

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC KINH TẾ QUỐC DÂN

**BÀI GIẢNG
HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**

**BIÊN SOẠN
TS NGUYỄN TRUNG TUẤN**

HÀ NỘI – 2020

MỤC LỤC

MỤC LỤC	2
CHƯƠNG I. GIỚI THIỆU MICROSOFT SQL SERVER	5
1.1. Tổng quan về hệ quản trị cơ sở dữ liệu.....	5
1.1.1 Hệ quản trị cơ sở dữ liệu	5
1.1.2 Một số hệ quản trị cơ sở dữ liệu thông dụng.....	9
1.2. Giới thiệu Microsoft SQL Server	15
1.2.1 Giới thiệu về Microsoft SQL Server.....	15
1.2.2 Kiến trúc và thành phần của Microsoft SQL Server	18
1.3. Cài đặt Microsoft SQL Server.....	20
1.4. Các công cụ thông dụng trong Microsoft SQL Server	28
1.4.1 MicroSoft SQL Server Management Studio	28
1.4.2 MicroSoft SQL Server Configuration Manager.....	32
1.4.3 MicroSoft SQL Server Profiler.....	33
1.4.4 MicroSoft Database Engine Tuning Advisor	34
1.4.5 SQLServer Books Online và SQLServer Tutorials.....	35
CHƯƠNG II. LÀM VIỆC VỚI CƠ SỞ DỮ LIỆU TRONG MICROSOFT SQL SERVER	36
2.1 Cơ sở dữ liệu quản lý bán hàng.....	36
2.2 Các kiểu dữ liệu trong MicroSoft SQL Server.....	39
2.2.1 Các kiểu dữ liệu	39
2.2.2 Các hàm cơ bản.....	44
2.3 Sử dụng MicroSoft SQL Server Management Studio để tạo cơ sở dữ liệu....	51
2.3.1 Tạo cơ sở dữ liệu.....	51
2.3.2 Tạo bảng.....	53
2.3.3 Tạo lược đồ quan hệ (Diagram).....	55

2.4 Khai thác cơ sở dữ liệu bằng lệnh truy vấn SELECT	57
2.4.1 Cú pháp lệnh.....	57
2.4.2 Một số ví dụ truy vấn khai thác dữ liệu	60
2.5 Các lệnh tạo lập cơ sở dữ liệu, bảng, chỉ mục.....	66
2.5.1 Các lệnh với cơ sở dữ liệu.....	66
2.5.2 Các lệnh với bảng	72
2.5.3 Thêm, sửa, xóa dữ liệu trong bảng	77
2.5.4 Các lệnh với chỉ mục (index)	79
2.6 Khung nhìn (View).....	85
CHƯƠNG III. LẬP TRÌNH TRONG SQL SERVER.....	91
3.1 Giới thiệu.....	91
3.2 Lập trình thủ tục lưu trữ.....	92
3.2.1 Tạo thủ tục lưu trữ	92
3.2.2 Thực thi thủ tục lưu trữ	94
3.2.3 Một số lệnh cơ bản trong lập trình thủ tục	97
3.2.4 Sửa, xóa thủ tục lưu trữ.....	101
3.2.5 Xem các dòng lệnh trong thủ tục lưu trữ	102
3.3 Các cấu trúc trong lập trình.....	103
3.3.1 Cấu trúc rẽ nhánh.....	103
3.3.2 Hàm CASE	106
3.3.3 Lệnh GOTO	111
3.3.4 Cấu trúc lặp.....	113
3.3.5 Kiểu dữ liệu con trỏ (CURSOR)	116
3.3.6 Làm việc với dữ liệu XML.....	123
3.3.7 Kiểm soát lỗi.....	127
3.4 Lập trình cho hàm.....	132

3.4.1 Tạo hàm trong MicroSoft SQL Server.....	132
3.4.2 Sửa, xóa hàm	137
3.5 Lập trình Trigger	138
3.5.1 Giới thiệu.....	138
3.5.2 Tạo, sửa, xóa Trigger	139
3.6 Làm việc với TRANSACTION.....	143
CHƯƠNG IV. QUẢN TRỊ SQL SERVER.....	148
4.1 Tự động hoá công việc quản trị.....	148
4.1.1 Cấu hình dịch vụ SQL Server Agent	148
4.1.2 Thực thi công việc (Jobs)	150
4.1.3 Cảnh báo (Alerts).....	157
4.1.4 Sinh mã lệnh cho các đối tượng trong Cơ sở dữ liệu	159
4.2 Sao lưu, phục hồi và quản lý cơ sở dữ liệu.....	164
4.2.1 Sao lưu.....	164
4.2.2 Phục hồi.....	168
4.2.3 Quản lý cơ sở dữ liệu	170
4.3 Nhập (Import) và xuất (Export) dữ liệu.....	173
4.4 Bảo mật trong SQL Server.....	178
4.4.1 Giới thiệu về bảo mật SQL Server.....	178
4.4.2 Thực thi và quản trị bảo mật CSDL.....	179
4.4.3 Gán quyền và thu hồi quyền.....	186
TÀI LIỆU THAM KHẢO	192

CHƯƠNG I. GIỚI THIỆU MICROSOFT SQL SERVER

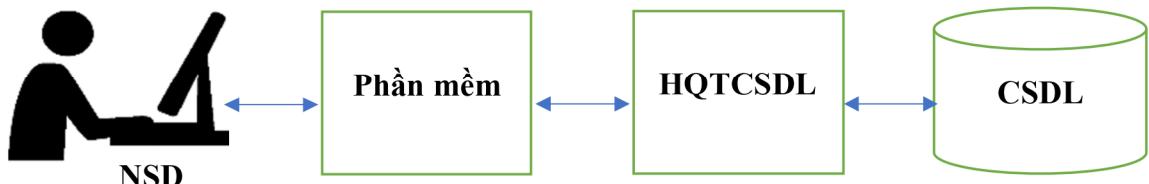
1.1. TỔNG QUAN VỀ HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

1.1.1 Hệ quản trị cơ sở dữ liệu

Quản trị cơ sở dữ liệu là một nhiệm vụ phức tạp, nhưng đã có những gói phần mềm được gọi là hệ quản trị cơ sở dữ liệu (Database Management System – DBMS) có thể thực hiện các công việc quản trị cơ sở dữ liệu cho chúng ta. Một hệ quản trị cơ sở dữ liệu là một phần mềm, hoặc một tập hợp các phần mềm mà qua đó người sử dụng có thể tương tác với cơ sở dữ liệu. Các thao tác xử lý thực tế trên cơ sở dữ liệu cơ bản được xử lý bằng hệ quản trị cơ sở dữ liệu. Trong một số trường hợp, người sử dụng có thể tương tác trực tiếp với cơ sở dữ liệu thông qua hệ quản trị cơ sở dữ liệu. Trong những trường hợp khác, người sử dụng tương tác thông qua các phần mềm được viết bằng một ngôn ngữ nào đó, phần mềm này sẽ tương tác với hệ quản trị cơ sở dữ liệu để truy cập đến cơ sở dữ liệu.



Hình 1. Người sử dụng tương tác trực tiếp CSDL thông qua HQTCSDL



Hình 2. Người sử dụng tương tác với CSDL thông qua phần mềm và HQTCSDL

Những ưu điểm:

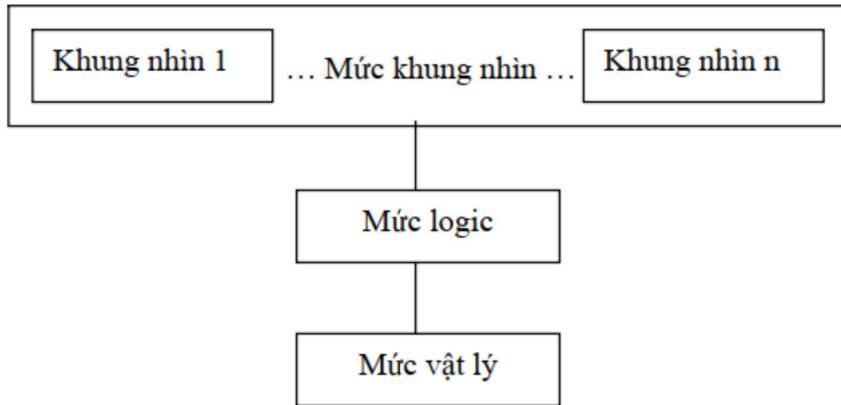
- Thu được lượng thông tin nhiều hơn với cùng một khối lượng dữ liệu: Mục đích cơ bản của các hệ thống máy tính là chuyển dữ liệu thành thông tin. Với các dữ liệu dưới dạng tệp, không phải là cơ sở dữ liệu, thì thường bị phân mảnh thành nhiều hệ thống không kết nối, mỗi hệ thống sẽ có một tập hợp các tệp của riêng mình. Với các yêu cầu thông tin mà cần phải truy cập dữ liệu từ nhiều tập hợp đó có thể sẽ rất khó khăn để thực hiện. Trong một số trường hợp

thực tế, điều đó còn có thể không thực hiện được. Vì vậy, thông tin mong muốn không sẵn sàng, dữ liệu được lưu trong máy tính nhưng không thể thu thập thông tin được thông qua nhiều tệp. Tuy nhiên, khi tất cả dữ liệu cho các hệ thống đa dạng được lưu trữ trong một cơ sở dữ liệu, thì thông tin trở thành sẵn sàng. Với năng lực của một hệ quản trị cơ sở dữ liệu, thông tin sẽ sẵn sàng và quá trình có được thông tin sẽ nhanh và dễ hơn.

- Chia sẻ dữ liệu: Dữ liệu của những người dùng khác nhau có thể được gộp với nhau và chia sẻ cho những người dùng được phép, cho phép toàn bộ người dùng có thể truy cập được khối lượng dữ liệu lớn hơn. Nhiều người dùng có thể truy cập cùng dữ liệu và có thể sử dụng nó theo các cách khác nhau, khi có thay đổi về dữ liệu thì những người dùng khác cũng sẽ có được dữ liệu cập nhật nhất. Hơn nữa, những dữ liệu đã tồn tại có thể được sử dụng với các cách mới, như tạo ra các loại báo cáo mới mà không cần tạo ra các tệp dữ liệu phụ thêm như trong trường hợp không sử dụng cơ sở dữ liệu.
- Cân bằng các yêu cầu có xung đột: Theo hướng tiếp cận cơ sở dữ liệu cho chức năng hoàn chỉnh của tổ chức, một người hoặc một nhóm tự chịu trách nhiệm về cơ sở dữ liệu, đặc biệt nếu cơ sở dữ liệu sẽ cung cấp cho nhiều người. Người này hoặc nhóm người này được gọi là Quản trị viên cơ sở dữ liệu (Database Administrator hoặc Database Administration – DBA). Bằng việc duy trì các nhu cầu tổng thể của toàn bộ tổ chức, DBA có thể xây dựng cơ sở dữ liệu theo cách để thỏa mãn cho cả tổ chức chứ không cho một nhóm cụ thể.
- Kiểm soát sự dư thừa: Xử lý với cơ sở dữ liệu, các dữ liệu trước kia được lưu trữ dưới dạng tệp phải được tích hợp vào một cơ sở dữ liệu, vì vậy những bản sao dữ liệu sẽ không tồn tại nữa. Loại bỏ sự dư thừa dữ liệu không chỉ tiết kiệm không gian mà còn làm cho các quá trình cập nhật trở nên đơn giản hơn. Dù loại bỏ dư thừa là lý tưởng, nhưng không phải lúc nào cũng thực hiện được. Thỉnh thoảng, vì lý do về hiệu năng, ta có thể chọn dư thừa với một mức độ nhất định. Tuy nhiên, thậm chí trong trường hợp này, ta vẫn phải kiểm soát chặt chẽ dư thừa để đạt được những lợi ích của cơ sở dữ liệu. Đó là lý do tốt nhất là ta kiểm soát dư thừa hơn là loại bỏ chúng.

- Duy trì tính nhất quán: Với dữ liệu của cùng một đối tượng, cần phải nhất quán khi lưu trữ ở những nơi khác nhau, thông thường hạn chế xảy ra vì đã loại bỏ dữ thừa khi ta sử dụng cơ sở dữ liệu.
- Tăng cường toàn vẹn: Ràng buộc toàn vẹn là luật bắt buộc phải tuân theo của dữ liệu trong cơ sở dữ liệu. Một cơ sở dữ liệu có tính toàn vẹn nếu nó thỏa mãn tất cả các ràng buộc toàn vẹn đã được xác lập. Một hệ quản trị cơ sở dữ liệu tốt có thể cung cấp cơ hội cho người sử dụng kết hợp các ràng buộc toàn vẹn này khi thiết kế cơ sở dữ liệu. Khi đó hệ quản trị cơ sở dữ liệu phải đảm bảo rằng những ràng buộc này không bị xung đột. Hệ quản trị cơ sở dữ liệu cũng sẽ không cho phép lưu những dữ liệu mà vi phạm các ràng buộc đã được định nghĩa.
- Mở rộng an ninh: An ninh là tránh người sử dụng không được phép truy cập vào cơ sở dữ liệu. Một hệ quản trị cơ sở dữ liệu có nhiều chức năng để giúp đảm bảo các thực thi các biện pháp an ninh. Hệ quản trị cơ sở dữ liệu có thể sử dụng tên người dùng và mật khẩu, nhóm người dùng và gán các quyền hạn khác nhau đối với các đối tượng trong cơ sở dữ liệu.
- Tăng năng suất: Hệ quản trị cơ sở dữ liệu cho phép lập trình viên có thể xây dựng các phần mềm để truy xuất dữ liệu, xử lý dữ liệu, thực hiện các thao tác cập nhật dữ liệu trong cơ sở dữ liệu, từ đó nâng cao năng suất lập trình. Một hệ quản trị cơ sở dữ liệu tốt có nhiều chức năng cho phép người sử dụng có thể truy xuất dữ liệu từ cơ sở dữ liệu mà không cần phải lập trình. Điều này làm tăng năng suất cho lập trình viên khi không phải viết các phần mềm phức tạp để thực hiện các công việc này, cũng như những người dùng không là lập trình viên có được các kết quả từ việc truy cập cơ sở dữ liệu mà không cần chờ đợi phần mềm được lập trình cho họ.
- Độc lập dữ liệu: Cấu trúc của cơ sở dữ liệu thường cần phải thay đổi. Ví dụ, sự thay đổi yêu cầu của người sử dụng có thể cần thêm trường vào thực thể, thay đổi các quan hệ hoặc một thay đổi nào đó để cải thiện hiệu năng. Một hệ quản trị cơ sở dữ liệu tốt cung cấp khả năng độc lập dữ liệu, đó là khả năng cho phép thay đổi cấu trúc của cơ sở dữ liệu mà không cần sửa đổi phần mềm có truy cập đến cơ sở dữ liệu đó.

Ngoài những đặc điểm trên, hệ quản trị cơ sở dữ liệu cho quản lý dữ liệu lâu dài, có khả năng quản lý và khai thác một khối lượng dữ liệu lớn, hỗ trợ trữ tượng hóa dữ liệu ở các mức khác nhau.



Hình 3. Hệ quản trị cơ sở dữ liệu hỗ trợ trữ tượng hóa ở các mức khác nhau

Ngoài những đặc điểm mang tính ưu điểm trên của việc sử dụng cơ sở dữ liệu và hệ quản trị cơ sở dữ liệu, có một số những nhược điểm sau:

- Kích thước file lớn hơn: Để hỗ trợ tất cả các chức năng phức tạp mà hệ quản trị cơ sở dữ liệu cung cấp cho người dùng, hệ quản trị cơ sở dữ liệu phải là một phần mềm lớn, chiếm nhiều dung lượng đĩa cũng như dung lượng bộ nhớ trong. Hơn nữa, vì tất cả các dữ liệu mà cơ sở dữ liệu quản lý nằm trong một hoặc một số tệp, do vậy nó yêu cầu dung lượng ổ đĩa và bộ nhớ trong lớn.
- Tăng độ phức tạp: Hệ quản trị cơ sở dữ liệu được coi là một sản phẩm phức tạp do độ phức tạp và nhiều các chức năng mà nó cung cấp cho người dùng. Người sử dụng hệ quản trị cơ sở dữ liệu phải học nhiều mới hiểu được các chức năng của hệ thống để có thể sử dụng được hết các lợi ích của nó. Trong việc thiết kế và xây dựng một hệ thống mới có sử dụng hệ quản trị cơ sở dữ liệu phải thực hiện nhiều lựa chọn, do vậy, sẽ nảy sinh trường hợp lựa chọn không đúng, đặc biệt là thiếu hiểu biết về hệ thống. Những lựa chọn sai lầm có thể dẫn đến thảm họa của cả hệ thống. Thiết kế cơ sở dữ liệu ảnh hưởng rất lớn đến sự thành công của việc sử dụng hệ quản trị cơ sở dữ liệu.
- Lỗi có tác động lớn hơn: Khi không sử dụng cơ sở dữ liệu, mỗi người sử dụng có một hệ thống hoàn chỉnh riêng biệt, khi xảy ra lỗi đối với người dùng nào thì chỉ ảnh hưởng đến người dùng đó và không ảnh hưởng đến người dùng

khác. Một khía cạnh khác, nếu nhiều người dùng chia sẻ cùng một cơ sở dữ liệu, có lỗi xảy ra đối với một người dùng mà làm phá hỏng cơ sở dữ liệu theo một cách nào đó thì sẽ ảnh hưởng đến tất cả người dùng.

- Khó khôi phục hơn: Một cơ sở dữ liệu phức tạp hơn một tệp đơn do vậy quá trình khôi phục khi có một sự cố xảy ra sẽ phức tạp hơn. Cụ thể khi cơ sở dữ liệu được nhiều người cập nhật đồng thời. Cơ sở dữ liệu cần phải được khôi phục lại ở thời điểm chưa lỗi, tất cả các thao tác cập nhật dữ liệu của người dùng từ thời điểm đó phải làm lại. Số lượng người dùng cập nhật dữ liệu càng lớn thì nhiệm vụ khôi phục càng phức tạp hơn.

1.1.2 Một số hệ quản trị cơ sở dữ liệu thông dụng

Trên thế giới, hiện có rất nhiều các hệ quản trị cơ sở dữ liệu khác nhau đang được sử dụng. Có các hệ quản trị cơ sở dữ liệu quan hệ, cơ sở dữ liệu hướng đối tượng, cơ sở dữ liệu phi cấu trúc, cơ sở dữ liệu lớn... hoặc tích hợp các mô hình cơ sở dữ liệu đó với nhau. Tương ứng với từng mô hình, chính sách bản quyền và giá thành sẽ khác nhau. Hình dưới đây là bảng xếp hạng các hệ quản trị cơ sở dữ liệu thông dụng vào tháng 4/2020 theo các tiêu chí được mô tả trong https://db-engines.com/en/ranking_definition. Trong đó, Microsoft SQL Server là một trong những hệ quản trị cơ sở dữ liệu thông dụng hàng đầu.

355 systems in ranking, April 2020									
Rank	DBMS			Database Model	Score			Apr 2020	Mar 2020
	Apr 2020	Mar 2020	Apr 2019		Apr 2020	Mar 2020	Apr 2019		
1.	1.	1.	1.	Oracle 	Relational, Multi-model 	1345.42	+4.78	+65.48	
2.	2.	2.	2.	MySQL 	Relational, Multi-model 	1268.35	+8.62	+53.21	
3.	3.	3.	3.	Microsoft SQL Server 	Relational, Multi-model 	1083.43	-14.43	+23.47	
4.	4.	4.	4.	PostgreSQL 	Relational, Multi-model 	509.86	-4.06	+31.14	
5.	5.	5.	5.	MongoDB 	Document, Multi-model 	438.43	+0.82	+36.45	
6.	6.	6.	6.	IBM Db2 	Relational, Multi-model 	165.63	+3.07	-10.42	
7.	7.	↑ 8.	8.	Elasticsearch 	Search engine, Multi-model 	148.91	-0.26	+2.91	
8.	8.	↓ 7.	7.	Redis 	Key-value, Multi-model 	144.81	-2.77	-1.57	
9.	↑ 10.	↑ 10.	10.	SQLite 	Relational	122.19	+0.24	-2.02	
10.	↓ 9.	↓ 9.	9.	Microsoft Access	Relational	121.92	-3.22	-22.73	
11.	11.	11.	11.	Cassandra 	Wide column	120.07	-0.88	-3.54	
12.	↑ 13.	12.	12.	MariaDB 	Relational, Multi-model 	89.90	+1.55	+4.67	

Nguồn: <https://db-engines.com/en/ranking>

a) Hệ quản trị cơ sở dữ liệu Oracle



Mô tả	Hệ quản trị cơ sở dữ liệu quan hệ được sử dụng rộng rãi
Mô hình cơ sở dữ liệu chính	Cơ sở dữ liệu quan hệ
Các mô hình cơ sở dữ liệu khác	Document store Graph DBMS RDF store
Nhà phát triển	Oracle
Năm ra mắt	1980
Phiên bản đến năm 2020	19c, February 2019
Bản quyền (License)	commercial
Nguồn ngữ phát triển	C and C++
Hệ điều hành máy chủ	Systems, AIX, HP-UX, Linux, OS X, Solaris, Windows, z/OS
Lược đồ dữ liệu	Có
Dùng lệnh SQL	Có
APIs và các phương thức truy cập khác	JDBC, ODBC, ODP.NET, Oracle Call Interface (OCI)
Hỗ trợ các ngôn ngữ lập trình	C, C#, C++, Clojure, Cobol, Delphi, Eiffel, Erlang, Fortran, Groovy, Haskell, Java, JavaScript, Lisp, Objective C, OCaml, Perl, PHP, Python, R, Ruby, Scala, Tcl, Visual Basic
Lập trình phía máy chủ (Server-side scripts)	PL/SQL

b) Hệ quản trị cơ sở dữ liệu MySQL



Mô tả	Hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở được sử dụng rộng rãi
Mô hình cơ sở dữ liệu chính	Cơ sở dữ liệu quan hệ

Các mô hình cơ sở dữ liệu khác	Document store
Nhà phát triển	Oracle
Năm ra mắt	1995
Phiên bản đến năm 2020	8.0.19, 2020
Bản quyền (License)	Open Source, commercial
Nguồn ngữ phát triển	C and C++
Hệ điều hành máy chủ	FreeBSD, Linux, OS X, Solaris, Windows
Lược đồ dữ liệu	Có
Dùng lệnh SQL	Có
APIs và các phương thức truy cập khác	ADO.NET, JDBC, ODBC, Proprietary native API
Hỗ trợ các ngôn ngữ lập trình	Ada, C, C#, C++, D, Delphi, Eiffel, Erlang, Haskell, Java, JavaScript (Node.js), Objective-C, OCaml, Perl, PHP, Python, Ruby, Scheme, Tcl
Lập trình phía máy chủ (Server-side scripts)	Có

c) Hệ quản trị cơ sở dữ liệu PostgreSQL



Mô tả	Hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở được sử dụng rộng rãi
Mô hình cơ sở dữ liệu chính	Cơ sở dữ liệu quan hệ
Các mô hình cơ sở dữ liệu khác	Document store
Nhà phát triển	PostgreSQL Global Development Group
Năm ra mắt	1989
Phiên bản đến năm 2020	12.2, February 2020
Bản quyền (License)	Open Source
Nguồn ngữ phát triển	C

Hệ điều hành máy chủ	FreeBSD, HP-UX, Linux, NetBSD, OpenBSD, OS X, Solaris, Unix, Windows
Lược đồ dữ liệu	Có
Dùng lệnh SQL	Có
APIs và các phương thức truy cập khác	ADO.NET, JDBC, native C library ODBC, streaming API for large objects
Hỗ trợ các ngôn ngữ lập trình	.Net, C, C++, Delphi, Java, JavaScript (Node.js), Perl, PHP, Python, Tcl
Lập trình phía máy chủ (Server-side scripts)	sử dụng các ngôn ngữ thông dụng như Perl, Python, ..

d) Hệ quản trị cơ sở dữ liệu MongoDB



Mô tả	Một trong những kho lưu trữ tài liệu phổ biến nhất hiện có cả dưới dạng dịch vụ đám mây được quản lý hoàn chỉnh và để triển khai trên cơ sở hạ tầng tự quản lý
Mô hình cơ sở dữ liệu chính	Lưu trữ tài liệu
Các mô hình cơ sở dữ liệu khác	Search engine
Nhà phát triển	MongoDB, Inc
Năm ra mắt	2009
Phiên bản đến năm 2020	4.2.5, March 2020
Bản quyền (License)	Open Source, commercial
Nguồn ngữ phát triển	C++
Hệ điều hành máy chủ	Linux, OS X, Solaris, Windows
Lược đồ dữ liệu	Không có lược đồ
Dùng lệnh SQL	Read-only SQL queries via the MongoDB Connector for BI
APIs và các phương thức truy cập khác	Giao thức độc quyền sử dụng JSON
Hỗ trợ các ngôn ngữ lập trình	Actionscript, C, C#, C++, Clojure, ColdFusion, D, Dart, Delphi, Erlang, Go, Groovy, Haskell, Java, JavaScript, Lisp,

	Lua, MatLab, Perl, PHP, PowerShell, Prolog, Python, R, Ruby, Scala, Smalltalk
Lập trình phía máy chủ (Server-side scripts)	JavaScript

e) Hệ quản trị cơ sở dữ liệu DB2



Mô tả	Thường sử dụng trong môi trường IBM, có 2 phiên bản khác cho Windows/Linux
Mô hình cơ sở dữ liệu chính	Cơ sở dữ liệu quan hệ
Các mô hình cơ sở dữ liệu khác	Document store RDF store
Nhà phát triển	IBM
Năm ra mắt	1983
Phiên bản đến năm 2020	12.1, October 2016
Bản quyền (License)	commercial, bản miễn phí bị hạn chế
Nguồn ngữ phát triển	C and C++
Hệ điều hành máy chủ	AIX, HP-UX, Linux, Solaris, Windows, z/OS
Lược đồ dữ liệu	Có
Dùng lệnh SQL	Có
APIs và các phương thức truy cập khác	ADO.NET, JDBC, JSON style queries, ODBC, Xquery
Hỗ trợ các ngôn ngữ lập trình	C, C#, C++, Cobol, Delphi, Fortran, Java, Perl, PHP, Python, Ruby, Visual Basic
Lập trình phía máy chủ (Server-side scripts)	Có

f) Hệ quản trị cơ sở dữ liệu Redis



Mô tả	Lưu trữ cấu trúc dữ liệu trong bộ nhớ (In-memory) và sử dụng như cơ sở dữ liệu, cache và message broker
Mô hình cơ sở dữ liệu chính	Lưu trữ dạng Key-value
Các mô hình cơ sở dữ liệu khác	Document store Graph DBMS Search engine Time Series DBMS
Nhà phát triển	Salvatore Sanfilippo
Năm ra mắt	2009
Phiên bản đến năm 2020	5.0.8, March 2020
Bản quyền (License)	Open Source, commercial
Nguồn ngữ phát triển	C
Hệ điều hành máy chủ	BSD, Linux, OS X, Windows
Lược đồ dữ liệu	Không có lược đồ
Dùng lệnh SQL	Không
APIs và các phương thức truy cập khác	Giao thức độc quyền
Hỗ trợ các ngôn ngữ lập trình	C, C#, C++, Clojure, Crystal, D, Dart, Elixir, Erlang, Fancy, Go, Haskell, Haxe, Java, JavaScript (Node.js), Lisp, Lua, MatLab, Objective-C, OCaml, Pascal, Perl, PHP, Prolog, Pure Data, Python, R, Rebol, Ruby, Rust, Scala, Scheme, Smalltalk, Swift, Tcl, Visual Basic
Lập trình phía máy chủ (Server-side scripts)	Lua

g) Hệ quản trị cơ sở dữ liệu Cassandra



Mô tả	Lưu trữ các trường dữ liệu kích thước lớn dựa trên ý tưởng của BigTable và DynamoDB
Mô hình cơ sở dữ liệu chính	Lưu trữ các trường dữ liệu kích thước lớn
Các mô hình cơ sở dữ liệu khác	Không
Nhà phát triển	Apache Software Foundation
Năm ra mắt	2008
Phiên bản đến năm 2020	3.11.6, February 2020
Bản quyền (License)	Open Source
Nguồn ngữ phát triển	Java
Hệ điều hành máy chủ	BSD, Linux, OS X, Windows
Lược đồ dữ liệu	Không có lược đồ
Dùng lệnh SQL	SQL-like SELECT, DML and DDL statements (CQL)
APIs và các phương thức truy cập khác	Giao thức độc quyền, Thrift
Hỗ trợ các ngôn ngữ lập trình	C#, C++, Clojure, Erlang, Go, Haskell, Java, JavaScript, Perl, PHP, Python, Ruby, Scala
Lập trình phía máy chủ (Server-side scripts)	Không

1.2. GIỚI THIỆU MICROSOFT SQL SERVER

1.2.1 Giới thiệu về Microsoft SQL Server



Mô tả	Hệ quản trị cơ sở dữ liệu quan hệ của MicroSoft
Mô hình cơ sở dữ liệu chính	Cơ sở dữ liệu quan hệ
Các mô hình cơ sở dữ liệu khác	Document store Graph DBMS
Nhà phát triển	Microsoft
Năm ra mắt	1989

Phiên bản đến năm 2020	SQL Server 2019, November 2019
Bản quyền (License)	commercial, bản miễn phí bị hạn chế
Nguồn ngữ phát triển	C++
Hệ điều hành máy chủ	Linux, Windows
Lược đồ dữ liệu	Có
Dùng lệnh SQL	Có
APIs và các phương thức truy cập khác	ADO.NET, JDBC, ODBC, OLE DB, Tabular Data Stream (TDS)
Hỗ trợ các ngôn ngữ lập trình	C#, C++, Delphi, Go, Java, JavaScript (Node.js), PHP, Python, R, Ruby, Visual Basic
Lập trình phía máy chủ (Server-side scripts)	Transact SQL, .NET languages, R, Python and (with SQL Server 2019) Java

MicroSoft SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System (RDBMS)) sử dụng câu lệnh SQL và Transact-SQL để trao đổi dữ liệu giữa máy Client và máy cài SQL Server. Một RDBMS bao gồm databases, database engine và các ứng dụng dùng để quản lý dữ liệu và các bộ phận khác nhau trong RDBMS. SQL Server được phát triển và tiếp thị bởi Microsoft. SQL Server hoạt động độc quyền trên môi trường Windows trong hơn 20 năm. Năm 2016, Microsoft đã cung cấp phiên bản trên Linux. SQL Server 2017 ra mắt vào tháng 10 năm 2016 chạy trên cả Windows và Linux, SQL Server 2019 sẽ ra mắt trong năm 2019.

SQL Server 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. SQL Server có thể kết hợp với các server khác như Microsoft Internet Information Server (IIS), E-Commerce Server, Proxy Server....

MicroSoft SQL Server hỗ trợ đầy đủ cho ngôn ngữ truy vấn dữ liệu chuẩn Structured Query Language (SQL) để truy xuất các đối tượng trong SQL server, đồng thời có ngôn ngữ riêng là T-SQL: Transact-SQL là ngôn ngữ SQL mở rộng dựa trên SQL chuẩn của ISO (International Organization for Standardization) và ANSI (American National Standards Institute) được sử dụng trong MS SQL Server. MicroSoft SQL Server cung cấp bộ công cụ phong phú cho phép người dùng tạo lập

các cơ sở dữ liệu, các đối tượng trong cơ sở dữ liệu, khai thác dữ liệu, lập trình và quản trị hệ thống một cách thuận lợi.

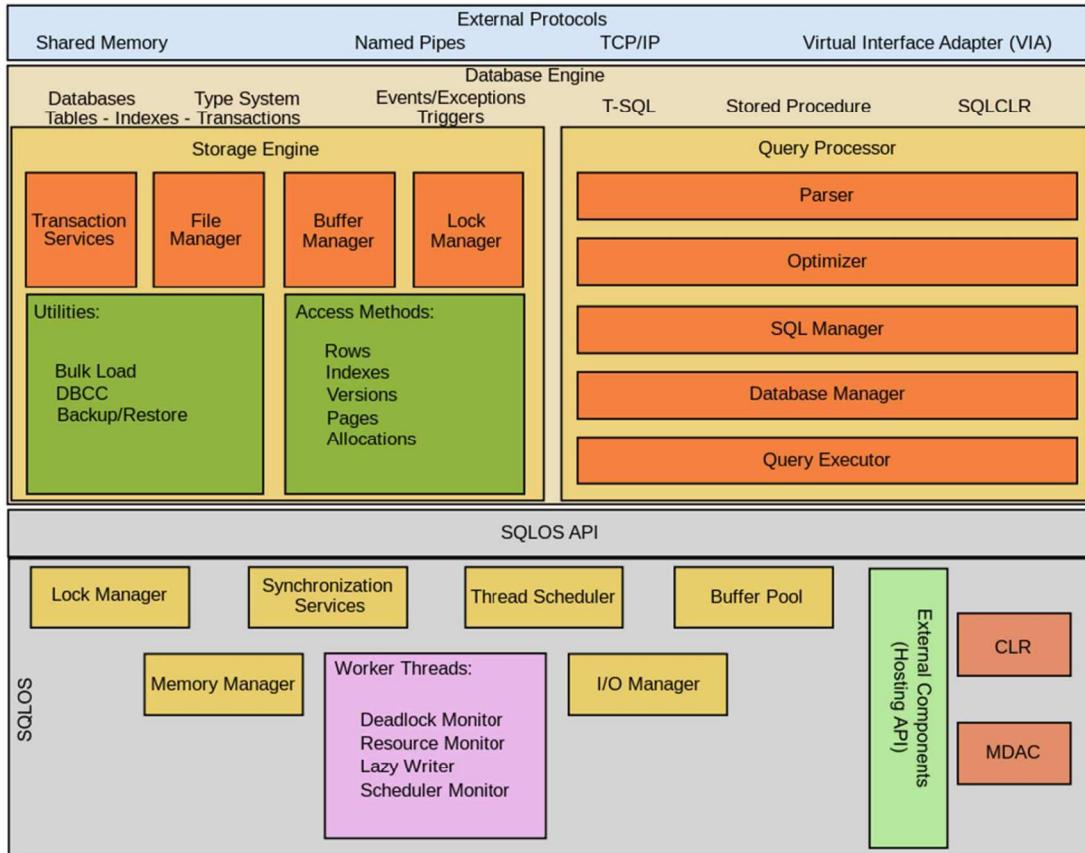
Các phiên bản MicroSoft SQL Server:

- **Enterprise:** chứa tất cả cá đặc điểm nổi bật của SQL Server, bao gồm nhân của bộ máy cơ sở dữ liệu và các dịch vụ đi kèm cùng với các công cụ cho tạo và quản lý phân cụm SQL Server (SQL Server Cluster). Nó có thể quản lý các cơ sở dữ liệu lớn tới 524 petabytes và đánh địa chỉ 12 terabytes bộ nhớ và hỗ trợ tới 640 bộ vi xử lý (các Core của CPU).
- **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 (Advanced features) khác, phiên bản này có thể chạy tốt trên hệ thống lên đến 4 CPU và 2 GB RAM.
- **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 phiên bản sử dụng cho phát triển và kiểm tra ứng dụng. Phiên bản này phù hợp cho các cá nhân, tổ chức xây dựng và kiểm tra ứng dụng. Phiên bản này được cung cấp miễn phí.
- **Workgroup:** phiên bản SQL Server Workgroup bao gồm chức năng lõi cơ sở dữ liệu nhưng không có các dịch vụ đi kèm. Chú ý phiên bản này không còn tồn tại ở SQL Server 2012.
- **Express:** SQL Server Express dễ sử dụng và quản trị cơ sở dữ liệu đơn giản. Được tích hợp với Microsoft Visual Studio, nên dễ dàng để phát triển các ứng dụng dữ liệu, an toàn trong lưu trữ, và nhanh chóng triển khai. SQL Server Express là phiên bản miễn phí, không giới hạn về số cơ sở dữ liệu hoặc người sử dụng, nhưng nó chỉ dùng cho 1 bộ vi xử lý với 1 GB bộ nhớ và 10 GB file cơ sở dữ liệu. SQL Server Express là lựa chọn tốt cho những người dùng chỉ cần một phiên bản SQL Server nhỏ gọn, dùng trên máy chủ có cấu hình thấp, những nhà phát triển ứng dụng không chuyên hay những người yêu thích xây dựng các ứng dụng nhỏ.

1.2.2 Kiến trúc và thành phần của Microsoft SQL Server

Kiến trúc của MicroSoft SQL Server bao gồm hai khối chức năng chính là Database Engine và SQLOS:

- **SQL Server Database Engine:** công cụ này kiểm soát việc lưu trữ, xử lý và bảo mật dữ liệu. Thành phần này bao gồm một công cụ quan hệ có chức năng xử lý các lệnh và truy vấn, một công cụ lưu trữ quản lý các tệp, bảng, trang, chỉ mục, bộ đệm dữ liệu và giao dịch cơ sở dữ liệu. Các thủ tục, trigger, job và các đối tượng khác trong cơ sở dữ liệu cũng được Database Engine khởi tạo và xử lý.
- **SQLOS:** là viết tắt của SQL Server Operating System - Hệ điều hành SQL Server là lớp phía dưới Database Engine. Hệ điều hành xử lý các chức năng ở cấp độ thấp hơn như quản lý bộ nhớ và I/O, lịch nhiệm vụ và khóa dữ liệu để tránh các xung đột xảy ra khi có các thao tác cập nhật dữ liệu hoặc đối tượng trong SQL Server. Một lớp giao diện mạng nằm trên lớp Database Engine và sử dụng một giao thức gọi là Tabular Data Stream của Microsoft để các yêu cầu và phản hồi tương tác với máy chủ cơ sở dữ liệu thuận tiện hơn. Ở cấp độ người sử dụng, người quản trị cơ sở dữ liệu và nhà phát triển SQL Server có thể viết các câu lệnh T-SQL để xây dựng và sửa đổi cấu trúc cơ sở dữ liệu, thao tác, thiết lập các bảo vệ, sao lưu cơ sở dữ liệu, cùng với nhiều nhiệm vụ khác.



Hình 4. Mô hình kiến trúc của MicroSoft SQL Server

Dưới góc độ phát triển ứng dụng, khai thác cơ sở dữ liệu, MicroSoft SQL Server bao gồm các thành phần sau đây:

- **Database Engine:** đây là lõi của SQL Server, cho phép tạo lập và quản lý cơ sở dữ liệu và các đối tượng trong cơ sở dữ liệu... như đã mô tả ở phần trên.
- **Integration Services:** là tập hợp các đối tượng lập trình và các công cụ đồ họa cho việc sao chép, di chuyển và chuyển đổi dữ liệu. Trong thực tế, khi ứng dụng và triển khai các hệ thống ứng dụng cho các môi trường phức tạp của doanh nghiệp, tổ chức thì nhu cầu chuyển đổi dữ liệu từ các dữ liệu được lưu trữ trong các hệ thống khác như Oracle, SQL Server, DB2, Microsoft Access,... để thao tác và lưu trữ trong SQL Server và ngược lại sẽ rất lớn và thường xuyên. Integration Services sẽ giúp giải quyết được công việc này dễ dàng và thuận lợi.

- **Analysis Services:** Đây là một dịch vụ phân tích dữ liệu rất hay của Microsoft. Dữ liệu khi được lưu trữ vào trong cơ sở dữ liệu mà không thể lấy được những thông tin hữu ích thì coi như không có ý nghĩa gì. Chính vì thế, công cụ này ra đời giúp ta trong việc phân tích dữ liệu một cách hiệu quả và dễ dàng bằng cách dùng kỹ thuật khai thác dữ liệu và phân tích dữ liệu. Microsoft hiện nay đã có thêm các tính năng quản lý dữ liệu đa dạng, tính năng dành cho doanh nghiệp (Business Intelligence – BI), và các công cụ phân tích SQL Server. Bên cạnh các dịch vụ Machine Learning mới được tích hợp lần đầu tiên trong phiên bản SQL Server 2016, các dịch vụ phân tích dữ liệu bao gồm SQL Server Analysis Services, công cụ phân tích xử lý dữ liệu sử dụng trong BI, các ứng dụng trực quan hóa dữ liệu và các dịch vụ SQL Server Reporting, hỗ trợ tạo và phân phối các báo cáo BI.
- **Notification Services:** Dịch vụ thông báo này là nền tảng cho sự phát triển và triển khai các ứng dụng soạn và gửi thông báo. Ngoài ra, dịch vụ này còn có chức năng gửi thông báo theo định thời đến hàng ngàn người đăng ký sử dụng trên nhiều loại thiết bị khác nhau.
- **Reporting Services:** là một công cụ tạo, quản lý và triển khai báo cáo bao gồm: server và client. Ngoài ra, nó còn là nền tảng cho việc phát triển và xây dựng các ứng dụng báo cáo.
- **Full Text Search Service:** là một thành phần đặc biệt trong việc truy vấn và đánh chỉ mục dữ liệu văn bản không cấu trúc được lưu trữ trong các cơ sở dữ liệu SQL Server.
- **Service Broker:** là một môi trường lập trình cho việc tạo ra các ứng dụng trong việc nhảy qua các Instance của SQL Server.

1.3. CÀI ĐẶT MICROSOFT SQL SERVER

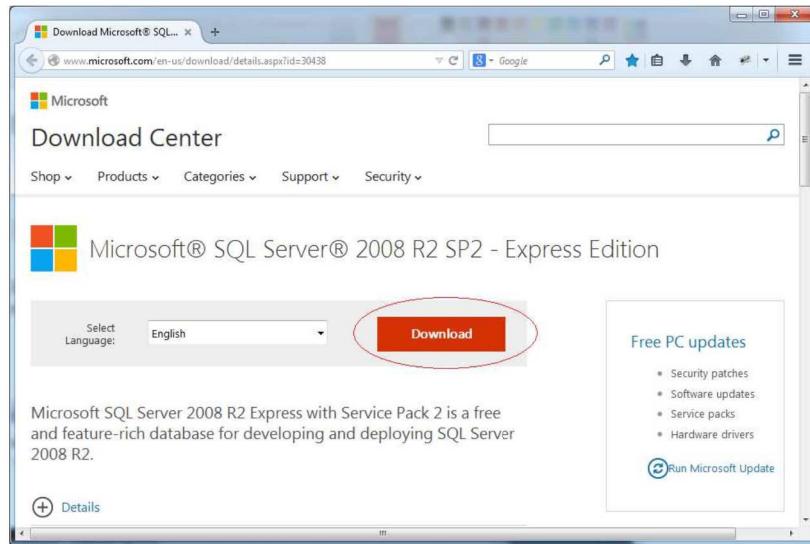
Để thực hiện cài đặt, ta có thể tải bản cài đặt MicroSoft SQL Server tại địa chỉ:

Phiên bản 2008: <http://www.microsoft.com/en-us/download/details.aspx?id=30438>

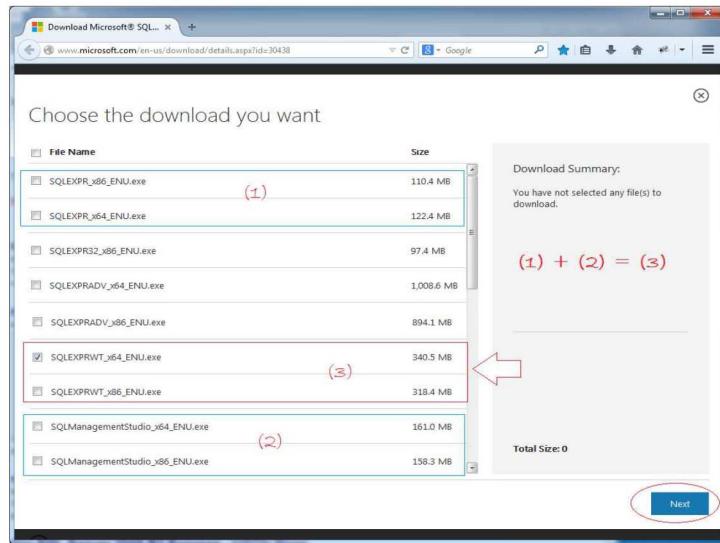
Phiên bản 2014: <http://www.microsoft.com/en-us/download/details.aspx?id=42299>

Sau đây là các bước để thực hiện cài đặt MicroSoft SQL Server phiên bản 2008, đối với các phiên bản khác, các bước thực hiện cài đặt cũng tương tự như khi cài đặt phiên bản này.

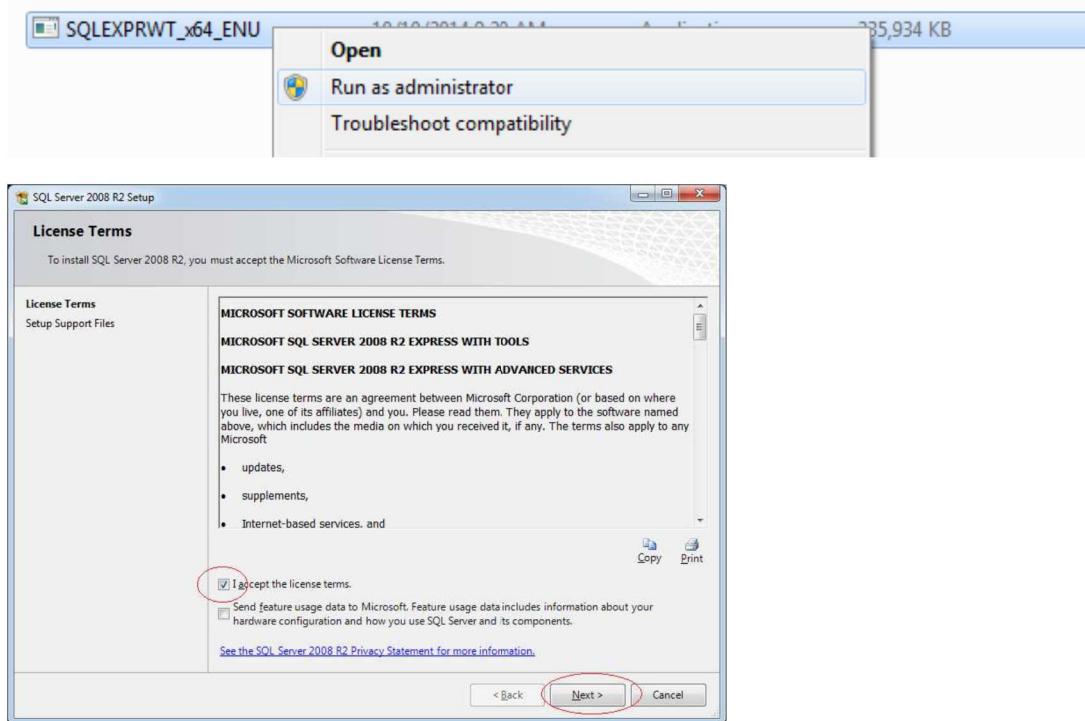
Bước 1: Tải file cài đặt tại website của MicroSoft



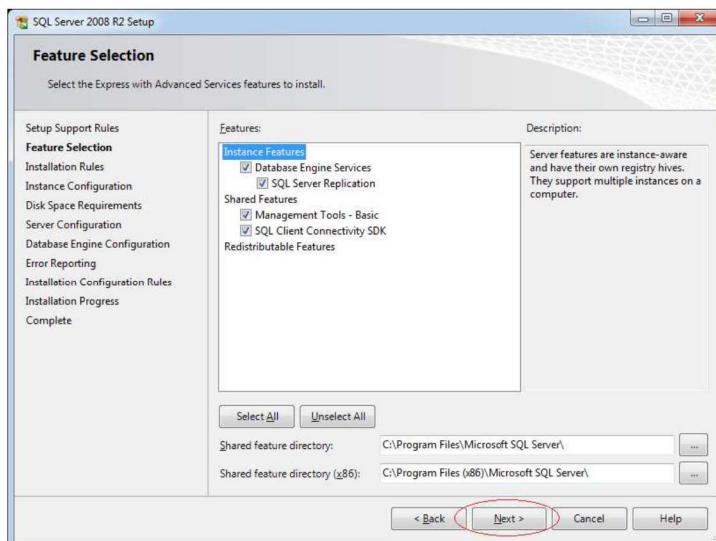
Để cài đặt cả phần lõi của MicroSoft SQL Server và công cụ trực quan để khai thác và quản trị các cơ sở dữ liệu, cần tải các file trong khung (1) và khung (2), hoặc có thể tải file tổng hợp tại khung (3) trong hình dưới đây.



