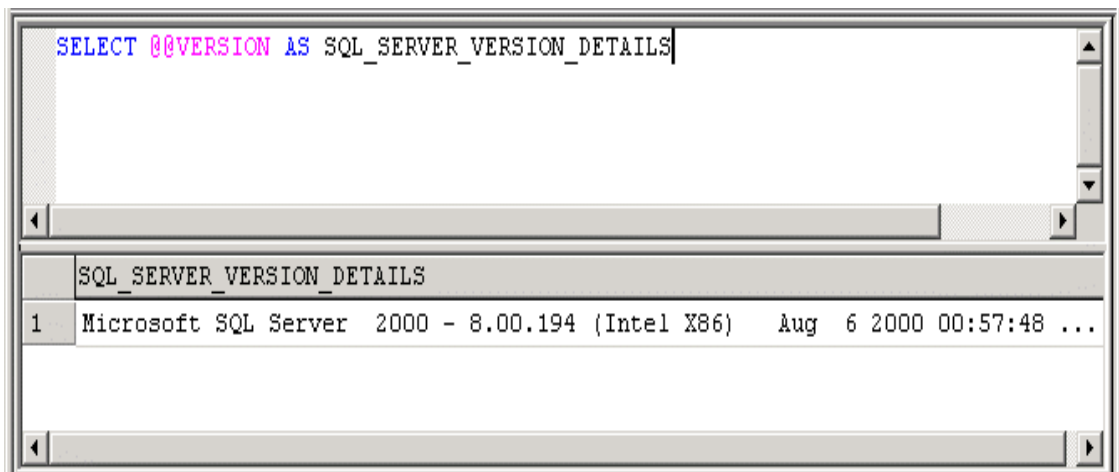
Chương 4. T-SQL PROGRAMING

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

4.3 Biến(Variables)

SQL Server hỗ trợ 2 loại biến:

- **Grobal variables**-Biến địa phương
- **Local variables**- Biến toàn cục



Hình 4.2. Sử dụng biến toàn cục

4.3.1 Grobal variables

Biến toàn cục trong SQL Server được bắt đầu bằng 2 ký hiệu `@`. Chúng ta có thể truy cập giá trị của bất kỳ biến nào cùng với câu lệnh `SELECT` đơn giản.

Danh sách các biến toàn cục:

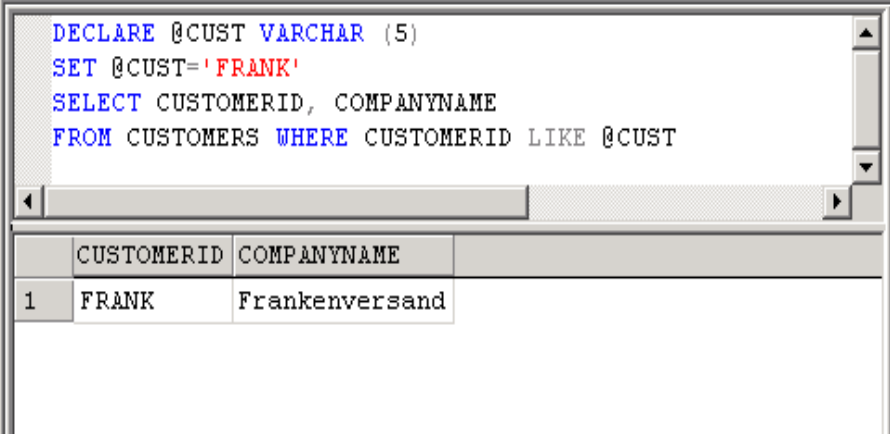
Tên biến	Ý nghĩa
<code>@@CONNECTIONS</code>	Số lượng những kết nối với Server từ khi Server

Chương 4. T-SQL PROGRAMING

	khởi động lại
@@CPU_BUSY	Thời gian (tính theo milliseconds) của hệ thống tính từ khi SQL Server khởi động
@@CURSOR_ROWS	Số lượng các dòng dữ liệu của con trỏ mới được mở gần đây nhất
@@DATEFIRST	Giá trị hiện tại của biến SET DATEFIRST – chỉ ra ngày đầu tiên của
@@ERROR	Mã số lỗi của câu lệnh T-SQL cuối cùng bị lỗi
@@FETCH_STATUS	Trạng thái truy nhập con trỏ: 0 nếu trạng thái truy nhập cuối cùng thành công; -1 nếu có lỗi
@@IDENTITY	Giá trị xác định (identity) cuối cùng được thêm vào
@@LANGUAGE	Tên ngôn ngữ hiện tại đang sử dụng
@@MAX_CONNECTIONS	Số lượng tối đa các kết nối có thể thực hiện đồng thời
@@ROWCOUNT	Số lượng các dòng kết quả của câu lệnh SQL gần đây nhất
@@SERVERNAME	Tên của Server địa phương
@@SERVICENAME	Tên của SQL Service trên máy tính hiện tại
@@TRANSCOUNT	Số lượng những giao dịch đang được mở
@@VERSION	Thông tin về phiên bản SQL Server đang sử dụng

4.3.2 Local variables

Cách khai báo và thực hiện:



```
DECLARE @CUST VARCHAR (5)
SET @CUST='FRANK'
SELECT CUSTOMERID, COMPANYNAME
FROM CUSTOMERS WHERE CUSTOMERID LIKE @CUST
```

	CUSTOMERID	COMPANYNAME
1	FRANK	Frankenversand

Hình 4.3. Sử dụng biến địa phương (Local variables)

4.4 Hàm (Functions)

Hàm có thể được chia làm 3 loại như sau:

Rowset Functions : Loại này thường trả về một object và được đối xử như một table. Ví dụ như hàm OPENQUERY sẽ trả về một recordset và có thể đứng vị trí của một table trong câu lệnh Select.

Aggregate Functions : Loại này làm việc trên một số giá trị và trả về một giá trị đơn hay là các giá trị tổng. Ví dụ như hàm AVG sẽ trả về giá trị trung bình của một cột.

Scalar Functions : Loại này làm việc trên một giá trị đơn và trả về một giá trị đơn. Trong loại này lại chia làm nhiều loại nhỏ như các hàm về toán học, về thời gian, xử lý kiểu dữ liệu String.... Ví dụ như hàm MONTH('2002-09-30') sẽ trả về tháng 9.

4.4.1 Hàm Conversion

Hàm conversion dùng để chuyển đổi giá trị của một kiểu dữ liệu này sang kiểu dữ liệu khác. Thêm nữa, chúng ta có thể sử dụng nó để đạt được một loạt định dạng ngày tháng đặc biệt. SQL Server cung cấp hàm chuyển đổi đơn giản CONVERT().

Cú pháp:

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

Ví dụ:

```
SELECT 'EMP ID:' + CONVERT (CHAR(4), EMPLOYEEID FROM EMPLOYEES
```

4.4.2 Hàm Data Parts

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

4.4.3 Hàm ngày tháng và hàm toán học

Hàm ngày tháng (Date Functions)

Date Functions

GETDATE()
DATEADD(datepart,number,date)
DATEDIFF(datepart,date1,date2)
DATENAME(datepart,date)
DATEPART(datepart,date)

Hàm toán học (Mathematical)

Mathematical Functions

ABS(num_expr)
CEILING(num_expr)
FLOOR(num_expr)
POWER(num_expr,y)
ROUND(num_expr,length)
Sign(num_expr)
Sqrt(float_expr)

4.4.4 Hàm hệ thống (System Function)

Function

DB_ID(['database_name'])
DB_NAME([database_id])
HOST_ID()
HOST_NAME()
ISNULL(expr,value)
OBJECT_ID('obj_name')
OBJECT_NAME(object_id)
SUSER_SID(['login_name'])
SUSER_ID(['login_name'])
SUSER_SNAME([server_user_id])
SUSER_NAME([server_user_id])
USER_ID(['user_name'])
USER_NAME([user_id])

4.4.5 Hàm nhóm

Sum(col_name)
Avg(col_name)
Count (col_name)
Min(col_name)
Max(col_name)

4.5 Câu hỏi trắc nghiệm

1. Batch là một tập hợp của một hoặc nhiều câu lệnh SQL được Client gửi tới Server như một khối thống nhất.

- A Đúng
- B Sai

2. Câu lệnh nào theo sau được dùng để thoát khỏi vòng lặp While.

- A Close
- B Exit
- C Break
- D Không có lệnh nào trong 3 lệnh trên

3. Tập hợp một số câu lệnh muốn được thực hiện 10 lần, sử dụng cấu trúc nào sau đây?

- A IF...ELSE
- B While
- C Case
- D Select

4. Câu lệnh nào theo sau được sử dụng để yêu cầu SQL Server đợi 15 giây trước khi nó tiếp tục?

- A WAITFOR '00:00:15' DELAY
- B WAITFOR DELAY BY '00:00:15'
- C WAITFOR DELAY '00:00:15'
- D WAIT FOR '00:00:15'

5. Kiểm tra đoạn mã sau đây:

```
DECLARE @v_empcount INT
SELECT @v_empcount = COUNT(*) FROM employee
IF (@v_empcount>20) GOTO great
ELSE
    GOTO less
SELECT * FROM employee
great:
PRINT 'The number is greater'
GOTO end1
GO
less:
PRINT 'The number is less'
end1:
GO
```

Chương 4. T-SQL PROGRAMING

Câu nào là kết quả của đoạn mã trên nếu có 20 bản ghi trong bảng Employee?

- A Kết quả in ra là: 'The number is great'
- B Kết quả in ra là: 'The number is less'
- C Không in ra gì cả
- D Có thông báo lỗi

6. ...functions được sử dụng khi cần trả lại các thông tin về thiết đặt (settings) của SQL Server?

- A Server
- B Aggregate
- C System
- D Text

7. Phát biểu nào sau đây là đúng cho việc chuyển đổi (conversion) kiểu dữ liệu?

- A Khi chuyển đổi bất kỳ một giá trị nào không phải là 0 (nonzero) sang kiểu dữ liệu bit thì dữ liệu đó sẽ trở thành 1.
- B Khi chuyển đổi từ kiểu dữ liệu Char thành kiểu dữ liệu Money bạn không thể có được ký hiệu \$ đi kèm.
- C Kiểu dữ liệu Image không thể được chuyển thành kiểu dữ liệu Binary.
- D Nếu bạn không chỉ ra độ dài cho kiểu dữ liệu trong biểu thức chuyển đổi, SQL Server sẽ tự động gán cho nó có độ dài là 20.

8. Truy vấn nào sau đây sẽ trả lại giá trị của tháng hiện tại?

- A `Select DATEPART(mm,getdate()) As 'Month'`
- B `Select DATEPART(mon,getdate()) As 'Month'`
- C `Select GETDATE(DATEPART(MM)) As 'Month'`
- D `Select GETDATE(DATEPART(MON)) As 'Month'`

5 Chương 5. TRANSACTIONS VÀ LOCKS

5.1 Giới thiệu Transactions-Giao dịch

Trong cơ sở dữ liệu đa người dùng, dữ liệu được lưu trữ ở hệ thống máy chủ và các chương trình của người sử dụng có thể đồng thời thao tác trên những dữ liệu đó. Việc thực hiện một chương trình truy cập và thay đổi nội dung của cơ sở dữ liệu được gọi là Transaction.

Một Transaction là một hoặc một tập các lệnh được thực hiện như một khối (unit of works), tức là nó có thể thành công hoặc thất bại hoàn toàn. Có nghĩa là, một transaction thành công nếu tất cả các lệnh trong transaction đó được thực hiện và thất bại nếu một trong số các lệnh của khối thất bại.

5.2 Các tính chất của Transaction

Để đảm bảo tính toàn vẹn dữ liệu, một giao dịch phải có các tính chất sau:

- Atomicity _Nguyên tử.
- Consistency_ Nhất quán
- Isolation _ Cô lập
- Durability _Bền vững

4 tính chất trên gọi là tính ACID của giao dịch

Atomicity Nguyên tử: Đảm bảo hoặc toàn bộ các hoạt động trong giao dịch thành công hoặc thất bại

Consistency Nhất quán: Khi transaction hoàn thành, dữ liệu phải ở trạng thái nhất quán. Ví dụ: Trong 1 transaction, tài khoản A thực hiện chuyển tiền cho 1 tài khoản B là 5 triệu, thì sau khi hoàn thành tài khoản B có số tiền là B+5triệu và A là A-5triệu.

Isolation Cô lập: Cho dù có nhiều giao dịch có thể thực hiện đồng thời, hệ thống phải đảm bảo rằng sự thực hiện đồng thời đó dẫn đến một trạng thái hệ thống tương đương khi các giao dịch thực hiện tuần tự theo một thứ tự nào đó.

Durability Bền vững: Sau khi giao dịch thành công, giả sử xảy ra sự cố thì tất cả các dữ liệu được thay đổi trong giao dịch vẫn được khôi phục lại theo yêu cầu giao dịch.

5.2.1 Phân loại transaction

Có 3 loại:

- Explicit transactions (rõ ràng)
- Implicit transactions (ngầm định)

- Auto-Commit transactions (Transaction ở chế độ tự động thực hiện)

Explicit transactions:

Explicit Transaction là transaction do người sử dụng tự định nghĩa. Chúng ta phải định nghĩa bắt đầu và kết thúc transaction, có dạng như sau:

```
BEGIN TRAN
INSERT RECORD
DELETE RECORD
COMMIT TRAN
```

Begin transaction, commit transaction, rollback transaction được sử dụng để định nghĩa Explicit Transaction. Trong đó

BEGIN TRANSACTION là câu lệnh đánh dấu bắt đầu transaction.

COMMIT TRANSACTION là câu lệnh kết thúc một transaction thành công.

ROLLBACK TRANSACTION quay trở thời điểm bắt đầu của transaction (tức là tất cả những thao tác được thực hiện từ lúc Begin transaction sẽ bị huỷ bỏ) hoặc là một vị trí nào đó được đánh dấu.

Implicit transactions: Là chuyển tác ngầm định.

Nó không yêu cầu phát biểu chuyển tác Begin transaction. Bản thân nó tự động khởi tạo. Trong SQLServer chuyển tác ngầm định mặc định ở chế độ nghỉ. Muốn sử dụng và tắt chế độ làm việc của Implicit transaction ta thực hiện như sau:

- Bật chế độ làm việc:

```
SET implicit_transactions ON
```

- Tắt chế độ làm việc:

```
SET implicit_transactions OFF
```

Auto-Commit transactions:

Tất cả các câu lệnh T-SQL hoặc thực hiện thành công, hoặc thất bại hoàn toàn. Nếu câu lệnh thành công thì nó được committed, nếu thất bại nó sẽ được rollback lại.

Đây là chế độ mặc định của SQL SERVER

5.3 Các mức cô lập của Transaction

5.3.1 Giới thiệu Dirty Read (Đọc các dữ liệu bẩn)

Các dữ liệu bẩn (dirty data) là một thuật ngữ chung chỉ các dữ liệu được ghi bằng một giao tác nhưng còn chưa được lưu giữ lại (committed). Một dirty read dùng để

Chương 5. TRANSACTIONS VÀ LOCKS

đọc các dữ liệu bản. Điều nguy hiểm của việc đọc các dữ liệu bản là ở chỗ một giao tác ghi nó có thể bị bỏ dở. Nếu vậy thì các dữ liệu bản sẽ bị đẩy ra khỏi cơ sở dữ liệu và mọi người được phép xử sự như là các dữ liệu đó chưa bao giờ tồn tại. Nếu một giao tác khác nào đó đã đọc các dữ liệu bản thì giao tác đó có thể lưu giữ hoặc thực hiện một hành động nào đó phản ánh sự hiểu biết của nó về dữ liệu bản.

Đôi lúc dirty read có ý nghĩa, đôi lúc nó không có ý nghĩa. Lúc khác nó có ý nghĩa rất nhỏ đủ để tạo ý nghĩa về nguy cơ của một dirty read phụ động và như vậy làm ngăn cản:

1. Công việc tốn thời gian của hệ quản trị cơ sở dữ liệu cần để ngăn ngừa dirty read và
2. Mất tính song song gây ra từ sự chờ đợi cho đến khi không có thể có một dirty read.

Sau đây là một số ví dụ về những cái có thể xảy ra khi cho phép có các dirty read.

Ví dụ 1: Chúng ta hãy xem xét việc chuyển tài khoản. Giả sử rằng các vụ chuyển được thực hiện bằng một chương trình P thực hiện dãy các bước sau đây:

1. Thêm tiền vào tài khoản 2
2. Kiểm tra nếu tài khoản 1 có đủ tiền
 - a) Nếu không có đủ tiền, lấy tiền ra khỏi tài khoản 2 và kết thúc
 - b) Nếu có đủ tiền, trừ số tiền từ tài khoản 1 và kết thúc.

Nếu chương trình P được thực hiện một cách có thứ tự thì việc chúng ta thêm tiền tạm thời vào tài khoản 2 sẽ không có ý nghĩa gì. Không ai sẽ nhìn thấy số tiền đó và nó sẽ bị loại bỏ nếu việc chuyển tiền là không thực hiện được.

Tuy nhiên, giả sử rằng có các dirty read. Hãy tưởng tượng có 3 tài khoản A1, A2, A3 với 100\$, 200\$ và 300\$ tương ứng. Giả sử rằng giao tác T1 thực hiện chương trình P để chuyển 150\$ từ A1 đến A2. Cùng một thời gian, giao tác T2 chạy chương trình P để chuyển 250\$ từ A2 đến A3. Có khả năng có các dãy sự kiện sau:

1. T2 thực hiện bước 1 và thêm 250\$ vào A3 và bây giờ A3 có 550\$
2. T1 thực hiện bước 1 và thêm 150\$ vào A2 và bây giờ A2 có 350\$
3. T2 thực hiện kiểm tra của bước 2 và tìm ra rằng A2 có đủ tiền (350\$) để cho phép chuyển 250\$ từ A2 sang A3.

Chương 5. TRANSACTIONS VÀ LOCKS

4. T1 thực hiện kiểm tra của bước 2 và tìm ra rằng T1 không có đủ tiền (100\$) để cho phép chuyển 150\$ từ A1 sang A2.
5. T2 thực hiện bước 2b. Nó trừ đi 250\$ khỏi A2 và bây giờ A2 có 100\$ và kết thúc.
6. T1 thực hiện bước 2a. Nó trừ 150\$ khỏi A2, bây giờ A2 có -50\$ và kết thúc.

Tổng số tiền không thay đổi; trong ba tài khoản vẫn còn 600\$. Nhưng bởi vì T2 đọc dữ liệu bản ở bước 3 trong 6 bước trên, chúng ta không bảo vệ được việc một tài khoản trở nên âm, đó là mục đích của việc kiểm tra tài khoản thứ nhất để xem tài khoản này có số tiền thích hợp hay không.

5.3.2 Các mức cô lập

Dựa vào mức độ “dung thứ” đối với những dữ liệu không chính xác. Isolations được phân loại như sau:

- Read Uncommitted
- Read Committed
- Repeatable read
- Serializable

Read Uncommitted: Cho phép đọc cả những dữ liệu bản.

SQL cho phép chúng ta chỉ ra rằng các dirty read là chấp nhận được với một giao tác cho trước. Chúng ta sử dụng lệnh SET TRANSACTION.

1) SET TRANSACTION READ WRITE

2) SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

Lệnh trên làm hai việc:

1. Dòng (1) khai báo rằng giao tác có thể ghi dữ liệu
2. Dòng (2) khai báo rằng giao tác có thể chạy với “ isolation level” read-uncommitted. Điều đó có nghĩa là giao tác được cho phép đọc các dữ liệu bản.

Chú ý rằng, nếu giao tác không phải là read-only (tức là có thể sửa đổi cơ sở dữ liệu) và chúng ta chỉ ra mức cô lập (isolation level) READ UNCOMMITTED thì chúng ta cũng phải chỉ ra READ WRITE. Mặc dù, ở chế độ ngầm định các giao tác là read-write. Tuy nhiên SQL có một ngoại lệ đối với trường hợp có cho phép các dirty-read. Trong trường hợp đó, giả thiết ngầm định là giao tác là read-only, bởi vì các giao tác read-write với dirty read gây ra các nguy hiểm đáng kể như chúng ta đã

thấy. Nếu chúng ta muốn một giao tác read-write chạy với read-uncommitted như là mức cô lập thì chúng ta cần chỉ ra READ WRITE một cách rõ ràng như ở trên.

Read Committed

SQLSV sử dụng chia sẻ khoá trong khi đọc dữ liệu (ứng dụng này không được phép đọc những dữ liệu mà ứng dụng khác đã thay đổi nhưng chưa được committed) đây là chế độ mặc định của SQLSV.

Cú pháp:

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED ;
```

Repeatable read:

Locks sẽ được đặt trên tất cả các truy vấn đang sử dụng, và các transaction khác không thể cập nhật -> không dirty read.

Serializable: Xếp hàng thứ tự (giao tác phải chạy trước hoặc sau một giao tác khác đã hoàn thành).

Đây mức Isolation hạn chế nhất, dữ liệu phantom không thể xảy ra. Nó ngăn cản người dùng thêm, sửa dữ liệu cho đến khi transaction hoàn thành.

5.4 Locks

5.4.1 Khái niệm

Lock là được sử dụng để hạn chế việc truy cập đến dữ liệu trong môi trường đa người dùng.

Microsoft SQL Server 2000 sử dụng khoá để đảm bảo tính toàn vẹn và đúng đắn của cơ sở dữ liệu

Nếu khoá không được sử dụng, dữ liệu trong CSDL có thể trở thành thiếu logic, và những truy vấn được thực hiện trên dữ liệu có thể trả về kết quả không như mong muốn

Khái niệm cơ bản đằng sau việc khoá là vì user cần có những truy cập riêng biệt tới bảng, vì thế server khoá bảng lại cho từng user.

5.4.2 Phân loại

Các loại Lock trong SQL SERVER

- Pessimistic Lock (Khoá bi quan)
- Optimistic Lock (Khoá lạc quan)
- Shared Locks (Khoá chia sẻ)
- Exclusive Locks (Khoá độc quyền)
- Update Locks (Khoá cập nhật)

Chương 5. TRANSACTIONS VÀ LOCKS

Pessimistic Lock: Khóa được áp dụng ngay khi dữ liệu được yêu cầu sửa chữa và khóa này chỉ giải phóng khi việc sửa chữa hoàn thành. Điều này để chắc chắn rằng không có người dùng khác đồng thời sửa dữ liệu.

Optimistic Lock: Khóa này chỉ áp dụng khi việc sửa chữa được hoàn thành và dữ liệu thực sự (actual data) chuẩn bị ghi lên bảng thực sự (actual table)

Shared Locks: Đây là loại khóa căn bản nhất, cho phép đọc dữ liệu, không cho phép thay đổi bất kỳ thuộc tính nào của dữ liệu.

Exclusive Locks: Lock này ngăn ngừa 2 người sử dụng cùng cập nhật, xóa, đọc, thêm mẫu tin đồng thời.

Update Locks: Kết hợp giữa Shared Locks và Exclusive Locks. Với phát biểu UPDATE chỉ ra mẫu tin cần cập nhật bằng lệnh Where, trong khi chưa cần cập nhật bạn ở chế độ Shared Locks, khi cần thực sự thực thi, bạn ở chế độ Exclusive Locks.

5.5 Câu hỏi trắc nghiệm

1. Câu lệnh nào cho phép quay trở thời điểm bắt đầu của transaction hoặc là một vị trí nào đó được đánh dấu.

- A COMMIT TRANSACTION
- B ROLLBACK TRANSACTION
- C SAVE TRANSACTION
- D BEGIN TRANSACTION

2. Sự thực hiện của một chương trình cho phép truy cập và thay đổi nội dung của cơ sở dữ liệu gọi là gì?

- A UPDATE
- B INSERT
- C Transaction
- D Không có cái nào ở trên

3. Khi sửa chữa cơ sở dữ liệu phải tuân theo nguyên tắc "tất cả hoặc không gì cả", thì mỗi transaction được nói tới như là:

- A Consistent
- B Durable
- C Isolated
- D Atomic

4. Explicit transaction không yêu cầu bạn định nghĩa tường minh bắt đầu và kết thúc transaction.

- A Đúng
- B Sai

5. Câu lệnh nào là câu lệnh kết thúc một transaction.

- A COMMIT TRANSACTION
- B ROLLBACK TRANSACTION
- C SAVE TRANSACTION
- D BEGIN TRANSACTION

6. Loại khóa nào sau đây ngăn ngừa 2 người sử dụng cùng cập nhật, xoá, đọc, thêm mẫu tin đồng thời.

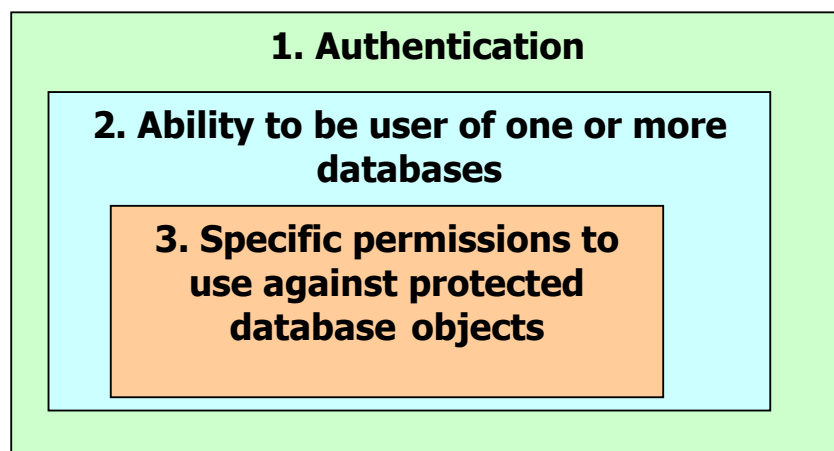
- A Optimistic Lock
- B Pessimistic Lock
- C Shared Lock
- D Physical Lock

6 Chương 6. BẢO MẬT VÀ QUẢN LÝ NGƯỜI DÙNG (USER AND SECURITY)

Cơ sở dữ liệu trong một tổ chức có thể chứa những dữ liệu quan trọng và cần giữ bí mật. Vì thế thực hiện bảo mật cơ sở dữ liệu rất cần thiết đối với một Hệ quản trị cơ sở dữ liệu. Trong phần này chúng ta sẽ bàn về các nguyên tắc bảo mật trên SQL Server 2000 và cách thực hiện nó.

6.1 Giới thiệu về SQL Server Security

SQL Server Security sử dụng mô hình 3 mức như sau:



Hình 6.1. Bảo mật 3 mức của SQL SERVER

- Mức ngoài cùng là mức xác thực người dùng của NT và SQL Server
- Mức thứ 2 kiểm tra người dùng có quyền sử dụng một hay nhiều Database.
- Mức trong cùng xác định quyền của người dùng thực hiện những câu lệnh SQL trên những đối tượng Database được bảo vệ.

6.2 Quản lý đăng nhập (Login)

SQL Server **xác thực** người dùng ở hai mức:

- Login authentication: Xác thực đăng nhập
- Permissions validation on user database: Kiểm tra quyền của đối tượng người dùng

6.2.1 Xác thực đăng nhập

Người dùng phải có Login account để kết nối với SQL SERVER. SQL Server xác nhận người dùng bằng 3 cơ chế:

Chương 6. BẢO MẬT VÀ QUẢN LÝ NGƯỜI DÙNG (USER AND SECURITY)

- SQL Server Authentication: Khi sử dụng SQL Server Authentication, người sử dụng phải nhập Account và Password đã được cấp.
- Windows NT Authentication: Khi sử dụng Windows NT Authentication, SQL Server sẽ xác minh lại cùng với WinNT để kiểm tra xem có trùng khớp không, nếu đúng thì cho phép đăng nhập.
- Mixed Mode Authentication: Người sử dụng có thể đăng nhập sử dụng SQL Server Authentication hoặc Windows NT Authentication.

6.2.2 Kiểm tra quyền (Permission)

Với mỗi cơ sở dữ liệu, người dùng được cấp một số quyền trên đó, và họ chỉ được thực hiện các thao tác trong giới hạn quyền của họ.

SQL Server thực hiện những bước sau trong khi kiểm tra quyền:

- Khi người dùng thực hiện một thao tác, ví dụ như thực thi câu lệnh T-SQL hoặc lựa chọn công việc trên thanh công cụ của EM, câu lệnh T-SQL sẽ được gửi tới SQL SERVER.
- Khi SQL Server nhận câu lệnh T-SQL, nó kiểm tra quyền của người dùng trên những đối tượng liên quan trong câu lệnh T-SQL.
 - o Nếu người dùng không được phép, SQL Server sẽ báo lỗi
 - o Ngược lại, nó thực hiện và trả về kết quả.

6.2.3 Tạo Login

Chúng ta có nhiều cách để tạo Login, sử dụng EM hoặc câu lệnh T-SQL SERVER.

Tạo Login bằng EM:

Sử dụng Create Login Wizard và thực hiện các bước theo chỉ dẫn.

Tạo Login bằng T-SQL:

Để tạo một login ta có lệnh sau:

```
EXEC sp_addlogin 'phnhung', '123456'
```

Để xoá một login ta có lệnh sau:

```
EXEC sp_droplogin 'Arwen'
```



Hình 6.2. Tạo Login

6.3 Quản lý người dùng

SQL Server cho phép hai kiểu User Accounts:

- System User
- Database User

6.3.1 SQL Server Users

Một user identifier (ID) xác định một User trong Database. Tất cả quyền và sở hữu các đối tượng trong Database được xác định bằng user accounts. User accounts được xác định cho từng Database. Ví dụ: User account: abc trong Database Book khác user account: abc trong Database Customer.

Quan hệ giữa Database User và Login name:

User trong Database được xác định bằng user ID, chứ không phải login ID.

Bản thân login ID không cấp bất cứ quyền nào cho người dùng truy cập vào các đối tượng trong Database. Ví dụ: sa là login account ánh xạ tới người dùng tới user account đặc biệt có tên là dbo(database owner) trong bất kỳ Database nào.

6.3.2 Quản lý Username và Login name

Ví dụ chỉ ra quá trình cấp cho một người dùng Windows 2000 truy cập Database và kết hợp với việc đăng nhập bằng user trong Database.

```
USE master
GO
sp_grantlogin 'OnlineDOMAIN\Arwen'
GO
sp_defaultdb @loginame = 'OnlineDOMAIN\Arwen', defdb = 'books'
GO
USE books
GO
sp_grantdbaccess 'OnlineDOMAIN\Arwen', 'Arwen'
GO
sp_revokedbaccess 'OnlineDOMAIN\Arwen', 'Arwen'
GO
```

Trong đó:

- **sp_grantlogin:** cho phép người dùng hoặc nhóm account của Windows NT/2000 truy cập vào Database bằng Windows authentication.
- **sp_defaultdb:** thay đổi Database mặc định cho việc đăng nhập.
- **sp_grantdbaccess:** thêm account và cấp quyền truy cập cho nó.
- **sp_revokedbaccess:** xoá account từ Database.

6.4 Quản lý Role

Roles rất quan trọng vì nó là cách chính để cung cấp quyền cho người dùng. Quyền có thể được cấp cho người dùng bằng cách cấp quyền trực tiếp, cách này làm cho người quản trị hệ thống làm việc rất mất thời gian và nhàm chán vì phải cấp chi tiết từng quyền cho từng người dùng. Hoặc quyền có thể được cấp thông qua Role - nó tương tự như khái niệm group trong NT. Chúng ta gán quyền cho từng Role và sau đó người dùng được xếp vào Role đó. Như vậy, công việc của người quản trị hệ thống trở nên dễ dàng hơn nhiều lần.

SQL Server có **Database Roles** và **Server Roles**. Database Roles được sử dụng để cung cấp các mức khác nhau để truy cập vào cơ sở dữ liệu. Server Roles được sử dụng để cho phép hoặc hạn chế người sử dụng thực hiện các thao tác (operations) trên cơ sở dữ liệu.

6.4.1 Database Roles

Có một kiểu Database Role rất đặc biệt, đó là **public role**.

- Nó có trong tất cả các Database.

Chương 6. BẢO MẬT VÀ QUẢN LÝ NGƯỜI DÙNG (USER AND SECURITY)

- Nó không thể xoá.
- Tất cả các người dùng đều thuộc về Public role, bao gồm cả sa account.

Sau đây là một số Database role và những quyền tương ứng:

- **db_owner:** Đây là Role cao nhất người dùng có thể có. Role này cho phép người dùng mọi quyền trên CSDL. Người dùng sa nằm trong Role này.
- **db_securityadmin:** Cho phép người dùng quản lý mọi Roles và nhóm người dùng trong role.
- **db_accessadmin:** Cung cấp cho người dùng quyền thêm hoặc xoá những người dùng khác trong CSDL.
- **db_ddladmin:** Cho phép người dùng thực thi mọi nhiệm vụ trên mọi đối tượng trong CSDL: người dùng có thể tạo, sửa, xoá các đối tượng.
- **db_backupoperator:** Cho phép người dùng thực hiện việc backup dữ liệu.
- **db_datareader:** Cho phép người dùng xem dữ liệu trên các bảng của CSDL.
- **db_denydatawriter:** Ngăn cản người dùng sửa bất kỳ dữ liệu nào trên bảng.
- **db_denydatareader:** Ngăn cản người dùng xem bất kỳ dữ liệu nào trên bảng.

6.4.2 Server Roles

Server roles được SQL Server 2000 cho phép:

- **Sysadmin:** Có đầy đủ mọi quyền trên SQL Server.
- **Securityadmin:** Cho phép tạo và quản lý việc đăng nhập cho Server.
- **Serveradmin:** Cho phép thiết lập cấu hình của các instance trên SQL Server.
- **Setupadmin:** Có khả năng để quản lý các thủ tục khởi động và các server được liên kết.
- **Processadmin:** Có khả năng để quản lý các tiến trình đang chạy trên SQL Server.
- **Diskadmin:** Có thể quản lý các file trên đĩa
- **Dbcreator:** Cho phép tạo, sửa, và xoá CSDL

6.4.3 Thêm thành viên cho Role

Để thêm Role mới, chúng ta có thể thực hiện bằng EM hoặc sử dụng thủ tục `sp_addrole`, và để thêm thành viên cho Role, chúng ta sử dụng thủ tục `sp_addrolemember`. Ví dụ sau chỉ ra cách tạo Role và thêm các thành viên cho role.

```
sp_addrole 'Teacher'
GO
sp_addrole 'Student'
GO
sp_addrole 'StudentTeacher'
GO
sp_addrolemember 'Teacher', 'NETDOMAIN\Peter'
GO
sp_addrolemember 'Teacher', 'NETDOMAIN\Cathy'
GO
sp_addrolemember 'StudentProfessor', 'NETDOMAIN\Diane'
GO
sp_addrolemember 'Student', 'NETDOMAIN\Mel'
GO
sp_addrolemember 'Student', 'NETDOMAIN\Jim'
GO
sp_addrolemember 'Student', 'NETDOMAIN\Lara'
GO
GRANT SELECT ON StudentGradeView TO Student
GO
GRANT SELECT, UPDATE ON ProfessorGradeView TO Professor
GO
```

6.5 Đối tượng và quyền trên đối tượng (Database Objects and Object Permission)

6.5.1 Đối tượng

Các đối tượng trong cơ sở dữ liệu:

Database Object
1. Table
2. Column
3. Row
4. Data type

5. Constraint
6. Default
7. Rule
8. Index
9. Views
10. Stored Procedure
11. Trigger

6.5.2 Quyền

Object permissions: điều khiển ai có thể truy cập và thao tác với dữ liệu trên bảng (tables) và khung nhìn (views) và ai có thể được chạy các stored procedures.

Statement permissions điều khiển users nào có thể xóa và tạo đối tượng trong Database.

Object Type	Possible Actions
Table	SELECT, UPDATE, DELETE, INSERT, REFERENCE
Column	SELECT, UPDATE
View	SELECT, UPDATE, INSERT, DELETE
Stored procedure	EXECUTE

6.5.3 Cho phép và hủy bỏ quyền trên đối tượng

User người mà tạo ra các đối tượng trong Database được gọi là object owner. Vì thế những user này phải có quyền để tạo ra những đối tượng trong Database.

SQL Server sử dụng lệnh GRANT, REVOKE, và DENY để quản lý quyền:

GRANT: Cho phép người dùng thực hiện thao tác như SELECT, UPDATE, INSERT, DELETE hoặc EXECUTED trên các đối tượng.

Ví dụ:

Chương 6. BẢO MẬT VÀ QUẢN LÝ NGƯỜI DÙNG (USER AND SECURITY)

```
GRANT INSERT, UPDATE, DELETE  
ON authors  
TO Mary, John, Tom
```

Ví dụ trên thực hiện việc cấp quyền INSERT, UPDATE, DELETE trên bảng Authors cho người dùng Mary, John, Tom.

REVOKE: Được sử dụng để xoá quyền của người sử dụng.

DENY: Được sử dụng để ngăn cản người sử dụng thực hiện các thao tác trên các đối tượng.

6.6 Câu hỏi trắc nghiệm

1. Thủ tục nào theo sau cho phép thêm một account và cấp quyền cho nó?

- A sp_grantdbaccess
- B sp_grantlogin
- C sp_grantuser
- D sp_grantall

2. Thủ tục nào theo sau cho phép thêm một nhóm tới Server Role ?

- A sp_addsrvrolemember
- B sp_addrolemember
- C sp_addserverolemember

3. Role cho phép chúng ta giới hạn hoặc cho phép người dùng thực hiện hàng loạt các thao tác được gọi là roles

- A Server
- B Database
- C User
- D System

4. Những người dùng tạo ra các đối tượng trong cơ sở dữ liệu gọi là

- A Database Owner
- B Table Owner
- C Account user
- D Administrator

5. Role nào sau đây không thể bị xóa

- A Public
- B Database
- C Server
- D System

6. Permissions nào sau đây quyết định người dùng có thể tạo hoặc xóa các đối tượng trong cơ sở dữ liệu?

- A Statement Permissions
- B User Permissions
- C Object Permissions
- D Database Permissions

7. Role nào sau đây có trong tất cả các cơ sở dữ liệu?

Chương 6. BẢO MẬT VÀ QUẢN LÝ NGƯỜI DÙNG (USER AND SECURITY)

- A Public
- B Database
- C Server

8. Về khái niệm, cái gì theo sau tương đương với group trong NT?

- A Login Account
- B Role
- C User Account
- D Permission

9. SQL Server sử dụng những lệnh ..., ..., để quản lý Permissions?

- A GRANT, DENY, REVOKE
- B ALLOW, DENY, REVOKE
- C ALLOW, DISALLOW, PERMIT

7 Chương 7. T-SQL PROGRAMMING, TRANSACTIONS, MANAGING SECURITY - Phần thực hành

Mục đích:

- Viết các câu lệnh T-SQL để thực hiện các kiểu transactions
- Định nghĩa và sử dụng biến local và global
- Tạo các logins và users

7.1 Hướng dẫn trực tiếp

7.1.1 Transactions

Là nhóm các yêu cầu được thực hiện như một khối (unit). Có 3 loại transactions:

- Implicit
- Explicit
- Auto-commit

7.1.1.1 Implicit Transactions

Là chuyển tác ngầm định, bản thân nó tự động khởi tạo. Mặc định, Implicit Transactions ở chế độ OFF.

Thực hiện các bước sau:

1. Mở QA
2. Thực hiện câu lệnh sau: Tạo Implicit transaction để thay đổi giá trị của cột Category_code từ 'D' thành 'Do' trong bảng Category_master(bảng cha). Đồng thời thay đổi luôn giá trị tương ứng của nó trong bảng Flight(bảng con chứa khoá ngoại). Việc thay đổi này bắt buộc phải thực hiện đồng thời trên hai bảng hoặc nếu không được thì không thay đổi gì cả .

```
SET implicit_transactions ON
Select Category_code FROM Category_master
Select Category FROM Flight

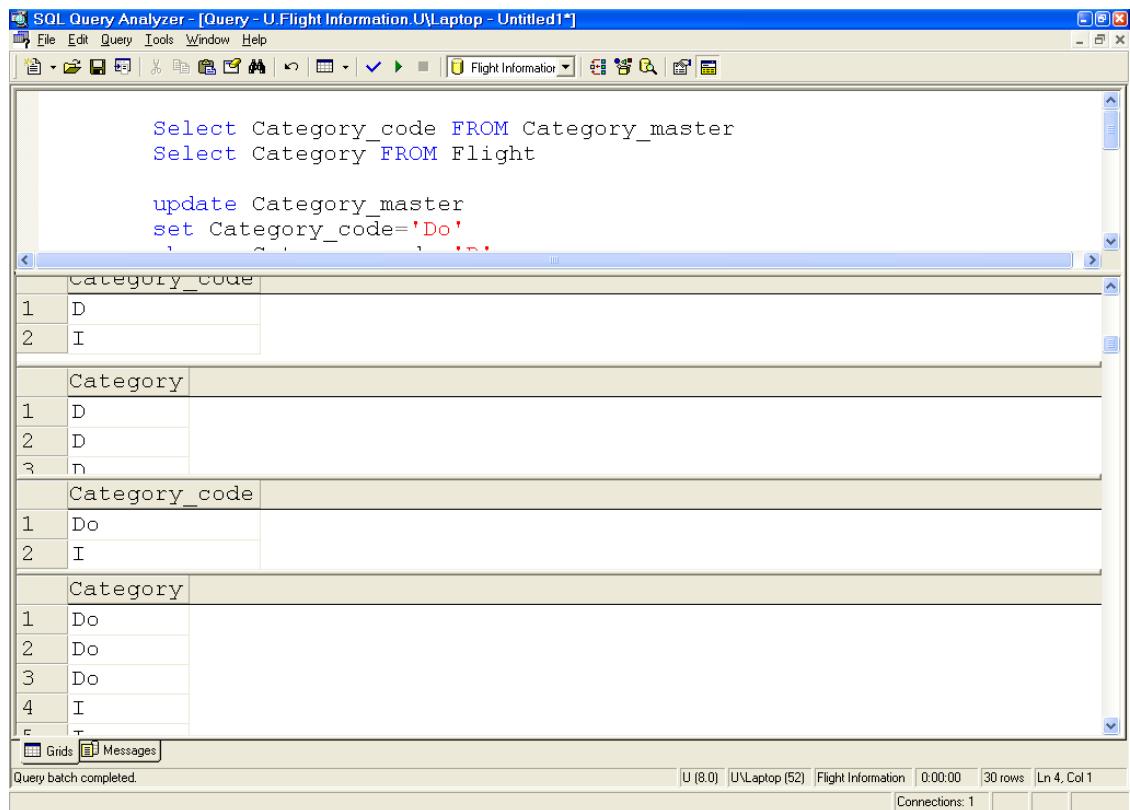
Update Category_master
Set Category_code='Do'
Where Category_code='D'

Update Flight
Set Category='Do'
where Category='D'
```

Chương 7. T-SQL PROGRAMMING, TRANSACTIONS, SECURITY – Phần thực hành

```
Select Category_code FROM Category_master
Select Category FROM Flight
COMMIT TRANSACTION
SET implicit transactions OFF
```

Kết quả:



Hình 7.1. Tạo Implicit Transactions

Chú ý khi sử dụng implicit transaction ta phải sử dụng SET... ON và SET... OFF.

7.1.1.2 Explicit Transactions

Là transaction tường minh, chúng ta phải định nghĩa bắt đầu và kết thúc transaction. Explicit Transactions còn được gọi là User-defined transactions.

Thực hiện transaction để tăng giá vé máy bay (cột fare trong bảng Flight_details) lên 200 cho hạng vé 'Ex' và hãng bay 'IC04', xem thông tin trước và sau khi cập nhật.

Câu lệnh và kết quả thực hiện như sau:

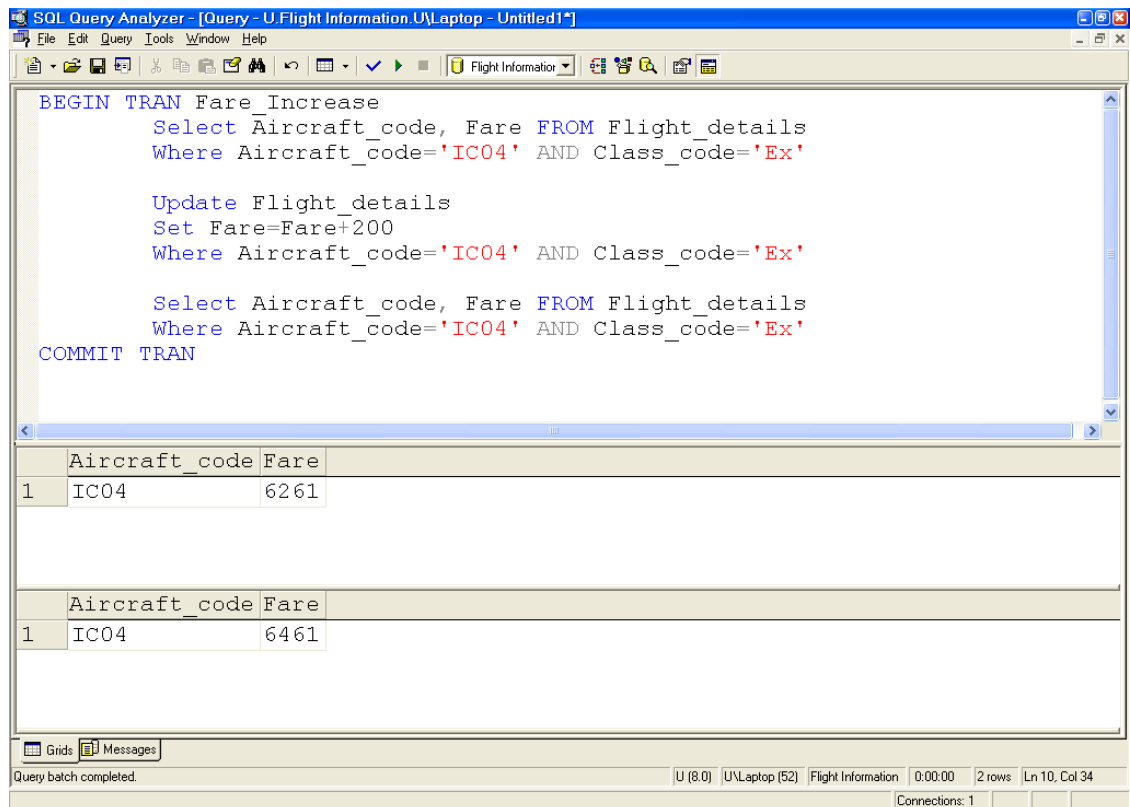
```
BEGIN TRAN Fare_Increase
Select Aircraft code, Fare FROM Flight_details
```

Chương 7. T-SQL PROGRAMMING, TRANSACTIONS, SECURITY – Phần thực hành

```
Where Aircraft_code='IC04' AND Class_code='Ex'

Update Flight_details
Set Fare=Fare+200
Where Aircraft_code='IC04' AND Class_code='Ex'

Select Aircraft_code, Fare FROM Flight_details
Where Aircraft_code='IC04' AND Class_code='Ex'
COMMIT TRAN
```



Hình 7.2. Tạo Implicit Transactions

Sử dụng Rollback Transaction

Thực hiện Transaction sau:

```
BEGIN TRAN Use_Rollback
  Select Aircraft_code, Fare FROM Flight_details
  Where Aircraft_code='IC04' AND Class_code='Ex'

  Update Flight_details
  Set Fare=Fare+200
  Where Aircraft_code='IC04' AND Class_code='Ex'
```

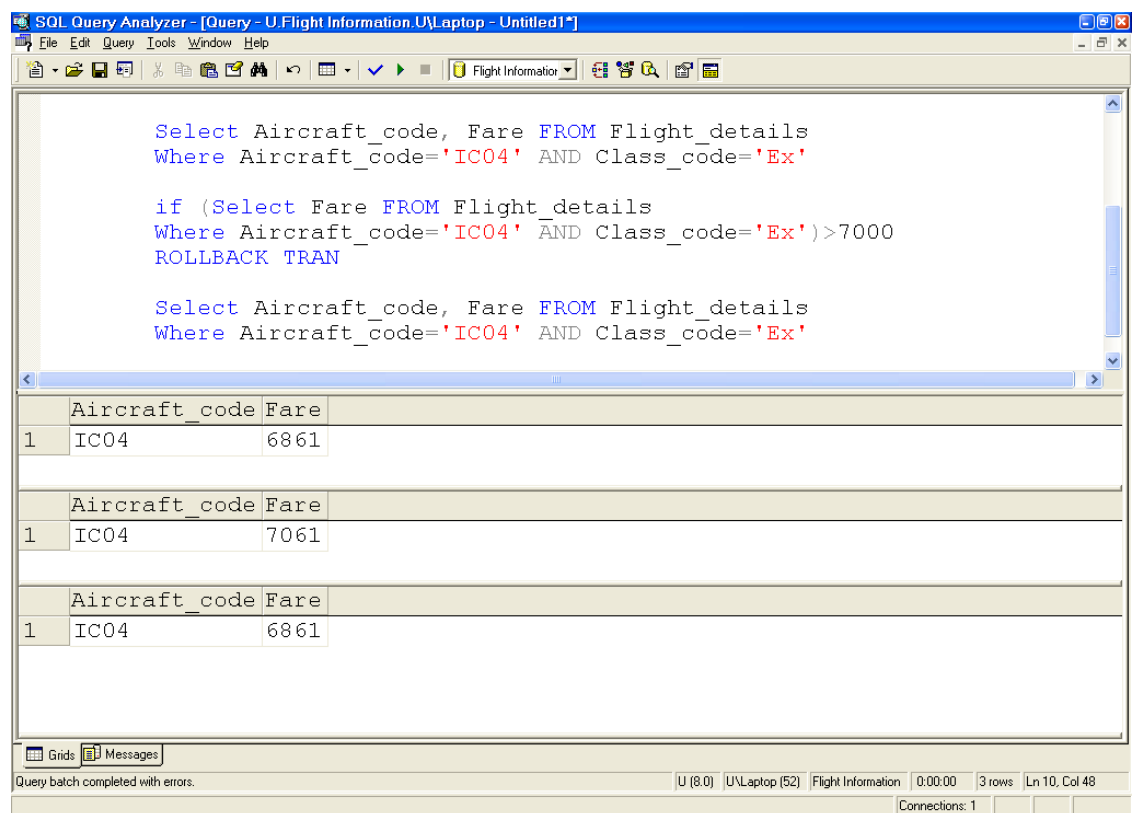
Chương 7. T-SQL PROGRAMMING, TRANSACTIONS, SECURITY – Phần thực hành

