# 

**MỤC LỤC**

CHƯƠNG 1. [GIỚI THIỆU VỀ SQL SERVER 2](#_Toc343455552)

[1.1. Cài Ðặt SQL Server 2005 (Installation) 3](#_Toc343455553)

[1.2 Các Version của SQL Server 4](#_Toc343455554)

[1.3. Các thành phần quan trọng trong SQL Server 2005 5](#_Toc343455555)

[1.4 Cấu trúc của SQL SERVER 7](#_Toc343455556)

[CHƯƠNG 2](#_Toc343455560).[TỔNG QUAN VỀ TRANSACT SQL (T-SQL) 13](#_Toc343455561)

[2.1. Cú pháp của T- SQL 13](#_Toc343455562)

[2.2. Data Definition Language (DDL): 21](#_Toc343455563)

[2.3. Manipulation Language (DML): 27](#_Toc343455564)

[2.4 Variables (Biến) trong SQL Server 47](#_Toc343455566)

[2.5. Phát biểu điều khiển 48](#_Toc343455567)

[CHƯƠNG 3. BẢNG ẢO (VIEW) 52](#_Toc343455568)

[3.1. Khái niệm cơ bản về View 52](#_Toc343455569)

[3.2. Ưu điểm của View 52](#_Toc343455570)

[3.3. Tạo view. 53](#_Toc343455571)

[3.4. Các hạn chế của view 55](#_Toc343455572)

[3.5. Thay đổi, hủy và đổi tên các View 56](#_Toc343455573)

[3.6. Sử dụng View để cập nhật dữ liệu 57](#_Toc343455574)

[CHƯƠNG 4.](#_Toc343455575)[THỦ TỤC LƯU TRỮ (STORED PROCEDURE – SP) 59](#_Toc343455576)

[4.1. Khái niệm cơ bản về Stored Procedure 59](#_Toc343455577)

[4.2. Các ưu điểu của thủ tục lưu trữ 59](#_Toc343455578)

[4.3. Phân loại Store Procedure 59](#_Toc343455579)

[4.4. Tạo thủ tục lưu trữ 60](#_Toc343455580)

[4.5. Ví dụ. 61](#_Toc343455581)

[4.6. Thay đổi cấu trúc store procedure 62](#_Toc343455582)

[4.7. Xóa thủ tục 63](#_Toc343455583)

[CHƯƠNG 5. BẪY LỖI - TRIGGER 64](#_Toc343455584)

[5.1. Khái niệm cơ bản về Trigger. 64](#_Toc343455585)

[5.2. Ứng dụng của Trigger. 64](#_Toc343455586)

[5.3. Phân loại Trigger. 64](#_Toc343455587)

[5.4. Tạo và quản lý Trigger 65](#_Toc343455588)

[CHƯƠNG 6. KIỂU DỮ LIỆU CON TRỎ (CURSOR) 72](#_Toc343455589)

[6.1. Khái niệm cơ bản về CURSOR 72](#_Toc343455590)

[6.2. Các bước thực hiện một Cursor 72](#_Toc343455591)

[6.3. Loại Cursor 79](#_Toc343455592)

[6.4. Thay đổi dữ liệu tại vị trí Cursor 81](#_Toc343455593)

[CHƯƠNG 7](#_Toc343455594) [QUẢN LÝ BẢO MẬT VÀ NGƯỜI DÙNG 84](#_Toc343455595)

[7.1. Khái niệm cơ bản về bảo mật 84](#_Toc343455596)

[7.2 Các chế độ bảo mật 84](#_Toc343455597)

[7.3. Quản lý người dùng 85](#_Toc343455598)

[7.4. Quyền người dùng 87](#_Toc343455599)

CHƯƠNG 1

# GIỚI THIỆU VỀ SQL SERVER

Microsoft SQL Server 2005 là một hệ quản trị CSDL quan hệ (RDBMS), cung cấp cách tổ chức dữ liệu bằng cách lưu chúng vào các bảng. Dữ liệu quan hệ có thể được nhóm vào các bảng và các quan hệ có thể được định nghĩa giữa các bảng với nhau. Người dùng truy cập dữ liệu trên Serrver thông qua một ứng dụng. Người quản trị CSDL truy cập Server trực tiếp để thực hiện các chức năng cấu hình, quản trị và thực hiện các tác vụ bảo trì CSDL. Ngoài ra, SQL Server là một CSDL có khả năng mở rộng (chúng có thể lưu một lượng lớn dữ liệu và hỗ trợ tính năng cho phép nhiều người dùng truy nhập dữ liệu đồng thời)

SQL Server 2005 là một hệ thống quản lý cơ sở dữ liệu (Relational Database Management System (RDBMS)) sử dụng Transact-SQL để trao đổi dữ liệu giữa Client computer và SQL Server computer. 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 2005 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 2000 có thể kết hợp "ăn ý" với các server khác như Microsoft Internet Information Server (IIS), E-Commerce Server, Proxy Server....

SQL Server có 7 editions:

* **Enterprise** : Chứa đầy đủ các đặc trưng của SQL Server và có thể chạy tốt trên hệ thống lên đến 32 CPUs và 64 GB RAM. Thêm vào đó nó có các dịch vụ giúp cho việc phân tích dữ liệu rất hiệu quả (Analysis Services)
* **Standard** : Rất thích hợp cho các công ty vừa và nhỏ vì giá thành rẻ hơn nhiều so với Enterprise Edition, nhưng lại bị giới hạn một số chức năng cao cấp (advanced features) khác, edition này có thể chạy tốt trên hệ thống lên đến 4 CPU và 2 GB RAM.
* **Personal**: được tối ưu hóa để chạy trên PC nên có thể cài đặt trên hầu hết các phiên bản windows kể cả Windows 98.
* **Developer** : Có đầy đủ các tính năng của Enterprise Edition nhưng được chế tạo đặc biệt như giới hạn số lượng người kết nối vào Server cùng một lúc.... Ðây là edition mà các bạn muốn học SQL Server cần có. Chúng ta sẽ dùng edition này trong suốt khóa học. Edition này có thể cài trên Windows 2000 Professional hay Win NT Workstation.
* **Desktop Engine** (MSDE): Ðây chỉ là một engine chạy trên desktop và không có user interface (giao diện). Thích hợp cho việc triển khai ứng dụng ở máy client. Kích thước database bị giới hạn khoảng 2 GB.
* **Win CE** : Dùng cho các ứng dụng chạy trên Windows CE
* **Trial**: Có các tính năng của Enterprise Edition, download free, nhưng giới hạn thời gian sử dụng.

## 1.1. Cài Ðặt SQL Server 2005 (Installation)

Các bạn cần có **Developer Edition** và ít nhất là 64 MB RAM, 500 MB hard disk để có thể install SQL Server. Bạn có thể install trên Windows Server hay Windows XP Professional, Windows 2000 Professional hay NT Workstation nhưng không thể install trên Win 98 family.

Vì một trong những đặc điểm của các sản phẩm Microsoft là dễ install nên chúng tôi không trình bày chi tiết về cách install hay các bước install mà chỉ trình bày các điểm cần lưu ý khi install mà thôi. Nếu các bạn gặp trở ngại trong việc install thì có thể đưa lên forum để hỏi thêm. Khi install bạn cần lưu ý các điểm sau:

Ở màn hình thứ hai bạn chọn **Install Database Server**. Sau khi install xong SQL Server bạn có thể install thêm Analysis Service nếu bạn thích.

Ở màn hình **Installation Definition** bạn chọn **Server and Client Tools**.

Sau đó bạn nên chọn kiểu **Custom** và **chọn tất cả** các bộ phận của SQL Server. Ngoài ra nên **chọn các giá trị mặc định** (default)

Ở màn hình **Authentication Mode** nhớ chọn **Mixed Mode**. Lưu ý vì SQL Server có thể dùng chung chế độ bảo mật (security) với Win NT và cũng có thể dùng chế độ bảo mật riêng của nó.  Trong Production Server người ta thường dùng Windows Authetication vì độ an toàn cao hơn và dễ dàng cho người quản lý mạng và cả cho người sử dụng. Nghĩa là một khi bạn được chấp nhận (authenticated) kết nối vào domain thì bạn có quyền truy cập dữ liệu (access data) trong SQL Server. Tuy nhiên ta nên chọn Mixed Mode để dễ dàng cho việc học tập.

Sau khi Install bạn sẽ thấy một icon nằm ở góc phải bên dưới màn hình, đây chính là Service Manager. Bạn có thể Start, Stop các SQL Server services dễ dàng bằng cách double-click vào icon này.

## 1.2 Các Version của SQL Server

SQL Server của Microsoft được thị trường chấp nhận rộng rãi kể từ version 6.5. Sau đó Microsoft đã cải tiến và hầu như viết lại một engine mới cho SQL Server 7.0. Cho nên có thể nói từ version 6.5 lên version 7.0 là một bước nhảy vọt. Có một số đặc tính của SQL Server 7.0 không tương thích với version 6.5. Trong khi đó từ Version 7.0 lên version 8.0 (SQL Server 2000) thì những cải tiến chủ yếu là mở rộng các tính năng về web và làm cho SQL Server 2000 đáng tin cậy hơn. Hiện nay SQL Server đang ra mắt version 2005

Một điểm đặc biệt đáng lưu ý ở version 2005 là **Multiple-Instance**. Nói cho dễ hiểu là bạn có thể install version 2005 chung với các version trước mà không cần phải uninstall chúng. Nghĩa là bạn có thể chạy song song version 6.5 hoặc 7.0 với version 2000 trên cùng một máy (điều này không thể xảy ra với các version trước đây). Khi đó version cũ trên máy bạn là **Default Instance** còn version 2005 mới vừa install sẽ là **Named Instance**.

## 1.3. Các thành phần quan trọng trong SQL Server 2005

SQL Server 2005 được cấu tạo bởi nhiều thành phần như Relational Database Engine, Analysis Service và English Query.... Các thành phần này khi phối hợp với nhau tạo thành một giải pháp hoàn chỉnh giúp cho việc lưu trữ và phân tích dữ liệu một cách dễ dàng.

**1.3.1. Relational Database Engine** - Lõi của SQL Server:

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

**1.3.2. Replication** - Cơ chế tạo bản sao (Replica):

Giả sử bạn có một database dùng để chứa dữ liệu được các ứng dụng thường xuyên cập nhật. Một ngày đẹp trời bạn muốn có một cái database giống y hệt như thế trên một server khác để chạy báo cáo (report database) (cách làm này thường dùng để tránh ảnh hưởng đến performance của server chính). Vấn đề là report server của bạn cũng cần phải được cập nhật thường xuyên để đảm bảo tính chính xác của các báo cáo. Bạn không thể dùng cơ chế backup and restore trong trường hợp này. Thế thì bạn phải làm sao? Lúc đó cơ chế replication của SQL Server sẽ được sử dụng để bảo đảm cho dữ liệu ở 2 database được đồng bộ (synchronized). Replication sẽ được bàn kỹ trong bài 12

**1.3.3. Data Transformation Service (DTS)** - Một dịch vụ chuyển dịch data Nếu bạn làm việc trong một công ty lớn trong đó data được chứa trong nhiều nơi khác nhau và ở các dạng khác nhau cụ thể như chứa trong Oracle, DB2 (của IBM), SQL Server, Microsoft Access....Bạn chắc chắn sẽ có nhu cầu di chuyển data giữa các server này (migrate hay transfer) và không chỉ di chuyển bạn còn muốn định dạng (format) nó trước khi lưu vào database khác, khi đó bạn sẽ thấy DTS giúp bạn giải quyết công việc trên dễ dàng như thế nào. DTS sẽ được bàn kỹ trong bài 8.