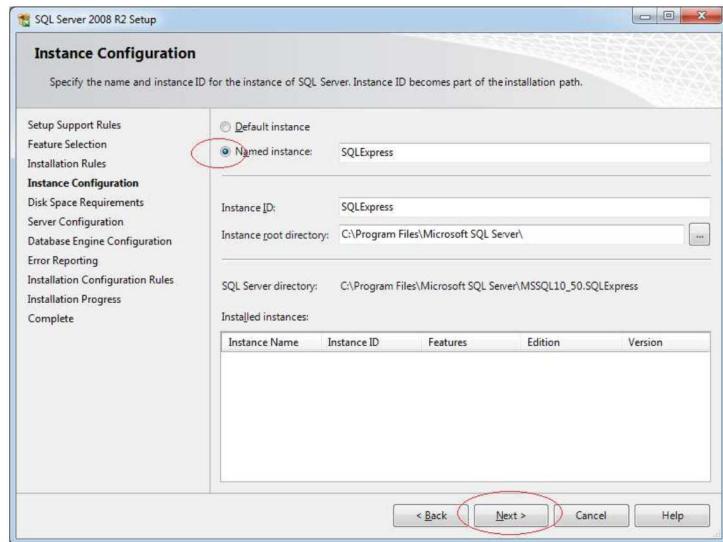
Bước 2: Thực thi chương trình cài đặt, cần thiết phải thực thi với quyền người dùng cao nhất của Server (Administrator).



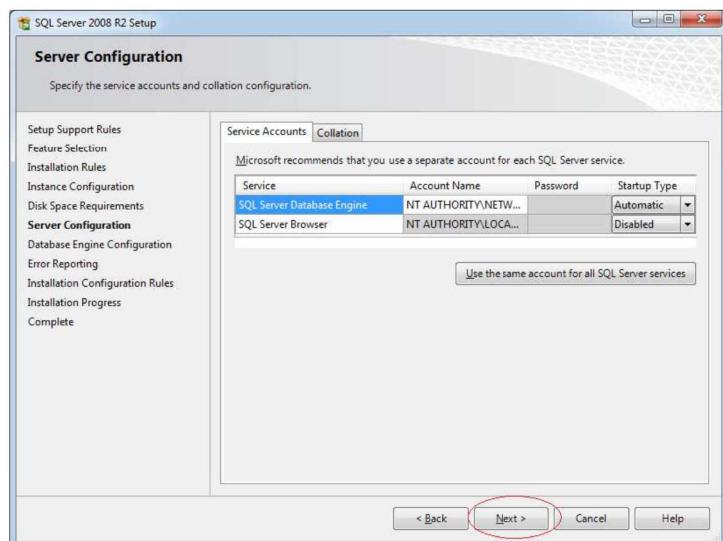
Chọn tất cả các tính năng



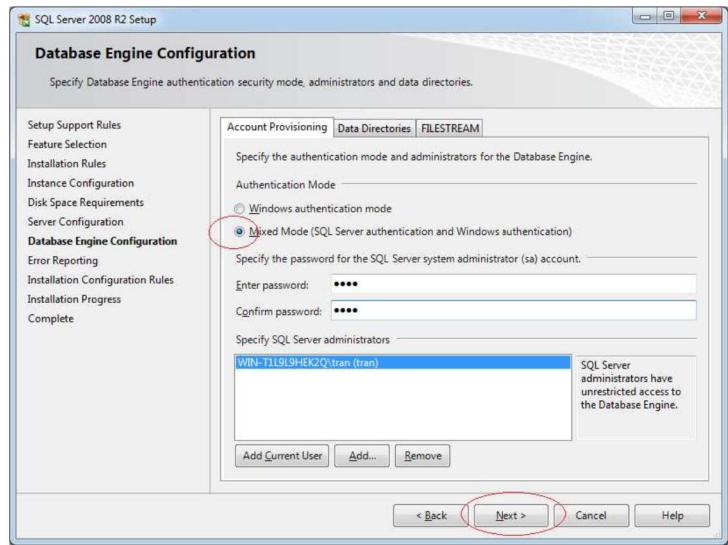
Đặt tên cho máy chủ (instance), trong có thẻ chọn tên ngầm định, khi đó sẽ là tên của máy hiện tại.



Chọn account của người dùng cho các dịch vụ SQL Server

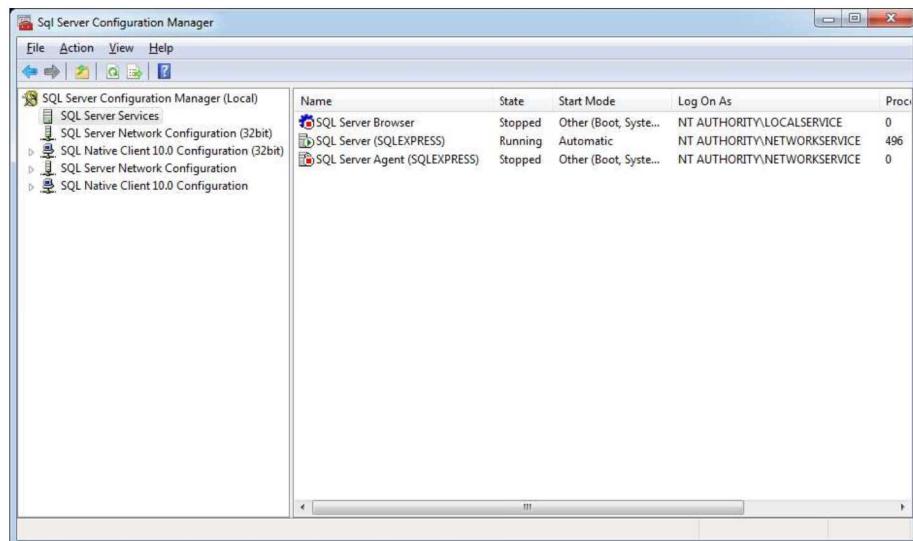


MicroSoft SQL Server cho phép xác thực người dùng khi đăng nhập bằng hai cách là sử dụng Username/Password của Windows hoặc kết hợp cả hai phương thức với Username/Password của SQL Server. Ta lựa chọn chế độ "Mix Mode" để sử dụng phương thức đăng nhập kết hợp.

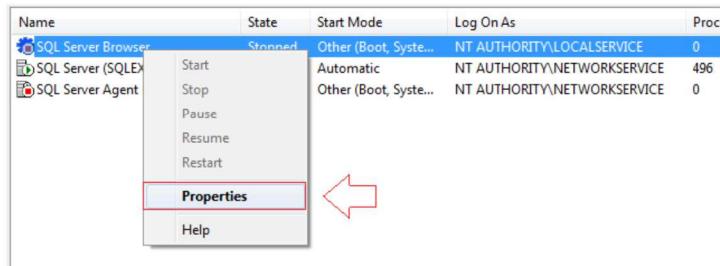


Kết thúc bước này, ta chọn Next cho đến khi quá trình cài đặt kết thúc.

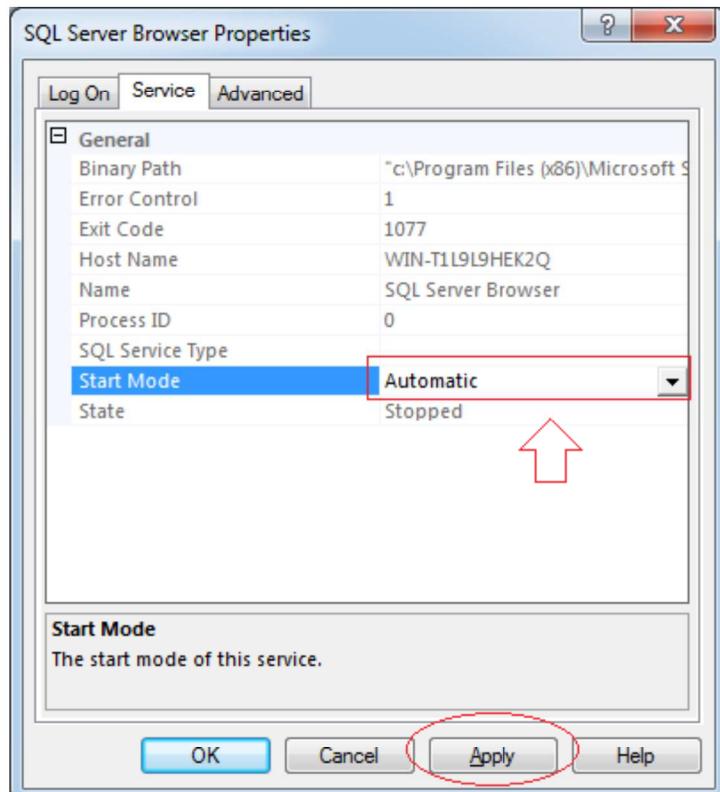
Bước 3: Cấu hình cho MicroSoft SQL Server bằng công cụ MicroSoft SQL Server Configuration Manager. Chức năng này cho phép bạn cấu hình để có thể từ một máy tính khác truy cập vào SQL Server thông qua IP hoặc Server name.



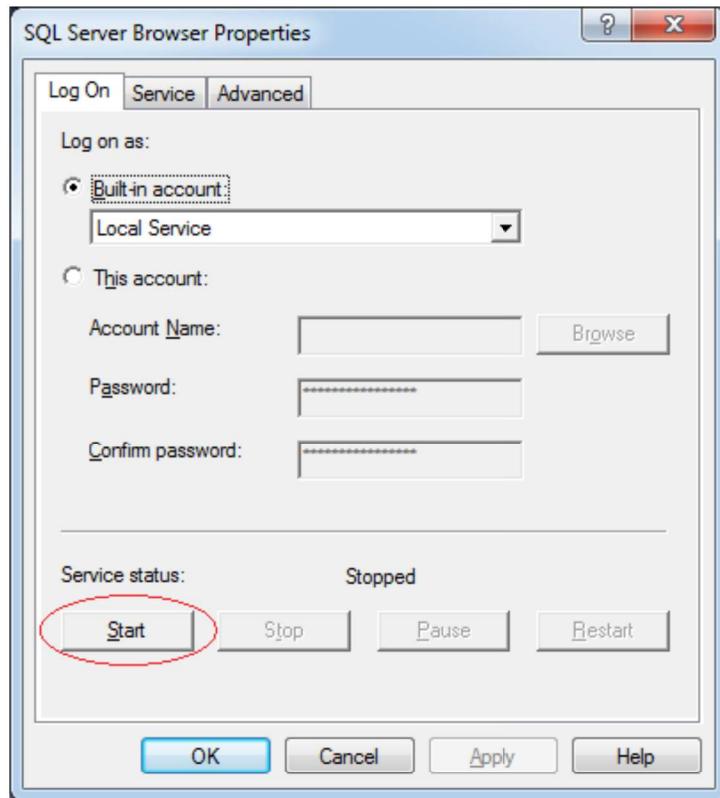
Nhấn phím phải chuột lên SQL Server Browser, chọn Properties để bật Service này lên



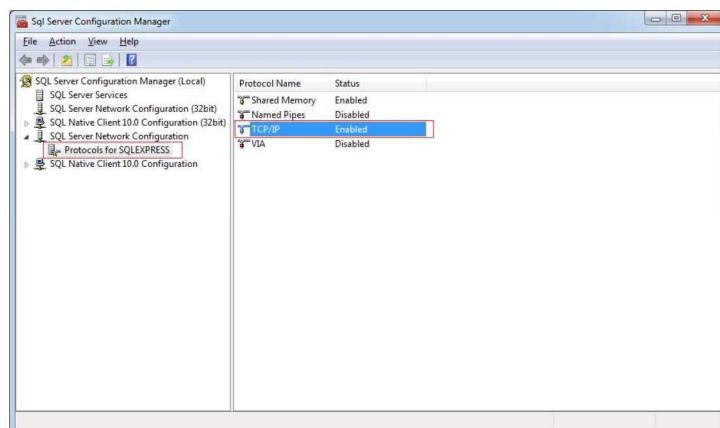
Chọn chế độ bật dịch vụ tự động

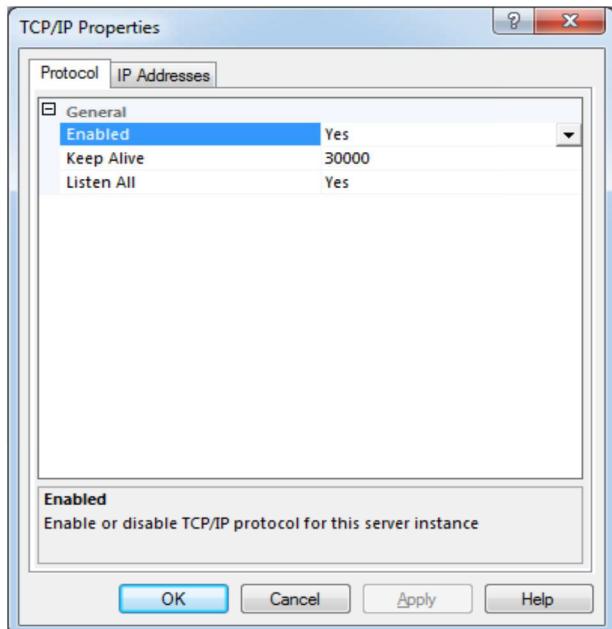


Chọn account của hệ thống, sau đó nhấn nút Start để bật dịch vụ, nhấn Ok để kết thúc.

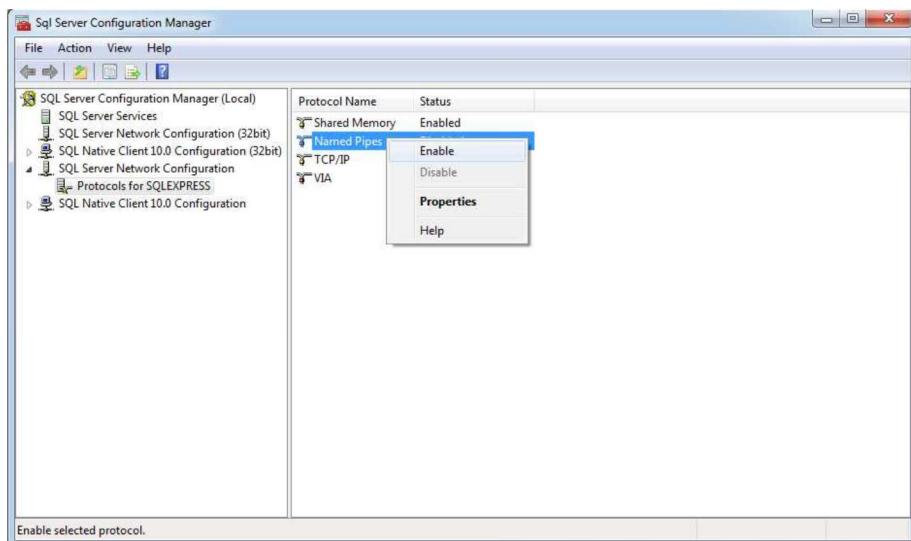


Sau đó, bật TPC/IP cho phép máy tính khác kết nối vào SQL Server thông qua IP

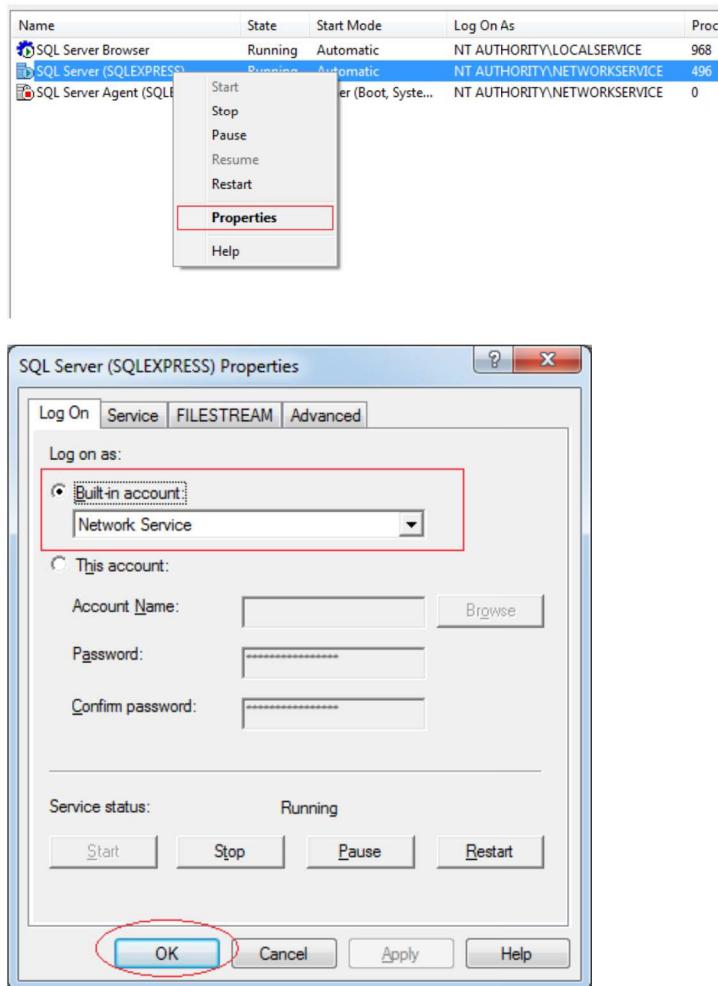




Trong trường hợp ta muốn cho máy tính khác truy cập vào MicroSoft SQL Server qua tên của máy chủ, ta bật chế độ cho phép Named pipes



Tiếp theo, chọn c.áu hình để **SQL Server** chạy dưới chế độ **Network Service**. Sau khi chọn các thông số như hình dưới, ta khởi động lại dịch vụ bằng cách chọn Restart



1.4. CÁC CÔNG CỤ THÔNG DỤNG TRONG MICROSOFT SQL SERVER

1.4.1 MicroSoft SQL Server Management Studio

MicroSoft SQL Server Management Studio là một công cụ trực quan để quản lý SQL Server. Với SQL Server Management Studio ta có thể thực hiện các tương tác với database trên giao diện người dùng hoặc bằng câu lệnh. SQL Server Management Studio được thiết kế thân thiện và dễ sử dụng.

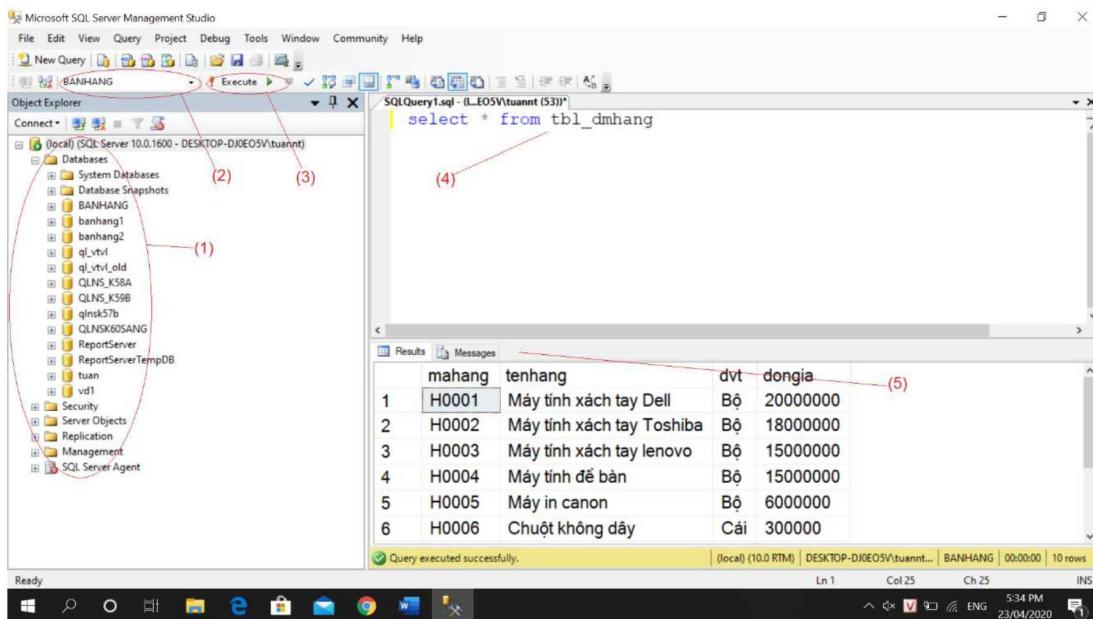
MicroSoft SQL Server Management Studio có những chức năng để thực hiện các nhiệm vụ chính sau đây:

- Quản lý hoạt động của dịch vụ lỗi của MicroSoft SQL Server: bao gồm bật/tắt, tạm dừng hoạt động của Server, quản lý các kết nối của Server tới các Server khác...
- Quản lý các cơ sở dữ liệu và các đối tượng trong cơ sở dữ liệu: quản lý các cơ sở dữ liệu, các đối tượng trong cơ sở dữ liệu như bảng, view, trigger, thủ tục, hàm...
- Quản lý người dùng và bảo mật: tạo, quản lý, phân quyền cho các người sử dụng Server (login) và người sử dụng cơ sở dữ liệu...
- Quản lý Server và các đối tượng của Server: cho phép quản lý hoạt động của Server và các thành phần trong Server, ghi lại tình trạng hoạt động của Server (log) và quản lý các đối tượng của Server như trigger, các thiết bị lưu trữ...
- Quản lý dịch vụ tự động hóa: quản lý các hoạt động của dịch vụ SQL Server Agent để tạo và thực thi các Job, cảnh báo... Cho phép bật/tắt/tạm dừng dịch vụ này.
- Môi trường khai thác dữ liệu trực tiếp và lập trình: Cung cấp giao diện để sử dụng các câu lệnh SQL, T-SQL để khai thác trực tiếp dữ liệu. Môi trường lập trình và gỡ lỗi chương trình (debug). Cung cấp các tiện ích để sinh mã lệnh tự động cho các đối tượng do MicroSoft SQL Server quản lý.

Để sử dụng MicroSoft SQL Server Management Studio, sau khi chạy công cụ, người dùng phải thực hiện kết nối đến máy chủ cơ sở dữ liệu và đăng nhập hệ thống. Người dùng có thể truy cập đến máy chủ cục bộ hoặc máy chủ từ xa bằng tên máy chủ hoặc địa chỉ IP của máy chủ.



Sau khi đăng nhập thành công vào SQL Server, MicroSoft SQL Server Management Studio sẽ hiển thị giao diện chính cho phép thực hiện các thao tác của người dùng bằng các công cụ có sẵn, sử dụng câu lệnh SQL hoặc lập trình.



Một số vị trí chủ yếu trên màn hình giao diện ngoài thanh menu và các biểu tượng thông dụng:

- Vị trí (1): Cây quản lý các đối tượng trong SQL Server và các dịch vụ của SQL Server. Nhấn các nút (+) hoặc (-) để mở/thu gọn các chi tiết của từng nhánh cây. Nhấn phím phải chuột lên từng nhánh cây để mở các menu nhanh thao

tác cho từng loại đối tượng trong nhánh cây (chi tiết những thao tác chủ yếu sẽ được giới thiệu trong các phần sau của tài liệu).

- Vị trí (2): Thể hiện và cho phép chọn cơ sở dữ liệu để làm việc. Khi thực hiện lệnh chọn cơ sở dữ liệu, tên cơ sở dữ liệu tại mục này sẽ tự động thay đổi theo.
- Vị trí (3): Cho phép thực hiện (Execute hoặc nhấn phím F5) các lệnh SQL, T-SQL... được chọn trong khung cửa sổ lệnh hoặc thực hiện gỡ lỗi chương trình (Debug hoặc nhấn tổ hợp phím Alt+F5).
- Vị trí (4): Nơi nhập các lệnh SQL, T-SQL hoặc lập trình trong SQL Server. Khi muốn thực hiện một hoặc nhiều lệnh, cần lựa chọn (bôi đen) các lệnh đó trong khung cửa sổ và chọn chức năng thực hiện lệnh. Trong trường hợp không chọn các lệnh cần thực hiện, SQL Server sẽ tự thực hiện tất cả các lệnh có trong khung cửa sổ này. Nếu trong quá trình thực hiện lệnh phát sinh lỗi, mã lỗi và các thông tin mô tả lỗi, dòng phát sinh lỗi tính trong các lệnh được chọn sẽ hiển thị tại Vị trí (5).
- Vị trí (5): Là vùng hiển thị kết quả của các lệnh đã thực hiện hoặc hiển thị các thông tin liên quan đến lỗi. Trong mục Results sẽ thể hiện các kết quả liên quan đến các lệnh thể hiện dữ liệu như SELECT, mục Messages sẽ thể hiện dữ liệu dạng text là kết quả của lệnh PRINT, các thông báo hoặc thông báo lỗi. Trong trường hợp thông báo lỗi, có thể nhấn đúp chuột lên thông báo lỗi để con trỏ chuyển về dòng lệnh có lỗi ở Vị trí (4).

Chú ý:

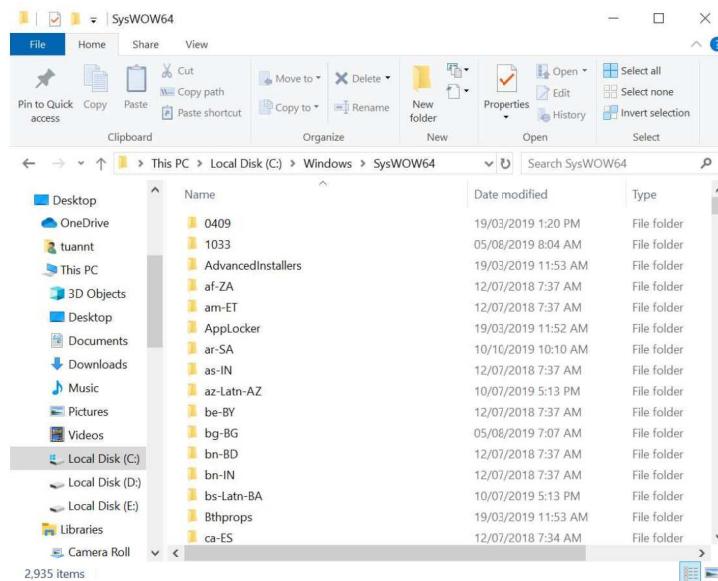
- + Trong các Vị trí (4), (5) có thể lưu lại, in các mã lệnh hoặc các kết quả.
- + Các chức năng chi tiết đối với từng loại đối tượng và dịch vụ trong SQL Server thực hiện thông qua MicroSoft SQL Server Management Studio sẽ được giới thiệu trong các nội dung của tài liệu.
- + Việc thực hiện các tính năng trên giao diện của MicroSoft SQL Server Management Studio cũng có thể sinh ra các mã lệnh tự động phục vụ cho lập trình hoặc thao tác qua cửa sổ lệnh sau này.

1.4.2 MicroSoft SQL Server Configuration Manager

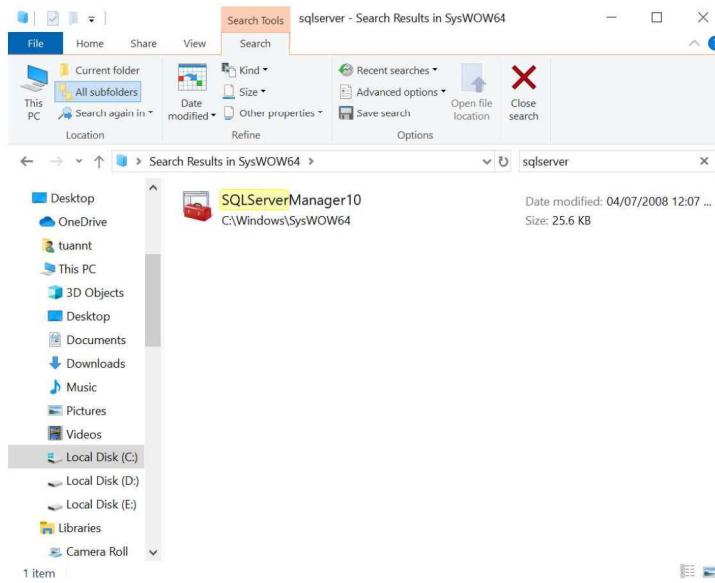
Công cụ MicroSoft SQL Server Configuration Manager cho phép quản trị cấu hình cơ sở cho các dịch vụ SQL Server (SQL Server services), các giao thức server (server protocols), các giao thức client (client protocols) và các bí danh client (client aliases). Các thao tác quản trị, cấu hình cho MicroSoft SQL Server đã được trình bày trong mục **Cài đặt MicroSoft SQL Server**.

Trong trường hợp sau khi cài đặt MicroSoft SQL Server, không tìm thấy MicroSoft SQL Server Configuration Manager trong Start của Windows, thực hiện các bước sau đây để tìm công cụ này:

Bước 1: Mở đường dẫn C:\Windows\SysWOW64



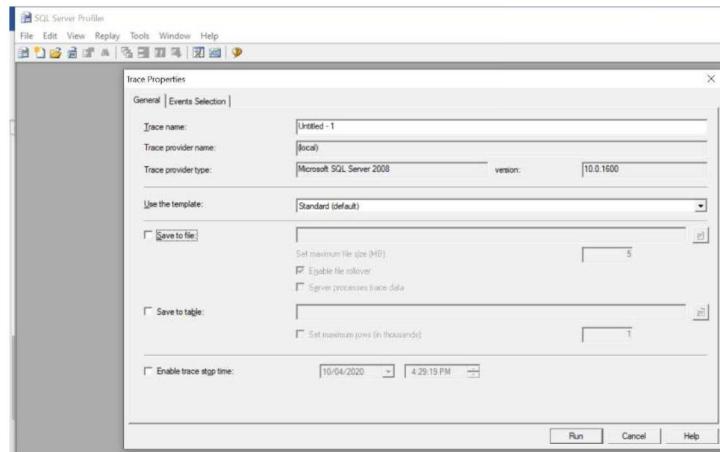
Bước 2: Nhập sqlserver vào ô tìm kiếm, sẽ xuất hiện mục sqlservermanager10.msc, (hoặc có các phiên bản sqlservermanager11.msc, sqlservermanager12.msc cho các phiên bản SQL cao hơn là 2012 và 2014)



1.4.3 MicroSoft SQL Server Profiler

Microsoft SQL Server Profiler là giao diện người dùng đồ họa để SQL Trace theo dõi thẻ hiện của cơ sở dữ liệu hoặc phân tích dịch vụ. Ta có thể chụp và lưu dữ liệu về sự kiện vào tập tin hoặc bảng để phân tích sau. Ví dụ, ta có thể giám sát môi trường phát triển để xem những thủ tục lưu trữ ảnh hưởng đến hiệu suất do thực hiện quá chậm. Profiler cho phép bạn quản lý và nắm bắt được những hoạt động đang diễn ra trong cơ sở dữ liệu của bạn, bao gồm adhoc truy vấn, lưu trữ các yêu cầu, đăng nhập, lỗi,...

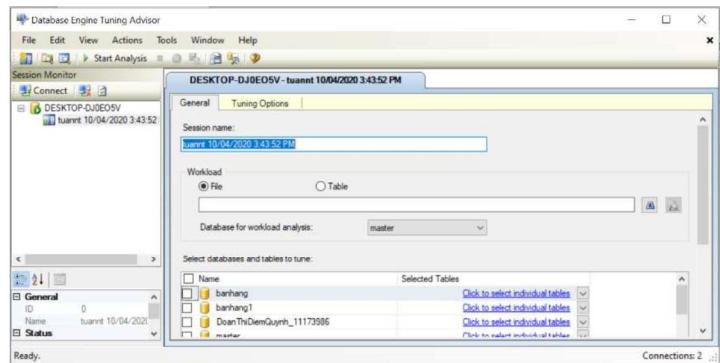
SQL Server Profiler là một công cụ cho phép ghi lại các hoạt động hàng ngày của cơ sở dữ liệu. SQL Server Profiler sẽ lưu trữ các câu truy vấn T-SQL mà chúng ta sẽ dùng làm dữ liệu đầu vào cho công cụ Turning Advisor. SQL Server Profiler là một công cụ để chúng ta có thể theo dõi được các câu truy vấn dài và giám sát hiệu năng của database. Chúng ta cũng có thể sử dụng nó để giám sát các hoạt động của database vì mục đích an ninh. SQL Server Profiler có thể sử dụng để giám sát các cơ sở dữ liệu quan hệ (Relational Databases) hoặc cơ sở dữ liệu đa chiều (Multidimensional databases).



1.4.4 MicroSoft Database Engine Tuning Advisor

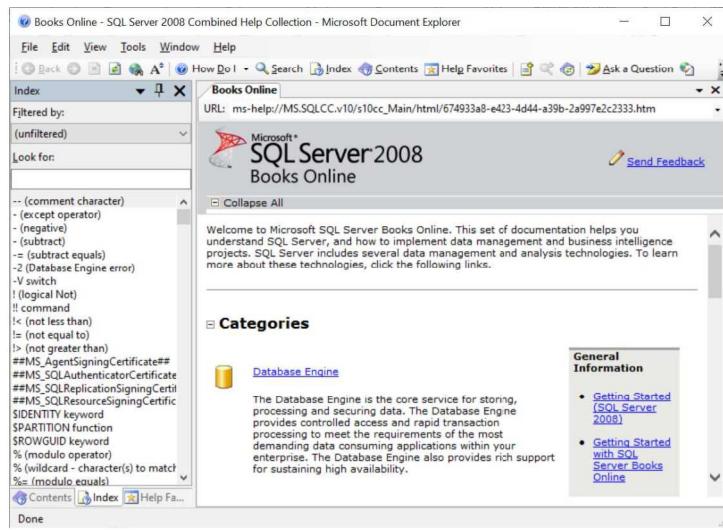
Turning Advisor là một công cụ được sử dụng để tìm ra các khuyến cáo về việc áp dụng các index, statistic và partition trong SQL Server. Các khuyến cáo này dựa trên thông tin từ các tập lệnh SQL hoặc từ một file XML hoặc từ một file Profile Trace (được SQL Server Profiler sinh ra). Công cụ này phân tích các câu truy vấn dữ liệu và đưa ra các khuyến cáo về các index, statistic và partition cho các bảng và các views trong cơ sở dữ liệu.

Turning Advisor là một công cụ rất hữu hiệu và sử dụng một cách đơn giản. Ta chỉ cần lưu trữ các câu lệnh truy vấn vào trong các file trace hoặc các bảng, sau đó đưa chúng vào Turning Advisor để phân tích. Ngoài ra, ta còn có thể sử dụng Turning Advisor để phân tích các câu lệnh đang được lưu trữ ở trong Plan cache.



1.4.5 SQLServer Books Online và SQLServer Tutorials

Đây là hai công cụ rất hữu ích, cho phép người dùng tra cứu các thông tin đầy đủ liên quan đến các MicroSoft SQL Server, bao gồm các kiến thức về kiến trúc của MicroSoft SQL Server, các kiểu dữ liệu, các hàm tương ứng với các kiểu dữ liệu, cú pháp các câu lệnh SQL và T-SQL... Với mỗi nội dung, ngoài việc trình bày và giải thích chi tiết về từng vấn đề, hai công cụ này còn có những ví dụ cụ thể mô phỏng cho nội dung vấn đề.



CHƯƠNG II. LÀM VIỆC VỚI CƠ SỞ DỮ LIỆU TRONG MICROSOFT SQL SERVER

2.1 CƠ SỞ DỮ LIỆU QUẢN LÝ BÁN HÀNG

Bài toán quản lý bán hàng là một bài toán thông dụng trong thực tế, do vậy tác giả sử dụng bài toán này và xây dựng Cơ sở dữ liệu Quản lý bán hàng để minh họa cho các ví dụ của tài liệu này. Để đơn giản, cơ sở dữ liệu của bài toán quản lý bán hàng sẽ được loại bỏ bớt thông tin và hạn chế các nghiệp vụ so với thực tế, bài toán được mô tả như sau:

Một doanh nghiệp bán lẻ xây dựng cơ sở dữ liệu Quản lý bán hàng cho doanh nghiệp của mình với các yêu cầu: Doanh nghiệp có một hệ thống nhiều đại lý bán các mặt hàng mà doanh nghiệp kinh doanh, mỗi đại lý có các nhân viên bán hàng tại đại lý của mình. Khi bán hàng cho khách, khách sẽ nhận được một phiếu bán hàng, trong đó có ghi số phiếu, ngày bán hàng, đại lý, nhân viên bán hàng, các mặt hàng, số lượng và thành tiền tương ứng của mặt hàng mà khách đã mua. Một phiếu bán hàng chỉ ghi cho một mặt hàng.

Với mô tả bài toán về Quản lý bán hàng, thiết kế cơ sở dữ liệu và các bảng như sau:

a) Cơ sở dữ liệu:

BANHANG

b) Các bảng:

Bảng Danh mục hàng: tbl_dmhang

Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
Mahang	Char(5)	Khóa chính	Mã hàng
Tenhang	Nvarchar(100)	Not Null	Tên hàng
Dvt	Nvarchar(20)	Not Null	Đơn vị tính

Dongia	float	Not Null, >0	Đơn giá
--------	-------	--------------	---------

Bảng Đại lý: tbl_daily

Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
Madl	Char(3)	Khóa chính	Mã đại lý
Tendl	Nvarchar(100)	Not Null	Tên đại lý
Diachi	Nvarchar(100)	Not Null	Địa chỉ

Bảng Nhân viên: tbl_nhanvien

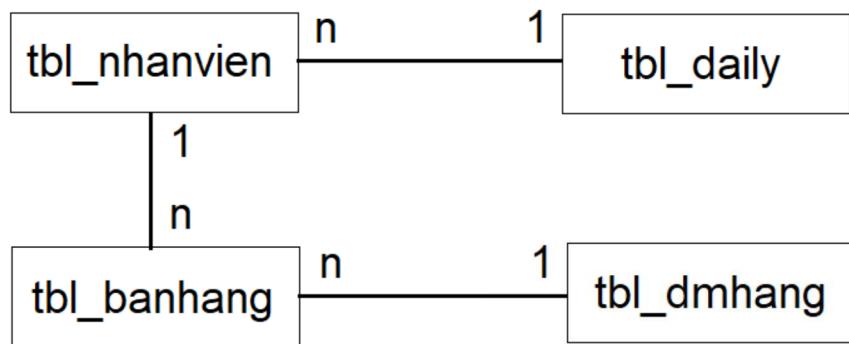
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
Manv	Char(3)	Khóa chính	Mã nhân viên
Madl	Char(3)	Not Null	Mã đại lý
Hoten	Nvarchar(30)	Not Null	Họ tên nhân viên
Dienthoai	Nvarchar(10)	Not Null, chứa 10 chữ số điện thoại	Số điện thoại của nhân viên

Bảng Bán hàng: tbl_banhang

Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
Sophieu	Char(5)	Khóa chính	Số phiếu bán hàng
Ngay	Smalldatetime	Not Null	Ngày bán hàng
Manv	Char(3)	Not Null	Mã nhân viên
Mahang	Char(5)	Not Null	Mã hàng
Soluong	Float	Not Null, >0	Số lượng hàng

c) Quan hệ giữa các bảng

Qua các phân tích các mô tả của bài toán quản lý bán hàng và các thông tin cần lưu trữ, mối quan hệ giữa các bảng được xác định như sau:



d) Dữ liệu mẫu

Bảng Danh mục hàng

	mahang	tenhang	dvt	dongia
1	H0001	Máy tính xách tay Dell	Bộ	20000000
2	H0002	Máy tính xách tay Toshiba	Bộ	18000000
3	H0003	Máy tính xách tay lenovo	Bộ	15000000
4	H0004	Máy tính để bàn	Bộ	15000000
5	H0005	Máy in canon	Bộ	6000000
6	H0006	Chuột không dây	Cái	300000
7	H0007	Bộ nhớ USB 32TB	Cái	500000
8	H0008	Tai nghe	Cái	200000
9	H0009	Điện thoại Samsung	Cái	10000000
10	H0010	Điện thoại IPhone	Cái	2000000

Bảng Đại lý

	madl	tendl	diachi
1	D01	Bách Khoa	10 Phố Tạ Quang Bửu, Bách Khoa, HN
2	D02	Phố Vọng	234 Phố Vọng, Đồng Tâm, HN
3	D03	Hoàn Kiếm	2, Tràng Tiền, Hoàn Kiếm, HN

Bảng Nhân viên

	manv	madl	hoten	dienthoai
1	N01	D01	Nguyễn Thị Hồng	0912345678
2	N02	D01	Trần Văn Mạnh	0878234561
3	N03	D02	Phạm Thúy Quỳnh	0903344586
4	N04	D03	Đào Đức Anh	0904345618
5	N05	D03	Ngô Huyền Trâm	0988765432

Bảng Bán hàng

	sophieu	ngay	manv	mahang	soluong
1	P0001	2020-01-10 00:00:00	N02	H0001	3
2	P0002	2020-01-20 00:00:00	N01	H0002	6
3	P0003	2020-04-06 00:00:00	N03	H0003	15
4	P0004	2020-02-17 00:00:00	N04	H0004	9
5	P0005	2020-07-25 00:00:00	N05	H0006	20
6	P0006	2019-06-05 00:00:00	N01	H0010	10
7	P0007	2020-03-11 00:00:00	N02	H0009	1
8	P0008	2019-08-13 00:00:00	N03	H0001	8
9	P0009	2020-09-04 00:00:00	N05	H0002	40
10	P0010	2020-11-19 00:00:00	N04	H0004	6
11	P0011	2020-06-28 00:00:00	N04	H0005	14
12	P0012	2020-11-04 00:00:00	N01	H0009	5
13	P0013	2020-09-09 00:00:00	N03	H0010	2
14	P0014	2020-02-07 00:00:00	N03	H0002	4
15	P0015	2020-05-15 00:00:00	N05	H0003	9

2.2 CÁC KIỂU DỮ LIỆU TRONG MICROSOFT SQL SERVER

2.2.1 Các kiểu dữ liệu

a) Các kiểu ký tự

Cú pháp	Kích thước tối đa	Giải thích
CHAR(kich_thuoc)	Tối đa 8000 kí tự.	<ul style="list-style-type: none"> • kich_thuoc là số kí tự lưu trữ. • Độ dài cố định. • Thêm dấu cách về bên phải để bù phần trống cho đủ số kí tự. • Không chứa kí tự Unicode.
VARCHAR(kich_thuoc) hoặc VARCHAR(toi_da)	Tối đa 8000 kí tự hoặc theo số tối đa.	<ul style="list-style-type: none"> • kich_thuoc là số kí tự lưu trữ. • Độ dài tùy biến. • Nếu chỉ định là toi_da thì tối đa là 2GB. • Không chứa kí tự Unicode.
TEXT	Tối đa 2GB.	<ul style="list-style-type: none"> • Độ dài tùy biến. • Không chứa kí tự Unicode.
NCHAR(kich_thuoc)	Tối đa 4000 kí tự.	<ul style="list-style-type: none"> • Độ dài cố định. • Kí tự Unicode.
NVARCHAR(kich_thuoc) hoặc NVARCHAR(toi_da)	Tối đa 4000 kí tự hoặc theo số tối da.	<ul style="list-style-type: none"> • kich_thuoc là số kí tự lưu trữ. • Độ dài tùy biến.

		<ul style="list-style-type: none"> Nếu số <code>toi_da</code> được chỉ định thì số kí tự tối đa là 2GB. Kí tự Unicode.
NTEXT	Tối đa 1.073.741.823 byte.	<ul style="list-style-type: none"> Độ dài tùy biến. Kí tự Unicode.

b) Các kiểu số nguyên

Cú pháp	Kích thước tối đa	Giải thích
BIT	số nguyên 0, 1 hoặc NULL	
TINYINT	từ 0 đến 255	
SMALLINT	từ -32768 đến 32767	
INT	-2,147,483,648 đến 2,147,483,647	
BIGINT	từ -9,223,372,036,854,775,808 đến 9,223,372,036,854,775,807	

c) Các kiểu số thực

Cú pháp	Kích thước tối đa	Giải thích
DECIMAL(m,d)	<ul style="list-style-type: none"> m mặc định là 18 nếu không được chỉ định cụ thể. d mặc định là 0 nếu không được chỉ định cụ thể. 	<p>m là tổng số lượng các số còn d là số lượng các số nằm sau dấu phẩy.</p>

DEC(m,d)	<ul style="list-style-type: none"> m mặc định là 18 nếu không được chỉ định cụ thể. d mặc định là 0 nếu không được chỉ định cụ thể. 	m là tổng số lượng các số còn d là số lượng các số nằm sau dấu phẩy. Đồng nghĩa với kiểu dữ liệu DECIMAL.
NUMERIC(m,d)	<ul style="list-style-type: none"> m mặc định là 18 nếu không được chỉ định cụ thể. d mặc định là 0 nếu không được chỉ định cụ thể. 	m là tổng số lượng các số còn d là số lượng các số nằm sau dấu phẩy. Đồng nghĩa với kiểu dữ liệu DECIMAL.
FLOAT(n)	số dấu phẩy động n mặc định là 53 nếu không được chỉ định cụ thể.	n là số lượng của số bit lưu trữ trong một kí hiệu hóa học.
REAL	tương đương với FLOAT(24)	
SMALLMONEY	từ - 214,748.3648 đến 214,748.3647	
MONEY	từ - 922,337,203,685,477.5808 đến 922,337,203,685,477.5807	

d) Các kiểu thời gian

Cú pháp	Kích thước tối đa	Giải thích

DATE	giá trị từ '0001-01-01' đến '9999-12-31.'	hiển thị dưới dạng 'YYYY-MM-DD'
DATETIME	<ul style="list-style-type: none"> Ngày lấy từ '1753-01-01 00:00:00' to '9999-12-31 23:59:59'. Giờ lấy từ '00:00:00' to '23:59:59:997' 	hiển thị dưới dạng 'YYYY-MM-DD hh:mm:ss[.mmm]'
DATETIME2(chính xác tới số thập phân của giây)	<ul style="list-style-type: none"> giá trị lấy từ '0001-01-01' đến '9999-12-31'. Thời gian lấy từ '00:00:00' đến '23:59:59:999999'. 	hiển thị dưới dạng 'YYYY-MM-DD hh:mm:ss[.số giây thập phân]'
SMALLDATETIME	<ul style="list-style-type: none"> giá trị lấy từ '1900-01-01' đến '2079-06-06'. Thời gian lấy từ '00:00:00' đến '23:59:59'. 	hiển thị dưới dạng 'YYYY-MM-DD hh:mm:ss'
TIME	<ul style="list-style-type: none"> giá trị lấy từ '00:00:00.0000000' đến '23:59:59.9999999'. Ngày lấy từ '0001-01-01' đến '9999-12-31'. 	hiển thị dưới dạng 'YYYY-MM-DD hh:mm:ss[.nnnnnnnn]'

DATETIMEOFFSET (chính xác tới số thập phân của giây)	<ul style="list-style-type: none"> giá trị thời gian lấy từ '00:00:00' đến '23:59:59:999999'. Múi giờ lấy từ -14:00 đến +14:00. 	hiển thị dưới dạng YYYY-MM-DD hh:mm:ss[.nnnnnnn] [{+-}hh:mm]
---------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------

e) Các kiểu khác

Cú pháp	Kích thước tối đa	Giải thích
BINARY(kich_thuoc)	Tối đa 8000 kí tự.	<ul style="list-style-type: none"> kich_thuoc là số kí tự lưu trữ. Độ dài cố định. Thêm dấu cách để bù phần trống cho đủ số kí tự. Dữ liệu nhị phân.
VARBINARY(kich_thuoc) hoặc VARBINARY(toi_da)	Tối đa 8000 kí tự hoặc theo số tối đa.	<ul style="list-style-type: none"> kich_thuoc là số kí tự lưu trữ. Độ dài tùy biến. Nếu chỉ định là toi_da thì tối đa là 2GB. Dữ liệu nhị phân.
IMAGE	kích thước tối đa là 2GB.	<ul style="list-style-type: none"> Độ dài tùy biến. Dữ liệu nhị phân.

2.2.2 Các hàm cơ bản

a) Các hàm với kiểu ký tự

Cú pháp	Ý nghĩa
ASCII(<i>character_expression</i>)	Trả về mã ASCII của ký tự bên trái cùng của chuỗi
CHAR(<i>integer_expression</i>)	Trả về ký tự có mã ASCII trong tham số
CHARINDEX(<i>expression1</i> , <i>expression2</i> [, <i>start_location</i>])	Tìm <i>expression1</i> trong chuỗi <i>expression2</i> và trả về vị trí đầu tiên tìm thấy, vị trí bắt đầu tìm tại <i>start_location</i> . Nếu vị trí bắt đầu tìm bỏ qua thì sẽ tìm từ ký tự đầu trong chuỗi.
DIFFERENCE(<i>character_expression</i> , <i>character_expression</i>)	Trả về một giá trị nguyên cho biết sự khác biệt giữa các giá trị SOUNDEX của hai biểu thức ký tự.
SOUNDEX(<i>character_expression</i>)	Trả về mã gồm bốn ký tự (SOUNDEX) để đánh giá sự giống nhau của hai chuỗi.
LEFT(<i>character_expression</i> , <i>integer_expression</i>)	Trả về phần bên trái của chuỗi ký tự với số lượng ký tự được chỉ định.
LEN(<i>character_expression</i>)	Trả về số lượng ký tự trong một biểu thức ký tự. LEN trả về các giá trị như nhau cho cùng một chuỗi các ký tự byte đơn và kép.
LOWER(<i>character_expression</i>)	Trả về một biểu thức ký tự sau khi chuyển đổi các ký tự viết hoa thành các ký tự chữ thường.
LTRIM(<i>character_expression</i>)	Trả về một biểu thức ký tự sau khi nó loại bỏ khoảng trắng ở đầu chuỗi.
NCHAR(<i>integer_expression</i>)	Trả về ký tự Unicode với mã số nguyên được chỉ định.