```
Select Aircraft_code, Fare FROM Flight_details
Where Aircraft_code='IC04' AND Class_code='Ex'

if (Select Fare FROM Flight_details
Where Aircraft_code='IC04' AND Class_code='Ex')>7000
ROLLBACK TRAN

Select Aircraft_code, Fare FROM Flight_details
Where Aircraft_code='IC04' AND Class_code='Ex'
COMMIT TRAN
```

Kết quả thực hiện:



Hình 7.3. Sử dụng Rollback Transaction

7.1.1.3 Autocommit Transaction

Thực hiện những câu lệnh sau:

```
create table Test(col1 int primary key, col2 char(3))
insert into Test values(1, 'aaa')
insert into Test values(2, 'bbb')
```

Transaction trên là Autocommit transaction, vì không có BEGIN TRANSACTION. Autocommit transaction sẽ tự động Commit nếu không xảy ra lỗi,

Chương 7. T-SQL PROGRAMMING, TRANSACTIONS, SECURITY – Phần thực hành

ngược lại nó sẽ Rollback. Vì thế, mặc dù lỗi xảy ra ở dòng 1 nhưng kết quả của những câu lệnh trước cũng không được thực hiện.

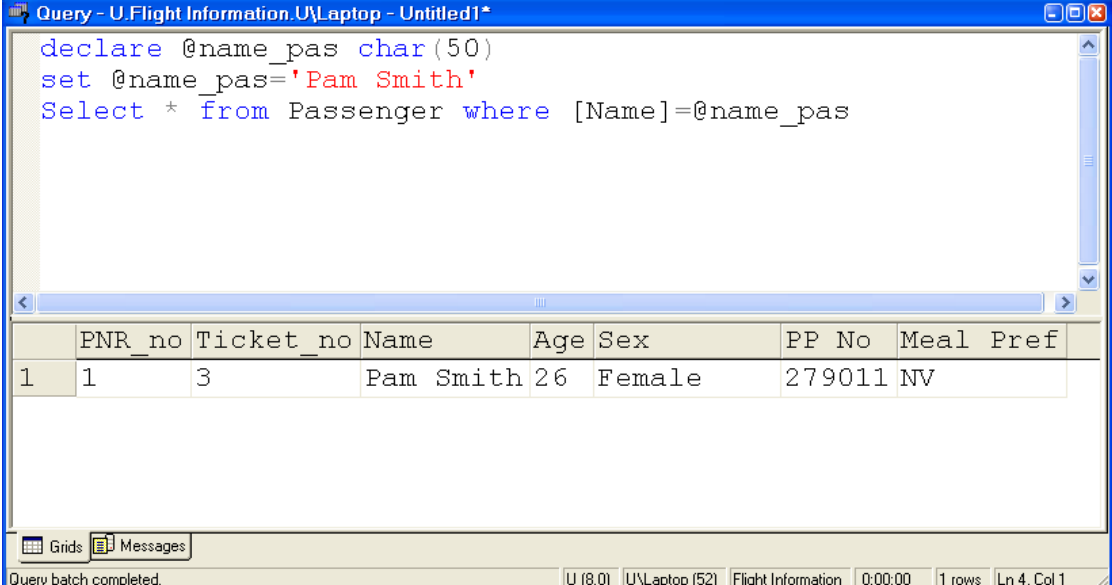
7.1.2 Biến địa phương (local) và biến toàn cục(Global)

Biến là đối tượng có thể nắm giá trị dữ liệu. Chúng ta có thể sử dụng biến địa phương để biểu diễn dữ liệu trong câu lệnh SQL SERVER.

Ví dụ:

```
declare @name_pas char(50)
set @name_pas='Pam Smith'
Select * from Passenger where [Name]=@name_pas
```

Kết quả:



The screenshot shows a SQL query window titled "Query - U.Flight Information.U\Laptop - Untitled1*". The query text is:

```
declare @name_pas char(50)
set @name_pas='Pam Smith'
Select * from Passenger where [Name]=@name_pas
```

The result is displayed in a table with the following data:

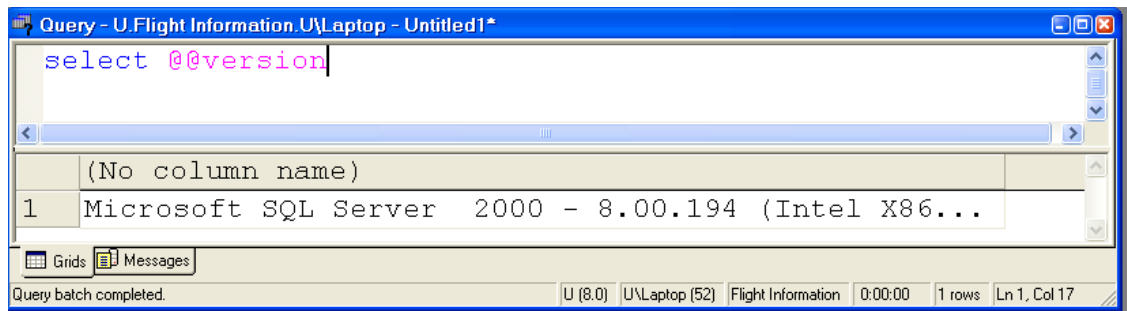
	PNR_no	Ticket_no	Name	Age	Sex	PP No	Meal Pref
1	1	3	Pam Smith	26	Female	279011	NV

The status bar at the bottom indicates "Query batch completed." and "U (8.0) U\Laptop (52) Flight Information 0:00:00 1 rows Ln 4, Col 1".

Hình 7.4. Sử dụng biến

SQL Server cũng hỗ trợ một số biến toàn cục hệ thống (system global), nắm giữ những thông tin hữu ích đối với những người sử dụng cơ sở dữ liệu. Trước những biến này phải có ký hiệu @@.

Ví dụ:



Hình 7.5

7.1.3 SQL Server Security

7.1.3.1 Tạo logins sử dụng Enterprise Manager.

Thực hiện các bước sau:

1. Kích Wizard trên Tool menu
2. Trong hộp thoại Select Wizard, mở rộng Database
3. Kích đúp vào Create Login Wizard
4. Hoàn thành các bước tiếp theo.

Ngoài ra, chúng ta cũng có thể sử dụng `sp_addlogin` để tạo một login mới.

Ví dụ:

EXEC `sp_addlogin` 'phnhung', '123', 'Flight Information'

7.2 Bài tập

1. Sử dụng một biến để lưu tình trạng đặt vé máy bay. Thực hiện truy vấn để đưa ra thông tin về mã số khách hàng (PNR_No), Số vé (Ticket_No), Tên khách hàng (Name) có tình trạng vé máy bay (Status) trong bảng Reservation bằng giá trị của biến nhập vào.
2. Hiển thị tất cả các tên khách hàng trong kiểu chữ in hoa.
3. Thực hiện các câu lệnh sau để biết kết quả:

```
SELECT DATENAME (DW, GETDATE ())  
SELECT DATENAME (DY, GETDATE ())  
SELECT DATENAME (YYYY, GETDATE ())  
SELECT DATENAME (QUARTER, GETDATE ())  
SELECT DATENAME (HH, GETDATE ())  
SELECT DATEADD (DAY, 25, GETDATE ())
```

Chương 7. T-SQL PROGRAMMING, TRANSACTIONS, SECURITY – Phần thực hành

4. Thực hiện một số câu lệnh sau và cho biết kết quả:

```
SELECT @@Language
SELECT @@Servicename
SELECT @@Servername
SELECT @@Rowcount
SELECT @@Connections
```

5. Thêm một tên login mới sử dụng thủ tục hệ thống sp_addlogin.

8 Chương 8. T-SQL VÀ SQL NÂNG CAO

8.1 Giới thiệu sơ lược về T- SQL (Transact -SQL)

Phần này chúng ta đã được tìm hiểu trong phần trước khi học về ngôn ngữ SQL, nên ở đây tôi chỉ giới thiệu một số ví dụ đơn giản.

Transact-SQL là ngôn ngữ SQL mở rộng dựa trên SQL chuẩn của ISO (International Organization for Standardization) và ANSI (American National Standards Institute) được sử dụng trong SQL Server khác với P-SQL (Procedural-SQL) dùng trong Oracle.

Trong bài này chúng ta sẽ tìm hiểu sơ qua về 3 nhóm lệnh T-SQL được thực hiện trong SQL Server.

8.1.1 Data Definition Language (DDL)

Đây là những lệnh dùng để quản lý các thuộc tính của một cơ sở dữ liệu như định nghĩa các hàng hoặc cột của một bảng, hay vị trí data file của một cơ sở dữ liệu...thường có dạng:

CREATE object_Name

ALTER object_Name

DROP object_Name

Trong đó *object_Name* có thể là một table, view, stored procedure, indexes...

Chúng ta xem xét một số ví dụ sau:

Lệnh **Create** sau sẽ tạo ra một bảng tên STUDENT với 3 cột StID, StName, StAddress.

```
USE Student_Management
CREATE TABLE Student (
    StID int NOT NULL PRIMARY KEY,
    StName varchar(40) NOT NULL,
    StAddress varchar(40)
)
```

Lệnh **Alter** sau đây cho phép ta thay đổi định nghĩa của một bảng như thêm(hay bớt) một cột hay một Constraint...Trong ví dụ này ta sẽ thêm cột StClass vào table Student.

```
USE Student_Management
ALTER TABLE Student
ADD StClass varchar(20)
```

Lệnh **Drop** sau đây sẽ hoàn toàn xóa bảng khỏi cơ sở dữ liệu

```
USE Northwind
```

```
DROP TABLE Student
```

8.1.2 Data Control Language (DCL):

Đây là những lệnh quản lý các quyền truy cập lên từng object (table, view, stored procedure...). Thường có dạng sau:

- **GRANT**
- **REVOKE**
- **DENY**

Ví dụ:

Lệnh sau sẽ cho phép user trong Public Role được quyền Select đối với table Student trong cơ sở dữ liệu Student_Management (Role là một khái niệm giống như Windows Group sẽ được bàn kỹ trong phần Security).

```
USE Student_Management
GRANT SELECT ON Customers TO PUBLIC
```

Lệnh sau sẽ từ chối quyền Select đối với table Student trong cơ sở dữ liệu Student_Management của các User trong Public Role.

```
USE Student_Management
DENY SELECT ON Student TO PUBLIC
```

Lệnh sau sẽ xóa bỏ tác dụng của các quyền được cho phép hay từ chối trước đó.

```
USE Student_Management
REVOKE SELECT ON Student TO PUBLIC
```

8.2 Data Manipulation Language (DML):

Đây là những lệnh phổ biến dùng để xử lý data: Update, Insert, Delete.

Sau đây là một số ví dụ:

Insert

```
USE Student_Management
INSERT INTO Student
VALUES ( 'TL01', 'Phạm Đình Thuận', 'Hà nội', '43th' )
```

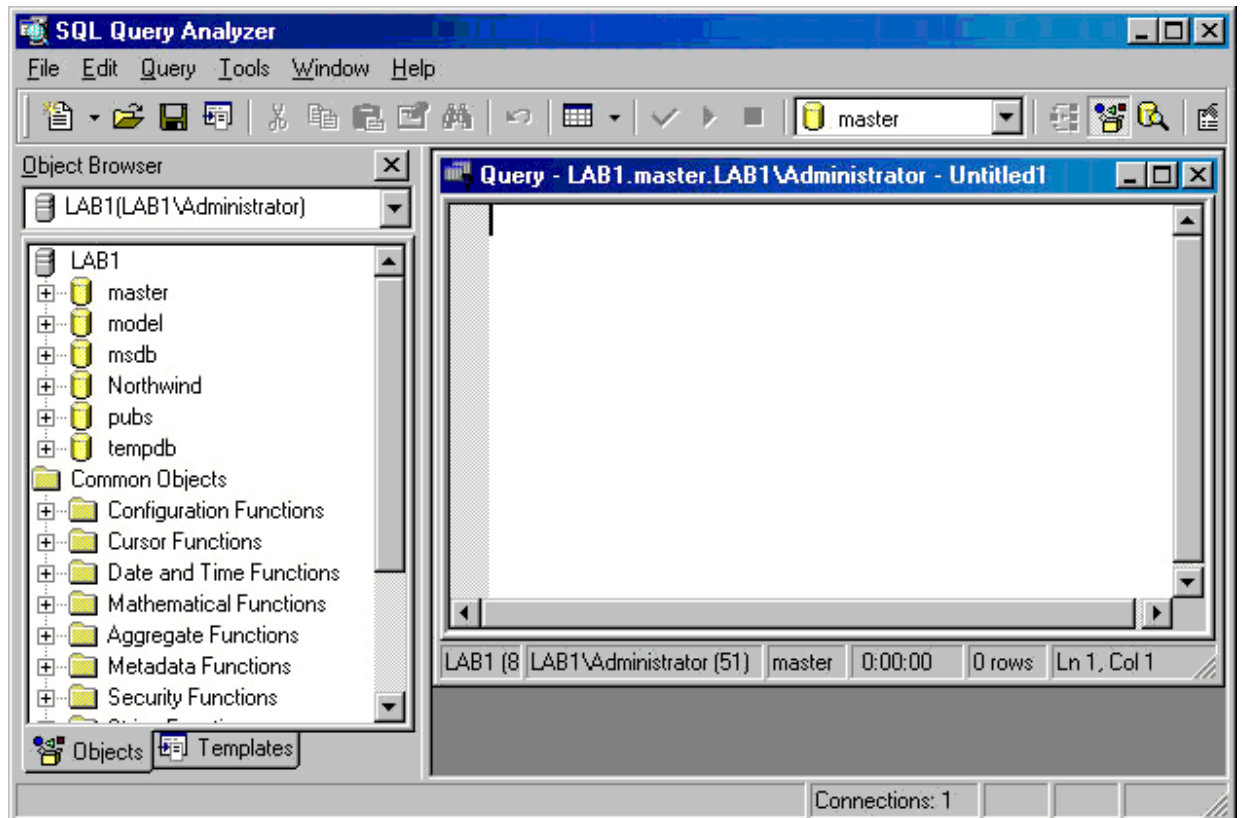
Update

```
USE Student_Management
UPDATE Student
SET StClass = '45th'
WHERE STId = 'TL01'
```

Delete

```
USE USE Student_Management
DELETE FROM Student
WHERE STId = 'TL01'
```

Để chạy các câu lệnh thí dụ ở trên bạn cần sử dụng Query Analyser để soạn thảo câu lệnh và thực thi.



Hình 8.1. Cửa sổ soạn thảo và thực hiện câu lệnh SQL

8.3 Các câu lệnh truy vấn dữ liệu

Các câu lệnh SQL đơn giản có cú pháp và cách thực hiện tương tự như đã giới thiệu. Sau đây là một số câu lệnh bổ sung và nâng cao.

8.3.1 Thực hiện Join để kết nối các bảng

Ta có thể thực hiện lấy dữ liệu từ hai bảng thông qua từ khóa JOIN.

8.3.1.1 INNER JOIN:

Cú pháp:

Chương 8. T-SQL VÀ SQL NÂNG CAO

```
SELECT field1, field2, field3
FROM table1
INNER JOIN table2
ON table1.keyfield=table2.foreign_keyfield
```

Ví dụ: Giả sử có hai bảng:

KHACHHANG:

MaKH	TenKH
01	Hoàng Thanh Vân
02	Lê Thị Nhân
03	Phan Thanh Hòa
04	Phạm Hồng Thanh

DONHANG:

MaSP	TenSP	MaKH
H102	Máy in	01
H106	Bàn	03
H301	Ghế	03

Yêu cầu: Đưa ra tên khách hàng và tên sản phẩm khách hàng đó mua.

```
SELECT KHACHHANG.TenKH, DONHANG.TenSP
FROM KHACHHANG
INNER JOIN DONHANG
ON KHACHHANG.MaKH=DONHANG.MaKH
```

Kết quả:

TenKH	TenSP
Hoàng Thanh Vân	Máy in
Phan Thanh Hòa	Bàn
Phan Thanh Hòa	Ghế

INNER JOIN trả về tất cả các dòng từ hai bảng thỏa mãn điều kiện. Nếu những dòng dữ liệu có bên table1 mà không có trong table2 thì sẽ không được hiển thị.

8.3.1.2 LEFT OUTER JOIN

Cú pháp:

```
SELECT field1, field2, field3
FROM table1
LEFT OUTER JOIN table2
ON table1.keyfield = table2.foreign_keyfield
```

Ví dụ:

```
SELECT KHACHHANG.TenKH, DONHANG.TenSP
FROM KHACHHANG
LEFT OUTER JOIN DONHANG
ON KHACHHANG.MaKH=DONHANG.MaKH
```

Kết quả:

TenKH	TenSP
Hoàng Thanh Vân	Máy in
Lê Thị Nhân	
Phan Thanh Hòa	Bàn
Phan Thanh Hòa	Ghế
Phạm Hồng Thanh	

LEFT OUTER JOIN trả về tất cả các dòng có ở bảng thứ nhất, mặc dù ở bảng thứ hai không thỏa mãn phép toán. Nếu dữ liệu có ở bảng thứ nhất mà không có ở bảng thứ hai thì dữ liệu vẫn hiển thị.

8.3.1.3 RIGHT OUTER JOIN

Cú pháp

```
SELECT field1, field2, field3
FROM table1
RIGHT OUTER JOIN table2
ON table1.keyfield = table2.foreign_keyfield
```

Ví dụ

```
SELECT KHACHHANG.TenKH, DONHANG.TenSP
FROM KHACHHANG
RIGHT OUTER JOIN DONHANG
ON KHACHHANG.MaKH=DONHANG.MaKH
```

Kết quả:

TenKH	TenSP
Hoàng Thanh Vân	Máy in
Phan Thanh Hòa	Bàn
Phan Thanh Hòa	Ghế

RIGHT OUTER JOIN trả về tất cả các dòng có ở bảng 2, mặc dù bảng 1 không thỏa mãn phép toán. Nếu dữ liệu có ở bảng 2 mà không có ở bảng 1 thì vẫn được hiển thị.

8.3.1.4 FULL OUTER JOIN

Dùng Full Outer Join để đưa dữ liệu từ 2 hay nhiều bảng trong đó tất cả cột bên bảng thứ nhất và thứ hai đều được chọn. Các giá trị bên hai bảng trùng nhau thì chỉ lấy một lần.

Ví dụ:

```
USE Pubs
SELECT a.Au_fname, a.Au_lname, p.Pub_name
FROM Authors a FULL OUTER JOIN Publishers p
ON a.City = p.City
ORDER BY p.Pub_name ASC, a.Au_lname ASC, a.Au_fname ASC
```

8.3.1.5 CROSS JOIN

Dùng Cross Join ghép dữ liệu từ hai bảng trong đó số hàng thu được bằng với số hàng của bảng thứ nhất nhân với số hàng của bảng thứ hai.

Ví dụ:

```
USE pubs
SELECT au_fname, au_lname, pub_name
FROM authors CROSS JOIN publishers
WHERE authors.city = publishers.city
ORDER BY au_lname DESC
```

8.3.2 Mệnh đề Top n:

Nếu ta muốn select n hàng đầu tiên mà thôi ta có thể dùng từ khoá Top. Nếu có thêm ORDER BY thì kết quả sẽ được order trước sau đó mới select. Chúng ta cũng có thể select số hàng dựa trên phần trăm bằng cách thêm từ khoá Percent. Ví dụ sau sẽ select 10 hàng đầu tiên theo thứ tự:

```
SELECT DISTINCT TOP 10 ShipCity, ShipRegion
FROM Orders
ORDER BY ShipCity
```

8.3.3 Mệnh đề INTO

INTO Clause cho phép ta lấy dữ liệu từ một hay nhiều bảng, sau đó kết quả sẽ được insert vào một bảng mới. Bảng mới này được tạo ra do kết quả của câu lệnh SELECT INTO.

Ví dụ:

```
SELECT FirstName, LastName
INTO EmployeeNames
FROM Employers
```

Chương 8. T-SQL VÀ SQL NÂNG CAO

Câu lệnh trên sẽ tạo ra một bảng mới có tên là EmployeeNames với 2 cột là FirstName và LastName. Sau đó kết quả select được từ table Employers sẽ được insert vào bảng mới này. Nếu table EmployeeNames tồn tại SQL Server sẽ báo lỗi. Câu lệnh này thường hay được sử dụng để lấy một lượng dữ liệu lớn từ nhiều bảng khác nhau vào một bảng mới (thường dùng cho mục đích tạm thời (temporary table)) mà khỏi phải thực thi câu lệnh Insert nhiều lần.

Một cách khác cũng select data từ một hay nhiều bảng và insert vào một bảng khác là dùng "**Insert Into...Select...**". Nhưng câu lệnh này không tạo ra một bảng mới. Nghĩa là table đó phải tồn tại trước.

Ví dụ:

```
INSERT INTO EmployeeNames
SELECT FirstName, LastName
FROM Employers
```

Chú ý là không có chữ "**Value**" trong câu Insert này.

8.3.4 Từ khoá UNION(Hợp)

Uninon có nhiệm vụ ghép nối kết quả của 2 hay nhiều truy vấn lại thành một kết quả.

Ví dụ:

Giả sử có table1(ColumnA varchar(10), ColumnB int) và table2(ColumnC varchar(10), ColumnD int). Ta muốn select data từ table1 và ghép với dữ liệu từ table2 để tạo thành một kết quả duy nhất ta làm như sau:

```
SELECT * FROM Table1
UNION ALL
SELECT * FROM Table2
```

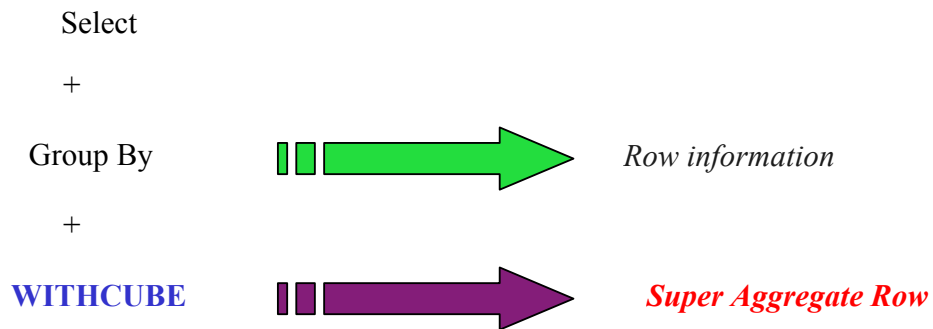
Nếu không có keyword ALL thì những hàng giống nhau từ 2 table sẽ chỉ xuất hiện một lần trong kết quả. Còn khi dùng ALL thì các hàng trong 2 table đều có trong kết quả bất chấp việc lặp lại.

Khi Dùng Union phải chú ý hai chuyện: số cột select ở 2 queries phải bằng nhau và data type của các cột tương ứng phải compatible (tương thích).

8.3.5 Từ khoá CUBE và ROLL UP

Được sử dụng để tổng kết dữ liệu ở mức cao hơn của GROUP BY.

8.3.5.1 CUBE



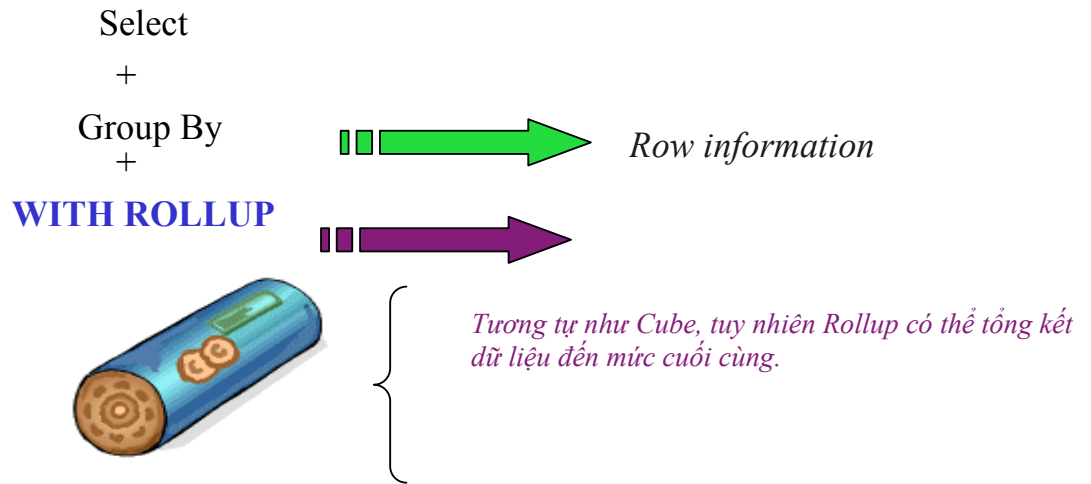
Ví dụ:

```
use Pubs
SELECT Stor_Id, Payterms, SUM(Qty)
AS Total_Quantity
FROM Sales
GROUP BY Stor_id, Payterms
WITH CUBE
```

Kết quả:

<u>Stor_Id</u>	<u>Payterms</u>	<u>Total Quantity</u>
6380	Net 60	5
6380	NULL	5
7066	Net 30	50
7066	NULL	50
7067	Net 30	80
7067	Net 60	10
7067	NULL	90
7131	Net 30	45
7131	Net 60	85
7131	NULL	130
7896	ON invoice	35
7896	NULL	35
8042	Net 30	30
8042	ON invoice	25
8042	NULL	55
NULL	NULL	365
NULL	Net 30	205
NULL	Net 60	100
NULL	ON invoice	60

8.3.5.2 ROLLUP



Ví dụ:

```
SELECT Stor_Id, Payterms, SUM(Qty) AS
Total_Quantity
FROM Sales
GROUP BY Stor_id, Payterms
WITH ROLLUP
```

Kết quả:

Stor_id	Payterms	Total_Quantity
6380	Net 60	5
6380	NULL	5
7066	Net 30	50
7066	NULL	50
7067	Net 30	80
7067	Net 60	10
7067	NULL	90
7131	Net 30	45
7131	Net 60	85
7131	NULL	130
7896	ON invoice	35
7896	NULL	35
8042	Net 30	30
8042	ON invoice	25
8042	NULL	55
NULL	NULL	365



Khi dùng With Rollup thay Cube, dòng này được thêm vào

8.3.6 Mệnh đề COMPUTE và COMPUTE BY

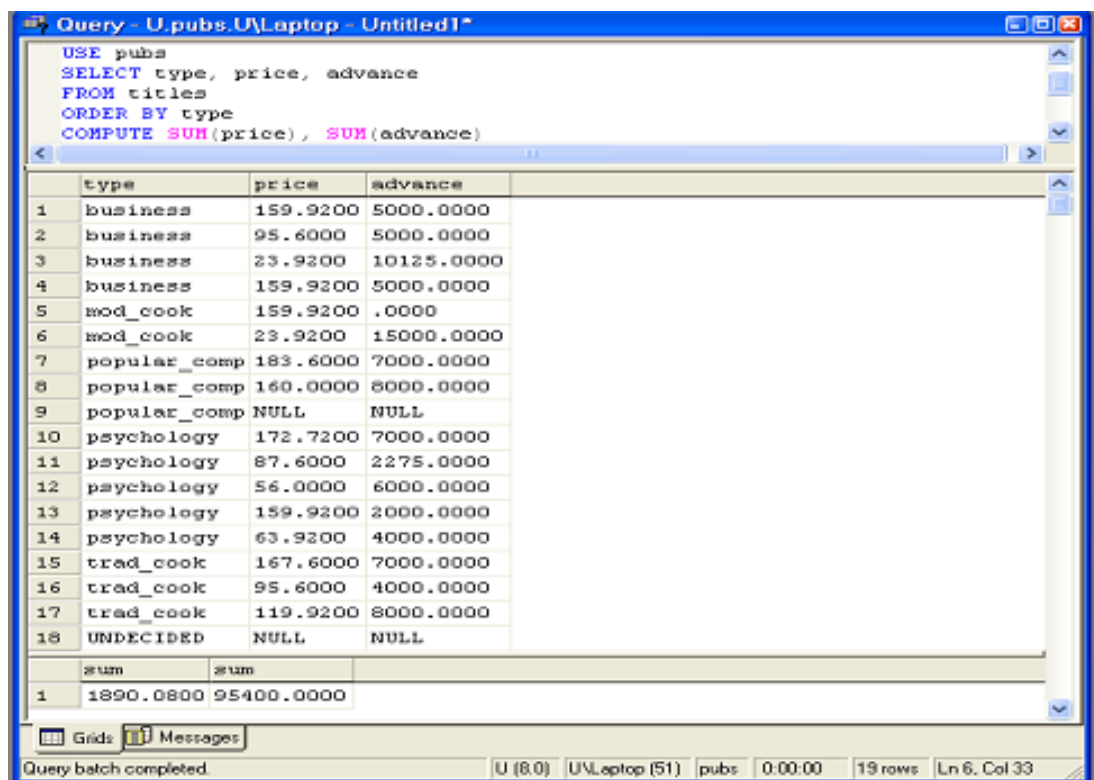
Mệnh đề COMPUTE đưa ra giá trị tổng hợp (thông qua các hàm nhóm) trong một hàng mới.

Kết quả chi tiết và dòng dữ liệu tổng hợp được nhìn thấy trên cùng 1 màn hình kết quả.

Cú pháp:

```
COMPUTE <Expression1>, <Expression2>
```

Ví dụ:



```
USE pubs
SELECT type, price, advance
FROM titles
ORDER BY type
COMPUTE SUM(price), SUM(advance)
```

	type	price	advance
1	business	159.9200	5000.0000
2	business	95.6000	5000.0000
3	business	23.9200	10125.0000
4	business	159.9200	5000.0000
5	mod_cook	159.9200	.0000
6	mod_cook	23.9200	15000.0000
7	popular_comp	183.6000	7000.0000
8	popular_comp	160.0000	8000.0000
9	popular_comp	NULL	NULL
10	psychology	172.7200	7000.0000
11	psychology	87.6000	2275.0000
12	psychology	56.0000	6000.0000
13	psychology	159.9200	2000.0000
14	psychology	63.9200	4000.0000
15	trad_cook	167.6000	7000.0000
16	trad_cook	95.6000	4000.0000
17	trad_cook	119.9200	8000.0000
18	UNDECIDED	NULL	NULL
	sum	sum	
1	1890.0800	95400.0000	

Hình 8.2. Thực hiện cú pháp COMPUTER

Nếu thay mệnh đề COMPUTE bằng COMPUTE BY ta sẽ nhận được kết quả sau:

Chương 8. T-SQL VÀ SQL NÂNG CAO

```
USE pubs
SELECT type, price, advance
FROM titles
ORDER BY type
COMPUTE SUM(price), SUM(advance) BY Type
```

	type	price	advance
1	business	159.9200	5000.0000
2	business	95.6000	5000.0000
3	business	23.9200	10125.0000
4	business	159.9200	5000.0000
	sum	sum	
1	439.3600	25125.0000	

	type	price	advance
1	mod_cook	159.9200	.0000
2	mod_cook	23.9200	15000.0000
	sum	sum	
1	183.8400	15000.0000	

	type	price	advance
1	popular_comp	183.6000	7000.0000
2	popular_comp	160.0000	8000.0000
3	popular_comp	NULL	NULL
	sum	sum	
1	343.6000	15000.0000	

Hình 8.3. Thực hiện cú pháp COMPUTE BY

8.4 Câu hỏi trắc nghiệm

1. Kết quả của truy vấn con có thể là bao nhiêu dòng mà câu lệnh không trả về lỗi?

- A Chỉ một
- B Chỉ một, trừ khi đằng trước nó có toán tử ANY, ALL, EXISTS hoặc IN.
- C Không giới hạn
- D Không giới hạn, trừ khi đằng trước nó có toán tử ANY, ALL, EXISTS hoặc IN.

2. Truy vấn con liên kết (correlated subqueries) không thể đứng độc lập.

- A Đúng
- B Sai

3. Những hàm nào sau đây không sử dụng được với kiểu dữ liệu Character?

- A AVG()
- B SUM()
- C MIN()
- D MAX()

4. Trong câu lệnh SELECT, nếu chứa các từ khoá WHERE, GROUP BY, HAVING thì chúng phải đứng theo thứ tự nào?

- A HAVING, GROUP BY, WHERE
- B WHERE, GROUP BY, HAVING
- C WHERE, HAVING, GROUP BY
- D GROUP BY, WHERE, HAVING

5. Kiểm tra truy vấn sau:

**SELECT batch_id, subject_id, AVG(Marks) FROM batchperformance
GROUP BY batch_id, subject_id WITH ROLLUP**

Kết quả câu lệnh trên là gì?

- A Điểm trung bình của từng khoá học (Batch) trong từng môn học(Subject).
- B Điểm trung bình của tất cả các khoá học trong từng môn học.
- C Điểm trung bình của tất cả các sinh viên của một khoá học trong tất cả môn học.
- D Điểm trung bình của từng khoá học tính theo tất cả các môn học.

6. Mặc định, phép toán UNION giúp ta nhân đôi số lượng của tập kết quả?

- A Đúng
- B Sai

9 Chương 9. T-SQL VÀ SQL NÂNG CAO Phần thực hành

Thực hiện những công việc sau đây bằng Query Analyzer:

1. Hiện tất cả các giá trị khác nhau của *PNR_no* từ bảng *Passenger*
2. Lấy ra 4 hàng đầu tiên từ bảng *Meal*
3. Liệt kê các bản ghi trong bảng *Flight* có mã *Aircraft_code* là 'BA01'
4. Liệt kê *Name*, *PP No*, *Meal Pref* cho các hành khách có *PNR_no* là 1 hoặc 2 từ bảng *Passenger*
5. Hiện thị tất cả các tên hành khách bắt đầu bằng chữ cái 'A'
6. Hiện thị chi tiết tất cả chuyến bay từ thành phố có mã 'NY', sắp xếp theo trường *source*
7. Hiện thị tên của các hành khách nam thêm vào trước tên 'Mr'
8. Hiện thị các thông tin chi tiết về chuyến bay như *aircraft code*, *regular fare*, *discounted fare* cho hạng nhất (FC). Discount (giảm giá) bằng 25% giá thông thường. Tiêu đề các cột là *Aircraft*, *Regular First Class fare*, *Discounted First Class fare*.
9. Hiện thị và sắp xếp chi tiết về các chuyến bay tới thành phố có mã là 'Lon'. Các chuyến bay có thời gian bay ít nhất được hiển thị đầu tiên.
10. Hiện thị các món ăn không phải là ăn chay (*non-vegetarian*) trên các chuyến bay
11. Hiện thị *status_code* và *description* trong bảng **Status_master** với điều kiện chữ cái cuối cùng trong trường *description* khác 'd'
12. Hiện thị *aircraft_code* của đường bay vào Chủ nhật (Sunday) và thứ Tư (Wednesday).
13. Hiện thị tên nước mà chuyến bay *Fly Safe Airways* đã đến, lưu ý là chỉ hiển thị những giá trị khác nhau.
14. Hiện thị số lượng bản ghi được hiển thị bởi truy vấn trước.
15. Hiện thị tên của dịch vụ SQL Server đang chạy trên máy tính.
16. Hiện thị tên của những món ăn (meal) được phục vụ trên British Airways (Sử dụng truy vấn con).
17. Hiện thị tên những hãng bay (airlines) có những chuyến bay (flights) nội địa xuất phát từ Mumbai (Sử dụng truy vấn con).
18. Hiện thị tuổi trung bình của những hành khách là nữ.

Chương 9. T-SQL và SQL nâng cao -Phần thực hành

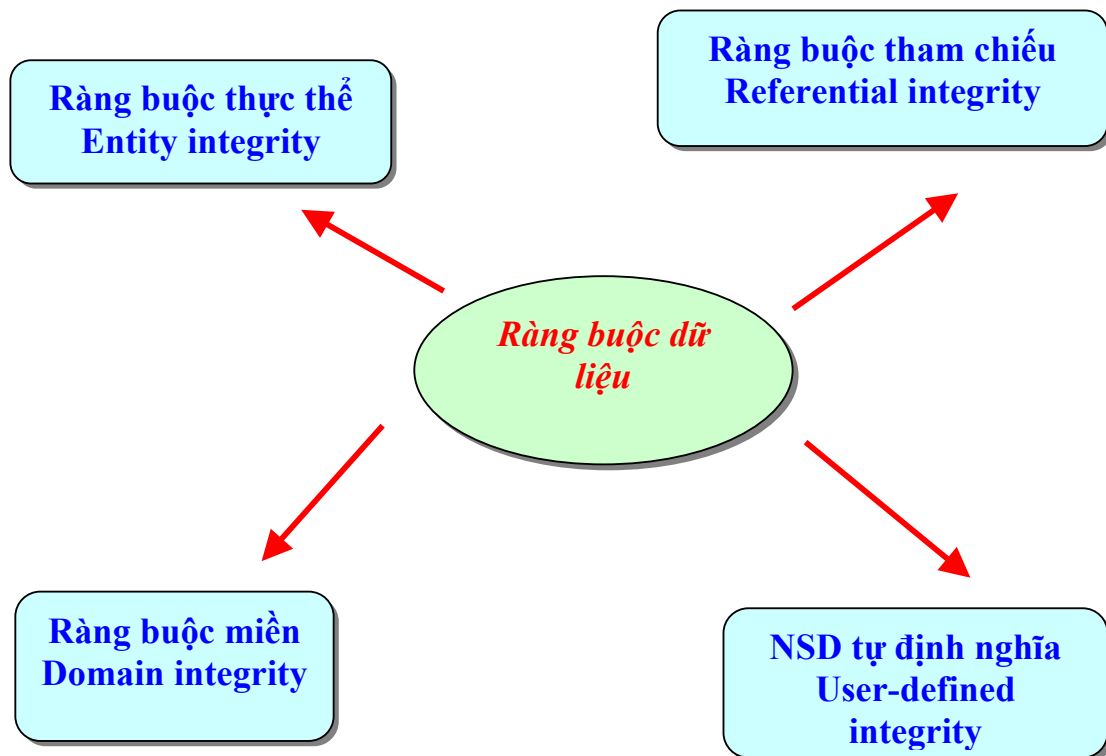
19. Tìm tổng số hành khách được Fly Safe Airways phục vụ.
20. Tính thời gian bay nhỏ nhất, lớn nhất, và trung bình của những chuyến bay đến thành phố có mã số là NY.
21. Hiện thị số vé (tickets) được đặt của từng PNR_no.
22. Hiện thị số lượng các chuyến bay mỗi tuần của từng Aircraft.
23. Hiện thị những aircraft có tổng số ghế phục vụ >500 ghế.
24. Hiện thị những PNR number và tuổi của người già nhất. Lưu ý, chỉ hiện thị những bản ghi có tuổi người già nhất >35.
25. Hiện thị số lượng những lựa chọn bữa ăn có Non-vegetarian trên mỗi hãng bay (airline).
26. Hiện thị số lượng của những hành khách có tuổi lớn hơn 40 của từng PNR number. Phải đảm bảo rằng tất cả các PNR number đều được hiện thị.
27. Hiện thị Airline code(mã hãng bay), destination city code (mã số của thành phố đích) và số những chuyến bay của hãng đó bay tới. Kết quả bao gồm cả tổng số những chuyến bay của từng hãng và tổng số của tất cả các chuyến bay trong cơ sở dữ liệu.

10 Chương 10. RÀNG BUỘC DỮ LIỆU VÀ CHỈ SỐ

10.1 Ràng buộc dữ liệu

10.1.1 Giới thiệu

Như chúng ta đã biết, thực hiện những ràng buộc dữ liệu giúp tất cả các giá trị của dữ liệu được lưu trữ đúng đắn. Tất cả dữ liệu được thêm vào cơ sở dữ liệu đều phải thỏa mãn các ràng buộc. Sau đây là một số loại ràng buộc dữ liệu:



Hình 10.1. Các loại ràng buộc dữ liệu

10.1.2 Ràng buộc thực thể

Xác định một dòng dữ liệu là duy nhất trong một bảng. Còn được biết đến như ràng buộc dòng dữ liệu (Row Integrity).

SQL Server có một số công cụ thực hiện ràng buộc thực thể như sau:

- PRIMARY KEY constraint
- UNIQUE constraint
- IDENTITY property

10.1.3 Ràng buộc miền dữ liệu

Là tập hợp những dữ liệu được phép nhập vào một cột trong bảng.

Công cụ thực hiện:

- DEFAULT definition
- FOREIGN KEY constraint
- CHECK constraint
- NOT NULL property
- Rules

10.1.4 Ràng buộc tham chiếu

Là ràng buộc dữ liệu giữa khoá chính của một bảng với khoá ngoại của một bảng khác.

Công cụ thực hiện:

- FOREIGN KEY constraint
- CHECK constraint

10.1.5 Ràng buộc NSD tự định nghĩa

Cho phép người quản trị thêm vào những ràng buộc để đảm bảo sự đúng đắn của dữ liệu.

Công cụ thực hiện:

- Tất cả SQL Server constraints
- Stored Procedures
- Triggers

10.2 Thực hiện các ràng buộc bằng T-SQL

- Được thiết đặt trên một hoặc một tập hợp các cột của bảng.
- Nhằm thiết đặt những giới hạn cho việc nhập giá trị cho cột dữ liệu.
- Có thể được định nghĩa ngay khi tạo bảng hoặc sửa cấu trúc bảng.

10.2.1 PRIMARY KEY Constraint

Thiết đặt một hoặc tập hợp các cột làm khoá chính của bảng.

Cú pháp:

```
CREATE TABLE Table_name  
(<Column_definition> PRIMARY KEY)
```

Ví dụ:

Chương 10. RÀNG BUỘC DỮ LIỆU VÀ CHỈ SỐ

```
CREATE TABLE Reservation_copy
( PNR_no int PRIMARY KEY )
```

10.2.2 UNIQUE Constraint

Quy định cột này phải có giá trị khác nhau trên mỗi dòng

Cú pháp:

```
CREATE TABLE Table_name
(<Column_definition> UNIQUE )
```

Ví dụ:

```
CREATE TABLE passenger_copy
( [PP no] VARCHAR(20) UNIQUE)
```

10.2.3 IDENTITY Property

Quy định giá trị của một cột nào đó trong bảng là tự động

- seed_value: giá trị ban đầu
- increment_value: giá trị tăng

Cú pháp:

```
CREATE TABLE Table_name
(Column_name Data_Type IDENTITY
[(<seed value>, increment value)])
```

Ví dụ:

```
CREATE TABLE Reservation_Copy
(ticket no INT IDENTITY(1,1))
```

10.2.4 DEFAULT Definition

Thiết đặt giá trị mặc định cho một cột nào đó. Nếu người sử dụng không nhập giá trị cho cột này thì nó sẽ nhận giá trị mặc định.

Cú pháp:

```
CREATE TABLE Table_name
(Column_name Data_Type DEFAULT default_value)
```

Ví dụ:

```
CREATE TABLE employee
(employee_cd char(4),
employee_nm varchar(50),
grade char(2),
hra varchar(10) default 'N.A.')
```

Chương 10. RÀNG BUỘC DỮ LIỆU VÀ CHỈ SỐ

10.2.5 FOREIGN Key Constraint

Chỉ ra một cột làm khoá ngoại của bảng (nhằm liên kết dữ liệu trong hai bảng)

Cú pháp:

```
CREATE TABLE Table_name
  Column_name Data_Type, .....
  FOREIGN KEY (Column_name) REFERENCES Primarykey_Tablename)
```

Ví dụ:

```
CREATE TABLE Passenger
  (PNR_no int, ticket_no int, name varchar(15),
  .....
  FOREIGN KEY (PNR_no)
    REFERENCES Reservation)
```

10.2.6 CHECK Constraint

Giới hạn dữ liệu được lưu trữ trong cột.

Cú pháp:

```
CREATE TABLE Table_name
  (Column_name Data_Type CHECK (value1, value2, ..), ..)
```

Ví dụ:

```
CREATE TABLE Reservation
  (.., class_code char(3) CHECK('EX', 'FC', 'E'), ..)
```

10.2.7 NOT NULL Constraint

Nếu một trường nào được quy định là NOT NULL, tức là không rỗng thì người sử dụng bắt buộc phải nhập dữ liệu cho trường này.

Cú pháp:

```
CREATE TABLE Table_name
  (Column_name Data_Type NOT NULL, .....)
```

Ví dụ:

```
CREATE TABLE Passenger
  (....., name varchar(15) NOT NULL, .....)
```

10.2.8 Rules

Tương tự như CHECK constraints, nhưng Rules được tạo như một đối tượng độc lập, sau đó mới thiết đặt cho cột dữ liệu.

Cú pháp:

```
CREATE RULE rule_name AS condition_expression
:
SP_BINDRULE rule_name, table_name.column_name
```

Ví dụ:

```
CREATE RULE check_PNR
AS @pnr BETWEEN 1 AND 500
:
SP_BINDRULE check_PNR, Reservation.PNR_no
```

10.3 Indexes

10.3.1 Giới thiệu

Khái niệm về index trong cơ sở dữ liệu tương tự như phần index của một cuốn sách. Khi cần tìm kiếm thông tin trên cuốn sách, ta không cần phải lật tất cả các trang trong đó, mà chỉ cần vào phần Index ở cuối quyển và dò xem thông tin mà ta cần nằm ở những trang nào. Vì lý do đó, tìm kiếm thông tin thông qua index sẽ nhanh hơn nhiều lần.

Trong cơ sở dữ liệu cũng vậy, hỗ trợ index cho phép người dùng tìm dữ liệu mà không cần quét toàn bộ bảng.

Index được dùng để tìm ra giá trị duy nhất. Mục đích của index để xác định dòng nào đang chứa dữ liệu cần tìm.

Index được sử dụng đúng cách có thể tăng hiệu quả thực hiện trên CSDL bằng cách giảm thời gian truy cập.

Index có thể được tạo bởi 1 hoặc nhiều trường.

SQL Server tự động tạo chỉ số cho những trường được xác định là trường khoá hoặc ràng buộc duy nhất (UNIQUE)

Tuy nhiên, những Tables có indexes yêu cầu nhiều vùng trống trên đĩa trong CSDL.

Những lệnh cần thực hiện dữ liệu yêu cầu nhiều thời gian hơn bởi vì cần phải cập nhật index. Vì thế, **Indexes thực sự là con dao 2 lưỡi, nếu không sử dụng đúng cách, nó sẽ làm giảm tốc độ của hệ thống.**

10.3.2 Lời khuyên khi sử dụng indexes

Chúng ta chỉ **nên** thiết đặt index trên những cột:

- Được sử dụng thường xuyên cho việc tìm kiếm.
- Khi cột được dùng để sắp xếp dữ liệu

và **không nên** áp dụng trong những trường hợp:

- Khi cột chỉ chứa đựng vài giá trị khác nhau.
- Khi bảng chỉ có vài dòng .

10.3.3 Tạo Indexes

Lệnh tạo chỉ số trên bảng: CREATE INDEX...

Chỉ có những người dùng làm chủ bảng mới có quyền tạo chỉ số cho bảng.

Cú pháp:

```
CREATE [UNIQUE] [CLUSTERED|NONCLUSTERED]
INDEX index_name
ON table_name
(column_name[, column_name]...)
[WITH
 [PAD_INDEX]
 [[,]FILLFACTOR=x]
 [[,]DROP_EXISTING]
]
```

Trong đó:

Fill factor được tạo ngay khi tạo index.

Khi index được tạo, bảng dữ liệu (table data) được lưu trữ trong trang dữ liệu (data page) theo thứ tự của giá trị trong cột được chỉ số. Khi bản ghi mới được thêm vào bảng hoặc một giá trị trong bảng được thay đổi, SQL Server phải tổ chức vùng trống cho bản ghi mới và cập nhật lại thứ tự sắp xếp của dữ liệu.

Trong khi thêm bản ghi mới vào trang chỉ số đã đầy (full index page), SQL Server sẽ di chuyển gần đúng nửa số bản ghi của bảng sang trang mới để tạo ra vùng trống cho việc thêm bản ghi, quá trình này gọi là phân trang (page splits).

Khi ở đó sẽ không có sự thay đổi dữ liệu, xác định giá trị đó là 100 thì trang sẽ đầy và sẽ chiếm một phần rất nhỏ bộ nhớ. Khi ở đó có sự thay đổi dữ liệu thường xuyên dữ liệu trong bảng, xác định giá trị của Fill Factor thấp để có nhiều hơn vùng trống cho trang dữ liệu.

10.3.4 Các kiểu Indexes

Có 2 kiểu Indexes:

- Clustered index: xác định thứ tự lưu trữ vật lý của dữ liệu trong bảng
- Non-clustered index: xác định sắp xếp logic của dữ liệu.

10.3.4.1 Clustered index

Một Table chỉ có thể có 1 clustered index.

Index có thể kết hợp nhiều cột (multiple columns).

Ví dụ:

```
CREATE CLUSTERED INDEX CLINDX_titleid ON roysched
(title_id)
```

10.3.4.2 Non-clustered index

- Non clustered xác định thứ tự logic của dữ liệu.
- Dữ liệu được lưu ở một vùng, index được lưu ở vùng khác, có con trỏ trỏ đến vùng lưu trữ dữ liệu.
- Một Table có thể có nhiều non-clustered indexes, lớn nhất 249.
- Hướng dẫn tạo Non-clustered Indexes:
- Chỉ nên thêm chỉ số khi nó thực sự cần thiết.
- Chỉ nên chỉ số những cột được truy cập thường xuyên.
- Mặc định, câu lệnh CREATE INDEX tạo ra non-clustered index.
- Sử dụng non-clustered index cho những cột có nhiều giá trị khác nhau, ví dụ như kết hợp last name và first name (nếu như clustered index đã được sử dụng cho những cột khác).

Ví dụ:

```
CREATE NONCLUSTERED INDEX NCLINDX_ordnum ON sales
(ord_num)
```

10.3.5 Tính chất của Indexes

Clustered and non-clustered indexes có thể được tạo như Unique hoặc Composite.

Unique indexes: không cho phép giá trị trùng nhau trong cột index.

Composite indexes: cho phép hai hoặc nhiều cột kết hợp để tạo ra index.

Hướng dẫn tạo Unique Indexes:

- Toàn vẹn thực thể được đảm bảo bằng unique indexes vì giá trị duy nhất tồn tại trong mỗi dòng.
- A NULL value trong key column được coi như là unique value.
- Unique index không thể được tạo trên cột chứa giá trị trùng nhau. Giá trị trùng nhau phải được xóa trước khi unique index được tạo.

10.3.6 Hiện thị Indexes

Sau khi tạo ra indexes, chúng ta có lẽ cần biết thông tin về về indexes.

Chúng ta có thể muốn nhìn thấy những indexes được tạo trên bảng và những cột tạo nên indexes trong bảng.

sp_helpindex là system stored procedure đưa ra những thông tin về indexes trên bảng.

Cú pháp:

```
sp_helpindex <Table name>
```

10.3.7 Cách sử dụng Indexes

SQL Server sử dụng Query Optimizer để lựa chọn cách nào là tốt nhất để thực hiện truy vấn, bao gồm indexes nào được sử dụng.

Tuy nhiên, chúng ta có thể yêu cầu truy vấn thực hiện trên index nào thông qua:

Cú pháp:

```
(INDEX=index_name)
```

Ví dụ:

```
SELECT * FROM sales(INDEX =nclindx_ordnum)  
WHERE ord_num = 'P3087a'
```

10.3.8 Xóa Indexes

Indexes không cần thiết nữa có thể được xóa khỏi CSDL để giải phóng vùng nhớ bằng cách sử dụng lệnh DROP INDEX.

Cú pháp:

```
DROP INDEX table name.index name
```

Ví dụ:

```
DROP INDEX sales.NCLINDX_ordnum
```

Chú ý: Câu lệnh DROP INDEX không được áp dụng cho những indexes được tạo bằng PRIMARY KEY hoặc UNIQUE constraints và index trên system table.

10.3.9 Full-text Searches

Full-text indexes được sử dụng trên SQL Server để thực hiện full-text searches.

Indexes có thể xây dựng trên unstructured text để cho phép tìm kiếm text trên những mục xác định.

Chức năng này được cung cấp bằng Microsoft Search Service cho phép thực hiện những tìm kiếm phức tạp sử dụng điều kiện tìm kiếm bằng ngôn ngữ (linguistic search criteria).

Linguistic searches cho phép tìm kiếm từ hoặc cụm từ, những từ mục tiêu được chỉ ra liên quan đến một từ khác, và những dạng khác nhau của từ.

10.3.10 Full-text Catalogs

Tất cả các full-text indexes được lưu trữ trong full-text catalogs.

Full-text catalog là thư mục chỉ có thể được xem bằng Windows và Search Service.

Mặc định, tất cả các full-text indexes trong cơ sở dữ liệu được đặt trong một full-text catalog.

Người quản trị hệ thống có thể chia một catalog thành nhiều catalogs nếu indexes quá lớn.

10.3.11 Sử dụng Full-text Indexes

Full-text indexes có thể được tạo thông qua Enterprise Manager, sử dụng Full-Text Indexing Wizard.

Sau khi tạo, full-text queries có thể được thực hiện trên những bảng đã được tạo indexes.

Người quản trị có thể thực hiện Full-text queries sử dụng hai từ khóa CONTAINS và FREETEXT.

10.4 Câu hỏi trắc nghiệm

1. Trong khi thiết kế bảng, bạn xác định ràng buộc PRIMARY KEY cho cột Emp_No. Kiểu ràng buộc nào sau đây hỗ trợ thực hiện ví dụ trên?

- A User-Defined Integrity
- B Referential Integrity
- C Entity Integrity
- D Domain Integrity

2. Trong khi thiết kế bảng PROJECT, bạn xác định cột Emp_No là khoá ngoại của bảng, được tham chiếu từ bảng EMPLOYEE. Loại ràng buộc nào sau đây được sử dụng?

- A User-Defined Integrity
- B Referential Integrity
- C Entity Integrity
- D Domain Integrity

3. Trong bảng EMPLOYEE, bạn cần có ràng buộc để kiểm tra cột Sex trong bảng chỉ được phép có 1 trong 2 giá trị là True hoặc False. Loại ràng buộc nào sau đây được sử dụng?

- A User-Defined Integrity
- B Referential Integrity
- C Entity Integrity
- D Domain Integrity

4. Hướng dẫn nào sau đây là đúng khi tạo một cột có thuộc tính IDENTITY?

- A Có nhiều hơn 1 cột trong 1 bảng có thể có thuộc tính IDENTITY.
- B Thuộc tính IDENTITY có thể thiết đặt trên cột kiểu dữ liệu là Char.
- C Cột có

5. Khi nói về Indexes, câu nào sai trong số các câu sau đây?

- A Index cho phép người dùng tìm dữ liệu mà không cần quét toàn bộ bảng
- B Index được dùng để tìm ra giá trị duy nhất. Mục đích của index để xác định dòng nào đang chứa dữ liệu cần tìm
- C Index nên được tạo trên tất cả các trường của bảng
- D Chỉ số có thể được tạo bởi 1 hoặc nhiều trường

6. Indexing nên được áp dụng trong những trường hợp nào sau đây?

- A Khi cột chỉ chứa đựng vài giá trị khác nhau
- B Khi cột được sử dụng cho việc tìm kiếm thường xuyên

Chương 10. RÀNG BUỘC DỮ LIỆU VÀ CHỈ SỐ

- C Khi bảng chỉ có vài dòng
- D Khi cột được dùng để sắp xếp dữ liệu

7. Trong những bảng có index, những câu lệnh thực hiện dữ liệu (DML) cần ít thời gian hơn trong những bảng không có index.

- A Đúng
- B Sai

8. Số lượng lớn nhất các cột có thể bao gồm trong index kết hợp?

- A 2
- B 16
- C 4
- D Không giới hạn

11 Chương 11. DATA INTEGRITY AND INDEXES

Phần thực hành

Mục đích:

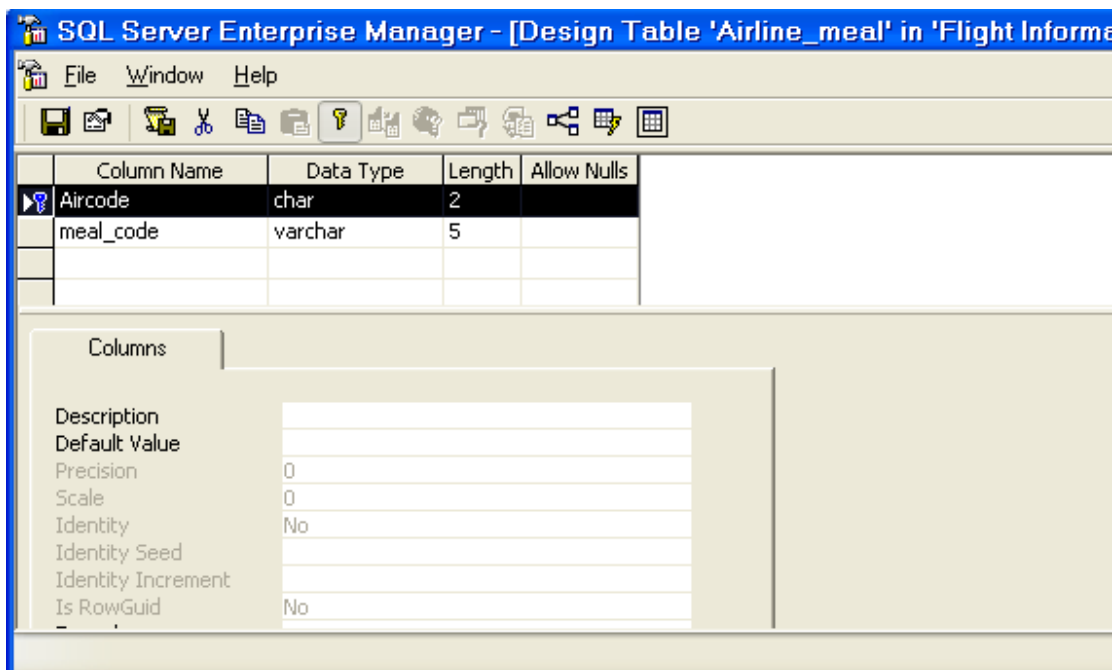
- Sử dụng Enterprise Manager để thực hiện các Constraints và Rules
- Sử dụng Enterprise Manager Wizard để tạo các chỉ số
- Sử dụng QA để thêm các constraints
- Sử dụng QA để tạo các rules và gán các rules cho các đối tượng
- Sử dụng QA để tạo và xem chỉ số

11.1 Hướng dẫn trực tiếp

11.1.1 Tạo ràng buộc PRIMARY KEY

Các bước thực hiện:

- Vào Enterprise Manager
- Vào phần thiết kế của bảng muốn tạo khoá chính
- Bôi đen một trường hoặc nhiều trường làm khoá chính
- Kích vào biểu tượng Khoá trên thanh công cụ



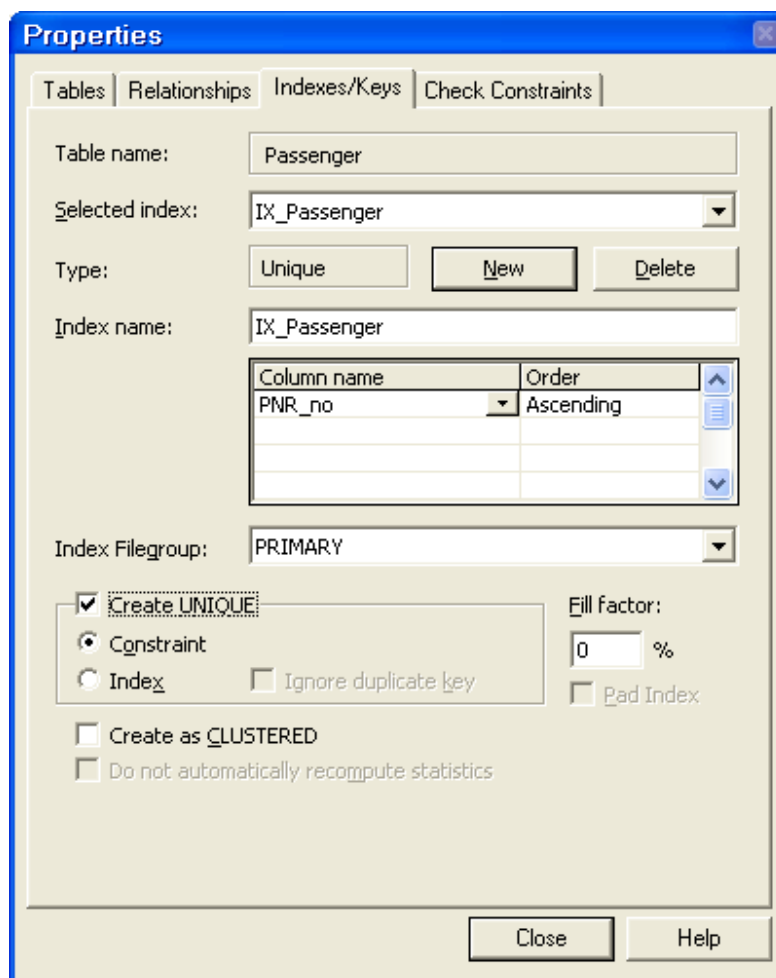
Hình 11.1

11.1.2 Tạo ràng buộc Unique

Ví dụ tạo ràng buộc Unique cho cột PNR_No của bảng Passenger.

Các bước thực hiện:

1. Chọn phần thiết kế của bảng Passenger
2. Chọn cột PNR_No, kích phải chuột, chọn Properties.
3. Chọn thẻ Indexes/Keys
4. Kích nút New
5. Lựa chọn như hình sau



Hình 11.2

6. Kích Close
7. Kích Save để ghi lại thiết kế bảng

11.1.3 Sử dụng thuộc tính IDENTITY

Ví dụ quy định cột PNR_No của bảng Passenger là IDENTITY.

Các bước thực hiện:

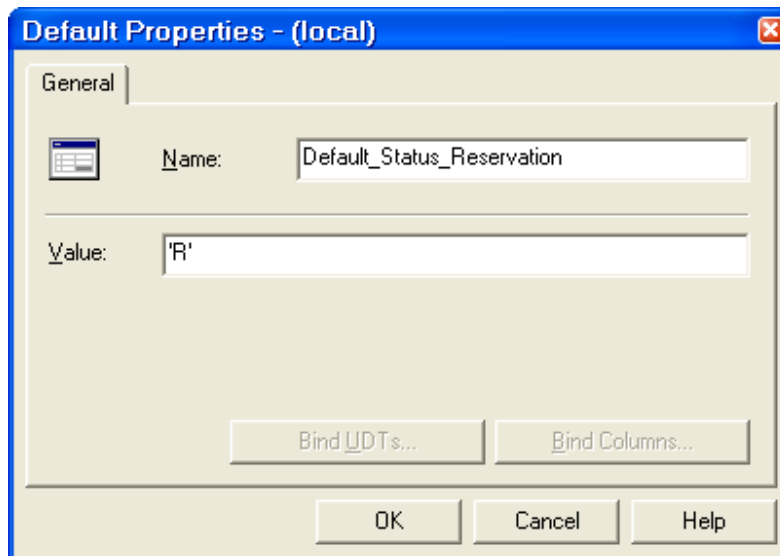
1. Chọn phần thiết kế bảng Passenger
2. Chọn cột PNR_No
3. Trong thẻ Columns bên dưới, chọn thuộc tính Identity = Yes
4. Chọn giá trị khởi đầu trong mục Identity Seed, giả sử=5
5. Chọn giá trị tăng trong mục Identity Increment, giả sử=1 (tăng với chỉ số tăng bằng 1)
6. Kích Save để ghi lại thiết kế bảng

11.1.4 Tạo ràng buộc Default

Giả sử đặt giá trị mặc định cho cột Status trong bảng Reservation = 'R'

Các bước thực hiện:

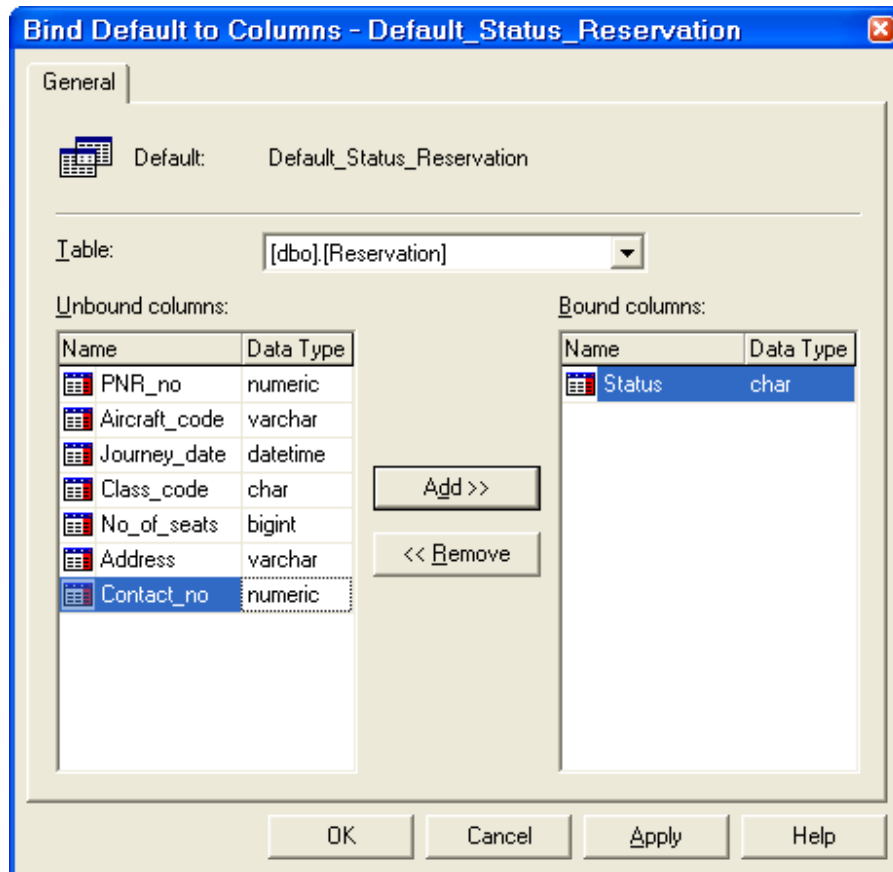
1. Chọn cơ sở dữ liệu Flight Information
2. Kích phải chuột vào đối tượng Default, chọn New Default...
3. Nhập tên của ràng buộc Default vào mục Name
4. Chọn giá trị Default vào mục Value



Hình 11.3

5. Kích phải chuột vào đối tượng Default_Status_Reservation vừa được tạo, chọn Properties

6. Kích vào Bind Columns...
7. Chọn tên bảng, tên cột chấp nhận ràng buộc này như hình sau:
8. Kích Add
9. Kích Apply
10. Kích OK



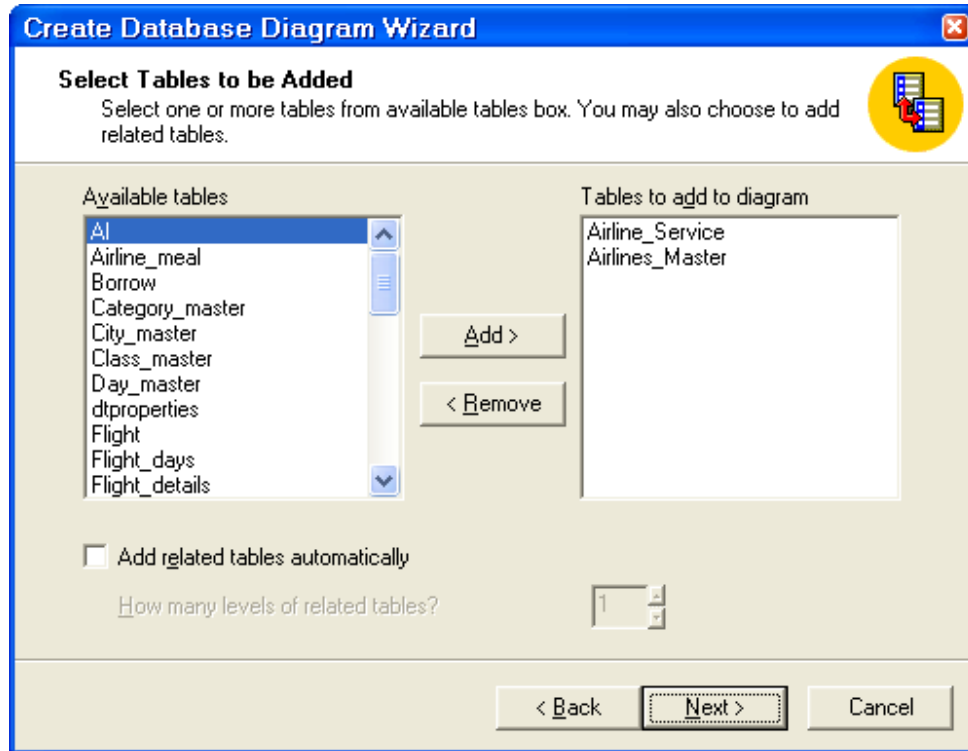
Hình 11.4

11.1.5 Tạo ràng buộc FOREIGN KEY

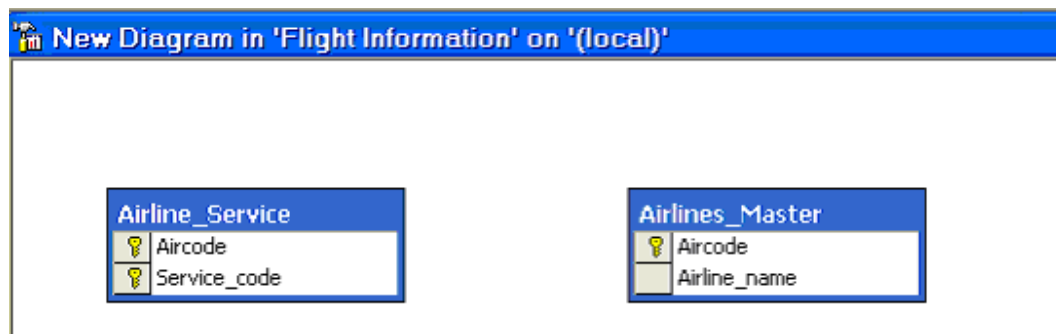
Giả sử cần thiết đặt ràng buộc cho thuộc tính Aircode trong bảng Airline_Service là khoá ngoại, được tham chiếu từ thuộc tính Aircode là khoá chính trong bảng Airline_Master. Thực hiện các bước sau:

1. Chọn cơ sở dữ liệu Flight Information
2. Chọn đối tượng Diagrams
3. Chọn New Database Diagram...
4. Kích Next
5. Chọn 2 bảng Airline_Master và Airline_Service. Kích Add

6. Kích Next
7. Kích Finish

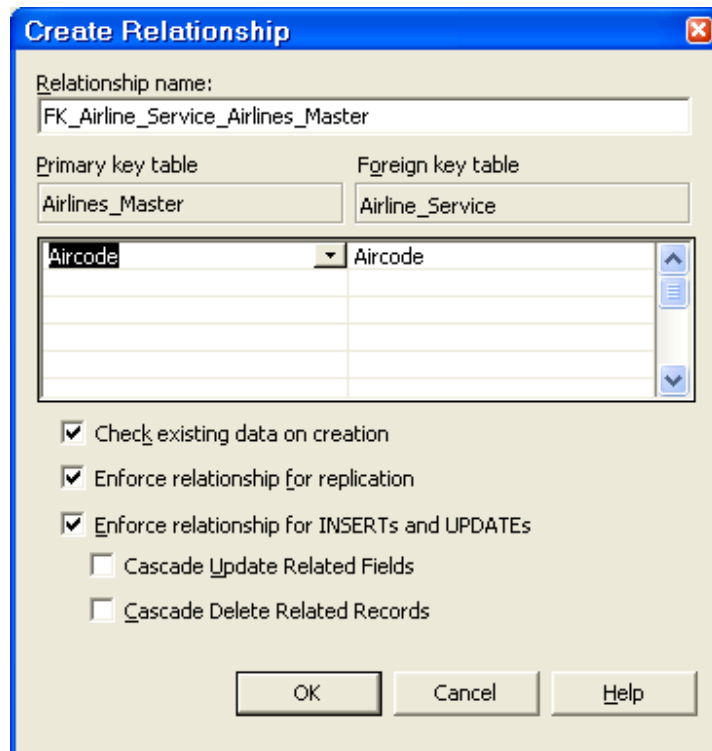


Hình 11.5



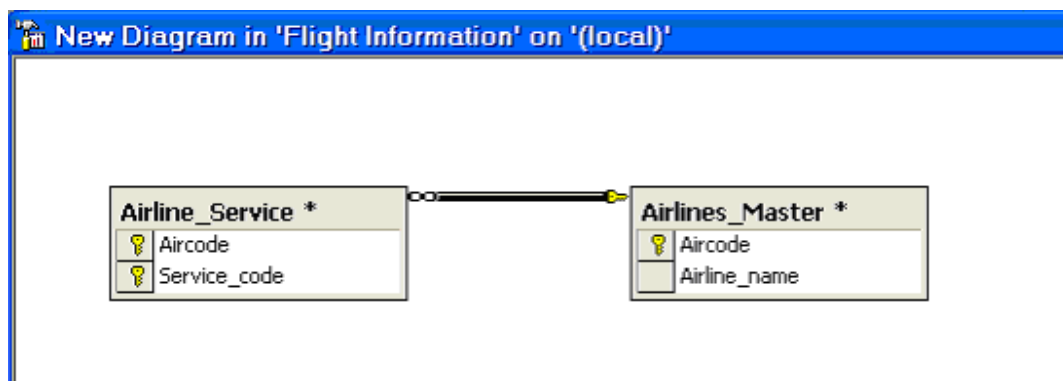
Hình 11.6

8. Chọn thuộc tính Aircode từ bảng Airlines_Master thả sang thuộc tính Aircode trong bảng Airline_Service
9. Hộp thoại Create Relationship xuất hiện, cho phép ta xác định các ràng buộc liên quan khi thiết lập khoá ngoại



Hình 11.7

10. Kích OK



Hình 11.8

11. Ghi lại Diagram.

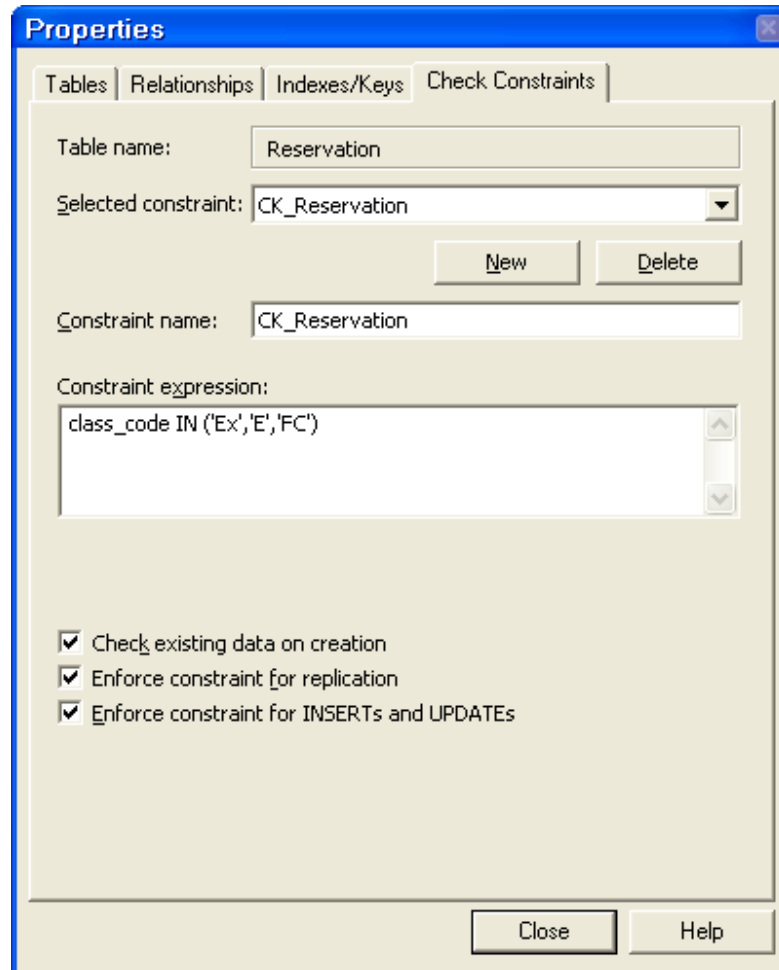
11.1.6 Tạo ràng buộc Check Constraint

Giả sử ta muốn giới hạn dữ liệu nhập vào cho cột class_code trong bảng Reservation chỉ nhận 3 giá trị: 'E', 'Ex', 'FC'. Thực hiện các bước sau:

1. Chọn phân Design của bảng Reservation

Chương 11. RÀNG BUỘC DỮ LIỆU VÀ CHỈ SỐ-Phần thực hành

2. Kích phải vào bất cứ thuộc tính nào, chọn Check Constraints
3. Kích New
4. Nhập tên của ràng buộc và giá trị như hình sau:

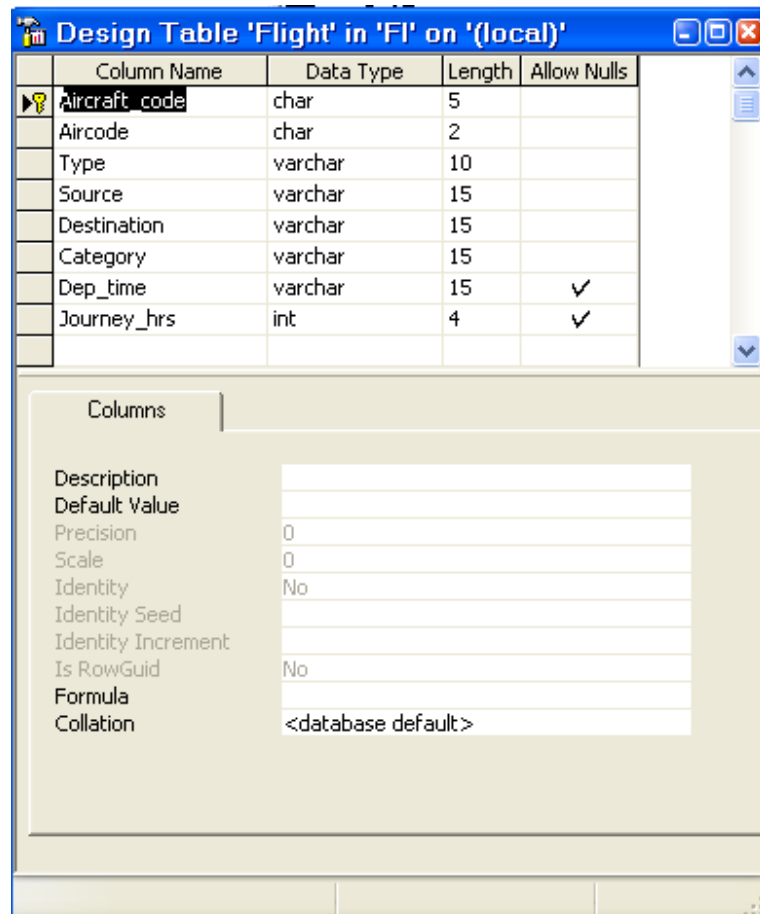


Hình 11.9

5. Kích Close
6. Đóng và ghi lại thiết kế bảng.

11.1.7 Tạo ràng buộc Not Null

1. Chọn phần thiết kế bảng.
2. Un-check vào Allow Nulls cho cột tương ứng.



Hình 11.10

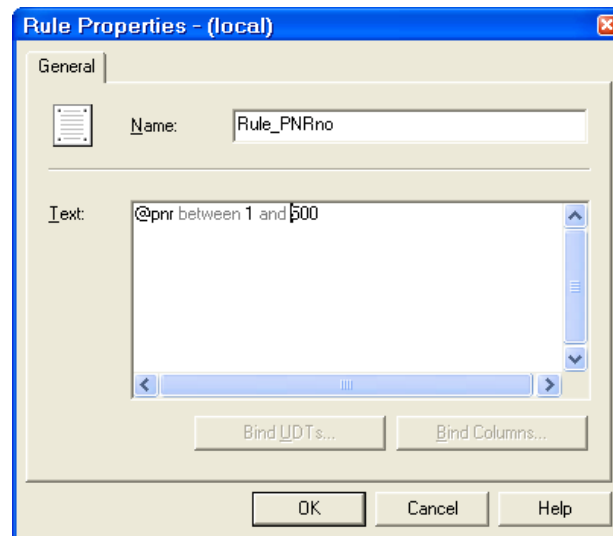
11.1.8 Tạo Rules

Tạo Rule để đảm bảo dữ liệu trong cột PNR_no của bảng Reservation phải nhập trong khoảng 1 và 500.

Các bước thực hiện:

1. Kích phải vào đối tượng Rules, chọn New Rule... từ menu pop-up.
2. Nhập tên của Rule, ví dụ Rule_PNRno.
3. Soạn nội dung của Rule.
4. Kích OK.

Sau khi tạo ra Rule_PNRno, áp dụng Rule này cho cột PNR_no của bảng Reservation.



Hình 11.11

5. Kích phải chuột vào Rule_PNRno, và chọn Properties.
6. Kích Bind Column...
7. Chọn bảng Reservation từ Tables list.
8. Chọn cột PNR_no từ Unbound columns list.
9. Chọn Add>>
10. Kích Apply
11. Kích OK

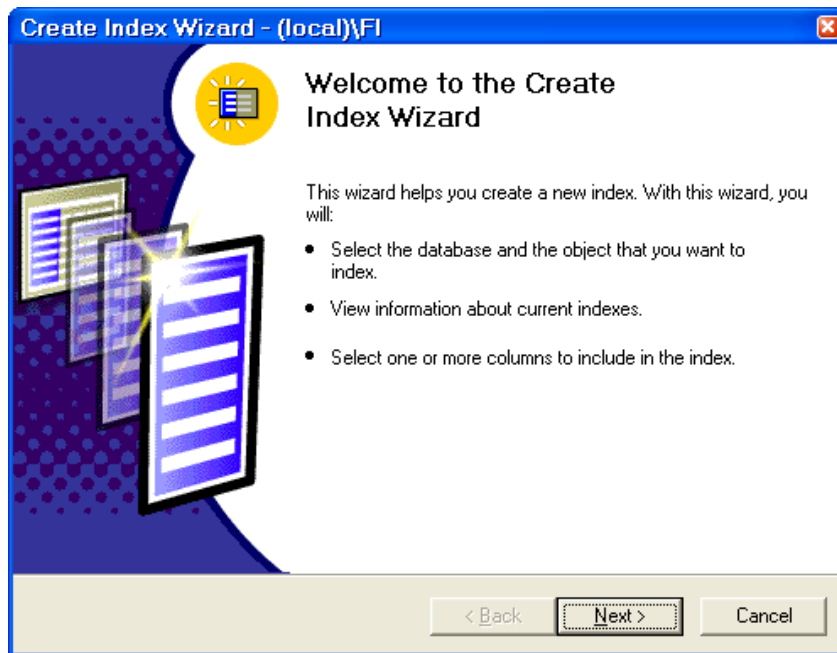
11.2 Indexes

11.2.1 Tạo indexes

Chúng ta có thể tạo Indexes bằng cách sử dụng Create Index Wizard trong EM, hoặc câu lệnh CREATE INDEX. Bây giờ, chúng ta cùng xem xét cách tạo indexes bằng Wizard.

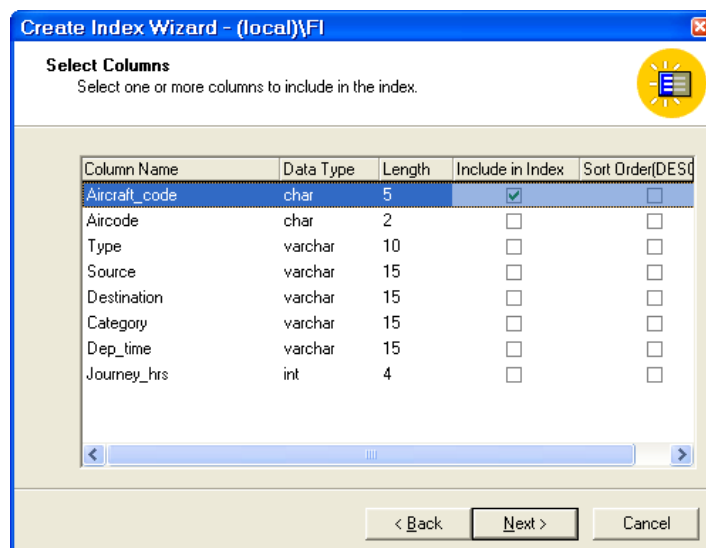
Các bước thực hiện:

1. Kích đúp vào cơ sở dữ liệu, ví dụ FI. Danh sách các đối tượng trong cơ sở dữ liệu FI được hiển thị.
2. Kích Run a Wizard icon trên tool bar hoặc kích vào Tool/ Wizard.
3. Mở rộng lựa chọn Database.
4. Chọn Create Index Wizard, kích OK.



Hình 11.12

5. Kích Next
6. Chọn cơ sở dữ liệu FI từ Database name list.
7. Chọn Flight từ Object name list box.
8. Kích Next.
9. Kích Next.
10. Check vào cột Aircraft_code trong Include in Index như hình dưới.



Hình 11.13

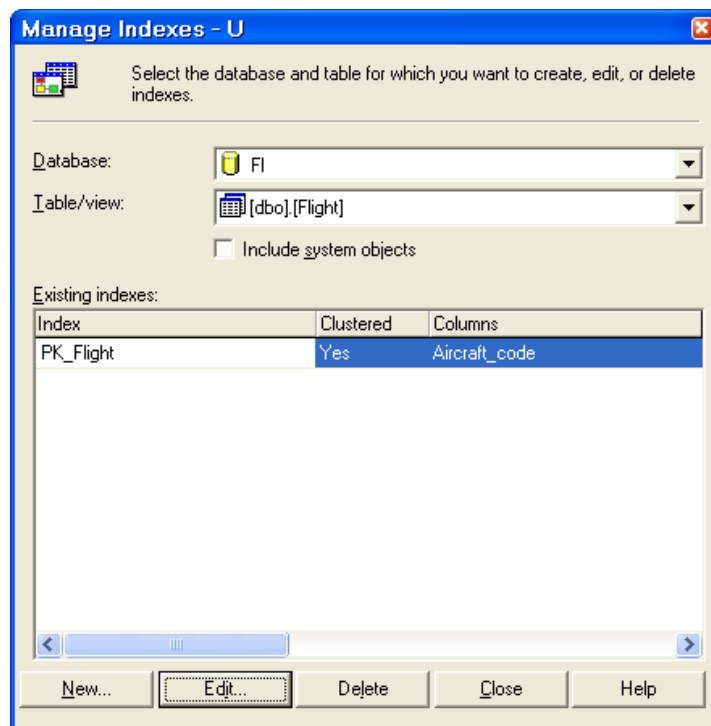
Chương 11. RÀNG BUỘC DỮ LIỆU VÀ CHỈ SỐ-Phần thực hành

11. Kịch Next
12. Kịch vào Make this a clustered index phía dưới Properties.
13. Nhập tên của index là PK_flight.
14. Kịch Finish.
15. Kịch OK.

11.2.2 Xem và sửa Indexes.

Bây giờ chúng ta sửa lại PK_flight index với Fill factor=60, thực hiện bằng EM:

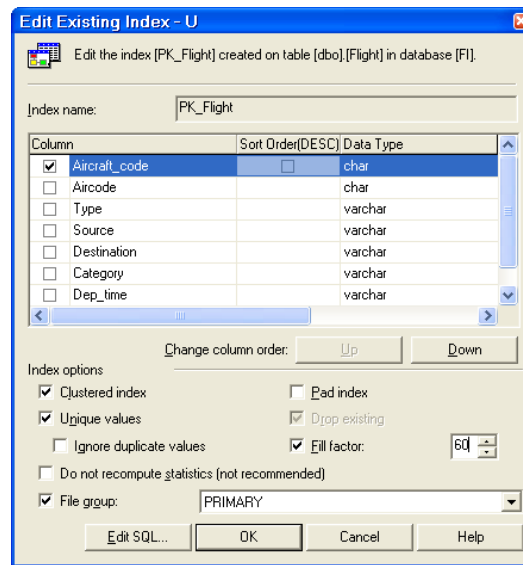
1. Kịch phải chuột vào bảng Flight.
2. Chọn Manage Indexes... từ All Task.



Hình 11.14

3. Chọn PK_Fligh và kịch Edit.
4. Trong mục Fill Factor, nhập vào giá trị 60.
5. Kịch OK.
6. Kịch Close

Chương 11. RÀNG BUỘC DỮ LIỆU VÀ CHỈ SỐ-Phần thực hành



Hình 11.15

11.2.3 Sử dụng Indexes

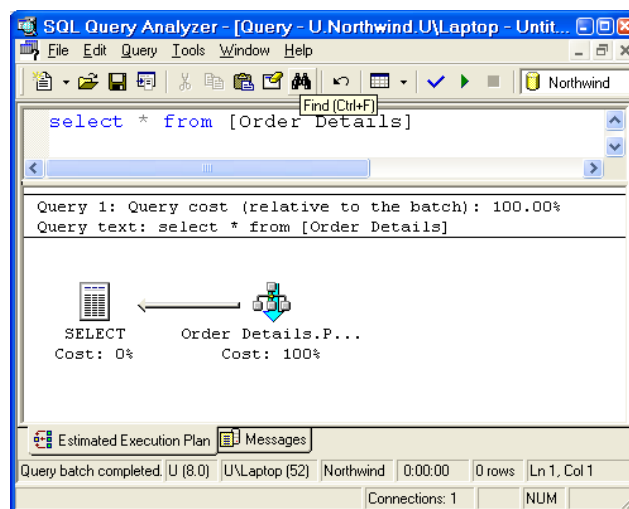
1. Mở QA.
2. Thực hiện câu lệnh:

```
use Northwind
Create nonclustered index ProductID_index
ON [Order Details](ProductID)
```

3. Bây giờ, thực hiện câu lệnh truy vấn sau:

```
select * from [Order Details]
```

3. Chọn Display Estimated Execution Plan từ Query menu.



Hình 11.17

Di chuột vào tên của Index trong hình trên, ta được những thông số sau đây:

Clustered Index Scan	
Scanning a clustered index, entirely or only a range.	
Physical operation:	Clustered Index Scan
Logical operation:	Clustered Index Scan
Estimated row count:	2,155
Estimated row size:	29
Estimated I/O cost:	0.0435
Estimated CPU cost:	0.00244
Estimated number of executes:	1.0
Estimated cost:	0.045953(100%)
Estimated subtree cost:	0.0459
Argument:	
OBJECT:([Northwind].[dbo].[Order Details].[PK_Order_Details])	

Khi thực hiện câu lệnh Select, nếu không bắt buộc nó phải chạy theo index nào thì bộ tối ưu hoá truy vấn (Optimizer) của SQL Server sẽ tự động tìm cách thực hiện.

Bây giờ, ta sẽ xem xét cách thực hiện câu lệnh truy vấn trong đó bắt buộc phải chạy theo index nào đó:

4. Thực hiện câu lệnh sau:

```
Select * from [Order Details] (INDEX=ProductID index)
```

Ta được các thông số như sau:

Index Scan	
Scanning a non-clustered index, entirely or only a range.	
Physical operation:	Index Scan
Logical operation:	Index Scan
Estimated row count:	2,155
Estimated row size:	35
Estimated I/O cost:	0.0398
Estimated CPU cost:	0.00244
Estimated number of executes:	1.0
Estimated cost:	0.042250(42%)
Estimated subtree cost:	0.0422
Argument:	
OBJECT:([Northwind].[dbo].[Order Details].[ProductID_index]), F ORCEDINDEX	

11.3 Bài tập

Thực hiện các công việc sau bằng QA:

1. Tạo ràng buộc mặc định (default constraint) cho cột Service_name trong bảng Service. Thiết đặt giá trị mặc định là 'First Aid'.
2. Tạo Rule và áp nó cho cột Service_code của bảng Airline_Service. Cho phép Service_code chỉ nhận 3 giá trị sau: 'CC', 'N' và 'WC'.
3. Tạo ràng buộc khoá chính (primary constraint) trên cột Aircode của bảng Airlines_Master. Để chắc chắn đã tạo được, hãy thử thêm một giá trị Null vào cột.
4. Thêm ràng buộc kiểm tra (check constraint) vào cột day_code của bảng Flight_days. Đảm bảo rằng dữ liệu được nhập vào nằm trong khoảng 1 đến 7.
5. Tạo khoá chính cho bảng Airline_master. Thiết đặt ràng buộc khoá ngoại trên cột Aircode của bảng Airline_meal.
6. Tạo Clustered index trên cột City_code của bảng City_master.
7. Tạo Non-clustered index trên 2 cột PNR_no và Ticket_no của bảng Passenger, trong đó thiết đặt Fill factor là 25%.
8. Xem tất cả các indexes của bảng Airlines_master.

12 Chương 12. KHUNG NHÌN & CON TRỎ (Views & Cursors)

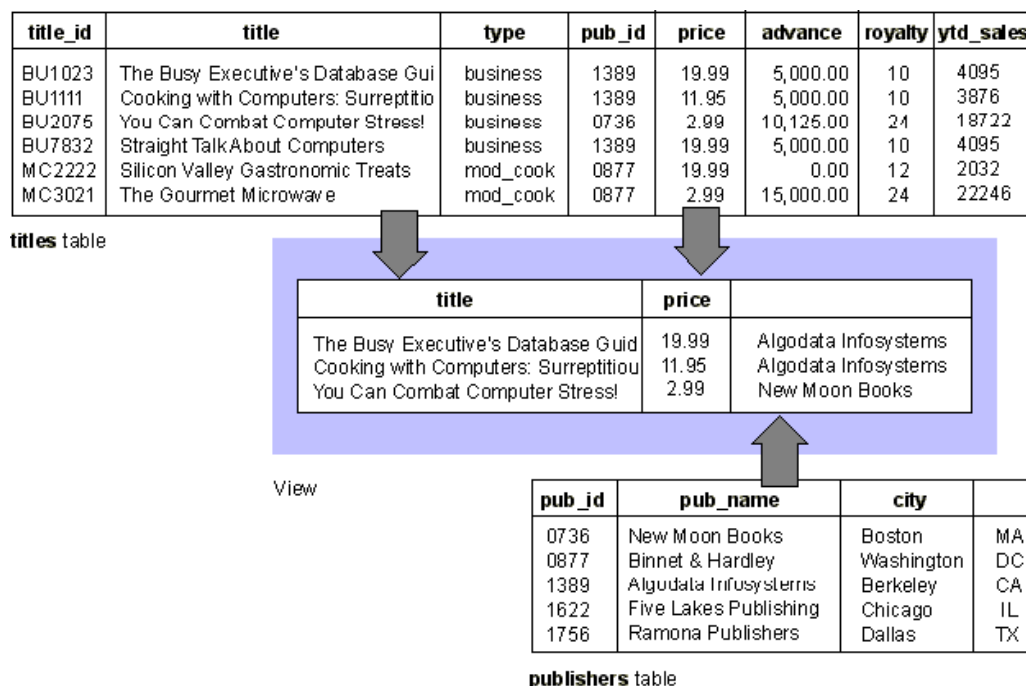
12.1 View

12.1.1 Giới thiệu

Truy vấn thường được sử dụng để xuất dữ liệu ra từ bảng. Nó được thực hiện trên dữ liệu (actual data) của bảng. Thay vì việc truy vấn và thực hiện trực tiếp trên dữ liệu thực (ví dụ như Queries trong Microsoft Access), SQL Server đã hỗ trợ một khái niệm mới, đó là View. View là một bảng tương tự như bảng chứa dữ liệu thực, nhưng nó chỉ là bảng logic (không phải là bảng vật lý), có nghĩa là nó không có vị trí lưu trữ vật lý của dữ liệu. Vì thế, View thường được gọi như một bảng ảo (Virtual table).

View là một cách khác để nhìn vào dữ liệu, và nó có thể nhìn thấy dữ liệu từ một hay nhiều bảng. Tuy nhiên, View không tồn tại như một tập dữ liệu được lưu trữ trên đĩa, mà nó chỉ là một tham chiếu tới những bảng dữ liệu vật lý.

View hoạt động như một bộ lọc dữ liệu từ cơ sở dữ liệu vì thực chất View là kết quả của câu lệnh truy vấn. Câu lệnh này có thể lấy dữ liệu từ nhiều bảng và nhiều cơ sở dữ liệu đồng thời.



Hình 12.1. Kết quả của khung nhìn tạo từ 2 bảng Titles và Publishers.

12.1.2 Tạo View

Cú pháp:

```
CREATE VIEW <Viewname> [WITH SCHEMABINDING]
AS <Select_Statement>
[WITH CHECK OPTION]
```

Trong đó:

WITH SCHEMABINDING: Đảm bảo rằng tất cả các đối tượng có trong câu lệnh tạo View không thể được xoá khi View đang tồn tại.

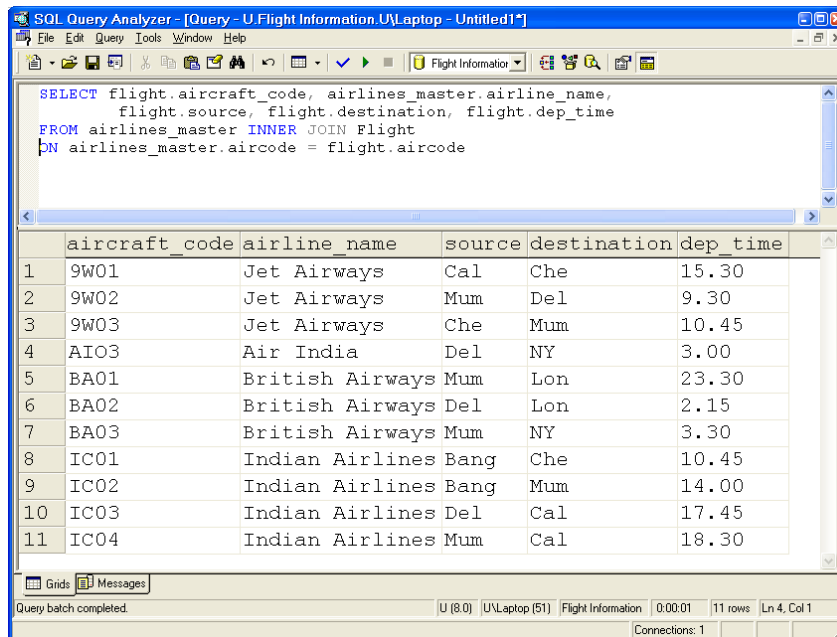
WITH CHECK OPTION: Đảm bảo rằng nếu bạn muốn sửa hoặc thêm dữ liệu thông qua View thì những dữ liệu đó phải thoả mãn tất cả các điều kiện trong câu lệnh Select

Ví dụ:

```
CREATE VIEW Try
AS
SELECT flight.aircraft_code, airlines_master.airline_name,
flight.source, flight.destination, flight.dep_time
FROM airlines_master INNER JOIN Flight
ON airlines_master.aircode = flight.aircode
```

Nếu ta thêm mệnh đề **WITH SCHEMABINDING** vào câu lệnh trên thì hệ thống sẽ không cho phép xoá 2 bảng **airlines_master** và **Flight** nếu như View có tên là **Try** còn tồn tại.

Kết quả của View trên sẽ là kết quả của câu lệnh **SELECT** chứa bên trong nó.



The screenshot shows the SQL Query Analyzer interface. The query window contains the following SQL code:

```
SELECT flight.aircraft_code, airlines_master.airline_name,
flight.source, flight.destination, flight.dep_time
FROM airlines_master INNER JOIN Flight
ON airlines_master.aircode = flight.aircode
```

The results window displays the following data:

	aircraft_code	airline_name	source	destination	dep_time
1	9W01	Jet Airways	Cal	Che	15.30
2	9W02	Jet Airways	Mum	Del	9.30
3	9W03	Jet Airways	Che	Mum	10.45
4	AIO3	Air India	Del	NY	3.00
5	BA01	British Airways	Mum	Lon	23.30
6	BA02	British Airways	Del	Lon	2.15
7	BA03	British Airways	Mum	NY	3.30
8	IC01	Indian Airlines	Bang	Che	10.45
9	IC02	Indian Airlines	Bang	Mum	14.00
10	IC03	Indian Airlines	Del	Cal	17.45
11	IC04	Indian Airlines	Mum	Cal	18.30

The status bar at the bottom indicates: Query batch completed. U (8.0) | U\Laptop (51) | Flight Information | 0.00.01 | 11 rows | Ln 4, Col 1 | Connections: 1

Hình 12.2

Như vậy, Views thường được sử dụng để:

- Lọc những bản ghi từ bảng theo yêu cầu.
- Bảo vệ dữ liệu từ những người dùng không có quyền.
- Giảm độ phức tạp của dữ liệu
- Tóm lược nhiều cơ sở dữ liệu vật lý vào một cơ sở dữ liệu logic.

12.1.3 Lợi ích của View đối với người sử dụng

Đối với người sử dụng cuối - End Users

- Dễ dàng để hiểu kết quả.
- Dễ hơn để thực hiện dữ liệu

Đối với người phát triển hệ thống- Developers

- Dễ dàng để truy cập dữ liệu
- Dễ dàng để bảo trì ứng dụng

12.1.4 Một số hướng dẫn khi tạo View

- Views chỉ có thể tạo trong cơ sở dữ liệu hiện tại.
- Tên View nên tương tự như tên bảng để dễ nhớ
- View có thể được xây dựng từ những View khác. SQL Server cho phép View lồng nhau 32 cấp. Nó có thể chứa 1024 cột từ một hoặc nhiều bảng hoặc Views.
- Mặc định, rules và triggers không được hỗ trợ bằng View.
- Views có thể được chỉ số. Tuy nhiên không phải tất cả các View đều có thể chỉ số được.
- Temporary tables không được tham gia trong views.
- Định nghĩa View tồn tại ngay cả khi bảng tham gia đã bị xoá
- Định nghĩa View không thể bao gồm các mệnh đề: ORDER BY, COMPUTE hoặc COMPUTE BY hoặc từ khoá INTO.

12.1.5 Sửa dữ liệu thông qua Views

- View có thể sửa chữa dữ liệu được xây dựng trên bảng:
- View chứa đựng ít nhất một bảng được định nghĩa sau mệnh đề FROM.
- Không chứa những hàm nhóm hoặc mệnh đề GROUP BY, UNION, DISTINCT, hoặc TOP

- View không chứa những cột được suy ra từ những cột khác

12.1.6 Indexed Views

Khái niệm Indexed Views được giới thiệu trong SQL Server 2000. Chúng ta có thể tạo Indexed Views trên bất kỳ phiên bản nào của SQL Server 2000. Trước khi Indexed Views được giới thiệu, người ta không hề có ý tưởng về việc tạo ra chỉ số trên View vì View không thực chất có lưu trữ vật lý của dữ liệu (nó chỉ là bảng ảo). Tuy nhiên, sau khi được thiết đặt chỉ số, View sẽ được xác định vị trí lưu trữ vật lý trên bảng (chúng ta sẽ bàn đến sau). Việc tạo indexes trên View sẽ giúp chúng ta cải thiện được tốc độ thực hiện của hệ thống (tương tự như đối với bảng).

Chúng ta có thể tạo cả Clustered và Non-Clustered Indexes trên Views. Tuy nhiên, những Indexes này phải là Unique clustered index.

Sau đây là một số quy tắc cho việc thiết đặt Clustered indexes:

- Chỉ nên tạo indexes trên những dữ liệu không thường xuyên được cập nhật. Nếu tạo indexes trên những dữ liệu này thì mỗi khi cập nhật dữ liệu, hệ thống sẽ phải cập nhật cả những thay đổi cho các file chứa indexes.
- Nó thực sự hữu ích cho những truy vấn bên trong có chứa hàm nhóm (aggregations) và kết nối nhiều bảng (Joins).

Một số giới hạn của Indexed View:

Việc thực hiện những kiểu truy vấn sau sẽ không cải thiện được hệ thống:

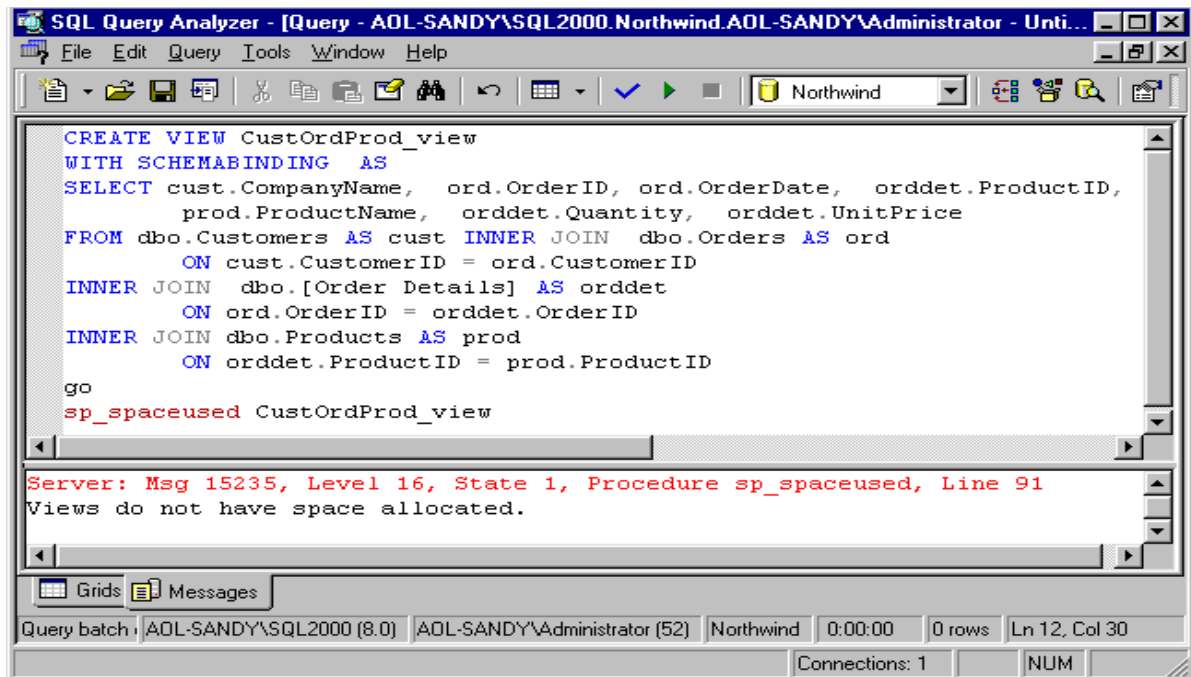
- Cơ sở dữ liệu phải thực hiện nhiều thao tác Updates.
- Hệ thống OLTP (Online Transaction Processing) chứa nhiều thao tác “write”.
- Những truy vấn không chứa những hàm nhóm và kết nối bảng.
- Hàm nhóm của dữ liệu trong câu lệnh truy vấn chia thành nhiều nhóm dữ liệu khác nhau.

Lưu ý:

- Lựa chọn SCHEMABINDING phải có trong định nghĩa View.
- View chỉ được tham chiếu đến các bảng, không được tham chiếu đến các View khác.
- Những đối tượng mà View tham chiếu tới phải trên cùng cơ sở dữ liệu với View.
- Lựa chọn ARITHABORT nên có trong khi tạo View.

Xem xét ví dụ sau, sử dụng cơ sở dữ liệu Northwind:

Chương 12. KHUNG NHÌN VÀ CON TRỎ (Views and Cursors)



Hình 12.3

Ví dụ trên đã tạo một View có tên là CustOrdPro_view và câu lệnh cuối cùng chỉ ra kết quả ở hình 12.3.

Ghi nhớ: **sp_spaceused** là một thủ tục hệ thống hiển thị kích thước lưu trữ vật lý trên đĩa của một đối tượng nào đó và các index của nó.

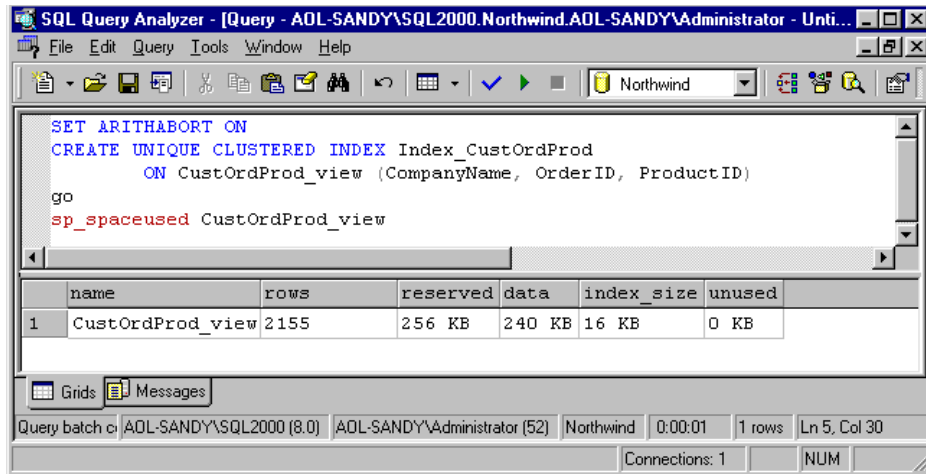
Bây giờ, chúng ta sẽ tạo Index đầu tiên trên CustOrdPro_view. Lưu ý, index đầu tiên được tạo trên View phải đáp ứng cả hai tính chất: Clustered và Unique.

```
SET ARITHABORT ON
CREATE UNIQUE CLUSTERED INDEX Index_CustOrdPro
ON CustOrdPro_view(CompanyName, OrderID, ProductID)
```

Ghi nhớ: Câu lệnh **SET ARITHABORT ON** sẽ giúp kết thúc ngay truy vấn nếu như nó gặp phải trường hợp tràn bộ nhớ (Overflow) hoặc lỗi chia cho 0. Bởi vì nó ảnh hưởng đến giá trị của biểu thức nên ta cần thiết đặt ON trong khi tạo indexed views hoặc trên những cột cần tính toán.

Sau khi tạo Clustered index, View trở nên thành đối tượng lưu trữ dữ liệu thực, tức là nó có vùng lưu trữ dữ liệu vật lý. Bây giờ ta thực hiện lại thủ tục **sp_spaceused** cùng với CustOrdProd_view, ta sẽ nhận được kết quả sau:

Chương 12. KHUNG NHÌN VÀ CON TRỎ (Views and Cursors)

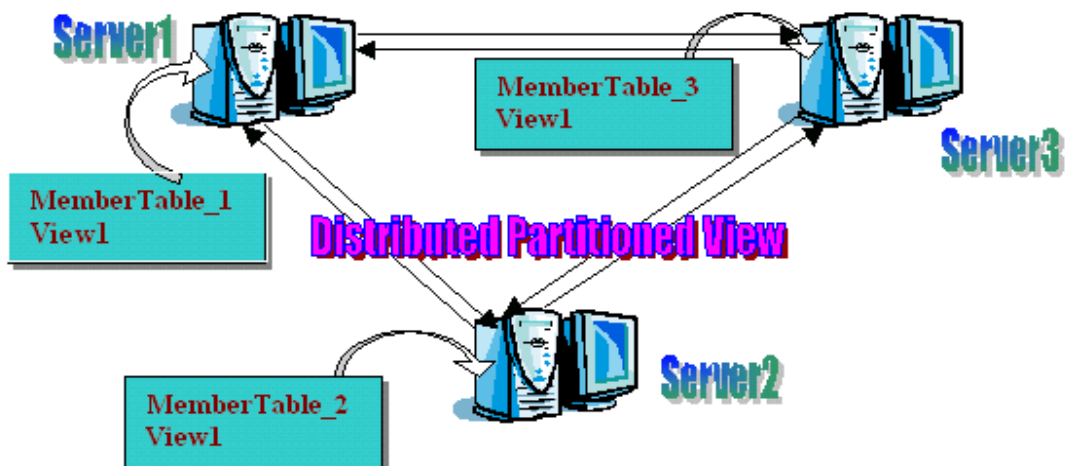


Hình 12.4

12.1.7 Distributed Partitioned Views- Khung nhìn phân tán

Trong Partitioned Views, dữ liệu được phân bố theo chiều ngang trên tập hợp những bảng được kết nối với nhau giống như là nó được lưu trữ trên một bảng duy nhất. SQL Server 2000 phân biệt giữa Local partitioned Views và Distributed partitioned views. Trong Local partitioned Views, tất cả các bảng mà View tham chiếu tới và bản thân View đó phải tồn tại trên cùng một instance của SQL SERVER. Còn trong Distributed partitioned views (DPVs), có ít nhất một bảng mà View tham chiếu tới sẽ nằm trên Server khác (Remote server).

DPVs cho phép chúng ta truy vấn dữ liệu trên các Server và các cơ sở dữ liệu khác nhau. Vì thế chúng ta có thể bố trí dữ liệu trên nhiều bảng khác nhau và trên nhiều Server khác nhau. Tuy nhiên, mỗi Server lại cần phải kết nối truy nhập tới tất cả các Server khác, vì thế chúng ta cần cấu hình tất cả các Server như là **Linked Server**. Khái niệm DPVs được chỉ ra như hình sau:



Hình 12.5. Distributed Partitioned Views

Chương 12. KHUNG NHÌN VÀ CON TRỎ (Views and Cursors)

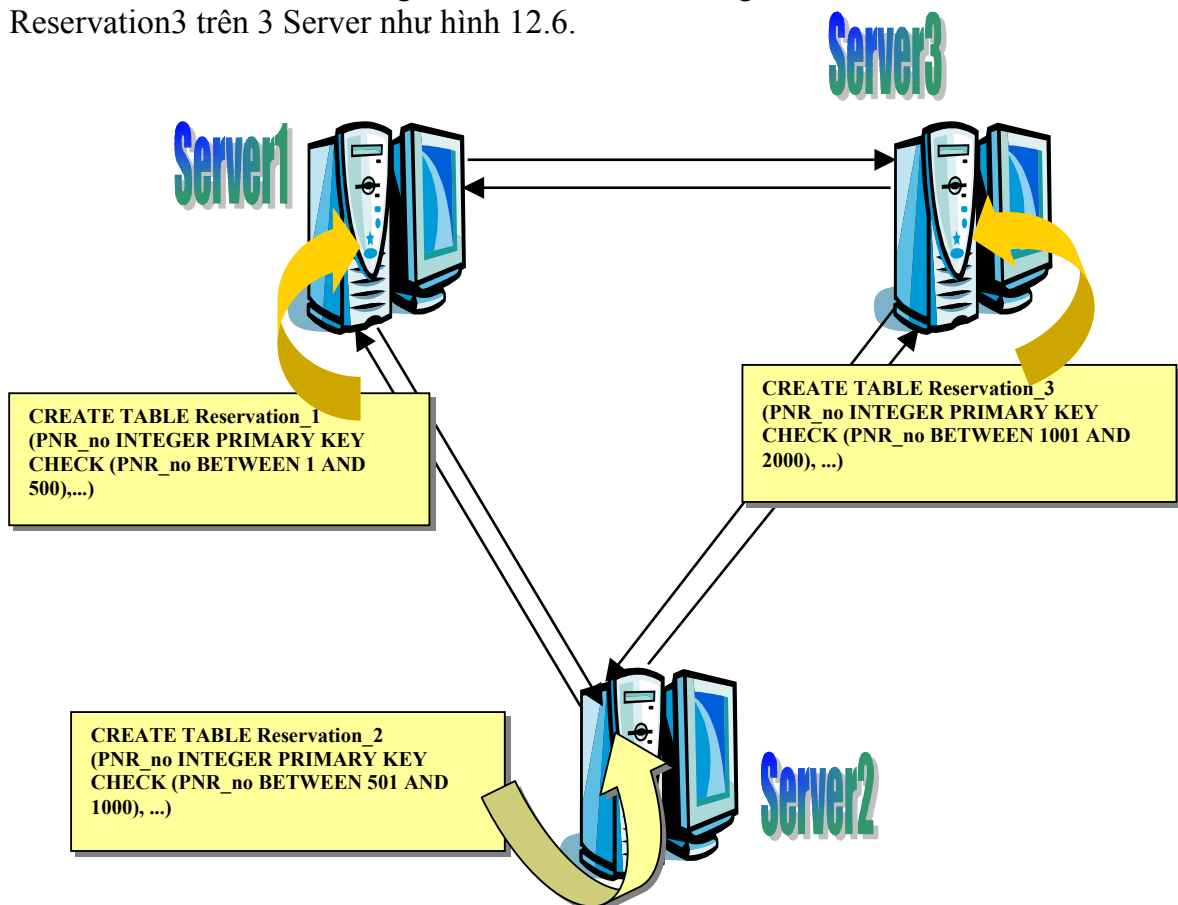
Giả sử rằng, bạn muốn phân bố bảng Customers và Orders trên 3 server, gọi là Server1, Server2, Server3. Để có được Linked Servers, mỗi Server phải liên kết được với 2 server còn lại. Trong SQL Server 2000, bạn có thể thực hiện thêm và cập nhật lại dữ liệu trên các DPVs.

Trước khi sử dụng DPVs, bạn nên làm những việc sau:

- Lên kế hoạch phân bố dữ liệu trên các Server khác nhau.
- Tạo Partitioned View cùng với khoá để chia dữ liệu. Ví dụ: bạn nên chia dữ liệu theo giá trị của một cột, ví dụ EmpID, lưu trữ các bản ghi có EmpID từ 1 ..1000 trên Server A, từ 1001..3000 trên Server B...
- Sử dụng Linked Server để liên kết giữa các Server.
- Tạo DPV trên mỗi Server để mỗi Server có thể biết được những đối tượng của nó.

Xem xét quá trình tạo ví dụ sau:

Chia dữ liệu của bảng Reservation vào 3 bảng Reservation1, Reservation2, Reservation3 trên 3 Server như hình 12.6.



Hình 12.6

Chương 12. KHUNG NHÌN VÀ CON TRỎ (Views and Cursors)

Sau đó, bạn cần thực hiện những công việc sau để xây dựng DPVs:

Thêm các định nghĩa liên kết (linked server definitions) cho mỗi nhóm server tham gia vào thực hiện View. Điều này sẽ giúp cho các DPVs có thể truy cập được các dữ liệu trên các server khác nhau.

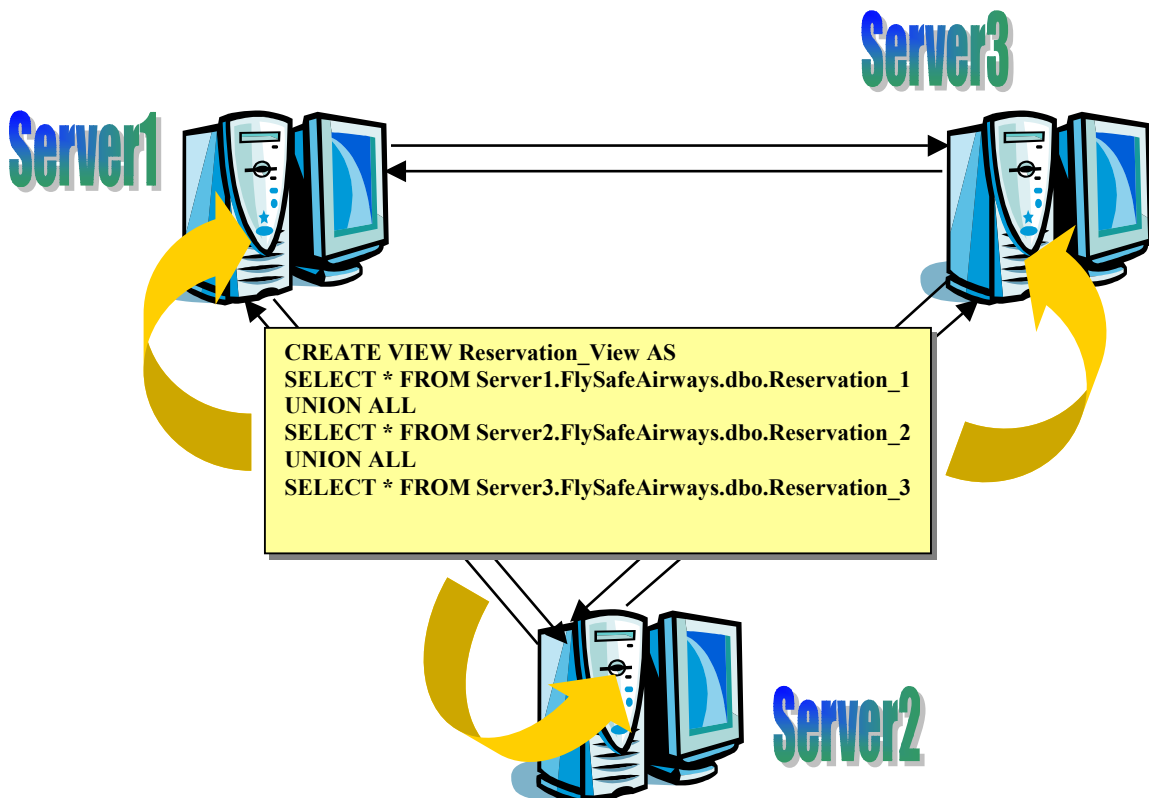
Sử dụng thủ tục hệ thống `sp_serveroption` để thiết đặt lựa chọn **lazy schema validation** cho mỗi server được tham chiếu trong View.

Tạo DPV trên từng server.

Bạn có thể tạo một DPV trên Server1 như sau:

```
CREATE VIEW Reservation_View AS
SELECT * FROM Server1.FlySafeAirways.dbo.Reservation_1
UNION ALL
SELECT * FROM Server2.FlySafeAirways.dbo.Reservation_2
UNION ALL
SELECT * FROM Server3.FlySafeAirways.dbo.Reservation_3
```

Sau đó, thực hiện việc tạo View này trên tất cả các server.



Hình 12.7. Tạo View lấy dữ liệu từ 3 Server

12.1.8 Sử dụng View để cập nhật dữ liệu

Các chức năng có thể thực hiện trên View tương tự như đối với bảng. Chúng ta có thể thực hiện các câu lệnh INSERT, UPDATE, và DELETE trên View.

Khi chúng ta thay đổi dữ liệu thông qua View, đồng nghĩa với việc chúng ta thay đổi dữ liệu trên các bảng mà View đó đang tham chiếu. Tuy nhiên, nên thực hiện một số các quy luật sau khi thực hiện sửa chữa dữ liệu thông qua View.

Câu lệnh SELECT trong định nghĩa View không nên chứa:

Các hàm nhóm dữ liệu (Aggregate functions)

Các mệnh đề TOP, GROUP BY, UNION, hoặc DISTINCT.

Cột có giá trị được suy ra từ các cột khác (derived columns)

Sau mệnh đề FROM trong câu lệnh SELECT nên có ít nhất một bảng. Ví dụ, View sau đây không thể cập nhật dữ liệu:

```
CREATE VIEW NoTable AS
SELECT Getdate() AS CurrentDate
@@LANGUAGE AS CurrentLanguage
```

Chúng ta chỉ có thể cập nhật và thêm dữ liệu vào 1 bảng đứng sau mệnh đề FROM của View. Nếu muốn cập nhật dữ liệu trên nhiều bảng, chúng ta phải sử dụng INSTEAD OF trigger. Chúng ta sẽ bàn đến trigger chi tiết trong chương sau.

Nếu như bảng được tham chiếu trong View chứa cột có ràng buộc NOT NULL không phải là một phần của View thì chúng ta phải gán giá trị mặc định cho cột này để có thể thêm dữ liệu cho bản ghi.

Nếu định nghĩa View có chứa lựa chọn WITH CHECK, tất cả các cột được sửa chữa phải thỏa mãn điều kiện trong câu lệnh SELECT. Ví dụ, nếu câu lệnh SELECT có chứa mệnh đề WHERE emp_id <= 500, thì chúng ta không thể sửa lại dữ liệu trong cột emp_id có giá trị lớn hơn 500.

Chúng ta có thể xóa dữ liệu nếu như View chỉ tham chiếu đến 1 bảng. Để xóa dữ liệu trên View có tham chiếu sang nhiều bảng, chúng ta phải sử dụng INSTEAD OF trigger.

Chúng ta cũng có thể sử dụng DVPs để cập nhật dữ liệu cho các bảng tham chiếu.

12.1.9 Sửa cấu trúc Views

Chúng ta có thể sử dụng câu lệnh ALTER VIEW để thực hiện sửa cấu trúc của View. Cú pháp của nó tương tự như cú pháp của lệnh CREATE VIEW, chỉ cần thay thế từ khóa CREATE bằng từ khóa ALTER.

Cú pháp:

```
ALTER VIEW <Viewname> [WITH SCHEMABINDING]
AS <Select_Statement>
[WITH CHECK OPTION]
```

Ví dụ:

```
ALTER VIEW Try AS
SELECT flight.aircraft_code,
airlines_master.airline_name,
flight.source, flight.destination, flight.dep_time,
flight.journey_hrs
FROM airlines_master INNER JOIN Flight
ON airlines_master.aircode = flight.aircode
```

12.1.10 Xoá Views

Khi một View nào đó không còn cần thiết nữa, chúng ta có thể xoá nó.

Cú pháp:

```
DROP VIEW <Viewname>
```

Ví dụ:

```
DROP VIEW Try
```

12.2 Con trỏ_Cursors

12.2.1 Giới thiệu

Con trỏ là một đối tượng được sử dụng trong ứng dụng để thực hiện truy cập dữ liệu trên từng dòng.

Sử dụng con trỏ, có thể:

Sử dụng con trỏ có thể đến vị trí một dòng nhất định trong tập kết quả.

Truy cập đến 1 dòng hoặc 1 tập hợp những dòng từ vị trí hiện tại của con trỏ trong tập kết quả.

Hỗ trợ sửa chữa dữ liệu ở 1 dòng nào đó.

Hỗ trợ nhiều cấp độ khác nhau cho phép biết rõ ràng những thay đổi mà những người sử dụng khác nhau đã làm với dữ liệu.

12.2.2 Tạo con trỏ

Câu lệnh DECLARE để khai báo con trỏ. Nó chứa đựng câu lệnh Select để đưa ra tập những bản ghi từ bảng.

Cú pháp:

```
DECLARE <Cursor_Name> CURSOR
  [LOCAL | GLOBAL]
  [FORWARD ONLY | SCROLL]
  [STATIC | KEYSET | DYNAMIC | FAST_FORWARD]
  [READ_ONLY | SCROLL_LOCKS | OPTIMISTIC]
  [TYPE_WARNING]
  FOR <Select Statements>
  [FOR UPDATE [OF Column_name[,...N]]]
```

12.2.3 Các bước trong sử dụng Cursor

Mở Cursor:

```
OPEN <Cursor_name>
```

Truy cập đến các bản ghi:

```
FETCH <Cursor_name>
```

Đóng Cursor:

```
CLOSE <Cursor_name>
```

Xoá tham chiếu của Cursor:

```
DEALLOCATE <Cursor_name>
```

12.2.4 Truy cập dữ liệu bằng cursor

FETCH FIRST: Truy cập đến dòng đầu tiên.

FETCH NEXT: Truy cập đến dòng tiếp theo.

FETCH PRIOR: Truy cập đến dòng trước dòng hiện tại của con trỏ.

FETCH LAST: Truy cập đến dòng cuối cùng.

FETCH ABSOLUTE n: Nếu n là số nguyên dương, nó truy cập đến dòng thứ n. Nếu n là số nguyên âm, nó truy cập đến dòng thứ n trước dòng cuối cùng của con trỏ. Nếu n=0 nó truy cập tới chính dòng hiện tại.

FETCH RELATIVE n: Nếu n là số dương, truy cập đến dòng thứ n sau dòng hiện tại của con trỏ. Nếu n là số âm, truy cập đến dòng thứ n, truy cập đến dòng thứ n trước vị trí hiện tại của con trỏ. Nếu bằng 0, truy cập đến dòng hiện tại một lần nữa.

@@FETCH _STATUS: Quay trở lại trạng thái thực hiện của con trỏ cuối cùng.

@@CURSOR_ROWS: Quay trở lại số dòng của con trỏ hiện tại đang mở.

12.2.5 Ví dụ

Sau đây là một ví dụ về khai báo và thực hiện của con trỏ:

```
DECLARE Pub_Cursor CURSOR SCROLL
        FOR SELECT * FROM publishers ORDER BY pub_name
OPEN Pub_Cursor
FETCH FIRST FROM Pub_Cursor
WHILE @@FETCH_STATUS = 0
BEGIN
    FETCH NEXT FROM Pub_Cursor
END
```

	pub_id	pub_name	city	state	country
1	1389	Algodata Infosystems	Berkeley	CA	USA
1	0877	Binnet & Hardley	Washington	DC	USA
1	1622	Five Lakes Publishing	Chicago	IL	USA
1	9901	GGG&G	München	NULL	Germany

Grids Messages

Hình 12.8. Khai báo và thực hiện con trỏ

12.3 Câu hỏi trắc nghiệm

1. View trong SQL Server có thể chứa tối đa bao nhiêu cột

- A 256
- B 32
- C 1024
- D Không giới hạn

2. Nêu sự khác nhau giữa 2 câu lệnh sau:

- a.

```
SELECT A,B INTO Practice
FROM Table1 INNER JOIN Table2
ON Table1.B=Table2.B
```
- b.

```
CREATE VIEW Practise AS
SELECT A,B FROM Table1 INNER JOIN Table2
ON Table1.B=Table2.B
```

3. View bị xoá khi các bảng liên quan đến nó bị xoá.

- A Đúng
- B Sai

4. Bạn tạo ra View có tên là vw_Hanoi_St xác định trên bảng Student để lưu những sinh viên có nơi sinh là Hà nội. Trong câu lệnh CREATE VIEW có bao gồm lựa chọn WITH CHECK OPTION. Nếu bạn cố gắng thêm một sinh viên có nơi sinh là TP.HCM vào bảng Student thông qua View thì điều gì sẽ xảy ra?

- A SQL Server vẫn cho phép thực hiện bình thường, bản ghi mới được thêm vào bảng.
- B SQL Server không cho phép thực hiện vì bạn không có quyền sửa View.
- C SQL Server không cho phép thực hiện vì bản ghi thêm vào có nơi sinh không phải là Hà nội.

5. Giả sử bạn sẽ sử dụng câu lệnh SELECT sau để tạo View:

```
SELECT animal_catcd, animal_category, AVG(animal_age), COUNT(*)
FROM Animal
GROUP BY animal_catcd, animal_category
```

Những cột nào sau đây bắt buộc phải gán tên cho cột?

- A animal_catcd
- B animal_category
- C AVG(animal_age),
- D COUNT(*)

6. Trong những trường hợp nào sau đây nên sử dụng indexed view?

Chương 13. KHUNG NHÌN VÀ CON TRỎ (Views and Cursors)

- A OLTP system
- B Có số lượng lớn các phép toán cập nhật trong thực hiện cơ sở dữ liệu.
- C Truy vấn có chứa hàm nhóm dữ liệu (Aggregation)
- D Truy vấn có chứa nhiều liên kết dữ liệu (Joins)

7. Sau khi tạo Clustered index trên View, View trở nên thành đối tượng lưu trữ dữ liệu thực?

- A Đúng
- B Sai

8. Trong Distributed Partitioned Views, tất cả các server phải được cấu hình như là ...?

9. Lệnh nào sau đây cho phép đóng con trỏ lại khi không cần thiết nữa?

- A CLOSE
- B EXIT
- C DELLOCATE
- D BREAK

10. Con trỏ sau khi được đóng (Closed) vẫn tồn tại trong bộ nhớ?

- A Đúng
- B Sai

11. Loại con trỏ nào sau đây cho phép tất cả các truy vấn (FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE) được thực hiện?

- A LOCAL
- B SCROLL
- C FORWARD_ONLY
- D GLOBAL
- E SCROLL_LOCKS

13 Chương 13. KHUNG NHÌN VÀ CON TRỎ Phần thực hành

Mục đích:

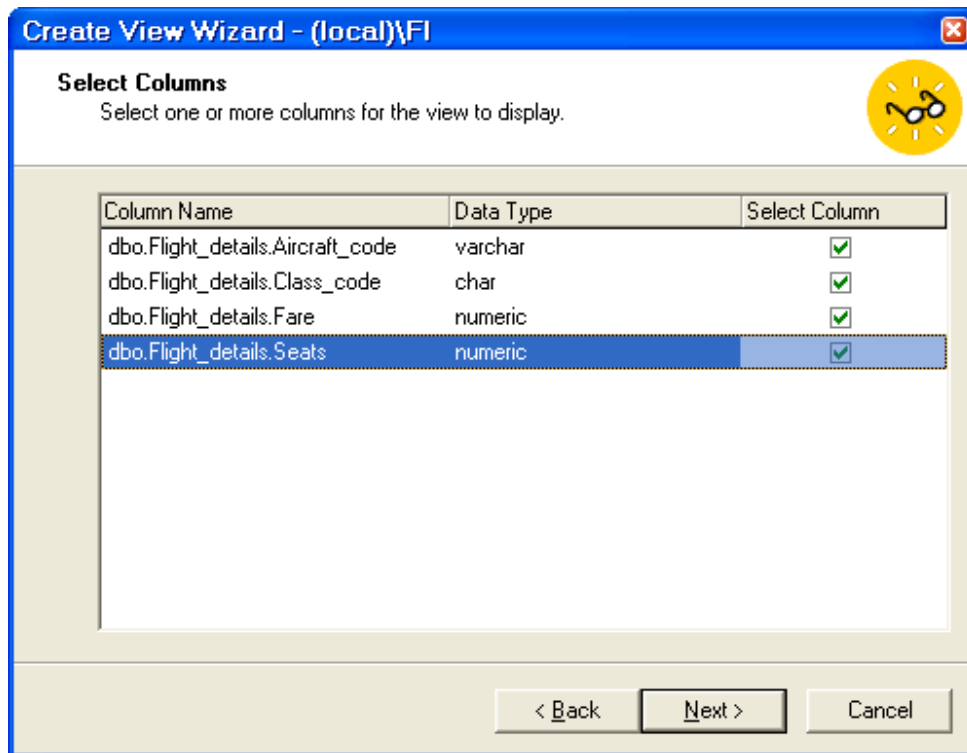
- Tạo View sử dụng EM và T-SQL.
- Sửa View sử dụng EM
- Đổi tên View
- Tạo các loại khác nhau của con trỏ
- Thực hiện các phép toán trên con trỏ
- Sửa dữ liệu thông qua con trỏ
- Đóng và xoá con trỏ khỏi bộ nhớ.

13.1 Tạo View

13.1.1 Sử dụng Create View Wizard

Các bước thực hiện:

1. Chọn cơ sở dữ liệu FI.
2. Kích Tool/Wizard...
3. Kích đúp vào lựa chọn Database
4. Kích chọn Create View Wizard, kích OK.
5. Kích Next.
6. Chọn Database name là FI.
7. Kích Next.
8. Kích vào các bảng mà View sẽ tham chiếu tới trong mục Include in View.
Giả sử bảng Flight_details.
9. Kích Next.
10. Kích vào những cột muốn đưa ra, trong mục Select Column.(Hình 13.1)
11. Kích Next.
12. Soạn 'where aircraft_code like 'IC%''.
13. Kích Next.
14. Kích Finish.
15. Kích OK.



Hình 13.1

13.1.2 Tạo View bằng T-SQL

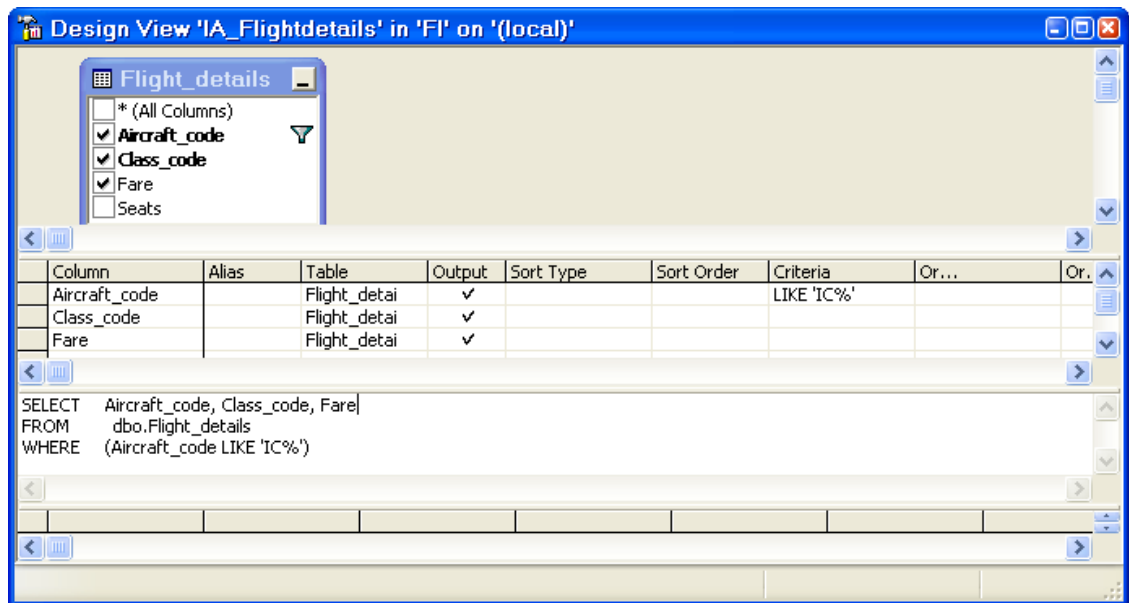
Thay vì tạo View bằng cách trên, ta có thể thực hiện câu lệnh sau trong QA:

```
CREATE VIEW IA_Flightdetails  
AS SELECT Aircraft_code, Class_code, Fare  
FROM Flight_details  
WHERE aircraft_code like 'IC%'
```

13.2 Sửa View

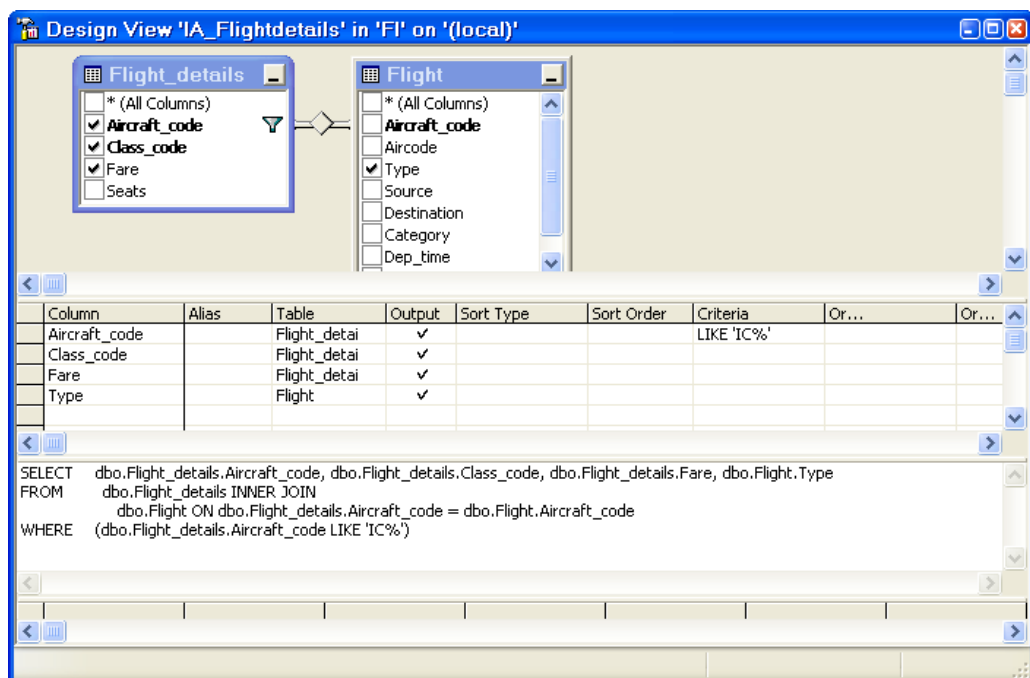
Thực hiện các bước sau:

1. Chọn cơ sở dữ liệu FI.
2. Kích vào đối tượng View.
3. Kích phải chuột vào IA_Flightdetails.
4. Chọn Design View trên thành menu. (Hình 13.2)
5. Kích phải chuột vào vùng trống, kích vào Add Table.
6. Chọn bảng Flight trong danh sách.
7. Kích Add. Kích Close.
8. Kích chọn Type trong bảng Flight.



Hình 13.2

9. Kích Save.



Hình 13.3

13.3 Con trỏ

13.3.1 Khai báo con trỏ (Cursor)

Trong QA thực hiện câu lệnh sau để tạo Cursor:

Chương 13. KHUNG NHÌN VÀ CON TRỎ- Phần thực hành

```
Use Pubs
DECLARE TitleCursor CURSOR
SCROLL
FOR
Select Title_id, Title, Price, ytd_sales
FROM Titles
Where type='psychology'
```

13.3.2 Mở con trỏ

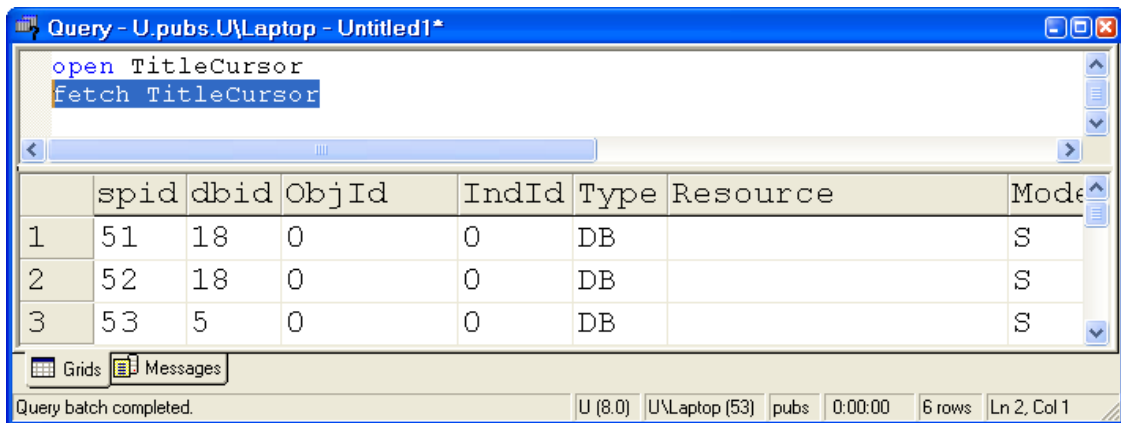
```
OPEN TitleCursor
```

13.3.3 Truy vấn dữ liệu

Sử dụng câu lệnh Fetch để truy vấn dòng dữ liệu trong tập kết quả của con trỏ.

```
FETCH TitleCursor
```

Kết quả:



	spid	dbid	ObjId	IndId	Type	Resource	Mode
1	51	18	0	0	DB		S
2	52	18	0	0	DB		S
3	53	5	0	0	DB		S

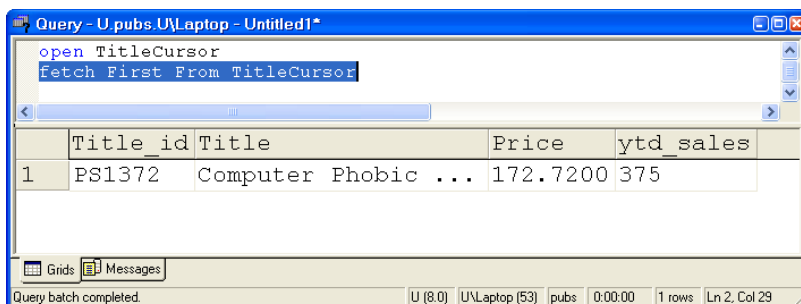
Query batch completed. U (8.0) U\Laptop (53) pubs 0:00:00 6 rows Ln 2, Col 1

Hình 13.4

13.3.4 Truy vấn dòng đầu tiên

```
FETCH FIRST FROM TitleCursor
```

Kết quả:



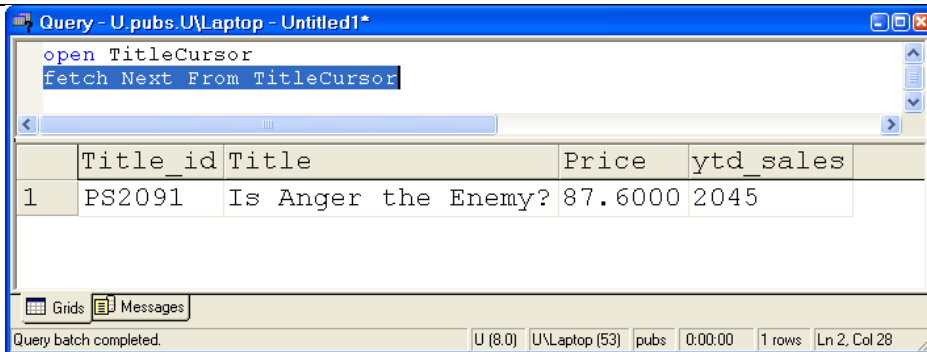
	Title_id	Title	Price	ytd_sales
1	PS1372	Computer Phobic ...	172.7200	375

Query batch completed. U (8.0) U\Laptop (53) pubs 0:00:00 1 rows Ln 2, Col 29

Hình 13.5

13.3.5 Truy vấn dòng tiếp theo

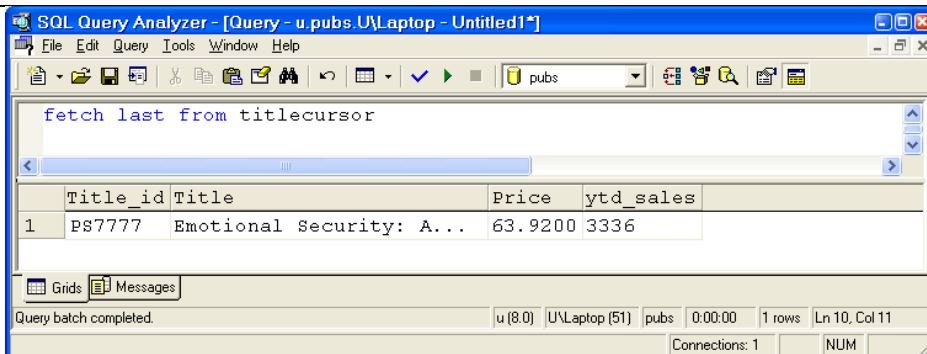
```
FETCH NEXT FROM TitleCursor
```



Hình 13.6

13.3.6 Truy vấn dòng cuối cùng

```
FETCH LAST FROM TitleCursor
```

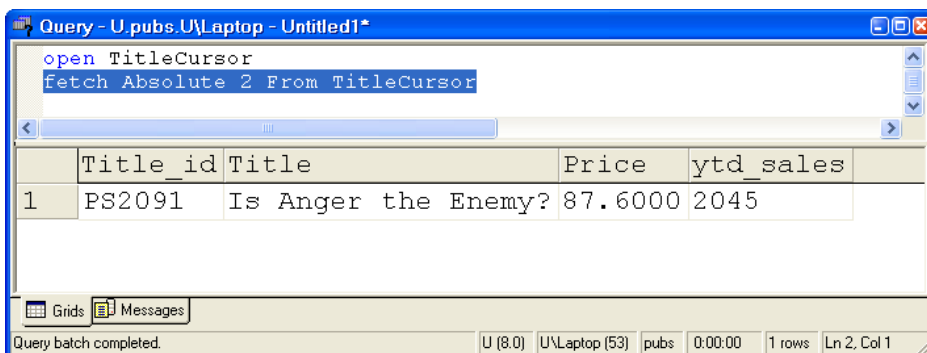


Hình 13.7

13.3.7 Truy vấn đến một dòng có vị trí xác định

```
FETCH ABSOLUTE 2 FROM TitleCursor
```

Kết quả:



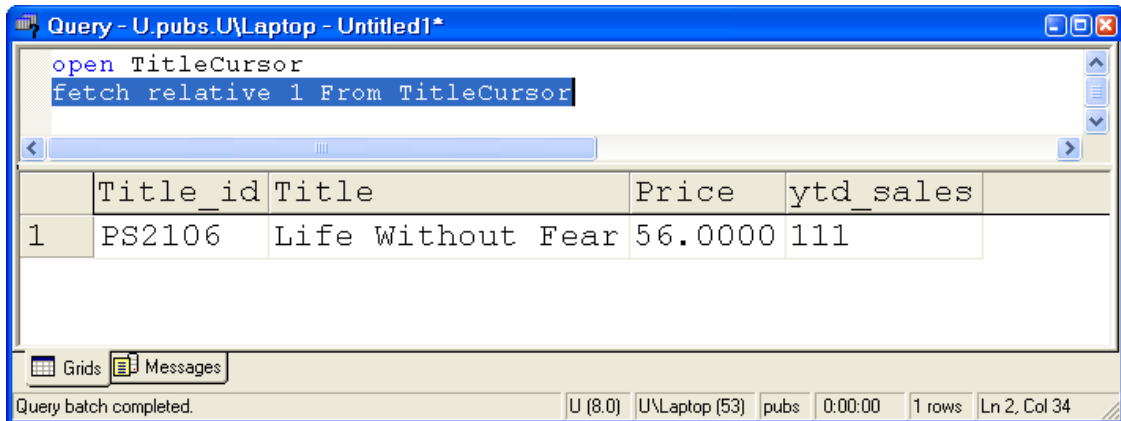
Hình 13.8

13.3.8 Truy vấn đến dòng liên quan

Truy vấn đến dòng liên quan đến vị trí hiện tại của con trỏ.