**1.3.4. Analysis Service -** Một dịch vụ phân tích dữ liệu

Dữ liệu (Data) chứa trong database sẽ chẳng có ý nghĩa gì nhiều nếu như bạn không thể lấy được những thông tin (Information) bổ ích từ đó. Do đó Microsoft cung cấp cho bạn một công cụ rất mạnh giúp cho việc phân tích dữ liệu trở nên dễ dàng và hiệu quả bằng cách dùng khái niệm hình khối nhiều chiều (multi-dimension cubes) và kỹ thuật "đào mỏ dữ liệu" (data mining) sẽ được chúng tôi giới thiệu trong bài 13.

**1.3.5. English Query:** Ðây là một dịch vụ giúp cho việc query data bằng tiếng Anh "trơn" (plain English).

**1.3.6. Meta Data Service:**

Dịch vụ này giúp cho việc chứa đựng và "xào nấu" Meta data dễ dàng hơn. Thế thì Meta Data là cái gì vậy? Meta data là những thông tin mô tả về cấu trúc của data trong database như data thuộc loại nào String hay Integer..., một cột nào đó có phải là Primary key hay không....Bởi vì những thông tin này cũng được chứa trong database nên cũng là một dạng data nhưng để phân biệt với data "chính thống" người ta gọi nó là Meta Data. Phần này chắc là bạn phải xem thêm trong một thành phần khác của SQL Server sắp giới thiệu sau đây là **SQL Server Books Online** vì không có bài nào trong loạt bài này nói rõ về dịch vụ này cả.

**1.3.7. SQL Server Books Online**

Cho dù bạn có đọc các sách khác nhau dạy về SQL server thì bạn cũng sẽ thấy books online này rất hữu dụng và không thể thiếu được (cho nên Microsoft mới hào phóng đính kèm theo SQL Server).

* + 1. **SQL Server Tools**
* **Enterprise Manager:** Đây là một công cụ cho ta thấy toàn cảnh hệ thống CSDL một cách rất trực quan. Nó rất hữu ích đặc biệt cho người mới học và không thông thạo lắm về SQL.
* **Query Analyzer:** Đối với một DBA giỏi thì hầu như chỉ cần công cụ này là có thể quản lý cả một hệ thống database mà không cần đến những thứ khác. Đây là một môi trường làm việc khá tốt vì ta có thể đánh bất kỳ câu lệnh SQL nào và chạy ngay lập tức đặc biệt là nó giúp cho chúng ta debug mấy cái Stored procedure dễ dàng.
* **SQL Profiler:** Nó có khả năng chụp tất cả các sự kiện hay hoạt động diễn ra trên một SQL Server và lưu lại dưới dạng text file rất hữu dụng trong việc kiểm soát hoạt động của SQL Server.
* Ngoài một số công cụ trực quan như trên chúng ta cũng thường hay dùng osql và bcp (bulk copy) trong command prompt.

## 1.4 Cấu trúc của SQL SERVER

**1.4.1 Các cơ sở dữ liệu hệ thống của SQL SERVER**

Đặc điểm của SQL Server 2005 là **Multiple-Instance** nên khi nói đến một (SQL) Server nào đó là ta nói đến một Instance của SQL Server 2000, thông thường đó là Default Instance. Một Instance của SQL Server 2000 có 4 system databases và một hay nhiều user database. Các system databases bao gồm:

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

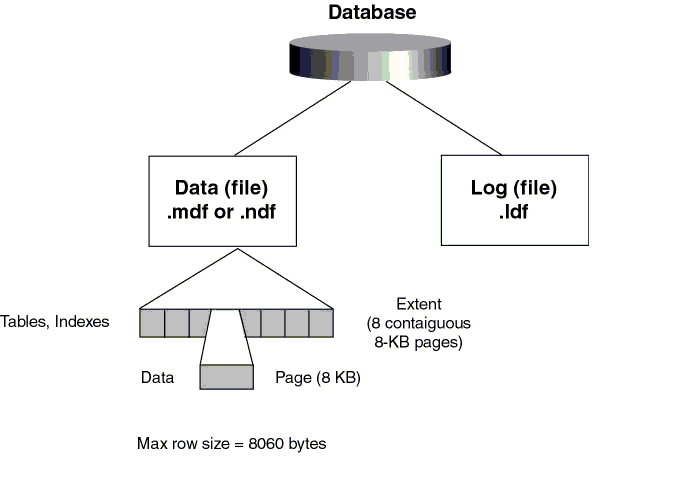
## 1.4.2 Cấu trúc vật lý của một SQL Server Database

Mỗi một database trong SQL Server được lưu trữ dưới dạng 2 tập tin:

* Tập tin dữ liệu (data file): Lưu trữ dữ liệu gồm 2 loại:
* **Primary data file** (tên file có phần mở rộng là **.mdf**): đây là file chính chứa data và những system tables.
* **Secondary data file** (tên file có phần mở rộng là **.ndf**): đây là file phụ thường chỉ sử dụng khi database được phân chia để chứa trên nhiều dĩa.
* Tập tin Log (Transaction log file – tên file có phần mở rộng là .ldf) : đây là file ghi lại tất cả những thay đổi diễn ra trong một database và chứa đầy đủ thông tin để có thể roll back hay roll forward khi cần.

Ghi chú; Mỗi một database trong SQL Server có ít nhất một tập tin dữ liệu và một tập tin log:

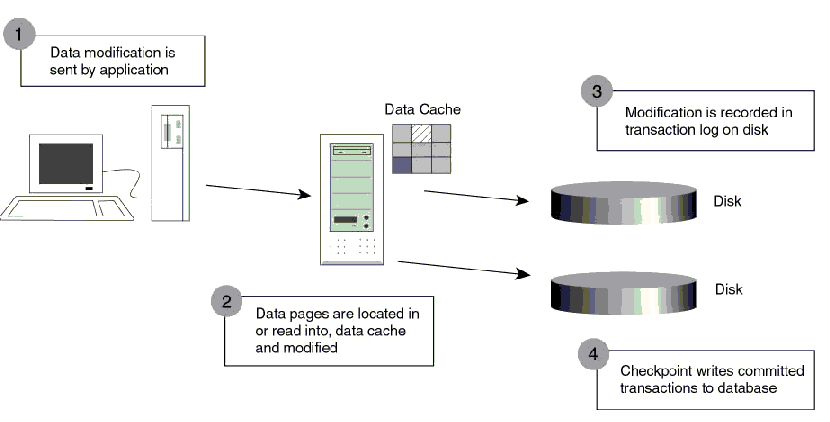
Data trong SQL Server được chứa thành từng **Page** 8KB và 8page liên tục tạo thành một **Extent** như hình vẽ dưới đây:



Trước khi SQL Server muốn lưu data vào một table nó cần phải dành riêng một khoảng trống trong data file cho table đó. Những khoảng trống đó chính là các extents. Có 2 loại Extents: **Mixed Extents** (loại hỗn hợp) dùng để chứa data của nhiều tables trong cùng một Extent và **Uniform Extent** (loại thuần nhất) dùng để chứa data của một table. Ðầu tiên SQL Server dành các Page trong Mixed Extent để chứa data cho một table sau đó khi data tăng trưởng thì SQL dành hẳn một Uniform Extent cho table đó.

## 1.4.3. Nguyên tắc hoạt động của Transaction Log Trong SQL Server

Transaction log file trong SQL Server dùng để ghi lại các thay đổi xảy ra trong database. Quá trình này diễn ra như sau: đầu tiên khi có một sự thay đổi data như Insert, Update, Delete được yêu cầu từ các ứng dụng, SQL Server sẽ tải (load) data page tương ứng lên memory (vùng bộ nhớ này gọi là data cache), sau đó data trong data cache được thay đổi(những trang bị thay đổi còn gọi là *dirty-page*). Tiếp theo mọi sự thay đổi đều được ghi vào transaction log file cho nên người ta gọi là *write-ahead* log. Cuối cùng thì một quá trình gọi là **Check Point Process** sẽ kiểm tra và viết tất cả những transaction đã được commited (hoàn tất) vào dĩa cứng (flushing the page).



Ngoài Check Point Process những dirty-page còn được đưa vào dĩa bởi một **Lazy writer**. Ðây là một anh chàng làm việc âm thầm chỉ thức giấc và quét qua phần data cache theo một chu kỳ nhất định sau đó lại ngủ yên chờ lần quét tới.

Xin giải thích thêm một chút về khái niệm transaction trong database. Một transaction hay một giao dịch là một loạt các hoạt động xảy ra được xem như một công việc đơn (unit of work) nghĩa là hoặc thành công toàn bộ hoặc không làm gì cả (all or nothing). Sau đây là một ví dụ cổ điển về transaction:

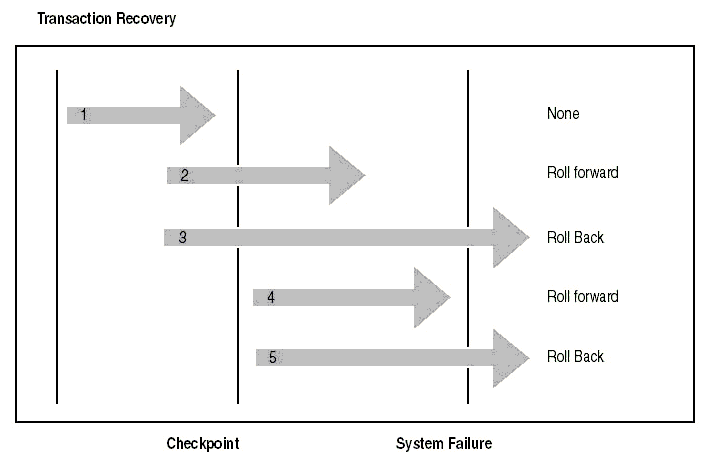
Chúng ta muốn chuyển một số tiền $500 từ account A sang

account B như vậy công việc này cần làm các bước sau:

1. Trừ $500 từ account A
2. Cộng $500 vào account B

Tuy nhiên việc chuyển tiền trên phải được thực hiện dưới dạng một transaction nghĩa là giao dịch chỉ được xem là hoàn tất (commited) khi cả hai bước trên đều thực hiện thành công. Nếu vì một lý do nào đó ta chỉ có thể thực hiện được bước 1 (chẳng hạn như vừa xong bước 1 thì điện cúp hay máy bị treo) thì xem như giao dịch không hoàn tất và cần phải được phục hồi lại trạng thái ban đầu (roll back).

Thế thì Check Point Process hoạt động như thế nào để có thể đảm bảo một transaction được thực thi mà không làm "dơ" database.



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

Ðiều đó có nghĩa là SQL Server sẽ không cần làm gì cả đối với transaction 1 vì tại thời điểm Check point data đã được lưu vào dĩa rồi. Trong khi đó transaction 2 và 4 sẽ được roll forward vì tuy đã được commited nhưng do sự cố xảy ra trước thời điểm check point kế tiếp nên data chưa kịp lưu vào dĩa. Tức là dựa trên những thông tin được ghi trên log file SQL Server hoàn toàn có đầy đủ cơ sở để viết vào dĩa cứng. Còn transaction 3 và 5 thì chưa được commited (do bị down bất ngờ) cho nên SQL Server sẽ roll back hai transaction này dựa trên những gì được ghi trên log file.