PATINDEX('% <i>pattern</i> %', <i>expression</i>)	Trả về vị trí bắt đầu của lần xuất hiện đầu tiên của mẫu trong biểu thức đã chỉ định hoặc 0 nếu không tìm thấy mẫu đó, trên tất cả các loại dữ liệu văn bản và ký tự hợp lệ.
QUOTENAME('character_string' [, 'quote_character'])	Trả về một chuỗi Unicode với các dấu phân cách được thêm vào để làm cho chuỗi đầu vào trở thành một định danh phân cách hợp lệ trong Microsoft SQL Server.
REPLACE(<i>string_expression1</i> , <i>string_expression2</i> , <i>string_expression3</i>)	Thay thế tất cả các lần xuất hiện của một chuỗi được chỉ định bằng một chuỗi khác.
REPLICATE(<i>string_expression</i> , <i>integer_expression</i>)	Lặp lại một chuỗi một số lần chỉ định.
REVERSE(<i>character_expression</i>)	Trả về thứ tự đảo ngược của biểu thức ký tự.
RIGHT(<i>character_expression</i> , <i>integer_expression</i>)	Trả về phần bên phải của chuỗi ký tự với số lượng ký tự được chỉ định.
RTRIM(<i>character_expression</i>)	Trả về một chuỗi ký tự sau khi cắt tất cả các khoảng trống cuối chuỗi.
SPACE(<i>integer_expression</i>)	Trả về một dãy các dấu trắng được chỉ ra trong hàm
STR(<i>float_expression</i> [, <i>length</i> [, <i>decimal</i>]])	Trả về chuỗi ký tự được chuyển đổi từ dữ liệu số.
STUFF(<i>character_expression</i> , <i>start</i> , <i>length</i> , <i>character_expression</i>)	Hàm STUFF chèn một chuỗi vào một chuỗi khác. Nó xóa <i>length</i> ký tự được chỉ định trong chuỗi đầu tiên tại vị trí bắt đầu

	và sau đó chèn chuỗi thứ hai vào chuỗi đầu tiên ở vị trí bắt đầu.
SUBSTRING(<i>value_expression</i> , <i>start_expression</i> , <i>length_expression</i>)	Trả về một phần của biểu thức chuỗi ký tự, nhị phân, văn bản hoặc hình ảnh.
UNICODE('ncharacter_expression')	Trả về giá trị số nguyên, như được định nghĩa bởi tiêu chuẩn Unicode, cho ký tự đầu tiên của biểu thức đầu vào.
UPPER(<i>character_expression</i>)	Trả về dãy ký tự chuyển đổi thành chữ in hoa

b) Các hàm với kiểu số

Cú pháp	Ý nghĩa
ABS(<i>numeric_expression</i>)	Trả về giá trị tuyệt đối của biểu thức
ACOS(<i>float_expression</i>)	Một hàm toán học trả về góc, tính bằng radian, có COSIN là biểu thức float xác định
ASIN(<i>float_expression</i>)	Một hàm toán học trả về góc, tính bằng radian, có SIN là biểu thức float xác định
ATAN(<i>float_expression</i>)	Một hàm toán học trả về góc, tính bằng radian, có TAN là biểu thức float xác định
ATN2(<i>float_expression</i>)	Trả về góc, tính bằng radian, giữa trực x dương và tia từ gốc đến điểm (y, x), trong đó x và y là giá trị của hai biểu thức float xác định.
CEILING (<i>numeric_expression</i>)	Trả về số nguyên nhỏ nhất lớn hơn hoặc bằng biểu thức số.
COS(<i>float_expression</i>)	Là một hàm toán học trả về cosin lượng giác của góc đã chỉ định, tính bằng radian, trong biểu thức đã chỉ định.

$\text{COT}(\text{float_expression})$	Một hàm toán học trả về cotang lượng giác của góc đã chỉ định, tính bằng radian, trong biểu thức float được chỉ định.
$\text{DEGREES}(\text{numeric_expression})$	Trả về góc tương ứng theo độ cho một góc được chỉ định bằng radian.
$\text{EXP}(\text{float_expression})$	Trả về giá trị hàm mũ của biểu thức float đã chỉ định.
$\text{FLOOR}(\text{numeric_expression})$	Trả về số nguyên lớn nhất nhỏ hơn hoặc bằng biểu thức số.
$\text{LOG}(\text{float_expression})$	Trả về logarit tự nhiên của biểu thức float đã chỉ định.
$\text{LOG10}(\text{float_expression})$	Trả về logarit cơ số 10 của biểu thức float đã chỉ định.
$\text{PI}()$	Trả về giá trị PI
$\text{POWER}(\text{float_expression}, y)$	Trả về giá trị của biểu thức đã chỉ định cho mũ được chỉ định.
$\text{RADIANS}(\text{numeric_expression})$	Trả về radian khi một biểu thức số, tính bằng độ, được nhập.
$\text{RAND}([\text{seed}])$	Trả về giá trị float ngẫu nhiên từ 0 đến 1.
$\text{ROUND}(\text{numeric_expression}, length [,function])$	Trả về một giá trị số, làm tròn đến độ dài hoặc độ chính xác đã chỉ định.
$\text{SIGN}(\text{numeric_expression})$	Trả về dấu dương (+1), 0 (0) hoặc âm (-1) của biểu thức đã chỉ định.
$\text{SIN}(\text{float_expression})$	Trả về sin lượng giác của góc đã chỉ định, tính bằng radian và trong một biểu thức số, float, gần đúng.

SQRT(<i>float_expression</i>)	Trả về căn bậc hai của giá trị float được chỉ định.
SQUARE(<i>float_expression</i>)	Trả về bình phương của giá trị float được chỉ định.
TAN(<i>float_expression</i>)	Trả về TAN biểu thức đầu vào.

c) Các hàm với kiểu thời gian

Cú pháp	Ý nghĩa
DAY (<i>date</i>)	Trả về ngày trong biểu thức thời gian
MONTH (<i>date</i>)	Trả về tháng trong biểu thức thời gian
YEAR (<i>date</i>)	Trả về năm trong biểu thức thời gian
ISDATE(<i>expression</i>)	Trả về 1 nếu một biểu thức đầu vào là giá trị ngày hoặc thời gian hợp lệ của các kiểu dữ liệu datetime hoặc smalldatetime; mặt khác, 0.
DATENAME(<i>datepart, date</i>)	Trả về chuỗi ký tự biểu thị thành phần được chỉ định của biểu thức ngày.
DATEPART (<i>datepart, date</i>)	Trả về một số nguyên biểu thị thành phần được chỉ định của biểu thức ngày.
GETDATE()	Trả về dấu thời gian của hệ thống cơ sở dữ liệu hiện tại dưới dạng giá trị datetime mà không có bù múi giờ của cơ sở dữ liệu. Giá trị này được lấy từ hệ điều hành của máy tính mà phiên bản SQL Server đang chạy.
DATEDIFF(<i>datepart, startdate, enddate</i>)	Trả về số lượng (số nguyên có dấu) của các ranh giới ngày tháng được chỉ định được giao giữa ngày bắt đầu và ngày kết thúc được chỉ định.

DATEADD (<i>datepart</i> , <i>number</i> , <i>date</i>)	Trả về một ngày được chỉ định với khoảng thời gian số được chỉ định (số nguyên đã ký) được thêm vào một ngày được chỉ định của ngày đó.
SYSDATETIME()	Trả về giá trị datetime2 (7) có chứa ngày và giờ của máy tính mà phiên bản SQL Server đang chạy.

d) Các hàm thống kê

Cú pháp	Ý nghĩa
AVG([ALL DISTINCT] <i>expression</i>)	Trả về giá trị trung bình của các giá trị trong một nhóm. Giá trị Null được bỏ qua.
COUNT({ [[<u>ALL</u> DISTINCT] <i>expression</i>] * })	Trả về số lượng các mục trong một nhóm. COUNT hoạt động giống như hàm COUNT_BIG. Sự khác biệt duy nhất giữa hai hàm là giá trị trả về của chúng. COUNT luôn trả về giá trị kiểu dữ liệu int. COUNT_BIG luôn trả về giá trị loại dữ liệu bigint.
COUNT_BIG({ [[<u>ALL</u> DISTINCT] <i>expression</i>] * })	Trả về số lượng các mục trong một nhóm. COUNT_BIG luôn trả về giá trị loại dữ liệu bigint.
MIN([<u>ALL</u> DISTINCT] <i>expression</i>)	Trả về giá trị nhỏ nhất trong biểu thức một nhóm.
MAX([<u>ALL</u> DISTINCT] <i>expression</i>)	Trả về giá trị lớn nhất trong biểu thức một nhóm.
SUM([ALL DISTINCT] <i>expression</i>)	Trả về tổng của các giá trị biểu thức trong một nhóm.
STDEV([<u>ALL</u> DISTINCT] <i>expression</i>)	Trả về độ lệch chuẩn thống kê của tất cả các giá trị trong biểu thức đã chỉ định.

STDEVP([<u>ALL</u> DISTINCT] <i>expression</i>)	Trả về độ lệch chuẩn thống kê cho tập tất cả các giá trị trong biểu thức đã chỉ định.
VAR([<u>ALL</u> DISTINCT] <i>expression</i>)	Trả về phương sai thống kê của tất cả các giá trị trong biểu thức đã chỉ định.
VARP([<u>ALL</u> DISTINCT] <i>expression</i>)	Trả về phương sai thống kê cho tập tất cả các giá trị trong biểu thức đã chỉ định.

e) Hàm chuyển kiểu

Cú pháp cho hàm CAST:

CAST (<biểu thức> AS <kiểu dữ liệu> [độ rộng)])

Cú pháp cho hàm CONVERT:

CONVERT (<kiểu dữ liệu> [(độ rộng)] , <biểu thức> [, <loại>])

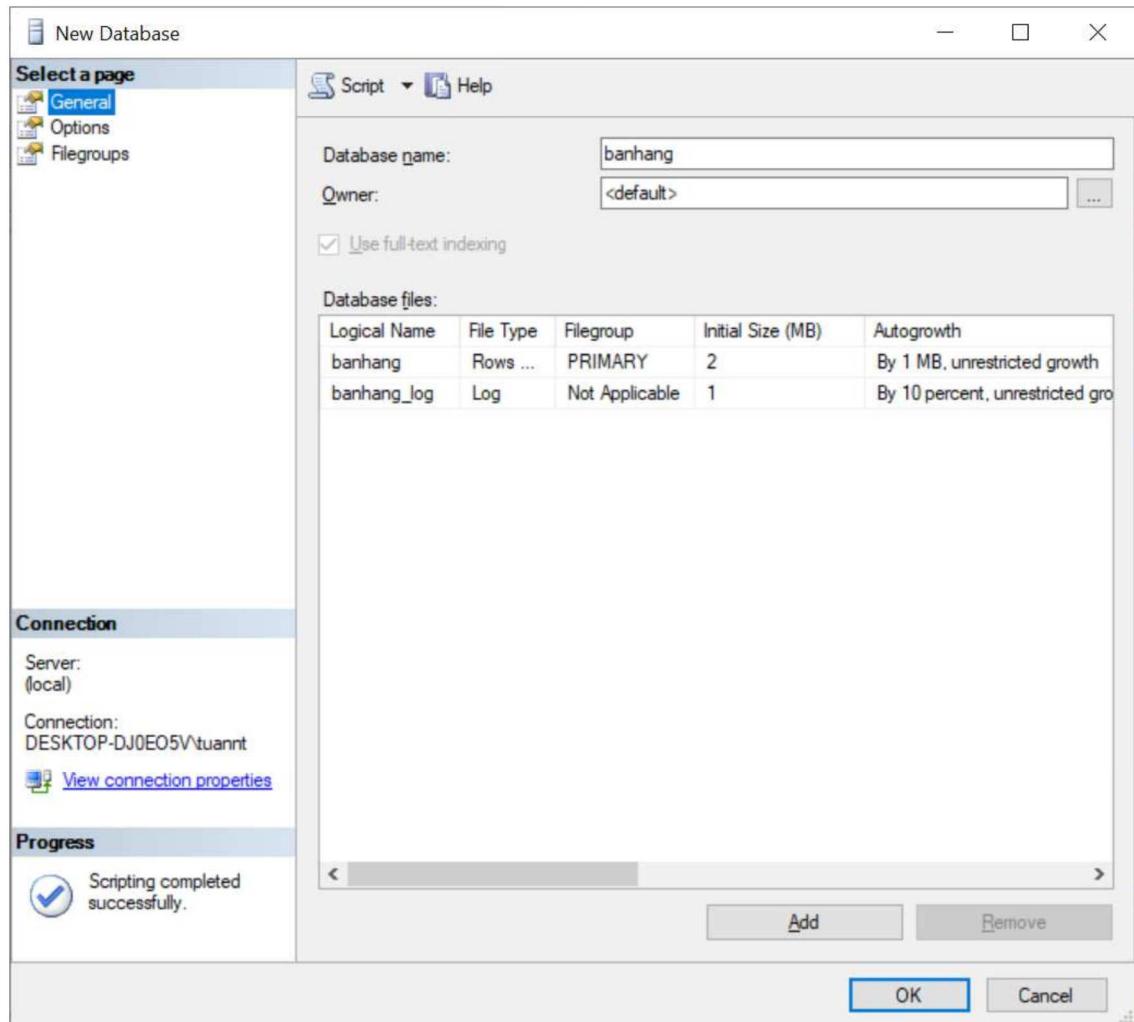
Các tham số:

- <biểu thức>: là dữ liệu hoặc biểu thức muốn chuyển kiểu
- <kiểu dữ liệu>: là kiểu dữ liệu mong muốn <biểu thức> sẽ được chuyển đổi.
- <độ rộng>: đây là thông tin tùy chọn, là một số nguyên để xác định độ rộng của kiểu dữ liệu muốn chuyển đổi đến. Giá trị ngầm định là 30.
- <loại> : là một biểu thức số nguyên xác định cách để hàm CONVERT dịch <biểu thức>. Nếu <loại> là NULL thì sẽ trả về NULL. Phạm vi được xác định bằng <kiểu dữ liệu>

2.3 SỬ DỤNG MICROSOFT SQL SERVER MANAGEMENT STUDIO ĐỂ TẠO CƠ SỞ DỮ LIỆU

2.3.1 Tạo cơ sở dữ liệu

Bước 1: Chạy MicroSoft SQL Server Management Studio, nhấn phím phải chuột lên mục Databases, chọn New Database



Bước 2: Nhập/chọn thông số cơ bản cho các mục:

- Database name: nhập tên cơ sở dữ liệu muốn tạo. Trong ví dụ, nhập **banhang**
- Owner: chọn tên người dùng tạo cơ sở dữ liệu, nếu không chọn, hệ thống sẽ lấy người sử dụng đang đăng nhập làm người dùng tạo cơ sở dữ liệu.
- Database files: cho phép đặt tên, thư mục lưu trữ, kích thước ban đầu, kích thước file tự động tăng và tên file cho các file chứa dữ liệu, file ghi log của cơ sở dữ liệu, tương ứng với các mục: **Logic Name, Initial Size, Autogrowth, Path, File Name**.

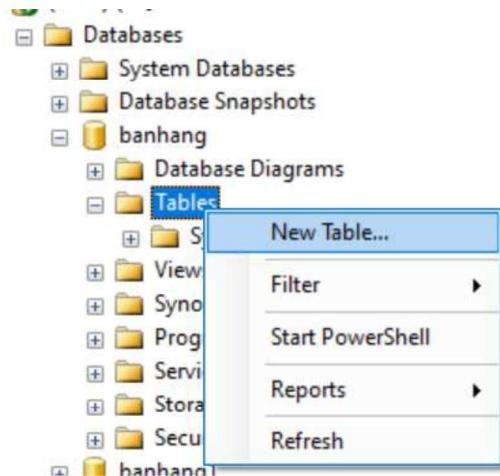
Sau khi chọn các thông số, nhấn nút **OK** để kết thúc. Cơ sở dữ liệu đã được tạo sẽ có tên xuất hiện trên cây các đối tượng của MicroSoft SQL Server và sẽ được hệ thống quản lý.

Một số thao tác có thể trực tiếp thực hiện trên cơ sở dữ liệu đã được tạo:

- Sửa tên cơ sở dữ liệu: Nhấn phím phải chuột lên tên cơ sở dữ liệu, chọn mục **Rename** sau đó đặt tên mới cho cơ sở dữ liệu.
- Xóa cơ sở dữ liệu: Nhấn phím phải chuột lên tên cơ sở dữ liệu, chọn mục **Delete**, lựa chọn các thông số để xóa: các dữ liệu lịch sử, các kết nối đến cơ sở dữ liệu.
- Thay đổi các thông số của cơ sở dữ liệu: Nhấn phím phải chuột lên tên cơ sở dữ liệu, chọn mục **Properties**, lựa chọn các thông số cần thiết lập lại.

2.3.2 Tạo bảng

Bước 1: Trên cây các đối tượng của MicroSoft SQL Server, mở **Databases**, chọn cơ sở dữ liệu, nhấn phím phải chuột lên **Tables** và chọn **New Table**



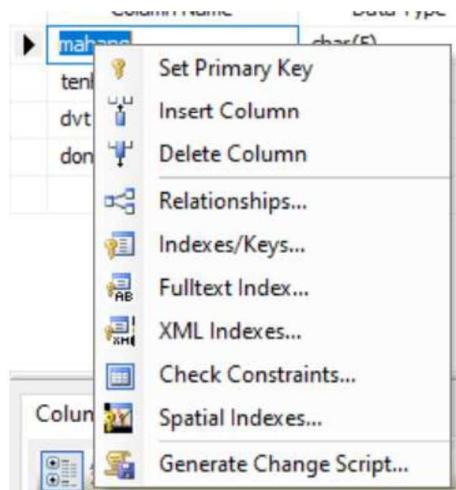
Bước 2: Lần lượt khai báo các trường của bảng

Column Name	Data Type	Allow Nulls
mahang	char(5)	<input type="checkbox"/>
tenhang	nvarchar(100)	<input type="checkbox"/>
dvt	nvarchar(20)	<input checked="" type="checkbox"/>
dongia	float	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

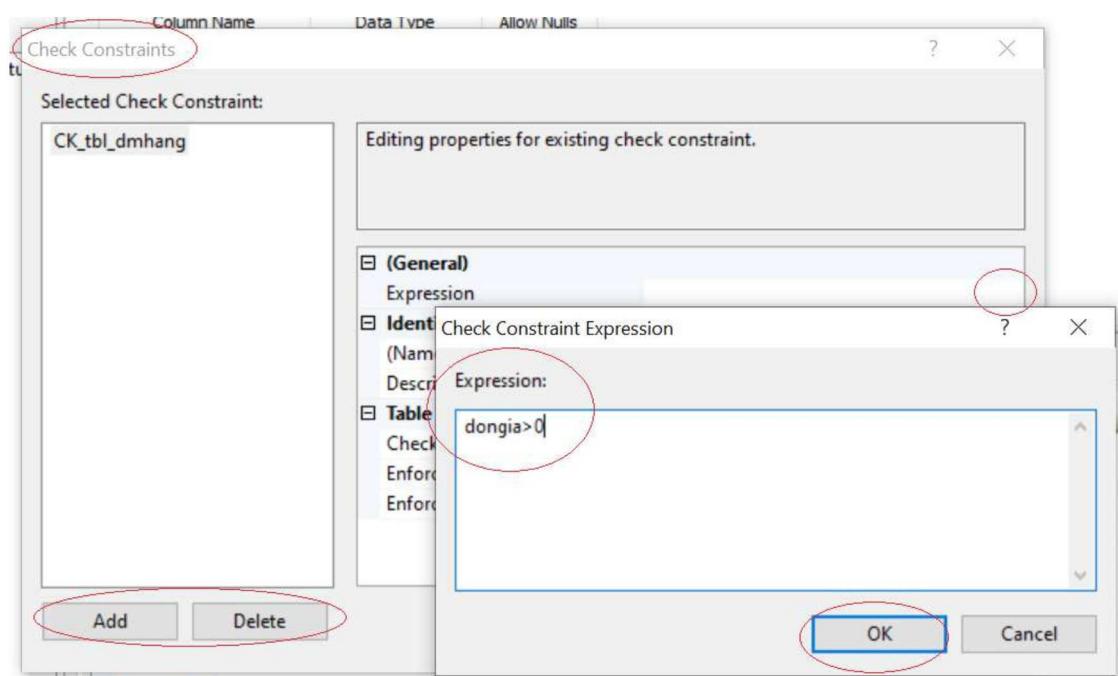
Bước 3: Chọn các ràng buộc, điều kiện cho các trường (có thể thực hiện ngay khi khai báo trường) bằng cách nhấn phím phải chuột lên trường sau đó chọn thông số muốn đặt.

Chú ý:

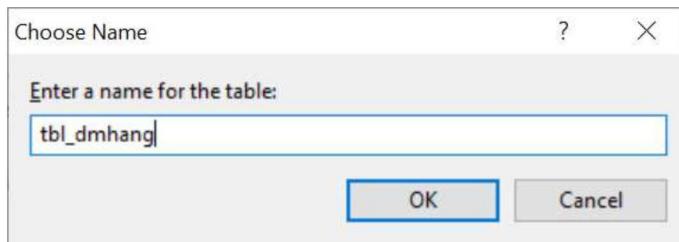
Trong trường hợp muốn chọn nhiều trường làm khóa chính, ta chọn các trường đó và chọn điều kiện **Primary Key**.



Ta có thể định nghĩa nhiều ràng buộc cho bảng:



Bước 4: Nhấn phím phải chuột lên bảng và chọn mục **Save**, hoặc nhấn biểu tượng **Save** trên thanh công cụ. Đặt tên cho bảng và chọn **OK**

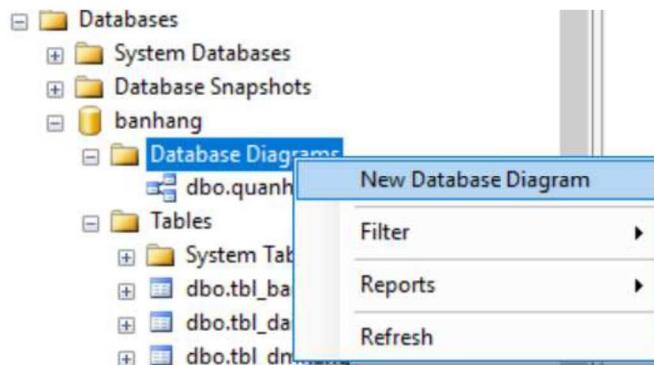


Sau khi tạo bảng, ta có thể thực hiện các thao tác:

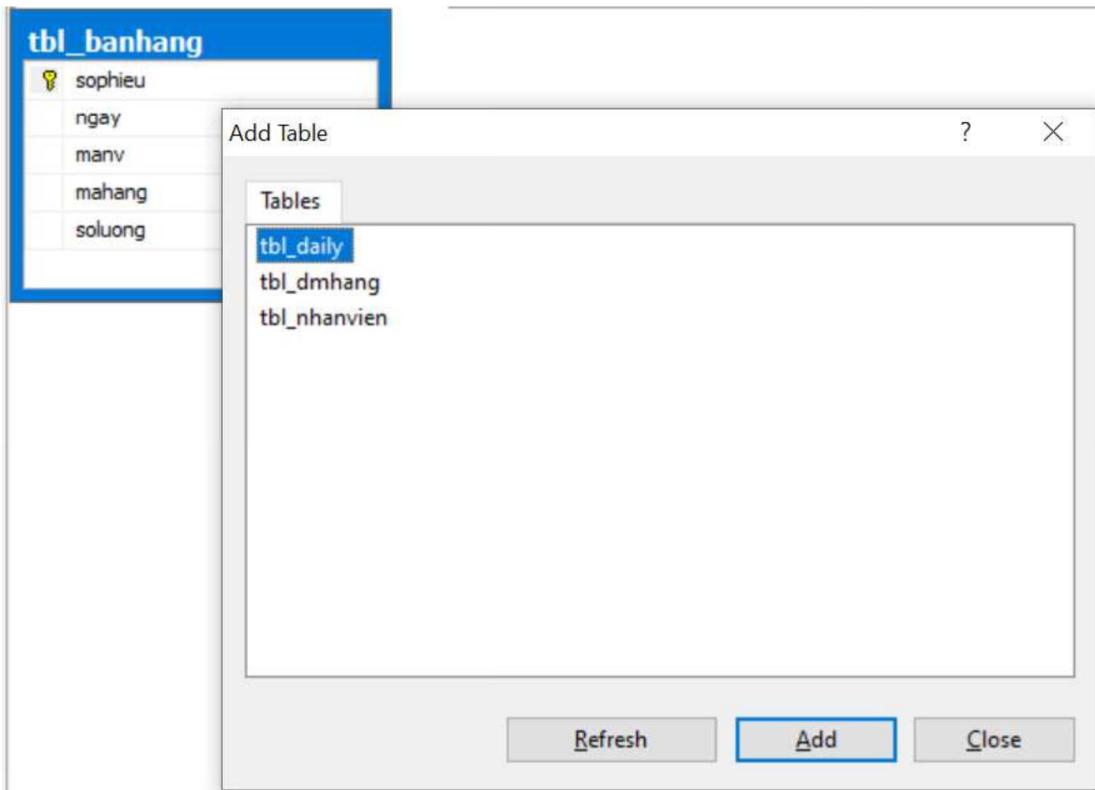
- Sửa cấu trúc bảng: Nhấn phím phải chuột lên bảng, chọn **Design**
- Nhập dữ liệu cho bảng: **Edit top .. rows**
- Xem dữ liệu của bảng: **Select top .. rows**

2.3.3 Tạo lược đồ quan hệ (Diagram)

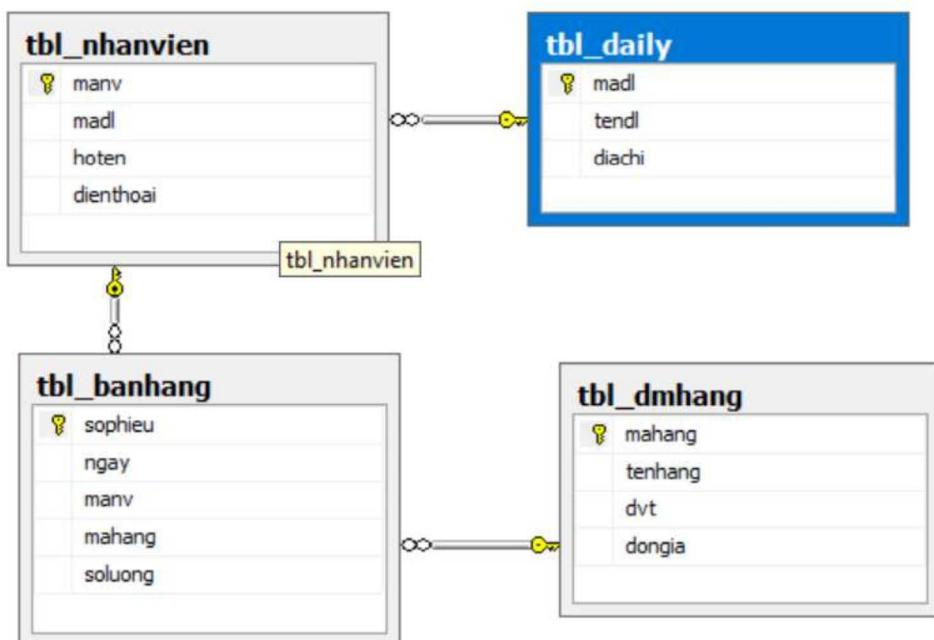
Bước 1: Trên cây các đối tượng của MicroSoft SQL Server, mở **Databases**, chọn cơ sở dữ liệu, nhấn phím phải chuột lên **Database Diagrams** và chọn **New Database Diagram**



Bước 2: Chọn các bảng của lược đồ.



Bước 3: Thiết lập mối quan hệ giữa các bảng bằng cách nhấn và kéo tên các trường khóa chính vào các khóa ngoại của các bảng



2.4 KHAI THÁC CƠ SỞ DỮ LIỆU BẰNG LỆNH TRUY VẤN SELECT

2.4.1 Cú pháp lệnh

```
SELECT [ ALL | DISTINCT ]
[ TOP expression [ PERCENT ] ]
<select_list>
[ WHERE search_condition ]
[ GROUP BY group_by_expression ]
[ HAVING search_condition ]
[ ORDER BY order_expression [ ASC | DESC ] ]
```

<select_list> ::=

```
{*
| { table_name | view_name | table_alias }.*}
| {
  [ { table_name | view_name | table_alias }. ]
  { column_name }
| expression
  [ [ AS ] column_alias ]
}
| column_alias = expression
} [ ,...n ]
```

Các tham số của lệnh:

- **ALL**: Cho phép các bản ghi trùng nhau có thể xuất hiện trong tập kết quả. ALL là giá trị ngầm định của lệnh. Các giá trị NULL cũng được coi là bằng nhau đối với từ khóa này.
- **DISTINCT**: Xác định chỉ cho phép các bản ghi xuất hiện một cách duy nhất trong kết quả.
- **TOP expression [PERCENT]**: Xác định chỉ có một số lượng bản ghi hoặc số phần trăm bản ghi đầu tiên được thể hiện trong kết quả truy vấn, *expression* có thể là một con số hoặc phần trăm các bản ghi. Mệnh đề TOP có thể được sử dụng trong các câu lệnh SELECT, INSERT, UPDATE và DELETE. Dấu ngoặc đơn phân cách biểu thức trong TOP là bắt buộc trong các câu lệnh INSERT, UPDATE và DELETE. Để tương thích ngược, sử dụng biểu thức TOP không có dấu ngoặc đơn trong các câu lệnh SELECT được hỗ trợ, nhưng không khuyến nghị điều đó.
- **<select_list>**: Các cột/trường được chọn cho tập kết quả. Danh sách chọn là một chuỗi các biểu thức được phân tách bằng dấu phẩy. Số lượng biểu thức tối đa có thể được chỉ định trong danh sách chọn là 4096.
- **:** Chỉ định rằng tất cả các cột từ tất cả các bảng và khung nhìn (view) trong mệnh đề FROM được trả về. Các cột được trả về theo bảng hoặc khung nhìn, như được chỉ định trong mệnh đề FROM và theo thứ tự chúng tồn tại trong bảng hoặc khung nhìn.
- ***table_name | view_name | table_alias.**** : Giới hạn chỉ cho các bảng hoặc các khung nhìn được xác định.
- ***column_name*** : Là tên của một cột để trả lại. Với nhiều bảng hoặc khung nhìn có tên cột trùng nhau, cần phải tránh việc tham chiếu không rõ ràng bằng cách chỉ ra tên bảng hoặc khung nhìn kèm theo tên cột, có dạng *table_name | view_name | table_alias.column_name*.
- ***expression***: Là một hằng số, hàm, bất kỳ sự kết hợp nào của tên cột, hằng số và các hàm được kết nối bởi một toán tử hoặc nhiều toán tử. hoặc một truy vấn con.

- *column_alias*: Là một tên thay thế để thay thế tên cột trong tập kết quả truy vấn. Tên thay thế này có thể được sử dụng trong mệnh đề **ORDER BY**. Tuy nhiên, nó không thể được sử dụng trong mệnh đề **WHERE**, **GROUP BY** hoặc **HAVING**.
- **WHERE**: kiểm tra điều kiện để chọn các kết quả truy vấn
- **GROUP BY**: nhóm các kết quả của truy vấn
- **HAVING**: kiểm tra kết quả theo từng nhóm của mệnh đề **GROUP BY**
- **ORDER BY**: Sắp xếp kết quả của truy vấn theo biểu thức lựa chọn. **ASC**: tăng dần, **DESC**: giảm dần
- *search_condition*: Biểu thức điều kiện để truy vấn lựa chọn các bản ghi đưa vào kết quả.
- *group_by_expression*: Biểu thức để gộp nhóm các bản ghi khi tính toán các giá trị trong truy vấn.
- *search_condition*: Điều kiện trong mỗi nhóm của GROUP BY.
- *order_expression*: Biểu thức để sắp xếp các kết quả của truy vấn.

Chú ý:

- + Các từ khóa không phân biệt chữ in, chữ thường. Một lệnh có thể viết trên nhiều dòng.
- + Có thể sử dụng các lệnh truy vấn lồng nhau
- + Sử dụng các phép toán logic: NOT, AND, OR. Các toán hạng là các biểu thức chứa các toán tử so sánh: =, >, >=, <> (hoặc ≠), <, <=, các phép toán số học: +, -, *, /.
- + Trong các truy vấn có thể sử dụng các hàm, các phép toán với tập hợp...
- + Nếu mệnh đề WHERE và HAVING có cùng biểu thức thì biểu thức ở WHERE áp dụng trước. Các bản ghi thỏa mãn WHERE và nhóm bởi GROUP BY, mệnh đề HAVING (nếu có) sau đó mới được áp dụng trên mỗi nhóm. Nếu các nhóm không thỏa mãn mệnh đề HAVING sẽ bị loại bỏ.

2.4.2 Một số ví dụ truy vấn khai thác dữ liệu

Truy vấn không có điều kiện

Yêu cầu:

Đưa ra danh sách gồm họ tên và số điện thoại của các nhân viên

Câu lệnh:

SELECT hoten as [Họ và tên], dienthoai as [Số điện thoại]

FROM tbl_nhanvien

Kết quả:

	Họ và tên	Số điện thoại
1	Nguyễn Thị Hồng	0912345678
2	Trần Văn Mạnh	0878234561
3	Phạm Thúy Quỳnh	0903344586
4	Đào Đức Anh	0904345618
5	Ngô Huyền Trâm	0988765432

Truy vấn với từ khóa DISTINCT

Yêu cầu:

Đưa ra danh sách không trùng lặp các mã hàng đã được bán

Câu lệnh:

SELECT DISTINCT mahang as [Mã hàng]

FROM tbl_banhang

Kết quả:

	Mã hàng
1	H0001
2	H0002
3	H0003
4	H0004
5	H0005
6	H0006
7	H0009
8	H0010

Truy vấn với điều kiện đơn giản từ nhiều bảng

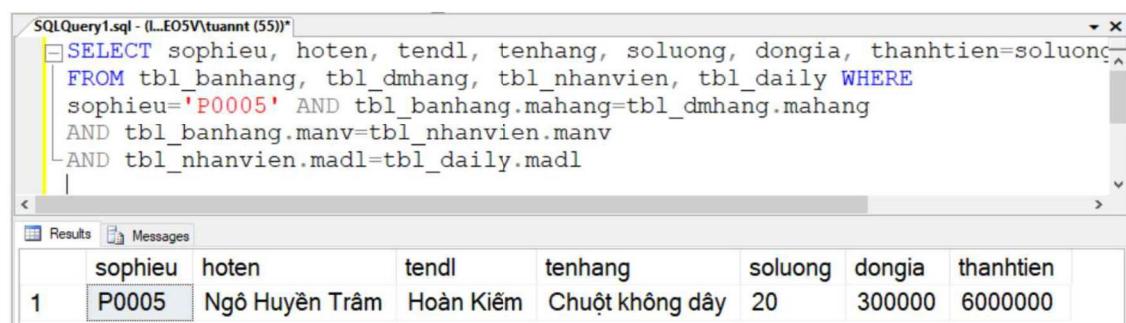
Yêu cầu:

Đưa ra thông tin chi tiết của phiếu bán hàng có số phiếu là ‘P0005’

Câu lệnh:

```
SELECT sophieu, hoten, tendl, tenhang, soluong, dongia,  
thanhtien=soluong*dongia  
  
FROM tbl_banhang, tbl_dmhang, tbl_nhanvien, tbl_daily  
  
WHERE sophieu='P0005' AND tbl_banhang.mahang=tbl_dmhang.mahang  
AND tbl_banhang.manv=tbl_nhanvien.manv AND  
tbl_nhanvien.madl=tbl_daily.madl
```

Kết quả:



The screenshot shows the SQL Query window with a query and its results. The query retrieves information from four tables: banhang, dmhang, nhanvien, and daily, filtering by the receipt number P0005. The results grid displays one row with the following data:

	sophieu	hoten	tendl	tenhang	soluong	dongia	thanhtien
1	P0005	Ngô Huyền Trâm	Hoàn Kiếm	Chuột không dây	20	300000	6000000

Truy vấn có điều kiện với kiểu ký tự

Yêu cầu:

Đưa ra thông tin gồm số phiếu, tên hàng, số lượng, thành tiền của các phiếu bán hàng Máy tính xách tay.

Câu lệnh:

```
SELECT sophieu, tenhang, soluong, dongia, thanhtien=soluong*dongia  
  
FROM tbl_banhang, tbl_dmhang  
  
WHERE tbl_banhang.mahang=tbl_dmhang.mahang AND  
tbl_dmhang.tenhang like N'Máy tính xách tay%'
```

Kết quả:

Results Messages

	sophieu	tenhang	soluong	dongia	thanhtien
1	P0001	Máy tính xách tay Dell	3	20000000	60000000
2	P0002	Máy tính xách tay Toshiba	6	18000000	108000000
3	P0003	Máy tính xách tay lenovo	15	15000000	225000000
4	P0008	Máy tính xách tay Dell	8	20000000	160000000
5	P0009	Máy tính xách tay Toshiba	40	18000000	720000000
6	P0014	Máy tính xách tay Toshiba	4	18000000	72000000
7	P0015	Máy tính xách tay lenovo	9	15000000	135000000

Chú ý:

Sử dụng toán tử LIKE trong mệnh đề WHERE. Với ký hiệu “ % ” thay cho một chuỗi con, dấu cách “ _ ” thay cho một ký tự nào đó.

A%B	Tìm một chuỗi bất kỳ bắt đầu là A và kết thúc là B.
%B	Tìm một chuỗi bất kỳ kết thúc là B.
A%	Tìm một chuỗi bất kỳ bắt đầu là A.
A_B	Tìm chuỗi gồm 3 ký tự với ký tự đầu A, ký tự cuối là B và ký tự giữa bất kỳ
_B	Tìm chuỗi gồm 2 ký tự với ký tự cuối là B.
A_	Tìm chuỗi gồm 2 ký tự với ký tự đầu là A, ký tự thứ hai là bất kỳ

Truy vấn có điều kiện với kiểu thời gian

Yêu cầu:

Đưa ra các phiếu bán hàng trong quý 2 năm 2020.

Câu lệnh:

SELECT *

FROM tbl_banhang

WHERE year(ngay)=2020 AND month(ngay)>=4 AND month(ngay)<=6

Kết quả:

	sophieu	ngay	manv	mahang	soluong
1	P0003	2020-04-06 00:00:00	N03	H0003	15
2	P0011	2020-06-28 00:00:00	N04	H0005	14
3	P0015	2020-05-15 00:00:00	N05	H0003	9

Truy vấn có sắp xếp dữ liệu

Yêu cầu:

Đưa ra các phiếu bán hàng trong tháng 1 năm 2020 theo số lượng hàng giảm dần.

Câu lệnh:

```
SELECT *
```

```
FROM tbl_banhang
```

```
WHERE year(ngay)=2020 AND month(ngay)=1
```

```
ORDER BY soluong DESC
```

Kết quả:

	sophieu	ngay	manv	mahang	soluong
1	P0002	2020-01-20 00:00:00	N01	H0002	6
2	P0001	2020-01-10 00:00:00	N02	H0001	3

Truy vấn có giá trị dữ liệu trong một khoảng

Yêu cầu:

Đưa ra các phiếu bán hàng có số lượng hàng trong khoảng từ 4 đến 8

Câu lệnh:

```
SELECT *
```

```
FROM tbl_banhang
```

```
WHERE soluong BETWEEN 4 AND 8
```

Kết quả:

	sophieu	ngay	manv	mahang	soluong
1	P0002	2020-01-20 00:00:00	N01	H0002	6
2	P0008	2019-08-13 00:00:00	N03	H0001	8
3	P0010	2020-11-19 00:00:00	N04	H0004	6
4	P0012	2020-11-04 00:00:00	N01	H0009	5
5	P0014	2020-02-07 00:00:00	N03	H0002	4

Chú ý:

Đối với một tập hợp dữ liệu có thể liệt kê được hoặc là kết quả của một truy vấn, ta có thể dùng toán tử IN hoặc NOT IN để kiểm tra giá trị có thuộc tập hợp đó hay không.

Truy vấn theo nhóm (GROUP BY)

Yêu cầu:

Đưa ra mã hàng, tên hàng, tổng số lượng, tổng tiền của các mặt hàng đã bán

Câu lệnh:

```
SELECT tbl_banhang.mahang as [Mã hàng], tenhang as [Tên hàng],
sum(soluong) as [Tổng số lượng], sum(dongia*soluong) as [Tổng tiền]
FROM tbl_banhang, tbl_dmharg
WHERE tbl_banhang.mahang=tbl_dmharg.mahang
GROUP BY tbl_banhang.mahang, tenhang
```

Kết quả:

	Mã hàng	Tên hàng	Tổng số lượng	Tổng tiền
1	H0001	Máy tính xách tay Dell	11	220000000
2	H0002	Máy tính xách tay Toshiba	50	900000000
3	H0003	Máy tính xách tay lenovo	24	360000000
4	H0004	Máy tính để bàn	15	225000000
5	H0005	Máy in canon	14	84000000
6	H0006	Chuột không dây	20	6000000
7	H0009	Điện thoại Samsung	6	60000000
8	H0010	Điện thoại IPhone	12	24000000

Truy vấn có điều kiện theo nhóm (GROUP BY ... HAVING)

Yêu cầu:

Đưa ra mã hàng, tên hàng, tổng tiền, số lần bán của các mặt hàng đã được bán từ 2 lần trở lên.

Câu lệnh:

```
SELECT tbl_banhang.mahang as [Mã hàng], tenhang as [Tên hàng],  
sum(dongia*soluong) as [Tổng tiền], count(*) as [Số lần bán]  
  
FROM tbl_banhang, tbl_dmharg  
  
WHERE tbl_banhang.mahang=tbl_dmharg.mahang  
  
GROUP BY tbl_banhang.mahang, tenhang  
  
HAVING count(*)>=2
```

Kết quả:

	Mã hàng	Tên hàng	Tổng tiền	Số lần bán
1	H0001	Máy tính xách tay Dell	220000000	2
2	H0002	Máy tính xách tay Toshiba	900000000	3
3	H0003	Máy tính xách tay lenovo	360000000	2
4	H0004	Máy tính để bàn	225000000	2
5	H0009	Điện thoại Samsung	60000000	2
6	H0010	Điện thoại iPhone	24000000	2

Truy vấn lồng nhau

Yêu cầu:

Đưa ra danh sách các mặt hàng chưa được bán lần nào.

Câu lệnh:

```
SELECT *  
  
FROM tbl_dmharg  
  
WHERE mahang NOT IN (SELECT mahang FROM tbl_banhang)
```

Kết quả:

	mahang	tenhang	dvt	dongia
1	H0007	Bộ nhớ USB 32TB	Cái	500000
2	H0008	Tai nghe	Cái	200000

Truy vấn có sử dụng ALL

Yêu cầu:

Đưa ra danh sách phiếu bán hàng có số lượng hàng bán trong một lần lớn nhất.

Câu lệnh:

SELECT *

FROM tbl_banhang

WHERE soluong >= ALL (SELECT soluong FROM tbl_banhang)

Kết quả:

	sophieu	ngay	manv	mahang	soluong
1	P0009	2020-09-04 00:00:00	N05	H0002	40

Chú ý:

Ta có thể sử dụng các lượng từ ANY, SOME để kiểm tra điều kiện

2.5 CÁC LỆNH TẠO LẬP CƠ SỞ DỮ LIỆU, BẢNG, CHỈ MỤC

2.5.1 Các lệnh với cơ sở dữ liệu

a) Tạo cơ sở dữ liệu

Cú pháp đầy đủ:

CREATE DATABASE database_name

[ON

[<filespec>[,...n]]

[,<filegroup>[...n]]

]

[LOGON {<filespec>[,...n] }]

<filespec> ::=

```
[PRIMARY]
([NAME = logical_file_name,]
FILENAME = 'os_file_name'
[,SIZE = size]
[,MAXSIZE = {maxsize| UNLIMITED } ]
[,FILEGROWTH = growth_increment]
) [,...n]
```

```
<filegroup> ::=  
FILEGROUP  
filegroup_name<filespec>[,...n]
```

Cú pháp rút gọn:

```
CREATE DATABASE database_name
ON
( [PRIMARY]
[ NAME = logical_file_name , ]
FILENAME = 'os_file_name'
[, SIZE = size ]
[, MAXSIZE = { max_size | UNLIMITED } ]
[, FILEGROWTH = growth_increment ] )
```

Trong đó:

- *<filespec>*: Mô tả các thông số của file chứa dữ liệu hoặc file chứa log của MicroSoft SQL Server. File chứa dữ liệu có phần mở rộng là MDF và NDF, file chứa log có phần mở rộng là LDF.
- *<filegroup>*: Mô tả thông số của nhóm file để chứa dữ liệu trong trường hợp tạo cơ sở dữ liệu được lưu trữ trên nhiều file.

Các thông số chi tiết:

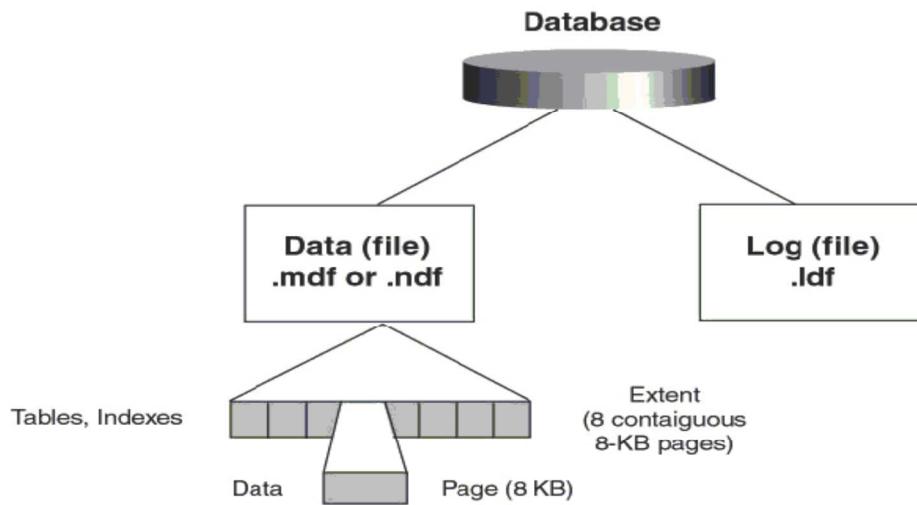
- *Name*: tên logic của cơ sở dữ liệu
- *Filename*: xác định đường dẫn và tên file để lưu trữ dữ liệu trong cơ sở dữ liệu (tên file và đường dẫn tuân theo các quy định của Hệ điều hành cài đặt MicroSoft SQL Server)
- *Size*: kích thước ban đầu của cơ sở dữ liệu
- *Maxsize*: xác định kích thước tối đa của cơ sở dữ liệu (UNLIMITED: tùy ý, phụ thuộc vào dung lượng đĩa còn trống của ổ đĩa chứa file dữ liệu)
- *Filegrowth*: cho phép độ tăng trưởng về kích thước cơ sở dữ liệu mỗi khi dung lượng hiện tại của cơ sở dữ liệu được dành sẵn đã hết (vượt quá giới hạn của 1 block dữ liệu mỗi khi cập nhật dữ liệu cho cơ sở dữ liệu).
- Đơn vị đo cho Size, Maxsize và Filegrowth: KB, MB, GB, TB

Chú ý:

Khi thực hiện lệnh tạo cơ sở dữ liệu, nếu không khai báo các thông số trên, MicroSoft SQL Server sẽ lấy các tham số của cơ sở dữ liệu mẫu trong hệ thống có tên là **model** và các thông số khác sẽ ngầm định theo hệ thống.

Về cấu trúc lưu trữ của các file trong MicroSoft SQL Server, dữ liệu và những đối tượng của cơ sở dữ liệu được lưu trong những file của hệ điều hành, gồm có:

- Primary data file*(.mdf): chứa bảng hệ thống và đối tượng dữ liệu
- Secondary data file(.ndf): chứa các đối tượng dữ liệu
- Transaction log file*(.ldf): ghi lại các thao tác lên cơ sở dữ liệu



File hệ thống có thể được lưu trên nhiều ổ đĩa, khi đó cần phải tạo ra nhóm file (filegroups). File và nhóm file chỉ thuộc duy nhất một cơ sở dữ liệu. Các file log không thuộc bất kỳ filegroup nào. Đối với các filegroup có thể được gắn các thuộc tính READONLY, READWRITE, DEFAULT.