```
FETCH RELATIVE 1 FROM TitleCursor
```

Kết quả:



Title_id	Title	Price	ytd_sales	
1	PS2106	Life Without Fear	56.0000	111

Hình 13.9

13.3.9 Đóng và xoá vùng nhớ (Deallocating) của con trỏ

```
CLOSE TitleCursor  
DEALLOCATE TitleCursor
```

13.4

13.5 Bài tập

Thực hiện các yêu cầu sau bằng QA:

1. Tạo View có tên `pass_view` chứa PNR number, Aircraft code, ticket number, passenger name, và reservation status của PNR number có giá trị nhỏ hơn 4. Đảm bảo rằng view có thể tránh được thêm dữ liệu có PNR number lớn hơn 3.
2. Hiển thị dữ liệu của `pass_view`.
3. Sử dụng `pass_view`, sửa tên khách hàng = 'Pam Houston' của ticket number=3. Xem lại bảng `Passenger` để kiểm tra lại.
4. Tạo View có tên là `Weekend_flights` chứa aircraft code, và day code của tất cả các chuyến bay có day code bằng 1 hoặc 7.
5. Aircode có code AI03 không bay trong ngày có day code bằng 7. Sử dụng `Weekend_flights` để xóa tất cả các thông tin liên quan. Xem lại bảng `Flight_days` để kiểm tra việc xóa đã thực hiện hay chưa.
6. Tạo 3 bảng `CC_table`, `WC_table`, và `N_table` lấy thông tin từ bảng `Airline_service`. `CC_table`, `WC_table`, và `N_table` lưu những bản ghi có service code tương ứng là 'CC', 'WC' và 'N'.
7. Tạo local partitioned view có tên là `all_services` chứa tất cả các dòng trong 3 bảng trên.
8. Thêm dữ liệu vào 2 cột aircode và service code của bảng `CC_table` thông qua View `all_services`. Xem lại bảng `CC_table` để kiểm tra xem bản ghi mới đã được thêm vào hay chưa.
9. Hiển thị tất cả các bản ghi trong bảng `Reservation` sử dụng con trỏ và sau đó xóa nó khỏi bộ nhớ.

14 Chương 14. THỦ TỤC- STORED PROCEDURES(SPS)

SPs là công cụ cần thiết cho bất kỳ hệ quản trị cơ sở dữ liệu nào. Người phát triển hoặc người quản trị viết SPs để thực hiện những công việc quản trị hoặc các quy tắc dữ liệu phức tạp. SPs có thể chứa những câu lệnh thực hiện dữ liệu (DML) hoặc những câu lệnh truy vấn dữ liệu(SELECT).

14.1 Định nghĩa

SPs là tập hợp của các câu lệnh T-SQL được biên dịch trước (**pre_compiled**). SPs được đặt tên và được xử lý như một khối lệnh thống nhất (chứ không phải thực hiện rời rạc các câu lệnh).

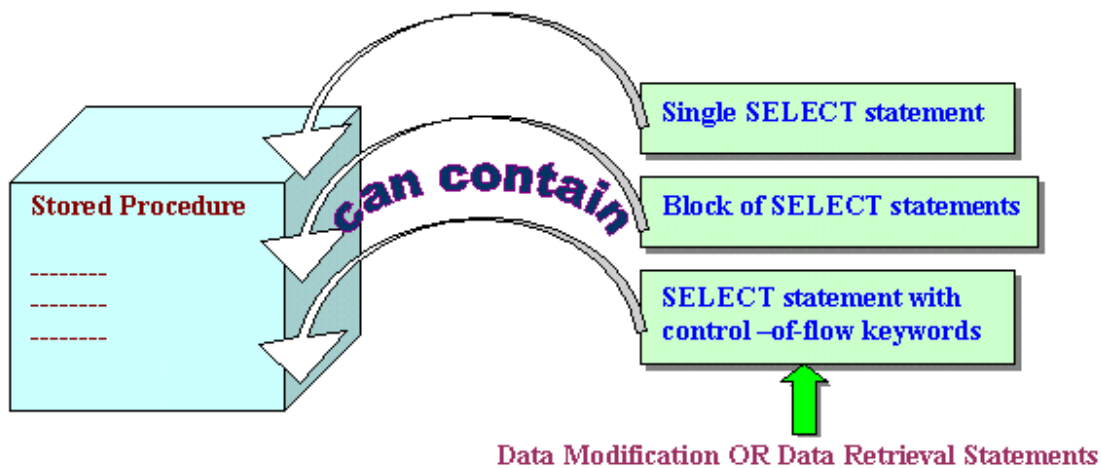
SQL Server cung cấp một số các thủ tục được lưu trữ sẵn trong hệ thống giúp thực hiện một số công việc thường xuyên. Nó được gọi là thủ tục hệ thống –System stored procedures. Còn những thủ tục do người sử dụng tự viết gọi là User stored procedures.

SPs trong SQL Server cũng tương tự như khái niệm về thủ tục trong các ngôn ngữ lập trình khác, bởi vì:

Chấp nhận biến đầu vào và trả lại kết quả khi thực hiện.

Chứa những câu lệnh dùng trong lập trình có thể thao tác với cơ sở dữ liệu và có thể gọi đến các thủ tục khác.

Trả lại giá trị trạng thái khi thủ tục được gọi để xác định việc thực hiện thủ tục thành công hay thất bại.



Hình 14.1. Các thành phần của SPS

14.2 Lợi ích khi quản lý dữ liệu bằng SPs

Tăng tốc độ thực hiện: Một trong những lợi ích lớn nhất khi sử dụng SPs là tốc độ. SPs được tối ưu hoá trong ngay ở lần biên dịch đầu tiên, điều này cho phép chúng có thể thực hiện nhanh hơn nhiều lần so với các câu lệnh T-SQL thông thường.

Tốc độ truy cập dữ liệu nhanh hơn: Khi thực thi một câu lệnh SQL thì SQL Server phải kiểm tra permission xem user gửi câu lệnh đó có được phép thực hiện câu lệnh hay không đồng thời kiểm tra cú pháp rồi mới tạo ra một execute plan và thực thi. Nếu có nhiều câu lệnh như vậy gửi qua network có thể làm giảm đi tốc độ làm việc của server. SQL Server sẽ làm việc hiệu quả hơn nếu dùng stored procedure vì người gửi chỉ gửi một câu lệnh đơn và SQL Server chỉ kiểm tra một lần sau đó tạo ra một execute plan và thực thi. Nếu stored procedure được gọi nhiều lần thì execute plan có thể được sử dụng lại nên sẽ làm việc nhanh hơn. Ngoài ra cú pháp của các câu lệnh SQL đã được SQL Sever kiểm tra trước khi lưu nên nó không cần kiểm lại khi thực thi.

Chương trình được modul hoá: Một khi stored procedure được tạo ra nó có thể được sử dụng lại. Điều này sẽ làm cho việc bảo trì (maintainability) dễ dàng hơn do việc tách rời giữa business rules (tức là những logic thể hiện bên trong stored procedure) và cơ sở dữ liệu. Ví dụ nếu có một sự thay đổi nào đó về mặt logic thì ta chỉ việc thay đổi code bên trong stored procedure mà thôi. Những ứng dụng dùng stored procedure này có thể sẽ không cần phải thay đổi mà vẫn tương thích với business rule mới.

Nhất quán: Lợi ích nữa của SPs là thiết đặt được ràng buộc dữ liệu để đảm bảo tính nhất quán. Người sử dụng không thể thực hiện tùy tiện dữ liệu để làm mất tính đúng đắn của dữ liệu.

Nâng cao khả năng bảo mật dữ liệu: Giả sử chúng ta muốn giới hạn việc truy xuất dữ liệu trực tiếp của một user nào đó vào một số bảng, ta có thể viết một stored procedure để truy xuất dữ liệu và chỉ cho phép user đó được sử dụng stored procedure đã viết sẵn mà thôi chứ không thể thao tác trực tiếp trên các bảng đó. Ví dụ, ta có thể tạo ra SPs để ta làm chủ và chỉ cung cấp quyền EXECUTE cho những SPs này, vì thế những người sử dụng khác không được phép trực tiếp làm việc với dữ liệu.

Ngoài ra stored procedure có thể được encrypt (mã hóa) để tăng cường tính bảo mật.

14.3 Các kiểu SPs

SPs chia làm 2 loại:

System stored procedures: Thủ tục mà những người sử dụng chỉ có quyền thực hiện, không được phép thay đổi.

User stored procedures: Thủ tục do người sử dụng tạo và thực hiện.

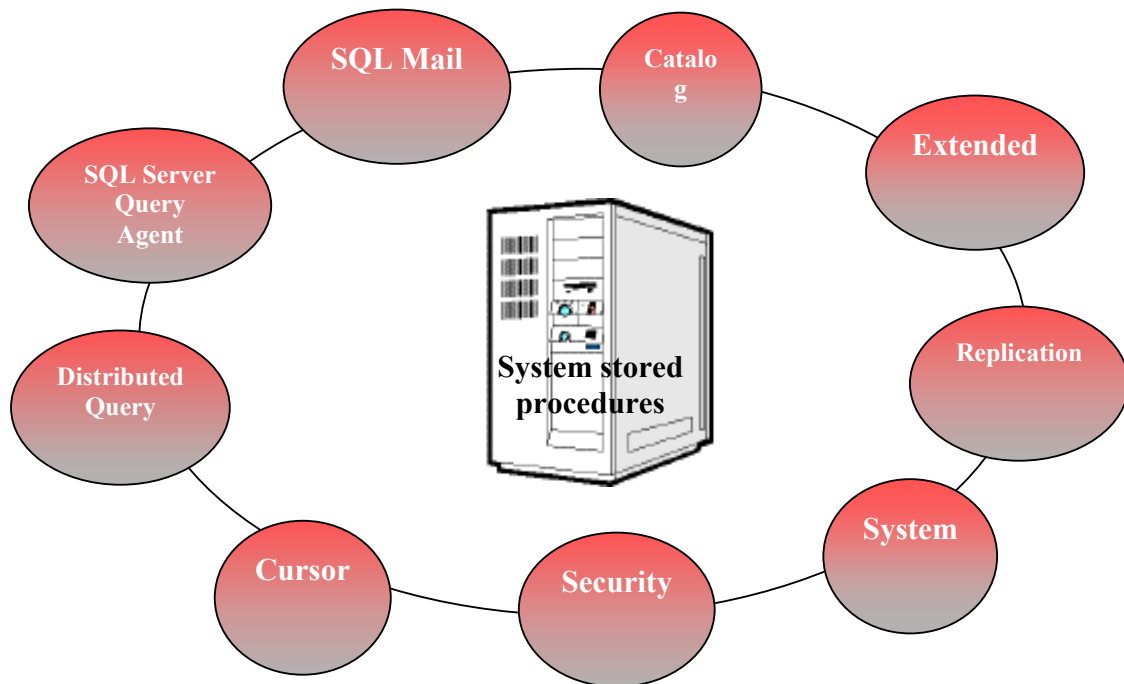
14.3.1 System stored procedures

Là những stored procedure chứa trong Master Database và thường bắt đầu bằng tiếp đầu ngữ **sp_**.

Các stored procedure này thuộc loại built-in và chủ yếu dùng trong việc quản lý cơ sở dữ liệu (administration) và bảo mật (security). Ví dụ bạn có thể kiểm tra tất cả các processes đang được sử dụng bởi user DomainName\Administrators bạn có thể dùng :

```
sp_who @loginame='DomainName\Administrators'
```

Người ta có thể chia các System stored procedures thành các loại sau:



Hình 14.2. Các loại của System stored procedures

Có hàng trăm system stored procedure trong SQL Server. Bạn có thể xem chi tiết phân loại và nội dung của từng thủ tục trong SQL Server Books Online.

Sau đây là một số thủ tục hệ thống thường sử dụng:

System stored procedure	Chức năng
sp_Databases	Danh sách những Database có thể (available) trên Server (Danh sách này sẽ là khác nhau tùy thuộc vào quyền của người sử dụng)
sp_server_info	Chi tiết những thông tin về Server, ví dụ như tập các đặc

Chương 14. STORED PROCEDURE

	tính, phiên bản...
sp_stored_procedures	Danh sách tất cả các thủ tục có thể trên môi trường hiện tại
sp_tables	Danh sách tất cả các bảng có thể trên môi trường hiện tại
sp_start_job	Khởi động tất cả các automated task ngay lập tức
sp_stop_job	Ngừng lại tất cả các automated task đang chạy
sp_password	Thay đổi password cho login account
sp_configure	Thay đổi lựa chọn cấu hình chung của SQL SERVER. Khi người sử dụng không lựa chọn thì hệ thống sẽ hiển thị cấu hình mặc định.
sp_help	Hiển thị thông tin về bất kỳ đối tượng nào trong Database
sp_helptext	Hiển thị nội dung (text) của các đối tượng

14.3.2 User-defined Stored Procedures

14.3.2.1 Cú pháp

Người sử dụng có thể sử dụng câu lệnh CREATE PROCEDURE để tạo thủ tục trong CSDL hiện tại.

Database owner mặc định có quyền sử dụng câu lệnh CREATE PROCEDURE.

Cú pháp:

```
CREATE PROC[EDURE] procedure name
```

Ví dụ:

```
CREATE PROCEDURE London_Flights AS  
PRINT 'This code displays the details of flights to  
London'  
SELECT * FROM flight WHERE destination='Lon'
```

14.3.2.2 Các chỉ dẫn

Tên thủ tục phải tuân theo quy tắc đặt tên

Tất cả các đối tượng của cơ sở dữ liệu có thể được tạo trong SPs, trừ những đối tượng: defaults, rules, triggers, procedures, và views.

Những đối tượng đã được tạo có thể được tham chiếu đến ngay khi nó được tạo.

Stored procedures có thể tham chiếu tới những bảng phụ (temporary tables).

Có thể có 2100 biến trong stored procedure.

Chúng ta có thể tạo nhiều biến địa phương trong stored procedure nếu bộ nhớ cho phép.

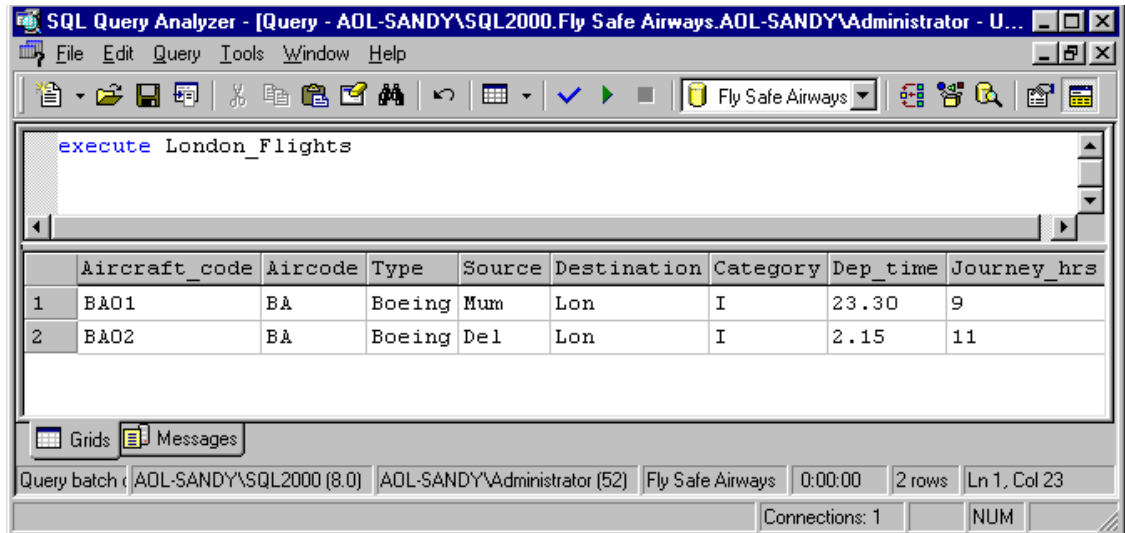
Kích thước tối đa cho stored procedure là 128 MB.

14.3.2.3 Thực hiện User-defined Stored Procedures

Cú pháp:

```
EXEC[UTE] procedure name
```

Ví dụ:



Hình 14.3. Thực hiện User-defined Stored Procedures

14.3.2.4 Sử dụng biến trong Stored Procedures

Biến có thể được sử dụng để nhập dữ liệu vào (INPUT) hoặc xuất dữ liệu ra ngoài (OUTPUT)

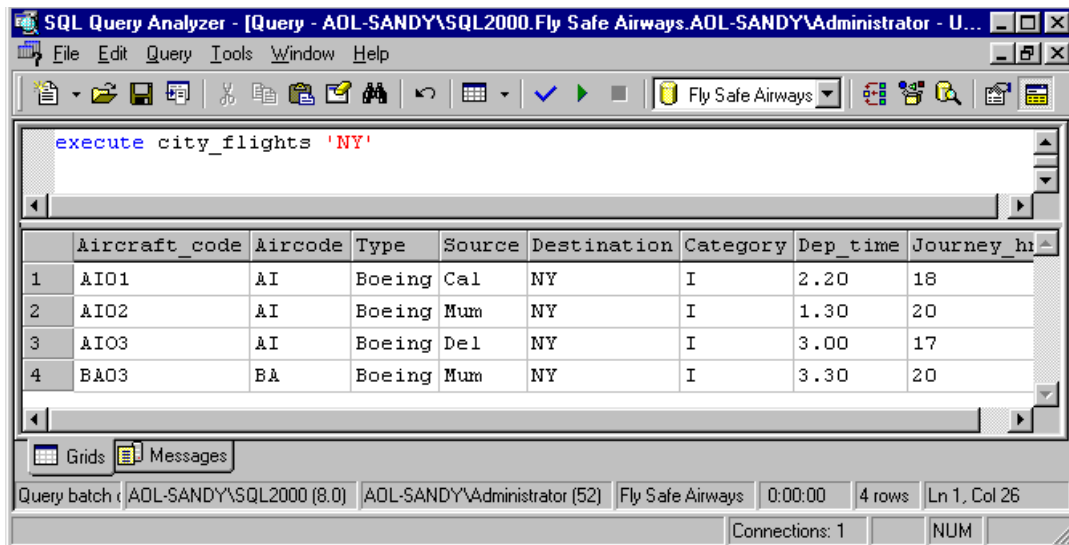
Cú pháp:

```
CREATE PROCEDURE procedure_name  
    @Parameter_name data_type  
AS  
:
```

Ví dụ:

```
CREATE PROCEDURE city_flights  
    @v_city varchar(15)  
AS  
SELECT * FROM flight WHERE destination=@v_city
```

Thực hiện thủ tục có biến:



Hình 14.4. Thực hiện User-defined Stored Procedures có biến

Nếu có nhiều biến trong thủ tục thì khi thực hiện ta liệt kê theo thứ tự các biến và phải cách nhau bằng dấu phẩy.

14.3.2.5 Biên dịch lại - Re-compiling Stored Procedures

Khi người sử dụng làm thay đổi tới những index của bảng. Stored procedures phải được biên dịch lại (recompiled) để chấp nhận những thay đổi đó.

Có 3 cách để biên dịch lại procedures:

Sử dụng `sp_recompile` system stored procedure: Bạn có thể sử dụng cách này để biên dịch lại thủ tục ở lần chạy kế tiếp của nó.

Cú pháp:

```
sp_recompile [@objectname=] 'object'
```

Chỉ ra `WITH RECOMPILE` trong câu lệnh `CREATE PROCEDURE`: SQL Server sẽ biên dịch lại thủ tục ở mỗi lần nó thực hiện.

Cú pháp:

```
CREATE PROCEDURE procedure_name  
    @Parameter_name data_type  
WITH RECOMPILE  
AS  
    :
```

Chỉ ra `WITH RECOMPILE` trong câu lệnh `EXECUTE`:

Biên dịch lại ngay ở lần thực hiện này.

Cú pháp:

```
EXEC[UTE] procedure_name WITH RECOMPILE
```

14.3.2.6 Sửa cấu trúc của Stored Procedures

Câu lệnh ALTER PROCEDURE được sử dụng để sửa SP.

Cú pháp tương tự CREATE PROCEDURE chỉ thay từ CREATE bằng ALTER.

Việc sửa chữa vẫn lưu lại quyền của người sử dụng (user permissions)

14.4 Thông báo lỗi

Trả về mã lỗi (Code) hoặc câu lệnh RAISERROR có thể được sử dụng để nhắc người sử dụng về lỗi.

Mã lỗi trả về là số nguyên.

Câu lệnh RAISERROR giải thích lỗi và chỉ ra mức độ lỗi.

14.4.1 Return Codes

Return codes là số nguyên, giá trị mặc định là 0.

Giá trị của **Return codes** phải được trả về vào một biến

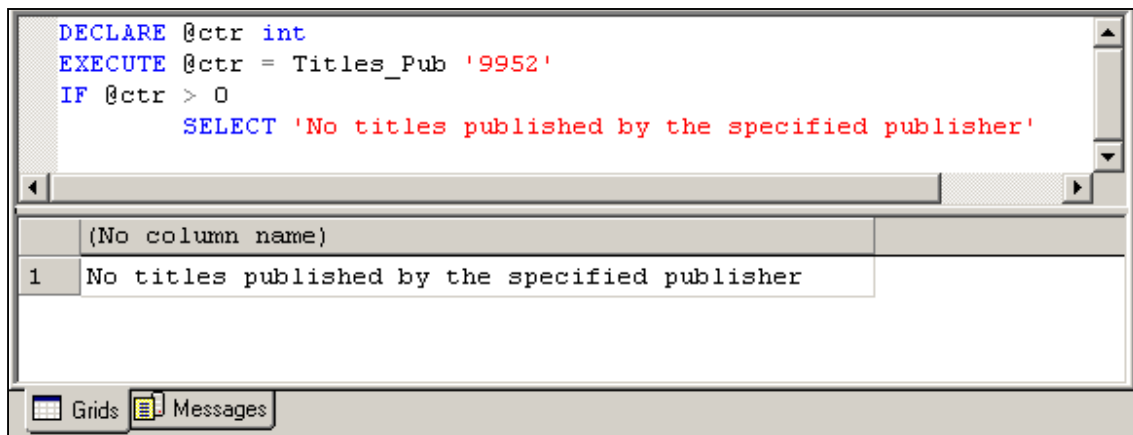
Cú pháp:

```
DECLARE @return_variable_name data_type  
EXECUTE @return_variable_name = procedure_name
```

Ví dụ:

```
ALTER PROCEDURE Titles_Pub  
@v_pubid char(4)  
AS  
DECLARE @v_return int  
SELECT @v_return=COUNT(*)  
FROM titles WHERE pub_id = @v_pubid  
IF @v_return>0  
    SELECT * FROM titles WHERE pub_id = @v_pubid  
ELSE  
    RETURN @v_return+1
```

Kết quả thực hiện:



```
DECLARE @ctr int
EXECUTE @ctr = Titles_Pub '9952'
IF @ctr > 0
    SELECT 'No titles published by the specified publisher'
```

(No column name)
1 No titles published by the specified publisher

14.4.2 Câu lệnh RAISERROR

Trong SPs, chúng ta có thể sử dụng câu lệnh PRINT để hiển thị thông báo lỗi cho người sử dụng. Tuy nhiên, những lời nhắc này chỉ là tạm thời và chỉ hiển thị cho người sử dụng chúng ta cần sử dụng câu lệnh RAISERROR để ghi lại những lỗi này và gán cho nó mức severity.

Cú pháp:

```
RAISERROR ({msg_id | msg_str}{,severity, state}
[WITH option[...n]])
```

Ví dụ:

```
WHILE @v_ctr > 0
BEGIN
    SELECT @v_ctr * @v_ctr
    SELECT @v_ctr = @v_ctr - 1
    IF @v_ctr = 2
    BEGIN
        RAISERROR('Counter has fallen below 3', 1, 2)
        BREAK
    END
END
```

Kết quả:

```
25
16
9
```

Nội dung thông báo:

```
(1 row(s) affected)
```

Chương 14. STORED PROCEDURE

(1 row(s) affected)

(1 row(s) affected)

Msg 50000, Level 1, State 50000

Counter has fallen below 3

14.5 Câu hỏi trắc nghiệm

1. Ký hiệu nào theo sau đứng trước tên biến trong câu lệnh EXECUTE?

- A &
- B #
- C ?
- D @

2. Câu lệnh nào sau đây sẽ kết thúc thực hiện thủ tục (Những câu lệnh đằng sau câu lệnh này sẽ không được thực hiện)?

- A RETURN
- B EXIT
- C HALT
- D FINISH

3. Người dùng nào sau đây có quyền mặc định để chạy Stored Procedure?

- A Data Owner
- B Row Owner
- C Table Owner
- D Database Owner

4. Hiệu quả của việc chỉ ra WITH RECOMPILE trong khi định nghĩa Stored Procedure (SP) là gì?

- A SP được biên dịch lại ở ngay lần tiếp theo nó thực hiện
- B SP được biên dịch lại ở tất cả các lần nó thực hiện
- C SP được biên dịch lại khi SQL Server khởi động
- D SP được biên dịch lại khi có chỉ số được tạo trên các bảng mà nó tham chiếu tới

5. Chúng ta có thể sửa chữa được những thủ tục hệ thống (System stored procedures)

- A Đúng
- B Sai

6. Thông báo được định nghĩa cùng RAISERROR có thể chứa đựng tối đa bao nhiêu ký tự?

- A 64
- B 510
- C 255
- D 128

15 Chương 15. STORED PROCEDURE Phần thực hành

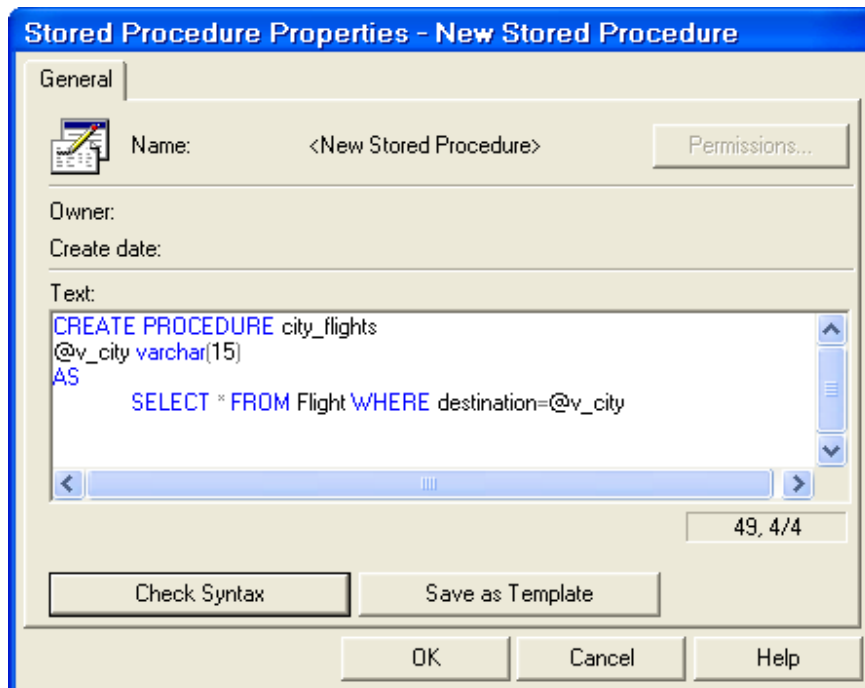
Mục đích:

- Tạo SP bằng Wizard (hướng dẫn bằng hình ảnh)
- Sử dụng QA để tạo SP
- Sử dụng biến và các câu lệnh điều khiển trong khi tạo SP
- Thực hiện SP
- Biên dịch lại (Recompile) SP.
- Sửa SP
- Thực hiện các SP hệ thống (System Stored Procedures)

15.1 Tạo SP bằng EM.

Thực hiện các bước sau:

1. Chọn cơ sở dữ liệu FI.
2. Kích phải chuột vào đối tượng Stored Procedure, kích New Stored Procedure...
3. Nhập nội dung của SP.



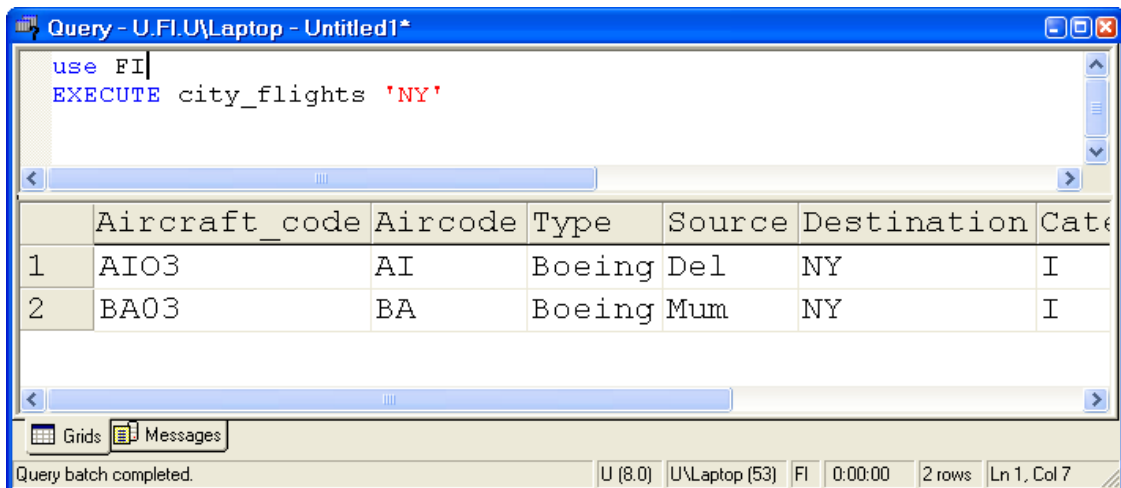
4. Kích Check Syntax để kiểm tra cú pháp.

5. Kịch OK

Chúng ta cũng có thể sử dụng Wizard để tạo SP. Thực hiện lần lượt theo các bước hệ thống hướng dẫn.

15.2 Thực thi SP

Để thực thi SP, chúng ta sử dụng câu lệnh EXECUTE trong QA.



Chúng ta có thể thực hiện sửa và xoá SP.

15.3 Bài tập

Thực hiện các yêu cầu sau bằng QA:

Thực hiện các công việc sau bằng QA.

1. Tạo thủ tục với tên Display_Service để hiển thị các hãng bay có dịch vụ chăm sóc trẻ em 'Child Care'.
2. Sửa thủ tục Display_Service để chấp nhận nhận hãng bay (airline) và tên dịch vụ như là biến truyền vào.
 - a. Nếu dịch vụ đó được phục vụ trên hãng bay thì đưa ra thông báo: "Hãng bay <airline_name> có phục vụ dịch vụ <service_name>"
 - b. Nếu biến airline_name truyền vào thủ tục có giá trị là '*' thì hiển thị tất cả các hãng bay có phục vụ dịch vụ <service_name>
 - c. Nếu biến service_name truyền vào thủ tục có giá trị là '*' thì hiển thị tất cả các dịch vụ được phục vụ trên hãng bay <airline_name>
3. Thực hiện thủ tục trên với dịch vụ service_name= 'Wheel Chair' và hãng bay airline_name= 'Jet Airways'. Đảm bảo rằng thủ tục sẽ được biên dịch lại khi thực hiện.
4. Hiển thị tên tất cả các thủ tục trong cơ sở dữ liệu Flight Information

Chương 15. STORED PROCEDUREn- Phần thực hành

5. Hiện thị đoạn mã lệnh của thủ tục Display_Service
6. Hiện thị cấu hình hiện tại của Server

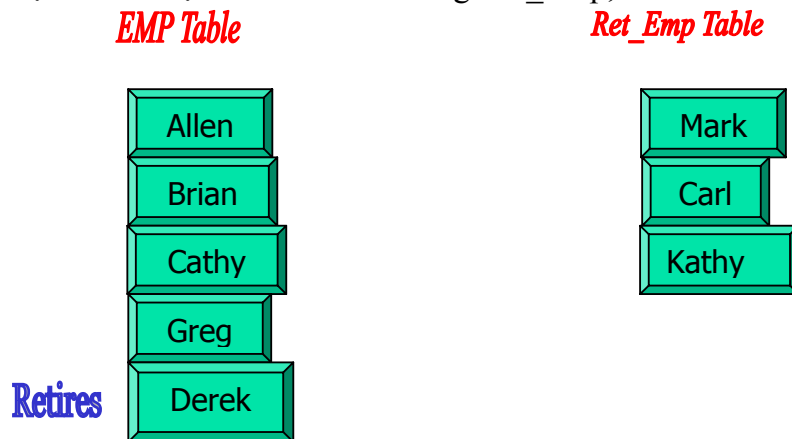
16 Chương 16. TRIGGER

16.1 Định nghĩa

Trigger là một loại stored procedure đặc biệt được thực thi một cách tự động khi có một sự kiện thay đổi dữ liệu (data modification event) xảy ra như Update, Insert hay Delete. Trigger được dùng để đảm bảo ràng buộc dữ liệu, tính nhất quán, hoặc thực hiện các quy tắc dữ liệu phức tạp.

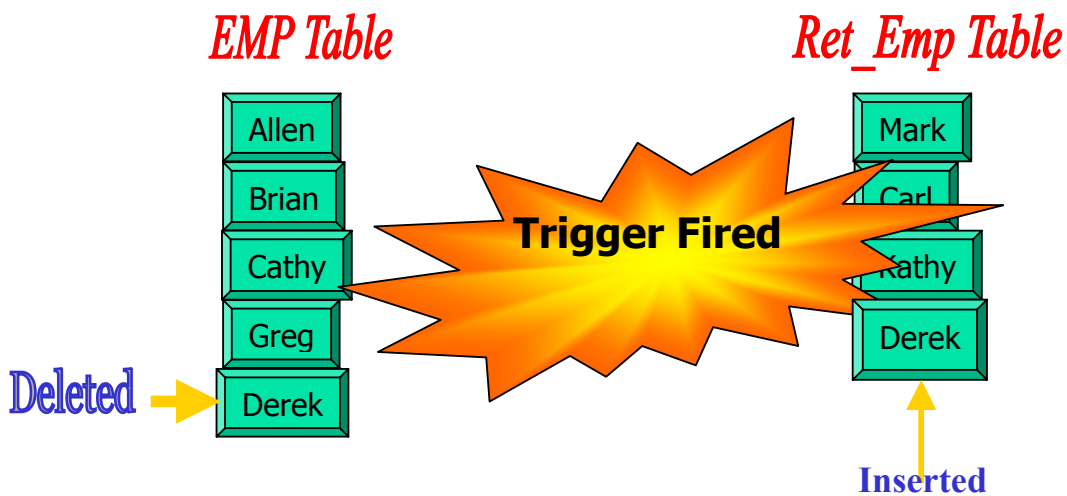
Trigger là đối tượng đặc biệt được tạo trên một bảng và là một phần của cơ sở dữ liệu. Trigger có thể truy vấn tới các bảng khác và có thể bao gồm các câu lệnh T-SQL phức tạp. Chúng ta có thể tạo trigger cho mỗi hành động trên bảng hoặc có thể kết hợp 2 hoặc 3 hành động trong một trigger.

Ví dụ: Khi muốn thực hiện một thao tác DELETE trên bảng EMP thì đồng thời dữ liệu bị xóa sẽ được INSERT vào bảng Ret_Emp, như hình dưới đây:



Hình 16.1. Dữ liệu trước khi thực hiện Delete Trigger

Sau khi thực hiện trigger DELETE, ta có kết quả sau:



Hình 16.2. Dữ liệu sau khi thực hiện Delete Trigger

Ghi nhớ: Trong trường hợp có nhiều trigger cùng liên quan đến một đối tượng thì ta phải xác định thứ tự thực hiện của chúng. Thủ tục hệ thống `sp_settriggerorder` cho phép chúng ta làm điều này.

Khi nào ta cần sử dụng Trigger:

Ta chỉ sử dụng trigger khi mà các biện pháp bảo đảm data integrity khác như Constraints không thể thỏa mãn yêu cầu của ứng dụng. Nên nhớ Constraint thuộc loại **Declarative Data Integrity** cho nên sẽ **kiểm tra data trước khi cho phép nhập vào table** trong khi Trigger thuộc loại Procedural Data Integrity nên việc insert, update, delete đã xảy ra rồi mới kích hoạt trigger. Chính vì vậy mà ta cần cân nhắc trước khi quyết định dùng loại nào trong việc đảm bảo Data Integrity.

Khi một cơ sở dữ liệu được phi chuẩn –denormalized (ngược lại quá trình chuẩn hóa -normalization), sẽ có một số dữ liệu thừa (redundant) được chứa trong nhiều tables. Nghĩa là sẽ có một số **dữ liệu được chứa cùng một lúc ở hai hay nhiều nơi khác nhau**. Khi đó để đảm bảo tính chính xác thì khi dữ liệu được cập nhật ở một bảng này thì cũng phải được cập nhật một cách tự động ở các bảng còn lại bằng cách dùng Trigger.

Ví dụ: Ta có table Item trong đó có field Barcode dùng để xác định một mặt hàng nào đó. Item table có vai trò như một cuốn catalog chứa những thông tin cần thiết mô tả từng mặt hàng. Ta có một table khác là Stock dùng để phản ánh món hàng có thực trong kho như được nhập về này nào được cung cấp bởi đại lý nào, số lượng bao nhiêu (tức là những thông tin về món hàng mà không thể chứa trong Item table được). Bảng table này cũng có field Barcode để xác định món hàng trong kho. Như vậy thông tin về Barcode được chứa ở hai nơi khác nhau do đó ta cần dùng trigger để đảm bảo là Barcode ở hai nơi luôn được đồng bộ.

Đôi khi ta có nhu cầu thay đổi dây chuyền (cascade) ta có thể dùng Trigger để bảo đảm chuyện đó. Nghĩa là khi có sự thay đổi nào đó ở table này thì một số table khác cũng được thay đổi theo để đảm bảo tính chính xác. Ví dụ như khi một món hàng được bán đi thì số lượng hàng trong table Item giảm đi một món đồng thời tổng số hàng trong kho (Stock table) cũng phải giảm theo một cách tự động. Như vậy ta có thể tạo một trigger trên Item table để mỗi khi một món được bán đi thì trigger sẽ được kích hoạt và giảm tổng số hàng trong Stock table.

16.2 Đặc điểm của Trigger

Một trigger có thể làm nhiều công việc khác nhau và có thể được kích hoạt bởi nhiều hơn một sự kiện(event). Ví dụ ta có thể viết một trigger được kích hoạt bởi bất kỳ event nào như Update, Insert hay Delete và bên trong trigger ta sẽ viết code để giải quyết cho từng trường hợp.

Trigger không thể được tạo ra trên temporary hay system table.

Trigger chỉ có thể được kích hoạt một cách tự động bởi một trong các event Insert, Update, Delete mà không thể chạy một mình được.

Có thể áp dụng trigger cho View.

Khi một trigger được kích hoạt thì dữ liệu mới vừa được insert hay mới vừa được thay đổi sẽ được chứa trong **Inserted** table còn dữ liệu mới vừa được delete được chứa trong **Deleted** table. Đây là 2 table tạm chỉ chứa trên memory và chỉ có giá trị bên trong trigger mà thôi (nghĩa là chỉ nhìn thấy và được query trong trigger mà thôi). Ta có thể dùng thông tin trong 2 table này để so sánh dữ liệu cũ và mới hoặc kiểm tra xem dữ liệu mới vừa thay đổi có hợp lệ trước khi commit hay roll back.

16.3 Tạo Trigger

16.3.1 Tạo Trigger

Triggers có thể được tạo bằng cách sử dụng Enterprise Manager, hoặc Query Analyzer.

Trong cả hai trường hợp, câu lệnh **CREATE TRIGGER** được sử dụng để tạo trigger.

Cú pháp:

```
CREATE TRIGGER Trigger_name
ON table
FOR [DELETE, INSERT, UPDATE]
[WITH ENCRYPTION]
AS Sql statements
```

16.3.2 Hướng dẫn khi tạo Trigger

- Trigger có thể hỗ trợ một số tác động trên bảng dữ liệu: INSERT, UPDATE, và DELETE.
- Lựa chọn **WITH ENCRYPTION** có thể được sử dụng để mã hóa định nghĩa của trigger, để người dùng không thể xem. Tuy nhiên, trigger được mã hóa không thể được giải mã.
- Trigger có thể tham chiếu tới View hoặc các bảng tạm, bảng hệ thống. Nhưng Trigger không thể được tạo trên bảng tạm và bảng hệ thống.
- Trigger có thể chứa số lượng bất kỳ các câu lệnh SQL.
- Triggers truy cập hai bảng logic gọi là **Inserted** và **Deleted**.
- Inserted và Deleted tables chứa hình ảnh của dữ liệu trước và sau khi cập nhật.

Chương 16. TRIGGER

- Dữ liệu ở trong bảng sẽ không bị ảnh hưởng bởi phép toán cập nhật nếu nó không có trong bảng Inserted và Deleted

Nội dung của 2 bảng Inserted và Deleted:

Kiểu Trigger	Inserted Table	Deleted Table
UPDATE	Lưu trữ bản sao của các bản ghi được cập nhật khi câu lệnh kết thúc	Lưu trữ những bản ghi trước khi cập nhật
DELETE	Không sử dụng	Lưu trữ những bản ghi bị xóa
INSERT	Lưu trữ những bản sao của những bản ghi được thêm.	Không sử dụng

16.4 Các kiểu Trigger

Có 3 kiểu trigger:

- INSERT trigger
- DELETE trigger
- UPDATE trigger

16.4.1 INSERT trigger

Thực hiện bất cứ khi nào có sự thêm dữ liệu vào bảng

Cách thực hiện của INSERT trigger:

Thêm bản sao của những dòng dữ liệu được thêm vào Inserted table.

Kiểm tra những dòng dữ liệu đó ở trong Inserted table, để xác định xem nó có hợp lệ không.

Nếu hợp lệ thì thêm những dòng đó vào trigger table.

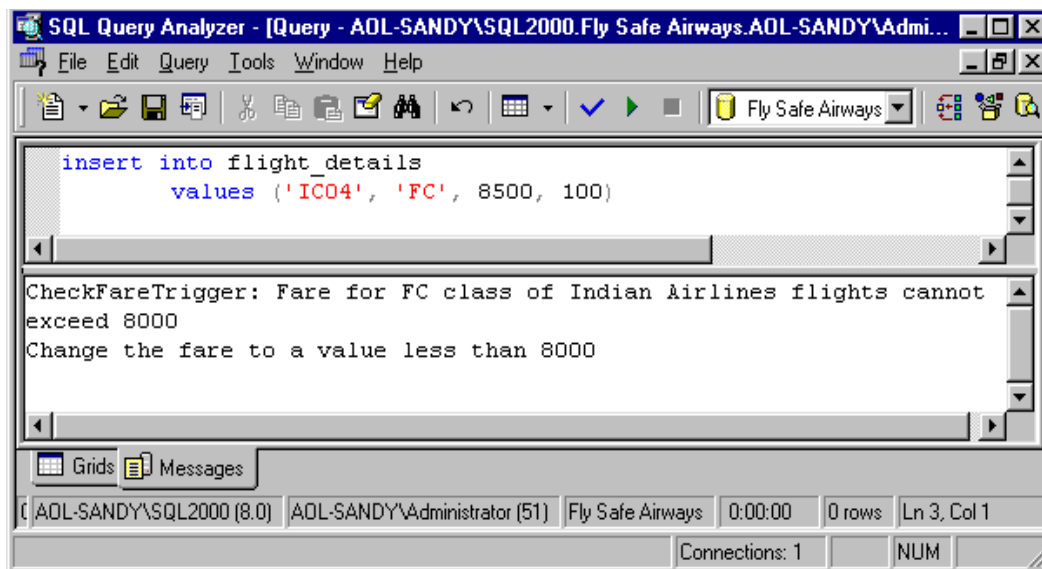
Ví dụ: Ví dụ sau tạo trigger có tên là CheckFare tác động lên bảng Flight_detail mỗi khi có hành động thêm (INSERT) dữ liệu. Trigger này đảm bảo rằng giá vé của hạng 'FC' trên chuyến bay Indian Airlines không được vượt quá 8000. Câu lệnh như sau:

```
CREATE TRIGGER CheckFare ON flight_details FOR INSERT AS
IF (SELECT fare FROM INSERTED AS i JOIN flight AS f
    ON i.aircraft_code = f.aircraft_code JOIN airlines_master AS am
    ON f.aircode = am.aircode WHERE i.class_code = 'FC'
    AND am.airline_name = 'Indian Airlines') > 8000
```


Chương 16. TRIGGER

```
BEGIN
    PRINT 'CheckFareTrigger: Fare for FC class of Indian Airlines
    flights cannot exceed 8000'
    PRINT 'Change the fare to a value less than 8000'
    ROLLBACK TRANSACTION
END
```

Kết quả:



Hình 16.3

16.4.2 UPDATE trigger

Thực hiện bất cứ khi nào có sự cập nhật dữ liệu trong bảng.

Cách thực hiện của UPDATE trigger:

- Chuyển những dòng dữ liệu cũ (trước khi cập nhật) vào Deleted table.
- Thêm những dòng có giá trị mới vào Inserted table, và trigger table.
- Kiểm tra lại giá trị ở trong Deleted và Inserted tables nếu có bất cứ yêu cầu liên quan nào.

Có thể được tạo để thực hiện cập nhật trên một cột hoặc trên toàn bộ bảng. Vì thế, người ta chia UPDATE trigger thành 2 loại:

16.4.2.1 Column Level

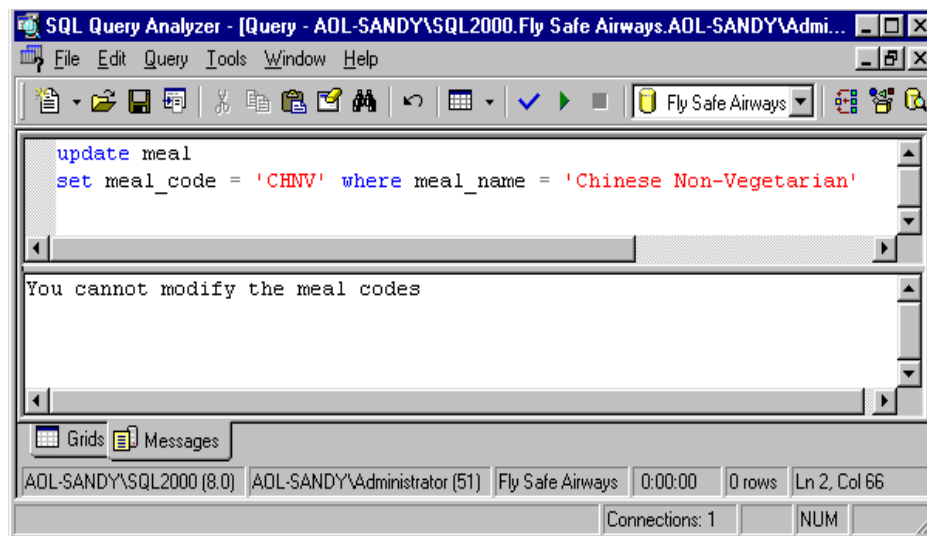
Trong ví dụ này, chúng ta tạo ra một UPDATE trigger tác động lên cột Meal_code của bảng Meal. Trigger này sẽ thực hiện mỗi khi có hành động cập nhật dữ liệu trên cột Meal_code. Câu lệnh như sau:

Chương 16. TRIGGER

```
CREATE TRIGGER NoUpdateMealcode
ON Meal
FOR UPDATE AS
IF UPDATE (Meal_code)
BEGIN
    PRINT 'You cannot modify the meal
codes'
    ROLLBACK TRANSACTION
END
```

Sau khi tạo ra trigger, sẽ không có phép toán cập nhật nào được thực hiện trên cột Meal_code, và hệ thống sẽ trả lại thông báo như trên:

Kết quả:



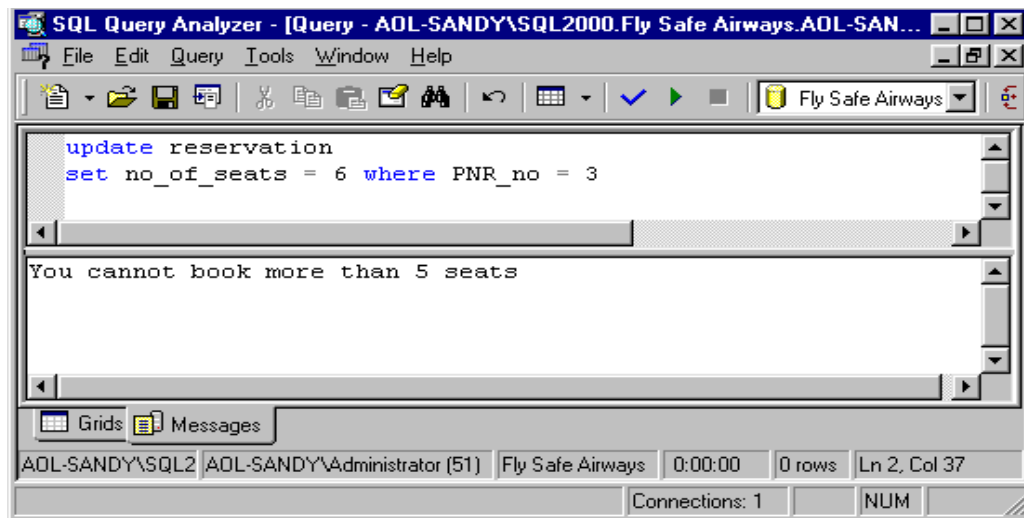
Hình 16.4

16.4.2.2 Table Level

Trong ví dụ này, chúng ta sẽ tạo ra một trigger đảm bảo rằng cột no_of_seats của bảng Reservation không được phép thay đổi giá trị lớn hơn 5. Trigger này sẽ thực hiện **mỗi lần** cập nhật **bất cứ** cột nào trên bảng (Đây là điểm khác biệt với Column Level).