## 1.4.4 Cấu Trúc Logic Của Một SQL Server Database

Mỗi Database trong SQL Server đều được tổ chức thành những objects như tables, views, stored procedures, indexes, constraints.... Những system objects trong SQL Server thường có bắt đầu bằng chữ *sys* hay *sp*. Các objects trên sẽ được nghiên cứu lần lượt trong các chương tiếp theo của giáo trình

# CHƯƠNG 2

# TỔNG QUAN VỀ TRANSACT SQL (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 SQL Server khác với P-SQL (Procedural-SQL) dùng trong Oracle.

## 2.1. Cú pháp của T- SQL

**2.1.1. Identifiers (định danh)**

Ðây chính là tên của các database object. Nó dùng để xác định một object. (Chú ý khi nói đến Object trong SQL Server là chúng ta muốn đề cập đến table, view, stored procedure, index, …Vì hầu như mọi thứ trong SQL Server đều được thiết kế theo kiểu hướng đối tượng (object-oriented)). Trong ví dụ sau TableX, KeyCol, Description là những identifiers

CREATE TABLE TableX

(KeyCol INT PRIMARY KEY, Description NVARCHAR (80))

Có hai loại Identifiers một loại thông thường (**Regular Identifier**) và một loại gọi là **Delimited Identifier**, loại này cần có dấu "" hay dấu [] để ngăn cách. Loại Delimited được dùng đối với các chữ trùng với từ khóa của SQL Server (reserved keyword) hay các chữ có khoảng trống.

Ví dụ:

SELECT \* FROM [My Table]

WHERE [Order] = 10

Trong ví dụ trên chữ Order trùng với keyword Order nên cần đặt trong dấu ngoặc vuông [].

Ðây chính là tên của các database object. Nó dùng để xác định một object. (Chú ý khi nói đến Object trong SQL Server là chúng ta muốn đề cập đến table, view, stored procedure, index, ...Vì hầu như mọi thứ trong SQL Server đều được thiết kế theo kiểu hướng đối tượng (object-oriented)). Trong ví dụ sau TableX, KeyCol, Description là những identifiers

CREATE TABLE TableX

(KeyCol INT PRIMARY KEY, Description NVARCHAR (80))

**2.1.2. Biểu thức**

Các biểu thức trong T-SQL thường có dạng: Identifier + Operators (như +, -, \*, /, =…) + value

**2.1.3. Comments (Chú thích)**

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

**2.1.4. Lệnh USE**

Dùng thay đổi ngữ cảnh CSDL để chỉ định CSDL mà các câu lệnh tiếp theo sẽ có tác dụng trên đó (khai báo rằng ta sẽ dùng CSDL nào cho các câu lệnh phía sau nó)

USE <database\_name>,

Trong đó <database\_name>: Là tên của CSDL.

**2.1.5. Các kiểu dữ liệu**

|  |  |  |
| --- | --- | --- |
| Tên loại dữ liệu | Độ lớn | Được sử dụng để lưu trữ |
| [bigint](ts_ia-iz_3ss4.htm) | From -2^63 (-9223372036854775808) through 2^63-1 (9223372036854775807) | Integer (whole number) data. Dùng 8 byte |
| [int](ts_ia-iz_3ss4.htm) | From -2^31 (-2,147,483,648) through 2^31 - 1 (2,147,483,647) | Integer (whole number) data. Dùng 4 byte |
| [smallint](ts_ia-iz_3ss4.htm) | Integer data from 2^15 (-32,768) through 2^15 - 1 (32,767) | Integer data. Dùng 2 byte |
| [tinyint](ts_ia-iz_3ss4.htm) | Integer data from 0 through 255 | Integer data (1 byte) |
| [bit](ts_ba-bz_2it0.htm) | Integer data with either a 1 or 0 value. | Integer data |
| [decimal](ts_de-dz_3grn.htm) | Fixed precision and scale numeric data from -10^38 +1 through 10^38 –1 |  |
| [numeric](ts_de-dz_3grn.htm) | Functionally equivalent to decimal. |  |
| [money](ts_ma-mz_49q1.htm) | Monetary data values from -2^63 (-922,337,203,685,477.5808) through 2^63 - 1 (+922,337,203,685,477.5807), with accuracy to a ten-thousandth of a monetary unit | Monetary data |
| [smallmoney](ts_ma-mz_49q1.htm) | Monetary data values from -214,748.3648 through +214,748.3647, with accuracy to a ten-thousandth of a monetary unit | Monetary data |
| [float](ts_fa-fz_6r3g.htm) | Floating precision number data from -1.79E + 308 through 1.79E + 308 | Floating precision data |
| [real](ts_fa-fz_6r3g.htm) | Floating precision number data from -3.40E + 38 through 3.40E + 38 | Floating precision data |
| [datetime](ts_da-db_9xut.htm) | Date and time data from January 1, 1753, through December 31, 9999, with an accuracy of three-hundredths of a second, or 3.33 milliseconds | Date and time data |
| [smalldatetime](ts_da-db_9xut.htm) | Date and time data from January 1, 1900, through June 6, 2079, with an accuracy of one minute | Date and time data |
| [char](ts_ca-co_7tpu.htm) | Fixed-length non-Unicode character data with a maximum length of 8,000 characters | Fixed-length non-Unicode character data |
| [varchar](ts_ca-co_7tpu.htm) | Variable-length non-Unicode data with a maximum of 8,000 characters | Variable-length non-Unicode character data |
| [text](ts_ta-tz_2838.htm) | Variable-length non-Unicode data with a maximum length of 2^31 - 1 (2,147,483,647) characters | Variable-length non-Unicode data |
| [nchar](ts_na-nop_9msy.htm) | Fixed-length Unicode data with a maximum length of 4,000 characters | Fixed-length Unicode data |
| [nvarchar](ts_na-nop_9msy.htm) | Variable-length Unicode data with a maximum length of 4,000 characters. **sysname** is a system-supplied user-defined data type that is functionally equivalent to **nvarchar(128)** and is used to reference database object names | Variable-length Unicode data |
| [ntext](ts_nos-nz_0lyd.htm) | Variable-length Unicode data with a maximum length of 2^30 - 1 (1,073,741,823) characters | Variable-length Unicode data |
| [binary](ts_ba-bz_75yx.htm) | Fixed-length binary data with a maximum length of 8,000 bytes | Fixed-length binary data |
| [varbinary](ts_ba-bz_75yx.htm) | Variable-length binary data with a maximum length of 8,000 bytes | Variable-length binary data |
| [image](ts_ia-iz_9rfp.htm) | Variable-length binary data with a maximum length of 2^31 - 1 (2,147,483,647) bytes | Variable-length binary data |
| [cursor](ts_cr-cz_414i.htm) | A reference to a cursor |  |

**2.1.6. Một số hàm của T-SQL**

**a) Các hàm về xâu ký tự**

|  |  |  |
| --- | --- | --- |
| Tên hàm | Ví dụ | Miêu tả |
| ASCII | SELECT ASCII(‘ABC’) | Trả về 65, là mã ASCII của ‘A’ |
| CHAR | SELECT CHAR (65) | Trả về ‘A’. |
| CHARINDEX | SELECT CHARINDEX(‘E’,’HELLO’) | Trả về 2 |
| DIFFRENCE | SELECT  DIFFERENCE(‘HELLO’,’HELL’) | Trả về 4. Giá trị 4 là có độ tương tự cao nhất (0 -> 4) |
| LEFT | SELECT LEFT(‘RICHARD’,4) | Trả lại ‘RICH’ |
| LEN | SELECT LEN(‘RICHARD’) | Trả lại 7 |
| LOWER | SELECT LOWER(‘RICHARD’) | Trả lại ‘richard’, bao gồm toán chữ thường |
| LTRIM | SELECT  LTRIM (‘RICHARD’) | Trả lại ‘RICHARD’, bỏ đi các ký tự trống bên trái |
| PATINDEX | SELECT  PATINDEX (‘%BOX%, ‘ACTION BOX’) | Trả lại 7, vị trí xuất hiện đầu tiên. |
| REVERSE | SELECT  REVERSE(‘ACTION’) | Trả lại ‘NOTICA’, đảo ngược của ‘ACTION’ |
| RIGHT | SELECT  RIGHT (‘RICHARD’, 4) | Trả về ‘HARD’ |
| RTRIM | SELECT  RTRIM (‘RICHARD ’) | RTRIM trả về ‘RICHARD’ (sau khi đã xóa bỏ các ký tự trống bên phải) |
| SPACE | SELECT  ‘RICHARD’ + SPACE (2) + ‘HILL’ | Trả về ‘RICHARD HILL’ |
| SUBSTRING | SELECT  SUBSTRING(‘weather’,2,2) | Trả về ‘EA’ |
| UPPER | SELECT UPPER(‘Richard’) | Trả về RICHARD, mọi ký tự ở dạng chữ hoa |
| STUFF | SELECT  STUFF(‘Weather,2,3,’i’) | Xóa 3 ký tự từ vị trí thứ 2, sau đó thêm vào ký tự i tại vị trí xóa.  Kết quả thu được là ‘Wiher’ |

**b) Các hàm ngày tháng**

|  |  |  |
| --- | --- | --- |
| Hàm | Tham số | Miêu tả |
| [DATEADD](ts_da-db_3vtw.htm) | (datepart,number,date) | Cộng thêm datepart vào date |
| [DATEDIFF](ts_da-db_5vxi.htm) | (datepart,date1,date2) | Tính toán độ sai lêch về datepart giữa date1 và date2 |
| [DATENAME](ts_da-db_1dph.htm) | (datepart,date) | Trả lại tên datepart, ví dụ October |
| [DATEPART](ts_da-db_2mic.htm) | (datepart,date) | Trả lại datepart dưới giá trị số |
| [GETDATE](ts_ga-gz_4z51.htm) | () | Trả lại ngày giờ hiện tại |

Trong đó

|  |  |
| --- | --- |
| Datepart | Abbreviations |
| Year | yy, yyyy |
| Quarter | qq, q |
| Month | mm, m |
| dayofyear | dy, y |
| Day | dd, d |
| Week | wk, ww |
| weekday | dw |
| Hour | hh |
| Minute | mi, n |
| Second | ss, s |
| millisecond | ms |

Ta có các ví dụ sau:

SELECT GETDATE()

Trả lại ngay giờ hiện tại

SELECT DATEDIF(yy,Ord\_date, GETDATE())

FROM Sales:

Tính độ lệch về năm giữa Ord\_date và ngày hiện tại của table Sales.

**c) Các hàm số học**

|  |  |
| --- | --- |
| Tên hàm | Miêu tả |
| ABS | Giá trị tuyệt đối |
| ACOS, ASIN, ATAN | Trả lại giá trị radian của các giá trị cos, sin, tan tương ứng |
| COS, SIN, COT, TAN | Tính các giá trị cos, sin, tan, cotan tương ứng |
| DEGREES | Trả về số nguyên nhỏ nhất lớn hơn hoặc bằng |
| EXP | Hàm mũ cơ số e |
| FLOOR | Trả về số nguyên lớn nhất nhỏ hơn hoặc bằng |
| LOG | Hàm logarit cơ số e. |
| LOG10 | Logarit cơ số 10 |
| PI | Trả về số PI (3.141592653) |
| POWER | Tính hàm mũ dạng a^b với a,b là tham số đưa vào |
| RADIANS | Chuyển đổi từ độ sang radian |
| RAND | Trả về một số thực trong khoảng [0,1] |
| ROUND | Hàm làm tròn, sẽ được nói rõ sau |
| SIGN | Trả về dấu, có thể là dương, âm hoặc bằng không |
| SQRT | Căn bậc 2 |

Ví dụ về hàm ROUND(làm tròn)

SELECT Score

= ROUND(79.9,0)

Kết quả trả về là 80.