Ví dụ:

Tạo cơ sở dữ liệu BANHANG, cơ sở dữ liệu này được đặt trong thư mục C:\QLBH, kích thước ban đầu của cơ sở dữ liệu là 100MB, kích thước tối đa không hạn chế, mỗi lần tăng trưởng 10MB.

Câu lệnh:

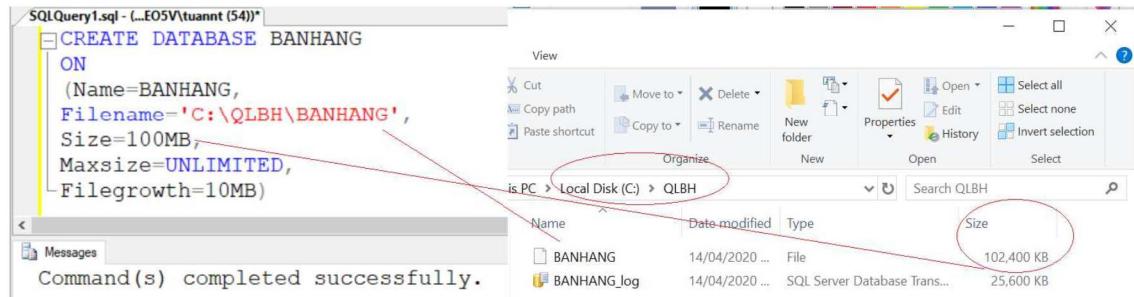
```
CREATE DATABASE BANHANG
```

```
ON
```

```
(
```

```
Name=BANHANG,  
Filename='C:\QLBH\BANHANG',  
Size=100MB,  
Maxsize=UNLIMITED,  
Filegrowth=10MB
```

```
)
```



b) Xem thông tin và chọn cơ sở dữ liệu

Với những cơ sở dữ liệu đã được tạo và đang được MicroSoft SQL Server quản lý, ta có thể xem các thông số của cơ sở dữ liệu đó, bao gồm những thông tin như: Tên cơ sở dữ liệu, tên logic của cơ sở dữ liệu, kích thước, ngày giờ tạo, người tạo, vị trí lưu trữ và tên của các file cơ sở dữ liệu, tình trạng hoạt động...

Cú pháp:

`EXEC sp_helpdb <database_name>`

Trong đó, `<database_name>` là tên cơ sở dữ liệu muốn xem thông tin.

Ví dụ:

Xem thông tin cơ sở dữ liệu BANHANG

Câu lệnh:

`EXEC sp_helpdb BANHANG`

Kết quả:

EXEC sp_helpdb BANHANG						
	name	db_size	owner	dbid	created	status
1	BANHANG	125.00 MB	DESKTOP-DJ0EO5V\tuannt	13	Apr 14 2020	Status=ONLINE, Updateability
	name	fileid	filename	filegroup	size	maxsize
1	BANHANG	1	C:\QLBH\BANHANG	PRIMARY	102400 KB	Unlimited
2	BANHANG_log	2	C:\QLBH\BANHANG_log.LDF	NULL	25600 KB	2147483648 KB
						10%

Khi thao tác khai thác cơ sở dữ liệu, tạo các đối tượng trong cơ sở dữ liệu hoặc lập trình trong cơ sở dữ liệu, ta cần phải chọn cơ sở dữ liệu sẽ làm việc. MicroSoft

SQL Server cho phép thực hiện chọn cơ sở dữ liệu bằng 2 cách: sử dụng lệnh hoặc chọn trực tiếp trên giao diện MicroSoft SQL Server Management Studio. Cơ sở dữ liệu khi bắt đầu chạy MicroSoft SQL Server Management Studio ngầm định là **master**.

Cú pháp:

`USE <database_name>`

Trong đó, `<database_name>` là tên cơ sở dữ liệu sẽ được lựa chọn làm việc.

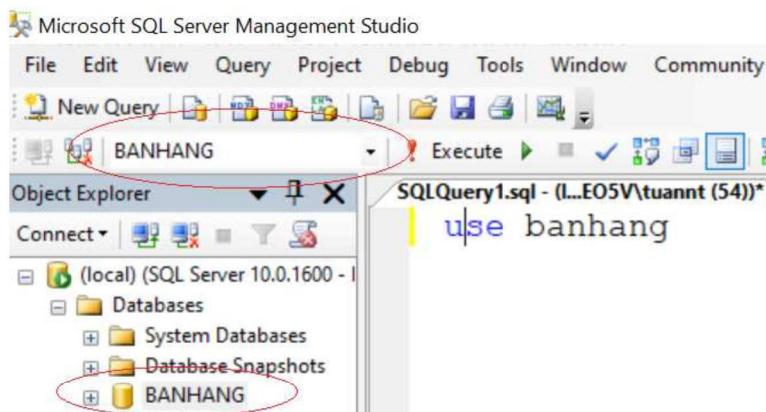
Ví dụ:

Chọn để làm việc với cơ sở dữ liệu BANHANG

Câu lệnh:

`USE BANHANG`

Kết quả:



Trong quá trình thực hiện chuỗi các lệnh SQL, đặc biệt là các lệnh thao tác với cơ sở dữ liệu, thông thường sẽ gửi tín hiệu kết thúc chuỗi các lệnh này đến máy chủ để các bộ đệm được làm mới dữ liệu, MicroSoft SQL Server cung cấp lệnh cho phép thực hiện thao tác này.

Cú pháp:

`GO`

Lệnh sẽ gửi tín hiệu kết thúc 1 chuỗi lệnh SQL đến SQL Server.

c) Sửa, xóa cơ sở dữ liệu

Để thay đổi các thông số của cơ sở dữ liệu như tên, các file của cơ sở dữ liệu và các thuộc tính khác của cơ sở dữ liệu, thực hiện lệnh theo cú pháp dưới đây.

Cú pháp:

```
ALTER DATABASE <database_name> { | MODIFY NAME =  
new_database_name | <file_and_filegroup_options> |  
<set_database_options> }
```

Ví dụ:

```
ALTER DATABASE BANHANG  
MODIFY NAME = QUANLYBANHANG
```

Để xóa cơ sở dữ liệu, MicroSoft SQL Server yêu cầu cơ sở dữ liệu đó không đang được chọn sử dụng, hay nói cách khác là phải ở ngoài cơ sở dữ liệu đó. Thực hiện lệnh xóa cơ sở dữ liệu theo cú pháp dưới đây.

Cú pháp:

```
DROP DATABASE <database_name>
```

Chú ý:

Lệnh xóa cơ sở dữ liệu sẽ xóa toàn bộ các đối tượng, dữ liệu trong cơ sở dữ liệu và không còn khả năng khôi phục được cơ sở dữ liệu

2.5.2 Các lệnh với bảng

a) Tạo bảng

Cú pháp:

```
CREATE TABLE   table_name  
(<column_definition1>[, <column_definition2>,..])
```

Ý nghĩa:

Tạo bảng trong cơ sở dữ liệu đang được sử dụng với các trường được định nghĩa trong danh sách.

Trong đó:

<column_definition> cho phép định nghĩa từng trường của bảng. Với mỗi trường sẽ được định nghĩa với cấu trúc:

<tên trường> <Kiểu dữ liệu> [<ràng buộc>]

<ràng buộc> sẽ được MicroSoft SQL Server kiểm tra cho trường mỗi khi có thao tác cập nhật dữ liệu vào bảng. Mỗi ràng buộc được định nghĩa theo cấu trúc:

[constraint <tên ràng buộc>] <ràng buộc>

Loại ràng buộc	Ý nghĩa
NULL / NOT NULL	Sử dụng để đảm bảo dữ liệu của cột có/không được nhận giá trị NULL
DEFAULT	Gán giá trị mặc định trong trường hợp dữ liệu của cột không được nhập vào hay không được xác định.
UNIQUE	Sử dụng để đảm bảo dữ liệu của cột là duy nhất, không trùng lặp giá trị trên cùng 1 cột.
PRIMARY KEY (Khóa chính)	Dùng để thiết lập khóa chính trên bảng, xác định giá trị trên tập các cột làm khóa chính phải là duy nhất, không được trùng lặp. Việc khai báo ràng buộc khóa chính yêu cầu các cột phải NOT NULL.
[FOREIGN KEY] [REFERENCES] (Khóa ngoại)	Dùng để thiết lập khóa ngoại trên bảng, tham chiếu đến bảng khác thông qua giá trị của cột được liên kết. Giá trị của cột được liên kết phải là duy nhất trong bảng kia.
CHECK	Bảo đảm tất cả giá trị trong cột thỏa mãn điều kiện nào đó. Đây là hình thức sử dụng phổ biến để kiểm tra tính hợp lệ của dữ liệu (validate data)

+ Đối với PRIMARY KEY:

- Nếu một trường sử dụng làm khóa chính cho bảng khi đó viết từ khóa PRIMARY KEY ngay sau khai báo trường
- Nếu sử dụng nhiều trường làm khóa chính, khi đó sử dụng cấu trúc PRIMARY KEY (danh sách các trường) sau khi khai báo tất cả các trường trong bảng.

+ Đối với CHECK

- CHECK(<bíểu thức logic>): dữ liệu sẽ được cập nhật vào bảng khi <bíểu thức logic> cho giá trị TRUE, ngược lại, MicroSoft SQL Server sẽ sinh lỗi cập nhật dữ liệu
- Chỉ sử dụng được cho các trường trong bảng

+ Đối với [FOREIGN KEY] [REFERENCES] (Khóa ngoại):

- Được sử dụng để đảm bảo tính toàn vẹn dữ liệu
- Khóa ngoại có dạng: <tên bảng>(tên trường khóa)

Ví dụ 1:

Yêu cầu tạo bảng Đại lý (tbl_daily)

Câu lệnh:

```
CREATE TABLE tbl_daily
```

```
(
```

```
    madl char(3) PRIMARY KEY,  
    tendl nvarchar(100) not null,  
    diachi nvarchar(100) not null
```

```
)
```

Ví dụ 2:

Yêu cầu tạo bảng Danh mục hàng (tbl_dmhang), có kiểm tra điều kiện cho đơn giá.

Câu lệnh:

```
CREATE TABLE tbl_dmhang
```

```
(  
    mahang char(5) primary key,  
    tenhang nvarchar(100) not null,  
    dvt nvarchar(20) not null,  
    dongia float check (dongia>0)  
)
```

Ví dụ 3:

Yêu cầu tạo bảng Nhân viên (tbl_nhanvien), bảng Bán hàng (tbl_banhang) có kiểm tra điều kiện và tham chiếu đến các bảng khác.

Câu lệnh:

--- Tao bang nhan vien

```
CREATE TABLE tbl_nhanvien
```

```
(  
    manv char(3) primary key,  
    madl char(3) references tbl_daily(madl),  
    hoten nvarchar(30) not null,  
    dienThoai char(10) check (dienThoai like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')  
)
```

--- Tao bang ban hang

```
CREATE TABLE tbl_banhang
```

```
(  
    sophieu char(5) primary key,  
    ngay smalldatetime not null,  
    manv char(3) references tbl_nhanvien(manv),  
    mahang char(5) references tbl_dmhang(mahang),
```

soluong float check (soluong>0)
)

Xem thông tin của đối tượng trong cơ sở dữ liệu (bảng, view...)

Cú pháp:

EXEC SP_HELP <tên đối tượng>

Ý nghĩa:

Thể hiện thông tin về các đặc tính của đối tượng cần xem, ví dụ: người tạo, các trường hoặc các thuộc tính tương đương, các ràng buộc...

Ví dụ:

Xem thông tin bảng Bán hàng

Câu lệnh:

EXEC sp_help tbl_banhang

Kết quả:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, a database named 'BANHANG' is selected. In the center pane, a query window displays the command: 'EXEC sp_help tbl_banhang'. Below the command, the results are shown in a table. The table has columns: Name, Owner, Type, Created_datetime, Column_name, Type, Computed, Length, Prec, Scale, Nullable, TrimTrailingBlanks, Fixed, Identity, Seed, Increment, Not For Replication, and RowGuidCol. The table data is as follows:

Name	Owner	Type	Created_datetime	Column_name	Type	Computed	Length	Prec	Scale	Nullable	TrimTrailingBlanks	Fixed	Identity	Seed	Increment	Not For Replication	RowGuidCol
tbl_banhang	dbo	user table	2020-04-15 10:41:19.193	sophieu	char	no	5			no	no	no	No identity column defined.	NULL	NULL	NULL	No rowguidcol column defined.
				ngay	smalldatetime	no	4			no	(n/a)	(n/a)					
				manv	char	no	3			yes	no	yes					
				mahang	char	no	5			yes	no	yes					
				soluong	float	no	8	53	NU...	yes	(n/a)	(n/a)					

At the bottom of the results pane, it says 'Query executed successfully.'

b) Sửa cấu trúc bảng, xóa bảng

MicroSoft SQL Server cho phép thực hiện sửa đổi cấu trúc của bảng đã được tạo bao gồm thêm, sửa, xóa các trường và các ràng buộc trong bảng. Đồng thời cũng có thể xóa bảng đã tạo trong cơ sở dữ liệu. Dưới đây là một số cú pháp thực hiện các thao tác trên.

Sửa trường:

ALTER TABLE <tên bảng>

ALTER COLUMN <tên trường>

<định nghĩa trường mới>

Thêm trường:

ALTER TABLE <tên bảng>

ADD <định nghĩa trường mới>

Xoá trường:

ALTER TABLE <tên bảng>

DROP COLUMN <tên trường>

Xoá bảng:

DROP TABLE <tên bảng>

Chú ý:

+ Sau khi xoá bảng, không còn khả năng khôi phục dữ liệu của bảng.

+ Đối với bảng có các ràng buộc với các bảng khác, chỉ có thể xóa được khi vẫn đảm bảo tính toàn vẹn của dữ liệu.

2.5.3 Thêm, sửa, xóa dữ liệu trong bảng

Thêm bản ghi:

Cú pháp:

INSERT INTO <tên bảng>[(danh sách trường)] VALUES (các dữ liệu)

hoặc

INSERT INTO <tên bảng>[(danh sách trường)] <Câu lệnh SELECT>

Ý nghĩa:

Thêm một bản ghi cho bảng qua các dữ liệu được đưa vào hoặc thêm các bản ghi vào bảng từ kết quả của một truy vấn bằng lệnh SELECT

Chú ý:

Các dữ liệu phải phù hợp với thứ tự của các trường trong bảng hoặc trong danh sách trường được liệt kê.

Ví dụ:

Thêm 3 bản ghi vào bảng Đại lý

Câu lệnh:

`INSERT INTO tbl_daily`

`VALUES`

`('D01', N'Bách Khoa', N'10 Phố Tạ Quang Bửu, Bách Khoa, HN'),`

`('D02', N'Phố Vọng', N'234 Phố Vọng, Đồng Tâm, HN'),`

`('D03', N'Hoàn Kiếm', N'2, Tràng Tiền, Hoàn Kiếm, HN')`

Sửa bản ghi:

Cú pháp:

`UPDATE <tên bảng> SET <tên trường1>=<giá trị1> [,<tên trường2>=<giá trị2>] WHERE <điều kiện>`

Ý nghĩa:

Cập nhật dữ liệu cho các trường của các bản ghi thỏa mãn điều kiện

Ví dụ:

Sửa địa chỉ của Đại lý có mã 'D03' thành '120, Phố Hué, Hà Nội'

Câu lệnh:

`UPDATE tbl_daily`

`SET`

Địa chỉ=N'120, Phó Hué, Hà Nội'

WHERE madl='D03'

Xóa bản ghi:

Cú pháp:

`DELETE <tên bảng> WHERE <điều kiện>`

Ý nghĩa:

Xóa các bản ghi thỏa mãn điều kiện.

Ví dụ:

Xóa Đại lý có mã 'D02'

Câu lệnh:

`DELETE tbl_daily`

WHERE madl='D02'

2.5.4 Các lệnh với chỉ mục (index)

Index trong MicroSoft SQL Server là một trong những yếu tố quan trọng nhất góp phần vào việc nâng cao hiệu suất của cơ sở dữ liệu. Index trong MicroSoft SQL Server tăng tốc độ của quá trình truy vấn dữ liệu bằng cách cung cấp phương pháp truy xuất nhanh chóng tới các dòng trong các bảng. Khi tìm kiếm dữ liệu, đầu tiên SQL Server sẽ tìm giá trị này trong Index, sau đó nó sử dụng Index để nhanh chóng xác định vị trí của dòng dữ liệu cần tìm. Nếu không có Index, MicroSoft SQL Server sẽ thực hiện động tác quét qua toàn bộ bảng (table scan) để xác định vị trí dòng cần tìm, việc quét dữ liệu là một trong những động tác có hại nhất cho hiệu suất của MicroSoft SQL Server.

Index trong MicroSoft SQL Server có thể tạo trên hầu hết các cột trong bảng hoặc View. Ngoại trừ các cột dùng để lưu trữ các đối tượng dữ liệu lớn như kiểu Image, Text... Ta cũng có thể tạo Index trên các cột kiểu XML, nhưng Index trên XML khác so với Index cơ bản nói chung và ít được dùng. Index trong SQL Server

được tạo thành từ một tập hợp các page (các Index Node) và chúng được tổ chức trong một cấu trúc có tên gọi là B-tree. Cấu trúc này là cấu trúc kiểu thứ bậc trong tự nhiên, với các nút gốc ở trên cùng của hệ thống phân cấp và các nút lá ở phía dưới.

Clustered index là loại index theo đó các bản ghi trong bảng được sắp thứ tự theo trường index. Khi bảng được tạo clustered index thì bản thân nó trở thành một cây index, với các node lá chứa khóa là các trường được index và cũng đồng thời chứa tất cả các trường còn lại của bảng. Vì các bản ghi chỉ có thể được sắp xếp trên cây index theo một thứ tự nhất định nên mỗi bảng chỉ có thể có tối đa một clustered index. Khi bảng đã có clustered index thì các index khác (nonclustered) sẽ dùng khóa của trường clustered index làm con trỏ để trả về bản ghi tương ứng. Ngoài ra, còn có một số kiểu index mở rộng khác như Composite index, Unique index...

Cú pháp tạo Index:

```
CREATE [ UNIQUE ] [ CLUSTERED | NONCLUSTERED ] INDEX  
index_name
```

```
    ON <object> ( column [ ASC | DESC ] [ ,...n ] )  
    [ INCLUDE ( column_name [ ,...n ] ) ]  
    [ WHERE <filter_predicate> ]
```

<object> ::=

```
{  
    [ database_name. [ schema_name ] . | schema_name. ]  
    table_or_view_name  
}
```

<filter_predicate> ::=

```
<conjunct> [ AND <conjunct> ]
```

<conjunct> ::=

<disjunct> | <comparison>

<disjunct> ::=

column_name IN (*constant* ,...)

<comparison> ::=

column_name <*comparison_op*> *constant*

<comparison_op> ::=

{ IS | IS NOT | = | <> | != | > | >= | !> | < | <= | !< }

Trong đó:

- UNIQUE, CLUSTERED, NONCLUSTERED: lựa chọn loại index sẽ tạo.
- *index_name*: Tên index
- *column*: Tên của một hoặc nhiều trường cho index.
- [ASC | DESC]: xác định cột được index sẽ sắp tăng dần hay giảm dần. Giá trị ngầm định là tăng dần.
- INCLUDE (*column* [,... *n*]) : chỉ ra những cột không phải khóa sẽ được thêm vào nonclustered index.
- WHERE <filter_predicate>: xác định điều kiện để lựa chọn các bản ghi được index. Khi sử dụng điều kiện này, bắt buộc phải dùng nonclustered index trên bảng.
- *database_name*: tên cơ sở dữ liệu
- *schema_name*: tên lược đồ chứa bảng hoặc view
- *table_or_view_name*: tên bảng hoặc view để tạo index

Cú pháp sửa Index:

```

ALTER INDEX { index_name | ALL }
    ON <object>
    { REBUILD
        [ [PARTITION = ALL]
            [ WITH ( <rebuild_index_option> [ ,...n ] ) ]
        | [ PARTITION = partition_number
            [ WITH ( <single_partition_rebuild_index_option>
                [ ,...n ] )
            ]
        ]
    | DISABLE
    | REORGANIZE
        [ PARTITION = partition_number ]
        [ WITH ( LOB_COMPACTION = { ON | OFF } ) ]
    | SET ( <set_index_option> [ ,...n ] )
    }
[ ; ]

<object> ::=

{
    [ database_name. [ schema_name ] . | schema_name. ]
        table_or_view_name
}

```

<rebuild_index_option> ::=

```

{
    PAD_INDEX = { ON | OFF }
    | FILLFACTOR = fillfactor
    | SORT_IN_TEMPDB = { ON | OFF }
    | IGNORE_DUP_KEY = { ON | OFF }
    | STATISTICS_NORECOMPUTE = { ON | OFF }
    | ONLINE = { ON | OFF }
    | ALLOW_ROW_LOCKS = { ON | OFF }
    | ALLOW_PAGE_LOCKS = { ON | OFF }
    | MAXDOP = max_degree_of_parallelism
    | DATA_COMPRESSION = { NONE | ROW | PAGE }
        [ ON PARTITIONS ( { <partition_number_expression> | <range> }
            [, ...n] ) ]
}
<range> ::= 
<partition_number_expression> TO <partition_number_expression>
}

<single_partition_rebuild_index_option> ::= 
{
    SORT_IN_TEMPDB = { ON | OFF }
    | MAXDOP = max_degree_of_parallelism
    | DATA_COMPRESSION = { NONE | ROW | PAGE } }
}

<set_index_option>::=

```

```
{  
    ALLOW_ROW_LOCKS = { ON | OFF }  
    | ALLOW_PAGE_LOCKS = { ON | OFF }  
    | IGNORE_DUP_KEY = { ON | OFF }  
    | STATISTICS_NORECOMPUTE = { ON | OFF }  
}
```

Ý nghĩa:

Lệnh cho phép sửa index cho bảng hoặc view bằng các thao tác: không cho phép (disabling), tái tạo lại (rebuilding) hoặc tổ chức lại (reorganizing) index hoặc là đặt lại các thông số cho index.

Cú pháp xóa Index:

DROP INDEX

index_name ON <object>

<object> ::=

```
{
```

[*database_name*. [*schema_name*] . | *schema_name*.]

table_or_view_name

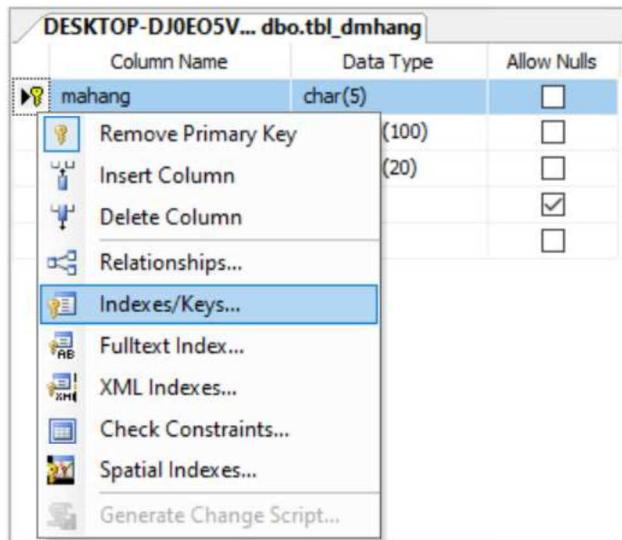
```
}
```

Ý nghĩa:

Xóa index của bảng hoặc view.

Chú ý:

Các thao tác tạo, sửa, xóa index có thể được thực hiện trong bộ công cụ MicroSoft SQL Server Management Studio khi ta tạo, sửa các bảng hoặc view.



2.6 KHUNG NHÌN (VIEW)

Các bảng trong cơ sở dữ liệu đóng vai trò là các đối tượng tổ chức và lưu trữ dữ liệu. Như vậy, ta có thể quan sát được dữ liệu trong cơ sở dữ liệu bằng cách thực hiện các truy vấn trên bảng dữ liệu. Ngoài ra, MicroSoft SQL Server còn cho phép chúng ta quan sát được dữ liệu thông qua việc định nghĩa các khung nhìn.

Một khung nhìn (view) có thể được xem như là một bảng “ảo” trong cơ sở dữ liệu có nội dung được định nghĩa thông qua một truy vấn (câu lệnh SELECT). Như vậy, một khung nhìn trông giống như một bảng với một tên khung nhìn và là một tập bao gồm các dòng và các cột. Điểm khác biệt giữa khung nhìn và bảng là khung nhìn không được xem là một cấu trúc lưu trữ dữ liệu tồn tại trong cơ sở dữ liệu. Thực chất dữ liệu quan sát được trong khung nhìn được lấy từ các bảng thông qua câu lệnh truy vấn dữ liệu. MicroSoft SQL Server sẽ lưu trữ các khung nhìn bằng các lệnh truy vấn trong cơ sở dữ liệu. Khi trong câu truy vấn xuất hiện khung nhìn, hệ quản trị cơ sở dữ liệu sẽ dựa vào định nghĩa của khung nhìn để chuyển yêu cầu truy vấn dữ liệu liên quan đến khung nhìn thành yêu cầu tương tự trên các bảng gốc và việc truy vấn dữ liệu được thực hiện bởi yêu cầu tương đương trên các bảng.

Việc sử dụng khung nhìn trong cơ sở dữ liệu đem lại các lợi ích sau đây:

- **Bảo mật dữ liệu:** Người sử dụng được cấp quyền trên các khung nhìn với những phần dữ liệu mà người sử dụng được phép. Điều này hạn chế được phần nào việc người sử dụng truy cập trực tiếp dữ liệu.
- **Đơn giản hóa các thao tác truy vấn dữ liệu:** Một khung nhìn đóng vai trò như là một đối tượng tập hợp dữ liệu từ nhiều bảng khác nhau vào trong một “bảng”. Nhờ vào đó, người sử dụng có thể thực hiện các yêu cầu truy vấn dữ liệu một cách đơn giản từ khung nhìn thay vì phải đưa ra những câu truy vấn phức tạp.
- **Tập trung và đơn giản hóa dữ liệu:** Thông qua khung nhìn ta có thể cung cấp cho người sử dụng những cấu trúc đơn giản, dễ hiểu hơn về dữ liệu trong cơ sở dữ liệu đồng thời giúp cho người sử dụng tập trung hơn trên những phần dữ liệu cần thiết.
- **Độc lập dữ liệu:** Một khung nhìn có thể cho phép người sử dụng có được cái nhìn về dữ liệu độc lập với cấu trúc của các bảng trong cơ sở dữ liệu cho dù các bảng cơ sở có bị thay đổi phần nào về cấu trúc.
- **Tránh dư thừa dữ liệu:** Khi thiết kế cơ sở dữ liệu, các bảng và các quan hệ trong cơ sở dữ liệu đã được chuẩn hóa, tránh dư thừa dữ liệu một cách tối đa. Tuy nhiên, trong thực tế rất nhiều các thao tác, bài toán cần được thực hiện trên những dữ liệu phái sinh, có thể được tính toán từ dữ liệu gốc. Khung nhìn cho phép khai thác các dữ liệu đó thông qua các câu lệnh truy vấn và không lưu dữ liệu cần tính trong các bảng.

Tuy nhiên, việc sử dụng khung nhìn cũng tồn tại một số nhược điểm sau:

- Do hệ quản trị cơ sở dữ liệu thực hiện việc chuyển đổi các truy vấn trên khung nhìn thành những truy vấn trên các bảng gốc nên nếu một khung nhìn được định nghĩa bởi một truy vấn phức tạp thì sẽ dẫn đến chi phí về mặt thời gian khi thực hiện truy vấn liên quan đến khung nhìn sẽ lớn.
- Mặc dù thông qua khung nhìn có thể thực hiện được thao tác bổ sung và cập nhật dữ liệu cho bảng cơ sở nhưng chỉ hạn chế đối với những khung nhìn đơn giản. Đối với những khung nhìn phức tạp thì thường không thực hiện được, hay nói cách khác là dữ liệu trong khung nhìn là chỉ đọc.

a) Tạo khung nhìn

Cú pháp:

```
CREATE VIEW view_name [ (column [ ,...n ] ) ]  
[ WITH [ ENCRYPTION ] [ SCHEMABINDING ] [ VIEW_METADATA ] ]  
AS select_statement
```

Ý nghĩa:

Tạo một view để lấy dữ liệu từ các bảng

Trong đó:

- *view_name*: Tên khung nhìn
- *column*: Tên các cột/trường của khung nhìn, đây là tham số tùy chọn. Trong trường hợp sử dụng các tham số này, cần phải có các tên tương ứng với các cột của kết quả lệnh SELECT trong khung nhìn.
- ENCRYPTION: Mã hóa các mục trong sys.syscomments có chứa văn bản của câu lệnh CREATE VIEW
- SCHEMABINDING: Đảm bảo các định nghĩa bảng không bị chỉnh sửa để không ảnh hưởng tới VIEW.
- VIEW_METADATA: Đảm bảo SQL Server có metadata của VIEW để các ứng dụng có thể khai thác được

Chú ý:

+ Nếu khung nhìn được định nghĩa bằng truy vấn phức tạp thì thời gian thực hiện sẽ lớn.

+ Mặc dù thông qua khung nhìn có thể thực hiện được thao tác bổ sung và cập nhật dữ liệu cho bảng cơ sở nhưng chỉ hạn chế đối với những khung nhìn đơn giản. Đối với những khung nhìn phức tạp thì thường không thực hiện được.

+ Trong khung nhìn không dùng:

- COMPUTE hoặc COMPUTE BY
- ORDER BY, ngoại trừ có một mệnh đề TOP trong lệnh SELECT

- Từ khóa INTO
 - Mệnh đề OPTION
 - Không tham chiếu đến bảng tạm hoặc biến bảng
- + Từ một khung nhìn này có thể gọi các khung nhìn khác.
- + Sử dụng khung nhìn tương tự như sử dụng một bảng thông thường, ngoại trừ các thao tác cập nhật dữ liệu.

Ví dụ 1:

Yêu cầu:

Tạo khung nhìn cho từng lần bán hàng, thông tin lấy ra bao gồm Mã hàng, Tên hàng, Thành tiền

Câu lệnh:

```
CREATE VIEW v_tinhthien([Mã hàng],[Tên hàng],[Thành tiền])
```

AS

```
SELECT tbl_dmhang.mahang, tenhang, soluong*dongia
FROM tbl_dmhang, tbl_banhang
WHERE tbl_dmhang.mahang=tbl_banhang.mahang
```

Sau khi tạo khung nhìn v_tinhthien, sử dụng lệnh truy vấn để xem kết quả:

```
SELECT *
```

```
FROM v_tinhthien
```

Kết quả:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, under the database 'BANHANG', a new view 'v_tinhtien' has been created, indicated by a red oval. In the center pane, a query window titled 'SQLQuery1.sql - (...EO5V\luannt (53))' contains the following T-SQL code:

```

CREATE VIEW v_tinhtien([Mã hàng],[Tên hàng],[Thành tiền])
AS
SELECT tbl_dmhong.mahang, tenhang, soluong*dongia
FROM tbl_dmhong, tbl_banhong
WHERE tbl_dmhong.mahang=tbl_banhong.mahang
select * from v_tinhtien

```

Below the code, the 'Results' tab displays a table with the following data:

	Mã hàng	Tên hàng	Thành tiền
1	H0001	Máy tính xách tay Dell	60000000
2	H0002	Máy tính xách tay Toshiba	108000000
3	H0003	Máy tính xách tay lenovo	225000000
4	H0004	Máy tính để bàn	135000000
5	H0006	Chuột không dây	6000000
6	H0010	Điện thoại iPhone	20000000
7	H0009	Điện thoại Samsung	10000000
8	H0001	Máy tính xách tay Dell	160000000
9	H0002	Máy tính xách tay Toshiba	720000000
10	H0004	Máy tính để bàn	00000000

A message at the bottom of the results pane says 'Query executed successfully.'

Ví dụ 2:

Yêu cầu:

Sử dụng khung nhìn v_tinhtien để tính tổng số tiền theo từng mặt hàng.

Câu lệnh:

CREATE VIEW v_tonghop

AS

```

SELECT [Mã hàng],[Tên hàng], sum([Thành tiền]) as [Tổng tiền]
FROM v_tinhtien
GROUP BY [Mã hàng],[Tên hàng]

```

Sau khi tạo khung nhìn v_tonghop, sử dụng lệnh truy vấn để xem kết quả:

SELECT *

FROM v_tonghop

Kết quả:

```

-- GROUP BY [Mã hàng], [Tên hàng]
select * from v_tinhien
select * from v_tonghop

```

	Mã hàng	Tên hàng	Thành tiền
1	H0001	Máy tính xách tay Dell	60000000
2	H0002	Máy tính xách tay Toshiba	108000000

	Mã hàng	Tên hàng	Tổng tiền
1	H0001	Máy tính xách tay Dell	220000000
2	H0002	Máy tính xách tay Toshiba	900000000
3	H0003	Máy tính xách tay lenovo	360000000
4	H0004	Máy tính để bàn	225000000
5	H0005	Máy in canon	84000000
6	H0006	Chuột không dây	6000000
7	H0009	Điện thoại Samsung	60000000
8	H0010	Điện thoại IPhone	24000000

b) Sửa, xóa khung nhìn

Cú pháp sửa khung nhìn:

```

ALTER VIEW view_name [ (column [ ,...n ] ) ]
[ WITH [ ENCRYPTION ] [ SCHEMABINDING ] [ VIEW_METADATA ] ]
AS select_statement

```

Cú pháp xóa khung nhìn:

```
DROP VIEW view_name
```

CHƯƠNG III. LẬP TRÌNH TRONG SQL SERVER

3.1 GIỚI THIỆU

Ngôn ngữ SQL được thiết kế và cài đặt như là một ngôn ngữ để thực hiện các thao tác trên cơ sở dữ liệu như tạo lập các cấu trúc trong cơ sở dữ liệu, bổ sung, cập nhật, xoá và truy vấn dữ liệu trong cơ sở dữ liệu. Các câu lệnh SQL được người sử dụng viết và yêu cầu hệ quản trị cơ sở dữ liệu thực hiện theo chế độ tương tác. Các câu lệnh SQL có thể được nhúng vào trong các ngôn ngữ lập trình, thông qua đó chuỗi các thao tác trên cơ sở dữ liệu được xác định và thực thi nhò vào các câu lệnh, các cấu trúc điều khiển của bản thân ngôn ngữ lập trình được sử dụng.

Trong MicroSoft SQL Server, Transaction SQL (T-SQL) là ngôn ngữ phát triển nâng cao của ngôn ngữ SQL chuẩn. Nó là ngôn ngữ dùng để giao tiếp giữa ứng dụng và SQL Server. T-SQL các khả năng của ngôn ngữ định nghĩa dữ liệu - DDL và ngôn ngữ thao tác dữ liệu - DML của SQL chuẩn cộng với một số hàm mở rộng, các thủ tục lưu trữ (stored procedure) hệ thống và cấu trúc lập trình cho phép lập trình trên SQL Server được linh động hơn. MicroSoft SQL Server cho phép lập trình thủ tục lưu trữ - gọi tắt là thủ tục (stored procedure), hàm (function) và trigger.

Một thủ tục hoặc hàm, trigger là một đối tượng trong cơ sở dữ liệu bao gồm một tập nhiều câu lệnh SQL, câu lệnh T-SQL được nhóm lại với nhau thành một nhóm với những khả năng sau:

- Có thể sử dụng các cấu trúc điều khiển (IF, WHILE...).
- Có thể sử dụng các biến như trong ngôn ngữ lập trình nhằm lưu giữ các giá trị tính toán được, các giá trị được truy xuất được từ cơ sở dữ liệu.
- Một tập các câu lệnh SQL được kết hợp lại với nhau thành một khối lệnh bên trong một thủ tục.
- Một thủ tục có thể nhận các tham số truyền vào cũng như có thể trả về các giá trị thông qua các tham số (như trong các ngôn ngữ lập trình). Khi một thủ tục lưu trữ đã được định nghĩa, nó có thể được gọi thông qua tên thủ tục, nhận các tham số truyền vào, thực thi các câu lệnh SQL bên trong thủ tục và có thể trả về các giá trị sau khi thực hiện xong.

- Các thủ tục, các hàm có thể gọi đến các thủ tục, hàm khác để thực hiện.

Sử dụng các thủ tục lưu trữ trong cơ sở dữ liệu sẽ giúp tăng hiệu năng của cơ sở dữ liệu, mang lại các lợi ích sau:

- Đơn giản hóa các thao tác trên cơ sở dữ liệu nhờ vào khả năng module hóa các thao tác này.
- Thủ tục lưu trữ được phân tích, tối ưu khi tạo ra nên việc thực thi chúng nhanh hơn nhiều so với việc phải thực hiện một tập rời rạc các câu lệnh SQL tương đương theo cách thông thường.
- Thủ tục lưu trữ cho phép chúng ta thực hiện cùng một yêu cầu bằng một câu lệnh đơn giản thay vì phải sử dụng nhiều dòng lệnh SQL. Điều này sẽ làm giảm thiểu sự lưu thông trên mạng.
- Thay vì cấp phát quyền trực tiếp cho người sử dụng trên các câu lệnh SQL và trên các đối tượng cơ sở dữ liệu, ta có thể cấp phát quyền cho người sử dụng thông qua các thủ tục lưu trữ, nhờ đó tăng khả năng bảo mật đối với hệ thống.

3.2 LẬP TRÌNH THỦ TỤC LUU TRỮ

3.2.1 Tạo thủ tục lưu trữ

a) Tạo thủ tục:

Cú pháp:

```
CREATE { PROCEDURE | PROC } procedure_name
[ @parameter datatype
[ VARYING ] [ = default ] [ OUT | OUTPUT | READONLY ]
, @parameter datatype
[ VARYING ] [ = default ] [ OUT | OUTPUT | READONLY ] ]
[ WITH { ENCRYPTION | RECOMPILE | EXECUTE AS Clause } ]
[ FOR REPLICATION ]
AS
<sql_statements>
```

Ý nghĩa:

Lệnh tạo thủ tục lưu trữ trong cơ sở dữ liệu đang được sử dụng.

Trong đó:

- procedure_name: Tên thủ tục
- @parameter: Một hay nhiều tham số được truyền vào hàm.
- Datatype: Kiểu dữ liệu cho @parameter.
- Default: Giá trị mặc định gán cho @parameter.
- OUT/OUTPUT: @parameter là một tham số đầu ra
- READONLY: @parameter không thể bị procedure ghi đè lên.
- VARYING: Chỉ sử dụng cho kiểu con trỏ (cursor), có thể làm tham số đầu ra.
- ENCRYPTION: Mã nguồn (source) của procedure sẽ không được lưu trữ dưới dạng text trong hệ thống.
- RECOMPILE: Truy vấn sẽ không được lưu ở bộ nhớ đệm (cache) cho thủ tục.
- EXECUTE AS clause: Xác định ngữ cảnh bảo mật để thực thi thủ tục.
- FOR REPLICATION: Procedure đã lưu sẽ chỉ được thực thi trong quá trình replication (nhân bản).

Ví dụ:

Lập một thủ tục để đưa ra tổng của 2 số a, b

Câu lệnh:

```
CREATE PROC sp_tonghaiso
```

```
AS
```

```
    DECLARE @a float, @b float
```

```
    SET @a=30
```

```
    SET @b=50
```

```
    PRINT N'Tổng hai số là'
```

```
    PRINT @a+@b
```

Sau khi nhập mã lệnh của thủ tục, bôi đen toàn bộ mã lệnh và thực hiện chạy lệnh tạo thủ tục bằng cách nhấn phím F5 hoặc chọn nút Execute trên thanh công cụ của MicroSoft SQL Server Management Studio.

Thực hiện thủ tục:

```
EXEC sp_tonghaiso
```

Kết quả:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, under the database 'BANHANG', the 'Programmability' folder is expanded, showing 'Stored Procedures'. A red oval highlights this folder. Inside it, 'sp_tonghaiso' is listed. Another red oval highlights the 'Messages' window at the bottom right, which displays the output of the executed query: 'Tổng hai số là 80|'. The code in the main query editor is as follows:

```
CREATE PROC sp_tonghaiso
AS
    DECLARE @a float, @b float
    SET @a=30
    SET @b=50
    PRINT N'Tổng hai số là'
    PRINT @a+@b
```

```
EXEC sp_tonghaiso
```

3.2.2 Thực thi thủ tục lưu trữ

Truyền giá trị cho thủ tục

Cách 1: Gán giá trị ngầm định ngay khi tạo thủ tục

Cách 2: Truyền các giá trị cho tham số chưa có giá trị ngầm định hoặc chỉ truyền cho một số tham số của thủ tục, thông thường sử dụng khi khai báo thủ tục có giá trị ngầm định cho tham số. Khi đó, với mỗi tham số ta truyền trực tiếp giá trị cho tham số đó dưới dạng:

```
@parameter=value
```

Cách 3: Truyền giá trị cho tất cả các tham số của thủ tục: viết lần lượt các giá trị tương ứng với các tham số được khai báo trong thủ tục, phải phù hợp các kiểu dữ liệu và ý nghĩa của các tham số.

Thực thi thủ tục:

Cú pháp:

`EXEC[CUTE] procedure_name [<parameter_values>][@parameter=value..]`

Ý nghĩa:

Thực thi một thủ tục với các giá trị được truyền vào cho tham số. Kết quả hoặc trạng thái thực thi thủ tục sẽ được thể hiện trên màn hình. Trong trường hợp thực thi thủ tục có xuất hiện lỗi, hệ thống sẽ sinh lỗi mã lỗi trong biên toàn cục của MicroSoft SQL Server, đồng thời, trên giao diện của MicroSoft SQL Server Management Studio cũng sẽ báo để người lập trình có thể dò vết.

Ví dụ 1:

Lập thủ tục tính lương cho nhân viên, tham số đầu vào là số ngày công. Lương một ngày công là 350,000 đồng.

Câu lệnh:

```
CREATE PROC sp_tinhluong
```

```
    @songaycong float
```

```
AS
```

```
    PRINT N'Lương của nhân viên là'
```

```
    PRINT @songaycong * 350000
```

Thực hiện thủ tục:

```
EXEC sp_tinhluong 21
```

Kết quả:

```

SQLQuery1.sql - (...EO5V\tuannt (51))*
CREATE PROC sp_tinhluong @songaycong float
AS
    PRINT N'Lương của nhân viên là'
    PRINT @songaycong * 350000

EXEC sp_tinhluong 21

Messages
Lương của nhân viên là
7.35e+006

```

Lấy kết quả ra từ tham số:

- + Với các tham số muốn sử dụng để lấy kết quả ra sau khi thực thi thủ tục, khi khai báo tham số, thêm từ khóa OUTPUT/OUT ngay sau mỗi tham số.
- + Khi thực thi thủ tục, cần có các biến tương ứng với các tham số đầu ra để chứa giá trị và viết kèm theo từ khóa OUTPUT/OUT với biến đó trong danh sách giá trị/biến truyền vào cho thủ tục.
- + Một thủ tục có thể trả ra nhiều kết quả từ các tham số.

Ví dụ 2:

Lập thủ tục tính thành tiền cho một mặt hàng với tham số đầu vào là số lượng, đơn giá, thuế VAT (%). Đầu ra là thành tiền.

Câu lệnh:

```
CREATE PROC sp_thanhien
```

```

@soluong float, @dongia float, @VAT float,
@thanhtien float OUTPUT
```

AS

```
SET @thanhtien=@soluong * @dongia * ( 1+ @VAT/100)
```

Thực hiện thủ tục:

```
DECLARE @tt float
```

```
EXEC sp_thanhien 30, 50, 10, @tt OUT
```

PRINT @tt

Kết quả:

The screenshot shows a SQL query window titled "SQLQuery1.sql - (L...EO5V\tuannt (51))". The code defines a stored procedure "sp_thanhien" with parameters "@soluong float" and "@dongia float". It calculates the total price using the formula: SET @thanhtien=@soluong * @dongia * (1+ @VAT/100). Inside the procedure, it declares a variable "@tt float", executes the procedure with values 30, 50, 10, and prints the result. The output in the "Messages" pane shows the value 1650.

```
CREATE PROC sp_thanhien @soluong float, @dongia float,
AS
    SET @thanhtien=@soluong * @dongia * ( 1+ @VAT/100)

DECLARE @tt float
EXEC sp_thanhien 30, 50, 10, @tt OUT
PRINT @tt
```

Messages
1650

3.2.3 Một số lệnh cơ bản trong lập trình thủ tục

Khai báo biến

Cú pháp:

DECLARE @<tên biến> <kiểu dữ liệu> [=<giá trị ngầm định>]

Ý nghĩa:

Khai báo biến cục bộ trong thủ tục. Trong một lệnh, có thể khai báo nhiều biến với nhiều kiểu dữ liệu khác nhau và được viết cách nhau bằng dấu phẩy (,). Các kiểu dữ liệu được sử dụng trong lệnh khai báo là kiểu dữ liệu được MicroSoft SQL Server hỗ trợ.

Ví dụ:

Khai báo biến Họ tên thuộc kiểu ký tự có độ rộng 30 ký tự

DECLARE @hoten nvarchar(30)

Gán giá trị cho biến

Cú pháp 1:

SET @<tên biến>= <bíểu thức>

Cú pháp 2:

`SELECT @<tên biến>= <biểu thức>`

Ý nghĩa:

Tính toán giá trị của biểu thức và gán vào biến. Đôi với cú pháp 2, biến có thể được gán kết quả của câu lệnh SELECT lấy giá trị từ bảng hoặc view.

Ví dụ:

Gán ‘Nguyễn Văn Minh’ vào biến @hoten

`SET @hoten=N'Nguyễn Văn Minh'`

Trong biểu thức có thể sử dụng các toán tử thông dụng, có thể sử dụng các hằng, các biến và các hàm có sẵn của MicroSoft SQL Server hoặc các hàm do người dùng tự định nghĩa. Ngoài các toán tử cơ bản, MicroSoft SQL Server còn hỗ trợ một số các phép toán mở rộng và rút gọn như:

Toán tử	Mô tả
<code>+=</code>	Cộng và gán
<code>-=</code>	Trừ và gán
<code>*=</code>	Nhân và gán
<code>/=</code>	Chia và gán
<code>%=</code>	Phép lấy phần dư và gán
<code>&=</code>	Phép AND nhị phân và gán
<code>^=</code>	Phép XOR nhị phân và gán
<code> =</code>	Phép OR nhị phân và gán
<code>ALL</code>	Được dùng để so sánh một giá trị với toàn bộ các giá trị trong một bộ giá trị khác.
<code>AND</code>	Cho phép nhiều điều kiện được tồn tại trong một phát biểu lệnh SQL

ANY	Dùng để so sánh một toán tử với bất kỳ một toán tử nào đó trong list theo một điều kiện nhất định.
BETWEEN	Toán tử này sẽ tìm kiếm một bộ giá trị nằm giữa hai giá trị điều kiện (giá trị nhỏ nhất và giá trị lớn nhất)
IN	Dùng để so sánh một giá trị với một danh sách các giá trị (để cho biết giá trị so sánh có nằm trong danh sách các giá trị đó hay không).
NOT	Dùng để nghịch đảo các toán tử logic khác
OR	Được dùng để kết hợp nhiều điều kiện trong một phát biểu lệnh SQL
EXISTS	Để tìm sự tồn tại của một hàng trong một bảng mà thỏa mãn một điều kiện cụ thể
LIKE	Toán tử LIKE được sử dụng để so sánh một giá trị với các giá trị tương tự bởi sử dụng các ký tự wildcard (Là ký tự sử dụng để thể hiện một hay nhiều ký tự. Ví dụ như ký tự sao * thường dùng để thể hiện một hay nhiều ký tự, ký tự ? thường dùng để thể hiện một ký tự nào đó.)

Thể hiện giá trị dữ liệu

Cú pháp 1:

PRINT <biểu thức>

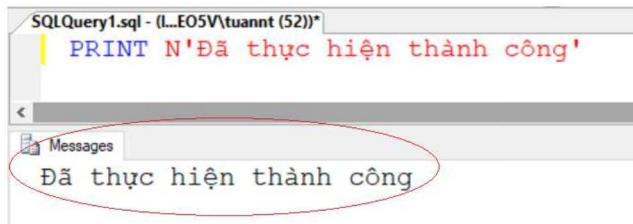
Sử dụng lệnh PRINT, mỗi lệnh chỉ in được một dữ liệu trên màn hình dưới dạng message. Do vậy, khi muốn thể hiện nhiều dữ liệu bằng lệnh PRINT, thường sẽ phải chuyển về cùng kiểu dữ liệu ký tự.