```
CREATE TRIGGER NoUpdateSeats
ON Reservation
FOR UPDATE AS
IF (SELECT no_of_seats FROM inserted) > 5
BEGIN
    PRINT 'You cannot book more than 5
seats'
    ROLLBACK TRANSACTION
END
```

Kết quả thực hiện trigger trên khi có hành động Update xảy ra:



Hình 16.5

16.4.3 DELETE trigger

Chúng ta sẽ tạo DELETE trigger khi chúng ta muốn có tác động lên thao tác Delete dữ liệu của bảng.

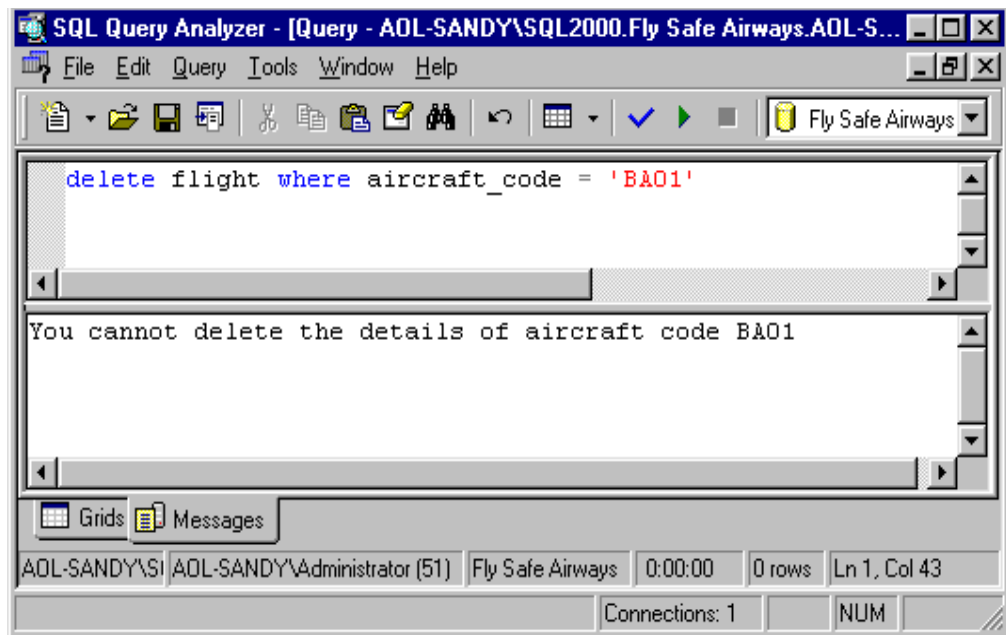
Cách thực hiện của DELETE trigger:

- Xoá dòng dữ liệu từ trigger table.
- Thêm những dòng bị xoá đó vào Deleted table.
- Kiểm tra những dòng dữ liệu trong Deleted table để thực hiện các câu lệnh của trigger

Ví dụ:

```
CREATE TRIGGER NoDeleteBA01
ON Flight
FOR DELETE AS
IF (SELECT aircraft_code FROM deleted)= 'BA01'
BEGIN
    PRINT 'You cannot delete the details of
aircraft code BA01'
    ROLLBACK TRANSACTION
END
```

Kết quả:



Hình 16.6

16.5 Các câu lệnh không thể sử dụng trong Triggers.

Danh sách các câu lệnh SQL không thể bao gồm trong triggers.

CÂU LỆNH SQL
CREATE Database
ALTER Database
DROP Database
LOAD Database
RECONFIGURE
RESTORE LOG
DISK INIT
DISK RESIZE
LOAD LOG
RESTORE Database

16.6 Triggers dây chuyền - Cascading Triggers

Cascading triggers thực hiện hiệu quả đối với các ràng buộc toàn vẹn tham chiếu.

Chương 16. TRIGGER

Cascading triggers sửa dữ liệu ở các bảng liên quan khi có sự thay đổi dữ liệu xảy ra trên một bảng

Triggers không thể thực hiện cập nhật và xoá “cascade” nếu nó làm ảnh hưởng đến ràng buộc khoá chính và khoá ngoại (foreign and primary key constraints).

Triggers được thực hiện sau khi kiểm tra ràng buộc, nếu có một ràng buộc bị vi phạm thì trigger sẽ không thực hiện.

16.7 Triggers lồng nhau - Nested Triggers

Thực hiện trigger lồng nhau khi thực hiện trigger này cần kết quả từ một trigger khác.

Triggers có thể lồng nhau tối đa 32 cấp.

Để có thể sử dụng được triggers lồng nhau ta phải thiết đặt lại thủ tục hệ thống `sp_configure` như sau:

`sp_configure 'nested trigger', 1`

Ngược lại, để làm mất tác dụng của triggers lồng nhau:

`sp_configure 'nested trigger', 0`

16.8 INSTEAD OF Triggers

Trong các phiên bản trước, chúng ta không thể thực hiện được phát biểu INSERT, UPDATE, DELETE trên dữ liệu của Views.

SQL Server khắc phục nhược điểm này, muốn thao tác với dữ liệu trên View, ta phải tạo trigger cho View đó.

Vì thế, INSTEAD OF trigger chứa mã lệnh **thay thế** cho những câu lệnh thực hiện dữ liệu **nguyên thủy** (INSERT, UPDATE, DELETE)

Xem xét ví dụ sau đây, chúng ta nhìn thấy View sau đây có điều kiện kết nối, cái mà không hỗ trợ thao tác xoá dữ liệu. INSTEAD OF trigger cung cấp giải pháp cho vấn đề này.

Câu lệnh tạo View:

```
CREATE VIEW service_view
AS
    SELECT      s.service_code      AS      scode1,
    service_name, a.service_code AS scode2, aircode
    FROM service s JOIN airline_service a
    ON s.service code=a.service code
```

Ta cố gắng thực hiện xoá dữ liệu trên View này bằng câu lệnh:

```
DELETE service_view WHERE scode1= 'CC'
```

Chương 16. TRIGGER

Kết quả thực hiện sẽ báo lỗi:

View or function 'service_view' is not updatable because the modification affects multiple base tables.

Vì service_view được xác định trên 2 bảng nên chúng ta muốn xoá dữ liệu phải sử dụng INSTEAD OF trigger. Định nghĩa như sau:

```
CREATE TRIGGER del_service
ON service_view
INSTEAD OF DELETE
AS
    DELETE service WHERE service_code IN
        (SELECT scode1 FROM DELETED)
    DELETE airline_service WHERE service_code
IN
        (SELECT scode2 FROM DELETED)
```

Sau khi có trigger này, câu lệnh DELETE được định nghĩa ở đây sẽ thay thế cho câu lệnh DELETE dữ liệu nguyên thủy và ta dễ dàng thực hiện được câu lệnh DELETE trên.

```
DELETE service view WHERE scode1= 'CC'
```

Lúc này, dữ liệu cũng được xoá tương ứng trên 2 bảng Airline_Service và Service.

16.9 Câu hỏi trắc nghiệm

1. Kiểu trigger nào sau đây là trigger chứa những câu lệnh thay thế cho những câu lệnh thực hiện nguyên gốc?

- A Cascade
- B Table Level
- C Column Level
- D INSTEAD OF

2. Trigger có thể lồng nhau bao nhiêu cấp?

- A 8
- B 16
- C 32
- D 256

3. Lựa chọn WITH ENCRYPTION trong câu lệnh CREATE TRIGGER làm chức năng gì?

- A Mã hóa nội dung trong trigger
- B Mã hóa cơ sở dữ liệu nơi mà có trigger định nghĩa
- C Mã hóa dữ liệu trong cơ sở dữ liệu nơi mà có trigger định nghĩa

4. Cascade Trigger có thể xóa và cập nhật cả những dữ liệu mà ảnh hưởng đến ràng buộc khóa chính và khóa ngoại

- A Đúng
- B Sai

5. Những bảng logic nào sau đây được sử dụng trong Trigger

- A Temp
- B Inserted
- C Hold
- D Deleted

6. Những câu lệnh nào sau đây có thể có trong Trigger

- A Rollback Transaction
- B Disk resize
- C Restore log
- D Load Database

17 Chương 17. TRIGGER – Phần thực hành

Mục đích:

- Tạo CREATE, DELETE, UPDATE và INSTEAD OF trigger.
- Mã hoá (encrypt) trigger.
- Xem cách trigger thực hiện.
- Hiển thị các thông tin về trigger.

17.1 Tạo INSERT trigger.

INSERT trigger đảm bảo dữ liệu nhập vào bảng được đúng đắn.

Xem xét ví dụ tạo INSERT trigger để đảm bảo không có vé nào được đặt vào một ngày trong quá khứ.

Các bước thực hiện:

1. Mở QA, chọn cơ sở dữ liệu FI.
2. Thực hiện đoạn lệnh sau trong QA.

```
CREATE TRIGGER insert_trigg
ON Reservation
FOR INSERT
AS
IF((Select journey_date From Inserted)<getdate())
BEGIN
PRINT 'journey_date không thể nhỏ hơn ngày hiện tại'
ROLLBACK TRAN
END
```

3. Sau đó, hãy thử thực hiện thêm một bản ghi có journey_date < ngày hiện tại của hệ thống.

17.2 Tạo DELETE Trigger

DELETE trigger ngăn cản việc xoá đi những dữ liệu quan trọng trong bảng.

Xem xét ví dụ sau: Tạo trigger để tránh xoá 2 bản ghi trong bảng Passenger đồng thời.

1. Trong QA thực hiện như sau:

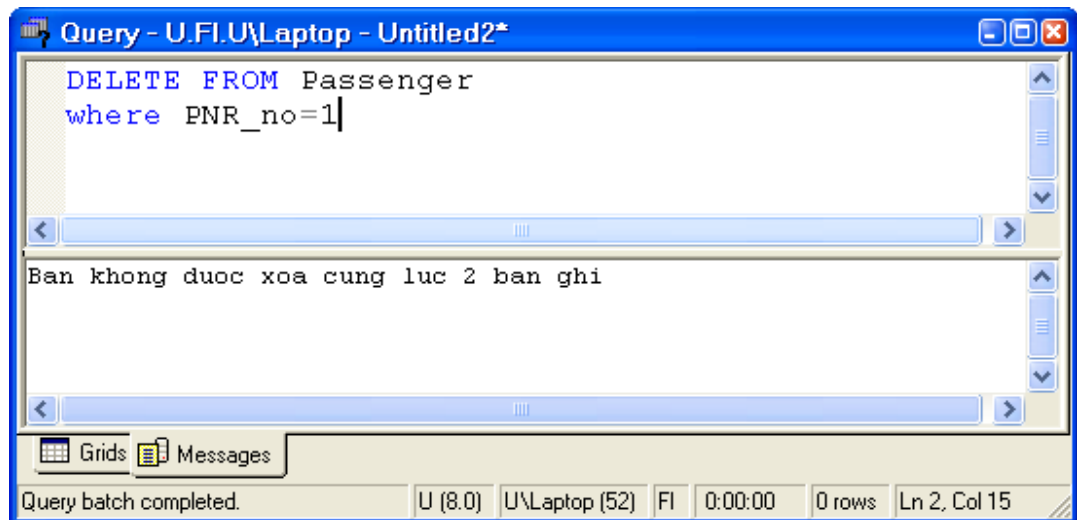
```
CREATE TRIGGER delete_trigg
ON Passenger
FOR Delete
AS
IF((Select count(*) From deleted)>2)
BEGIN
```


Chương 17. TRIGGER – Phần thực hành

```
PRINT 'Ban khong duoc xoa cung luc 2 ban ghi'  
ROLLBACK TRAN  
END
```

2. Thực hiện câu lệnh xoá nhiều hơn 2 bản ghi từ bảng Passenger, giả sử như sau:

Kết quả:



Hình 17.1

17.3 Tạo UPDATE Trigger.

17.3.1 Tạo Table Level UPDATE Trigger.

Trigger UPDATE sẽ được thực hiện bất cứ khi nào dữ liệu trong bảng được cập nhật.

Xem xét ví dụ: Tạo UPDATE trigger đảm bảo rằng cột `No_of_seats` trong bảng `Reservation` không được cập nhật giá trị lớn hơn 5 và `journey_date` không nhỏ hơn ngày hiện tại.

1. Thực hiện như sau trong QA.

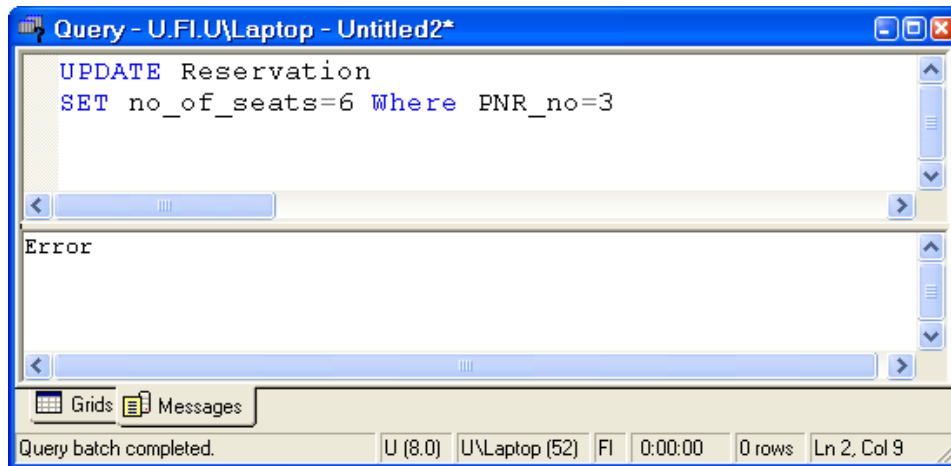
```
CREATE TRIGGER CheckingUpdate  
ON Reservation  
FOR UPDATE  
AS  
IF((Select no_of_seats From inserted)>5)  
OR ((Select journey_date From Inserted)<getdate())  
BEGIN  
    PRINT 'Error'  
    ROLLBACK TRAN  
END
```

Chương 17. TRIGGER – Phần thực hành

2. Thực hiện truy vấn sau để kiểm tra Trigger:

```
UPDATE Reservation  
SET no_of_seats=6 Where PNR_no=3
```

Kết quả:



Hình 17.2

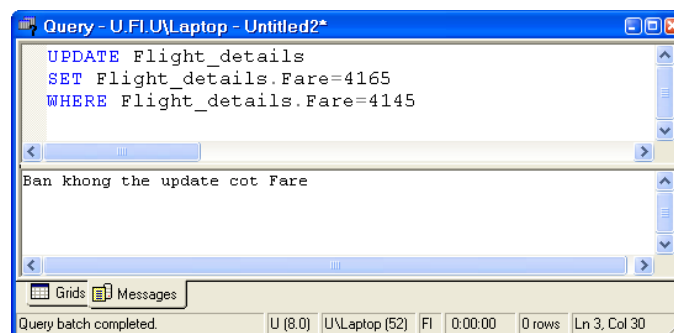
17.3.2 Tạo Column Level Update Trigger

Loại Trigger được thực hiện khi dữ liệu trong cột nào đó được cập nhật.

1. Thực hiện như sau trong QA:

```
CREATE TRIGGER Col_Update_trig  
ON Flight_details  
FOR UPDATE  
AS  
IF UPDATE (Fare)  
BEGIN  
    PRINT 'Ban khong the update cot Fare'  
    ROLLBACK TRAN  
END
```

2. Bây giờ, chúng ta sẽ thử cập nhật cột Fare.



Hình 17.3

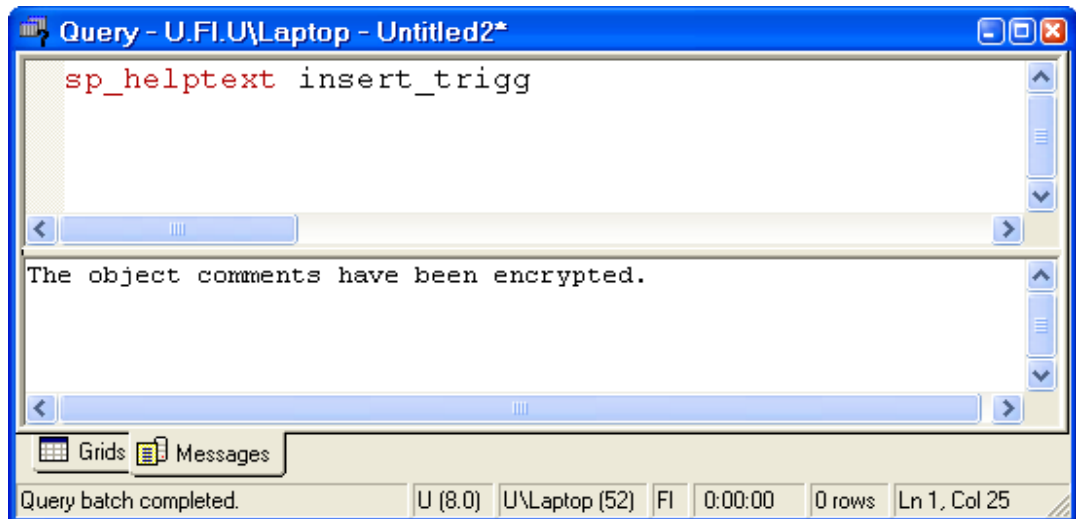
17.4 Tạo Trigger có lựa chọn Encryption

Encryption (mã hoá) là phương pháp giữ bí mật cho Trigger. Nội dung của Trigger sau khi được mã hoá sẽ không đọc được.

1. Thực hiện câu lệnh ALTER TRIGGER để sửa insert_trigg:

```
ALTER TRIGGER insert_trigg
ON Reservation
WITH ENCRYPTION
FOR INSERT
AS
IF ((Select          journey_date          From
Inserted) < getdate())
BEGIN
PRINT 'journey_date không thể nhỏ hơn ngày hiện
tai'
ROLLBACK TRAN
END
```

3. Thực hiện câu lệnh sau:



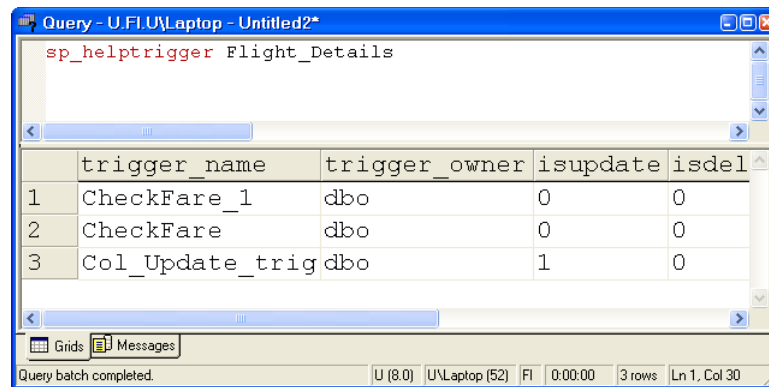
Hình 17.4

17.5 Hiện thị danh sách các trigger trong Database

Sử dụng thủ tục hệ thống sp_helptrigger để hiện thị danh sách các trigger trong cơ sở dữ liệu hiện tại.

```
sp_helptrigger Flight_Details
```

Kết quả:



	trigger_name	trigger_owner	isupdate	isdel
1	CheckFare_1	dbo	0	0
2	CheckFare	dbo	0	0
3	Col_Update_trig	dbo	1	0

Hình 17.5

17.6 Sử dụng Triggers để tạo ràng buộc tham chiếu (Enforce Referential Integrity)

Xem xét ví dụ sau: Tạo Trigger để kiểm tra dữ liệu nhập vào cột Meal Pref của bảng Passenger phải là dữ liệu đã tồn tại trong cột Meal codes của bảng Meal.

1. Thực hiện như sau trong QA:

```
CREATE TRIGGER ins_trig
ON Passenger
FOR INSERT
AS
IF (Select [Meal Pref]FROM INSERTED)
    NOT IN (Select meal_code FROM Meal)
BEGIN
    Print 'Ban khong the insert gia tri nay'
    ROLLBACK TRAN
END
```

2. Bạn hãy thử kiểm tra hoạt động của Trigger trên.

17.7 Cascade Delete sử dụng Nested trigger.

Trong Nested trigger, một trigger có thể được thực hiện lồng trong trigger khác. Chúng ta có thể lồng trigger tối đa 32 mức. Nested trigger cho phép cascade update và cascade delete.

Thực hiện câu lệnh để kích hoạt Nested trigger:

```
sp_configure 'nested trigger', 1
```

Ngược lại:

```
sp_configure 'nested trigger', 0
```

Tạo Cascade delete trigger để thực hiện công việc sau: Nếu xoá một chuyến bay trong bảng Flight, thì tất cả các thông tin liên quan trong bảng Flight_Details sẽ bị xoá.

Chương 17. TRIGGER – Phần thực hành

1. Thực hiện như sau trong QA:

```
CREATE TRIGGER Casc_del
ON Flight
FOR DELETE
AS
DELETE Flight_details FROM Flight_details, DELETED
WHERE
Flight_details.aircraft_code=DELETED.aircraft_code
```

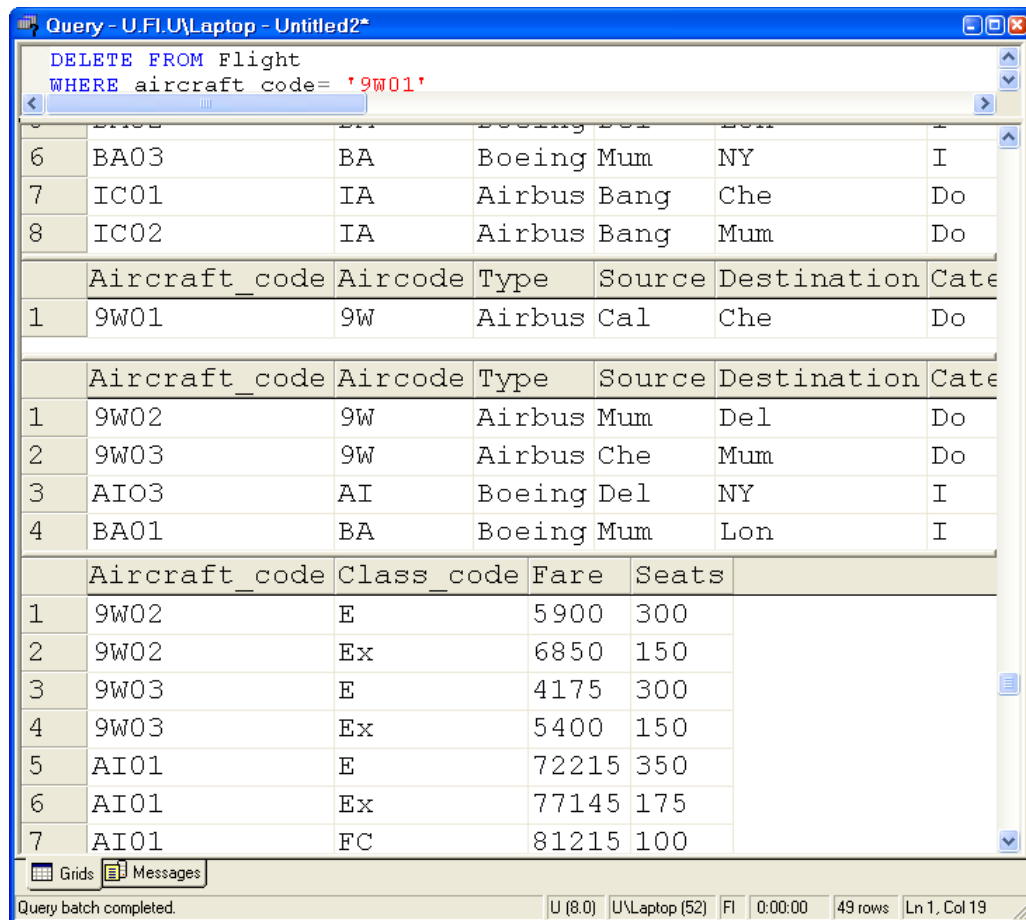
2. Tạo Delete Trigger khác trên bảng Flight. Trigger này sẽ thực hiện khi trigger Casc_del thực hiện.

```
CREATE TRIGGER del_aircraftcode
ON Flight_details
FOR DELETE
AS
SELECT * FROM Flight
SELECT * FROM Flight Details
```

3. Thực hiện câu lệnh sau:

```
DELETE FROM Flight
WHERE aircraft code= '9W01'
```

Chương 17. TRIGGER – Phần thực hành



```
DELETE FROM Flight
WHERE aircraft code= '9W01'
```

	Aircraft_code	Aircode	Type	Source	Destination	Cate
6	BA03	BA	Boeing	Mum	NY	I
7	IC01	IA	Airbus	Bang	Che	Do
8	IC02	IA	Airbus	Bang	Mum	Do
1	9W01	9W	Airbus	Cal	Che	Do
1	9W02	9W	Airbus	Mum	Del	Do
2	9W03	9W	Airbus	Che	Mum	Do
3	AI03	AI	Boeing	Del	NY	I
4	BA01	BA	Boeing	Mum	Lon	I
	Aircraft_code	Class_code	Fare	Seats		
1	9W02	E	5900	300		
2	9W02	Ex	6850	150		
3	9W03	E	4175	300		
4	9W03	Ex	5400	150		
5	AI01	E	72215	350		
6	AI01	Ex	77145	175		
7	AI01	FC	81215	100		

Query batch completed. U (8.0) U\Laptop (52) FI 0:00:00 49 rows Ln 1, Col 19

Hình 17.6

17.8 Tạo INSTEAD OF Trigger

Chúng ta có thể thực hiện INSTEAD OF trigger trên bảng, nó thay thế cho câu lệnh INSERT, UPDATE, DELETE nguyên thủy.

1. Thực hiện như sau bằng QA:

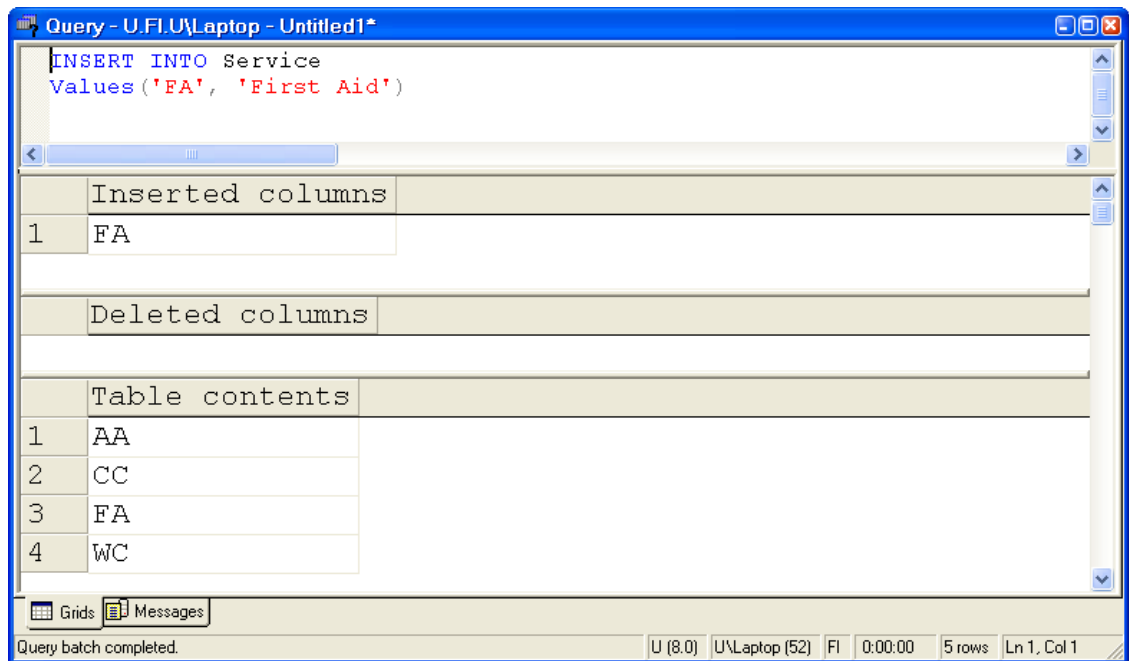
```
CREATE TRIGGER instead_trigg
ON Service
INSTEAD OF INSERT
AS
BEGIN
Select Service_code AS 'Inserted columns' From Inserted
Select Service_code AS 'Deleted columns' From Deleted
Select Service_code AS 'Table contents' From Service
END
```

2. Thực hiện câu lệnh sau:

```
INSERT INTO Service
Values('FA', 'First Aid')
```

Chương 17. TRIGGER – Phần thực hành

Kết quả:



The screenshot shows a SQL query execution window titled "Query - U.FI.U\Laptop - Untitled1*". The query executed is:

```
INSERT INTO Service
Values ('FA', 'First Aid')
```

The results are displayed in three sections:

- Inserted columns:** A table with one row containing the value 'FA'.
- Deleted columns:** An empty table.
- Table contents:** A table with four rows containing the values 'AA', 'CC', 'FA', and 'WC'.

The status bar at the bottom indicates "Query batch completed." and shows the current row and column: "Ln 1, Col 1".

Inserted columns	
1	FA

Deleted columns	
-----------------	--

Table contents	
1	AA
2	CC
3	FA
4	WC

Hình 17.7

17.9 Bài tập

Thực hiện những yêu cầu sau bằng QA:

1. Tạo INSERT trigger có tên ins_chkclass trên bảng Reservation. Trigger đảm bảo rằng dữ liệu được nhập vào trường class code tồn tại trong bảng Class_master và số ghế được đặt không quá 2.
2. Tạo Cascading UPDATE trigger có tên upd_mealcode trong bảng Meal. Khi meal code trong bảng Meal được cập nhật thì những dữ liệu liên quan trong bảng Airline_meal cũng được cập nhật tương ứng.
3. Hiển thị mã lệnh (code) của trigger ins_chkclass trên bảng Reservation.
4. Sửa trigger upd_mealcode để người sử dụng không nhìn thấy mã lệnh của nó.
5. Tạo View có tên all_day chứa thông tin chi tiết sau: day code, day name, và aircraft code trong bảng Day_master và Flight_days. Tạo DELETE trigger tên là del_day để xoá dữ liệu trong View. Kiểm tra lại sự thực hiện của trigger.

18 Chương 18. SAO LƯU & PHỤC HỒI (Backup & Restore)

18.1 Giới thiệu

Những nguyên nhân gây ra mất dữ liệu:

- Đĩa cứng hư
- Vô ý hay cố ý sửa đổi dữ liệu như xóa hay thay đổi dữ liệu.
- Trộm cắp
- Virus

Để tránh việc mất dữ liệu, chúng ta nên thường xuyên sao lưu cơ sở dữ liệu. Nếu như dữ liệu hay cơ sở dữ liệu bị hư thì ta có thể dùng bản sao lưu (backup) này để khôi phục lại cơ sở dữ liệu bị mất.

18.2 Sao lưu cơ sở dữ liệu

Sao lưu-backup một cơ sở dữ liệu (CSDL) là tạo một bản sao CSDL, ta có thể dùng bản sao để khôi phục lại CSDL nếu CSDL bị mất. Bản sao gồm tất cả những file có trong CSDL kể cả transaction log.

Transaction log (hay log file) chứa những dữ liệu thay đổi trong CSDL (Ví dụ như khi ta thực hiện các lệnh INSERT, UPDATE, DELETE). Transaction log được sử dụng trong suốt quá trình khôi phục để roll forward những transaction hoàn thành và roll back những transaction chưa hoàn thành.

Roll back là hủy bỏ giao dịch chưa hoàn thành khi hệ thống xảy ra sự cố,... (hoặc trong trường hợp sao lưu, khi đã thực hiện xong việc sao lưu mà giao dịch chưa hoàn thành) (xem chi tiết ở phần Transaction).

Roll forward là khôi phục tất cả giao dịch đã hoàn thành khi hệ thống xảy ra sự cố,... (hoặc trong trường hợp sao lưu, những giao dịch đã hoàn thành khi đã thực hiện xong việc sao lưu) (xem chi tiết ở phần Transaction).

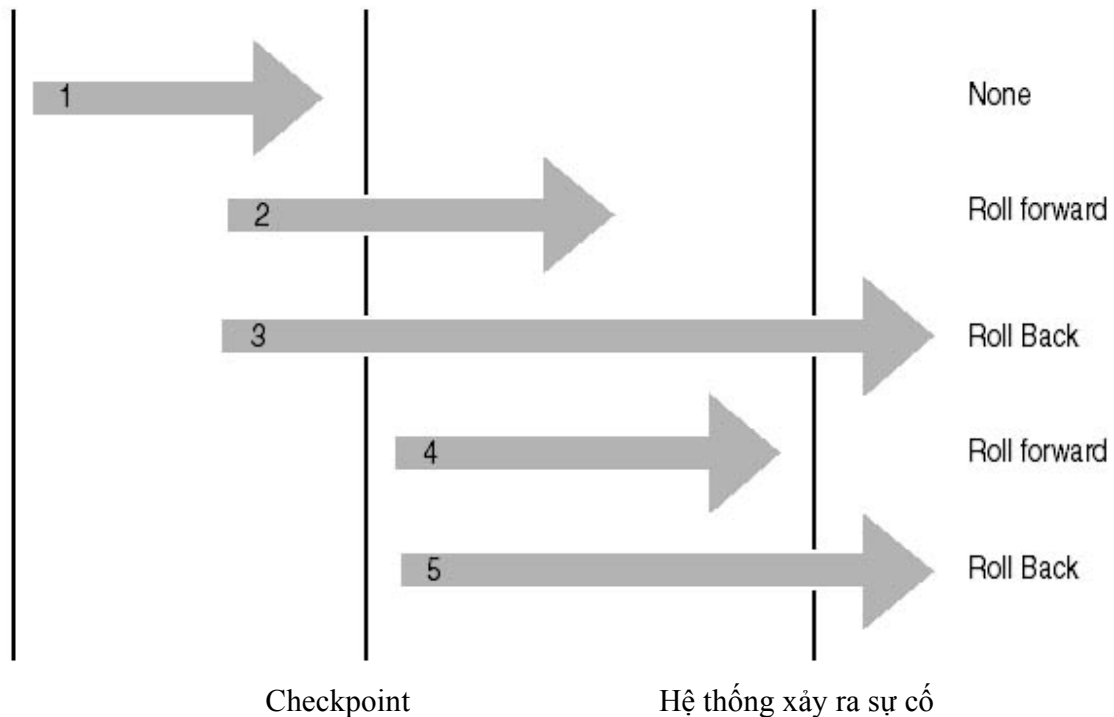
Checkpoint là thời điểm ghi lại tất cả những trang dữ liệu thay đổi lên đĩa.

Ví dụ minh họa roll back và roll forward:

- Giao dịch 1 commit trước khi checkpoint, không làm gì cả vì dữ liệu đã được thay đổi trong CSDL (ứng với số 1 trong hình).
- Giao dịch 2 và 4 commit sau khi checkpoint nhưng trước khi hệ thống xảy ra sự cố, do đó những giao dịch này được tạo lại từ log file. Điều này gọi là roll forward (ứng với số 2 và 4 trong hình).

- Giao dịch 3 và 5 chưa commit khi hệ thống xảy ra sự cố, do đó những giao dịch này không được thực hiện và trả về CSDL khi chưa xảy ra giao dịch. Điều này gọi là roll back (ứng với số 3 và 5 trong hình).

Hình minh họa quá trình khôi phục giao dịch:



Hình 18.1. Khôi phục giao dịch

Sao lưu một transaction log là chỉ sao lưu những thay đổi xảy ra trong transaction log kể từ lần sao lưu transaction log cuối cùng.

Sao lưu một CSDL ghi lại toàn bộ trạng thái của dữ liệu tại thời điểm thực hiện xong sao lưu.

Trong thời gian sao lưu, SQL Server 2000 cho phép thực hiện việc giao dịch (transaction).

18.3 Phục hồi cơ sở dữ liệu

Việc khôi phục một bản sao lưu CSDL sẽ trả về CSDL cùng trạng thái của CSDL khi ta thực hiện việc sao lưu. Giao dịch (transaction) nào không hoàn thành trong khi sao lưu (backup) CSDL được roll back để đảm bảo tính nhất quán CSDL.

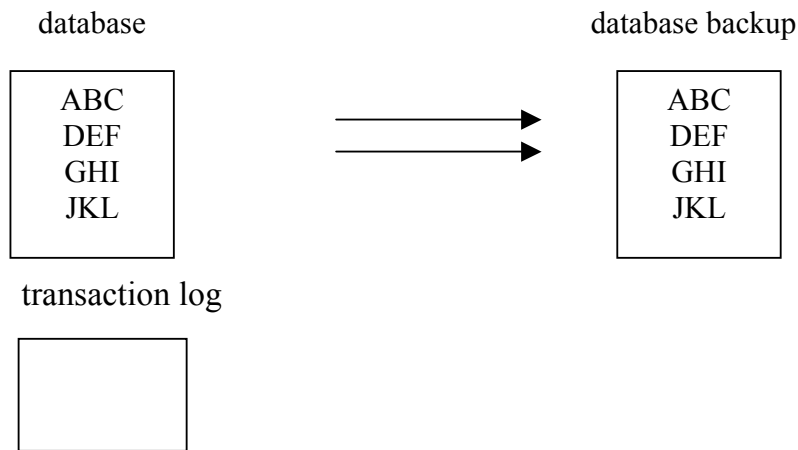
Khôi phục một bản sao lưu transaction log là áp dụng lại tất cả giao dịch (transaction) hoàn thành trong transaction log đối với CSDL. Khi áp dụng bản sao lưu transaction log, SQL Server đọc trước transaction log, roll forward tất cả các

Chương 18. SAO LƯU VÀ PHỤC HỒI (Backup & Restore)

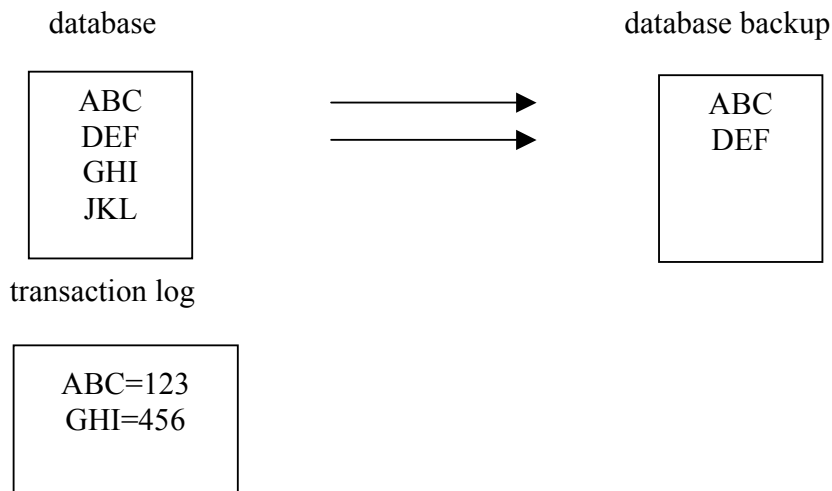
transaction . Khi đến cuối bản sao lưu transaction log, SQL Server roll back tất cả transaction mà không hoàn thành khi ta bắt đầu thực hiện sao lưu, tạo lại trạng thái chính xác của CSDL tại thời điểm bắt đầu thực hiện sao lưu.

Ví dụ minh họa sao lưu (backup) và khôi phục (restore) một CSDL có xảy ra giao dịch (transaction) khi thực hiện sao lưu:

1. Bắt đầu backup: Giả sử CSDL gồm có các dữ liệu ABC, DEF, GHI, JKL, transaction log file không có dữ liệu vì không có giao dịch nào xảy ra. Khi đang thực hiện sao lưu (backup) được một phần dữ liệu thì xảy ra giao dịch, SQL Server 2000 sẽ ưu tiên cho việc giao dịch trước, việc sao lưu (backup) tạm thời dừng lại.

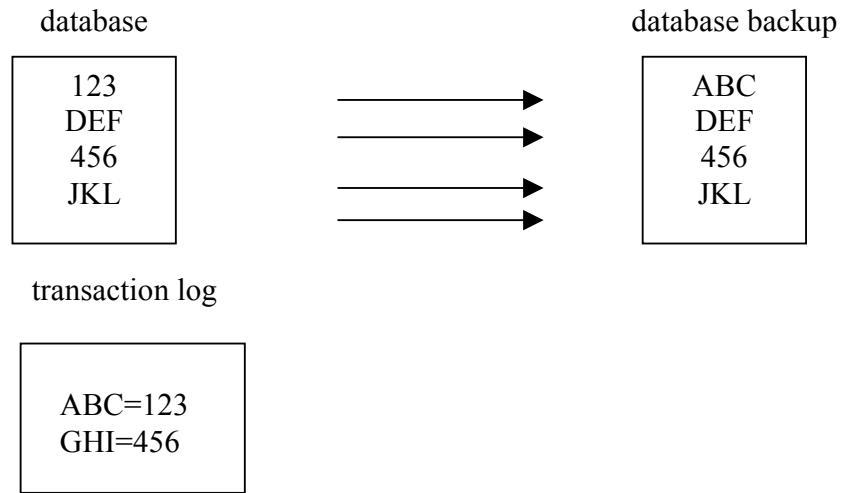


2. Xảy ra giao dịch (transaction), dữ liệu ABC được thay bằng 123, GHI được thay bằng 456.

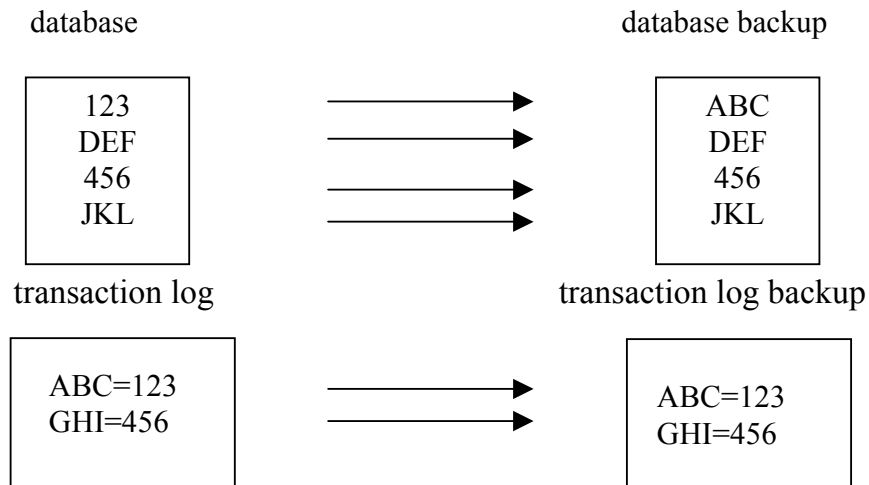


3. Khi thực hiện giao dịch (transaction) xong, SQL Server thực hiện tiếp việc sao lưu (backup), sẽ chép phần còn lại của dữ liệu nhưng dữ liệu đã thay đổi do xảy ra giao dịch.

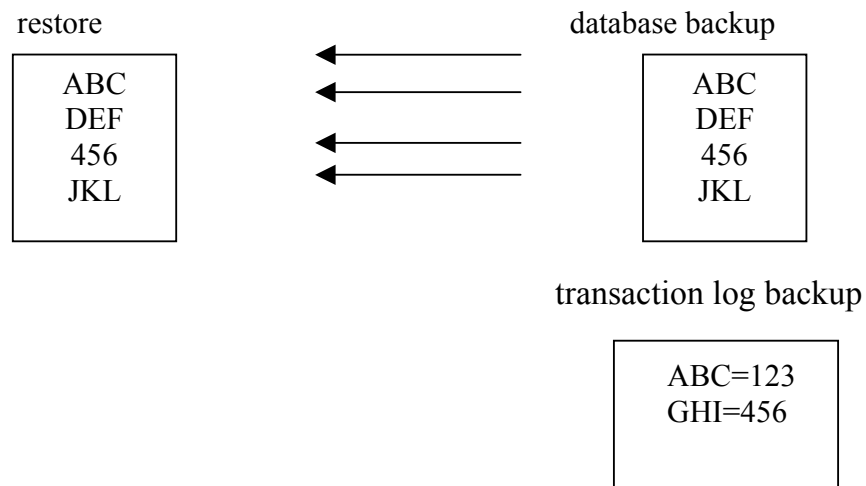
Chương 18. SAO LƯU VÀ PHỤC HỒI (Backup & Restore)



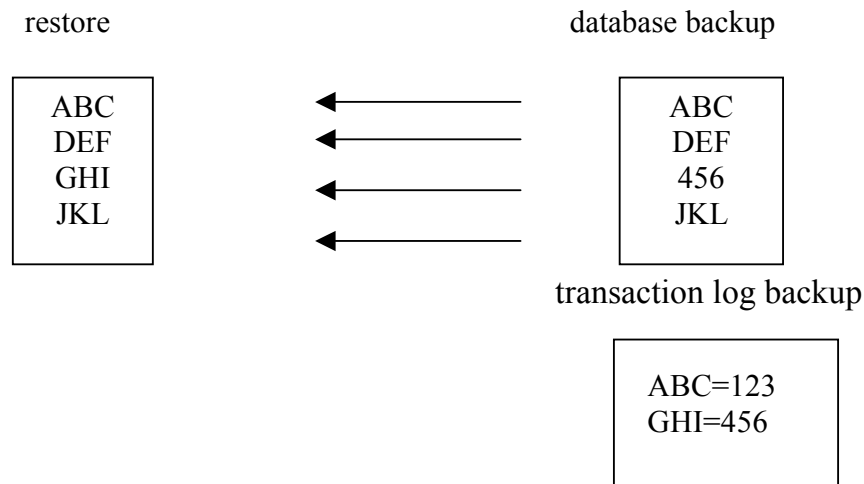
4. Khi sao lưu xong phần dữ liệu thì sẽ chép tiếp phần transaction log.



5. Khi có yêu cầu khôi phục (restore) CSDL, CSDL được khôi phục trước, chép lại toàn bộ CSDL của bản sao lưu CSDL đó.



6. Sau đó SQL Server sẽ khôi phục tiếp phần transaction log. Trước tiên sẽ roll forward nhưng khi đọc đến dữ liệu thứ ba thì nó thấy dữ liệu này đã được thay đổi rồi do đó nó sẽ roll back (trả về dữ liệu ban đầu khi chưa thực hiện giao dịch) để nhất quán dữ liệu.



18.4 Các loại Backup và Restore

18.4.1 Các loại sao lưu-Backups

Full Database Backups: Copy tất cả data files, user data và database objects như system tables, indexes, user-defined tables trong một database.

Differential Database Backups: Copy những thay đổi trong tất cả data files kể từ lần full backup gần nhất.

File or File Group Backups : Copy một data file đơn hay một file group.

Transaction Log Backups: Ghi nhận một cách thứ tự tất cả các transactions chứa trong transaction log file kể từ lần transaction log backup gần nhất. Loại backup này cho phép ta phục hồi dữ liệu trở ngược lại vào một thời điểm nào đó trong quá khứ mà vẫn đảm bảo tính nhất quán.

18.4.2 Các mô hình khôi phục- Recovery Models

Full Recovery Model: Đây là model cho phép phục hồi dữ liệu với ít rủi ro nhất. Nếu một database ở trong mode này thì tất cả các hoạt động không chỉ insert, update, delete mà kể cả insert bằng **Bulk Insert**, hay **Bcp** đều được log vào transaction log file. Khi có sự cố thì ta có thể phục hồi lại dữ liệu ngược trở lại tới

Chương 18. SAO LƯU VÀ PHỤC HỒI (Backup & Restore)

một thời điểm trong quá khứ. Khi data file bị hư nếu ta có thể backup được transaction log file thì ta có thể phục hồi database đến thời điểm transaction gần nhất được committed.

Bulk-Logged Recovery Model : Ở mode này các hoạt động mang tính hàng loạt như Bulk Insert, bcp, Create Index, WriteText, UpdateText chỉ được log minimum vào transaction log file đủ để cho biết là các hoạt động này có diễn ra mà không log toàn bộ chi tiết như trong Full Recovery Mode. Các hoạt động khác như Insert, Update, Delete vẫn được log đầy đủ để dùng cho việc phục hồi sau này.

Simple Recovery Model : Ở mode này thì Transaction Log File được truncate thường xuyên và không cần backup. Với mode này bạn chỉ có thể phục hồi tới thời điểm backup gần nhất mà không thể phục hồi tới một thời điểm trong quá khứ.

Muốn biết database của bạn đang ở mode nào bạn có thể **Right-click lên một database nào đó** trong SQL Server Enterprise Manager chọn **Properties->Options->Recovery**.

Bảng thống kê các mô hình phục hồi(Recovery Models) và kiểu sao lưu(Backup Type) tương ứng:

Model	Backup Type			
	Database	Database differential	Transaction log	File or file differential
Simple	Required	Optional	Not allowed	Not allowed
Full	Required (or file backups)	Optional	Required	Optional
Bulk-Logged	Required (or file backups)	Optional	Required	Optional

18.5 Full Database backup

Chép lại toàn bộ CSDL, lược đồ của tất cả các bảng và cấu trúc file tương ứng. Không phải tất cả những trang được chép đến backup set, chỉ những trang chứa dữ liệu thật sự mới được chép đến backup set vì thế database backup thường nhỏ hơn CSDL mà nó sao lưu. Cả những trang dữ liệu và những trang transaction log được chép đến backup set.

Khi tạo full database backup nhưng không có những bản transaction log backup riêng thì khi có yêu cầu khôi phục , ta chỉ có thể khôi phục lại được trạng thái của CSDL khi quá trình sao lưu hoàn thành. Không có cách gì khôi phục lại được CSDL mới nhất.

Ta nên dùng Full database backup nếu hệ thống có những đặc điểm sau:

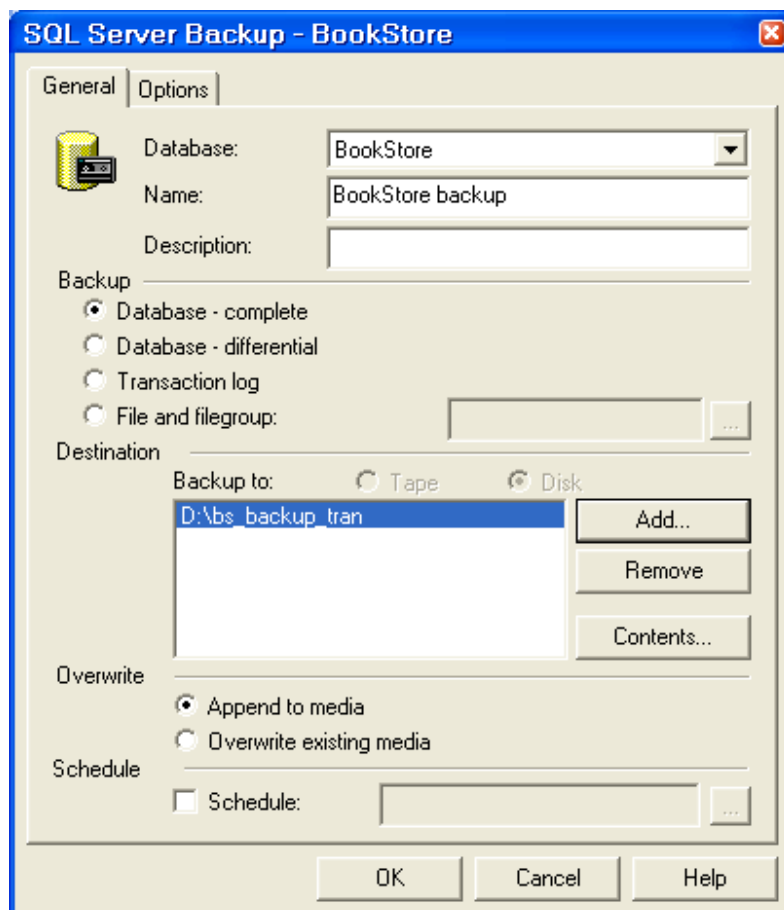
Chương 18. SAO LƯU VÀ PHỤC HỒI (Backup & Restore)

- Dữ liệu ít quan trọng và những thay đổi của CSDL có thể tạo lại bằng tay tốt hơn là dùng transaction log.
- CSDL ít thay đổi, như CSDL chỉ đọc.
- Sao lưu 1 CSDL là sao lưu toàn bộ CSDL mà không để ý đến nó có thay đổi so với lần sao lưu cuối cùng không. Điều này có nghĩa là sẽ mất nhiều vùng nhớ cho 1 bản sao và tốn nhiều thời gian để thực hiện sao lưu so với việc dùng transaction log backup và differential backup.

18.5.1 Cách tạo Full database backup bằng EM

Thực hiện các bước sau:

1. Kích vào server group, và kích vào server chứa Database muốn backup.
2. Kích **Databases**, kích phải chuột vào database, trở chuột vào **All Tasks**, sau đó kích **Backup Database**.



Hình 18.2

Chương 18. SAO LƯU VÀ PHỤC HỒI (Backup & Restore)

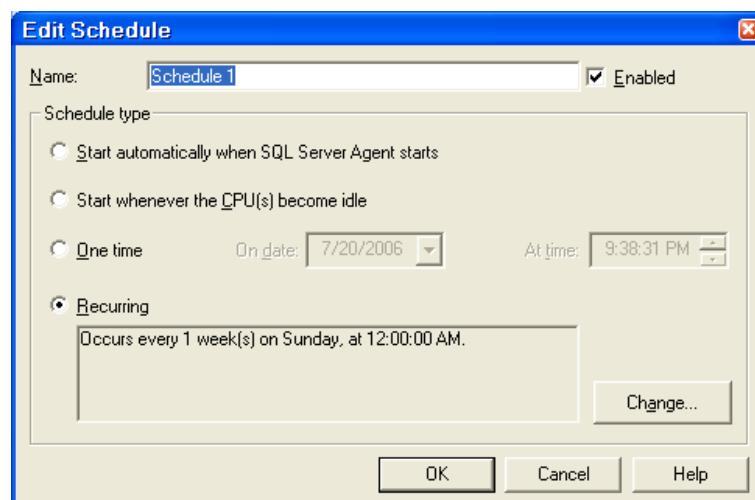
3. Trong **Name** box, nhập tên của backup set. Trong **Description**, có thể soạn chú thích cho backup set này.
4. Dưới mục Backup, kích **Database - complete**.
5. Dưới mục **Destination**, kích **Tape** hoặc **Disk (tùy thuộc bạn muốn backup vào loại thiết bị nào)**, sau đó chỉ ra đường dẫn chứa tệp tin backup.

Nếu không có xuất hiện nơi để chọn đường dẫn _backup destinations, kích Add để thêm đường dẫn mới.



Hình 18.3

6. Dưới Overwrite, thực hiện như sau:
 - Kích **Append to media** để thêm một tệp backup mới
 - Kích **Overwrite existing media** để ghi đè lên tệp đang tồn tại.
7. [Chức năng không bắt buộc] Chọn **Schedule** check box để xếp lịch cho việc backup (backup operation). Ví dụ bạn muốn tự động backup hàng tuần vào 12h00 ngày chủ nhật, hoặc... thì bạn có thể sử dụng chức năng này.

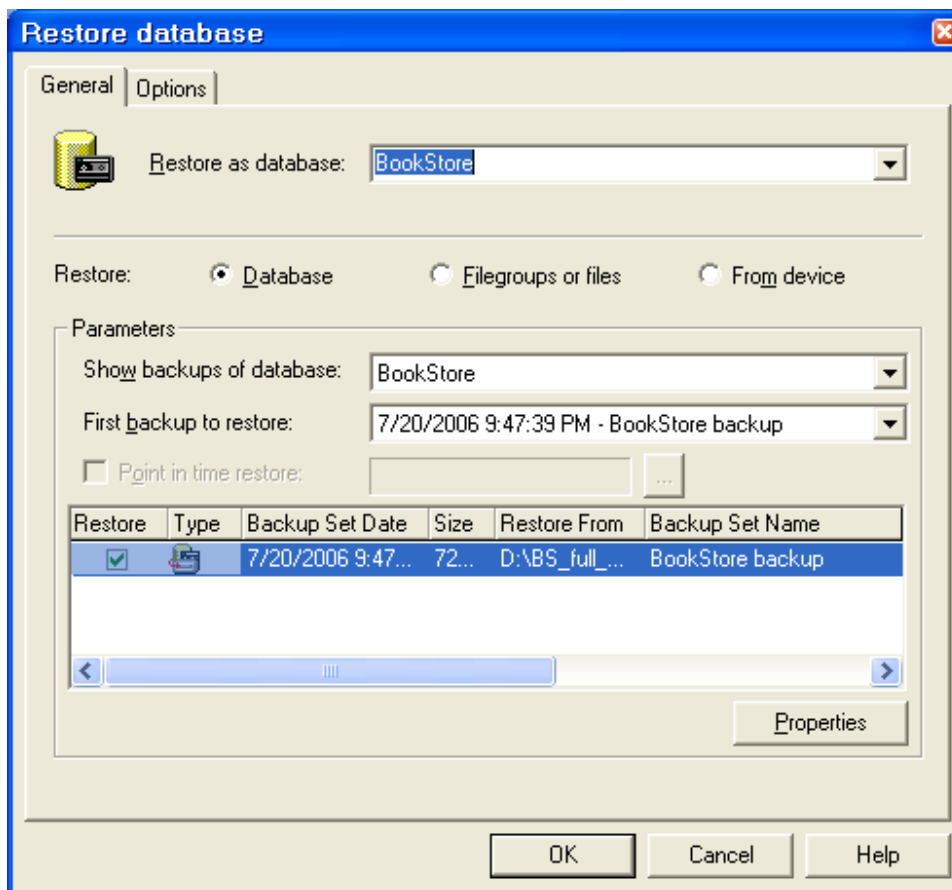


Hình 18.4

18.5.2 Khôi phục Full database backup bằng EM

Thực hiện các bước sau:

1. Kích vào server group, và kích vào server chứa Database muốn backup.
2. Kích **Databases**, kích phải chuột vào database, trở chuột vào **All Tasks**, sau đó kích **Restore Database**.



Hình 18.5

3. Trong **Restore as database** box, soạn thảo hoặc chọn tên database nếu muốn thay đổi tên Database mặc định không.
4. Kích **Database**.
5. Trong danh sách **First backup to restore**, kích vào bản sao lưu muốn được phục hồi(vì có thể có nhiều bản đã được Backup).
6. Trong danh sách **Restore**, kích vào database muốn được phục hồi.

Mô tả quá trình thực hiện:

Khôi phục lại database backup là trả về trạng thái của CSDL khi lệnh backup được thực thi. SQL Server tạo lại CSDL theo các bước sau:

Chương 18. SAO LƯU VÀ PHỤC HỒI (Backup & Restore)

- Chép tất cả dữ liệu trong bản sao vào CSDL khôi phục.
- Bất kỳ giao dịch nào không hoàn thành trong database backup thì được roll back để bảo đảm tính nhất quán dữ liệu.

Quá trình này bảo đảm CSDL sau khi khôi phục là một bản sao của CSDL khi thực hiện sao lưu, trừ những giao dịch không hoàn thành được roll back. Điều này đảm bảo tính toàn vẹn dữ liệu.

Ngoài ra, để tránh việc cố tình viết đè lên CSDL đã tồn tại, quá trình khôi phục thực hiện kiểm tra an toàn một cách tự động. Quá trình khôi phục không thực hiện nếu:

- Tên CSDL khôi phục đã tồn tại trên server và tên CSDL cần khôi phục không tương ứng với tên CSDL ghi trong backup set.
- Tên CSDL khôi phục đã tồn tại trên server nhưng dữ liệu bên trong không giống với dữ liệu bản sao database backup. Ví dụ: CSDL khôi phục có cùng tên với CSDL đã có trong SQL Server nhưng dữ liệu thì khác ví dụ như có những bảng dữ liệu khác.
- Một hoặc nhiều file yêu cầu tạo tự động bằng thao tác khôi phục (không để ý đến CSDL đó tồn tại hay chưa) nhưng những file này có cùng tên với CSDL đã tồn tại rồi.

Tuy nhiên việc kiểm tra an toàn có thể không có tác dụng nếu có mục đích viết đè.

18.6 Transaction log backup

18.6.1 Giới thiệu

Chỉ ghi lại những thay đổi trong transaction log. Transaction log backup chỉ chép lại log file. Nếu chỉ có bản sao log file thì không thể khôi phục lại được CSDL. Nó được sử dụng sau khi CSDL đã được khôi phục lại.

Sao lưu transaction log định kỳ để tạo ra 1 chuỗi transaction log backup cho phép user linh động lựa chọn để khôi phục CSDL. Tạo transaction log backup làm cho CSDL có thể khôi phục đến thời điểm xảy ra sự cố.

Khi tạo transaction log backup, điểm bắt đầu backup là:

- Điểm kết thúc của transaction log backup trước đó (nếu có một transaction log backup tạo ra trước đó).
- Transaction log backup như là một phần cuối của database backup hoặc differential backup mới nhất nếu không có transaction log backup nào được tạo ra trước đó (database backup hoặc differential backup chứa một bản sao vùng tích cực của transaction log).

Ta nên dùng transaction log backup nếu hệ thống có những đặc điểm sau:

- Tài nguyên để thực hiện database backup giới hạn như thiếu vùng lưu trữ hoặc thời gian thực hiện backup. Ví dụ: CSDL 10 terabyte đòi hỏi nhiều thời gian và vùng lưu trữ để backup.
- Bất kỳ việc mất những thay đổi sau lần database backup cuối cùng là không thể chấp nhận được. Ví dụ : hệ thống CSDL kinh doanh tài chính, nó không thể chấp nhận mất bất kỳ giao dịch nào.
- Mong muốn trả về CSDL tại thời điểm xảy ra sự cố. Ví dụ muốn khôi phục lại CSDL trước khi xảy ra sự cố 10 phút.
- CSDL thay đổi thường xuyên.

Vì transaction log backup thường sử dụng tài nguyên ít hơn nên chúng được backup thường xuyên hơn. Điều này giảm khả năng mất dữ liệu hoàn toàn.

Ít gặp trường hợp transaction log backup lớn hơn database backup. Ví dụ CSDL có tỉ lệ giao dịch cao và những giao dịch ảnh hưởng đến phần lớn CSDL gây ra transaction log tăng nhanh hoặc ít sao lưu transaction log. Trong trường hợp này tạo transaction log backup thường xuyên hơn.

Khôi phục CSDL và áp dụng transaction log backup:

- Sao lưu transaction log hiện hành nếu sự cố xảy ra (trừ khi đĩa chứa file transaction log bị hư).
- Khôi phục database backup mới nhất.
- Áp dụng tất cả các transaction log backup được tạo ra sau khi thực hiện full database backup.
- Áp dụng transaction log backup cuối cùng được tạo ra ở bước 1 để khôi phục lại CSDL đến thời điểm xảy ra sự cố.

Vì thế, mặc dù sử dụng transaction log backup tăng khả năng khôi phục, nhưng tạo và áp dụng chúng cũng phức tạp hơn dùng Full Database backup. Khôi phục CSDL sử dụng cả full database backup và transaction log backup chỉ khi ta có chuỗi transaction log backup liên tục.

SQL Server 2000 không cho phép lưu transaction log trong cùng file lưu CSDL. Vì nếu file này hư thì ta không thể sử dụng nó để khôi phục tất cả những thay đổi kể từ lần sao lưu full database backup cuối cùng.

18.6.2 Cắt (truncate) transaction log

Khi SQL Server sao lưu xong transaction log, nó cắt phần không tích cực của transaction log. SQL Server sử dụng lại phần cắt này. Phần không tích cực là phần của transaction log không còn sử dụng nữa trong quá trình khôi phục CSDL vì tất cả

giao dịch trong phần này đã hoàn tất. Ngược lại, phân tích cục của transaction log chứa những giao dịch đang chạy và chưa hoàn thành.

Điểm kết thúc phần không tích cục của transaction log, điểm cắt, là điểm đầu tiên của những sự kiện sau:

- Checkpoint gần nhất tương ứng với điểm đầu tiên mà tại đó SQL Server sẽ roll forward những giao dịch trong quá trình khôi phục.
- Bắt đầu của giao dịch tích cục cũ nhất; 1 giao dịch chưa commit hoặc roll back. Tương ứng với điểm đầu tiên mà SQL Server roll back giao dịch trong suốt quá trình khôi phục.

18.6.3 Điều kiện transaction log backups

Transaction log không nên sao lưu:

- Nếu CSDL thiết lập **trunc. log on chkpt** (truncate log on checkpoint) là TRUE (thì không thể tạo ra log record dùng để roll forward); tạo database backup hoặc differential backup thay thế.
- Nếu bất kỳ thao tác nonlogged nào xảy ra trong CSDL kể từ khi thực hiện sao lưu full database backup lần cuối cùng; tạo full database backup hoặc differential backup thay thế.
- Cho đến khi thực hiện sao lưu full database backup vì transaction log backup chứa những thay đổi của database backup.
- Nếu transaction log bị cắt, trừ khi database backup hoặc differential backup được tạo ra sau khi cắt transaction log .
- Nếu bất kỳ file nào được thêm vào hay xóa khỏi CSDL; database backup nên tạo ra thay thế ngay lúc đó.

18.6.4 Cách tạo transaction log backup bằng EM

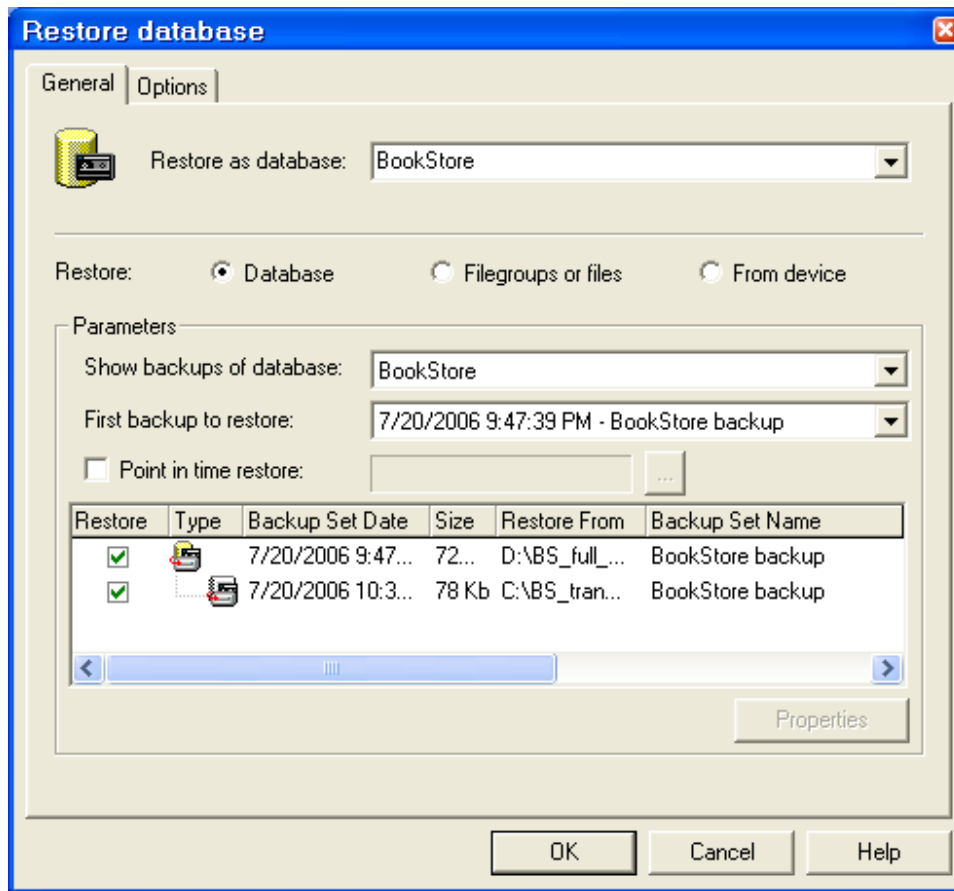
Cách làm tương tự như đối với tạo Full Database backup, tuy nhiên ở bước 4 ta phải chọn **Transaction log**.

*Lưu ý: Nếu lựa chọn **Transaction Log** không được phép thì ta phải kiểm tra lại **recovery model** để thiết đặt là **Full** hoặc **Bulk-Logged**. Bởi vì chỉ những model này mới hỗ trợ **transaction log backup**.*

18.6.5 Khôi phục transaction log backup bằng EM

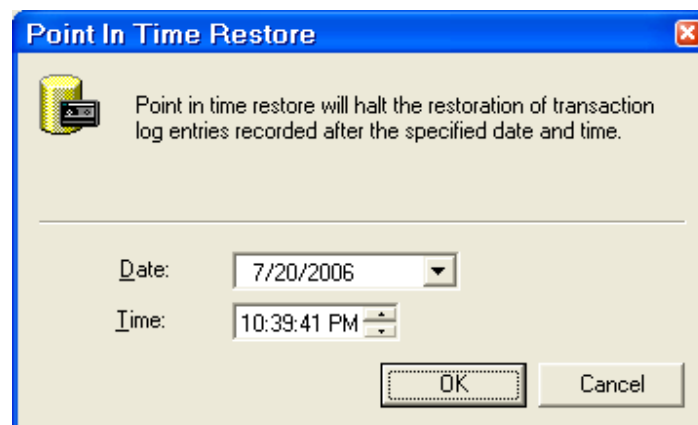
Thực hiện tương tự như cách **Khôi phục Full database backup**.

Chương 18. SAO LƯU VÀ PHỤC HỒI (Backup & Restore)



Hình 18.6

Tuy nhiên, ta có thể xác định được thời điểm nào đó muốn khôi phục dữ liệu trong quá khứ.



Hình 18.7

18.7 Differential backup

Differential backup chỉ ghi lại những trang thay đổi ngay sau khi thực hiện sao lưu full database lần cuối cùng. Do đó, sẽ nhanh hơn thực hiện full database backup rất nhiều.

Không như transaction log backup, differential backup không tạo lại CSDL chính xác tại thời điểm xảy ra sự cố, nó cũng như full database backup, chỉ tạo lại CSDL tại thời điểm backup cuối cùng. Vì thế, differential backup thường được bổ sung bằng cách tạo transaction log sau mỗi differential backup. Sử dụng kết hợp database backup, differential backup, và transaction log backup ta có thể giảm tối thiểu khả năng mất dữ liệu và thời gian khôi phục dữ liệu.

Ta nên dùng differential backup nếu hệ thống có những đặc điểm sau:

- Dữ liệu ít quan trọng và những thay đổi của CSDL có thể tạo lại bằng tay tốt hơn là dùng transaction log.
- Tài nguyên để thực hiện database backup giới hạn như thiếu vùng lưu trữ hoặc thời gian thực hiện sao lưu. Ví dụ: CSDL 10 terabyte đòi hỏi nhiều thời gian và vùng lưu trữ để thực hiện sao lưu.
- Tối thiểu hóa thời gian khôi phục và giảm việc mất những giao dịch bằng cách kết hợp differential backup với full database backup và transaction log backup.

Xem xét ví dụ sau:

24:00 (Thứ 3)	Full Database backup (1)
6:00 (Thứ 4)	Differential database backup (2)
12:00 (Thứ 4)	Differential database backup (3)
18:00 (Thứ 4)	Differential database backup (4)
24:00 (Thứ 4)	Full Database backup (5)
6:00 (Thứ 5)	Differential database backup (6)
12:00 (Thứ 5)	Differential database backup (7)

Differential backup tạo vào lúc 6:00 ngày thứ tư (2) chứa tất cả những thay đổi của database backup tạo từ lúc 24:00 ngày thứ ba(1).

Differential backup tạo vào lúc 6:00 ngày thứ năm(6) chứa tất cả những thay đổi của database backup tạo từ lúc 24:00 ngày thứ tư (5).

Nếu có khôi phục CSDL đến trạng thái vào 12:00 ngày thứ năm, ta thực hiện những bước sau:

- Khôi phục database tạo lúc 24:00 ngày thứ tư.

- Khôi phục differential backup tạo lúc 12:00 ngày thứ năm.

Bất kỳ thay đổi nào sau trưa thứ năm đều bị mất trừ khi có khôi phục transaction log backup.

Chú ý: Các tạo Differential backup và khôi phục cũng tương tự như thực hiện với Full database backup.

Sự khác nhau giữa differential backup và transaction log backup:

- **Giống nhau:** tối thiểu hóa thời gian sao lưu.
- **Khác nhau:**
 - o Differential backup: chỉ lưu lần thay đổi cuối cùng
 - o Transaction log backup: chứa tất cả những thay đổi kể từ lần sao lưu full database backup cuối cùng.

Do differential backup lưu những trang thay đổi, gồm những trang dữ liệu và cả trang transaction log thay đổi. Vì sao lưu differential backup có kích thước lớn hơn sao lưu transaction backup nên ta ít sao lưu differential backup thường xuyên so với sao lưu transaction backup. Do đó ta không thể khôi phục CSDL đến thời điểm xảy ra sự cố khi sao lưu differential backup và không thể khôi phục CSDL đến thời điểm mà ta mong muốn.

18.8 File hoặc Filegroup backup

Chỉ sao lưu những file CSDL chỉ định. File hoặc file group backup thường được sử dụng chỉ khi không có đủ thời gian để sao lưu toàn bộ CSDL.

Sử dụng file hoặc file group backup có thể tăng tốc độ khôi phục bằng cách chỉ khôi phục những file hoặc filegroup bị hư. Khi sao lưu file hoặc file group thì SQL Server không có sao lưu file transaction log do đó ta phải tạo transaction log backup sau khi sao lưu file hoặc file group.

Ví dụ: Một CSDL có 2 filegroup filegroup_a và filegroup_b nhưng chỉ có đủ thời gian để sao lưu 1 nửa filegroup, do đó:

- Sao lưu filegroup_a vào các ngày thứ hai, tư, sáu.
- Sao lưu transaction log ngay sau khi sao lưu filegroup.
- Sao lưu filegroup_b vào các ngày thứ năm, sáu, bảy.
- Sao lưu transaction log ngay sau khi sao lưu filegroup.

Khôi phục file hoặc filegroup backup:

File hoặc file group có thể được khôi phục từ database backup hoặc file hoặc file group. Ta không cần thiết backup transaction log nếu không có thay đổi từ khi sao lưu file hoặc file group.

Ví dụ: Nếu filegroup_b cần khôi phục vì 1 bảng trong filegroup bị hư, ta sẽ:

Chương 18. SAO LƯU VÀ PHỤC HỒI (Backup & Restore)

- Khôi phục filegroup_b backup tạo vào ngày thứ năm.
- Áp dụng transaction log backup của filegroup_b.

19 Chương 19. CHUYỂN ĐỔI GIỮA CÁC LOẠI CƠ SỞ DỮ LIỆU (DTS- Data Transformation Service)

Một trong những cách trao đổi dữ liệu giữa các loại cơ sở dữ liệu với nhau là sử dụng tiện ích Import hay Export dữ liệu từ định dạng này sang định dạng khác.

SQL Server 2000 cung cấp hai chức năng chính là Import dùng để nhập dữ liệu vào cơ sở dữ liệu SQL Server 2000 từ các loại cơ sở dữ liệu khác và Export dùng để xuất dữ liệu từ SQL Server 2000 ra các loại cơ sở dữ liệu khác. Tuy nhiên, tùy thuộc vào mức độ tương thích giữa cơ sở dữ liệu Import hay Export mà hệ thống cho phép ta Import hay Export dữ liệu và các đối tượng giữa hai cơ sở dữ liệu với nhau ở mức độ nào đó.

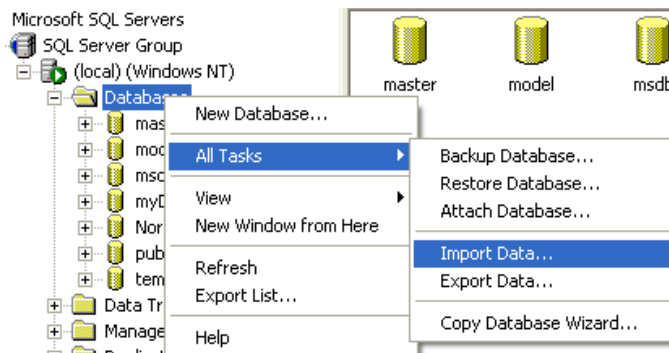
19.1 Import cơ sở dữ liệu

Việc đầu tiên ta phải xác định cơ sở dữ liệu nguồn thuộc loại cơ sở dữ liệu nào để chọn và dĩ nhiên cơ sở dữ liệu đích chính là SQL Server 2000.

19.1.1 Import cơ sở dữ liệu từ SQL Server 2000

Giả sử cần Import cơ sở dữ liệu Northwind từ SQL Server 2000

Trong cửa sổ SQL Server Enterprise Manager, kích chuột phải vào Database > All Tasks > Import Data.



Hình 19.1

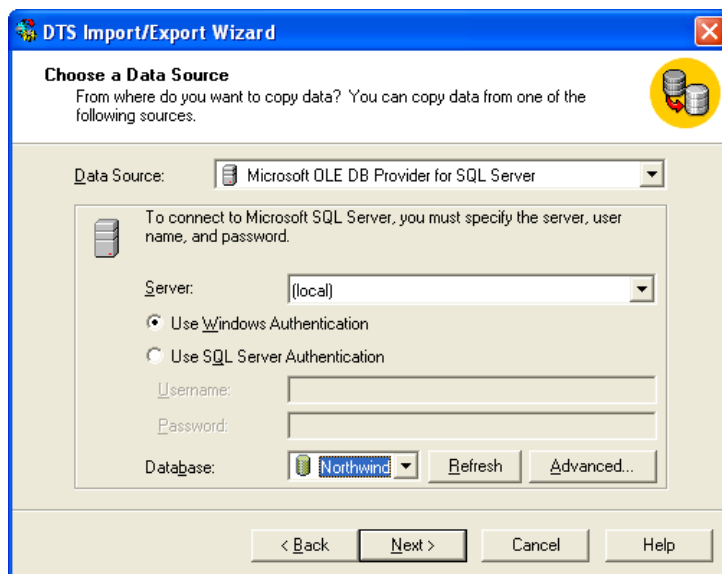
Hoặc từ menu Tools > Data Transformation Services > Import Data



Hình 19.2

Tiếp theo cần lựa chọn nguồn dữ liệu để Import

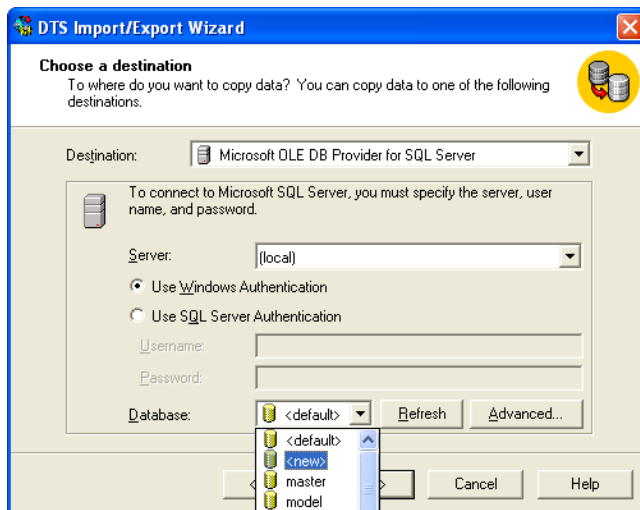
Chương 19. CHUYỂN ĐỔI GIỮA CÁC LOẠI CƠ SỞ DỮ LIỆU (DTS)



Hình 19.3

- **Data Source:** Microsoft OLD DB Provider for SQL Server
- **Server:** Nếu nguồn dữ liệu nằm ngay tại máy thì chọn là local, nếu nằm trên SQL Server 2000 khác thì cần đánh tên server hoặc địa chỉ IP vào. Ví dụ: haidv hoặc 192.168.0.1
- **Chế độ xác thực (Authentication):** Tùy thuộc vào cách xác thực của server nguồn
- **Database:** Chọn database cần Import trong trường hợp này chọn Northwind

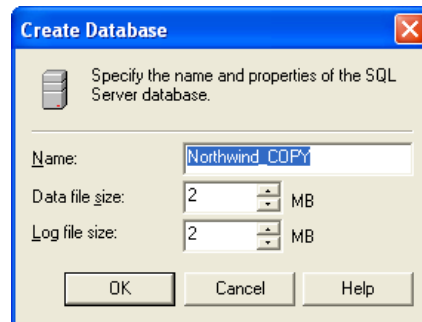
Chọn **Next** để tiếp tục chọn thông tin của cơ sở dữ liệu đích



Hình 19.4

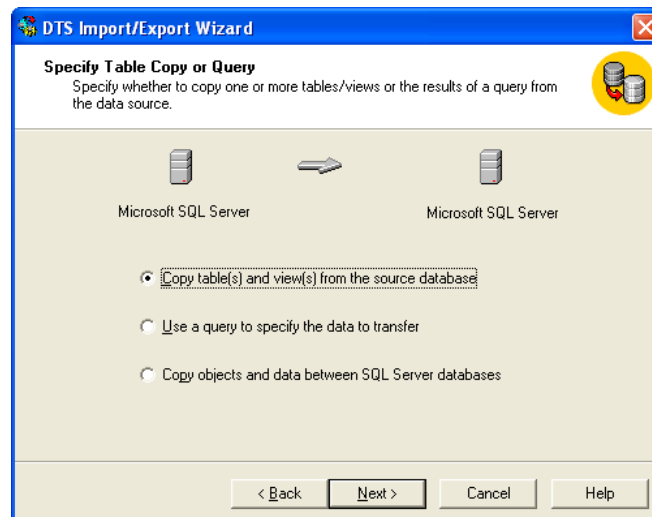
Chương 19. CHUYỂN ĐỔI GIỮA CÁC LOẠI CƠ SỞ DỮ LIỆU (DTS)

Trong mục *Database* ta chọn <new> để tạo một cơ sở dữ liệu mới. Sau đó đánh tên cơ sở dữ liệu mới, giả sử là *Northwind_COPY*.



Hình 19.5

Sau đó ấn Next



Hình 19.6

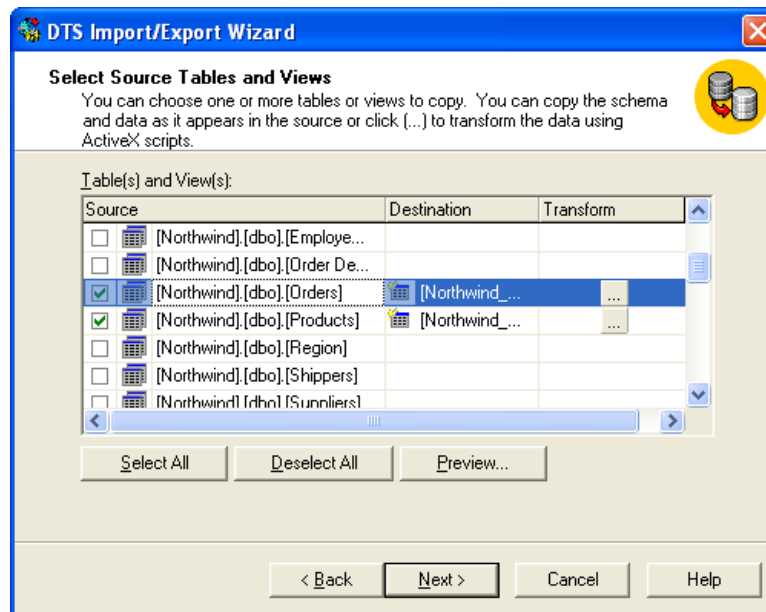
Có ba sự lựa chọn, ta sẽ tìm hiểu lần lượt từng lựa chọn một

19.1.1.1 Copy table(s) and view(s) from the source database

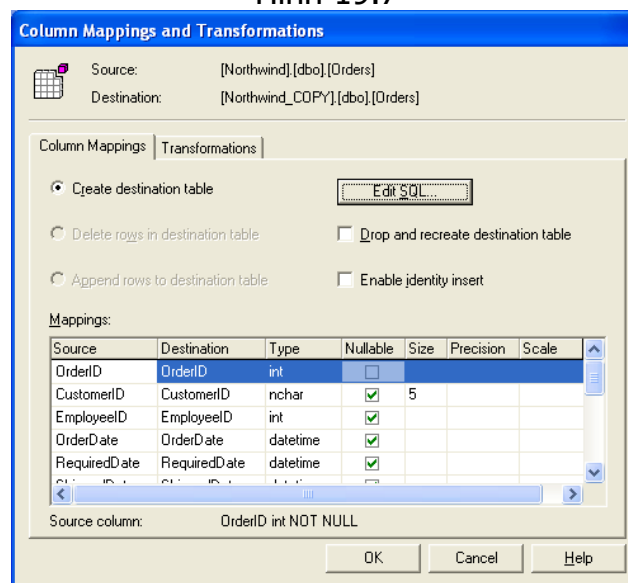
Ở lựa chọn này ta sẽ nhận được danh sách các đối tượng Table và View của cơ sở dữ liệu Northwind. Muốn lựa chọn Table hay View nào cần Import ta kích vào Checkbox của từng đối tượng. (Hình 19.7)

Nếu muốn đối tượng mà SQL Server 2000 sẽ sao chép có cấu trúc khác với các đối tượng sẽ Import, ta có thể chọn nút ... trên cột Transform. Ta có thể sửa được cấu trúc của bảng đích theo ý muốn.

Chương 19. CHUYỂN ĐỔI GIỮA CÁC LOẠI CƠ SỞ DỮ LIỆU (DTS)



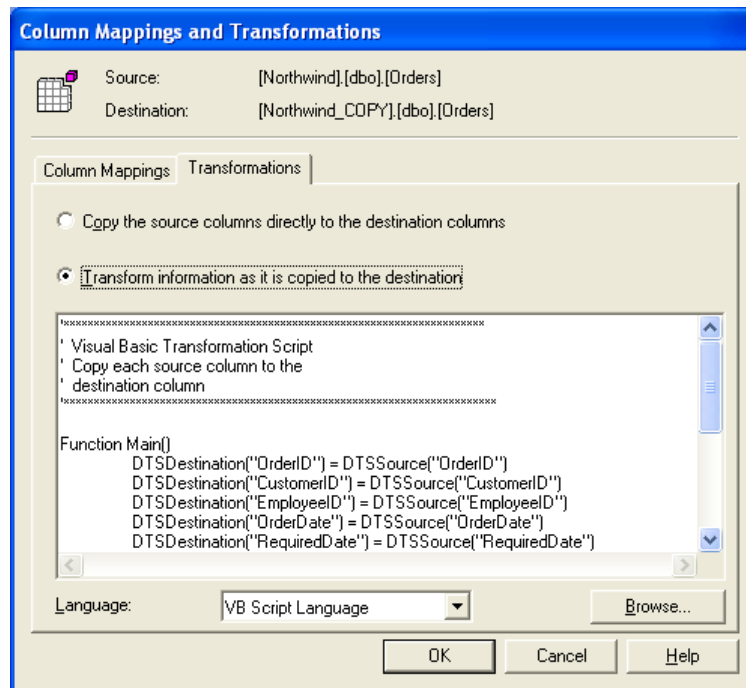
Hình 19.7



Hình 19.8

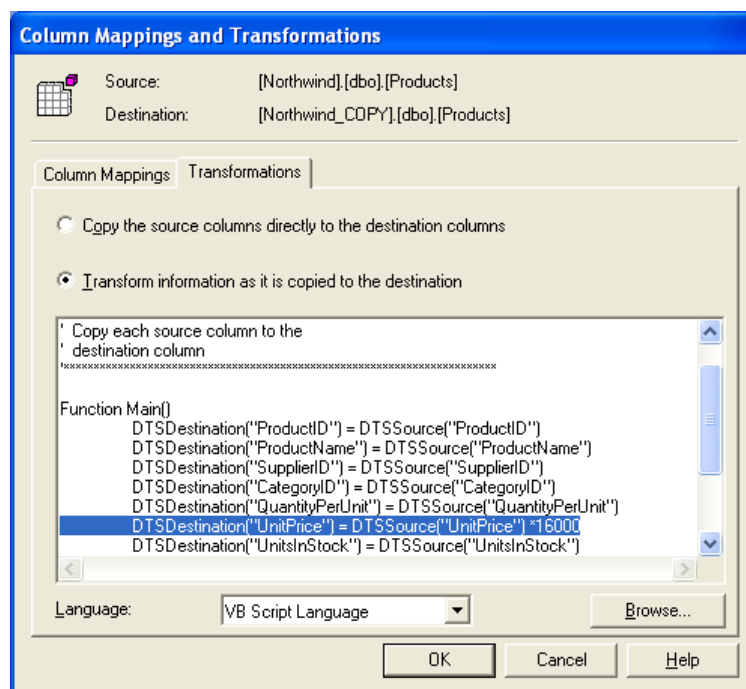
Nếu muốn không Import một số cột dữ liệu nào đó trong bảng hoặc khi Import muốn thay đổi giá trị theo ý muốn thì kích vào mục Transformations

Chương 19. CHUYỂN ĐỔI GIỮA CÁC LOẠI CƠ SỞ DỮ LIỆU (DTS)



Hình 19.9

Và kích vào mục *Transform information as it is copied to the destination*



Hình 19.10

Ta có thể viết các lệnh theo ý muốn ở đây. Ví dụ ta muốn trường *UnitPrice* giá tính bằng Đồng tức ta cần nhân trường này với tỉ giá USD/VND (16000)

Chương 19. CHUYỂN ĐỔI GIỮA CÁC LOẠI CƠ SỞ DỮ LIỆU (DTS)

Function Main()

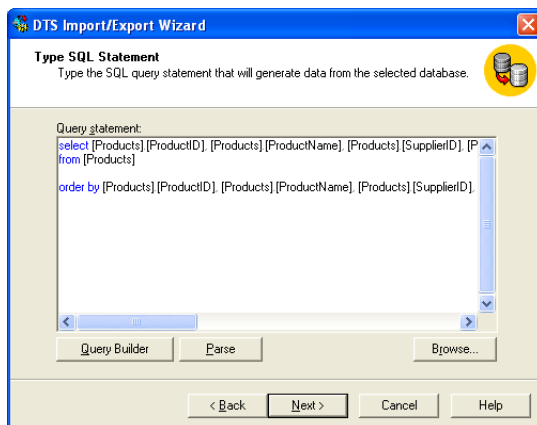
```
DTSDestination("ProductID") = DTSSource("ProductID")
DTSDestination("ProductName") = DTSSource("ProductName")
```

```
DTSDestination("SupplierID") = DTSSource("SupplierID")
DTSDestination("CategoryID") = DTSSource("CategoryID")
DTSDestination("QuantityPerUnit")=DTSSource("QuantityPerUnit")
DTSDestination("UnitPrice") = DTSSource("UnitPrice") *16000
DTSDestination("UnitsInStock") = DTSSource("UnitsInStock")
DTSDestination("UnitsOnOrder") = DTSSource("UnitsOnOrder")
DTSDestination("ReorderLevel") = DTSSource("ReorderLevel")
DTSDestination("Discontinued") = DTSSource("Discontinued")
Main = DTSTransformStat_OK
```

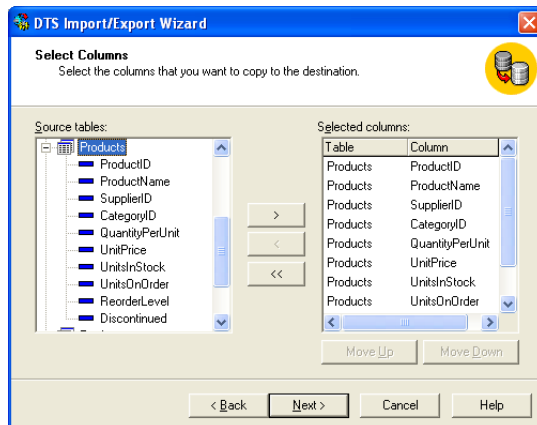
End Function

19.1.1.2 Use a query to specify the data to transfer

Với lựa chọn này cửa sổ kế tiếp sẽ như hình dưới và yêu cầu ta cung cấp phát biểu SQL .



Hình 19.11



Hình 19.12

Có thể gõ trực tiếp câu lệnh SQL:

```
select [Products].[ProductID], [Products].[ProductName],
[Products].[UnitPrice]
from [Products]
```

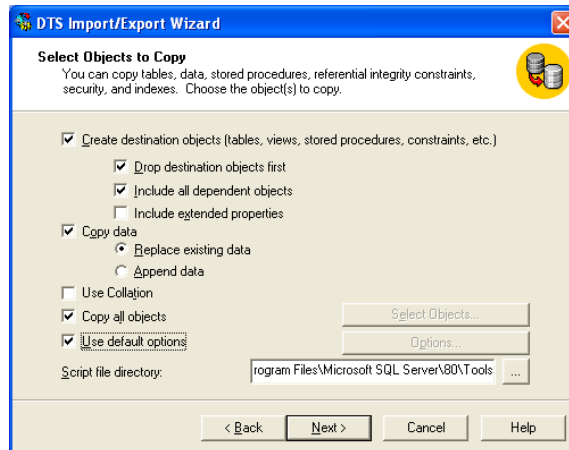
Hoặc sử dụng công cụ Query Builder (hình trên). Sau khi thực hiện xong có thể nhấn nút Parse để kiểm tra cú pháp của phát biểu SQL. Ngoài ra ta cũng có thể nạp phát biểu SQL từ bên ngoài (tập tin dạng text) bằng cách nhấn nút Browse

19.1.1.3 Copy objects and data between SQL Server databases

Với lựa chọn này cửa sổ kế tiếp sẽ như hình dưới:

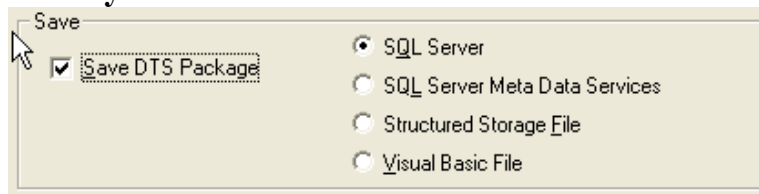
Yêu cầu ta chọn đối tượng muốn Import (mặc định là tất cả các đối tượng của cơ sở dữ liệu cần Import đang có).

Chương 19. CHUYỂN ĐỔI GIỮA CÁC LOẠI CƠ SỞ DỮ LIỆU (DTS)



Hình 19.13

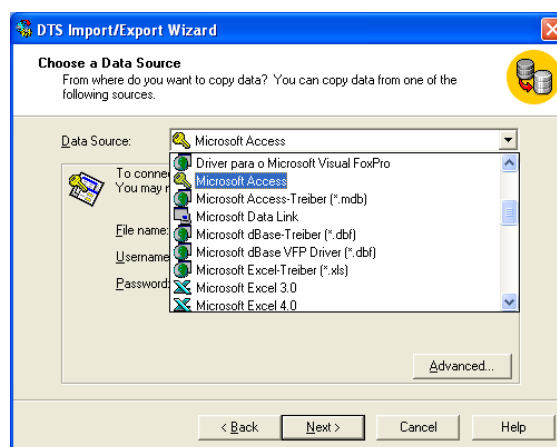
19.1.1.4 Chú ý



- Ta có thể lưu quy trình Import cơ sở dữ liệu ra tập tin dts hoặc .bas theo định dạng Visual Basic File. Khi ta chọn tùy chọn lưu DTS Package(Data) ra định dạng Visual Basic File thì SQL Server 2000 không cho phép ta chọn tùy chọn Schedule
- Ta có thể sử dụng lại Package của DTS nếu chọn và lưu lại DTS Package này

19.1.2 Import từ cơ sở dữ liệu Access

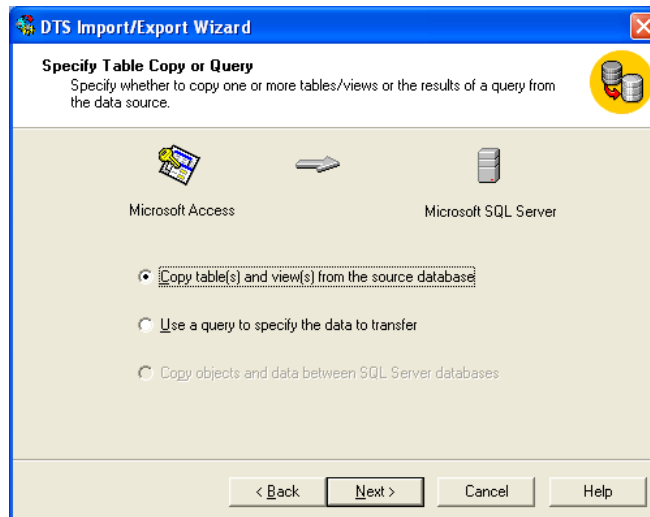
Cũng như phần Import từ SQL Server 2000, trong cửa sổ lựa chọn cơ sở dữ liệu nguồn để Import ta chọn mục Microsoft Access (hình 19.14)



Hình 19.14

Chương 19. CHUYỂN ĐỔI GIỮA CÁC LOẠI CƠ SỞ DỮ LIỆU (DTS)

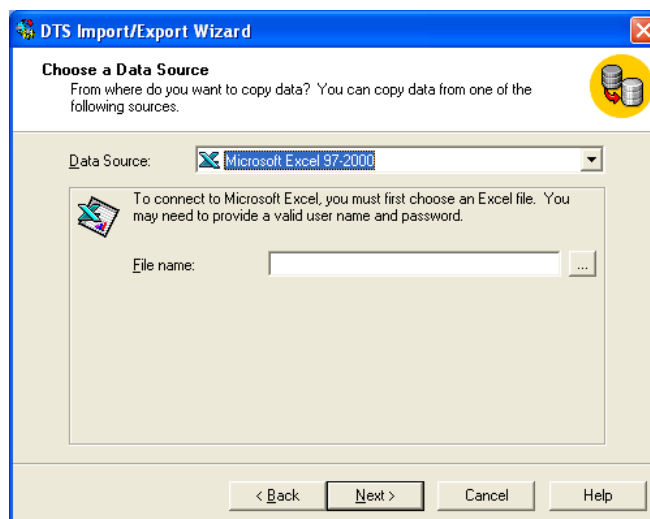
Sau đó kích vào nút ... để chọn file Access. Nếu file Access đó có mật khẩu thì ta cần nhập Username và Password vào (nếu không thì để trống). Sau đó chọn Next để chọn hai phương thức như hình dưới. Quá trình thực hiện cũng tương tự như cách thực hiện khi Import từ SQL Server 2000.



Hình 19.15

19.1.3 Import từ tập tin Excel

Ta lựa chọn cơ sở dữ liệu nguồn là Microsoft Excel 97-2000, sau đó chọn tên file excel trong mục filename. Các bước tiếp được tiến hành tương tự như với các loại cơ sở dữ liệu khác. Chú ý rằng trong Excel các Sheet được coi là các Table của cơ sở dữ liệu.

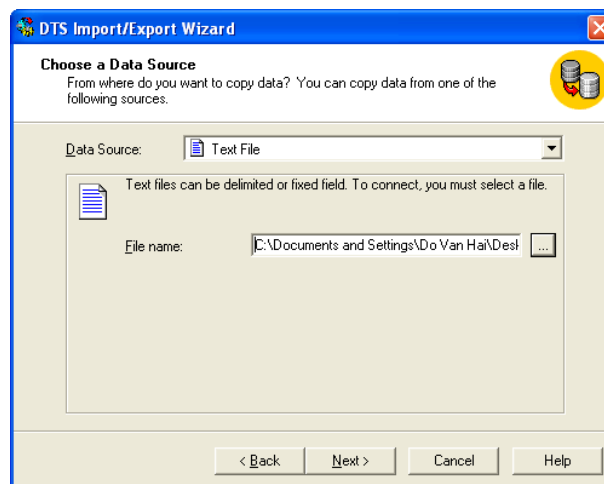


Hình 19.16

19.1.4 Import từ tập tin dạng Text

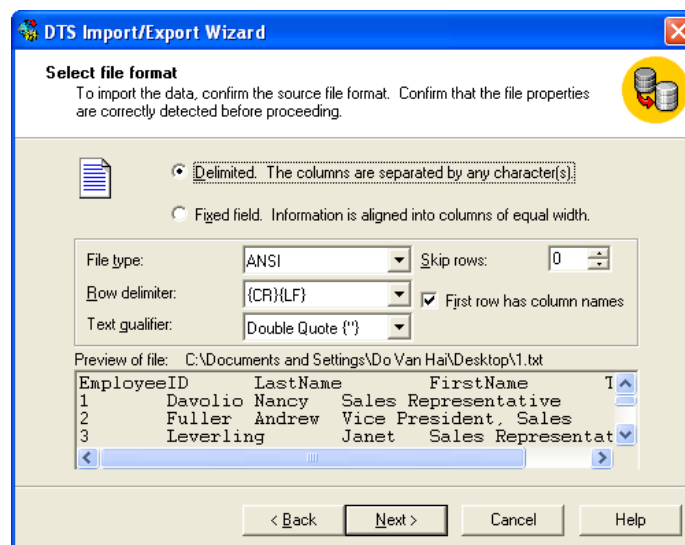
Để Import dữ liệu từ tập tin dạng Text phải đảm bảo rằng dữ liệu trong tập tin đó có một thứ tự nhất định. Giả sử rằng chúng ta có tập tin Text có dạng như sau:

ID	LastName	FirstName	Title
1	Davolio	Nancy	Sales Representative
2	Fuller	Andrew	Vice President, Sales
3	Leverling	Janet	Sales Representative
4	Peacock	Margaret	Sales Representative
5	Buchanan	Steven	Sales Manager
6	Suyama	Michael	Sales Representative
7	King	Robert	Sales Representative
8	Callahan	Laura	Inside Sales Coordinator
9	Dodsworth	Anne	Sales Representative



Hình 19.17

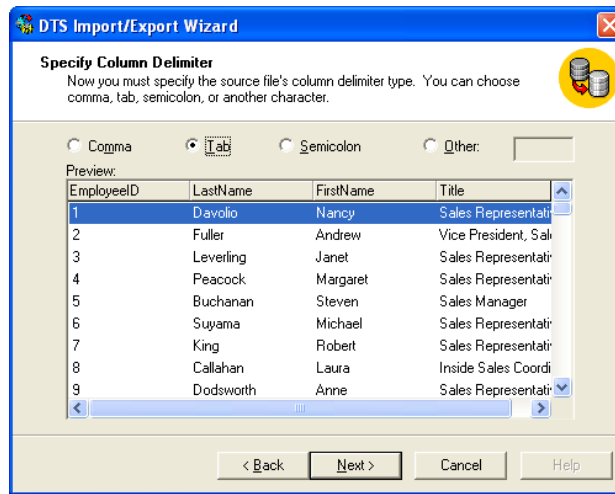
Trong phần Data Source ta chọn Text File, sau đó chọn đường dẫn đến file text trong mục File name.



Hình 19.18

Chương 19. CHUYỂN ĐỔI GIỮA CÁC LOẠI CƠ SỞ DỮ LIỆU (DTS)

Kích vào mục *First row has column names* nếu dòng đầu tiên là tên của trường. Sau đó chọn Next.



Hình 19.19

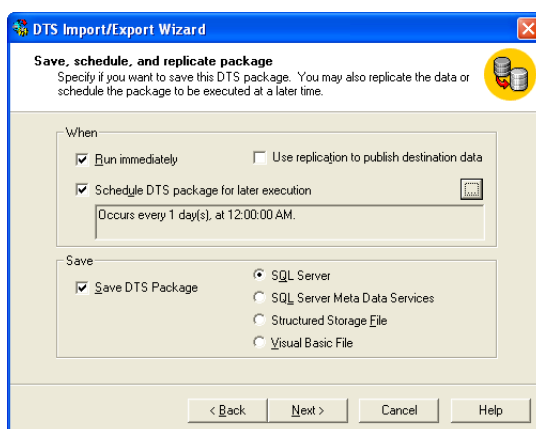
Phần chọn Specify Column Delimiter ta chọn ký tự nào dùng để phân tách các cột dữ liệu trong trường hợp này là ký tự TAB. Khi chọn các ký tự phân tách cột khác ta có thể thấy dữ liệu được phân tách ở dưới.

19.2 Export cơ sở dữ liệu

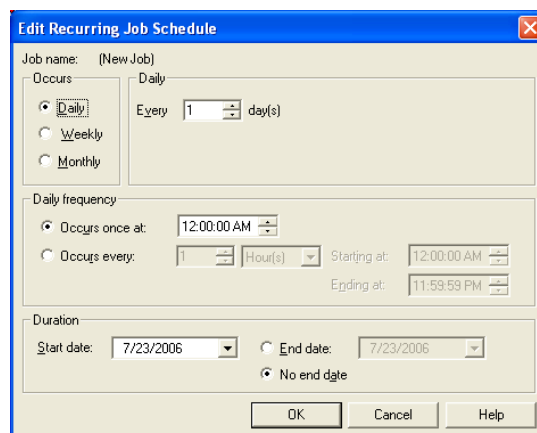
Export cơ sở dữ liệu cũng tương tự như Import cơ sở dữ liệu. Đây là quá trình ngược lại với quá trình Import các thao tác với Export cũng tương tự như với Import.

19.3 Xây dựng lịch trình Import và Export cơ sở dữ liệu

Quá trình Import và Export cơ sở dữ liệu có thể cài đặt một lịch trình tự động theo một thời gian nhất định. Theo mặc định là tất cả các ngày vào lúc 12 giờ đêm. Ta có thể thay đổi thời gian biểu này thông qua cửa sổ sau:



Hình 19.20



Hình 19.21

19.4 Những điều cần giải quyết sau khi Import hay Export

Khi Import dữ liệu từ cơ sở dữ liệu khác vào cơ sở dữ liệu SQL Server 2000, các kiểu dữ liệu thường không giống như ý ta muốn. Chẳng hạn, trong cơ sở dữ liệu Access kiểu dữ liệu True/False nhưng đối với SQL Server 2000 thì dữ liệu tương ứng là bit(0,1). Tương tự như vậy, trong cơ sở dữ liệu Access kiểu dữ liệu là Text nhưng đối với SQL Server 2000 thì chia ra thành nhiều loại như char, nchar, nvarchar.

Như vậy, sau khi Import dữ liệu từ cơ sở dữ liệu Access vào SQL Server 2000 ta phải khai báo lại dữ liệu cho từng cột dữ liệu cho phù hợp mặc dù hầu hết dữ liệu đều được Import thành công.

Đối với trường hợp Export từ cơ sở dữ liệu SQL Server 2000 ra cơ sở dữ liệu khác cũng tương tự, ta luôn khai báo lại kiểu dữ liệu cho phù hợp với loại dữ liệu mà cơ sở dữ liệu được Import hỗ trợ.

20 Chương 20. KIẾN TRÚC NHÂN BẢN (REPLICATION)

Nhân bản là một kỹ thuật quan trọng và hữu hiệu trong việc phân bố cơ sở dữ liệu (CSDL) và thực thi các Stored procedure. Kỹ thuật nhân bản trong SQL Server cho phép bạn tạo ra những bản sao dữ liệu giống hệt nhau, di chuyển các bản sao này đến những vùng khác nhau và đồng bộ hoá dữ liệu một cách tự động để tất cả các bản sao có cùng giá trị dữ liệu. Nhân bản có thể thực thi giữa những CSDL trên cùng một server hay những server khác nhau được kết nối bởi mạng LANs, WANs hay Internet.

SQL Server đã đưa ra nhiều cơ chế nhân bản để đáp ứng các yêu cầu khác nhau của ứng dụng. Mỗi loại cung cấp các khả năng và thuộc tính khác nhau nhằm đạt đến mục tiêu của tính độc lập “Site” và sự nhất quán các giao dịch.

20.1 Mục tiêu chính của nhân bản

SQL Server đã đưa ra nhiều cơ chế nhân bản để đáp ứng các yêu cầu khác nhau của ứng dụng. Mỗi loại cung cấp các khả năng và thuộc tính khác nhau nhằm đạt đến mục tiêu của tính độc lập “Site” và sự nhất quán dữ liệu.

20.1.1 Nhất quán dữ liệu (Data consistency)

Có 2 cách để đạt được tính nhất quán dữ liệu:

- Nhất quán giao dịch (Transactional Consistency)
- Hội tụ dữ liệu (Data Convergence)

20.1.1.1 Nhất quán giao dịch

- Bảo đảm tất cả dữ liệu giống nhau tại mọi site ở bất kỳ thời điểm.
- Tất cả giao dịch thực hiện tại một site duy nhất.

Có 2 loại :

Nhất quán lập tức (Immediate Transactional Consistency hay Tight Consistency):

Ở kiểu này, tất cả các site được bảo đảm là luôn thấy cùng giá trị dữ liệu tại cùng một thời điểm. Cách duy nhất để đạt được nhất quán giao dịch (transactional consistency) trong môi trường cập nhật phân tán (distributed update environment) là sử dụng 2-phase commit protocol giữa tất cả site tham gia (participating site). Mỗi site phải commit đồng thời mọi thay đổi hoặc không site nào commit những thay đổi. Giải pháp này rõ ràng không khả thi khi số lượng site quá lớn.

Nhất quán ngầm (Latent Transactional Consistency hay Loose Consistency)

:

Chương 20. KIẾN TRÚC NHÂN BẢN (REPLICATION)

Có một sự nhất quán ngầm giữa các site tham gia do có một sự trì hoãn trong việc phản ánh các giá trị dữ liệu đến các site tham gia và vào lúc này các site không bảo đảm có cùng giá trị dữ liệu. Việc sửa đổi các giá trị dữ liệu có thể bị trì hoãn đủ lâu để tất cả các site cùng cập nhật, sau đó tất cả các site sẽ có cùng giá trị dữ liệu. Ngoài ra các giá trị dữ liệu này cũng phải giống với những giá trị đạt được khi thực hiện các công việc tại một site. Sự khác nhau duy nhất giữa nhất quán giao dịch lập tức và nhất quán giao dịch ngầm là dữ liệu có nhất quán tại cùng một lúc hay không.

20.1.1.2 Hội tụ dữ liệu

Với sự hội tụ dữ liệu, tất cả các site có thể quy về cùng một giá trị dữ liệu nhưng không nhất thiết là giá trị dữ liệu này bị gây ra bởi những tác vụ được làm trên một site duy nhất. User có thể tự do thao tác trên các site theo các cách khác nhau. Khi các nút (node) đồng bộ, tất cả các site sẽ hội tụ về cùng một giá trị.

Nếu độ trễ gây ra bởi sự sửa đổi cùng một dữ liệu tại những site khác nhau thì những sửa đổi này sẽ được giải quyết một cách tự động (chọn site có độ ưu tiên cao hơn hay site đưa sửa đổi đến trước...).

20.1.2 Độc lập site (site autonomy)

Độc lập site xét đến ảnh hưởng của những thao tác trên một site đến các site khác. Thường độc lập site càng tăng thì tính nhất quán dữ liệu giảm. Nhân bản kết hợp (Merge replication) có mức độc lập site cao nhất, tạo ra sự hội tụ nhưng lại không đảm bảo tính nhất quán dữ liệu. 2PC (two phase commit) có tính nhất quán dữ liệu cao nhưng lại không có tính độc lập site. Những giải pháp khác thì thường ở giữa hai tính này.

20.2 Kiến trúc nhân bản

20.2.1 Các thành phần chính của nhân bản:

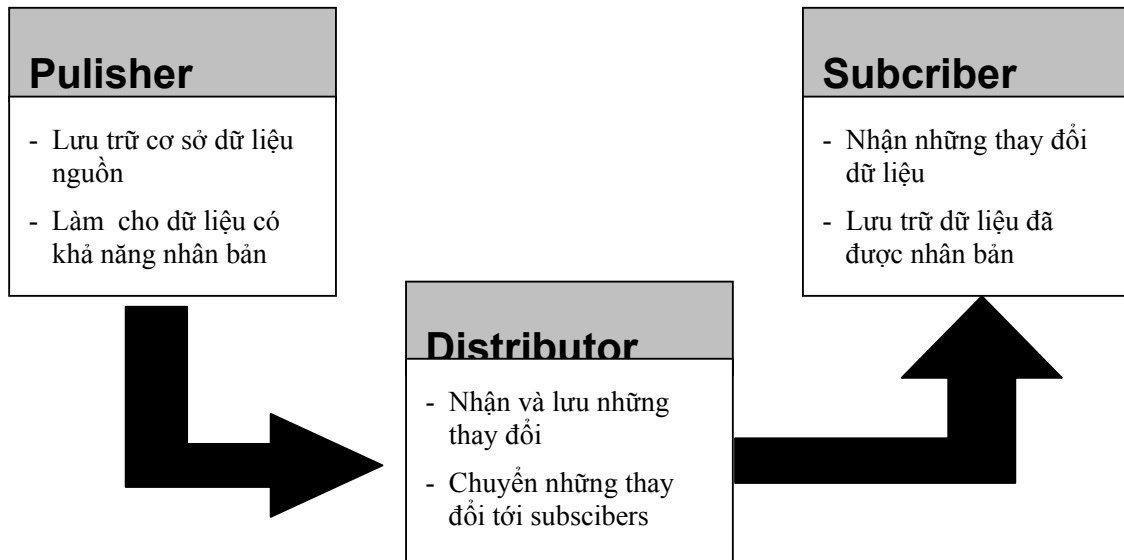
- Publisher: Là một server tạo dữ liệu để nhân bản đến các server khác. Nó xác định dữ liệu nào được nhân bản, dữ liệu nào thay đổi và duy trì những thông tin về các công bố tại site đó.

- Subscriber: Là một server lưu giữ nhân bản và nhận các tác vụ cập nhật. SQL Server 2000 cho phép Subscriber cập nhật dữ liệu nhưng quá trình cập nhật ở Subscriber không giống như ở Publisher. Một Subscriber có thể là một Publisher của các Subscriber khác.

- Distributor: Là một server mà chứa CSDL phân tán (distribution database) và lưu trữ metadata, history data và transaction. SQL Server sử dụng CSDL phân tán để lưu và chuyển (store_and_forward) dữ liệu nhân bản từ Publisher đến các Subscriber. Có 2 loại Distributor : Local Distributor và remote Distributor.

Chương 20. KIẾN TRÚC NHÂN BẢN (REPLICATION)

- Publication: Đơn giản là một tập hợp các mẫu dữ liệu (article). Một mẫu là một nhóm dữ liệu được nhân bản. Một mẫu có thể bao gồm một table hay chỉ là một vài hàng (horizontal fragment) hay cột (vertical fragment). Một Publication thường gồm nhiều mẫu.



20.2.2 Chiều di chuyển dữ liệu

Có 2 kiểu di chuyển dữ liệu:

20.2.2.1 Push subscription

- Publisher đẩy (push) những thay đổi đến Subscriber mà không quan tâm Subscriber có cập nhật hay không.
- Push subscription được sử dụng trong những ứng dụng mà yêu cầu gửi những thay đổi đến Subscriber ngay khi những thay đổi này xảy ra ở Publisher.
- Push Subscription giúp việc quản lý các Subscriber đơn giản và tập trung hơn, đồng thời giúp bảo mật tốt hơn vì quá trình khởi động (initialization process) sẽ được quản lý tại một chỗ. Nhưng vì thế, Distributor có thể phải đảm nhận nhiều quá trình phân bổ subscription đến các Subscriber cùng một lúc. Điều này dễ dẫn đến hiện tượng thắt cổ chai (bottleneck).
- Mô hình này không thích hợp khi số lượng các Subscriber trở nên quá lớn.
- Push subscription gây ra 1 phí xử lý cao hơn tại Publisher. Để tránh hiện tượng này, những thay đổi có thể được đẩy đến Subscriber theo một lịch định kì.

20.2.2.2 Pull subscription

- Subscriber kéo (pull) những thay đổi tại Publisher về theo một khoảng thời gian định kì.
- Tốt cho những user độc lập thay đổi bởi vì chúng cho phép user xác định khi nào thì những thay đổi dữ liệu được đồng bộ
- Ngược với push subscription ,pull subscription bảo mật thấp nhưng cho phép số lượng Subscriber cao hơn .
- Một publication có thể sử dụng cả hai push và pull subscription.

20.3 Tác nhân (Agent)

Việc thiết kế các nhân bản có thể tạo ra 1 hay nhiều agent.

Snapshot agent:

- Chuẩn bị lược đồ, data file, stored procedure
- Lưu snapshot lên Distributor và ghi lại những thông tin về trạng thái đồng bộ vào CSDL phân bố (distribution database) .
- Mỗi publication có 1 snapshot agent riêng chạy trên Distributor và liên kết với Publisher.

Log Reader agent:

- Di chuyển những transaction cần nhân bản từ transaction log trên Publisher đến CSDL phân bố .
- Mỗi publication dùng nhân bản transaction có một log reader agent, chạy trên Distributor và liên kết (connect) đến Publisher.

Distribution Agent:

- Di chuyển transaction và những tác vụ sao chép giữ trong CSDL phân bố đến Subscriber.
- TH: Nhân bản transaction hay snapshot mà đồng bộ lập tức (immediate synchronization): khi 1 push subscription được tạo, mỗi publication có 1 distribution agent riêng, chạy trên Distributor và liên kết với Subscriber.
- TH: Nhân bản transaction và snapshot không đồng bộ lập tức : Publisher và Subscriber sẽ dùng chung distribution agent , chạy trên Distributor và liên kết với Subscriber.
- TH: pull subscription đến snapshot publication hay transactional publication: có distribution agent, chạy trên Subscriber
- Nhân bản kết hợp (merge replication) không có distribution agent.

Merge agent:

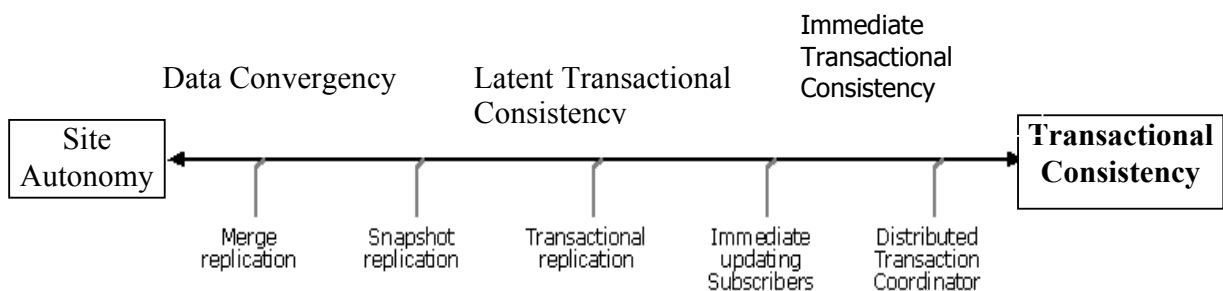
Di chuyển và điều hòa những thay đổi dữ liệu xảy ra sau khi 1 snapshot khởi động (initial snapshot) được tạo. Mỗi merge publication có một merge agent, liên kết và cập nhật được với cả hai Publisher và Subscriber.

20.4 Các loại nhân bản

Trong thực tế khó có thể có được một loại nhân bản phù hợp mọi yêu cầu. Công việc kinh doanh thường đòi hỏi nhiều ứng dụng khác nhau vì thế SQL Server đã đưa ra nhiều cách thức nhân bản để đáp ứng các yêu cầu đó.

SQL Server đưa ra 3 loại nhân bản để sử dụng khi thiết kế ứng dụng:

- Nhân bản snapshot
- Nhân bản transaction
- Nhân bản kết hợp



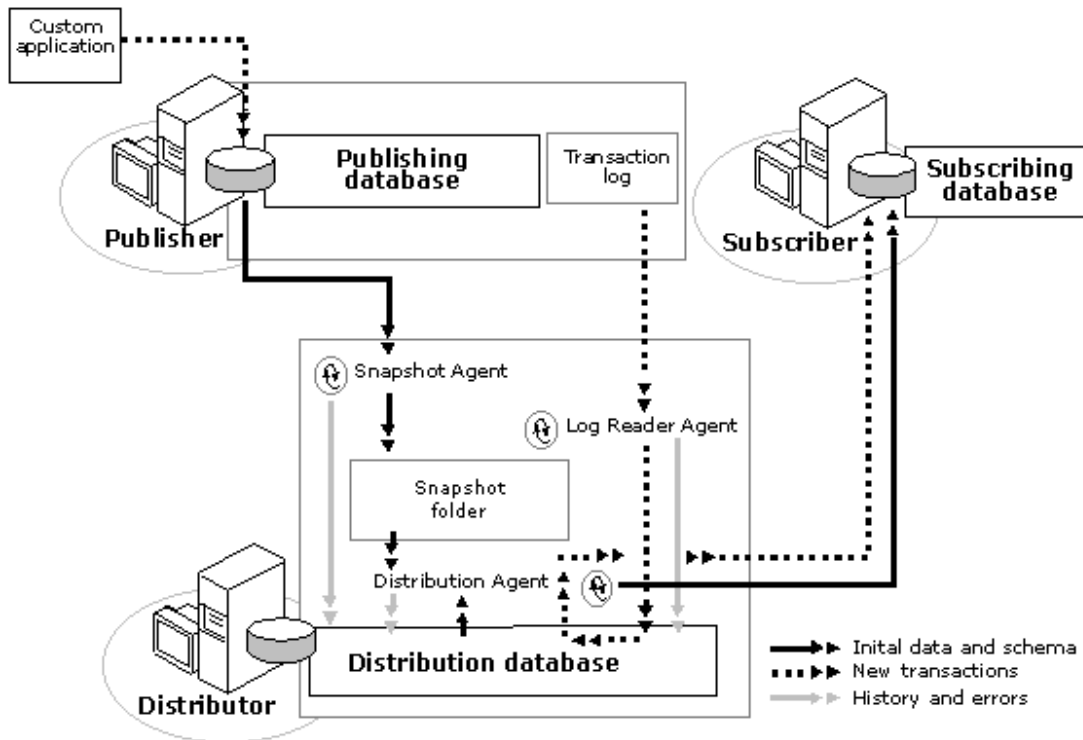
Mỗi loại cung cấp các khả năng và thuộc tính khác nhau nhằm đạt đến mục tiêu của tính độc lập site và sự nhất quán dữ liệu.

20.5 Nhân bản snapshot(Snapshot replication)

20.5.1 Giới thiệu

Nhân bản snapshot là loại nhân bản đơn giản nhất, nhân bản snapshot sao chép toàn bộ dữ liệu cần nhân bản (còn gọi là quá trình làm tươi dữ liệu) từ Publisher đến các Subscriber. Nó đảm bảo sự nhất quán tiềm ẩn (Latent Transactional Consistency) giữa Publisher và Subscriber. Nhân bản snapshot được đánh giá cao trong các ứng dụng chỉ đọc như tìm kiếm hay các hệ thống không yêu cầu dữ liệu mới nhất và dung lượng dữ liệu không lớn.

Nhân bản Snapshot gửi tất cả dữ liệu đến cho Subscriber thay vì chỉ gửi những thay đổi. Nếu mẫu dữ liệu rất lớn nó phải cần đến hệ thống mạng đủ mạnh để truyền dữ liệu. Khi sử dụng nhân bản snapshot cần phải tính đến tỉ lệ giữa kích cỡ của toàn bộ dữ liệu và những thay đổi của nó.



Hình 20.1. Nhân bản snapshot(Snapshot replication)

20.5.2 Tác nhân (agent)

Cập nhật snapshot được thực hiện bởi snapshot agent và distribution agent. Snapshot agent chuẩn bị những snapshot file (*snapshot file là file sao chép lược đồ và dữ liệu của những table phân bố*) chứa lược đồ và dữ liệu của những table phân bố, lưu những file này vào snapshot folder trên Distributor và ghi lại những công việc đồng bộ trong CSDL phân bố (distribution database). Distribution agent gửi những snapshot job (tác vụ sao chép dữ liệu) giữ trong bảng dữ liệu phân bố đến Subscriber.

CSDL phân bố (distribution database) chỉ được sử dụng trong nhân bản, không chứa user table.

20.5.2.1 Snapshot agent

Snapshot agent thực hiện theo các bước sau:

- Thiết lập một **share-lock** lên tất cả table (article) trong publication. Share-lock ngăn không cho các user khác cập nhật lên table đó.
- Sao chép lược đồ dữ liệu của mỗi article (*.sch file*) và các index, các ràng buộc (*.idx file*) lên Distributor. Các file này được lưu vào 1 thư mục con trong thư mục làm việc của Distributor.

- Nếu tất cả các Subscriber đều là MS SQL Server thì bản sao của dữ liệu được lưu thành **.bcp file**. Nếu các Subscriber không đồng nhất (các Subscriber chứa nhiều loại CSDL khác nhau , ví dụ: Access, Oracle...) thì bản sao của dữ liệu được lưu thành **.txt file**.
- Cuối cùng agent gỡ bỏ **share-lock** trên mỗi table phân bố và hoàn tất việc viết vào 1 log history file (*log history file ghi lại quá trình làm việc của các agent*).

20.5.2.2 Distribution agent

Tác nhân áp dụng những lược đồ và những dữ liệu vào CSDL của Subscriber. Nếu Subscriber không phải là SQL Server, distribution agent sẽ chuyển đổi kiểu dữ liệu trước khi những dữ liệu này được áp dụng vào Subscriber.

Ví dụ: Publisher sử dụng SQL Server, Subscriber sử dụng Oracle. Trước khi những dữ liệu được áp dụng lên Subscriber, nó sẽ được chuyển đổi kiểu từ SQL Server sang Oracle.

Snapshot có thể được áp dụng khi subscription được tạo hay theo 1 khoảng thời gian nhất định.

20.6 Nhân bản giao dịch (transactional replication)

20.6.1 Giới thiệu

Sử dụng nhân bản giao dịch để nhân bản hai kiểu đối tượng khác nhau: table và stored procedure. Bạn có thể chọn tất cả hay một phần của một table được nhân bản như là một article trong publication. Tương tự, bạn cũng có thể chọn một hay nhiều stored procedure được nhân bản như là một article trong cùng hay khác publication.

Nhân bản giao dịch sử dụng transaction log để giữ những thay đổi được làm trên dữ liệu trong một article. SQL Server giám sát những lệnh insert, update, delete hay những sửa đổi trên dữ liệu và lưu những thay đổi đó lên CSDL phân bố (distribution database). Những thay đổi đó sẽ được gửi đến Subscriber và tuân theo một trật tự nhất định.

Với nhân bản giao dịch, bất cứ yếu tố dữ liệu nào cũng có một publication. Những thay đổi được làm tại Publisher tiếp tục chảy đến một hay nhiều các Subscriber hay theo những khoảng thời gian định trước.

20.6.2 Tác nhân (agent)

Nhân bản giao dịch được thực hiện bởi Snapshot agent, Log Reader agent và Distribution agent. Log Reader agent giám sát transaction log của mỗi CSDL được

thiết lập để nhân bản và sao chép những transaction cần nhân bản từ transaction log vào CSDL phân bố (distribution database). Distribution agent di chuyển những transaction và những tác vụ khởi tạo snapshot được giữ trong table của CSDL phân bố.

20.6.2.1 Snapshot agent

Trước khi một Subscriber mới có thể nhận những thay đổi từ Publisher, nó phải chứa những table có cùng lược đồ và dữ liệu với những table tại Publisher. Quá trình copy toàn bộ publication từ Publisher qua Subscriber được gọi là **initial snapshot**. Việc nhân bản những dữ liệu thay đổi chỉ xảy ra sau khi nhân bản giao dịch chắc chắn rằng Subscriber có snapshot (bản sao của những lược đồ và dữ liệu). Khi những snapshot đó được phân bố và áp dụng lên các Subscriber thì chỉ những Subscriber chờ để khởi tạo snapshot mới bị ảnh hưởng. Những Subscriber khác ứng với publication đó mà nhận insert, delete, update hay những thay đổi dữ liệu rồi thì không bị ảnh hưởng. Những hàm mà Snapshot agent thực thi để khởi tạo snapshot trong nhân bản giao dịch tương tự như các hàm được sử dụng trong nhân bản Snapshot.

20.6.2.2 Log Reader agent

Log reader agent chạy tiếp tục hay theo một khoảng thời gian xác định mà bạn thiết lập vào lúc publication được tạo. Khi thực thi, đầu tiên Log reader agent đọc transaction log của publication và xác định lệnh (insert, delete, update) hay những sửa đổi làm trên dữ liệu được đánh dấu nhân bản. Kế tiếp agent sao chép những transaction đó vào CSDL phân bố tại Distributor. CSDL phân bố (distribution database) trở thành hàng lưu và đẩy (store-and-forward queue) những thay đổi dữ liệu đến Subscriber. Chỉ có commit transaction mới được gửi đến CSDL phân bố.

Có sự tương ứng 1-1 giữa transaction trên Publisher và transaction được nhân bản trong CSDL phân bố. Một transaction có thể bao gồm nhiều lệnh. Sau khi toàn bộ transaction được viết vào CSDL phân bố một cách thành công, nó sẽ được commit. Sau đó những transaction này sẽ được đẩy đến các Subscriber. Cuối cùng, agent đánh dấu những hàng (row) đã được công bố đến Subscriber trong transaction log để sẵn sàng loại bỏ. Điều này đảm bảo những hàng (row) còn chờ để nhân bản sẽ không bị loại bỏ. Vì thế, transaction log trên Publisher có thể được đổ xuống mà không cản trở việc nhân bản bởi vì chỉ những transaction bị đánh dấu mới bị loại bỏ.

Log read agent thực thi trên Distributor.

20.6.2.3 Distribution agent

Những transaction được lưu trong CSDL phân bố cho đến khi distribution agent “đẩy” chúng đến tất cả các Subscriber (hoặc một distribution agent tại Subscriber “kéo” những thay đổi về). CSDL phân bố chỉ được sử dụng bởi nhân bản

Chương 20. KIẾN TRÚC NHÂN BẢN (REPLICATION)

và không chứa bất cứ user table. Trong bất kì trường hợp nào bạn cũng không nên tạo những object khác vào trong CSDL phân bố. Những tác vụ làm thay đổi dữ liệu tại Publisher sẽ chảy đến Subscriber và Subscriber sẽ thay đổi dữ liệu theo cùng cách chúng được thay đổi tại Publisher. Điều này đảm bảo rằng các Subscriber sẽ nhận những transaction theo một trật tự như là chúng được làm tại Publisher .

Những hàm mà distribution agent sử dụng để di chuyển những lệnh đến Subscriber cũng tương tự như những hàm được sử dụng trong nhân bản snapshot.

20.6.3 Thu dọn trong nhân bản transaction

Tương tự cho nhân bản snapshot.

Khi CSDL phân tán(distribution database) được tạo, SQL Server sẽ tự động thêm vào 3 tác vụ tại Distributor:

- Agent checkup
- Transaction cleanup
- History cleanup

Những tác vụ này giúp cho việc nhân bản hiệu quả hơn trong môi trường long_running. Tác vụ cleanup giữ lại những transaction của mỗi publication trong một giai đoạn xác định sau khi tất cả Subscriber đã nhận chúng. Trong suốt giai đoạn này những transaction sẽ được giữ trong CSDL phân bố. Thiết lập một giai đoạn giữ lại kết hợp với lịch backup có thể được sử dụng khôi phục CSDL đích một cách tự động khi xảy ra sự cố.

20.7 Các dạng nhân bản giao dịch

Có hai dạng:

20.7.1 Cập nhật Subscriber lập tức(Immediate_Updating Subscriber)

Trong trường hợp đơn giản nhất, cả hai nhân bản snapshot và giao dịch làm việc dựa trên mô hình nhân bản một chiều (dữ liệu chỉ được nhân bản từ Publisher đến Subscriber). Tuy nhiên MS SQL Server cung cấp thêm một mô hình mới cho phép Subscriber sửa đổi dữ liệu nhân bản, tùy chọn Immediate_Updating Subscriber sẽ cung cấp sự nhất quán giao dịch ngầm (latent transactional consistency) giữa các Subscriber (những sửa đổi sẽ xảy ra lập tức giữa Subscriber thực hiện tác vụ cập nhật và Publisher) mà không yêu cầu cập nhật chỉ được làm tại Publisher site. Tùy chọn này được thiết lập vào lúc publication được tạo và cho phép Subscriber cập nhật bản sao dữ liệu cục bộ của nó và những thay đổi đó sẽ được phản ánh lập tức đến Publisher bằng cách sử dụng two-phase commit protocol (2PC). 2PC được quản lý bởi Microsoft Distributed Transaction Coordinator (MS DTC). Nếu cập nhật có thể được thực hiện bằng 2PC thì Publisher sẽ phổ biến những thay đổi đến tất cả các

Chương 20. KIẾN TRÚC NHÂN BẢN (REPLICATION)

Subscriber khác theo lịch làm việc của distribution agent (hay là theo lần làm tươi snapshot kế tiếp). Bởi vì Subscriber gốc đã cập nhật những thay đổi dữ liệu rồi, nên user có thể tiếp tục làm việc với những dữ liệu đó và đảm bảo những thay đổi đó sẽ được cập nhật tại Publisher. Mô hình này đảm bảo tính toàn vẹn giao dịch.

Với mô hình này, tính độc lập site sẽ giảm, nhưng vẫn cao hơn khi tất cả các Subscriber cập nhật lập tức.

Tùy chọn Immediate-updating Subscribers được hỗ trợ sử dụng những công cụ :

- Triggers
- Stored procedues
- Microsoft Distributed Transaction Coordinator (MS DTC)
- Phát hiện tranh chấp
- Phát hiện Loopback

Triggers:

Triggers tại Subscriber theo dõi các transaction và báo về cho các publication bằng cách sử dụng những stored procedure gọi từ xa (remote stored procedure call) trong 2- phase commit protocol (2PC) mà được điều khiển bởi MS DTC . Trigger sẽ thực hiện các công việc:

- Trích giá trị từ những bảng insert hay delete
- Gọi lệnh BEGIN DISTRIBUTED TRAN {SACTION}
- Thực thi một remote procedure để gọi stored procedure thích hợp tại Publisher, thông qua những giá trị từ những bảng insert hay delete.
- Điều khiển cập nhật giá trị timestamp & indentity mới tại Subscriber
- Nếu gọi những stored procedure từ xa thành công, thì commit transaction phản ánh chính xác cùng những thay đổi tại cả hai Publisher và Subscriber. Sau đó, Publisher bảo đảm rằng những thay đổi được phổ biến đến tất cả các Subscriber khác. Ngược lại nếu không được Subscriber sẽ hủy bỏ (rollback) giao dịch và trả về những lỗi cho user.

Stored procedures:

Stored procedure tại Publisher thực thi những giao dịch khi giao dịch đó không tranh chấp với những thay đổi được làm tại Publisher. Nếu một tranh chấp được phát hiện, giao dịch bị hủy bỏ (roll back) ở cả hai site. Mỗi article có ba hàm insert, delete, update. Mỗi hàm tại Publisher sẽ thực hiện các công việc sau:

Chương 20. KIẾN TRÚC NHÂN BẢN (REPLICATION)

- Insert procedure : Cố gắng insert hàng (row). Sau đó kiểm tra giá trị @@ROWCOUNT/@@ERROR và trả về tín hiệu thành công hay sự cố cho lời gọi trigger đó của Subscriber.
- Delete procedure : Cố gắng xóa hàng . Sau đó kiểm tra giá trị @@ROWCOUNT/@@ERROR và trả về tín hiệu thành công hay sự cố cho lời gọi trigger đó của Subscriber.
- Update procedure : cố gắng cập nhật hàng có cùng giá trị khoá và timestamp với hàng trong bảng delete. Kiểm tra @@ROWCOUNT / @@ERROR. Nếu thành công, trả về một giá trị timestamp mới.

SQL Server tổ chức 2 bảng (table) : delete và insert để lưu những dữ liệu thay đổi được làm trên một bảng (table) mà chưa được commit. Thực tế lệnh update bao gồm 1 hàng trong bảng insert và 1 hàng trong bảng delete.

Chú ý: Một giao dịch mà ảnh hưởng lên nhiều hàng thì giao dịch đó chỉ được commit khi tất cả các hàng đều được cập nhật lên cả 2 site .

MS DTC (Microsoft Distributed Transaction Coordinator):

MS DTC quản lý những tác vụ 2-phase commit giữa một Subscriber và Publisher trong một lệnh gọi từ xa (BEGIN DISTRIBUTED TRANSACTION trong Transact-SQL).

Phát hiện tranh chấp:

Những stored procedure của Publisher sử dụng cột timestamp để phát hiện một hàng có thay đổi hay không sau khi nó được nhân bản đến Subscriber. Khi Subscriber yêu cầu một giao dịch cập nhật lập tức (immediate-update transaction), nó gửi giá trị timestamp đến Publisher với tất cả những cột khác trong hàng. Khi đó stored procedure của Publisher so sánh giá trị này với giá trị timestamp hiện tại của hàng. Nếu giá trị này giống nhau thì hàng không được sửa đổi sau khi nó được nhân bản đến Subscriber và vì thế giao dịch này được chấp nhận.

Timestamp là một giá trị tự động tăng và duy nhất trong một cơ sở dữ liệu.

Phát hiện loopback :

Nếu một transaction được thực thi thành công lên một Subscriber và Publisher, không cần thiết phổ biến những thay đổi trở về cho Subscriber gốc (Subscriber đưa những thay đổi đến Publisher). SQL Server có một cơ chế gọi là phát hiện loopback (loopback detection) để xử lý tình huống này.

Những thông tin sử dụng để phát hiện loopback được lưu thành một transaction bởi cơ sở transaction. Những bảng mà định cư trong những cơ sở dữ liệu khác nhau tại immediate updating Subscriber hay những bảng mà định cư trong những cơ sở dữ liệu khác nhau ở phía bên kia của immediate-updating Subscriber không nên được update trong cùng 1 transaction.

20.7.2 Nhân bản những thực thi của Stored procedure

SQL server không những cho phép bạn nhân bản dữ liệu trong bảng mà còn hỗ trợ bạn nhân bản stored procedure một trong hai cách. Nếu bạn có một hay nhiều stored procedure như là những article trong một snapshot publication, SQL server sao chép toàn bộ store procedure từ Publisher đến Subscriber. Nếu bạn gồm một hay nhiều stored procedure như là article trong một nhân bản giao dịch, SQL Server sẽ nhân bản thực thi của stored procedure hơn là những dữ liệu thay đổi gây ra bởi sự thực thi những stored procedure đó. Cách làm này đặc biệt hữu ích trong nhân bản mà kết quả của những stored procedure có thể ảnh hưởng một số lượng lớn dữ liệu. Nhân bản những thay đổi như thực thi một lệnh đơn làm tăng hiệu quả ứng dụng của bạn.

Có 2 loại:

- Procedure execution
- Serializable Procedure execution

Procedure execution:

Nhân bản procedure execution đến tất cả Subscriber. Điều này xảy ra bất chấp những lệnh đơn trong stored procedure có thành công hay không. Bởi vì những thay đổi dữ liệu được làm bởi stored procedure có thể xảy ra trong nhiều giao dịch, nên dữ liệu tại Subscriber không thể bảo đảm là sẽ nhất quán với dữ liệu tại Publisher.

Serializable Procedure execution:

Chỉ thực hiện nhân bản procedure execution khi procedure được thực thi trong một chuỗi giao dịch tuần tự. Cách này đảm bảo dữ liệu tại Subscriber nhất quán với dữ liệu tại Publisher.

20.8 Nhân bản kết hợp (Merge replication)

20.8.1 Giới thiệu

Nhân bản kết hợp có tính độc lập site (site autonomy) cao nhất. Publisher và Subscriber có thể làm việc hoàn toàn độc lập và sẽ kết nối với nhau theo những khoảng thời gian để hội tụ các kết quả lại. Nếu đụng độ gây ra bởi các site cùng sửa đổi trên cùng một phần tử dữ liệu thì những đụng độ này sẽ được giải quyết một cách tự động. Khi đụng độ xảy ra, bộ giải quyết đụng độ sẽ chọn site có độ ưu tiên cao hơn hay site sửa đổi dữ liệu đó trước. Các xung đột này có thể được phát hiện và giải quyết theo cấp độ hàng hay cột của bảng dữ liệu.

Nhân bản kết hợp nhận biết những thay đổi trong một CSDL nguồn và đồng bộ những giá trị giữa Publisher và Subscriber. Cả hai Publisher và Subscriber đều có thể cập nhật dữ liệu. Trong nhân bản kết hợp, Publisher là server tạo publication. Mặc dù Publisher tạo publication nhưng nó không tự động “thắng” 1 tranh chấp với

1 Subscriber. "Người thắng cuộc" được xác định bởi tiêu chuẩn do bạn thiết lập và những thay đổi dữ liệu tại CSDL đích sẽ được phổ biến đến CSDL nguồn.

20.8.2 Tác nhân (agent)

Nhân bản kết hợp được thực hiện bởi snapshot agent và merge agent. Snapshot agent chuẩn bị những snapshot file chứa lược đồ và dữ liệu của những table phân bố, lưu những file này vào snapshot folder trên Distributor và ghi lại những công việc đồng bộ trong publication. Merge agent thực hiện những công việc khởi tạo snapshot được tổ chức trong bảng (table) của publication đến Subscriber. Nó cũng kết hợp những dữ liệu thay đổi xảy ra tại Publisher sau khi initial snapshot được tạo và giải quyết tranh chấp theo những luật mà bạn đặt ra hay sử dụng bộ giải quyết tranh chấp (conflict resolver).

Snapshot agent:

Tương tự như Snapshot agent của nhân bản transaction.

Merge agent:

Khi một hàng được cập nhật trong một mẫu (article) một trigger tạo cột generation cho hàng đó và gán nó bằng 0. Khi merge agent được thực thi nó sẽ thu thập tất cả những hàng có generation bằng 0 thành một hay nhiều nhóm và gán cho generation một giá trị lớn hơn tất cả những generation trước đó. Merge agent tại mỗi site sẽ theo dõi generation cao nhất mà nó gửi đến các site khác và các generation cao nhất mà các site khác đã gửi đến nó. Những generation này được lưu trong hàng (row) có thể khác nhau giữa các site bởi vì generation tại một site phản ánh thứ tự những thay đổi được xử lý tại site đó.

Vào lúc đồng bộ, merge agent gửi tất cả những dữ liệu thay đổi đến site khác. Tại CSDL nguồn, những giá trị đến được kết hợp với những giá trị đã tồn tại. Merge agent ước lượng cả hai giá trị dữ liệu đến và hiện có và bất cứ tranh chấp nào giữa hai giá trị cũ và mới cũng được giải quyết một cách tự động dựa theo độ ưu tiên hay user thay đổi dữ liệu trước hay kết hợp giữa hai cách trên (dùng với nhóm site có độ ưu tiên tương đương nhau). Bạn cũng có thể thực thi những chiến lược giải quyết tranh chấp thông qua COM hay bộ giải quyết stored procedure (stored procedure resolver). Những dữ liệu thay đổi được nhân bản đến những site khác chỉ khi một đồng bộ xảy ra và việc đồng bộ này có thể mất vài phút, vài ngày hay thậm chí vài tuần.

Nhân bản kết hợp có tính độc lập site rất cao. Tất cả site đều có thể thực hiện update, delete, insert trên dữ liệu phân bố trên site của nó và độc lập với những thay đổi được làm trên những site khác. Tuy nhiên nhân bản kết hợp không đảm bảo tính toàn vẹn giao dịch. Thay vì vậy nó đẩy mạnh sự hội tụ dữ liệu. Tất cả những thay đổi được làm tại tất cả các site sẽ hội tụ về cùng một giá trị, mặc dù giá trị đó không đảm bảo là giống nhau như là tất cả những thay đổi đó được áp dụng lên một site. Vì vậy kiểu này không thích hợp cho những ứng dụng yêu cầu toàn vẹn giao dịch.

Merger agent là một công cụ của SQL Server Agent và có thể được quản lý trực tiếp bằng cách sử dụng SQL Server Enterprise Manager. Snapshot agent thực thi trên Distribution. Merge agent thực thi trên Distribution khi dùng push subscription hay thực thi trên Subscriber khi dùng pull subscription.

20.8.3 Giải quyết tranh chấp trong nhân bản kết hợp

MA phát hiện tranh chấp thông qua một cột hệ thống gọi là lineage trong bảng MSmergecontents, đại diện cho quá trình thay đổi trong một hàng. Agent cập nhật cột lineage trong MS mergecontents một cách tự động khi một user cập nhật hàng. Mỗi cột chứa một mục (entry) cho mỗi site mà cập nhật lên hàng. Mỗi entry kết hợp giữa chỉ số (id) của site và version cuối cùng của hàng được tạo bởi site đó. Khi MA kết hợp những thay đổi, và nó đụng phải một hàng mới thay đổi, nó sẽ xem xét cột lineage của hàng trên mỗi site để xác định có tranh chấp hay không? Khi tranh chấp xảy ra, agent khởi động một bộ hoà giải tự động. “Người thắng” tranh chấp có thể dựa theo độ ưu tiên hoặc giải pháp chọn người đến trước nhất hay phương pháp truyền thống sử dụng bộ giải quyết tranh chấp COM hay store procedure.

Tranh chấp dữ liệu trong table có thể được nhận biết ở cấp độ cột hoặc cấp độ hàng. Chọn lựa (option) mặc định là cột (column-tracked articles). Chọn lựa này cho phép những thay đổi được làm trên từng cột riêng biệt nhau, chỉ những thay đổi trên cùng một cột bị đánh dấu như là một tranh chấp. Tuy nhiên trong một vài ứng dụng, những luật của ứng dụng của bạn có thể đối xử đồng thời những thay đổi đến toàn bộ hàng như là một tranh chấp. Trong trường hợp này, cấp độ hàng là một chọn lựa.

20.9 Giải quyết tranh chấp

Khi tranh chấp xảy ra, một bộ giải quyết tranh chấp (conflict resolver) sẽ xác định tranh chấp được giải quyết như thế nào. Resolver áp dụng một tập luật qui định lên dữ liệu tranh chấp và chọn tác vụ thích hợp. Conflict resolver tạo ra những bảng conflict-usertablename để lưu những thông tin về tranh chấp. Bảng này được giữ tại Publisher cho những ứng dụng sử dụng centralized conflict logging và tại Subscriber đối với những ứng dụng sử dụng decentralized conflict logging. Bảng này có cùng cấu trúc với bảng gốc, và conflict resolver sao chép phiên bản (version) gần nhất của hàng vào bảng. Version “thắng cuộc” của hàng định cư trong user table thật sự. Những cột trong bảng hệ thống sysmergearticles giữ những thông tin về những bảng có liên quan đến bảng tranh chấp, và tên của những bảng đó. Xoá bỏ những tranh chấp sẽ lần theo bảng MSmerge-delete-conflicts. Với một vài tranh chấp bạn không thể phổ biến những thay đổi từ một site đến các site khác. Ví dụ hai site cùng insert một hàng có cùng một khoá dẫn đến xảy ra tranh chấp. Nếu mỗi lệnh insert đều thành công, thì những ràng buộc vi phạm sẽ không được biết cho đến khi quá trình đồng bộ site xảy ra. Lúc này MS SQL Server tự động xoá một trong hai hàng có cùng khoá chính đó. Những thông tin lỗi này cũng được lưu trong bảng tranh chấp. Những vấn đề khác như là vi phạm ràng buộc duy nhất yêu cầu một vài tác vụ user

Chương 20. KIẾN TRÚC NHÂN BẢN (REPLICATION)

để khôi phục sự hội tụ. Vi phạm tính toàn vẹn hay gặp nhất là insert một hàng với một khoá ngoại trong khi site khác thì đang xoá hàng với khoá chính tương ứng. SQL Server cũng tự xoá những hàng vi phạm các ràng buộc này.

SQL Server nhận ra sự cần thiết cho những ứng dụng để có một lược đồ giải quyết tranh chấp mà xảy ra trong suốt quá trình kết hợp. Khi xây dựng ứng dụng, bạn có 3 thay đổi theo cơ chế giải quyết tranh chấp:

- Giải quyết tranh chấp dựa trên độ ưu tiên là mặc định khi bạn tạo ra ứng dụng.
- Giải quyết theo những store procedure mà bạn xây dựng theo những luật hay theo những dữ liệu xác định của bạn.
- Bộ giải quyết COM.

SQL server cho phép bạn xây dựng một bộ giải quyết tranh chấp (conflict resolver). Tuy nhiên, sử dụng bộ giải quyết tranh chấp COM thì phức tạp hơn là thực thi một custom store procedure resolver. Bạn nên sử dụng store procedure conflict resolver bất cứ khi nào có thể.

TÀI LIỆU THAM KHẢO

1. Design Team: H.O. Mumbai, *Implementing RDBMS Concepts with SQL Server 2000*, 2002.
2. Tiện ích Book Online của SQL Server 2000.
3. Elmasri & Navathe: *Fundamentals of Database Systems*, International Edition.
4. Design Team: H.O. Mumbai, *Database Design with MS Access 2000*, 2002.
5. Date, C.J., and Darwen, H.: *A Guide to the SQL Standard*, 3rd ed., Addison-Wesley.