Chi tiết về hàm ROUND như sau:

|  |  |
| --- | --- |
| Hàm | Kết quả |
| ROUND(1234.567,2) | 1234.570 |
| ROUND(1234.567,1) | 1234.600 |
| ROUND(1234.567,0) | 1235.000 |
| ROUND(1234.567,-1) | 1230.000 |
| ROUND(1234.567,-2) | 1200.000 |
| ROUND(1234.567,32) | 1000.000 |

## 2.2. Data Definition Language (DDL):

Ðây là những lệnh dùng để quản lý các thuộc tính của một database như định nghĩa các hàng hoặc cột của một table, hay vị trí data file của một database...thường có dạng sau:

* Create *object\_Name*
* Alter *object\_Name*
* Drop *object\_Name*

Trong đó *object\_Name* có thể là một table, view, stored procedure, indexes...

**2.2.1. Lệnh tạo cơ sở dữ liệu – Create database**

Khi tạo một cơ sở dữ liệu, một không gian cơ sở dữ liệu cũng tự động được tạo ra và các bảngdữ liệu được đặt trong không gian cơ sở dữ liệu này.

**a) Cách 1: Dùng Query Analyzer**

CREATE DATABASE database\_name   
ON [ PRIMARY ]  
 ( NAME = <logical\_file\_name> ,   
     FILENAME = <'os\_file\_name'>   
     [ , SIZE = <size> ]   
     [ , MAXSIZE = <max\_size> ]   
     [ , FILEGROWTH = <growth\_increment> ] ) [ ,...n ]

LOG ON

( NAME = <logical\_file\_name> ,   
     FILENAME = <'os\_file\_name'>   
     [ , SIZE = <size> ]   
     [ , MAXSIZE = < max\_size >]   
     [ , FILEGROWTH = <growth\_increment> ] ) [ ,...n ]

Trong đó:

Database\_name: tên CSDL

Logical\_file\_name:Là tên file logic mà các câu lệnh T-SQL khi được thực hiện tham chiếu đến.

Os\_file\_name:đường dẫn thực đến file CSDL

Size: kích thước ban đầu của file CSDL

Max\_size:kích thước tối đa của file CSDL

Growth\_incremen: mức độ tăng trưởng dung lượng file dữ liệu mỗi khi cần nới rộng file CSDL (ngầm định là MB)

Chú ý:Có thể tạo CSDL bằng lệnh Create Database sau: (khi đó các thông số sẽ được chọn ngầm định)

Create Database <database\_name>

b) Cách 2: Dùng Enterprise Manager

Nháy chuột phải vào Database/New Database

Đặt tên cho cơ sở dữ liệu

Ví dụ:

Lệnh Create sau sẽ tạo ra một table tên Importers với 3 cột CompanyID,CompanyName,Contact

USE Northwind

CREATE TABLE Importers(

CompanyID int NOT NULL,

CompanyName varchar(40) NOT NULL,

Contact varchar(40) NOT NULL

)

**2.2.2. Lệnh tạo bảng – CREATE TABLE**

**a) Cú pháp**

CREATE TABLE <Tên\_bảng>  
    **(**  < Tên\_cột\_1> <Kiểu\_dữ\_liệu> [ràng\_buộc\_cột],

< Tên\_cột \_2> <Kiểu\_dữ\_liệu> [ràng\_buộc\_cột],

………………………

,< Tên\_cột \_n> <Kiểu\_dữ\_liệu> [ràng\_buộc\_cột]

[ < Ràng\_buộc\_bảng >] **)**

**Trong đó:**

+ Tên\_bảng, Tên\_cột: Do người sử dụng tự định đặt (phải tuân theo quy tắc đặt tên)

+ Kiểu\_dữ\_liệu: Chọn một kiểu dữ liệu sao cho phù hợp nhất với dữ liệu người dùng sẽ nhập vào

+ ràng\_buộc\_cột: Là một số các ràng buộc về khóa, ràng buộc mặc định, hoặc các quy định về khuôn dạng dữ liệu

* Null: Dữ liệu giá trị tại cột này có thể nhận giá trị rỗng
* Not Null: Bắt buộc phải nhập giá trị cho cột này
* Unique: giá trị nhập vào cột phải duy nhất
* Ràng buộc khóa chính: Primary key
* Ràng buộc khóa ngoài: Foreign Key References <Tên\_bảng\_tham\_ chiếu>(Tên\_cột\_tham\_chiếu).
* Ràng buộc mặc định: Default <giá trị mặc định>, dùng để quy định giá trị mặc định cho một cột. Giá trị này sẽ tự động được gán cho cột khi người sử dụng bổ sung một bản ghi mà không chỉ định giá trị cho cột (trên mỗi cột chỉ có tối đa một ràng buộc mặc định)
* Ràng buộc kiểm tra: CHECK **(***Biểu\_thức\_logíc* **)**

**+ Ràng\_buộc\_bảng**

* Ràng buộc khóa chính: Giá trị dùng để xác định duy nhất một đối tượng nên giá trị của chúng phải duy nhất, không thể nhận giá trị Null. Để khai báo ràng buộc khóa chính ta sử dụng cú pháp:

CONSTRAINT <Tên\_ràng\_buộc> PRIMARY KEY (<Ds\_thuộc\_tính\_khóa >)

* Ràng buộc khóa ngoài: dùng để kiểm tra giá trị của cột có phù hợp với cột trong bảng khác quan hệ với bảng hiện có cột ràng buộc hay không. Để khai báo một ràng buộc khóa ngoài ta dùng cú pháp sau:

CONSTRAINT <Tên\_ràng\_buộc> FOREIGN KEY (Tên\_thuộc\_tính\_khóa\_ngoài ) REFERENCES <Tên\_bảng\_liên\_kết> ( Tên\_thuộc\_tính\_liên\_kết ).

* Ràng buộc Check: Dùng để chỉ định các giá trị hay khuôn dạng dữ liệu có thể được chấp nhận đối với một cột. Trên một cột có thể sử dụng nhiều ràng buộc CHECK. Để khai báo một ràng buộc CHECK đối với một cột nào đó ta sử dụng cú pháp sau:

CONSTRAINT <Tên\_ràng\_buộc> CHECK **(** *Biểu\_thức\_logíc* **) ,** trong đó Biểu\_thức\_logic quy định giá trị hay khuôn dạng của dữ liệu. Khi đó, chỉ những giá trị dữ liệu nào làm cho biểu thức logíc nhận giá trị đúng mới được chấp nhận.

**b) Identity:** Khi bạn muốn một cột có giá trị tăng tự động như AutoNumber nên định nghĩa cột đó như IDENTITY [ **(** *seed* **,** *increment* **)** ].

Trong đó Seed là giá trị khởi đầu khi QSL tự động tăng giá trị. Increment là bước tăng cho biết mỗi lần tăng cần bao nhiêu giá trị

Ví dụ: Create table MonHoc(MaM tinyint Identity(1,1) Primary key, TenMon VarChar(30))

***Chú ý:*** Identity chỉ áp dụng với kiểu số nguyên.

**c) Cột có khả năng tính toán - Cột ảo (Trường ảo):**

Khi khai báo cột có thuộc tính Computed ta không cần không gian đĩa để lưu trữ dữ liệu cho cột đó, giá trị của chúng chính là sự kết hợp dữ liệu từ nhiều cột khác

Cú pháp: <Tên\_cột > AS <Biểu\_thức>

**2.2.3 Lệnh ALTER**

Lệnh Alter sau đây cho phép ta thay đổi một phần cấu trúc của các đối tượng table, view, ….

Cú pháp :

Alter <Object type> <Object namre>

**2.2.3.1 ALTER DATABASE:**

Dùng để chỉnh sửa các thông số của cơ sở dữ liệu

Cú pháp:

ALTER DATABASE database   
ADD FILE < filespec > [ ,...n ]   
| ADD LOG FILE < filespec > [ ,...n ]   
| REMOVE FILE logical\_file\_name   
| MODIFY FILE < filespec >   
| MODIFY NAME = new\_dbname

Filespec::=(NAME = logical\_file\_name   
    [ , NEWNAME = new\_logical\_name ]   
    [ , FILENAME = 'os\_file\_name' ]   
    [ , SIZE = size ]   
    [ , MAXSIZE = max\_size ]   
    [ , FILEGROWTH = growth\_increment ] )

**2.2.3.2. Sửa đối cấu trúc bảng – ALTER TABLE**

Sau khi đã tạo bảng, ta có thể tiến hành sửa đổi cấu trúc hay thuộc tính của bảng như bổ sung cột, bổ sung khóa, thay đổi các ràng buộc,… Để có thể sửa đổi cấu trúc bảng ta sử dụng lệnh ALTER với cú pháp sau:

**+ Thêm cột mới**

ALTER TABLE <table\_name>

ADD <column\_name> <data\_type> [column\_contraint], [… n]

**+ Thêm ràng buộc bảng**

ALTER TABLE <table\_name>

ADD <table\_contraint>

**+ Xóa cột**

ALTER TABLE <table\_name>

DROP <column\_name>[,…n]

**+ Xóa ràng buộc bảng**

ALTER TABLE <table\_name>

DROP CONSTRAINT<Contraint\_name>[…,n]

**2.2.4. Lệnh xóa – DROP**

Lệnh Drop dùng để xóa đối tượng của cơ sở dữ liệu

DROP <Oject\_type > <Oject\_namre>[,..n]

Oject\_type bao gồm các đối tượng: Database, table, view, trigger,…

## 2.3. Manipulation Language (DML):

Ðây là những lệnh phổ biến dùng để xử lý data như Select, Update, Insert, Delete

**2.3.1. Lệnh truy vấn dữ liệu - SELECT**

Việc truy cập và lấy các thông tin từ database được SQL cho phép thực hiện qua câu lệnh SELECT. Câu lệnh SELECt có phạm vi ứng dụng rất rộng, có thể truy cập dữ liệu từ một table, hay từ nhiều table

Các từ khóa SELECT, FROM, WHERE được sử dụng để tạo nên một câu lệnh SELECT đơn giản nhất

Cú pháp tổng quát có dạng sau:

SELECT [ ALL | DISTINCT ] <danh\_sách\_chọn>  
FROM <danh\_sách\_bảng>  
[ WHERE <điều\_kiện >]   
[ GROUP BY <danh\_sách\_cột>]   
[ HAVING <điều\_kiện>]   
[ ORDER BY <tên\_cột> [ ASC | DESC ] [,..n]]

**a) Mệnh đề FROM**

Mệnh đề FROM trong câu SELECT được sử dụng nhằm chỉ định các bảng cần truy xuất dữ liệu. Sau FROM là danh sách tên của các bảng và khung nhìn tham gia vào truy vấn. Tên của các bảng và các khung nhìn được phân cách nhau bởi dấu phẩy.

Ví dụ: Câu lệnh dưới đây hiển thị danh sách các công ty.

SELECT MaCongTy, TenCongTy, DiaChi

FROM CongTy

***Chú ý:*** Ta có thể sử dụng các bí danh cho các bảng hay khung nhìn trong câu lệnh SELECT. Bí danh được gán trong mệnh đề FROM bằng cách chỉ định bí danh ngay sau tên bảng.