Ví dụ:

Hiện thông báo ‘Đã thực hiện thành công’ lên màn hình

PRINT N’ Đã thực hiện thành công’

Kết quả:



A screenshot of the SQL Server Management Studio interface. The query window title is "SQLQuery1.sql - (L...EO5V\tuannt (52))". The query text is: "PRINT N'Dã thực hiện thành công'". Below the query window, the "Messages" tab is selected, showing the output: "Đã thực hiện thành công". A red oval highlights the output text.

Cú pháp 2:

SELECT <danh sách biểu thức>

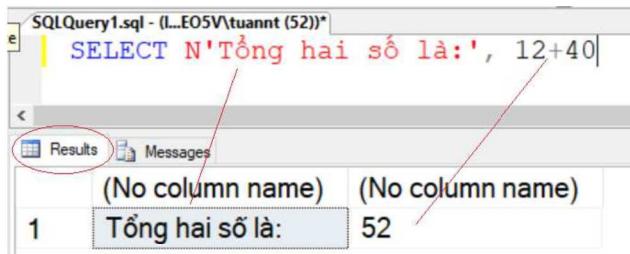
Sử dụng lệnh SELECT, có thể dùng để thể hiện dữ liệu cho nhiều biểu thức hoặc biến có các kiểu dữ liệu khác nhau. Kết quả lệnh được thể hiện dạng grid trên màn hình.

Ví dụ:

Hiện thông báo ‘Tổng hai số là’, $12+40$

SELECT N'Tổng hai số là:', 12+40

Kết quả:



A screenshot of the SQL Server Management Studio interface. The query window title is "SQLQuery1.sql - (L...EO5V\tuannt (52))". The query text is: "SELECT N'Tổng hai số là:', 12+40". Below the query window, the "Results" tab is selected, showing the output in a grid format. The grid has two columns: "(No column name)" and "(No column name)". The first row contains the text "Tổng hai số là:" and the value "52". A red oval highlights the "Results" tab.

(No column name)	(No column name)
Tổng hai số là:	52

Lệnh gộp/khối lệnh (statement block)

Trong MicroSoft SQL Server, khi lập trình sẽ có nhiều cấu trúc lệnh hoặc chương trình chỉ cho phép thực hiện một lệnh, do vậy, để có thể đưa nhiều lệnh vào cấu trúc lập trình, MicroSoft SQL Server cung cấp cấu trúc khối lệnh hay còn được gọi cách khác là lệnh gộp, khi đó các lệnh nằm trong cấu trúc này sẽ được MicroSoft SQL Server coi như một lệnh duy nhất.

Cú pháp:

BEGIN

<các lệnh>

END

Ví dụ:

Tạo một khối lệnh để thực thi chương trình tính thành tiền cho mặt hàng

Câu lệnh:

BEGIN

```
DECLARE @tt float  
EXEC sp_thanhien 100, 80, 10, @tt OUT  
PRINT @tt
```

END

3.2.4 Sửa, xóa thủ tục lưu trữ

Sửa thủ tục lưu trữ

Cú pháp:

ALTER PROC[EDURE] <tên thủ tục>

<mô tả thủ tục mới>

Ý nghĩa:

Cho phép sửa đổi các nội dung trong thủ tục đã được tạo.

Ví dụ:

Sửa thủ tục tính thành tiền cho mặt hàng, có kiểm tra điều kiện số lượng, đơn giá và VAT là các số lớn hơn hoặc bằng 0.

Câu lệnh:

CREATE PROC sp_thanhien

```
@soluong float, @dongia float, @VAT float,  
@thanhien float OUTPUT
```

AS

```
IF (@soluong>=0) AND (@dongia>=0) AND (@VAT>=0)
    SET @thanhtien=@soluong * @dongia * ( 1+ @VAT/100)
ELSE
    PRINT N'Tham số truyền vào không đúng'
```

Xóa thủ tục lưu trữ

Cú pháp:

DROP PROC[EDURE] <tên thủ tục>

Ý nghĩa:

Xóa thủ tục lưu trữ trong cơ sở dữ liệu đang được sử dụng.

Ví dụ:

Xóa thủ tục tính thành tiền cho mặt hàng.

Câu lệnh:

DROP PROC sp_thanhtien

3.2.5 Xem các dòng lệnh trong thủ tục lưu trữ

Cú pháp:

EXEC sp_helptext <tên đối tượng>

Ý nghĩa:

Cho phép xem nội dung hoặc các dòng lệnh của các đối tượng không mã hóa, bao gồm thủ tục, hàm, trigger, view, các trường tính toán, ràng buộc...

Ví dụ:

Xem nội dung của thủ tục sp_thanhtien

Câu lệnh:

EXEC sp_helptext sp_thanhtien

Kết quả:

```

SQLQuery1.sql - (L...EO5V\tuannt (52))*
EXEC sp_helptext sp_thanhien

Results Messages
Text
1 CREATE PROC sp_thanhien @soluong float, @donggia float, @VA...
2 AS
3 SET @thanhtien=@soluong * @donggia * ( 1+ @VAT/100)

```

3.3 CÁC CÂU TRÚC TRONG LẬP TRÌNH

3.3.1 Câu trúc rẽ nhánh

Cú pháp:

```

IF Boolean_expression
    { sql_statement 1 | statement_block 1}
[ ELSE
    { sql_statement 2| statement_block 2} ]

```

Ý nghĩa:

Câu trúc rẽ nhánh chương trình, khi thực hiện, giá trị biểu thức điều kiện sẽ được tính toán, nếu biểu thức điều kiện có giá trị TRUE thì sẽ thực hiện lệnh 1 hoặc khôi lệnh 1, ngược lại, sẽ thực hiện lệnh 2 hoặc khôi lệnh 2.

Chú ý:

- + Trong câu trúc có thể có không dùng ELSE
- + Trong các chương trình, có thể lồng nhiều lệnh IF vào nhau nhưng chúng không được giao nhau.

Ví dụ 1:

Lập thủ tục để tính thành tiền cho một mặt hàng. Tham số đầu vào gồm số lượng, đơn giá, VAT. Đầu ra là thành tiền cho mặt hàng. Nếu số lượng lớn hơn 15 thì đơn giá sẽ được giảm 5% cho toàn bộ mặt hàng.

Câu lệnh:

```

CREATE PROC sp_thanhien
    @soluong float, @dongia float, @VAT float,
    @thanhien float OUTPUT
AS
IF @soluong>15
    SET @thanhien=@soluong * @dongia * ( 1+ @VAT/100)*0.95
ELSE
    SET @thanhien=@soluong * @dongia * ( 1+ @VAT/100)

```

Ví dụ 2:

Lập thủ tục để kiểm tra 3 số truyền vào có là 3 cạnh của tam giác hay không, nếu là 3 cạnh của tam giác thì tam giác đó là tam giác gì?.

Câu lệnh:

```

CREATE PROC sp_tamgiac
    @a float, @b float, @c float, @ketluan nvarchar(100) OUTPUT
AS
IF (@a+@b>@c) AND (@a+@c>@b) AND (@b+@c>@a)
BEGIN
    IF (@a=@b) AND (@a=@c)
        SET @ketluan=N'Tam giác đều'
    ELSE
        IF (@a=@b) OR (@a=@c) OR (@b=@c)
            IF (@a * @a = @b * @b + @c * @c) OR
                (@b * @b = @a * @a + @c * @c) OR
                (@c * @c = @a * @a + @b * @b)
                    SET @ketluan=N'Tam giác vuông cân'
            ELSE

```

```

        SET @ketluan=N'Tam giác cân'
    ELSE
        SET @ketluan=N'Tam giác thường'
    END
ELSE
    SET @ketluan=N'Không là 3 cạnh tam giác'

```

Ví dụ 3:

Lập thủ tục để thêm một nhân viên mới vào bảng Nhân viên. Tham số đầu vào là các thông tin của nhân viên. Thủ tục có kiểm tra tính đúng đắn của dữ liệu đầu vào.

Câu lệnh:

```

CREATE PROC sp_themnhanvien
    @manv char(3), @madl char(3), @hoten nvarchar(30), @dienthoai
    char(10), @thongbao nvarchar(50) OUTPUT
AS
    IF NOT ((@manv IS NULL) OR (@madl IS NULL) OR (@hoten IS
NULL) OR (@dienthoai IS NULL))
        IF NOT EXISTS(SELECT * FROM tbl_nhanvien WHERE
manv=@manv)
            IF EXISTS((SELECT * FROM tbl_daily WHERE madl=@madl)
                IF @dienthoai like '[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-
9]')
                    BEGIN
                        INSERT INTO tbl_nhanvien
                        VALUES
                            (@manv, @madl, @hoten, @dienthoai)
                        SET @thongbao=N'Thành công'
                    END

```

```
ELSE SET @thongbao=N'Sai số điện thoại'  
ELSE SET @thongbao=N'Sai mã đại lý'  
ELSE SET @thongbao=N'Trùng mã nhân viên '
```

3.3.2 Hàm CASE

Trong MicroSoft SQL Server, ngoài lệnh rẽ nhánh IF trong các chương trình, khi tính toán hoặc lập trình, ta có thể sử dụng hàm CASE để lựa chọn các giá trị khác nhau dựa trên giá trị điều kiện hoặc biểu thức điều kiện. Cụ thể như sau:

Cú pháp 1:

```
CASE input_expression  
WHEN when_expression THEN result_expression [ ...n ]  
[ELSE else_result_expression]  
END
```

Cú pháp 2:

```
CASE  
WHEN Boolean_expression THEN result_expression [ ...n ]  
[ELSE else_result_expression]  
END
```

Ý nghĩa:

Đối với cú pháp 1 của hàm CASE, hàm sẽ trả về giá trị của biểu thức tương ứng với trường hợp giá trị của biểu thức nằm trong WHEN bằng với giá trị biểu thức cần kiểm tra, trong trường hợp tất cả các biểu thức WHEN không bằng giá trị kiểm tra thì hàm CASE sẽ trả về giá trị biểu thức được viết tại ELSE. Tương tự như vậy, cú pháp 2 của hàm CASE sẽ kiểm tra giá trị logic của biểu thức WHEN, nếu có giá trị TRUE thì trả về kết quả tương ứng cho CASE. Nếu kiểm tra tất cả các biểu thức WHEN mà không có trường hợp nào có giá trị TRUE thì biểu thức tại ELSE sẽ trả về cho hàm CASE.

Chú ý:

- + Cú pháp 1 của hàm CASE thường và chỉ dùng được cho các biểu thức có giá trị thuộc kiểu đếm được (hữu hạn), như kiểu số nguyên, ký tự...
- + Cú pháp 2 của hàm CASE có thể sử dụng cho kiểu dữ liệu bất kỳ, chỉ phụ thuộc vào các biểu thức logic trong WHEN.
- + Trong cả 2 cú pháp hàm CASE, có thể không có vế ELSE.
- + Có thể lồng các hàm CASE với nhau, nhưng không được giao nhau

Ví dụ 1:

Đưa ra danh sách nhân viên, trong đó thể hiện giới tính là Nam hoặc Nữ.

Câu lệnh:

SELECT manv as [Mã nhân viên], hoten as [Họ tên],

[Giới tính] = CASE gioitinh

WHEN 0 THEN N'Nam'

WHEN 1 THEN N'Nữ'

END,

dienthoai as [Điện thoại]

FROM tbl_nhanvien

Kết quả:

The screenshot shows the SQL Query window with the following query:

```
SQLQuery2.sql - (...EO5V\tuannt (52))*
SELECT manv as [Mã nhân viên], hoten as [Họ tên],
[Giới tính] = CASE gioitinh
    WHEN 0 THEN N'Nam'
    WHEN 1 THEN N'Nữ'
END,
dienthoai as [Điện thoại]
FROM tbl_nhanvien
```

The Results window displays the following table:

	Mã nhân viên	Họ tên	Giới tính	Điện thoại
1	N01	Nguyễn Thị Hồng	Nữ	0912345678
2	N02	Trần Văn Mạnh	Nam	0878234561
3	N03	Phạm Thúy Quỳnh	Nữ	0903344586
4	N04	Đào Đức Anh	Nam	0904345618
5	N05	Ngô Huyền Trâm	Nữ	0988765432

Ví dụ 2:

Lập một thủ tục có tên là sp_docso để chuyển đổi các số nguyên từ 0-9 thành các chữ tương ứng. Đầu vào là số nguyên từ 0-9, đầu ra của thủ tục là dãy ký tự chứa chữ tương ứng.

Câu lệnh:

```
CREATE PROC sp_docso
    @so smallint,
    @chu nvarchar(5) OUTPUT
AS
    SET @chu = CASE @so
        WHEN 0 THEN N'Không'
        WHEN 1 THEN N'Một'
        WHEN 2 THEN N'Hai'
        WHEN 3 THEN N'Ba'
        WHEN 4 THEN N'Bốn'
        WHEN 5 THEN N'Năm'
        WHEN 6 THEN N'Sáu'
        WHEN 7 THEN N'Bảy'
        WHEN 8 THEN N'Tám'
        WHEN 9 THEN N'Chín'
        ELSE ""
    END
```

Ví dụ 3:

Lập một thủ tục có tên là sp_xeploaihoc tap để xếp loại học tập cho học sinh khi biết điểm trung bình chung. Đầu vào của thủ tục là điểm trung bình chung (DTBC), đầu ra là xếp loại theo qui định sau:

DTBC<3: Yếu

3<=DTBC<5: Kém
5<=DTBC<6.5: Trung bình
6.5<=DTBC<7: Trung bình khá
7<=DTBC<8: Khá
8<=DTBC<9: Giỏi
9<=DTBC: Xuất sắc

Câu lệnh:

```
CREATE PROC sp_xeploaihocap
    @dtbc float,
    @xeploai nvarchar(30) OUTPUT
AS
    SET @xeploai = CASE
        WHEN (@dtbc<3) AND (@dtbc>=0) THEN N'Yếu'
        WHEN (@dtbc<5) AND (@dtbc>=3) THEN N'Kém'
        WHEN (@dtbc<6.5) AND (@dtbc>=5) THEN N'Trung bình'
        WHEN (@dtbc<7) AND (@dtbc>=6.5) THEN N'Trung bình khá'
        WHEN (@dtbc<8) AND (@dtbc>=7) THEN N'Khá'
        WHEN (@dtbc<9) AND (@dtbc>=8) THEN N'Giỏi'
        WHEN (@dtbc<=10) AND (@dtbc>=9) THEN N'Xuất sắc'
        ELSE ""
    END
```

Ví dụ 4:

Lập một thủ tục có tên là sp_docso0_99 để chuyển các số nguyên có giá trị từ 0 đến 99 thành dãy ký tự biểu diễn số đó. Yêu cầu sử dụng thủ tục sp_docso đã thực hiện tại Ví dụ 2 trong khi lập trình thủ tục này. Đầu vào của thủ tục là một số nguyên từ 0-99, đầu ra là dãy ký tự thể hiện số đó.

Câu lệnh:

```
CREATE PROC sp_docso0_99
    @so int, @ketqua nvarchar (30) out
AS
DECLARE @tam nvarchar (30)
DECLARE @so1 int, @so2 int
IF @so<0
    SET @ketqua= ""
ELSE
    IF (@so>=0) AND (@so<=9)
        EXEC sp_docso @so, @ketqua OUT
    ELSE
        IF (@so<20)
            BEGIN
                SET @so=@so % 10
                EXEC sp_docso @so, @tam OUT
                IF @so=0
                    SET @tam=""
                SET @ketqua= N'Mười ' + @tam
            END
        ELSE
            IF (@so<=99)
                BEGIN
                    SET @so1= @so/10
                    SET @so2 = @so % 10
                    EXEC sp_docso @so1 , @tam OUT
                END
            END
```

```

SET @ketqua=@tam

EXEC sp_docso @so2 , @tam OUT

IF @so2=0

    set @tam=""

SET @ketqua=@ketqua +N' Mươi '+ @tam

END

ELSE

SET @ketqua=""

```

3.3.3 Lệnh GOTO

MicroSoft SQL Server cho phép sử dụng lệnh GOTO để chuyển hướng chương trình vô điều kiện đến một vị trí nào đó được xác định bằng nhãn (label).

Cú pháp:

GOTO label

Trong đó, để xác định label, sử dụng cấu trúc:

label:

Chú ý:

- + label là điểm mà GOTO sẽ chuyển hướng chương trình đến để thực hiện, nó cũng có thể được sử dụng để chú thích (comment chương trình).

- + GOTO có thể nằm trong các cấu trúc lập trình khác, giữa các lệnh, các khối lệnh và trong các thủ tục, nhưng nó không thể chuyển đến vị trí ngoài lô lệnh (batch). GOTO có thể chuyển hướng đến một nhãn khai báo ở trước nó.

Ví dụ:

Lập một thủ tục với tham số đầu vào là một số nguyên, tham số đầu ra là kết quả, thủ tục thực hiện thao tác sau:

Nếu tham số đầu vào là 1: tính tổng tiền của tất cả các lần bán hàng

Nếu tham số đầu vào là 2: đưa ra số tiền lớn nhất của một lần bán hàng

Câu lệnh:

```
CREATE PROC sp_tinhtoan
@loai int, @ketqua float OUTPUT
AS
IF @loai=1
    GOTO tongtien
ELSE
    GOTO lonhat
tongtien:
SELECT
    @ketqua=sum(soluong*dongia)
FROM tbl_dmhang, tbl_banhang
WHERE tbl_dmhang.mahang=tbl_banhang.mahang
GOTO ketthuc
lonhat:
SELECT
    @ketqua=max(soluong*dongia)
FROM tbl_dmhang, tbl_banhang
WHERE tbl_dmhang.mahang=tbl_banhang.mahang
ketthuc:
```

Kết quả:

```

SQLQuery1.sql - (...E05\vtuannt (53))*
lonnhat:
SELECT
    @ketqua=max(soluong*dongia)
    FROM tbl_dmhang, tbl_banhang
    WHERE tbl_dmhang.[table BANHANG.dbo.tbl_banhang].hang.mahang
    ketthuc:

declare @kq float
EXEC sp_tinhtoan 1, @kq OUT
select @kq

```

	Results	Messages
	(No column name)	
1	1879000000	

3.3.4 Cấu trúc lặp

Cú pháp:

```

WHILE boolean_expression
{ sql_statement | statement_block }
[ BREAK ]
{ sql_statement | statement_block }
[ CONTINUE ]
{ sql_statement | statement_block }

```

Ý nghĩa:

Cấu trúc sẽ cho phép thực hiện các lệnh lặp đi lặp lại cho đến khi biểu thức điều kiện boolean_expression có giá trị FALSE thì sẽ kết thúc vòng lặp. Trong lập trình, có thể sử dụng nhiều vòng lặp WHILE lồng nhau, nhưng chúng không được giao nhau. Trong cấu trúc này cũng có thể sử dụng các cấu trúc lập trình khác. Trong một số trường hợp, có thể sử dụng từ khóa BREAK hoặc CONTINUE để xử lý các tình huống đặc biệt trong quá trình lặp.

- **BREAK:** Khi gặp từ khóa này, vòng lặp sẽ ngừng ngay lập tức mà không cần thực hiện hết các lệnh trong vòng lặp và không cần kiểm tra biểu thức điều kiện của vòng lặp.

- CONTINUE: Khi gặp từ khóa này, chương trình sẽ quay lại đầu vòng lặp ngay lập tức để bắt đầu một vòng lặp mới từ bước kiểm tra biểu thức điều kiện, khi đó các lệnh viết sau CONTINUE sẽ bị bỏ qua không thực hiện.

Ví dụ 1:

Lập một thủ tục để tính tổng các số $1^2 + 2^2 + \dots + n^2$, trong đó n là số nguyên dương.

Câu lệnh:

```
CREATE PROC sp_tinh tongso
```

```
@n int
```

```
AS
```

```
DECLARE @i int, @tong float
```

```
SET @i=1
```

```
SET @tong=0
```

```
WHILE @i<=@n
```

```
BEGIN
```

```
PRINT @i
```

```
SET @tong = @tong + @i * @i
```

```
SET @i=@i+1
```

```
END
```

```
PRINT N'Tổng là:'
```

```
PRINT @tong
```

Kết quả:

```

SQLQuery1.sql - (L...E05V\tnuant (53))*
    SET @i=1
    SET @tong=0
    WHILE @i<=@n
    BEGIN
        PRINT @i
        SET @tong = @tong + @i * @i
        SET @i=@i+1
    END
    PRINT N'Tổng là:'
    PRINT @tong

    EXEC sp_tinhhtongso 4

```

Messages

```

1
2
3
4
Tổng là:
30

```

Ví dụ 2:

Lập một thủ tục để tách các từ trong một dãy ký tự và in các từ trên màn hình.

Câu lệnh:

```
CREATE PROC sp_tachtu
```

```
    @cau nvarchar(100)
```

```
AS
```

```
    DECLARE @tmp nvarchar(100), @i int
```

```
    SET @tmp=LTRIM(RTRIM(@cau))
```

```
    WHILE LEN(@tmp)>0
```

```
        BEGIN
```

```
            SET @i=CHARINDEX(' ',@tmp)
```

```
            IF @i>0
```

```
                BEGIN
```

```
                    PRINT SUBSTRING(@tmp,1,@i-1)
```

```

        SET @tmp=LTRIM(RIGHT(@tmp,LEN(@tmp)-
    (@i)))
END
ELSE
IF LEN(@tmp)>0
BEGIN
PRINT @tmp
SET @tmp=""
END
END

```

Kết quả:

```

SQLQuery1.sql - (L...EO5V\tuannt (53))*
        END
    ELSE
        IF LEN (@tmp) >0
        BEGIN
            PRINT @tmp
            SET @tmp=''
        END
    END
EXEC sp_tachtu N'  Nguyễn   Trung Tuấn  '

```

Messages

Nguyễn
Trung
Tuấn

3.3.5 Kiểu dữ liệu con trỏ (CURSOR)

Con trỏ (CURSOR) trong cơ sở dữ liệu là một cấu trúc điều khiển cho phép duyệt qua các bản ghi trong các bảng, view. Trong các truy vấn T-SQL của thủ tục ta có thể sử dụng các con trỏ để duyệt qua dữ liệu. Con trỏ là một tập hợp kết quả truy vấn (các bản ghi/các hàng), với con trỏ ta có thể duyệt qua từng hàng kết quả để thi hành những tác vụ phức tạp. Con trỏ chỉ có thể tham chiếu một hàng tại một thời điểm, nhưng có thể di chuyển đến các hàng khác của bộ kết quả khi cần.

Khai báo con trỏ

Cú pháp khai báo con trỏ chuẩn ISO:

```
DECLARE cursor_name [ INSENSITIVE ] [ SCROLL ]  
CURSOR  
FOR select_statement  
[ FOR { READ ONLY | UPDATE [ OF column_name [ ,...n ] ] } ]
```

Ý nghĩa:

Khai báo biến con trỏ chỉ đến các hàng dữ liệu là kết quả của câu lệnh SELECT

Các tham số:

- cursor_name: tên con trỏ
- INSENSITIVE: tạo một bản sao dữ liệu là kết quả của truy vấn trong con trỏ để làm việc. Khi sử dụng tham số này, nếu các dữ liệu tại bảng gốc có sự thay đổi thì dữ liệu con trỏ không được cập nhật theo, tốc độ thực thi các câu lệnh liên quan đến con trỏ sẽ nhanh hơn nhưng đổi lại sẽ tốn bộ nhớ của hệ thống nhiều hơn.
- SCROLL: cho phép thực hiện các lệnh duyệt con trỏ một cách tự do theo cả hai chiều xuôi và ngược với các tham số FIRST, LAST, PRIOR, NEXT, RELATIVE, ABSOLUTE. Trường hợp không sử dụng SCROLL, con trỏ chỉ có thể duyệt theo hướng xuôi từ bản ghi/hàng đầu tiên đến hàng cuối cùng, không thể duyệt ngược.
- READ ONLY: Con trỏ chỉ đọc dữ liệu, không cho phép cập nhật dữ liệu qua con trỏ.
- UPDATE: xác định các trường của bảng dữ liệu gốc có thể được cập nhật qua con trỏ, các trường được liệt kê trong danh sách các trường kèm theo.

Các lệnh mở, đóng con trỏ:

Cú pháp mở Cursor:

```
OPEN { [ GLOBAL ] cursor_name } | cursor_variable_name }
```

Ý nghĩa:

Mở con trỏ có tên được chỉ ra trong lệnh để làm việc với con trỏ. Khi thực hiện mở con trỏ, MicroSoft SQL Server sẽ thực thi câu lệnh truy vấn của con trỏ để lấy dữ liệu từ các bảng/view gốc, đồng thời con trỏ sẽ ở bản ghi định của kết quả truy vấn.

Cú pháp đóng Cursor:

```
CLOSE { { [ GLOBAL ] cursor_name } | cursor_variable_name }
```

Ý nghĩa:

Đóng con trỏ sẽ tạm thời không cho phép thực hiện các thao tác với con trỏ đồng thời một số tài nguyên của hệ thống sẽ được giải phóng (ví dụ như bộ nhớ). Sau khi đóng con trỏ, có thể mở lại con trỏ bằng lệnh OPEN.

Cú pháp xóa Cursor khỏi bộ nhớ:

```
DEALLOCATE { { [ GLOBAL ] cursor_name } | @cursor_variable_name }
```

Ý nghĩa:

Lệnh sẽ đóng con trỏ và loại bỏ hoàn toàn con trỏ, bao gồm các tài nguyên con trỏ đã sử dụng và cả tên con trỏ. Khi dùng lệnh này, không còn khả năng mở lại con trỏ bằng lệnh OPEN. Lệnh này thường dùng khi kết thúc của các chương trình.

Sử dụng con trỏ:

Cú pháp lấy dữ liệu từ con trỏ:

FETCH

```
[ [ NEXT | PRIOR | FIRST | LAST  
    | ABSOLUTE { n | @nvar }  
    | RELATIVE { n | @nvar }  
]  
FROM
```

```
{ { [ GLOBAL ] cursor_name } | @cursor_variable_name }
```

```
[ INTO @variable_name [ ,...n ] ]
```

Ý nghĩa:

Lấy dữ liệu từ con trỏ vào các biến tương ứng

Các tham số:

- NEXT: Lấy dữ liệu ở bản ghi kế tiếp bản ghi hiện tại
- PRIOR: Lấy dữ liệu ở bản ghi trước bản ghi hiện tại
- FIRST: Lấy dữ liệu ở bản ghi đầu tiên của tập kết quả
- LAST: Lấy dữ liệu ở bản ghi cuối cùng của tập kết quả
- ABSOLUTE { n | @nvar }: Lấy dữ liệu ở bản ghi thứ n trong tập kết quả, bản ghi đầu tiên có thứ tự là 1.
- RELATIVE { n | @nvar }: Lấy dữ liệu ở bản ghi cách bản ghi hiện tại n bản ghi. Nếu $n > 0$, cách n bản ghi kế tiếp, $n < 0$, cách n bản ghi về trước.
- [INTO @variable_name [,...n]]: Lệnh sẽ nạp các dữ liệu của bản ghi được lấy vào các biến liệt kê trong danh sách. Danh sách các biến phải tương ứng (về số lượng, kiểu dữ liệu và thứ tự) với các cột liệt kê trong lệnh SELECT của con trỏ. Trong trường hợp không sử dụng tham số này, lệnh FETCH chỉ mang tính chất dịch chuyển con trỏ bản ghi.

Sau khi thực hiện lệnh FETCH, trạng thái thực hiện của lệnh được thể hiện qua hàm hệ thống @@FETCH_STATUS, hàm có các giá trị sau:

- 0: Lệnh thực hiện thành công
- -1: Câu lệnh FETCH không thực hiện thành công hoặc bản ghi cần lấy nằm ngoài tập kết quả của SELECT
- -2: Bản ghi cần lấy không có

Ví dụ 1:

Lập một thủ tục để đưa ra danh sách các mặt hàng theo yêu cầu sau: Các mặt hàng có số thứ tự chẵn trong danh sách được thể hiện trước, sau đó đến các mặt hàng có thứ tự lẻ trong danh sách ban đầu.

Câu lệnh:

```
CREATE PROC sp_danh sachhang
```

```
AS
```

```

DECLARE @mahang nvarchar(5), @tenhang nvarchar(100), @i int
DECLARE cs_dmhang SCROLL CURSOR
FOR
    SELECT mahang, tenhang
    FROM tbl_dmhang
OPEN cs_dmhang
--- In danh sach mat hang co so thu tu chan
SET @i=1
FETCH NEXT FROM cs_dmhang INTO @mahang, @tenhang
WHILE @@FETCH_STATUS=0
BEGIN
    IF @i % 2 =0
        PRINT @mahang + ‘‘ + @tenhang
    SET @i = @i+1
    FETCH NEXT FROM cs_dmhang INTO @mahang, @tenhang
END
--- In danh sach mat hang co so thu tu le
PRINT “
SET @i=1
FETCH FIRST FROM cs_dmhang INTO @mahang, @tenhang
WHILE @@FETCH_STATUS=0
BEGIN
    IF @i % 2 !=0
        PRINT @mahang + ‘‘ + @tenhang
    SET @i = @i+1
    FETCH NEXT FROM cs_dmhang INTO @mahang, @tenhang

```

```

END
CLOSE cs_dmharg
DEALLOCATE cs_dmharg

```

Kết quả:

```

code nhap du lieu...EO5V\tuannt (53)*
      SET @i = @i+1
      FETCH NEXT FROM cs_dmharg INTO @mahang, @tenhang
    END
    CLOSE cs_dmharg
    DEALLOCATE cs_dmharg

exec sp_danhsachhang

```

Messages

H0002	Máy tính xách tay Toshiba
H0004	Máy tính để bàn
H0006	Chuột không dây
H0008	Tai nghe
H0010	Điện thoại iPhone
H0001	Máy tính xách tay Dell
H0003	Máy tính xách tay lenovo
H0005	Máy in canon
H0007	Bộ nhớ USB 32TB
H0009	Điện thoại Samsung

Query executed successfully. | (local) (10.0 RTM) | DESKTOP-DJ0E | Ln 34

Ví dụ 2:

Cho một bảng quản lý điểm học sinh gồm các thông tin: mã học sinh, họ tên học sinh, điểm toán, điểm lý, điểm hóa. Hãy lập một thủ tục để tính và in ra danh sách gồm mã học sinh, họ tên, điểm trung bình chung, xếp loại học tập. Biết điểm trung bình chung = (điểm toán *2 + điểm lý + điểm hóa)/4. Xếp loại học sinh sử dụng thủ tục sp_xeploaihoc tap đã được tạo.

Câu lệnh:

```
CREATE PROC sp_danhsachhocsinh
```

```
AS
```

```

DECLARE @tbc float, @xeploai nvarchar(30)

DECLARE @mahs char(3), @hoten nchar(30), @toan float, @ly float,
@hoa float

```

```

DECLARE cs_hocsinh CURSOR
FOR
    SELECT *
    FROM tbl_diemhs
OPEN cs_hocsinh
FETCH NEXT FROM cs_hocsinh
INTO @mahs, @hoten, @toan, @ly, @hoa
WHILE @@FETCH_STATUS=0
BEGIN
    SET @tbc=(@toan*2+@ly+@hoa)/4
    EXEC sp_xeploaihoc tap @tbc, @xeploai OUT
    PRINT @mahs + ' ' + @hoten + ' ' + CAST(@tbc AS char(5)) +
    + @xeploai
    FETCH NEXT FROM cs_hocsinh
    INTO @mahs, @hoten, @toan, @ly, @hoa
END
CLOSE cs_hocsinh
DEALLOCATE cs_hocsinh

```

Kết quả:

```
tinh diem hsql ...EO5V(tuannt (52))*
|_ FETCH NEXT FROM cs_hocsinh
|   INTO @mabs, @hoten, @toan, @ly, @hoa
|_ WHILE @@FETCH_STATUS=0
|_ BEGIN
|_     SET @tbc=(@toan*2+@ly+@hoa)/4
|_     EXEC sp_xeploaihoc tap @tbc, @xeploai OUT
|_     PRINT @mabs + ' ' + @hoten + ' ' + CAST(@tbc AS char(5)) +' ' + @xeploai
|_   FETCH NEXT FROM cs_hocsinh
|   INTO @mabs, @hoten, @toan, @ly, @hoa
|_ END
|_ CLOSE cs_hocsinh
|_ DEALLOCATE cs_hocsinh
|
| EXEC sp_danh sachhoc sinh

< Messages >
001 Phạm Mỹ Linh          7      Khá
002 Trần Văn Mạnh        4.75   Kém
003 Hồ Thị Lý             3.25   Kém
004 Trương Văn Việt       6.5    Trung bình khá
005 Nguyễn Văn Tú          8      Giỏi

< Query executed successfully. | (local) (10.0 RTM) | DESKTOP-D\EO5V\tuannt... | BANHANG | 00:00:00 | 0 rows
```

3.3.6 Làm việc với dữ liệu XML

XML là ngôn ngữ đánh dấu, được dùng để miêu tả dữ liệu. Các thẻ (tag) trong XML chưa xác định trước. Người dùng tự định nghĩa trong quá trình tạo tài liệu XML. Trong thực tế XML được sử dụng để đóng gói và trao đổi dữ liệu giữa các hệ thống. MicroSoft SQL Server cho phép lưu trữ, xử lý dữ liệu dạng XML, giúp cho các nhà phát triển có thêm một lựa chọn trong việc xử lý dữ liệu định dạng XML. Nội dung trong phần này chỉ giới thiệu phương pháp đơn giản để sử dụng XML trong CSDL thông qua ví dụ cụ thể để:

- Tạo bảng có trường XML
 - Thêm dữ liệu vào bảng
 - Truy vấn dữ liệu

Các nội dung này sẽ được trình bày thông qua một ví dụ cụ thể.

Ví dụ:

- Tạo bảng sinh viên có các trường: ngành học, mã sinh viên, họ, tên đệm.
 - Thêm dữ liệu cho 4 sinh viên dưới dạng XML của 2 ngành học CNTT và HTTTQL.

- Dùng lệnh Select để hiển thị thông tin.
- Đưa ra thông tin sinh viên có mã SV3.

Câu lệnh tạo bảng sinh viên:

CREATE TABLE sinhvien

(nganhhoc char (10) primary key,
thongtin XML)

Câu lệnh thêm dữ liệu cho 4 sinh viên dưới dạng XML:

INSERT INTO sinhvien VALUES

```
('CNTT',N'<doc>
<s svid="SV1">
    <ho>Nguyễn</ho>
    <ten>Trung Tuấn</ten>
</s>
<s svid="SV2">
    <ho>Nguyễn</ho>
    <ten>Văn Minh</ten>
</s>
</doc>')
```

INSERT INTO sinhvien VALUES

```
('HTTTQL',N'<doc>
<s svid="SV3">
    <ho>Nguyễn</ho>
    <ten>Hùng Sơn</ten>
</s>
<s svid="Sv4">
    <ho>Trần</ho>
```

```

<ten>Thị Lan</ten>
</s>
</doc>')

```

Truy vấn dữ liệu:

Trong trường hợp sử dụng lệnh truy vấn thông thường, kết quả sẽ trả về các bản ghi và dữ liệu XML sẽ được thể hiện dưới dạng các siêu liên kết (hyperlink). Khi ta nhấp vào các liên kết đó, thông tin sẽ được trả ra dưới dạng mã XML.

	nganhhoc	thongtin
1	CNTT	<doc><s svid="SV1"><ho>Nguyễn</ho><ten>Trung Tuân</ten></s>
2	HTTTQL	<doc><s svid="SV3"><ho>Nguyễn</ho><ten>Hùng Sơn</ten></s>

Thông tin khi nhấp vào liên kết của kết quả SELECT

Thực hiện truy vấn tới thông tin của từng trường trong XML cần phải chỉ rõ theo cú pháp khai thác dữ liệu XML, ví dụ, đưa ra danh sách gồm ngành học, họ, tên đệm của tất cả các sinh viên, ta thực hiện câu lệnh sau:

SELECT

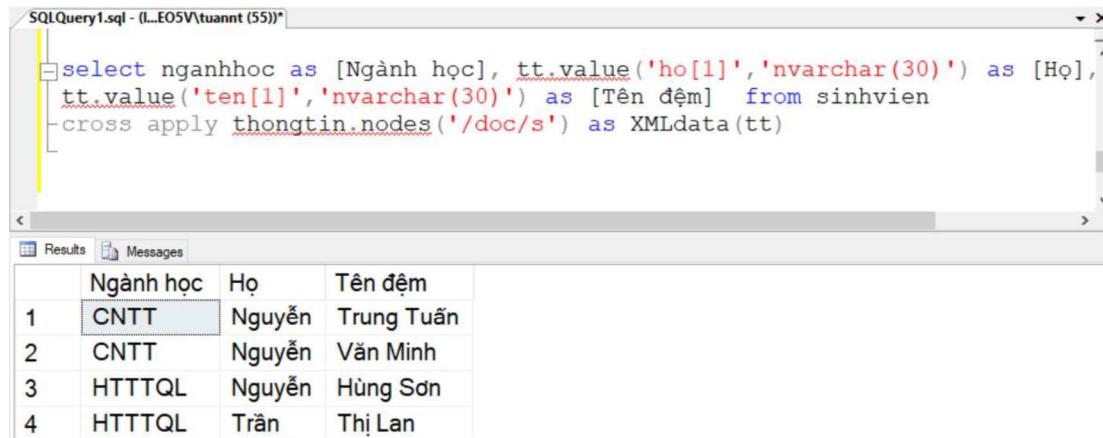
```
nganhhoc AS [Ngành học], tt.value('ho[1]', 'nvarchar(30)') AS [Họ],
```

```
tt.value('ten[1]', 'nvarchar(30)') AS [Tên đệm]
```

```
FROM sinhvien
```

CROSS APPLY thongtin.nodes('/doc/s') AS XMLdata(tt)

Kết quả thực hiện:



```
SQLQuery1.sql - (...EO5V\tuannt (55))*
select nganhhoc as [Ngành học], tt.value('ho[1]', 'nvarchar(30)') as [Họ],
tt.value('ten[1]', 'nvarchar(30)') as [Tên đệm] from sinhvien
cross apply thongtin.nodes('/doc/s') as XMLdata(tt)
```

	Ngành học	Họ	Tên đệm
1	CNTT	Nguyễn	Trung Tuân
2	CNTT	Nguyễn	Văn Minh
3	HTTTQL	Nguyễn	Hùng Sơn
4	HTTTQL	Trần	Thị Lan

Truy vấn có điều kiện, đưa ra thông tin của sinh viên có mã SV3:

SELECT

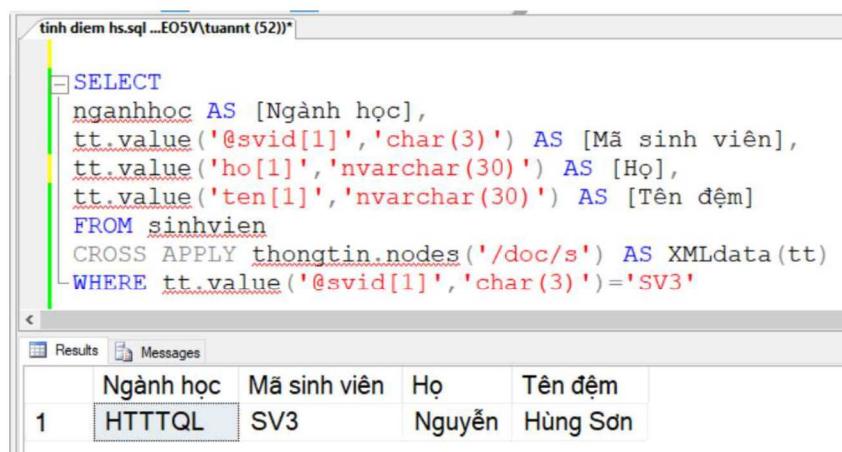
```
nganhhoc AS [Ngành học], tt.value('@svid[1]', 'char(3)') AS [Mã sinh
viên], tt.value('ho[1]', 'nvarchar(30)') AS [Họ],
tt.value('ten[1]', 'nvarchar(30)') AS [Tên đệm]
```

FROM sinhvien

CROSS APPLY thongtin.nodes('/doc/s') AS XMLdata(tt)

WHERE tt.value('@svid[1]', 'char(3)')='SV3'

Kết quả thực hiện:



```
tinh diem hs.sql - (...EO5V\tuannt (52))*
SELECT
nganhhoc AS [Ngành học],
tt.value('@svid[1]', 'char(3)') AS [Mã sinh viên],
tt.value('ho[1]', 'nvarchar(30)') AS [Họ],
tt.value('ten[1]', 'nvarchar(30)') AS [Tên đệm]
FROM sinhvien
CROSS APPLY thongtin.nodes('/doc/s') AS XMLdata(tt)
WHERE tt.value('@svid[1]', 'char(3)')='SV3'
```

	Ngành học	Mã sinh viên	Họ	Tên đệm
1	HTTTQL	SV3	Nguyễn	Hùng Sơn

3.3.7 Kiểm soát lỗi

a) Thông báo lỗi bằng RAISERROR

Sinh ra thông báo lỗi và khởi động quá trình xử lý lỗi cho phiên làm việc. RAISERROR có thể tham chiếu đến thông điệp được lưu trữ trong hệ thống hoặc tự tạo thông điệp một cách động. Thông điệp được trả về dưới dạng thông điệp lỗi của máy chủ tới ứng dụng được gọi hoặc tới cấu trúc TRY-CATCH.

Cú pháp:

```
RAISERROR ( { msg_id | msg_str | @local_variable }  
          { ,severity ,state }  
          [ ,argument [ ,...n ] ] )  
          [WITH option [ ,...n ] ]
```

Các tham số:

- *msg_id*: đây là số hiệu lỗi do người dùng định nghĩa được lưu trong danh sách sys.messages sử dụng sp_addmessage. Số hiệu lỗi này nên lớn hơn 50000. Nếu số hiệu lỗi không được chỉ ra, RAISERROR sẽ đưa ra thông báo lỗi với số hiệu là 50000.
- *msg_str*: là thông báo lỗi của người sử dụng có định dạng tương tự như hàm printf trong thư viện chuẩn của C. Thông báo này tối đa có 2,047 ký tự, nếu nhiều hơn thì hệ thống sẽ chỉ hiện 2,044 ký tự và kèm dấu chấm lửng. Khi xác định msg_str, lỗi sẽ được sinh ra với mã là 50000.
- *@local_variable*: là một biến kiểu ký tự, nó chứa chuỗi ký tự có định dạng được quy định như *msg_str*. Biến này phải có kiểu char hoặc varchar hoặc có thể chuyển đổi hoàn toàn sang những kiểu này.
- *severity*: là mức độ nghiêm trọng do người dùng định nghĩa tương ứng với thông điệp. Khi sử dụng *msg_id* để sinh thông điệp lỗi được tạo bằng sp_addmessage, mức độ nghiêm trọng xác định trên RAISERROR sẽ đè lên mức độ nghiêm trọng được xác định trong sp_addmessage. Mức độ nghiêm trọng từ 0 đến 18 có thể được bất kỳ người dùng nào xác định. Các mức từ 19

đến 25 chỉ có thể xác định bởi các thành viên quản trị hệ thống hoặc có quyền ALTER TRACE, đồng thời phải có tùy chọn WITH LOG. Các mức nhỏ hơn 0 được xác định là 0, các mức lớn hơn 25 được xác định là 25.

Chú ý:

- + Với các mức nghiêm trọng từ 20 đến 25 được coi là nguy hiểm, khi đó các client sẽ nhận được thông báo lỗi và bị ngắt kết nối, thông tin lỗi sẽ được ghi vào log của máy chủ và log của ứng dụng.
- + Có thể sử dụng -1 để trả về mức nghiêm trọng của lỗi.
- *State*: là số nguyên từ 0 đến 255. Nếu với cùng một lỗi được sinh ra ở nhiều vị trí, sử dụng số trạng thái duy nhất cho mỗi vị trí có thể giúp tìm thấy đoạn sinh ra lỗi.
- *argument*: là các tham số được sử dụng cho các biến được xác định trong *msg_str* hoặc trong thông điệp tương ứng với *msg_id*. Số lượng các tham số này không được vượt quá 20, và chỉ thuộc các kiểu **tinyint**, **smallint**, **int**, **char**, **varchar**, **nchar**, **nvarchar**, **binary**, hoặc **varbinary**.
- *Option*: là tùy chọn của người dùng cho lỗi, là một trong các giá trị sau:

Giá trị	Mô tả
LOG	Ghi log cho lỗi máy chủ và lỗi của ứng dụng. Lỗi này tối đa ghi được 440byte. Chỉ có người dùng là quản trị hệ thống hoặc có quyền ALTER TRACE mới có thể sử dụng WITH LOG.
NOWAIT	Gửi ngay lập tức thông điệp tới client.
SETERROR	Đặt giá trị của @@ERROR và ERROR_NUMBER là <i>msg_id</i> hoặc 50000, bất kể mức nghiêm trọng nào.

Ví dụ:

Sinh lỗi sai tham số cho thủ tục sp_Vidu, mã lỗi 15600.

Câu lệnh:

RAISERROR (15600,-1,-1, 'sp_Vidu')

Kết quả:

```

SQLQuery1.sql - (L...E05\vtuannt(52))*
RAISERROR (15600,-1,-1, 'sp_Vidu');

< Messages
Msg 15600, Level 15, State 1, Line 1
An invalid parameter or option was specified for procedure 'sp_Vidu'.

```

Một số hàm cung cấp thông tin về lỗi:

Tên hàm	Chức năng
ERROR_NUMBER()	Trả về mã lỗi (dưới dạng số)
ERROR_SEVERITY()	Trả về mức độ nghiêm trọng của lỗi
ERROR_STATE()	Trả về trạng thái của lỗi (dưới dạng số)
ERROR_PROCEDURE()	Trả về tên của Stored Procedure hoặc Trigger đã phát sinh lỗi
ERROR_LINE()	Trả về vị trí dòng lệnh phát sinh lỗi
ERROR_MESSAGE()	Trả về thông báo lỗi dưới dạng một đoạn ký tự

b) Xử lý lỗi bằng TRY-CATCH

MicroSoft SQL Server cho phép thực hiện bắt lỗi và xử lý các lỗi bằng các cặp câu trúc TRY-CATCH.

Cú pháp:

BEGIN TRY

{ *sql_statement* | *statement_block* }

END TRY

BEGIN CATCH

[{ *sql_statement* | *statement_block* }]

END CATCH

Ý nghĩa:

Cấu trúc TRY - CATCH bắt tất cả các lỗi thực thi có mức độ nghiêm trọng cao hơn 10 mà không đóng kết nối cơ sở dữ liệu. Khi có lỗi trong khối TRY, các lệnh trong khối CATCH sẽ thực hiện, ngược lại, các lệnh trong CATCH sẽ được bỏ qua.

Chú ý:

- + Hai cấu trúc này luôn đi liền nhau và không được nằm trong hai khối BEGIN - END khác nhau.
- + Cấu trúc này không được sử dụng trong định nghĩa Hàm

Ví dụ:

Các lệnh dưới đây sẽ chỉ ra cách sử dụng RAISERROR bên trong cấu trúc TRY để khi thực hiện sẽ nhảy đến khối CATCH tương ứng. Nó cũng chỉ ra cách sử dụng RAISERROR để trả về thông tin lỗi đặt trong khối CATCH.

Câu lệnh:

BEGIN TRY

- RAISERROR with severity 11-19 will cause execution to
- jump to the CATCH block.

RAISERROR ('Error raised in TRY block.', -- Message text.

16, -- Severity.

1 -- State.

);

END TRY

BEGIN CATCH

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

SELECT

```

    @ErrorMessage = ERROR_MESSAGE(),
    @ErrorSeverity = ERROR_SEVERITY(),
    @ErrorState = ERROR_STATE();

-- Use RAISERROR inside the CATCH block to return error
-- information about the original error that caused
-- execution to jump to the CATCH block.

RAISERROR (@ErrorMessage, -- Message text.

            @ErrorSeverity, -- Severity.

            @ErrorState -- State.

        );

```

END CATCH;

Kết quả:

The screenshot shows a SQL Server Management Studio interface. The top window is titled "SQLQuery1.sql - (l...E05V\tuannt (52))". It contains the following T-SQL code:

```

-- RAISERROR with severity 11-19 will cause execution to
-- jump to the CATCH block.
RAISERROR ('Error raised in TRY block.', -- Message text.
           16, -- Severity.
           1 -- State.
        );
END TRY
BEGIN CATCH
    DECLARE @ErrorMessage NVARCHAR(4000);
    DECLARE @ErrorSeverity INT;
    DECLARE @ErrorState INT;

    SELECT
        @ErrorMessage = ERROR_MESSAGE(),
        @ErrorSeverity = ERROR_SEVERITY(),
        @ErrorState = ERROR_STATE();

```

The bottom window is titled "Messages" and displays the following output:

```

Msg 50000, Level 16, State 1, Line 22
Error raised in TRY block.

```

The status bar at the bottom of the screen shows "Query completed with errors." and other system information.