Ví dụ: Câu lệnh sau gán bí danh là CC cho bảng CongTy

SELECT \* FROM CongTy CT

**b) Mênh đề SELECT**

Danh sách chọn trong câu lệnh SELECT được sử dụng để chỉ định các trường các biểu thức cần hiển thị trong các cột của kết quả truy vấn. Các trường các biểu thức được chỉ định ngay sau từ khoá SELECT và phân cách nhau bời dấu phẩy. Sử dụng danh sách chọn trong câu lệnh SELECT bao gồm các trường hợp sau:

**b.1 Chọn tất cả các cột trong bảng**

Khi cần hiển thị tất cả các trường trong bảng, sử dụng ký tự \* trong danh sách chọn thay vì phải liệt kê danh sách tất cả các cột. Trong trường hợp này, các cột được hiển thị trong kết quả truy vấn sẽ tuân thủ theo thứ tự mà chúng đã được tạo ra khi bảng được định nghĩa.

Ví dụ: Câu lệnh dùng để liệt kê danh sách các mặt hàng

SELECT \* FROM HangHoa

**b.2 Liệt kê tên cột trong danh sách chọn**

Trong trường hợp cần chỉ định cụ thể các cột cần hiển thị trong kết quả truy vấn, ta chỉ định danh sách các tên cột trong danh sách chọn. Thứ tự của các cột trong kết quả truy vấn tuân theo thứ tự của các trường trong danh sách chọn.

Ví dụ: Liệt kê danh sách các mặt hàng gồm các thuộc tính sau: MaHang, TenHang, DonViTinh

SELECT MaHang, TenHang, DonViTinh

FROM HangHoa

***Chú ý:*** Nếu truy vấn được thực hiện trên nhiều bảng và các bảng có các trường trùng tên thì tên của những trường này nếu xuất hiện trong danh sách chọn phải được viết dưới dạng:

<Tên\_bảng>.<Tên\_trường>

Ví dụ: Liệt kê danh sách các công ty đã cung cấp hàng hoá

SELECT CC.MaCT, TenCongTy, DiaChi

FROM CongTy CT, CungCap CC

WHERE CT.MaCT=CC.MaCT

**b.3 Thay đổi tiêu đề các cột**

Trong kết quả truy vấn, tiêu đề của các cột mặc định sẽ là tên của các trường tương ứng trong bảng. Tuy nhiên, để tiêu đề trở thành thân thiện hơn, ta có thể đổi lại tên tiêu đề của các cột. Để đặt tiêu đề cho một cột nào đó, ta sử dụng cách viết:

<tên\_trường > AS <tiêu\_đề\_cột>

Ví dụ: Cho biết mã và tên của các công ty.

SELECT MaCongTy, TenCongTy AS Tên\_Công\_Ty

FROM CongTy

**b.4 Hằng và biểu thức trong danh sách chọn**

Ngoài danh sách trường, trong danh sách chọn của câu lệnh SELECT còn có thể sử dụng các biểu thức. Mỗi biểu thức trong danh sách chọn trở thành một cột trong kết quả truy vấn.

Ví dụ: Câu lệnh dưới đây cho biết mã công ty, mã hàng, số lượng, đơn giá và thành tiền của các mặt hàng đã được các công ty cung cấp.

SELECT MaCongTy, MaHang, SoLuong, DonGia, SoLuong\*DonGia AS ThanhTien

FROM CungCap

**b.4 Loại bỏ các bản ghi trùng nhau trong kết quả truy vấn**

Trong kết quả của truy vấn có thể xuất hiện các dòng dữ liệu trùng nhau. Để loại bớt các dòng này, ta chỉ định thêm từ khoá DISTINCT ngay sau từ khoá SELECT.

Ví dụ: Cho biết thông tin về mã của từng loại mặt hàng đã được cung cấp.

SELECT DISTINCT MaHang FROM CungCap

**b.5 Giới hạn số lượng bản ghi trong kết quả truy vấn**

Kết quả của truy vấn được hiển thị thường sẽ là tất cả các dòng dữ liệu truy vấn được. Trong trường hợp cần hạn chế số lượng các dòng

**c) Mệnh đề điều kiện WHERE**

Mệnh đề WHERE trong câu lệnh SELECT được sử dụng nhằm xác định các điều kiện đối với việc truy xuất dữ liệu. Sau mệnh đề WHERE là một biểu thức logíc và chỉ những dòng dữ liệu nào thoả mãn điều kiện được chỉ định mới được hiển thị trong kết quả truy vấn.

Ví dụ: Câu lệnh dưới đây hiển thị mã số của các nhà cung cấp đã cung cấp mặt hàng có mã là H2

SELECT MaCongTy

FROM CungCap

WHERE MaHang='H2'

Trong mệnh đề WHERE thường sử dụng

Các toán tử kết hợp điều kiện (AND, OR)

Các toán tử so sánh.

Toán tử phạm vi và toán tử tập hợp

Các giá trị NULL

**c.1 Các toán tử so sánh**

|  |  |
| --- | --- |
| Toán tử | Ý nghĩa |
| [= (Equals)](ts_operator_0mnt.htm) | Ngang bằng |
| [> (Greater Than)](ts_operator_196h.htm) | Lớn hơn |
| [< (Less Than)](ts_operator_4qt5.htm) | Nhỏ hơn |
| [>= (Greater Than or Equal To)](ts_operator_3r21.htm) | Lớn hơn hoặc bằng |
| [<= (Less Than or Equal To)](ts_operator_1ewp.htm) | Nhỏ hơn hoặc bằng |
| [<> (Not Equal To)](ts_operator_8to9.htm) | Không bằng |
| [!= (Not Equal To)](ts_operator_4ent.htm) | Không bằng |
| [!< (Not Less Than)](ts_operator_5j95.htm) | Không nhỏ hơn |
| [!> (Not Greater Than)](ts_operator_5eih.htm) | Không lớn hơn |

### c.2 Toán tử phạm vi (Range Operator): [NOT] BETWEEN a AND b

Để kiểm tra xem giá trị dữ liệu nằm trong (ngoài) một khoảng nào đó, ta sử dụng toán tử [NOT] BETWEEN như sau:

|  |  |
| --- | --- |
| **Cách sử dụng** | **Ý nghĩa** |
| giá\_tri BETWEEN a AND b | a ≤ giá\_trị ≤ b |
| giá\_tri NOT BETWEEN a AND b | (giá\_trị < a) AND (giá\_trị) > b |

Ví dụ: Cho cơ sở dữ liệu quản lý điểm gồm các bảng sau:

+ LopHoc(MaLop, TenLop)

+ SinhVien(MaSV, HoDem, Ten, NgaySinh, GioiTinh, DiaChi, MaLop)

+ MonHoc(MaMon, TenMon, SoDvht)

+ Diem(MaSV, MaMon, DiemThi)

Câu lệnh dưới đây cho biết danh sách các môn học có số đơn vị học trình nằm trong khoảng từ 4 đến 5

SELECT \*

FROM MonHoc

WHERE SoDvht Between 4 And 5

Câu lệnh dưới đây cho biết danh sách các môn học có số đơn vị học trình không nằm trong khoảng từ 4 đến 5

SELECT \*

FROM MonHoc

WHERE SoDvht NOT Between 4 And 5

**c.3 Toán tử tập hợp (IN và NOT IN)**

Toán tử IN được sử dụng khi ta cần chỉ định điều kiện tìm kiếm dữ liệu cho câu SELECT là một danh sách các giá trị. Sau IN (hoặc NOT IN) có thể là một danh sách các giá trị hoặc là một câu lệnh SELECT khác.

Ví dụ: Để biết danh sách các môn học có số đơn vị học trình là 3, 4 hoặc 5 thay vì sử dụng câu lệnh

SELECT \* FROM MonHoc

WHERE SoDvht =3 OR SoDvht = 4 OR SoDvht = 5

Ta có thể sử dụng câu lệnh

SELECT \* FROM MonHoc

WHERE SoDvht IN (3,4,5)

**c.4 Toán tử LIKE và các ký tự đại diện**

Toán tử LIKE (hoặc NOT LIKE ) sử dụng trong câu lệnh SELECT nhằm mô tả khuôn dạng của cần tìm kiếm. Chùng thường kết hợp với các ký tự đại diện sau đây:

Dấu phần trăm (%): Chỉ một chuỗi các ký tự bất kỳ.

Dấu gạch dưới (\_): Chỉ một ký tự đơn bất kỳ

Ví dụ: Cho biết họ và tên của các sinh viên có họ đệm là Dương

SELECT HoDem, Ten FROM SinhVien

WHERE HoDem = 'Dương%'

**c.5 Giá trị NULL**

Trong mệnh đề WHERE, để kiểm tra giá trị của một cột có giá trị NULL hay không ta sử dụng cách viết:

WHERE tên\_cột IS NULL hoặc WHERE tên\_cột IS NOT NULL

**c.6 Toán tử SOME, ANY, ALL**

+ Toán tử SOME, ANY: Chúng được sử dụng kết hợp thêm với các phép toán so sánh căn bản như: =, >, >=, <, <=, !=, !>, !<... Trả về TRUE nếu một vài biểu thức thỏa mãn

+ Toán tử ALL: Chúng được sử dụng kết hợp thêm với các phép toán so sánh căn bản như: =, >, >=, <, <=, !=, !>, !<... Trả về TRUE nếu tất cả các biểu thức đều thỏa mãn

**d) Sắp xếp kết quả truy vấn**

Mặc định các dòng dữ liệu trong kết quả của câu truuy vấn tuân theo thứ tự của chúng trong bảng dữ liệu hoặc được sắp xếp theo chỉ mục (nếu trên bảng có chỉ mục). Trong trường hợp muốn dữ liệu được sắp xếp theo chiều tăng hoặc giảm của giá trị của một hoặc nhiều trường, ta sử dụng thêm mệnh đề ORDER BY trong câu lệnh SELECT.

Sau ORDER BY là danh sách các cột cần sắp xếp (tối đa là 16 cột). Dữ liệu được sắp xếp có thể theo chiều tăng (ASC) hoặc giảm (DESC), mặc định là sắp xếp theo chiều tăng.

Ví dụ: Câu lệnh dưới đây hiển thị danh sách các môn học và sắp xếp theo chiều giảm dần của số đơn vị học trình

SELECT \* FROM MonHoc

ORDER BY SoDvht DESC

***Chú ý:*** Nếu sau ORDER BY có nhiều cột thì việc sắp xếp dữ liệu sẽ được ưu tiên theo chiều từ trái qua phải.

Ví dụ: Liệt kê danh sách sinh viên và sắp xếp theo tên sinh viên theo Alphaber, nếu trùng tên thì sắp theo giới tính

SELECT \* FROM SinhVien

ORDER BY Ten, GioiTinh

**e) Phép kết nối**

Khi cần thực hiện một yêu cầu truy vấn dữ liệu từ hai hay nhiều bảng, ta phải sử dụng đến phép kết nối

Để thực hiện được một phép nối, cần phải xác định được những yếu tố sau:

Những cột nào cần hiển thị trong kết quả truy vấn.

Những bảng nào có tham gia vào truy vấn.

Điều kiện để thực hiện phép nối giữa các bảng dữ liệu là gì?

Trong các yếu tố kể trên, việc xác định chính xác điều kiện để thực hiện phép nối giữa các bảng đóng vai trò quan trọng nhất. Trong đa số các trường hợp, điều kiện của phép nối được xác định nhờ vào mối quan hệ giữa các bảng cần phải truy xuất dữ liệu. Thông thường, đó là điều kiện bằng nhau giữa khoá chính và khoá ngoài của hai bảng có quan hệ với nhau.