3.4 LẬP TRÌNH CHO HÀM

3.4.1 Tạo hàm trong MicroSoft SQL Server

Trong MicroSoft SQL Server, ngoài việc lập trình viên có thể lập trình để tạo ra các thủ tục để thực hiện các tác vụ nào đó, hệ thống còn cho phép lập trình để tạo các hàm tính toán và trả về kết quả cho người dùng. MicroSoft SQL Server có hai loại hàm, đó là hàm hệ thống do MicroSoft SQL Server tạo sẵn, lập trình viên chỉ cần gọi tên hàm và truyền các tham số vào hàm để nhận kết quả, loại hàm thứ hai là hàm do người dùng tự định nghĩa và lập trình. Hàm do người dùng định nghĩa cũng là một đối tượng trong cơ sở dữ liệu sử dụng trong các câu lệnh T-SQL, được biên dịch sẵn và lưu trong cơ sở dữ liệu nhằm mục đích thực hiện xử lý nào đó như tính toán phức tạp và trả về kết quả là giá trị nào đó. Hàm khác với thủ tục lưu trữ ở đặc điểm hàm luôn trả về một giá trị nào đó, trong khi đó, thủ tục có thể không trả về giá trị nào hoặc có thể trả về nhiều giá trị với tham số được khai báo bằng từ khóa OUTPUT. Về cơ bản, lập trình cho các hàm cũng tương tự như lập trình cho thủ tục lưu trữ, các cấu trúc và câu lệnh trong thủ tục lưu trữ cũng được sử dụng trong lập trình cho hàm.

Hàm do người dùng tự định nghĩa cũng được chia làm hai loại:

- Scalar-valued: Trả về giá trị vô hướng của các kiểu dữ liệu T-SQL
- Table-valued: Trả về bảng, là kết quả của một hoặc nhiều câu lệnh T-SQL.

a) Hàm trả về giá trị vô hướng

Cú pháp:

```
CREATE FUNCTION <Tên function> ([@<tên tham số> <kiểu dữ liệu> [= <giá trị mặc định>], ..., [...]])
```

```
RETURNS <kiểu dữ liệu> [WITH ENCRYPTION]
```

```
[AS]
```

```
BEGIN
```

```
[Thân của hàm]
```

```
RETURN <Biểu thức giá trị đơn>
```

```
END
```

Ý nghĩa:

Tạo một hàm mới để tính toán và trả về giá trị cho người dùng.

Các thành phần trong lệnh:

- Tên function: Tên của hàm sẽ tạo
- Tên tham số: Là các tham số đầu vào (input) cho hàm. Quy cách khai báo tham số gồm tên của tham số (trước tên tham số sử dụng tiền tố @), kiểu dữ liệu của tham số, ta có thể chỉ định giá trị mặc định cho tham số. Trong một hàm, có thể chỉ định nhiều tham số đầu vào.
- RETURNS: từ khóa này chỉ định kiểu dữ liệu hàm sẽ trả về. Kiểu dữ liệu phải được chỉ định bao gồm kiểu và độ dài dữ liệu. Ví dụ: varchar(100).
- WITH ENCRYPTION: Từ khóa chỉ định các mã lệnh của hàm sẽ được mã hóa trong bảng syscomments.
- AS: Từ khóa cho biết các lệnh của hàm bắt đầu.
- RETURN: Từ khóa này sẽ gửi giá trị sau khi được tính toán tới thủ tục gọi hàm.

Chú ý:

+ Tên hàm phải là duy nhất trong một cơ sở dữ liệu. Hàm được tạo hoặc định nghĩa trong cơ sở dữ liệu nào thì chỉ sử dụng trong cơ sở dữ liệu đó, khác với hàm hệ thống có sẵn của MicroSoft SQL Server được truy cập ở bất cứ đâu.

+ Danh sách tham số có tối đa 1024 tham số.

Ví dụ:

Tạo hàm để đưa ra số lượng các loại mặt hàng chưa được bán lần nào.

Câu lệnh:

```
CREATE FUNCTION f_somathangchuaban()
```

```
RETURNS int
```

```
AS
```

```
BEGIN
```

```

DECLARE @soluong int
SELECT
    @soluong = count(mahang)
FROM tbl_dmhang
WHERE mahang NOT IN (SELECT mahang FROM tbl_banhang)
RETURN @soluong
END

```

Kết quả:

```

CREATE FUNCTION f_somathangchuaban()
RETURNS int
AS
BEGIN
DECLARE @soluong int
SELECT
@soluong = count(mahang)
FROM tbl_dmhang
WHERE mahang NOT IN (SELECT mahang FROM tbl_banhang)
RETURN @soluong
END
PRINT (dbo.f_somathangchuaban())

```

b) Hàm trả về giá trị bảng (Table-valued)

Cú pháp hàm trả giá trị bảng đơn giản:

CREATE FUNCTION <Tên function> ([@<tên tham số> <kiểu dữ liệu> [= <giá trị mặc định>], ..., [...]])

RETURNS TABLE

[WITH ENCRYPTION]

[AS]

RETURN <Câu lệnh SQL>

Ví dụ:

Lập hàm để đưa ra các phiếu bán hàng có số lượng đã bán lớn hơn tham số được truyền vào.

Câu lệnh:

```
CREATE FUNCTION f_chonphieubanhang (@soluong float)
```

```
RETURNS TABLE
```

```
AS
```

```
RETURN
```

```
SELECT *
```

```
FROM tbl_banhang
```

```
WHERE soluong > @soluong
```

Kết quả:

The screenshot shows the SQL Server Management Studio interface. On the left, the Object Explorer pane displays the database structure, including the 'BANHANG' database and its tables. In the center, the SQL Query Editor window contains the T-SQL code for creating a function named 'f_chonphieubanhang'. The code defines the function to return a table of rows from the 'tbl_banhang' table where the 'soluong' column is greater than the input parameter '@soluong'. Below the code, a query is executed: 'SELECT * FROM dbo.f_chonphieubanhang'. The results pane on the right shows a table with five rows, each containing values for columns: 'sophieu', 'ngay', 'manv', 'mahang', and 'soluong'. The 'soluong' column values are circled in red, showing they are greater than the parameter value used in the query. The status bar at the bottom indicates the query was executed successfully.

sophieu	ngay	manv	mahang	soluong
P0003	2020-04-06 00:00:00	N03	H0003	15
P0005	2020-07-25 00:00:00	N05	H0006	20
P0006	2019-06-05 00:00:00	N01	H0010	10
P0009	2020-09-04 00:00:00	N05	H0002	40
P0011	2020-06-28 00:00:00	N04	H0005	14

Cú pháp hàm trả giá trị bằng bảng bằng đa câu lệnh:

```
CREATE FUNCTION <Tên function> ([@<tên tham số> <kiểu dữ liệu> [= <giá trị mặc định>], ..., [...]])
```

```
RETURNS @<tên biến trả về> TABLE (<tên cột 1> <kiểu dữ liệu> [tùy chọn thuộc tính], ..., <tên cột n> <kiểu dữ liệu> [tùy chọn thuộc tính])
```

[AS]

```
BEGIN  
    <Câu lệnh SQL>  
RETURN  
END
```

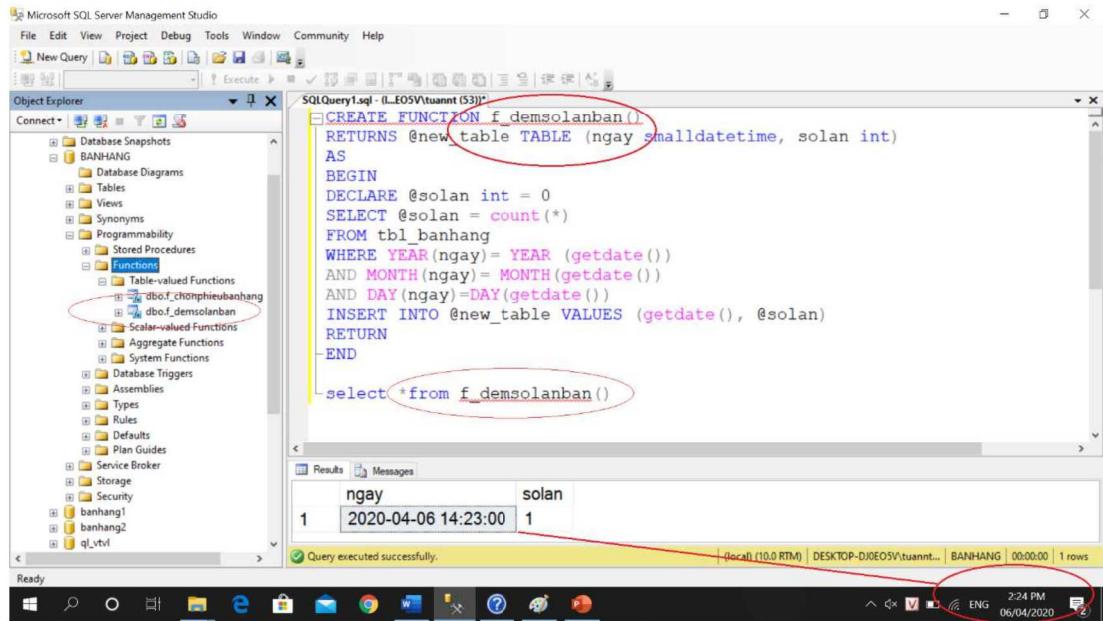
Ví dụ:

Lập hàm trả về bảng có 2 trường là ngày, số lần tương ứng với ngày hiện tại trong hệ thống và số lần bán hàng trong ngày đó.

Câu lệnh:

```
CREATE FUNCTION f_demsolanban()  
RETURNS @new_table TABLE (ngay smalldatetime, solan int)  
AS  
BEGIN  
    DECLARE @solan int = 0  
    SELECT @solan = count(*)  
    FROM tbl_banhang  
    WHERE YEAR(ngay)= YEAR (getdate())  
        AND MONTH(ngay)= MONTH(getdate())  
        AND DAY(ngay)=DAY(getdate())  
    INSERT INTO @new_table VALUES (getdate(), @solan)  
    RETURN  
END
```

Kết quả:



3.4.2 Sửa, xóa hàm

Tương tự như lập trình thủ tục lưu trữ, MicroSoft SQL Server cho phép thực hiện sửa nội dung của hàm hoặc xóa hàm đã tạo ra bằng các cấu trúc lệnh dưới đây.

Cú pháp sửa nội dung của hàm:

ALTER FUNCTION <tên hàm>

<mô tả hàm mới>

Ví dụ:

Sửa hàm f_demsolanban thành hàm trả về bảng có 2 trường là mã hàng, số lần bán tương ứng với số lần bán của từng mặt hàng đó.

Câu lệnh:

ALTER FUNCTION f_demsolanban()

RETURNS @new_table TABLE (mahang char(5), solan int)

AS

BEGIN

INSERT INTO @new_table (mahang, solan)

```

SELECT mahang,count(mahang)
FROM tbl_banhang
GROUP BY mahang
RETURN
END

```

Kết quả:

The screenshot shows the SQL Server Management Studio interface. In the top pane, a query window titled "SQLQuery1.sql - (L-E05\vtuannt (53))" contains the following T-SQL code:

```

ALTER FUNCTION f_demsolanban()
RETURNS @new_table TABLE (mahang char(5), solan int)
AS
BEGIN
    INSERT INTO @new_table (mahang, solan)
    SELECT mahang, count(mahang)
    FROM tbl_banhang
    GROUP BY mahang
    RETURN
END
select *from f_demsolanban()

```

In the bottom pane, the "Results" tab displays the output of the function execution. The results are presented in a table:

	mahang	solan
1	H0001	2
2	H0002	3
3	H0003	2
4	H0004	2
5	H0005	1
6	H0006	1
7	H0009	2
8	H0010	2

The status bar at the bottom indicates "Query executed successfully." and "8 rows".

Cú pháp xóa hàm đã tạo:

DROP FUNCTION <tên hàm>

3.5 LẬP TRÌNH TRIGGER

3.5.1 Giới thiệu

Trigger là một thủ tục đặc biệt mà việc thực thi của nó tự động khi có sự kiện xảy ra, các sự kiện gọi thủ tục đặc biệt này được định nghĩa trong câu lệnh, thông thường được thực hiện với các sự kiện liên quan đến các thao tác cập nhật dữ liệu

như Insert, Update, Delete với các bảng hoặc các view, các thao tác liên quan đến cập nhật các đối tượng trong cơ sở dữ liệu hoặc thao tác truy cập hệ thống. Trigger chủ yếu được sử dụng để duy trì tính toàn vẹn của thông tin trên cơ sở dữ liệu. và cũng có thể được sử dụng để ghi lại dữ liệu lịch sử.

Sử dụng Trigger khi các biện pháp toàn vẹn dữ liệu như constraint, rule,... không bảo đảm. Khác với các công cụ bảo đảm toàn vẹn dữ liệu đã nêu là sẽ thực hiện kiểm tra tính toán vẹn trước khi đưa dữ liệu vào cơ sở dữ liệu, còn Trigger thực hiện kiểm tra tính toán vẹn khi công việc đã và đang thực hiện. Khi cơ sở dữ liệu chưa được chuẩn hóa thì có thể xảy ra dữ liệu thừa, chứa ở nhiều vị trí trong cơ sở dữ liệu thì yêu cầu đặt ra là dữ liệu cần cập nhật thống nhất trong mọi nơi, trong trường hợp này ta phải sử dụng trigger.

Trigger có một số đặc điểm sau:

- Trigger là một thủ tục lưu trữ đặc biệt không có tham số
- Trigger sẽ được tự động thực hiện khi có hoạt động cập nhật trên bảng dữ liệu hoặc view như INSERT, UPDATE, DELETE. Trigger chỉ thực thi tự động thông qua các sự kiện mà không thực hiện bằng tay.
- Một bảng hoặc view có thể có nhiều trigger
- Một trigger có thể thực hiện nhiều công việc (theo kịch bản), có thể nhiều sự kiện kích hoạt thực thi trigger, có thể tách rời các sự kiện trong một trigger.
- Trigger không được tạo trên bảng temprate hay system.
- Trigger còn được sử dụng cho cơ sở dữ liệu và cho SERVER đối với các lệnh CREATE, ALTER, DROP, GRANT, DENY, REVOKE, UPDATE STATISTICS, LOGON. Trong nội dung của tài liệu này sẽ không đề cập đến các trigger đó.

3.5.2 Tạo, sửa, xóa Trigger

Cú pháp:

```
CREATE TRIGGER trigger_name
ON { table | view }
```

```
{ FOR | AFTER | INSTEAD OF } { [ INSERT ] [,] [ UPDATE ] [,] [ DELETE ]
]
[ WITH APPEND ] [ NOT FOR REPLICATION ]
AS sql_statement
```

Ý nghĩa:

Tạo trigger cho bảng hoặc view với thao tác cập nhật dữ liệu cho đối tượng đó.

Các tham số:

- TRIGGER_NAME: tên trigger sẽ tạo trên bảng hoặc view.
- ON: tên bảng hoặc view muốn tạo trigger.
- FOR/AFTER: after được sử dụng để trigger tự động thực hiện sau khi tất cả các thao tác cập nhật dữ liệu đã thực hiện thành công, đây cũng là thông số ngầm định khi chỉ có từ khóa FOR duy nhất được chỉ định. AFTER không sử dụng cho view.
- INSTEAD OF: các câu lệnh trong trigger sẽ được thực thi thay thế các hành động INSERT, UPDATE, DELETE của bảng hoặc view chứa trigger. Khi đó các hành động INSERT, UPDATE, DELETE ban đầu của bảng hoặc view sẽ bị bỏ qua.
- WITH APPEND: không sử dụng với AFTER và INSTEAD OF, chỉ sử dụng khi mức độ tương thích từ 65 trở xuống (phiên bản SQL Server cũ).
- NOT FOR REPLICATION: xác định trigger không thực hiện đối với đối tượng nhân bản.

Chú ý:

Trước khi bắt cứ câu lệnh nào chứa trigger thực sự được thực hiện, SQL Server tạo ra hai bảng đặc biệt lưu trong bộ nhớ có cùng cấu trúc với bảng mà trên đó trigger được tạo. Table INSERTED chứa các giá trị đang được Add vào bảng. Table DELETED chứa các giá trị đang bị xóa từ bảng. Nếu trigger được định nghĩa là INSERT trigger thì SQL Server sẽ chỉ tạo bảng INSERTED, còn nếu là DELETE trigger thì SQL Server

sẽ chỉ tạo bảng DELETED, nếu trigger được định nghĩa là UPDATE trigger thì SQL Server sẽ tạo cả hai bảng vì INSERTED sẽ chứa ảnh của hàng sau khi thay đổi còn bảng DELETED chứa ảnh của hàng trước khi thay đổi.

Ví dụ:

Hãy tạo các trigger để lưu lại vết cho các thao tác thêm, sửa, xóa mặt hàng trong bảng tbl_dmhang. Thông tin lưu vết bao gồm thao tác thực hiện là gì, thời điểm thao tác (lấy theo thời gian của máy chủ).

Câu lệnh:

Tạo bảng my_log để chứa dữ liệu lưu vết:

```
CREATE TABLE my_log  
(thaotac nvarchar(100),  
thoidiem smalldatetime)
```

Tạo trigger cho thao tác thêm dữ liệu vào bảng tbl_dmhang:

```
CREATE TRIGGER tg_themdmhang  
ON tbl_dmhang  
FOR INSERT  
AS  
    INSERT INTO my_log VALUES  
(N'Thêm mặt hàng mới vào bảng', GETDATE())
```

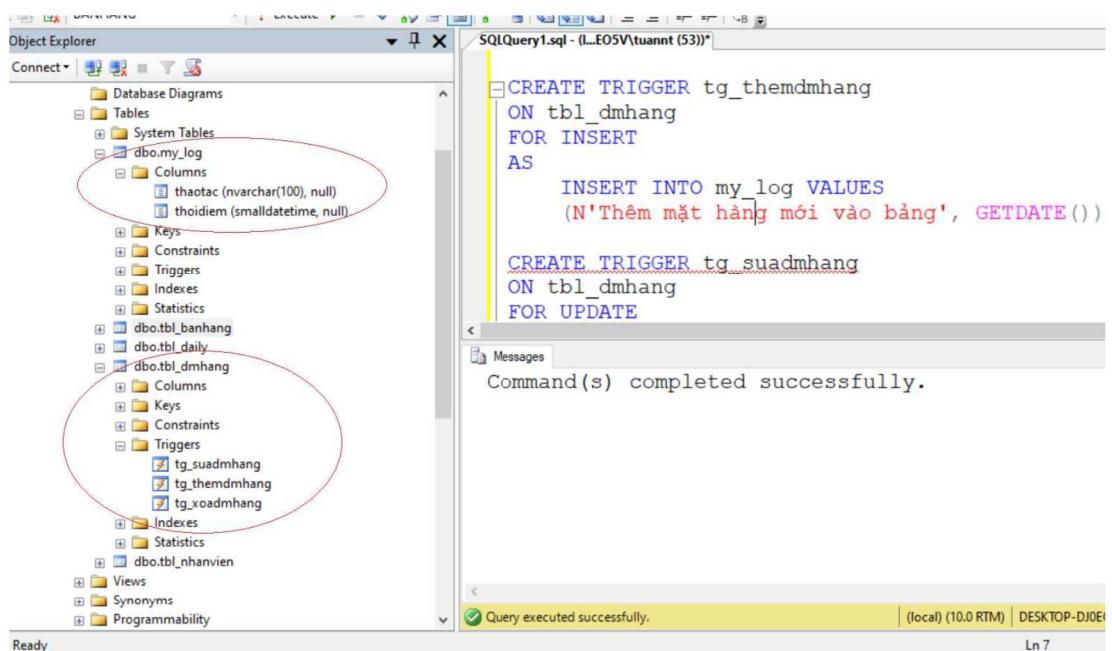
Tạo trigger cho thao tác sửa dữ liệu của bảng tbl_dmhang:

```
CREATE TRIGGER tg_suadmhang  
ON tbl_dmhang  
FOR UPDATE  
AS  
    INSERT INTO my_log VALUES  
(N'Sửa mặt hàng trong bảng', GETDATE())
```

Tạo trigger cho thao tác xóa dữ liệu của bảng tbl_dmhang:

```
CREATE TRIGGER tg_xoadmhang
ON tbl_dmhang
FOR DELETE
AS
    INSERT INTO my_log VALUES
    (N'Xóa mặt hàng trong bảng', GETDATE())
```

Kết quả:



Sửa, xóa Trigger

Cú pháp sửa trigger:

```
ALTER TRIGGER <tên trigger>
```

```
<mô tả mới trigger>
```

Cú pháp xóa trigger:

```
DROP Trigger <tên trigger>
```

3.6 LÀM VIỆC VỚI TRANSACTION

Transaction (phiên giao dịch) là một đơn vị công việc trong nó bao gồm nhiều việc nhỏ, các việc này được thực hiện thành công thì Transaction thành công, dữ liệu thay đổi trong quá trình thực hiện của Transaction sẽ được cập nhật. Nếu trong quá trình có phát sinh lỗi thì Transaction sẽ có khả năng khôi phục lại các trạng thái trước đó (Rollback), dữ liệu không được cập nhật. Một phiên giao dịch có 4 đặc tính ACID (Atomicity, Consistency, Isolation, Durability).

- **Atomicity - Tính bảo toàn:** nguyên tắc "all or nothing", đảm bảo rằng tất cả các câu lệnh trong nhóm lệnh được thực thi thành công. Nếu không, Transaction bị hủy bỏ tại thời điểm thất bại và tất cả các thao tác trước đó được khôi phục về trạng thái cũ đồng nghĩa với việc không có gì thay đổi về mặt dữ liệu.
- **Consistency - Tính nhất quán:** đảm bảo rằng cơ sở dữ liệu thay đổi chính xác các trạng thái khi một transaction được thực thi thành công.
- **Isolation - Tính độc lập:** cho phép các Transaction hoạt động độc lập và minh bạch với nhau.
- **Durability - Tính bền vững:** đảm bảo rằng kết quả của một transaction được xác định, không có chuyện dữ liệu của Transaction sau khi thực thi có thể chuyên lại trạng thái dữ liệu lúc trước khi thực hiện.

Theo dòng thời gian thực hiện phiên giao dịch (transaction), mỗi phiên có điểm bắt đầu (Begin), các điểm mốc khi thực hiện được một số công việc nhất định và muốn ghi lại trạng thái tại thời điểm đó (save point), điểm kết thúc của phiên giao dịch. Trong một transaction, chỉ có một điểm bắt đầu, một điểm kết thúc và có thể có nhiều điểm mốc. Trong quá trình thực hiện, transaction có thể khôi phục lại tại trạng thái bất kỳ ngược theo dòng thời gian và chỉ khôi phục được tại điểm mốc trước điểm mốc đã khôi phục trước đó. Khi một transaction kết thúc, không còn khả năng khôi phục được nữa.

Cú pháp tạo transaction:

```
BEGIN TRAN [ TRANSACTION ] [ transaction_name | @tran_name_variable  
[ WITH MARK [ 'description' ] ] ]
```

Ý nghĩa:

Tạo một transaction mới với tên gọi hoặc biến chứa tên transaction. Trong trường hợp muốn mô tả thêm thông tin về transaction, có thể thay thế các nội dung trong phần *description*.

Ví dụ:

Tạo transaction có tên trans_themhang, sau đó thêm 2 mặt hàng mới vào bảng tbl_dmharg.

Câu lệnh:

BEGIN TRAN trans_themhang

 INSERT INTO tbl_dmharg

 VALUES

 ('H0011',N'Ti vi' , N'Bộ',5000000)

 INSERT INTO tbl_dmharg

 VALUES

 ('H0012',N'Tủ lạnh' , N'Bộ',10000000)

Cú pháp kết thúc transaction:

 COMMIT [TRAN [SACTION] [*transaction_name* | @*tran_name_variable*]]

Ý nghĩa:

Kết thúc transaction xác định trong tên được chỉ ra trong lệnh. Khi thực hiện lệnh này, toàn bộ các công việc của transaction đã được thực hiện thành công, toàn bộ các dữ liệu sẽ được lưu vào trong cơ sở dữ liệu. Khi đó transaction sẽ tự động bị xóa bỏ và không còn khả năng khôi phục được các trạng thái trước đó bằng lệnh ROLLBACK của transaction.

Ví dụ:

Kết thúc phiên trans_themhang và ghi dữ liệu vào cơ sở dữ liệu.

Câu lệnh:

 COMMIT TRAN trans_themhang

Cú pháp lưu các điểm mốc trong transaction:

SAVE TRAN [SACTION] { savepoint_name | @savepoint_variable }

Ý nghĩa:

Cho phép lưu lại điểm mốc trong quá trình thực hiện của transaction, các điểm mốc này có thể được sử dụng khi transaction muốn khôi phục các trạng thái đến thời điểm đó.

Ví dụ:

Tạo transaction có tên trans_themhang, sau đó thêm 2 mặt hàng mới vào bảng tbl_dmharg. Mỗi lần thêm một mặt hàng, ghi lại mốc.

Câu lệnh:

BEGIN TRAN trans_themhang

```
INSERT INTO tbl_dmharg  
VALUES  
('H0011',N'Ti vi' , N'Bộ',5000000)
```

```
SAVE TRAN trans_themhang_save1
```

```
INSERT INTO tbl_dmharg  
VALUES  
('H0012',N'Tủ lạnh' , N'Bộ',10000000)
```

```
SAVE TRAN trans_themhang_save2
```

Cú pháp khôi phục trạng thái đến điểm mốc hoặc đầu transaction:

**ROLLBACK [TRAN [SACTION]
[transaction_name | @tran_name_variable
| savepoint_name | @savepoint_variable]]**

Ý nghĩa:

Khôi phục trạng thái của transaction đến điểm mốc hoặc đến điểm bắt đầu của transaction. Có thể sử dụng lệnh khôi phục trạng thái nhiều lần với các điểm mốc khác nhau nhưng phải tuân thủ thứ tự thời gian.

Ví dụ:

Khôi phục dữ liệu của phiên về điểm mốc trans_themhang_save1.

Câu lệnh:

ROLLBACK TRAN trans_themhang_save1

Ví dụ:

Lập một thủ tục có sử dụng cấu trúc TRY – CATCH để xử lý lỗi và khôi phục các trạng thái cho transaction khi thực hiện lệnh sửa đổi đơn giá của 2 mặt hàng.

Câu lệnh:

```
CREATE PROC sp_TransMathang
```

```
AS
```

```
BEGIN
```

```
    BEGIN TRY
```

```
        BEGIN TRANSACTION tran_suamathang
```

```
            UPDATE tbl_dmhang
```

```
                SET dongia = 12000000
```

```
                WHERE mahang = 'H0006'
```

```
            UPDATE tbl_dmhang
```

```
                SET dongia = 9000000
```

```
                WHERE mahang = 'H0008'
```

```
        COMMIT TRANSACTION tran_suamathang
```

```
        PRINT N'trực hiện xong'
```

```
    END TRY
```

```
    BEGIN CATCH
```

```
PRINT N'Khôi phục lại';
ROLLBACK TRAN tran_suamathang
END CATCH
END
```

CHƯƠNG IV. QUẢN TRỊ SQL SERVER

4.1 TỰ ĐỘNG HOÁ CÔNG VIỆC QUẢN TRỊ

4.1.1 Cấu hình dịch vụ SQL Server Agent

SQLServer Agent là một module trong MicroSoft SQL Server dùng để thực hiện một công việc nào đó một cách tự động theo lịch trình mà chúng ta định ra. Trong thực tế, bất kỳ nhiệm vụ hay một công việc nào đó phải thực thi nhiều hơn một lần đều có thể tự động hóa và các công việc này xuất hiện khá nhiều đối với các máy chủ cơ sở dữ liệu.

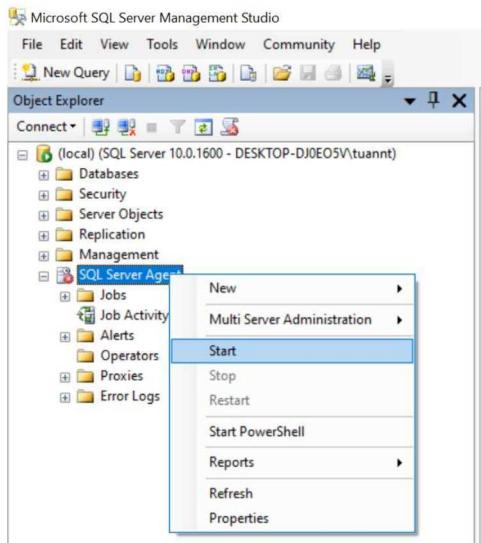
Việc thực hiện các công việc đó một cách tự động đem lại những lợi ích:

- Giảm thời gian khi triển khai công việc thực thi lặp lại.
- Bảo đảm các nhiệm vụ lặp lại luôn được thực hiện một cách đồng nhất, chính xác vào các thời điểm định trước.

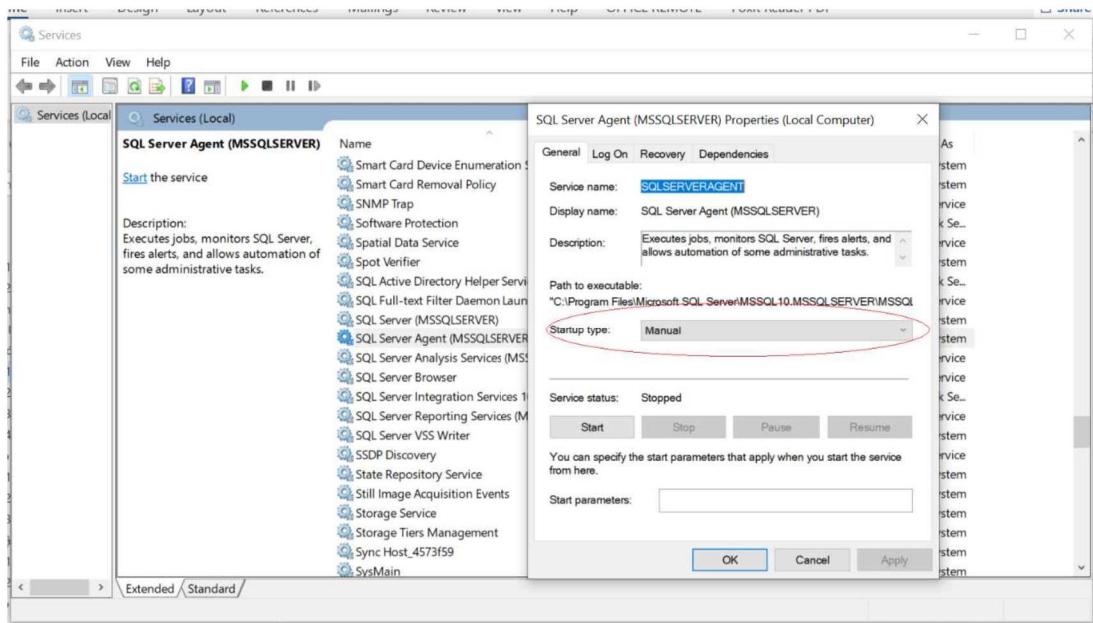
SQL Server Agent cho phép tự động hóa các nhiệm vụ quản trị của SQL Server (SQL Server Administration Task). Dưới đây là các thành phần của SQL Server Agent:

- **Jobs:** là một nhiệm vụ hay một nhóm nhiệm vụ mà SQLServer Agent sẽ thực thi. Sử dụng Job để xác định các nhiệm vụ quản trị và giám sát cả lúc thành công cũng như thất bại. Jobs có thể chạy trên một server cục bộ hay trên nhiều server từ xa.
- **Schedules:** xác định lịch trình cho Job sẽ chạy, có thể một hay nhiều job chạy với cùng một schedule và một hay nhiều schedule có thể áp dụng cho cùng một Job.
- **Alerts:** là một đáp ứng tự động ứng với một sự kiện xác định nào đó.
- **Operators:** cho phép định nghĩa thông tin của một cá nhân có nhiệm vụ bảo trì một hay nhiều instances của MicroSoft SQL Server. Operator không chứa thông tin bảo mật.

Để sử dụng được SQL Server Agent, cần phải khởi động dịch vụ này, bằng cách nhấn phím phải chuột lên cây SQL Server Agent trong MicroSoft SQL Server Management Studio, sau đó chọn **Start**.



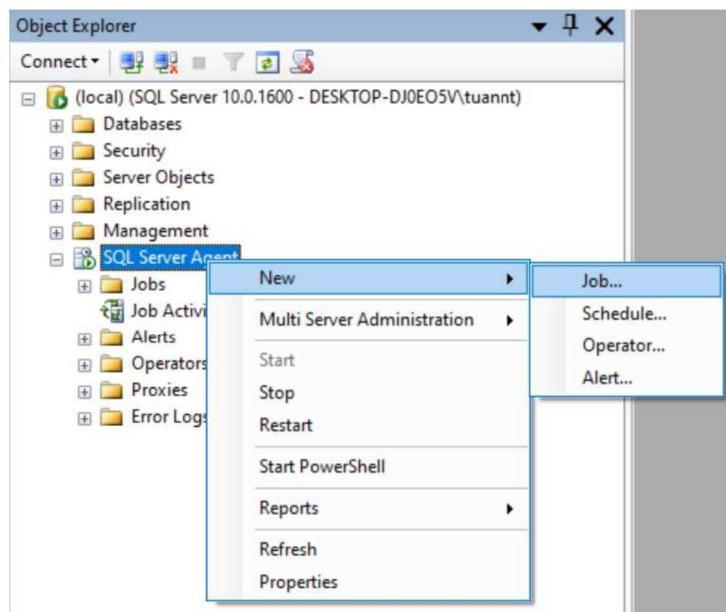
Một cách khác để khởi động dịch vụ SQL Server Agent và có thể đặt thông số để dịch vụ tự động khởi động bằng cách chạy ứng dụng **Services** của hệ điều hành, sau đó chọn đến mục **SQL Server Agent**, nhấn phím phải chuột, chọn **Properties**. Trong mục **Startup Type**, chọn **Automatic**. Nhấn nút **Start** và **Ok**.



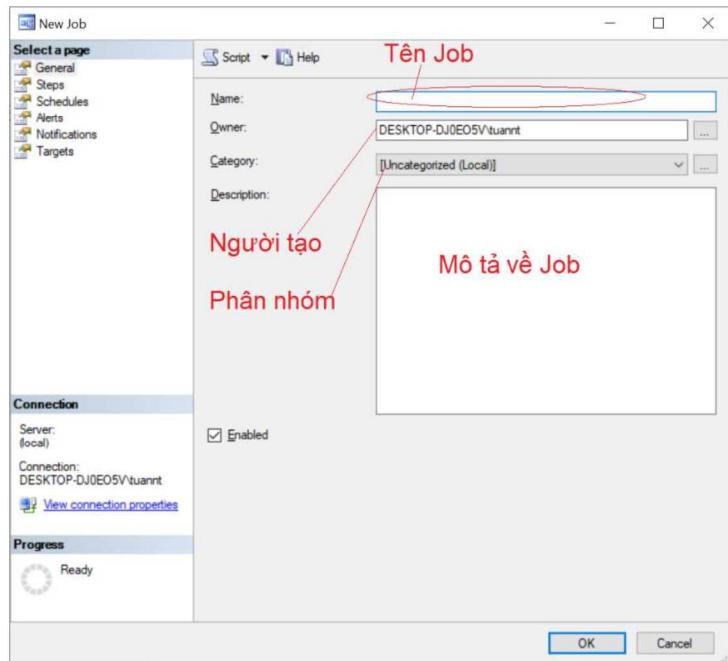
4.1.2 Thực thi công việc (Jobs)

Trong những công việc phải thực hiện lặp lại nhiều lần với những thao tác giống nhau, vào các thời điểm giống nhau, thông thường người quản trị có thể tạo ra các Job để thực hiện công việc này. Trong một Job cho phép thực hiện nhiều nhiệm vụ khác nhau, mỗi nhiệm vụ như vậy được MicroSoft SQL Server gọi là một Step. Dưới đây là các bước để thực hiện tạo mới một Job trong MicroSoft SQL Server.

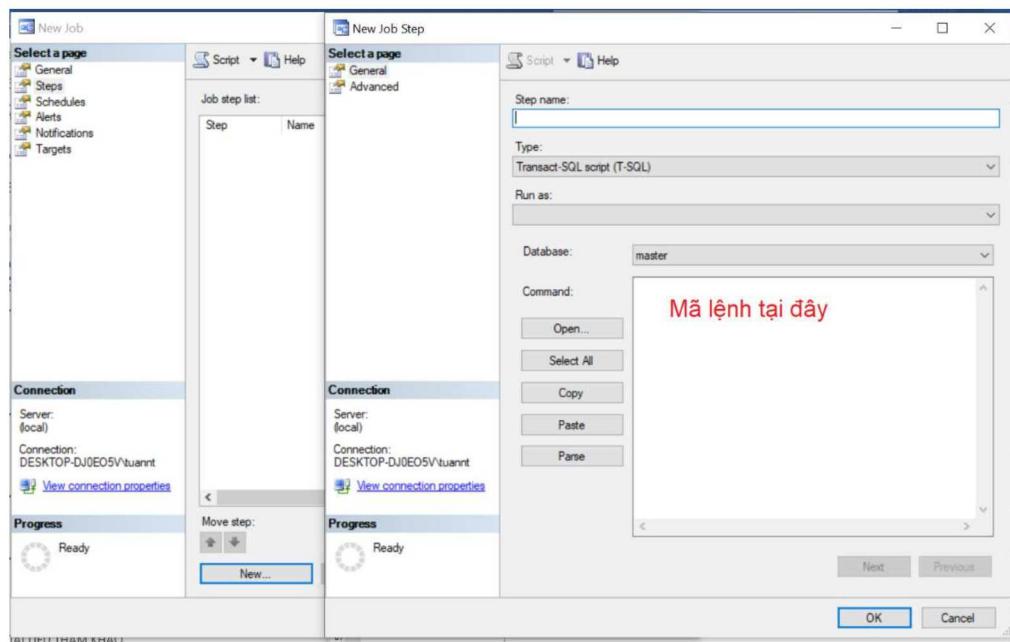
Bước 1: Sau khi chạy dịch vụ SQL Server Agent, nhấn phải chuột lên cây **SQL Server Agent** hoặc lên mục **Jobs** trong cây này, chọn **New ->Job...**



Bước 2: Đặt tên cho Job và các thông tin khác trong cửa sổ **General**



Bước 3: Tạo các nhiệm vụ cho **Job** trong cửa sổ **Steps**. Trong cửa sổ này, có thể tạo nhiều Step khác nhau bằng cách lập trình trực tiếp hoặc lấy từ mã lệnh đã có. Thứ tự các Step trong danh sách sẽ là thứ tự thực hiện khi Job được thực thi, ta có thể thay đổi thứ tự này bằng cách di chuyển lên/xuống (Move step). Có thể thêm/sửa/xóa các nhiệm vụ trong danh sách.



Các thông số trong một nhiệm vụ mới:

- Step name: tên của nhiệm vụ
- Type: loại lệnh hoặc thao tác của nhiệm vụ cần thực hiện. MicroSoft SQL Server hỗ trợ nhiều loại lệnh khác nhau. Trong trường hợp làm việc với cơ sở dữ liệu, có thể chọn Transact – SQL script.
- Run as: xác định người dùng đủ quyền để thực hiện nhiệm vụ đang tạo.
- Database: xác định cơ sở dữ liệu mà nhiệm vụ sẽ tác động đến.
- Parse: dùng bộ dịch có sẵn trong hệ thống MicroSoft SQL Server để kiểm tra cú pháp lệnh.

Ngoài các thông tin cơ bản trên, mục Advanced cho phép khai báo thêm các thông tin để kiểm soát việc thực hiện của nhiệm vụ này.

Bước 4: Tạo lịch biểu cho Job trong mục **Schedules**. Trong một Job có thể có nhiều lịch biểu để thực hiện. Nếu có lịch biểu phù hợp với Job, nhấn nút **Pick** để chọn các lịch biểu, ngược lại, nhấn nút **New** để tạo lịch biểu mới cho Job (chi tiết thông số xem mục **Tạo lịch biểu mới**).

Bước 5: Đặt các thông số để xử lý các sự kiện xảy ra (xem mục **Alerts** ở phần sau) hoặc cách thức thông báo cho người quản trị (**Notifications**) nếu cần.

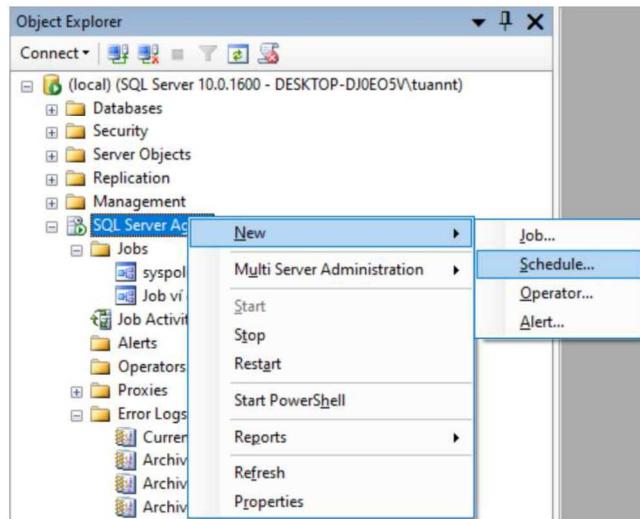
Bước 6: Chọn nút Ok để hoàn thành việc tạo Job mới.

Chú ý:

- + Để sửa các nội dung của Job đã tạo, nhấn phím phải chuột lên Job và chọn mục **Properties**.
- + Để xóa Job đã tạo, nhấn phím phải chuột lên Job và chọn **Delete**.
- + Để vô hiệu hóa Job đã tạo, nhấn phím phải chuột lên Job và chọn **Disable**.

Tạo lịch biểu mới (Schedule)

Bước 1: Nhấn phím phải chuột lên SQL Server Agent, chọn **New -> Schedule**



Bước 2: Khai báo các thông tin cho lịch biểu: tên, chọn cách thực hiện, các thời điểm thực hiện...

New Job Schedule

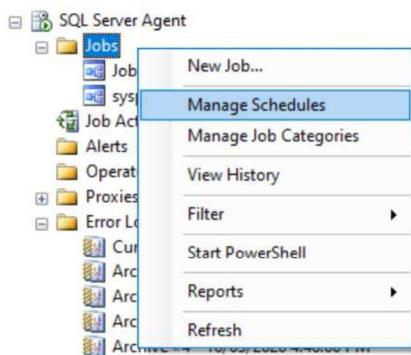
Name:	Lịch mới 1	Jobs in Schedule
Schedule type:	Recurring	<input checked="" type="checkbox"/> Enabled
One-time occurrence		
Date:	21/04/2020	Time: 3:24:59 PM
Frequency		
Occurs:	Weekly	
Recurs every:	1 week(s) on	
<input checked="" type="checkbox"/> Monday <input checked="" type="checkbox"/> Wednesday <input type="checkbox"/> Friday <input type="checkbox"/> Saturday <input type="checkbox"/> Tuesday <input type="checkbox"/> Thursday <input checked="" type="checkbox"/> Sunday		
Daily frequency		
(<input checked="" type="radio"/> Occurs once at:	12:00:00 AM	
(<input type="radio"/> Occurs every:	1 hour(s)	
Starting at:	12:00:00 AM	
Ending at:	11:59:59 PM	
Duration		
Start date:	21/04/2020	
End date:	21/04/2020	
<input checked="" type="radio"/> No end date:		
Summary		
Description:	Occurs every week on Monday, Wednesday, Sunday at 12:00:00 AM. Schedule will be used starting on 21/04/2020	
<input type="button" value="OK"/> <input type="button" value="Cancel"/> <input type="button" value="Help"/>		

Trong đó:

- Schedule type: chọn cách lịch sẽ thực hiện, ví dụ lặp lại, thực hiện một lần, thực hiện khi khởi động SQL Server Agent hoặc khi hệ thống chuyển sang trạng thái chờ.
- Các thông tin khác được liệt kê chi tiết cho từng cách lịch biểu được chọn để thực hiện, như thực hiện những ngày nào trong tuần, thời gian điểm thực hiện, thời gian bắt đầu và kết thúc...

Sửa và quản lý lịch

Nhấn phím phải vào Jobs, chọn mục **Manage Schedules**:

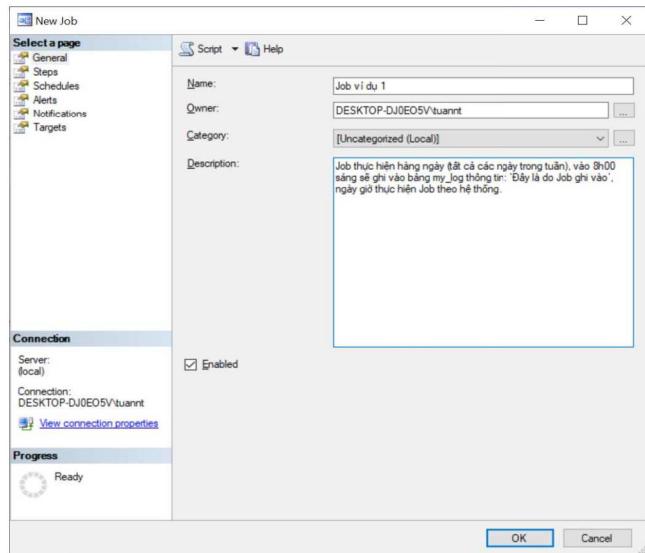


Ví dụ:

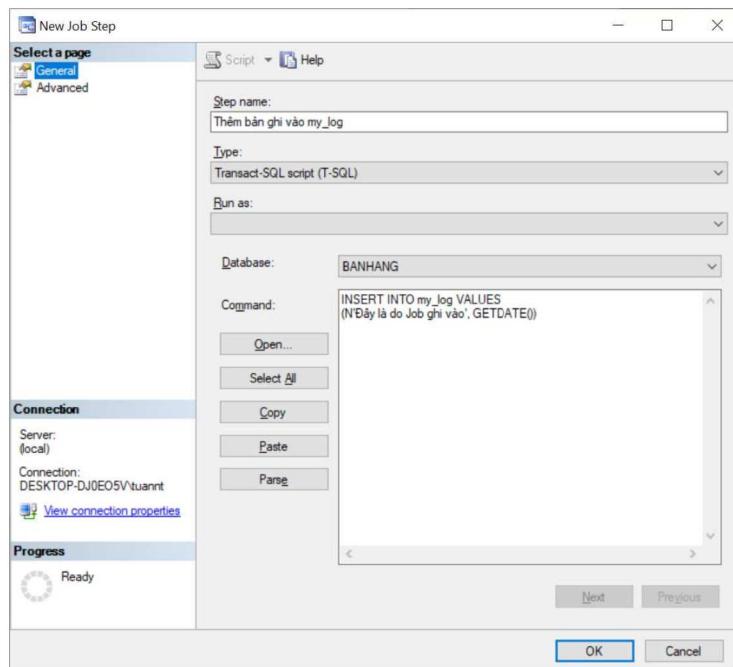
Tạo một Job để hàng ngày (tất cả các ngày trong tuần), vào 8h00 sáng sẽ ghi vào bảng my_log thông tin: ‘Đây là do Job ghi vào’, ngày giờ thực hiện Job theo hệ thống.

Thực hiện:

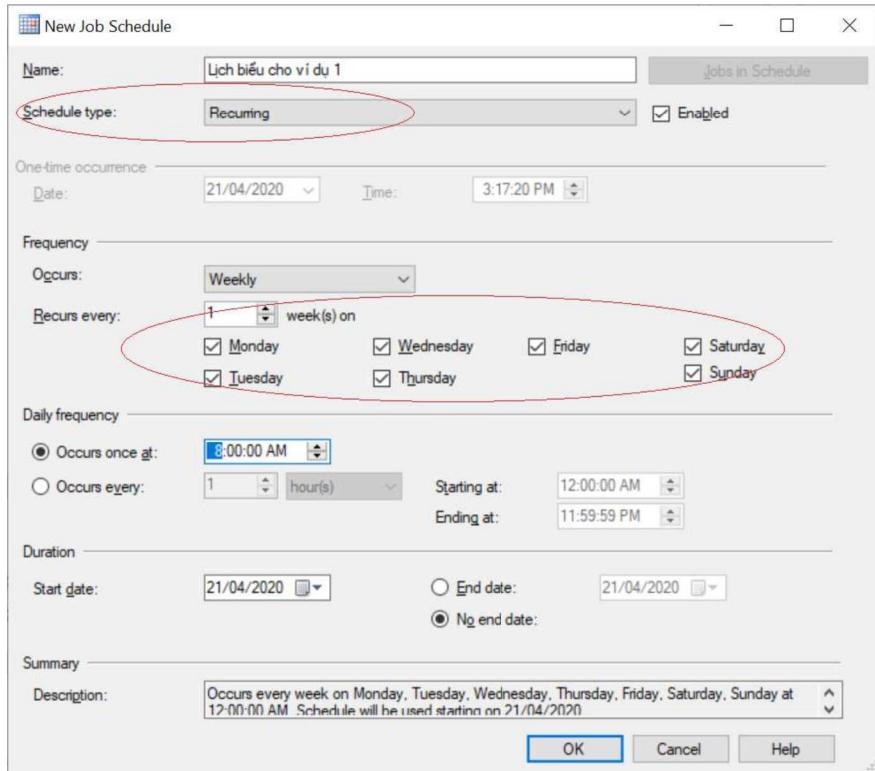
Bước 1: Tạo Job mới và đặt tên là **Job ví dụ 1**



Bước 2: Tạo một Step là ‘Thêm bản ghi vào my_log’



Bước 3: Tạo một lịch mới theo yêu cầu của bài toán. Sau khi kết thúc tạo lịch, chọn nút **Ok** để hoàn thành việc tạo Job.