Ví dụ: Câu lệnh dưới đây hiển thị danh sách các sinh viên với các thông tin: Mã sinh viên, họ đệm và tên, mã lớp và tên lớp

SELECT MaSV, HoDem, Ten, Lop.MaLop, TenLop

FROM SinhVien , Lop

WHERE SinhVien.MaLop = Lop.MaLop

Trong câu lệnh trên, các bảng tham gia vào truy vấn bao gồm: SinhVien và Lop. Điều kiện để thực hiện phép kết nối giữa hai bảng là điều kiện sau:

SinhVien.MaLop = Lop.MaLop

***Chú ý:***

Tên của một số cột nào đó trong các bảng có tham gia vào truy vấn. Nếu tên cột trong các bảng trùng tên nhau thì tên cột phải được viết dưới dạng: ***Tên\_bảng.tên\_cột***

Dấu sao (\*) được sử dụng trong danh sách chọn khi cần hiển thị tất cả các cột của các bảng tham gia truy vấn.

Trong trường hợp cần hiển thị tất cả các cột của một bảng nào đó, ta sử dụng cách viết: ***tên\_bảng.\****

Ví dụ: Liệt kê danh sách cac sinh viên tham gia học môn 'Cơ sở dữ liệu'

SELECT SinhVien.MaSV, HoDem, Ten, MaLop, NgaySinh, GioiTinh, DiaChi

FROM SinhVien , MonHoc , Diem

WHERE SinhVien.MaMon = Diem.MaMon AND MonHoc.MaMon=Diem.MaMon

AND TenMon='Cơ sở dữ liệu'

hoặc viết dưới dạng ngắn gọn hơn:

SELECT SinhVien.\*

FROM SinhVien , MonHoc , Diem

WHERE SinhVien.MaMon = Diem.MaMon AND MonHoc.MaMon=Diem.MaMon

AND TenMon='Cơ sở dữ liệu'

**f) Thống kê dữ liệu với GROUP BY**

Mệnh đề GROUP BY sử dụng trong câu lệnh SELECT nhằm phân hoạch các dòng dữ liệu trong bảng thành các nhóm dữ liệu và trên mỗi nhóm dữ liệu thực hiện tính toán các giá trị thống kê như tính tổng, tính giá trị trung bình…

Các hàm nhóm được sử dụng để tính giá trị thống kê cho toàn bảng hoặc trên mỗi nhóm dữ liệu. Chúng có thể được sử dụng như là các cột trong danh sách chọn của câu lệnh SELECT hoặc xuất hiện trong mệnh đề HAVING, nhưng không được phép xuất hiện trong mệnh đề WHERE

SQL cung cấp các hàm nhóm dưới đây:

|  |  |
| --- | --- |
| **Hàm nhóm** | **Chức năng** |
| SUM(biểu\_thức) | Tính tổng các giá trị |
| AVG(biểu\_thức) | Tính trung bình của các giá trị |
| COUNT(biểu\_thức) | Đếm số các giá trị trong biểu thức |
| COUNT(\*) | Đếm số các dòng được chọn |
| MAX(biểu\_thức) | Tính giá trị lớn nhất |
| MIN(biểu\_thức) | Tính giá trị nhỏ nhất |

Trong đó:

Hàm SUM, AVG chỉ làm việc với các biểu thức số

Hàm SUM, AVG, COUNT, MIN và MAX bỏ qua các giá trị NULL khi tính toán.

Hàm COUNT(\*) không bỏ qua các giá trị NULL

**f.1 Thống kê trên toàn bộ dữ liệu**

Khi cần tính toán giá trị thống kê trên toàn bộ dữ liệu, ta sử dụng các hàm nhóm trong danh sách chọn của câu lệnh SELECT. Trong trường hợp này, trong danh sách chọn không được sử dụng bẩt kỳ một tên cột hay biểu thức nào ngoài các hàm gộp.

Ví dụ: Để thống kê trung bình điểm thi của tất cả các môn học ta sử dụng câu lệnh như sau:

SELECT AVG(DiemThi) FROM Diem

**f.2 Thống kê dữ liệu trên các nhóm**

Trong trường hợp cần thực hiện tính toán các giá trị thống kê trên các nhóm dữ liệu, ta sử dụng mệnh đề GROUP BY để phân hoạch dữ liệu vào trong các nhóm. Các hàm nhóm được sử dụng sẽ thực hiện thao tác tính toán trên mỗi nhóm và cho biết giá trị thống kê theo các nhóm dữ liệu.

Ví dụ: Câu lệnh dưới đây cho biết sĩ số sinh viên của mỗi lớp

SELECT Lop.MaLop, TenLop, COUNT(MaSV) AS SiSo

FROM Lop, SinhVien

WHERE Lop.MaLop = SinhVien. MaLop

GROUP BY Lop.MaLop, TenLop

***Chú ý:***

Biểu thức nào điều khiển việc phân nhóm dữ liệu thì các biểu thức đó phải được liệt kê sau mện đề GROUP BY.

Trong trường hợp danh sách chọn của câu lệnh SELECT có các hàm nhóm và những biểu thức không phải là đối số của các hàm nhóm thì những biểu thức này phải được liệt kê đầy đủ trong mệnh đề GROUP BY, nếu không câu lệnh sẽ không hợp lệ

Ví dụ: Dưới đây là một câu lệnh sai do thiếu trường TenLop sau mệnh đề GROUP BY

SELECT Lop.MaLop, TenLop, COUNT(MaSV) AS SiSo

FROM Lop, SinhVien

WHERE Lop.MaLop = SinhVien. MaLop

GROUP BY Lop.MaLop

**g) Mệnh đề điều kiện đối với hàm nhóm - HAVING**

Mệnh đề HAVING là mệnh đề đặt điều kiện lên các nhóm dữ liệu. Mệnh đề HAVING thường không thực sự có nghĩa nếu như không sử dụng kết hợp với mệnh đề GROUP BY. Một điểm khác biệt giữa HAVING và WHERE là trong điều kiện của WHERE không được có các hàm nhóm trong khi HAVING lại cho phép sử dụng các hàm nhóm trong điều kiện của mính.

Ví dụ: Đưa ra danh sách sinh viên có trung bình điểm thi lớn hơn hoặc bằng 5.

SELECT SinhVien.MaSV, HoDem, Ten, SUM(DiemThi\*SoDvht)/SUM(SoDvht)

FROM SinhVien , MonHoc , Diem

WHERE SinhVien.MaMon = Diem.MaMon AND

MonHoc.MaMon=Diem.MaMon

GROUP BY SinhVien.MaSV, HoDem, Ten

HAVING SUM(DiemThi\*SoDvht)/SUM(SoDvht) >=5

**h) Truy vấn con (Subquery)**

Truy vấn con là một câu lệnh SELECT được lồng vào bên trong một câu lệnh SELECT, INSERT, UPDATE, DELETE hoặc bên trong một truy vấn con khác. Loại truy vấn này được sử dụng để biểu diễn cho những truy vấn trong đó điều kiện truy vấn dữ liệu cần phải sử dụng đến kết quả của một truy vấn khác.

+ Tìm tên các công ty đã cung cấp mặt hàng H2

SELECT TenCongTy

FROM CongTy, CungCap

WHERE (CongTy.MaCongTy = CungCap.MaCongTy)

AND (MaHang = 'H2')

Hoặc có thể viết như sau:

SELECT TenCongTy

FROM CongTy

WHERE MaCongTy IN (SELECT MaCongTy

FROM CungCap

WHERE MaHang = 'H2')

Ví du: Câu lệnh sau liệt kê tên các công ty chưa từng cung cấp mặt hàng nào

SELECT TenCongTy

FROM CongTy

WHERE MaCongTy NOT IN (SELECT MaCongTy

FROM CungCap)

Hoặc có thể viết như sau:

SELECT \*

FROM CongTy

WHERE MaCongTy <> ALL (SELECT MaCongTy FROM CungCap)

+ Tìm những công ty cung cấp ít nhất một mặt hàng màu đỏ

SELECT TenCongTy

FROM CongTy

WHERE MaCongTy IN (SELECT MaCongTy

FROM CungCap

WHERE MaHang IN (SELECT MaHang

FROM HangHoa

WHERE màu = “ đỏ”))

+ Câu lệnh sau liệt kê tên các công ty chưa từng cấp mặt hàng có mã là H1.

SELECT TenCongTy

FROM CongTy

WHERE EXISTS (SELECT MaCongTy

FROM CungCap CC

WHERE CC.MaCongTy=CongTy.MaCongTy

AND MaHang <> 'H1')

**2.3.2 Truy vấn sử dụng phép nối bảng**

Trong hầu hết các câu lệnh truy vấn, kết quả mà ta mong muốn đều có liên quan đến một hoặc nhiều bảng (VIEW) khác nhau. Trong trường hợp như vậy ta cần sử dụng mệnh đề JOIN để kết hợp dữ liệu trên hai hay nhiều bảng dữ liệu lại với nhau.

Có 2 kiểu kết nối sau: Kết nối theo chiều ngang và kết nối theo chiều dọc

Kết nối theo chiều ngang: INNER JOIN (Kết nối tự nhiên), LEFT JOIN (Kết nối trái), RIGHT JOIN (Kết nối trái) , FULL JOIN, CROSS JOIN (Phép Tích đề các)

Kết nối theo chiều dọc: Sử dụng phép toán UNION

**a) Kết nối theo chiều ngang**

SELECT [ ALL | DISTINCT ] <danh\_sách\_chọn>  
FROM <Tên\_bảng\_thứ\_nhất>

INNER JOIN | LEFT JOIN| RIGHT JOIN | FULL JOIN <Tên\_bảng\_thứ\_hai>

ON <Biểu\_thức\_điều\_kiện\_kết\_nối\_bảng >

[INNER JOIN | LEFT JOIN| RIGHT JOIN | FULL JOIN <Tên\_bảng\_thứ\_3>

ON <Biểu\_thức\_điều\_kiện\_kết\_nối\_bảng >][…n]  
[ WHERE <Biểu\_thức\_điều\_kiện\_chọn >]   
[ GROUP BY <danh\_sách\_cột>]   
[ HAVING <điều\_kiện>]   
[ ORDER BY <tên\_cột> [ ASC | DESC ] [,..n]]

**Giải thích :**

+ Nếu lệnh truy vấn sử dụng mệnh đề INNER JOIN (kết nối tự nhiên) để kết hợp hai hay nhiều bảng dữ liệu lại với nhau thì những bản ghi nào được đưa ra kết quả của câu truy vấn nếu bộ giá trị của nó thỏa mãn biểu thức điều kiện nối bảng.

+ Nếu lệnh truy vấn sử dụng mệnh đề LEFT JOIN (kết nối trái) để kết hợp hai hay nhiều bảng dữ liệu lại với nhau ta được kết quả sau : Những bản ghi của bảng thứ nhất (bảng bên trái) so sánh được với các bản ghi của bảng thứ hai (dựa vào biểu thức điều kiện nối bảng) và những bản ghi của bảng thứ nhất không so sánh được với các bản ghi của bảng thứ hai (bảng bên phải) khi đó phần giá trị của các thuộc tính có trong bảng thứ 2 sẽ nhận giá trị NULL.

+ Nếu lệnh truy vấn sử dụng mệnh đề RIGHT JOIN (kết nối phải) để kết hợp hai hay nhiều bảng dữ liệu lại với nhau ta được kết quả sau : Những bản ghi của bảng thứ hai (bảng bên phải) so sánh được với các bản ghi của bảng thứ nhất (dựa vào biểu thức điều kiện nối bảng) và những bản ghi của bảng thứ hai không so sánh được với các bản ghi của bảng thứ nhất (bảng bên trái) khi đó phần giá trị của các thuộc tính có trong bảng thứ nhất sẽ nhận giá trị NULL.