Thông tin tình trạng thực thi của các Job

MicroSoft SQL Server cung cấp công cụ Job Activity Monitor cho phép xem thông tin về tình trạng thực thi của các Job ngoài việc các Job thông báo kết quả qua việc phản hồi các sự kiện (Alerts) hay Notifications.

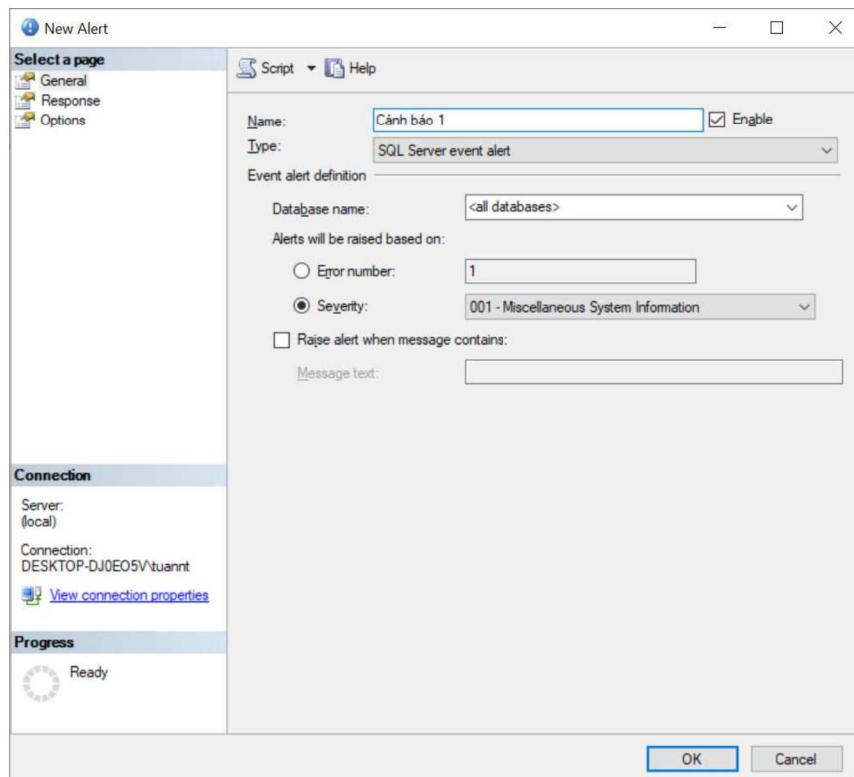
Name	Enabled	Status	Last Run Outcome	Last Run	Next Run	Category	Runnable	Scheduled	Category ID
Job ví dụ 1	yes	Idle	Unknown	never	22/04/2020 8:00:00 AM	[Uncategorized (Local)]	yes	yes	0
syspolicy_purge_history	yes	Idle	Succeeded	15/04/2020 8:03:26 AM	22/04/2020 2:00:00 AM	[Uncategorized (Local)]	yes	yes	0

Trong công cụ này thể hiện các thông tin thực hiện của các Jobs, lịch trình tiếp theo. Công cụ sẽ tự động cập nhật trạng thái theo khoảng thời gian nhất định do người sử dụng thiết lập.

4.1.3 Cảnh báo (Alerts)

Các sự kiện do SQL Server sinh ra và được đưa vào nhật ký (log) của ứng dụng Windows. SQL Server Agent đọc nhật ký ứng dụng và so sánh với các sự kiện được ghi ở đó với các cảnh báo mà bạn đã xác định. Khi SQL Server Agent tìm thấy sự trùng khớp, nó sẽ đưa ra một cảnh báo, đó là phản hồi tự động cho một sự kiện. Ngoài việc theo dõi các sự kiện SQL Server, SQL Server Agent cũng có thể theo dõi các điều kiện thực hiện và các sự kiện của Công cụ quản lý Windows (WMI).

Trong MicroSoft SQL Server, tạo một cảnh báo mới bằng cách nhấn phím phải chuột lên **SQL Server Agent** và chọn **New -> Alert** hoặc chọn mục **Alert** trong khi khai báo **Job**. Các thông số của khung hội thoại tạo Alert được mô tả chi tiết sau đây.



Để thiết lập cảnh báo (Alert), ta xác định:

- Tên của cảnh báo.
- Sự kiện hoặc điều kiện thực hiện để kích hoạt cảnh báo.

- Hành động mà SQL Server Agent thực hiện để đáp ứng với điều kiện hiệu suất hoặc sự kiện.

Một cảnh báo phản ứng với một sự kiện của một loại cụ thể. Cảnh báo phản hồi các loại sự kiện sau: An alert responds to an event of a specific type. Alerts respond to the following event types:

- Sự kiện SQL Server
- Các điều kiện thực thi của SQL Server
- Sự kiện WMI

Cảnh báo cho sự kiện SQL Server

Ta có thể chỉ định một cảnh báo xảy ra để đáp ứng với một hoặc nhiều sự kiện. Sử dụng các tham số sau để chỉ định các sự kiện kích hoạt cảnh báo:

- **Error number:** SQL Server Agent kích hoạt cảnh báo khi xảy ra lỗi cụ thể. Ví dụ: có thể chỉ định số lỗi 2571 để phản hồi các nỗ lực trái phép để gọi Lệnh điều khiển cơ sở dữ liệu (DBCC).
- **Severity level:** SQL Server Agent kích hoạt cảnh báo khi có bất kỳ lỗi nào về mức độ nghiêm trọng cụ thể xảy ra. Ví dụ: có thể chỉ định mức độ nghiêm trọng là 15 để phản hồi các lỗi cú pháp trong các câu lệnh Transact-SQL.
- **Database:** SQL Server Agent chỉ phát cảnh báo khi sự kiện xảy ra trong cơ sở dữ liệu cụ thể. Tùy chọn này áp dụng ngoài số lỗi hoặc mức độ nghiêm trọng.
- **Event text:** SQL Server Agent kích hoạt cảnh báo khi sự kiện được chỉ định chứa một chuỗi văn bản cụ thể trong thông báo sự kiện.

Với loại cảnh báo điều kiện thực thi (Performance Condition)

Bạn có thể chỉ định một cảnh báo xảy ra để đáp ứng với một điều kiện thực thi cụ thể. Trong trường hợp này, bạn chỉ định bộ đếm thực thi để giám sát, ngưỡng cho cảnh báo và hành vi mà bộ đếm phải hiển thị nếu cảnh báo xảy ra. Các thông số như sau:

- **Object:** Đối tượng cần theo dõi.

- **Counter:** Là một thuộc tính cần theo dõi.
- **Instance:** Xác định bản cụ thể SQL Server (nếu có) của thuộc tính sẽ được theo dõi.
- **Alert if counter và Value:** Nguồn cho cảnh báo và hành vi tạo ra cảnh báo. Nguồn là một con số. Hành vi là một trong những điều sau đây: **falls below**, **becomes equal to**, hoặc **rises above a number specified for Value**. Giá trị là một số mô tả bộ đếm điều kiện thực thi.

Cảnh báo sự kiện WMI

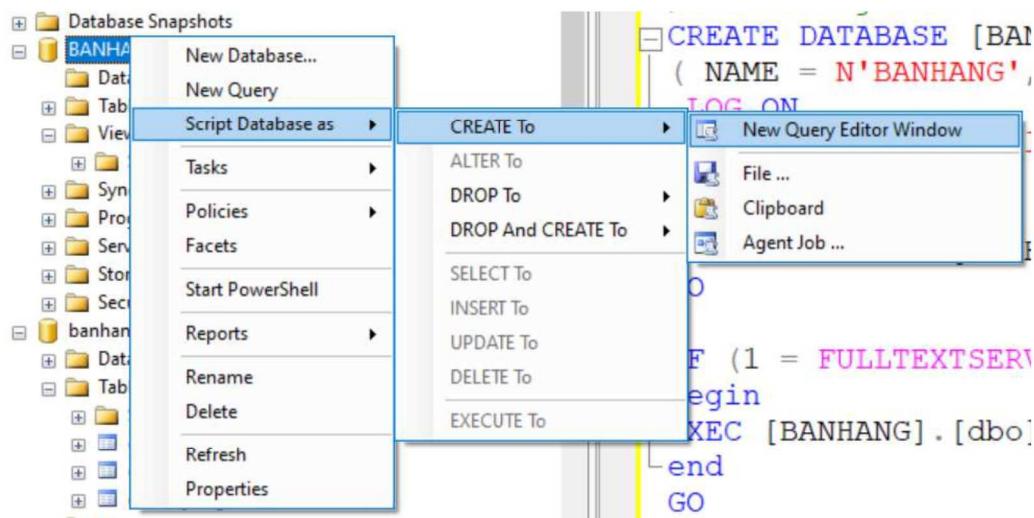
Ta có thể chỉ định rằng một cảnh báo xảy ra để phản ứng với một sự kiện WMI cụ thể, các thông số thiết lập như sau:

- **Namespace:** SQL Server Agent đăng ký dưới dạng máy khách WMI vào không gian tên WMI được cung cấp để truy vấn các sự kiện.
- **Query:** SQL Server Agent sử dụng câu lệnh Windows Management Instrumentation Query Language (WQL) được cung cấp để xác định sự kiện cụ thể.

4.1.4 Sinh mã lệnh cho các đối tượng trong Cơ sở dữ liệu

MicroSoft SQL Server Management Studio cung cấp công cụ cho phép tự động sinh mã lệnh từ các đối tượng do người dùng đã tạo như cơ sở dữ liệu, bảng, view, thủ tục, trigger... Các thao tác quản trị được thực hiện trực tiếp trong công cụ có thể được sinh mã lệnh để sử dụng trong lập trình thủ tục, job...

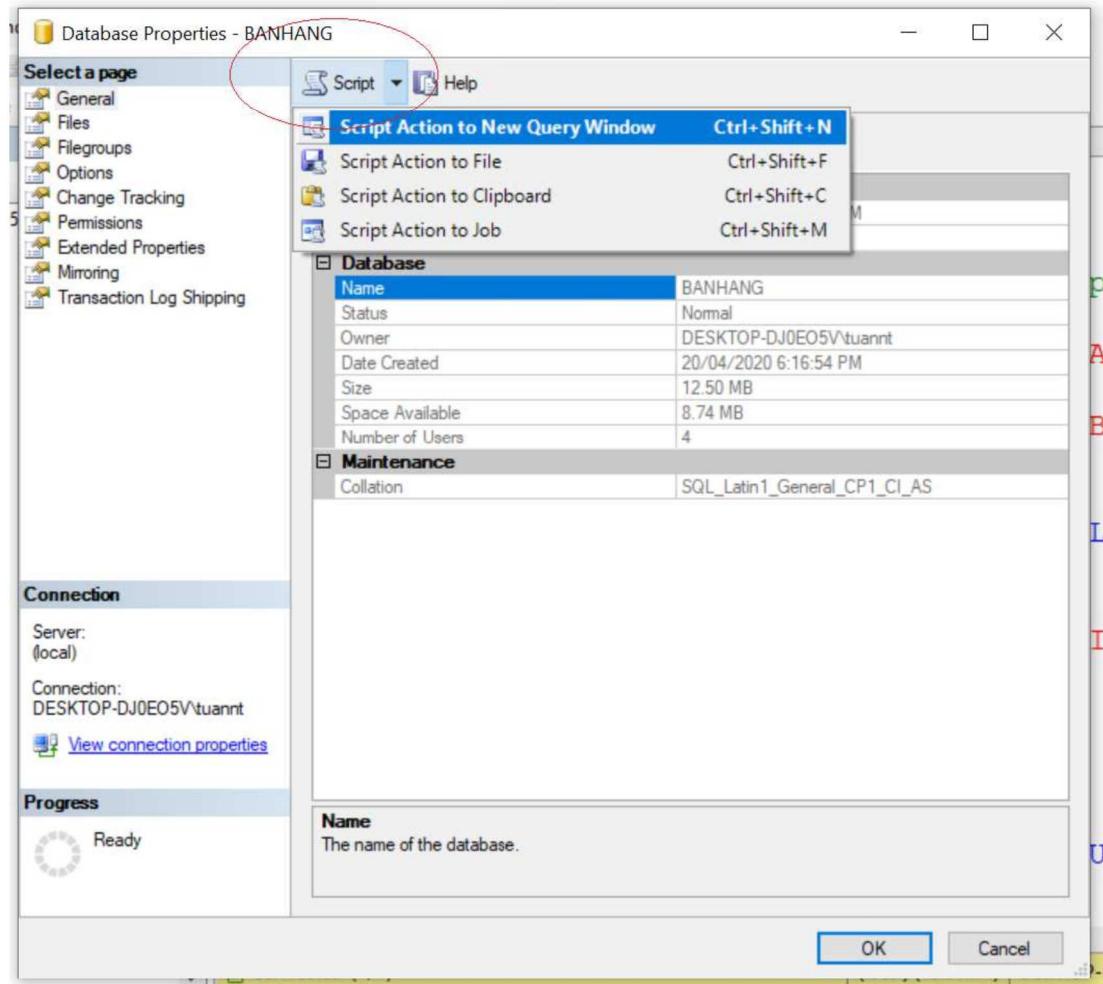
Để thực hiện, nhấn phím phải chuột lên đối tượng cần sinh mã lệnh, sau đó, chọn **Script <Loại đối tượng> as -> <hành động> -> <Đích lưu mã lệnh>**



Trong đó:

- <Loại đối tượng> là đối tượng mà ta muốn sinh mã lệnh để tạo, sửa, xóa..:
- <hành động>: mã lệnh thêm, sửa, xóa... cho đối tượng
- <Đích lưu mã lệnh>: MicroSoft SQL Server cho phép có thể sinh mã lệnh sau đó đưa vào cửa sổ lệnh của MicroSoft SQL Server Management Studio, ghi vào file, ghi vào bộ đệm hoặc tạo Job để thực hiện thao tác đó.

Phương pháp khác để thực hiện sinh các mã lệnh cho đối tượng là nhấn phím phải chuột lên đối tượng, sau đó chọn **Properties**, chọn nút **Script** trong hộp thoại của đối tượng đó.

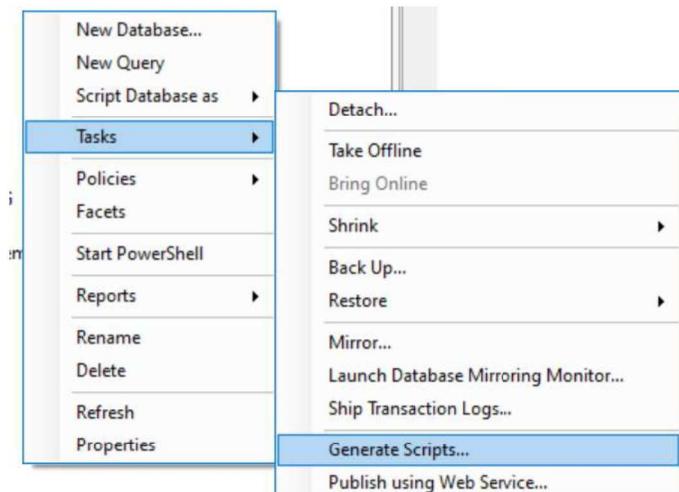


Chú ý:

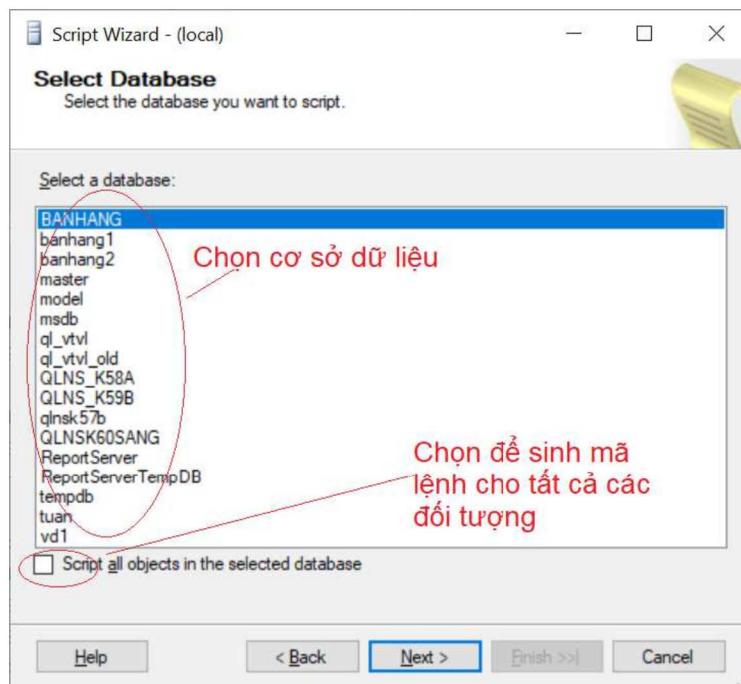
- + Khi sinh mã lệnh tự động, sẽ có nhiều lệnh được hệ thống thêm để đảm bảo các tính chất của đối tượng theo lý thuyết, đồng thời kèm theo những thông số hiện tại đối tượng trên MicroSoft SQL Server đang sử dụng, do đó, cần sửa đổi phù hợp khi sử dụng trong lập trình hoặc trong các hoạt động khác.

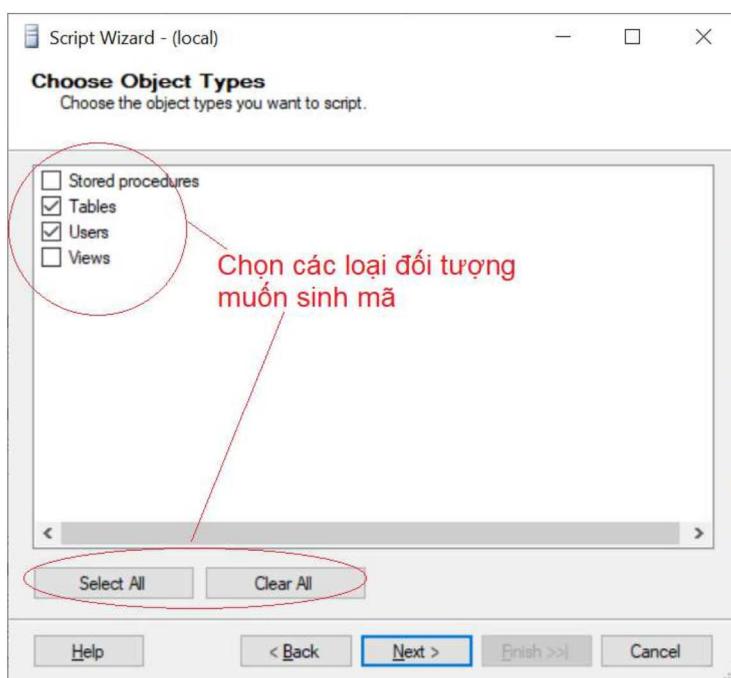
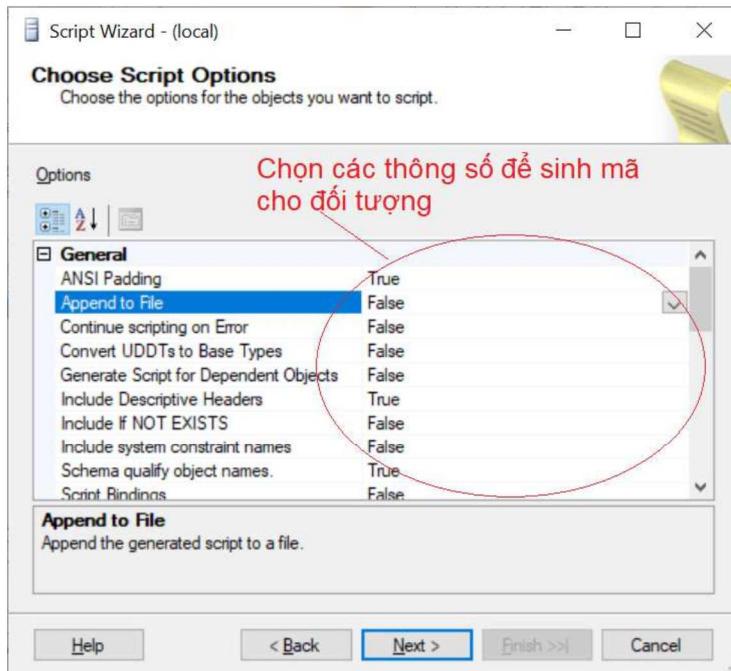
Sinh mã lệnh cho tất cả các đối tượng của cơ sở dữ liệu

Bước 1: Trong vùng Databases, nhấn phím phải chuột, chọn Tasks -> Generate Scripts

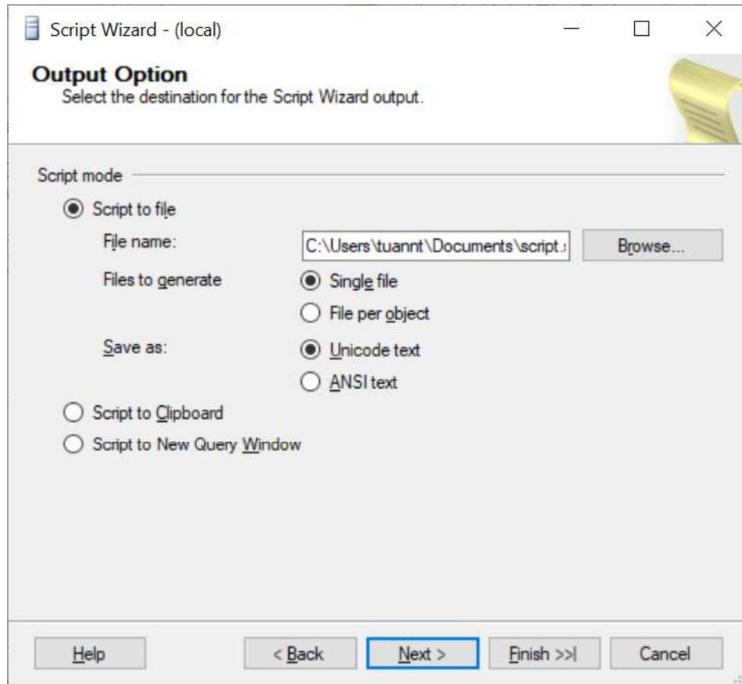


Bước 2: Lần lượt thực hiện các bước của quá trình sinh mã lệnh trên các cửa sổ màn hình hướng dẫn, qua mỗi cửa sổ chọn phím Next.





Các màn hình tiếp theo sẽ cho phép lựa chọn chi tiết cho từng loại đối tượng đã được chọn để sinh mã. Kết thúc ta chọn cách lưu mã lệnh đã sinh.



4.2 SAO LUƯU, PHỤC HỒI VÀ QUẢN LÝ CƠ SỞ DỮ LIỆU

4.2.1 Sao lưu

MicroSoft SQL Server cung cấp công cụ để cho quản trị viên hoặc lập trình viên có thể cấu hình hoặc lập trình để sao lưu (backup), phục hồi (restore) cơ sở dữ liệu. Các công việc sao lưu hoặc phục hồi có thể thực hiện tự động thông qua Job. Tùy thuộc vào các ứng dụng cụ thể, quản trị viên hoặc lập trình viên có thể lựa chọn các phương án sao lưu và phục hồi khác nhau, trên cùng một server hoặc các server khác nhau. Để đảm bảo an toàn dữ liệu, cần có kế hoạch sao lưu định kỳ cho các cơ sở dữ liệu.

MicroSoft SQL Server hỗ trợ ba loại sao lưu cơ sở dữ liệu:

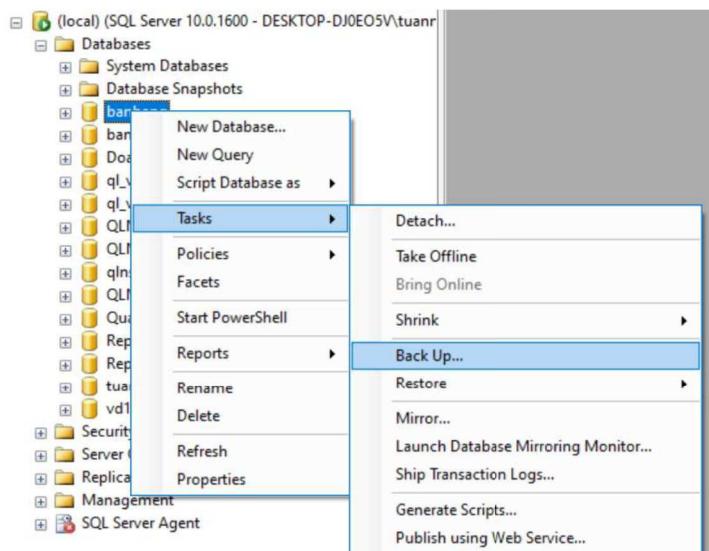
- **Full Backup:** Backup toàn bộ dữ liệu tại thời điểm thực hiện, đây là loại sao lưu thường được dùng nhất. Thực hiện sao lưu loại này sẽ đảm bảo được cơ sở dữ liệu đầy đủ nhất tính đến thời điểm sao lưu và đơn giản khi thực hiện thao tác phục hồi dữ liệu. Tuy nhiên, với những cơ sở dữ liệu có dung lượng lớn, loại sao lưu này sẽ mất nhiều thời gian và dung lượng lưu trữ.

- **Differential Backup:** Backup các dữ liệu mới được cập nhật kể từ lần **full backup** trước đó, được gọi là sao lưu sự sai khác hoặc sao lưu gia tăng. Loại sao lưu này sẽ có thời gian thực hiện nhanh hơn sao lưu toàn bộ, dung lượng lưu trữ bộ sao lưu nhỏ hơn. Tuy nhiên, quá trình thực hiện phục hồi cơ sở dữ liệu khi hệ thống có lỗi sẽ phức tạp hơn, nguy cơ xảy ra mất dữ liệu hoặc không thực hiện được việc phục hồi nếu thấy lạc bản sao lưu trung gian.
- **Transaction Log Backup:** Backup các *log record* hiện có trong log file, sao lưu các *hành động* (các thao tác xảy ra đối với database), không sao lưu dữ liệu. Đồng thời nó cũng cắt bỏ (truncate) log file, loại bỏ các log record vừa được backup ra khỏi log file.

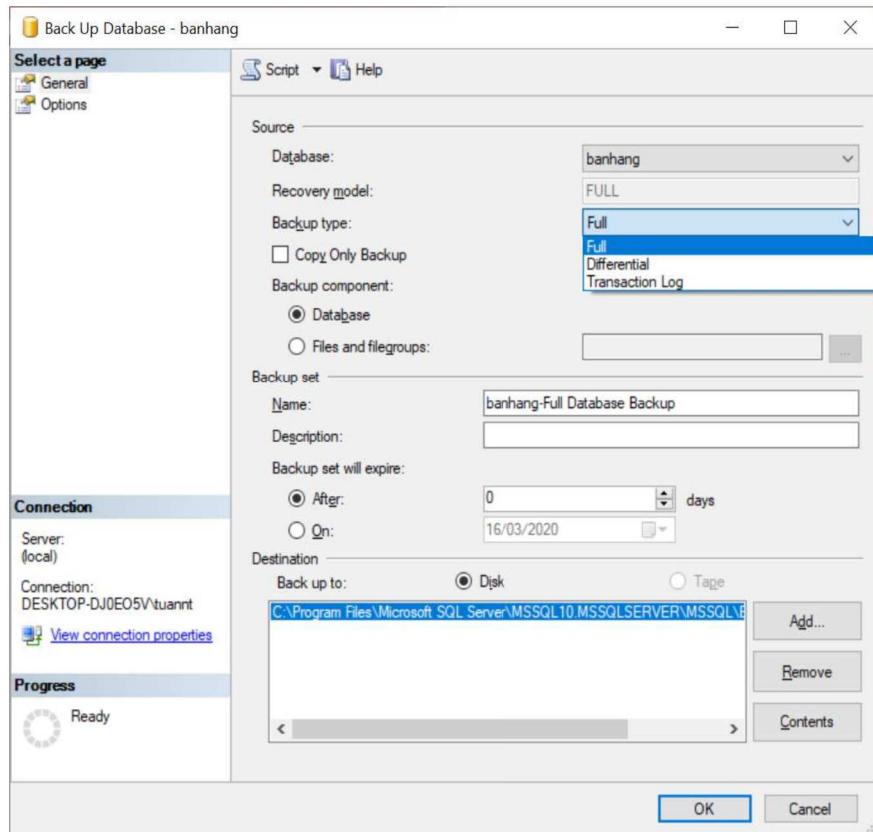
Với cơ sở dữ liệu có dung lượng lớn và được cập nhật liên tục, thì việc thực hiện **full backup** với tần suất cao là khó khả thi, vì nó dùng rất nhiều tài nguyên (CPU, I/O). Nhờ có **differential backup** và **transaction log backup**, có thể tạo lập các phương án sao lưu thích hợp, đảm bảo dữ liệu được backup thường xuyên hơn mà không chiếm nhiều tài nguyên của hệ thống. Người quản trị hệ thống cần có các chiến lược, kế hoạch sao lưu cụ thể cho từng cơ sở dữ liệu khác nhau và cần phải có các tài liệu lưu trữ cho các phương án sao lưu đó.

Để thực hiện sao lưu dữ liệu, ta làm theo các bước sau:

Bước 1: Chọn cơ sở dữ liệu cần sao lưu. Bấm phím phải chuột, chọn **Tasks -> Backup**



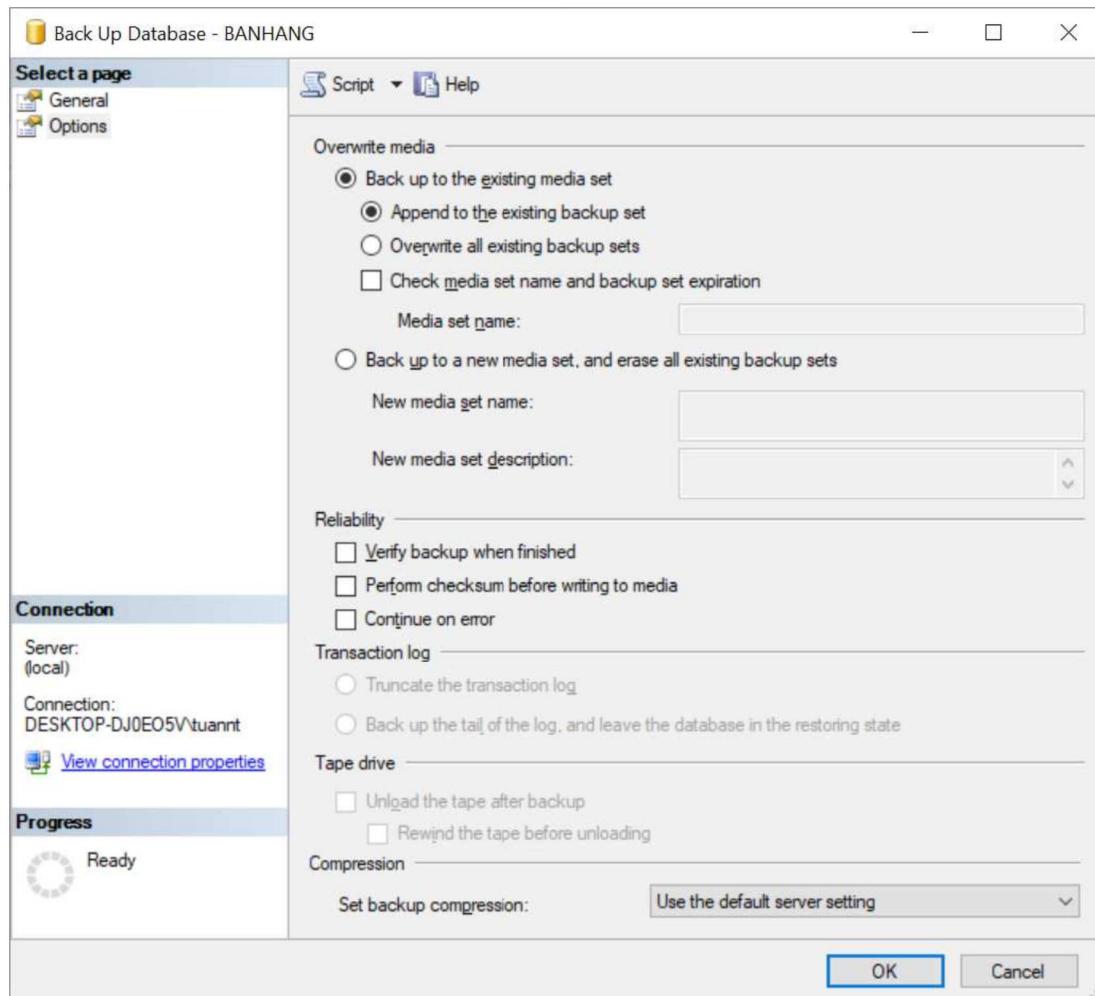
Bước 2: Thiết lập các thông số cho chức năng sao lưu trên khung giao diện



Trong đó:

- Database: chọn cơ sở dữ liệu muốn backup (nếu có thay đổi so với cơ sở dữ liệu đã chọn ban đầu)
- Recovery model: chọn cách phục hồi
- Backup type: chọn loại sao lưu (là một trong 3 loại trên)
- Backup component: sao lưu thành phần nào của cơ sở dữ liệu.
- Backup set: cho phép đặt tên và xác định thời hạn backup hiệu lực.
- Destination: Cho phép chọn nơi ghi dữ liệu backup. Trong một lần sao lưu, có thể ghi bản sao ra nhiều nơi khác nhau. Muốn thêm các vị trí lưu bản sao, chọn nút **Add**, muốn xóa vị trí bản sao không sử dụng, nhấn chọn vị trí và sau đó nhấn nút **Remove**.

Bước 3: Chọn mục Options để thiết lập các thông tin thực hiện sao lưu, ví dụ như chọn chế độ ghi đè dữ liệu, kiểm tra dữ liệu, nén dữ liệu...



Chú ý:

- + Việc sao lưu dữ liệu thường được thực hiện định kỳ, do vậy thao tác sao lưu thường được tạo Job để thực hiện tự động.
- + Tương tự như các đối tượng, sau khi chọn các thông số sao lưu trên công cụ, có thể sinh mã lệnh để lập trình...

Ví dụ mã lệnh sao lưu toàn bộ cơ sở dữ liệu Bán hàng do hệ thống tự sinh ra:

BACKUP DATABASE [BANHANG]

TO DISK = N'C:\QLBH\saoluuBH' WITH NOFORMAT, NOINIT,

```

NAME = N'BANHANG-Full Database Backup',
SKIP, NOREWIND, NOUNLOAD, STATS = 10
GO

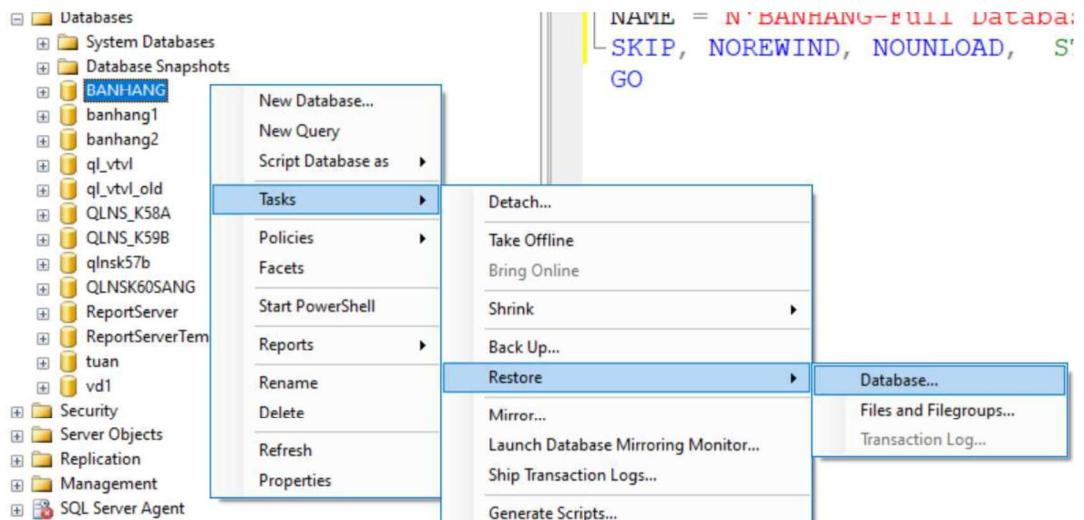
```

4.2.2 Phục hồi

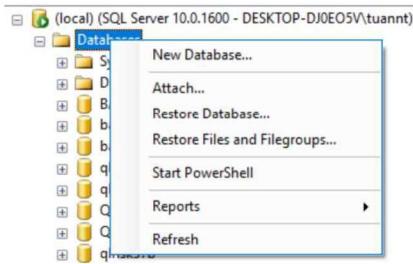
MicroSoft SQL Server cung cấp công cụ cho phép thực hiện phục hồi cơ sở dữ liệu khi gặp sự cố từ các bản sao lưu trước. Đôi với các cơ sở dữ liệu được sao lưu đầy đủ (Full Backup) và sao lưu nhiều lần, hệ thống sẽ có thể tự chọn các bản sao lưu mới nhất để phục hồi hoặc cho phép chọn bản sao bất kỳ để phục hồi. Với phương pháp sao lưu sự sai khác (Differential Backup), cần phải thực hiện khôi phục lần lượt các bản sao lưu theo dòng thời gian từ bản sao đầy đủ, cho tới các bản sao sai khác mà có dữ liệu được phục hồi tốt nhất.

Các bước thực hiện phục hồi dữ liệu như sau:

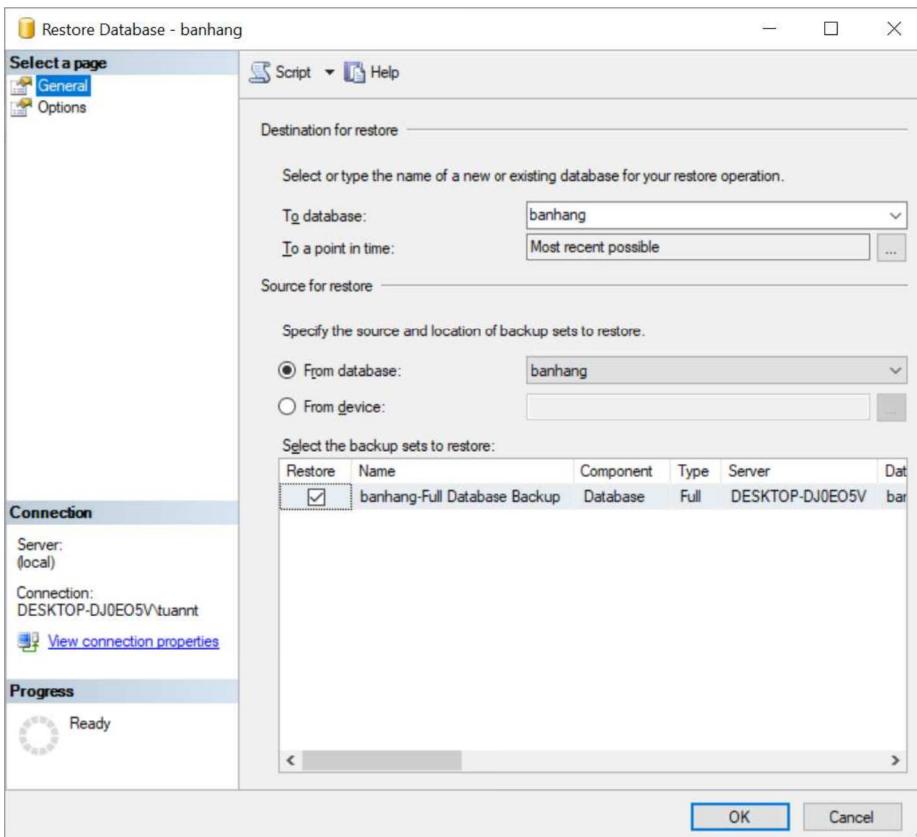
Bước 1: Nhấn phím phải chuột lên cơ sở dữ liệu cần phục hồi hoặc nhấn vào vùng có tên cơ sở dữ liệu của cây **Databases**, chọn **Tasks -> Restore**. Chọn **Database** nếu muốn khôi phục cho cả cơ sở dữ liệu, chọn **File and filegroups** nếu khôi phục các file trong cơ sở dữ liệu (tùy thuộc loại đã chọn khi sao lưu).



Hoặc

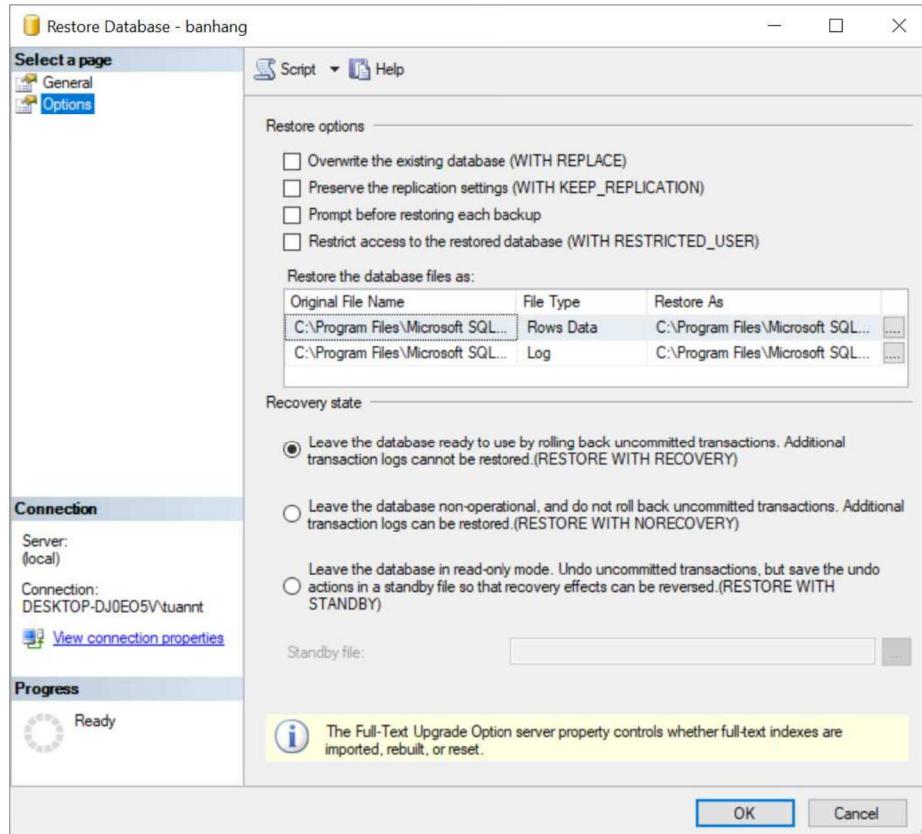


Bước 2: Chọn cơ sở dữ liệu để phục hồi và thời điểm phục hồi. Có thể phục hồi cho cơ sở dữ liệu đang được MicroSoft SQL Server quản lý trong danh sách hiện tại, hoặc phục hồi vào một cơ sở dữ liệu mới.



Cẩn trọng: Cơ sở dữ liệu sau khi phục hồi có thể ghi đè dữ liệu vào dữ liệu mới nhất, làm mất dữ liệu và không có khả năng khôi phục được.

Bước 3: Trong mục Options: lựa chọn các tham số để khôi phục dữ liệu, trong trường hợp vẫn còn dữ liệu cũ, có thể cảnh báo, ghi đè...



4.2.3 Quản lý cơ sở dữ liệu

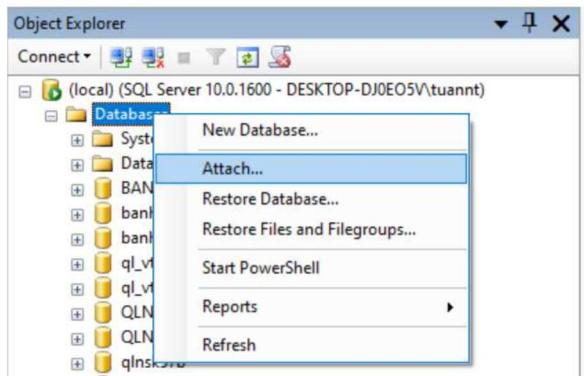
Detach và Attach cơ sở dữ liệu

MicroSoft SQL Server Management Studio quản lý các cơ sở dữ liệu do vậy, trong quá trình sử dụng, ta không thể thực hiện các lệnh của hệ điều hành lên cơ sở dữ liệu, ví dụ như sao chép cơ sở dữ liệu từ vị trí này sang vị trí khác... MicroSoft SQL Server Management Studio cung cấp công cụ để có thể tạm thời tách và dán các cơ sở dữ liệu vào hệ thống:

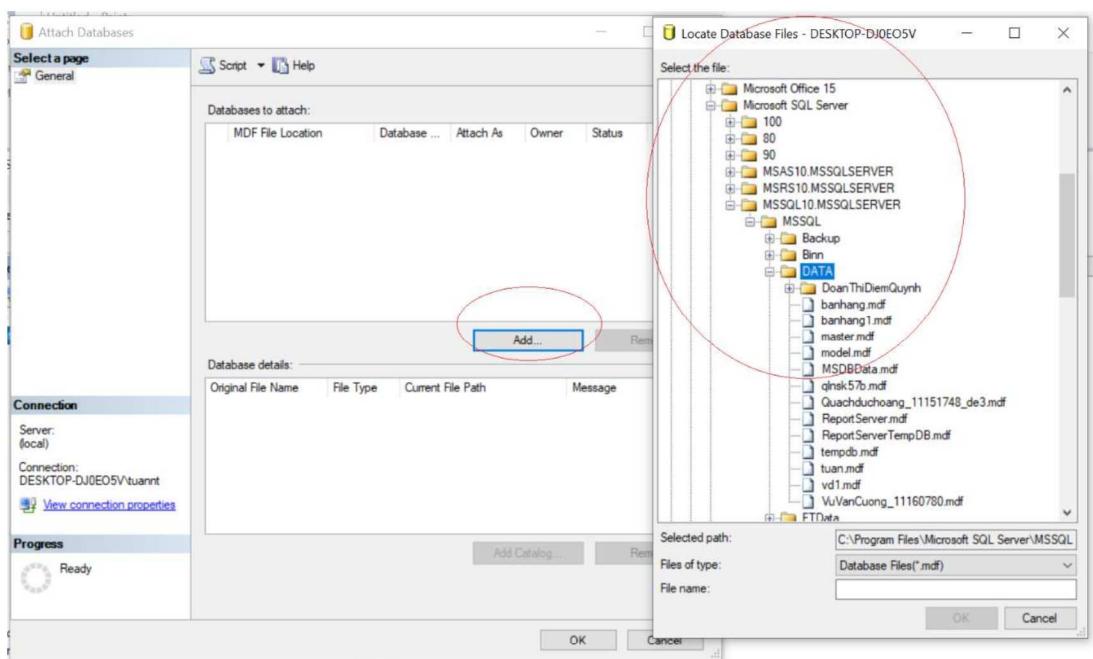
- **Attach:** Kết nối/dán file cơ sở dữ liệu vào SQL Server để quản lý và khai thác.
- **Detach:** Ngắt kết nối/tách cơ sở dữ liệu từ SQL Server để trở thành file dữ liệu trên phương tiện lưu trữ.

Để thực hiện Attach:

Bước 1: Nhấn phím phải chuột lên **Databases**, chọn **Attach**

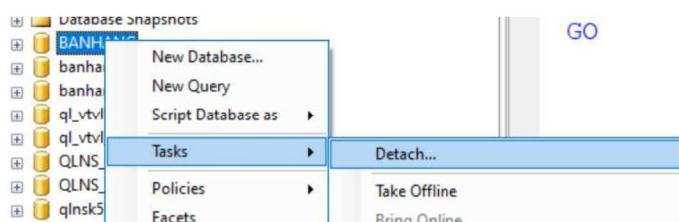


Bước 2: Chọn file cơ sở dữ liệu và các thông số khác khi kết nối cơ sở dữ liệu.