+ Khi không cần quan tâm đến dữ liệu hiện ra cho dù dữ liệu bảng bên trái có tồn tại tương ứng dữ liệu bảng bên phải hay không (hoặc ngược lại) ta dùng FULL JOIN.

Ví dụ : Cho 2 bảng dữ liệu sau :

+ Bảng **LopHoc**

|  |  |
| --- | --- |
| **MaLop** | **TenLop** |
| 1 | K2A |
| 2 | K2B |
| 3 | K2C |

+ Bảng **SinhVien** để lưa danh sách sinh viên

|  |  |  |  |
| --- | --- | --- | --- |
| **MaSV** | **HoTen** | **GioiTinh** | **MaLop** |
| SV01 | Nguyễn Thị Hồng | 0 | 1 |
| SV02 | Trần Hùng | 1 | 2 |
| SV03 | Phạm Văn Tân | 1 | 1 |

**+ Sử dụng phép kết nối tự nhiên**

SELECT SinhVien.\*

FROM LopHoc

INNER JOIN SinhVien

ON LopHoc.MaLop=SinhVien.MaLop

**Kết quả  ta được bảng dữ liệu sau :**

|  |  |  |  |
| --- | --- | --- | --- |
| **MaSV** | **HoTen** | **GioiTinh** | **MaLop** |
| SV01 | Nguyễn Thị Hồng | 0 | 1 |
| SV03 | Phạm Văn Tân | 1 | 1 |
| SV02 | Trần Hùng | 1 | 2 |

**+ Sử dụng phép kết nối trái**

SELECT LopHoc.\*, MaSV, HoTen

FROM LopHoc

LEFT JOIN SinhVien

ON LopHoc.MaLop=SinhVien.MaLop

**Kết quả  ta được bảng dữ liệu sau :**

|  |  |  |  |
| --- | --- | --- | --- |
| **MaLop** | **TenLop** | **MaSV** | **HoTen** |
| 1 | K2A | SV01 | Nguyễn Thị Hồng |
| 1 | K2A | SV03 | Phạm Văn Tân |
| 2 | K2B | SV02 | Trần Hùng |
| 3 | K2C | NULL | NULL |

**+ Sử dụng phép kết nối phải**

SELECT LopHoc.\*, MaSV, HoTen

FROM LopHoc

RIGHT JOIN SinhVien

ON LopHoc.MaLop=SinhVien.MaLop

**+ Sử dụng phép kết nối FULL JOIN**

SELECT LopHoc.\*, MaSV, HoTen

FROM LopHoc

FULL JOIN SinhVien

ON LopHoc.MaLop=SinhVien.MaLop

**+ Sử dụng phép Tích Đề Các - CROSS JOIN**

SELECT LopHoc.\*,MaSV, HoTen, GioiTinh

FROM LopHoc

CROSS JOIN SinhVien

Kết quả : Bảng kết quả trả về 9 bản ghi sau

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MaLop | TenLop | MaSV | HoTen | GioiTinh |
| 1 | K2A | SV01 | Nguyễn Thị Hồng | 0 |
| 1 | K2A | SV02 | Trần Hùng | 1 |
| 1 | K2A | SV03 | Phạm Văn Tân | 1 |
| 2 | K2B | SV01 | Nguyễn Thị Hồng | 0 |
| 2 | K2B | SV02 | Trần Hùng | 1 |
| 2 | K2B | SV03 | Phạm Văn Tân | 1 |
| 3 | K2C | SV01 | Nguyễn Thị Hồng | 0 |
| 3 | K2C | SV02 | Trần Hùng | 1 |
| 3 | K2C | SV03 | Phạm Văn Tân | 1 |

**b) Kết nối theo chiều dọc – Dùng phép hợp (UNION)**

UNION là phép toán dùng để nối hai hay nhiều câu truy vấn lại với nhau

Ví dụ

SELECT \*

FROM SinhVien

WHERE MaLop=1

UNION

SELECT \*

FROM SinhVien

WHERE MaLop=2

**Một số chú ý khi sử dụng UNION**

Tất cả những truy vấn trong UNION phải cùng số cột hay trường. Nếu truy vấn thứ nhất có hai cột thì truy vấn thứ hai được UNION cũng có hai cột tương tự.

Tiêu đề cột trong kết quả của phép UNION sẽ là tiêu đề cột được chỉ định trong truy vấn đầu tiên.

Kiểu dữ liệu của các cột của truy vấn 2 phải tương thích với kiểu dữ liệu các cột tương ứng trong truy vấn thứ nhất.

Khi các kiểu dữ liệu khác nhau được kết hợp trong câu lệnh UNION, chúng sẽ được chuyển sang kiểu dữ liệu cao hơn (nếu có thể).

Phép toán UNION không thể sử dụng trong câu lệnh CREATE VIEW

**2..3.3 Bổ sung dữ liệu.**

Dữ liệu trong các bảng được thể hiện dưới các dòng (bản nghi). Để bổ xung thêm các dòng dữ liệu vào một bảng, ta sử dụng câu lệnh INSERT. SQL cung cấp các cách dưới đây để thực hiện thao tác bổ sung dữ liệu cho bảng:

Bổ sung từng dòng dữ liệu với mỗi câu lệnh INSERT.

Bổ sung nhiều dòng dữ liệu bằng cách truy xuất dữ liệu từ các bảng dữ liệu khác.

**a) Bổ sung từng dòng dữ liệu với lệnh INSERT**

INSERT INTO <tên\_bảng>[(<danh\_ sách\_tên\_cột>)]

VALUES (<danh\_sách\_các\_ giá\_ trị>)

Trong câu lệnh INSERT, danh\_sách\_tên\_cột ngay sau tên bảng không cần thiết phải chỉ định nếu giá trị các trường của bản ghi mới được chỉ định đầy đủ trong danh sách các giá trị. Trong trường hợp này, thứ tự các giá trị trong danh sách trị phải bằng số lượng các trường của bảng cần bổ sung dữ liệu cũng như phải tuân theo đúng thứ tự của các trường như khi bảng được định nghĩa.

Ví dụ: Câu lệnh dưới đây sẽ bổ sung thêm một công ty vào quan hệ congty

INSERT INTO CongTy

VALUES (C12, 'Hoà bình', 600000000, 'Thái nguyên','0280754379')

Trong trường hợp chỉ nhập giá trị cho một số cột trong bảng, ta phải chỉ định danh sách các cột cần nhập dữ liệu ngay sau tên bảng. Khi đó các cột không được nhập dữ liệu sẽ nhận giá trị mặc định (nếu có) hoặc nhận giá trị NULL (nếu cột cho phép nhận giá trị NULL). Nếu một cột không có giá trị mặc định và không chấp nhận giá trị NULL mà không được nhập dữ liệu, câu lệnh sẽ bị lỗi.

Ví dụ: Câu lệnh sau sẽ bổ sung một bản ghi mới cho bảng CongTy

INSERT INTO CongTy ( MaCongTy, TenCongTy, DiaChi, NganSach)

VALUES (C13,'Trần Anh', 'Hà nội' ,500000000).

Khi đó trường SoDT sẽ nhận giá trị NULL.

Câu lệnh trên còn có thể viết như sau:

INSERT INTO CongTy

VALUES (C13,'Trần Anh', 500000000,'Hà nội' ,NULL).

**b) Bổ sung nhiều dòng dữ liệu từ bảng khác**

Một cách sử dụng khác của câu lệnh INSERT được sử dụng để bổ sung nhiều dòng dữ liệu vào một bảng, các dòng dữ liệu này được lấy từ một bảng khác thông qua câu lệnh SELECT. Ở cách này, các giá trị dữ liệu được bổ sung vào bảng không được chỉ định tường minh mà thay vào đó là một câu lệnh SELECT truy vấn dữ liệu từ bảng khác.

Cú pháp của câu lệnh có dạng như sau:

INSERT INTO <tên\_bảng>[(<danh\_ sách\_tên\_cột>)]

<câu\_lệnh\_SELECT>

Ví dụ: Giả sử ta có bảng LuuCongTy bao gồm các trường MaCongTy, TenCongTy, NganSach, DiaChi. Câu lệnh dưới đây sẽ bổ sung vào bảng LuuCongTy các dòng dữ liệu có được từ câu truy vấn SELECT.

INSERT INTO LuuCongTy

SELECT MaCongTy, TenCongTy, NganSach, DiaChi

FROM CongTy WHERE DiaChi = 'Hà nội'

***Ghi chú:*** Kết quả của câu lệnh SELECT phải có số cột bằng với số cột được chỉ định trong bảng đích và phải tương thích về cả kiểu dữ liệu.

**2.3.4 Cập nhật dữ liệu**

Câu lệnh UPDATE trong SQL được sử dụng để cập nhật dữ liệu trong các bảng. Câu lệnh này có cú pháp như sau:

UPDATE <tên\_bảng>

SET <tên\_cột\_1>=<biểu\_thức\_1> [, …., <tên\_cột\_k> = <biểu\_thức\_k>]

[WHERE <điều\_kiện>]

Sau UPDATE là tên của bảng cần cập nhật dữ liệu. Một câu lệnh UPDATE có thể cập nhật dữ liệu cho nhiều cột bằng cách chỉ định danh sách tên cột và biểu thức tương ứng sau từ khoá SET. Mệnh đề WHERE trong câu lệnh UPDATE thường được sử dụng đề chỉ định các dòng dữ liệu chịu tác động của câu lệnh (nêu không chỉ định, phạm vi tác động của câu lệnh là toàn bộ các dòng trong bảng)

Ví dụ: Cập nhật lại địa chỉ của công ty có mã là CT1

UPDATE TABLE CongTy

SET DiaChi = 'Thái nguyên' WHERE MaCongTy ='CT1'

2.3.5 Xóa dữ liệu

Để xoá dữ liệu trong một bảng, ta sử dụng câu lệnh DELETE . Cú pháp của câu lệnh như sau:

DELETE FROM <tên\_bảng>

[WHERE <điều\_kiện>]

Trong đó, tên của bảng cần xoá dữ liệu được chỉ định sau DELETE FROM. Mệnh đề WHERE trong câu lệnh được sử dụng để chỉ định điều kiện đối với các dòng dữ liệu cần xoá. Nếu câu lệnh DELETE không có mệnh đề WHERE thì toàn bộ các dòng trong bảng đều bị xoá. Mệnh đề FROM chỉ định danh sách các bảng có dữ liệu liên quan đến việc xoá dữ liệu.

+Ví dụ: Câu lệnh sau sẽ xoá khỏi bảng CONGTY những công ty có địa chỉ ở Hà nội

DELETE FROM CongTy

WHERE DiaChi = 'Hà nội'

Ví dụ: Xoá khỏi bảng CongTy danh sách các công ty đã cung cấp mặt có mã là H1

DELETE FROM CongTy

WHERE MaCongTy In (SELECT DISTINCT MaCongTy

FROM CungCap

WHERE MaHang='H1')

+ Xoá khỏi bảng CongTy nhưng công ty chưa từng cung cấp mặt hàng nào.

DELETE FROM CongTy

WHERE MaCongTy NOT IN (SELECT DISTINCT MaCongTy

FROM CungCap)

+ Xoá tất cả các thông tin trong bảng CungCap : DELETE FROM CungCap

## 2.4 Variables (Biến) trong SQL Server

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

**2.4.1 Khai báo biến:**

DECLARE @<Tên\_biến> <Kiểu dữ liệu>[,

@<Tên\_biến> <Kiểu dữ liệu>[,

@<Tên\_biến> <Kiểu dữ liệu>]]

**Ghi chú:**

Khi khai báo nhiều biến, bạn không cần dùng lại từ khóa mà chỉ cần dùng dấu (,) để phân biệt.