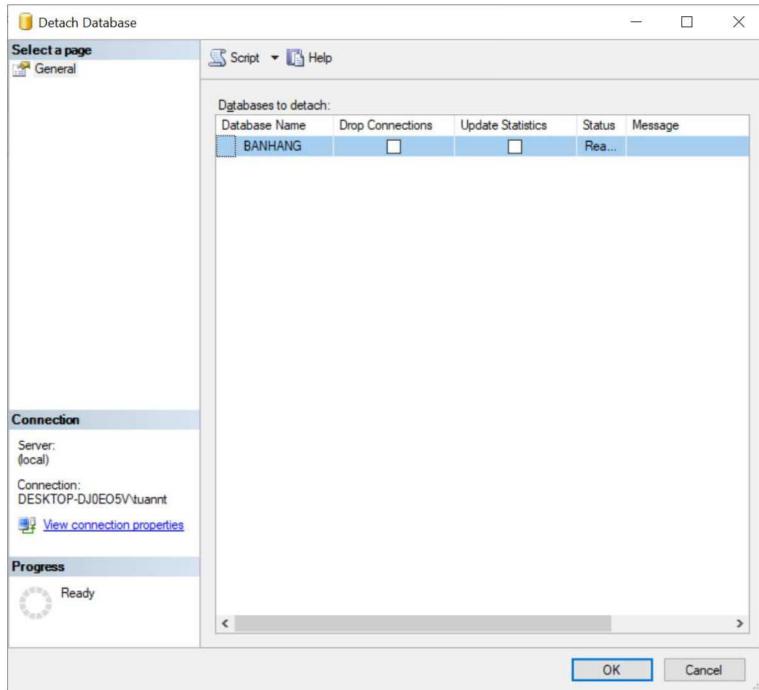


Để thực hiện Detach:

Bước 1: Nhấn phím phải chuột lên cơ sở dữ liệu muốn ngắt kết nối, chọn **Tasks -> Detach**



Bước 2: Lựa chọn các tùy chọn khi ngắt cơ sở dữ liệu khỏi hệ thống



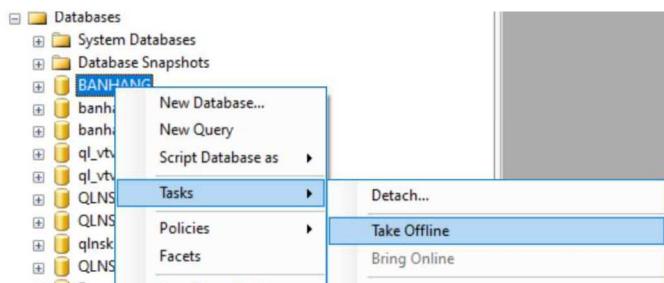
Chú ý:

- + Trước khi thực hiện thao tác Detach, cần nắm rõ vị trí lưu trữ file cơ sở dữ liệu để sau đó có thể tìm được lại cơ sở dữ liệu khi cần Attach.

Take Offline và Bring Online cơ sở dữ liệu

MicroSoft SQL Server Management Studio cung cấp công cụ cho phép tạm ngừng sử dụng cơ sở dữ liệu hoặc bật lại cơ sở dữ liệu để tiếp tục hoạt động. Để thực hiện tính năng này, nhấn phím phải chuột lên cơ sở dữ liệu, sau đó chọn **Tasks -> Take Offline hoặc Bring Online**. Với các tính năng này, cơ sở dữ liệu vẫn do SQL Server quản lý mà chỉ cho phép/không cho phép người sử dụng tương tác với cơ sở dữ liệu.

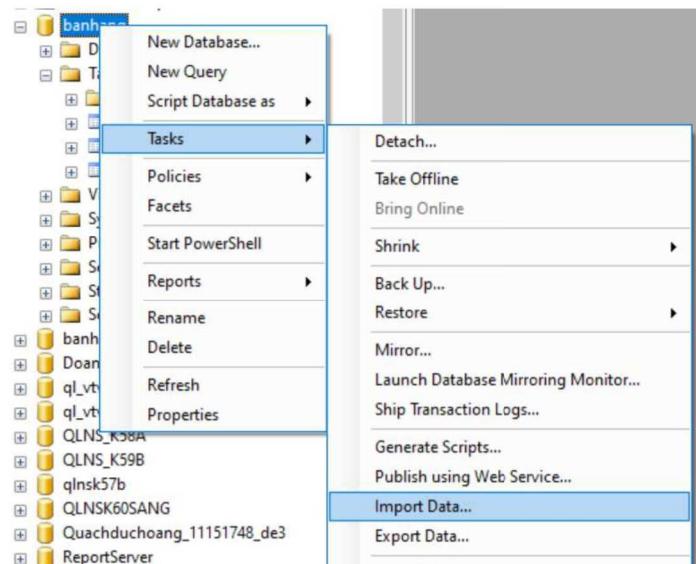
- **Take Offline:** Chuyển chế độ của cơ sở dữ liệu từ trực tuyến sang ngoại tuyến trong SQL Server, khi đó, các truy cập không thể thực hiện được với cơ sở dữ liệu.
- **Bring Online:** Chuyển chế độ của cơ sở dữ liệu từ ngoại tuyến sang trực tuyến, người sử dụng có thể truy cập một cách bình thường.



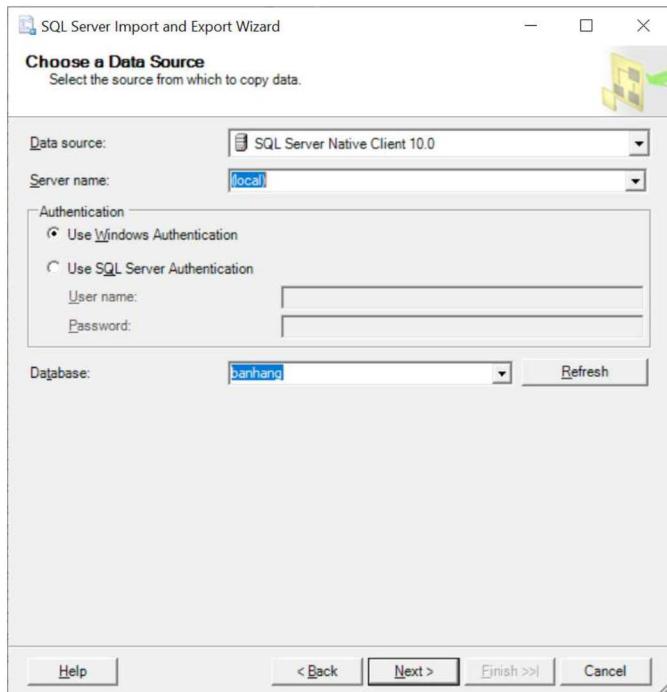
4.3 NHẬP (IMPORT) VÀ XUẤT (EXPORT) DỮ LIỆU

Để thực hiện việc chuyển đổi dữ liệu từ các hệ thống và định dạng khác vào MicroSoft SQL Server hoặc ngược lại, MicroSoft SQL Server Management Studio cung cấp công cụ cho phép thực hiện thao tác xuất dữ liệu (Export) hoặc nhập dữ liệu (Import) thông qua các giao diện hội thoại. Các bước thực hiện như sau:

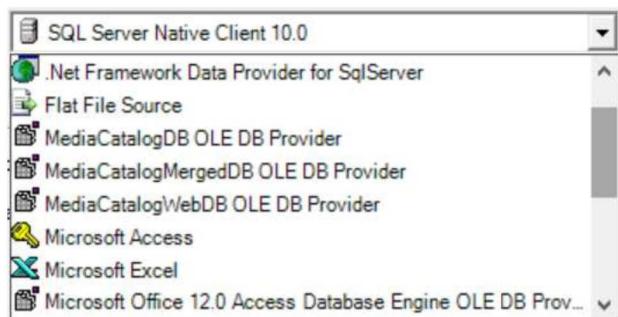
Bước 1: Chọn tính năng Import hoặc Export trong hệ thống bằng cách nhấn phím phải chuột lên cơ sở dữ liệu muốn thực hiện, sau đó chọn **Tasks -> Import data** hoặc **Export data**.



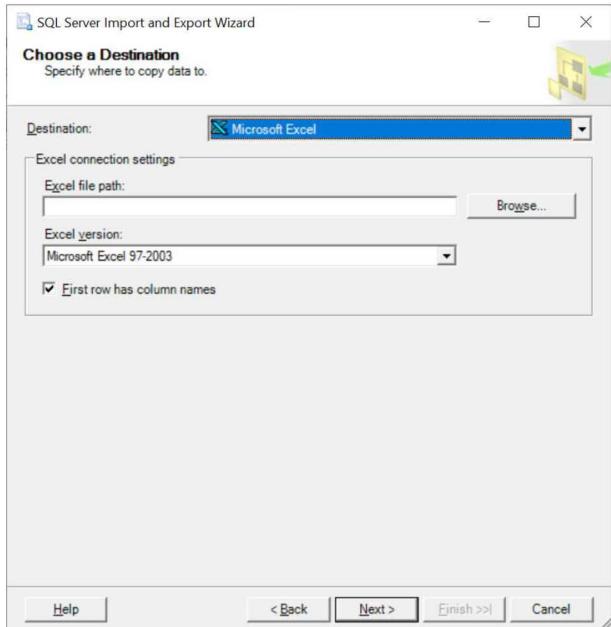
Bước 2: Chọn nguồn để nhập/xuất. Mục **Data source**: nguồn dữ liệu cần import/export. Tùy thuộc nguồn dữ liệu, giao diện hệ thống sẽ thay đổi theo, cho phép kết nối, chọn dữ liệu và khai báo thêm các tham số cho các nguồn đó.



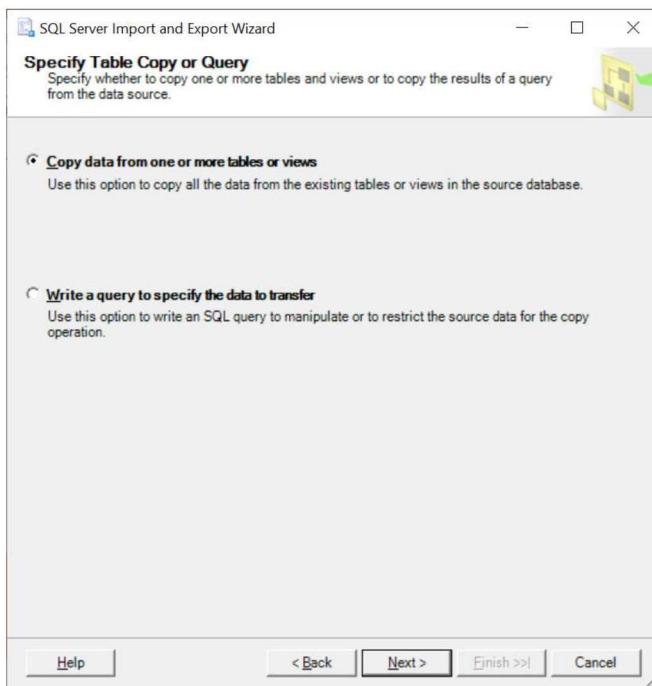
Các nguồn dữ liệu



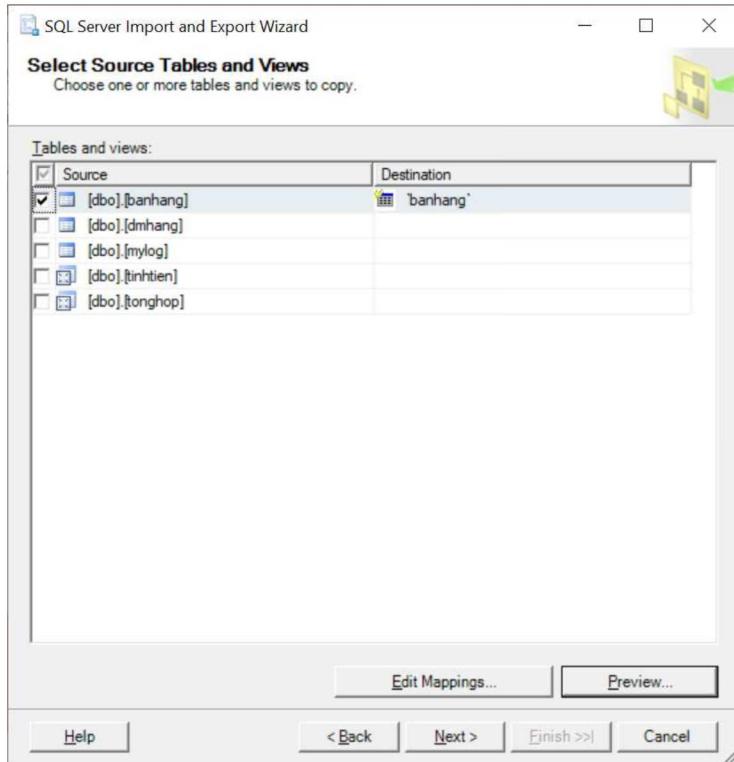
Bước 3: Chọn đích để xuất/nhập dữ liệu. Khung hội thoại cho phép lựa chọn nơi để import dữ liệu vào hoặc export dữ liệu ra cũng tương tự như khi chọn nguồn. Khung hội thoại cũng sẽ tự thay đổi tham số khi ta chọn các loại đích khác nhau trong mục Destination.



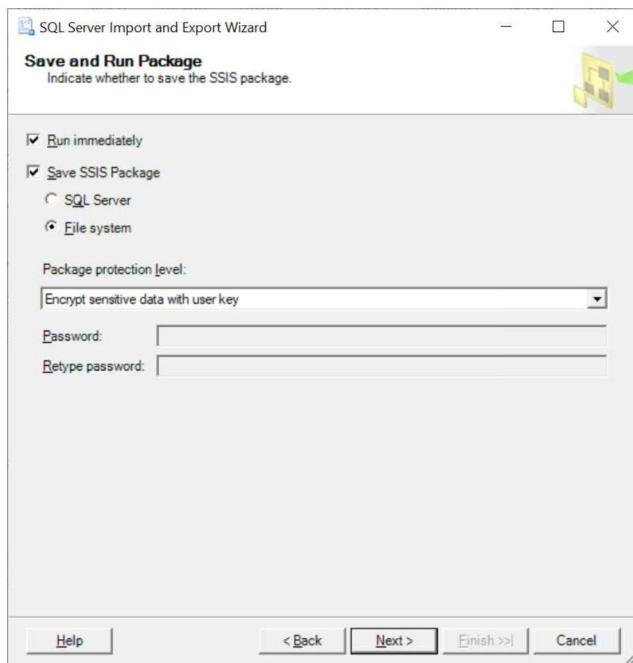
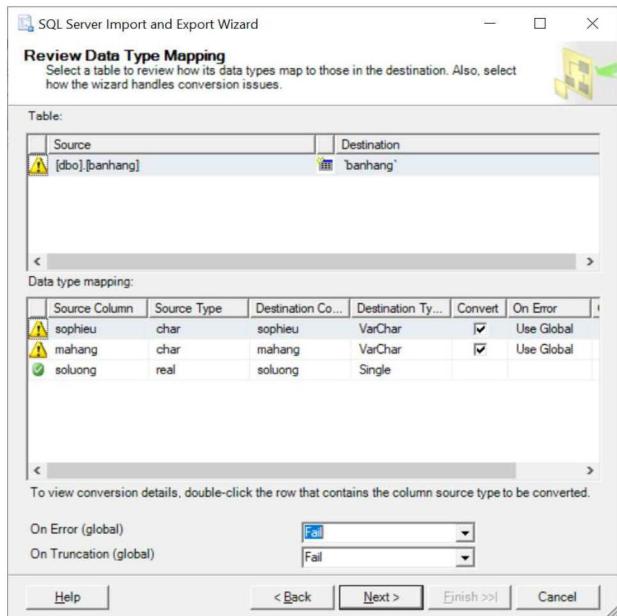
Bước 4: Cho phép chọn bảng để lưu trữ hoặc sinh lệnh SQL để thực hiện.



Bước 5: Chọn các bảng/view muốn xuất dữ liệu. Trong trường hợp nhập dữ liệu, chọn bảng đã tồn tại hoặc tạo bảng mới. **Edit mappings:** cho phép thiết lập ánh xạ các trường khi nhập/xuất dữ liệu.



Bước 6: Cho phép xem lại ánh xạ của các trường trước khi thực hiện. Bước tiếp theo cho phép đặt cấu hình thực hiện xuất dữ liệu và có thể ghi dữ liệu thành gói **SSIS** (SSIS - viết tắt của SQL Server Integration Services, là gói dữ liệu để cho phép thực hiện dịch vụ tích hợp dữ liệu giữa các hệ thống). Thực hiện quá trình nhập/xuất dữ liệu bằng cách nhấn **Finish**.



4.4 BẢO MẬT TRONG SQL SERVER

4.4.1 Giới thiệu về bảo mật SQL Server

Bảo mật là một trong những yếu tố đóng vai trò quan trọng đối với sự sống còn của cơ sở dữ liệu. Hầu hết các hệ quản trị cơ sở dữ liệu thương mại hiện nay đều cung cấp khả năng bảo mật cơ sở dữ liệu với những chức năng như:

- Cấp phát quyền truy cập cơ sở dữ liệu cho người dùng và các nhóm người dùng, phát hiện và ngăn chặn những thao tác trái phép của người sử dụng trên cơ sở dữ liệu.
- Cấp phát quyền sử dụng các câu lệnh, các đối tượng cơ sở dữ liệu đối với người dùng.
- Thu hồi (huỷ bỏ) quyền của người dùng.

Bảo mật dữ liệu trong SQL Server được thực hiện dựa trên ba khái niệm chính sau đây:

- **Người dùng cơ sở dữ liệu (Database user):** là đối tượng sử dụng cơ sở dữ liệu, thực thi các thao tác trên cơ sở dữ liệu như tạo bảng, truy xuất dữ liệu,... Mỗi một người dùng trong cơ sở dữ liệu được xác định thông qua tên người dùng (User ID). Một tập nhiều người dùng có thể được tổ chức trong một nhóm và được gọi là nhóm người dùng (User Group). Chính sách bảo mật cơ sở dữ liệu có thể được áp dụng cho mỗi người dùng hoặc cho các nhóm người dùng.
- **Các đối tượng cơ sở dữ liệu (Database objects):** tập hợp các đối tượng, các cấu trúc lưu trữ được sử dụng trong cơ sở dữ liệu như bảng, khung nhìn, thủ tục, hàm được gọi là các đối tượng cơ sở dữ liệu. Đây là những đối tượng cần được bảo vệ trong chính sách bảo mật của cơ sở dữ liệu.
- **Đặc quyền (Privileges):** Là tập những thao tác được cấp phát cho người dùng trên các đối tượng cơ sở dữ liệu. Chẳng hạn một người dùng có thể truy xuất dữ liệu trên một bảng bằng câu lệnh SELECT nhưng có thể không thể thực hiện các câu lệnh INSERT, UPDATE hay DELETE trên bảng đó.

Trong MicroSoft SQL Server, mỗi người dùng được gọi là một login hoặc user, khi làm việc với server được gọi là login, khi làm việc với cơ sở dữ liệu hoặc

đối tượng trong cơ sở dữ liệu được gọi là user. MicroSoft SQL Server có chính sách sử dụng chung login của hệ điều hành, hoặc có thể tạo login riêng.

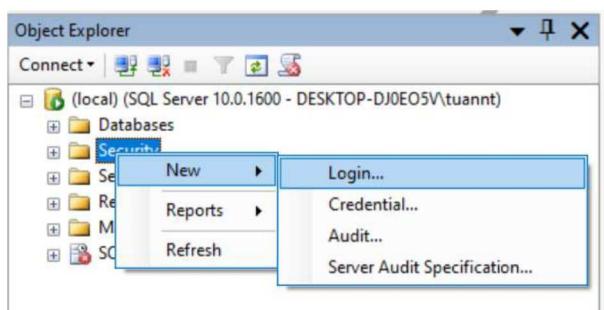
- **Login:** dùng để truy cập vào hệ thống SQL Server, các login chỉ mới có quyền truy cập vào Server, chưa hẳn có quyền truy cập vào các Database trên Server, các quyền truy cập vào Database được gắn liền với các người dùng cơ sở dữ liệu (User).
- **User:** Mỗi cơ sở dữ liệu có một danh sách các người dùng được phép truy cập cơ sở dữ liệu của mình, mỗi user luôn được gắn (mapped) với một login ở mức Server. Khi đăng nhập vào SQL Server thông qua login này, sẽ có quyền truy cập vào cơ sở dữ liệu theo quyền hạn mà user tương ứng với nó được cấp. Mỗi login có thể gắn với một hoặc nhiều user với quyền hạn khác nhau trên các cơ sở dữ liệu.

Việc thực hiện quản trị và phân quyền có thể thông qua công cụ trực quan MicroSoft SQL Server Management Studio hoặc có thể lập trình trực tiếp. Các phần dưới đây sẽ trình bày về phương thức quản trị SQL Server bằng công cụ và bằng lệnh.

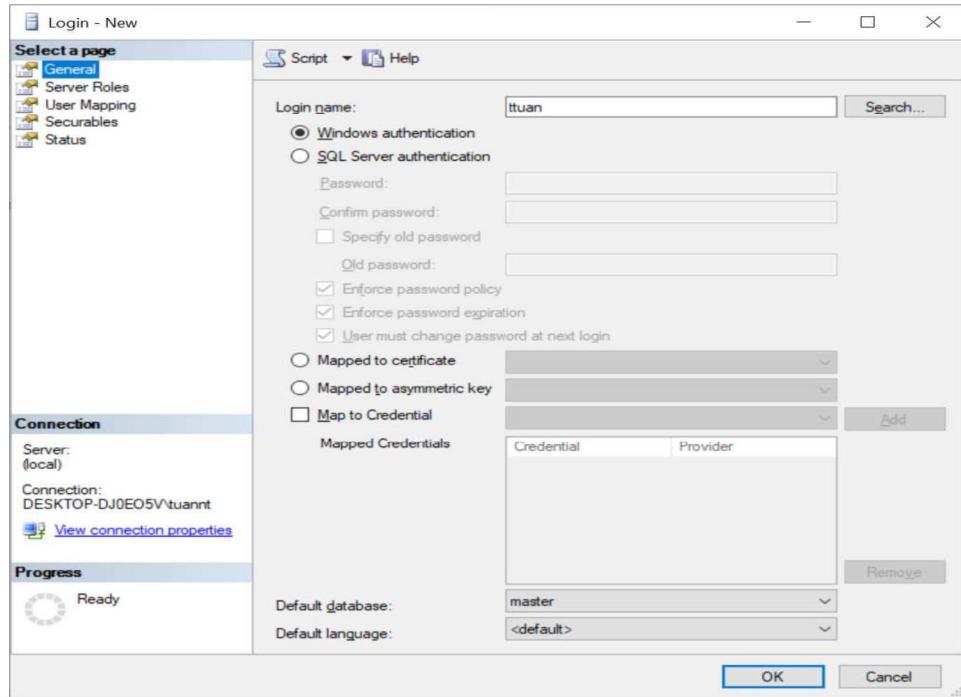
4.4.2 Thực thi và quản trị bảo mật CSDL

Quản trị người dùng cho máy chủ (login)

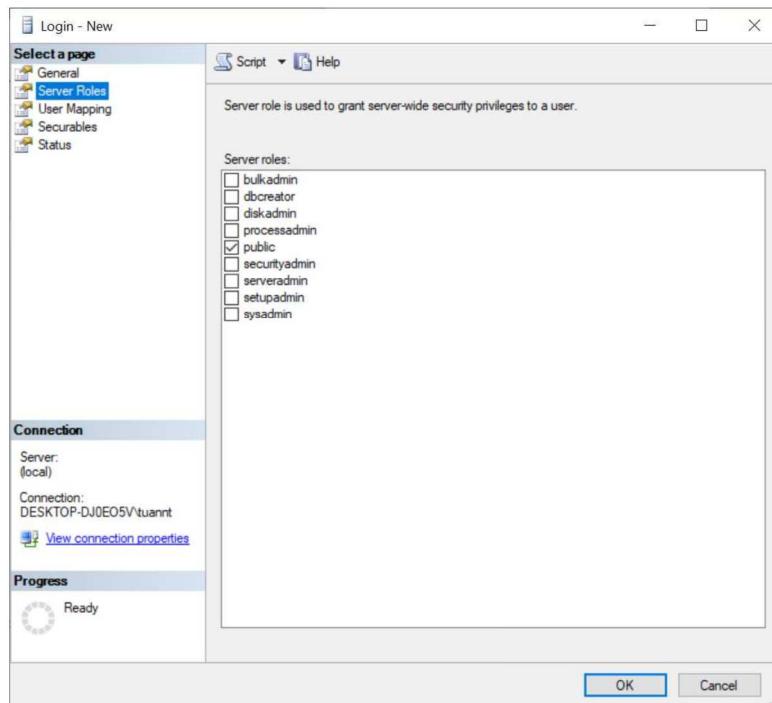
Bước 1: Tạo người dùng mới bằng cách nhấn phím phải chuột lên **Security**, chọn **New -> Login**.



Bước 2: Khai báo các thông tin chung cho người dùng trên giao diện: tên người dùng, cách xác thực người dùng bằng tài khoản Windows hoặc tài khoản của SQL Server, cơ sở dữ liệu ngầm định khi login,...



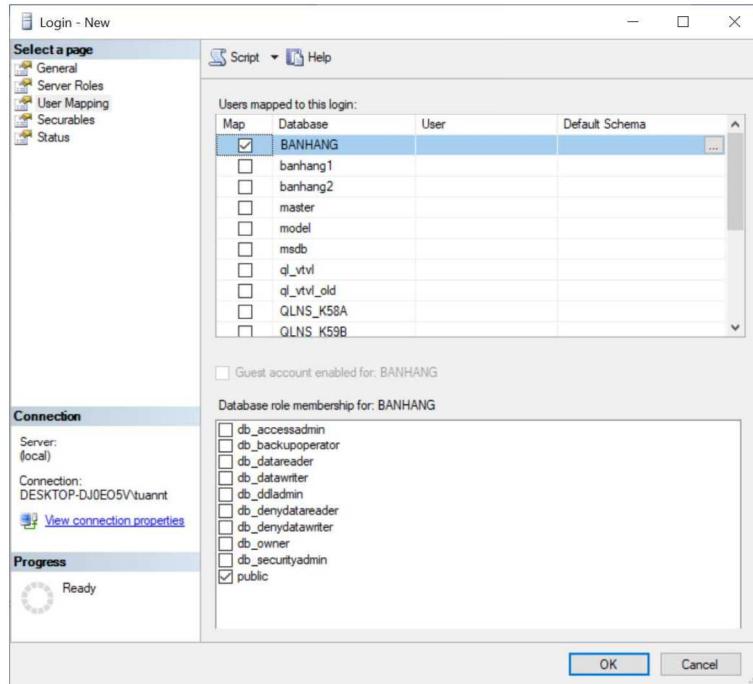
Bước 3: Chọn quyền trên Server cho người dùng tại mục **Server Roles**.



Ý nghĩa của các role với server như sau:

- **bulkadmin** : Account nào là thành viên của Role này có thể thực hiện các lệnh BULK INSERT
- **dbcreator** : Account nào là thành viên của Role này có thể Create, Alter, Drop, và restore bất cứ một database nào
- **diskadmin**: Account nào là thành viên của Role này có thể quản lý các tập tin trên đĩa cứng
- **processadmin**: Account nào là thành viên của Role này có thể tắt các xử lý đang chạy trong instance của database engine
- **public**: Account nào là thành viên của Role này có quyền mặc định của Microsoft SQL Server.
- **securityadmin**: Account nào là thành viên của Role này có quyền quản lý các account khác và cấp quyền cũng như thu hồi quyền của các account khác. Hơn nữa còn có quyền re-set mật khẩu của các account khác,
- **setupadmin**: Account nào là thành viên của Role này có thể thêm hoặc xóa các linked Server và thực thi một số store procedure hệ thống.
- **sysadmin**: Account nào là thành viên của Role này có toàn quyền trên hệ quản trị cơ sở dữ liệu

Bước 4: Chọn các DataBase mà tài khoản có thể kết nối tới, **Schema** bên trong Database. Chọn các quyền đối với cơ sở dữ liệu, các quyền ở phần này có phạm vi ảnh hưởng là trên cơ sở dữ liệu đã chọn.



Các quyền đối với cơ sở dữ liệu gồm:

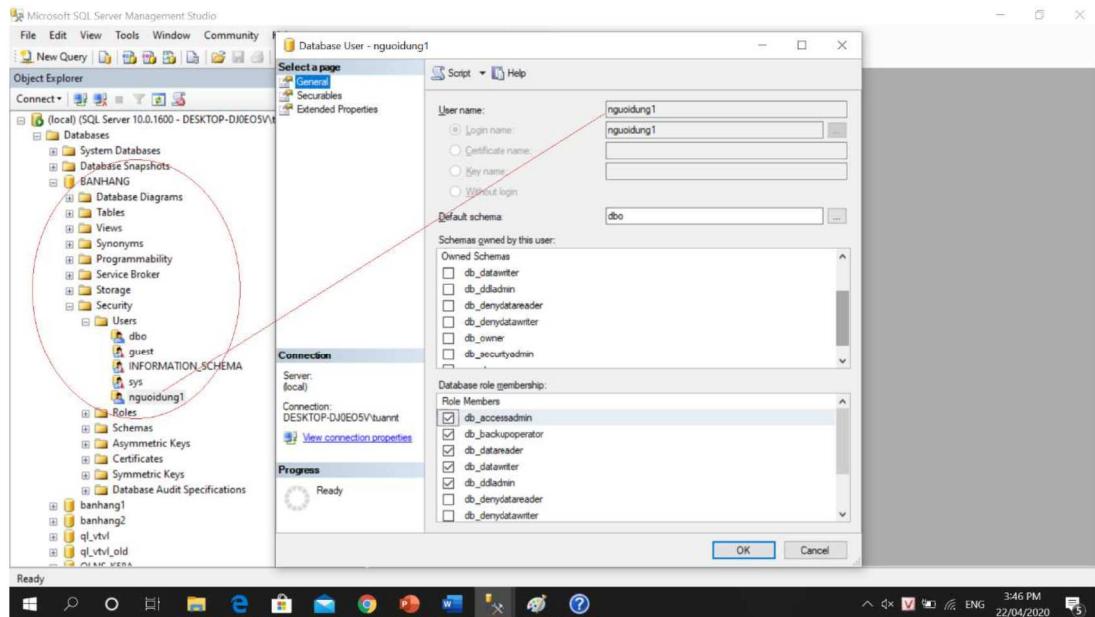
- **db_owner:** Account nào có role này có toàn quyền trên Database đó
- **db_securityadmin:** Account nào có role này có thể cấp quyền cho các account khác trong phạm vi database của account đó
- **db_accessadmin:** Account nào có role này có thể cấp quyền truy xuất database đó Windows login, Windows group và SQL Server logins.
- **db_backupoperator:** Account nào có role này có thể backup database
- **db_ddladmin:** Account nào có role này có thể chạy bất cứ câu lệnh DDL (Data Definition Language) trong database
- **db_datawriter:** Account nào có role này có thể thêm/sửa/xóa các dòng dữ liệu trong tất cả các user table trong database
- **db_denydatawriter:** Account nào có role này không thể thêm/sửa/xóa dữ liệu trong tất cả các user table trong database
- **db_datareader:** Account nào có role này có thể đọc dữ liệu tất cả các user table trong database

- **db_denydatareader:** Account nào có role này không thể đọc dữ liệu tất cả các user table trong database

Bước 5: Đặt các trạng thái cho login trong mục **Status**, nhấn nút **Ok** để kết thúc.

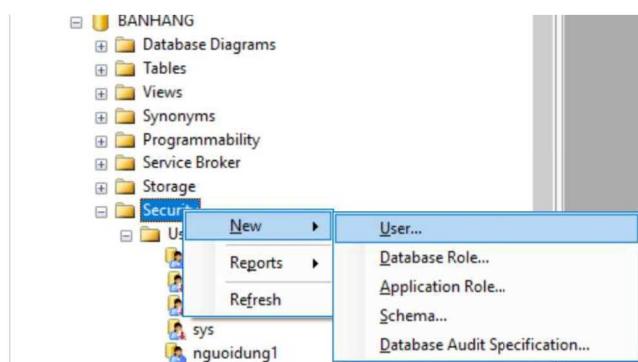
Chia quyền người dùng cho các đối tượng cơ sở dữ liệu

Bước 1: Mở cơ sở dữ liệu muốn quản trị, trong **Security**, chọn **Users**, nhấn phím phải chuột lên tên người dùng muốn phân quyền, chọn **Properties**.

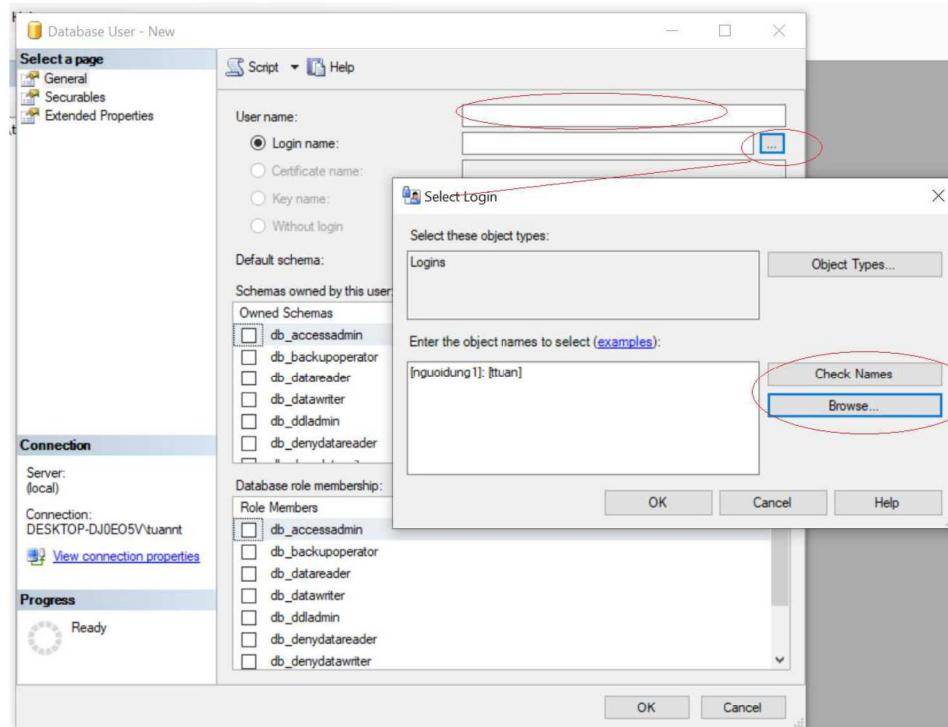


Bước 2: Thực hiện chọn các quyền cho người dùng.

Đối với người dùng của cơ sở dữ liệu, có thể tạo ra người dùng mới dựa trên các quyền của những người dùng trong máy chủ (login), bằng cách nhấn phím phải chuột trong mục **Security**, chọn **New -> User**.

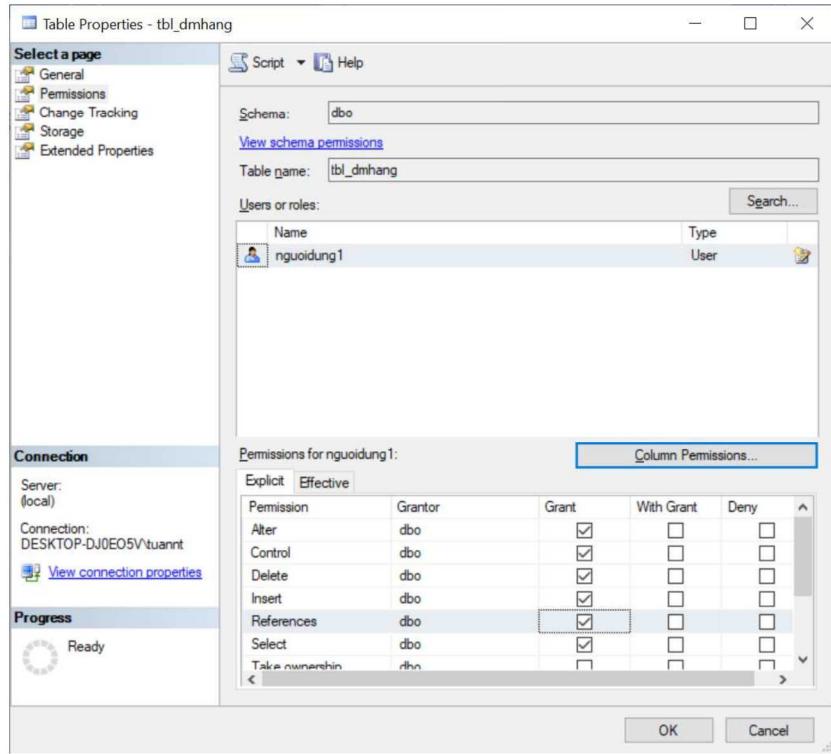


Thiết lập các thông tin cho người dùng mới trong hội thoại sau:

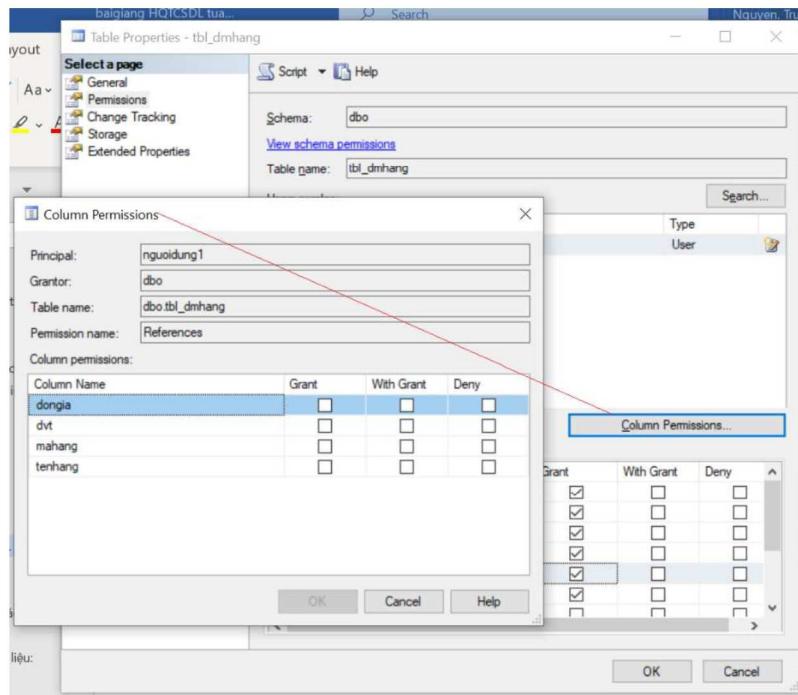


Phân quyền cho người dùng cho từng đối tượng trong cơ sở dữ liệu

Bước 1: Chọn đối tượng cần phân quyền, nhấn phím phải chuột lên đối tượng, chọn **Properties**, trong hội thoại chọn mục **Permissions**.



Bước 2: Chọn quyền cho người dùng. Trong hội thoại, có thể chọn các chi tiết quyền cho từng loại đối tượng khác nhau. Ví dụ, đối với bảng, có thể phân quyền đến từng cột.



Chú ý:

- + Đối với người dùng, quyền cho phép (allow) sẽ là số quyền ít nhất, quyền cấm (deny) sẽ là số quyền nhiều nhất giữa các nhóm.
- + Để xem tất cả các login của Server, dùng lệnh EXEC sp_helplogins.
- + Để xem tất cả các users của một database, dùng lệnh EXEC sp_helpuser.

4.4.3 Gán quyền và thu hồi quyền

MicroSoft SQL Server cung cấp hai câu lệnh cho phép thiết lập các chính sách bảo mật trong cơ sở dữ liệu.

- Lệnh GRANT: Sử dụng để cấp phát quyền cho người sử dụng trên các đối tượng cơ sở dữ liệu hoặc quyền sử dụng các câu lệnh SQL trong cơ sở dữ liệu.
- Lệnh REVOKE: Được sử dụng để thu hồi quyền đối với người sử dụng.

a) Cấp phát quyền

Câu lệnh GRANT được sử dụng để cấp phát quyền cho người dùng hay nhóm người dùng trên các đối tượng cơ sở dữ liệu. Chỉ người sở hữu cơ sở dữ liệu hoặc người sở hữu đối tượng cơ sở dữ liệu mới có thể cấp phát quyền cho người dùng trên các đối tượng cơ sở dữ liệu. Câu lệnh này thường được sử dụng trong các trường hợp sau:

- Người sở hữu đối tượng cơ sở dữ liệu muốn cho phép người dùng khác quyền sử dụng những đối tượng đang sở hữu.
- Người sở hữu cơ sở dữ liệu cấp phát quyền thực thi các câu lệnh (ví dụ CREATE TABLE, CREATE VIEW,...) cho những người dùng khác.

Cú pháp:

GRANT ALL [PRIVILEGES]

| các_quyền_cấp_phát [(danh_sách_cột)]]) ON tên_bảng|tên_khung_nhìn
|ON tên_bảng| tên_khung_nhìn[(danh_sách_cột)])]
|ON tên_thủ_tục

|ON tên_hàm

TO danh_sách_người_dùng| nhóm_người_dùng [WITH GRANT OPTION]

Trong đó:

- ALL [PRIVILEGES]: cấp phát tất cả các quyền cho người dùng trên đối tượng cơ sở dữ liệu được chỉ định, không cần liệt kê các lệnh.
- Các quyền có thể cấp phát cho người dùng bao gồm:
 - + Đối với bảng, khung nhìn, và hàm trả về dữ liệu kiểu bảng: SELECT, INSERT, DELETE, UPDATE và REFERENCES (quyền REFERENCES được sử dụng nhằm cho phép tạo khóa ngoài tham chiếu đến bảng cấp phát).
 - + Đối với cột trong bảng, khung nhìn: SELECT và UPDATE.
 - + Đối với thủ tục lưu trữ và hàm vô hướng: EXECUTE.
- *các_quyền_cấp_phát*: Danh sách các quyền cần cấp phát cho người dùng trên đối tượng cơ sở dữ liệu được chỉ định. Các quyền được phân cách nhau bởi dấu phẩy
- *tên_bảng | tên_khung_nhìn*: Tên của bảng hoặc khung nhìn cần cấp phát quyền.
- *danh_sách_cột*: Danh sách các cột của bảng hoặc khung nhìn cần cấp phát quyền.
- *tên_thủ_tục*: Tên của thủ tục được cấp phát cho người dùng.
- *tên_hàm*: Tên hàm (do người dùng định nghĩa) được cấp phát quyền.
- *danh_sách_người_dùng*: Danh sách tên người dùng nhận quyền được cấp phát. Tên của các người dùng được phân cách nhau bởi dấu phẩy.
- WITH GRANT OPTION Cho phép người dùng chuyển tiếp quyền cho người dùng khác.

Ví dụ 1:

Cho phép người dùng có tên nguoidung1 thực hiện câu lệnh SELECT, INSERT, UPDATE trên bảng tbl_dmhang.

Câu lệnh:

```
GRANT SELECT,INSERT,UPDATE  
ON tbl_dmhang  
TO nguoidung1
```

Ví dụ 2:

Cho phép người dùng có tên nguoidung1 quyền xem cột mã hàng, tên hàng trên bảng tbl_dmhang.

Câu lệnh:

```
GRANT SELECT  
(mahang,tenhang) ON tbl_dmhang  
TO nguoidung1
```

hoặc:

```
GRANT SELECT  
ON tbl_dmhang (mahang,tenhang) TO nguoidung1
```

Ví dụ 3:

Cho phép người dùng có tên nguoidung1 quyền xem bảng tbl_dmhang và cho phép chuyển quyền tới người dùng khác.

Câu lệnh:

```
GRANT SELECT  
ON tbl_dmhang  
TO nguoidung1  
WITH GRANT OPTION
```

Cấp phát quyền thực thi các câu lệnh

Ngoài chức năng cấp phát quyền cho người sử dụng trên các đối tượng cơ sở dữ liệu, câu lệnh GRANT còn có thể sử dụng để cấp phát cho người sử dụng một số quyền trên hệ quản trị cơ sở dữ liệu hoặc cơ sở dữ liệu. Những quyền có thể cấp phát trong trường hợp này bao gồm: CREATE DATABASE, CREATE TABLE, CREATE

VIEW, CREATE PROCEDURE, CREATE FUNCTION, BACKUP DATABASE.

Câu lệnh GRANT sử dụng trong trường hợp này có cú pháp như sau:

GRANT ALL | danh_sách_câu_lệnh

TO danh_sách_người_dùng

Ví dụ 4:

Cho phép người dùng có tên nguoidung1 quyền tạo bảng và tạo view.

Câu lệnh:

GRANT CREATE TABLE,CREATE VIEW

TO nguoidung1

Chú ý:

+ Với câu lệnh GRANT, ta có thể cho phép người sử dụng tạo các đối tượng cơ sở dữ liệu trong cơ sở dữ liệu. Đối tượng cơ sở dữ liệu do người dùng nào tạo ra sẽ do người đó sở hữu và do đó người này có quyền cho người dùng khác sử dụng đối tượng và cũng có thể xóa bỏ (DROP) đối tượng do mình tạo ra.

+ Khác với trường hợp sử dụng câu lệnh GRANT để cấp phát quyền trên đối tượng cơ sở dữ liệu, câu lệnh GRANT trong trường hợp này không thể sử dụng tùy chọn WITH GRANT OPTION, tức là người dùng không thể chuyển tiếp được các quyền thực thi các câu lệnh đã được cấp phát.

b) Thu hồi quyền

Câu lệnh REVOKE được sử dụng để thu hồi quyền đã được cấp phát cho người dùng. Tương ứng với câu lệnh GRANT, câu lệnh REVOKE được sử dụng trong hai trường hợp:

- Thu hồi quyền đã cấp phát cho người dùng trên các đối tượng cơ sở dữ liệu.
- Thu hồi quyền thực thi các câu lệnh trên cơ sở dữ liệu đã cấp phát cho người dùng.

Khi ta sử dụng câu lệnh REVOKE để thu hồi quyền trên một đối tượng cơ sở dữ liệu từ một người dùng nào đó, chỉ những quyền mà ta đã cấp phát trước đó mới

được thu hồi, những quyền mà người dùng này được cho phép bởi những người dùng khác vẫn còn có hiệu lực.

Thu hồi quyền trên đối tượng cơ sở dữ liệu

Cú pháp:

REVOKE [GRANT OPTION FOR]

ALL [PRIVILEGES] | các_quyền_cần_thu_hồi
[(danh_sách_cột)] ON tên_bảng | tên_khung_nhìn
|ON tên_bảng | tên_khung_nhìn [(danh_sách_cột)]
|ON tên_thủ_tục
|ON tên_hàm
FROM danh_sách_người_dùng
[CASCADE]

Các tham số trong lệnh REVOKE tương tự như trong lệnh GRANT. Nếu ta đã cấp phát quyền cho người dùng nào đó bằng câu lệnh GRANT với tùy chọn WITH GRANT OPTION thì khi thu hồi quyền bằng câu lệnh REVOKE phải chỉ định tùy chọn CASCADE. Trong trường hợp này, các quyền được chuyển tiếp cho những người dùng khác cũng đồng thời được thu hồi.

Trong trường hợp cần thu hồi các quyền đã được chuyển tiếp và khả năng chuyển tiếp các quyền đối với những người đã được cấp phát quyền với tùy chọn WITH GRANT OPTION, trong câu lệnh REVOKE ta chỉ định mệnh đề GRANT OPTION FOR.

Ví dụ 5:

Thu hồi quyền INSERT trên bảng tbl_dmhang của nguoidung1.

Câu lệnh:

REVOKE INSERT ON tbl_dmhang
FROM nguoidung1

Thu hồi quyền thực thi các câu lệnh

Việc thu hồi quyền thực thi các câu lệnh trên cơ sở dữ liệu (CREATE DATABASE, CREATE TABLE, CREATE VIEW,...) được thực hiện đơn giản với câu lệnh REVOKE có cú pháp:

```
REVOKE ALL | các_câu_lệnh_cần_thu_hồi  
FROM danh_sách_người_dùng
```

Ví dụ 6:

Không cho phép người dùng nguoidung1 thực hiện lệnh CREATE TABLE trên cơ sở dữ liệu.

Câu lệnh:

```
REVOKE CREATE TABLE  
FROM nguoidung1
```

TÀI LIỆU THAM KHẢO

- MicroSoft SQL Server 2008 Book Online
- Philip J. Pratt, Joseph J. Adamski, Concepts of Database Management, Fifth edition, Course Technology, a division of Thomson Learning, 2005
- Website tài liệu của Microsoft <https://docs.microsoft.com/en-us/>