Khi khai báo một biến, biến có giá trị ban đầu là NULL cho đến khi biến đó được gán giá trị khác.

**2.4.2 Gán giá trị cho biến**

SET @<Tên\_biến>=<Giá\_trị>|<biểu\_thức>| (Câu lệnh SELECT)

**Ghi chú:**

Giá trị gán vào biến phải có kiểu dữ liệu tương tự với kiểu dữ liệu đã khai báo cho biến.

Không nhất thiến gán biến với một giá trị, đôi khi giá trị đó là kết quả của câu lệnh truy vấn SELECT.

Ví dụ: Gán biến bằng giá trị cụ thể

Declare @x int, @y int

Set @y = 5

Set @x = @y + 3

Ví dụ: Gán biến bằng giá trị của SELECT

Declare @x int

Set @x=(Select count(\*) From SinhVien)

Print ‘Tổng số sinh viên là’ + Str(@x)

## 2.5. Phát biểu điều khiển

***2.5.1. Phát biểu IF…ELSE***

Cú pháp:

IF <biểu thức điều kiện>

<Lệnh đơn>|<Khối lệnh>

ELSE

<Lệnh đơn>|<Khối lệnh>

**Ghi chú:**

Khối lệnh là một tập các câu lệnh của T-SQL, khối lệnh phải được bao trong cặp từ khóa BEGIN …… END

Khối lệnh là một tập các câu lệnh của T-SQL

**VD:** Giả sử có 2 bảng LOP và LOP1 có cấu trúc giống hệt nhau. Ta viết SP nhập dữ liệu dùng chung cho cả 2 bảng nhưng cần phân biệt bởi một biến cờ bằng cách sử dụng tham số @flag. Nếu @flag =1 thì thêm vào bảng LOP ngược lại thì thêm vào bảng LOP1

CREATE STORED PROC NHAP

@flag bit, @malop char(3), @tenlop nvarchar(20)

AS

IF (@flag=1)

INSERT INTO LOP

VALUES (@malop,@tenlop)

ELSE

INSERT INTO LOP

VALUES (@malop,@tenlop)

***2.5.2 Phát biểu CASE***

Cú pháp:

CASE <biểu thức vào>

WHEN <gia tri 1> THEN <biểu thức 1>

[…n]

[ELSE <bieu thức n+1>]

Ví : Sử dụng phát biểu CASE để tính ngày nhập hàng là ngày thứ mấy trong tuần.

SELECT SHDN, MAKHACH, NGAYNHAP, NGAYTHU=

CASE DATEPART(W,NGAYNHAP)

WHEN 1 THEN ‘CN’

WHEN 2 THEN ‘THU 2’

WHEN 3 THEN ‘THU 3’

WHEN 4 THEN ‘THU 4’

WHEN 5 THEN ‘THU 5’

WHEN 6 THEN ‘THU 6’

[ELSE ‘THU 7’]

WHEN 7 THEN ‘THU 7’

END

FROM NHAP

+ Cú pháp 2:

CASE

WHEN <biểu thực điều kiện 1> THEN <biểu thức kết quả 1>

WHEN <biểu thực điều kiện 2> THEN <biểu thức kết quả 2>

………………………………………….

WHEN <biểu thực điều kiện n> THEN <biểu thức kết quả n>

[ELSE <biểu thức kết quả\_ELSE]

Ví dụ: Cho bảng Diem(SBD, HoTen,NS,GT,TongDiem) để lưu tổng hợp điểm thi của các thí sinh trong một đợt tuyển sinh. Hãy dựa vào bảng trên để đưa ra màn hình danh sách gồm các thuộc tính (SBD, HoTen,NS,GT,TongDiem, Đo/Truot). Với Đo/Truot (Đỗ/Trượt ) được tính theo nguyên tắc nếu sinh viên nào có tổng điểm<20 thì ghi **‘Truot’** ngược lại ghi **‘Đo’**

Select Diem.\*, Do\_Truot=CASE

WHEN TongDiem <20 THEN ‘Truot’

ELSE ‘Đo’

From Diem

***2.5.3 Phát biểu WHILE***

Phát biểu WHILE là phát biểu điều khiển lặp. Vòng lặp sẽ được thực hiện cho đến khi biểu thức logic trong While nhận giá trị False.

Cú pháp:

WHILE <Biểu thức logic>

<Các phát biểu SQL>

[Begin

<Các phát biểu SQL>

[BREAK]

<Các phát biểu SQL>

[CONTINUE]

End]

Trong đó:

- Phát biểu BREAK dùng để thoát khỏi vòng lặp. Khi gặp BREAK thì quá trình xử lý sẽ bỏ qua các phát biểu còn lại chưa xử lý.

- Phát biểu CONTINUE đối ngược với phát biểu BREAK, nếu gặp phát biểu này thì quá trình xử lý sẽ quay lại đầu vòng lặp.

VD:

DECLARE @DEM INT

SET @DEM=1

WHILE @DEM<5

BEGIN

INSERT INTO THU1

VALUES(@DEM,'THEM DONG MOI')

SET @DEM=@DEM+1

END

# CHƯƠNG 3. BẢNG ẢO (VIEW)

## 3.1. Khái niệm cơ bản về View

View là một **bảng ảo** có nội dung được xác định từ một truy vấn. Tuy nhiên khung nhìn không tồn tại như là một cấu trúc lưu trữ dữ liệu trong cơ sở dữ liệu. Dữ liệu bên trong khung nhìn thực chất là dữ liệu được xác định từ một hay nhiều bảng cơ sở dó đó nó phụ thuộc vào các bảng cơ sở.

View có thể được dùng như một kỹ thuật bảo mật. Nó đảm bảo rằng người sử dụng chỉ có thể truy nhập và sửa đổi dữ liệu mà view cung cấp. Người sử dụng không thể truy nhập hay sửa chữa phần dữ liệu còn lại trong các bảng cơ sở.

Việc sử dụng các truy vấn phức tạp cũng có thể được đơn giản hoá nhờ sử dụng view. Các truy vấn phức tạp đó có thể được lưu dưới dạng view và dữ liệu từ view có thể được lấy ra nhờ sử dụng các truy vấn đơn giản hơn.

Sau khi định nghĩa, view được sử dụng tương đương như bất kỳ bảng nào khác trong CSDL

## 3.2. Ưu điểm của View

**Cung cấp dữ liệu thích hợp cho người dùng (đầy đủ nhưng cũng chỉ vừa đủ):** Nhìn chung, View thường được dùng để tập trung, một cách đơn giản, và tuỳ biến dữ liệu tùy theo nhu cầu và quyền hạn mỗi người. Nó có thể được sử dụng làm môi trường bảo mật bằng cách cho phép người sử dụng truy cập và xử lý dữ liệu qua view. Những dữ liệu không cần thiết, bí mật, hay không phù hợp có thể bị loại bỏ khỏi view.

**Giấu độ phức tạp của CSDL**: View giấu đi sự phức tạp của thiết kế CSDL đối với người sử dụng. Điều này cho phép người phát triển thay đổi thiết kế CSDL mà không làm ảnh hưởng đến sự tương tác giữa người sử dụng và CSDL. Ngoài ra, người sử dụng có thể xem dữ liệu bằng cách sử dụng các tên dễ hiểu hơn so với các tên mã hoá thường được dùng trong CSDL.

Trong các câu truy vấn phức tạp, bao gồm các cấu truy vấn từ những dữ liệu không đồng bộ, các dữ liệu chi tiết được che dấu qua view (để hiểu rõ hơn về dữ liệu không dồng bộ, xem phần ”Tổ chức dữ liệu từ các nguồn không đồng nhất” dưới đây). Người sử dụng truy vấn view thay vì viết truy vấn để lấy dữ liệu hay thực hiện một script.

**Tổ chức dữ liệu từ các nguồn không đồng nhất:** Bạn có thể tạo view dựa vào những truy vấn phức tạp kết hợp hai hay nhiều bảng (table). View, đối với người sử dụng, giống như một bảng đơn được gọi là view phân vùng. Chẳng hạn, nếu một bảng chứa thông tin chi tiết về lương cho nhân viên ở Mỹ, và bảng khác chứa thông tin chi tiết về lương cho nhân viên ở Anh, một view có thể được tạo từ sự kết hợp (UNION) của những table này. View sẽ thể hiện thông tin chi tiết về lương cho các nhân viên trong cả 2 nước.

Các view phân vùng có thể dựa vào dữ liệu từ các nguồn không đồng nhất, như máy chủ từ xa (remote server), chứ không phải chỉ từ các bảng trong cùng CSDL. Ví dụ, để kết hợp dữ liệu từ các máy chủ từ xa (remote server), trong đó mỗi một máy chủ lưu dữ liệu cho một khu vực khác nhau của một tổ chức, bạn có thể tạo các truy vấn phân tán (những truy vấn truy xuất dữ liệu từ các nguồn dữ liệu không đồng nhất được lưu trên chính máy đó hoặc trên một máy khác) lấy dữ liệu từ mỗi nguồn dữ liệu. Do đó, bạn có thể tạo một view dựa vào các truy vấn phân tán (distributed queries) đó. Truy vấn chỉ đọc dữ liệu được yêu cầu từ một bảng trên máy chủ từ xa có chứa dữ liệu mà truy vấn đó yêu cầu. Các server khác được tham chiếu bởi các truy vấn phân tán trong view đó không truy cập được.

**Hạn chế kích cỡ đối tượng:**Các view không chứa dữ liệu. SQL server chỉ lưu định nghĩa của view trong CSDL.

## 3.3. Tạo view.

Sử dụng câu lệnh Create View để tạo View.

Cú pháp

CREATE VIEW view\_name

[(column\_name[,column\_name]…)]

[WITH ENCRYPTION |SCHEMABINDING ]

AS

< select\_Statement >

[WITH CHECK OPTION]

Trong đó

+ **view\_name**: xác định tên của view và cần phải tuân theo các qui tắc đặt tên.

+ **column\_name**: xác định tên của các cột sẽ xuất hiện trong view. Nếu không chỉ ra **column\_name** thì view được tạo có tên của các cột cũng như kiểu dữ liệu của chúng sẽ tương ứng với các cột trong danh sách chọn của câu lệnh SELECT.

+ **WITH ENCRYPTION**: Nếu có mệnh đề này SQL Server sẽ mã hoá đoạn mã lệnh dùng để định nghĩa view, phục vụ cho mục đích bảo mật.

WITH SCHEMABINDING: Tất cả các Table liên quan đến việc tạo View không thể xoá được khi View vẫn tồn tại.

+ **select\_statement:** xác định khai báo câu lệnh SELECT định nghĩa view. View có thể sử dụng dữ liệu chứa trong các view và bảng khác.

+ **WITH CHECK OPTION:** buộc các lệnh sửa đổi dữ liệu phải thoã mãn tiêu chí mà câu lệnh SELECT xác định view đưa ra.

**Ghi chú:**

+ Lệnh Create View phải được viết đầu tiên trong một Batch

+ Phải chỉ định phần **column\_name** trong câu lệnh CREATE VIEW trong các trường hợp sau:

Trong View có cột được pháp sinh từ một biểu thức số học.

Hai hay nhiều cột của khung nhìn có trùng tên.

Bạn muốn thay đổi tên cột trong khung nhìn khác với tên cột của bảng cơ sở

Ví dụ: Cho cơ sở dữ liệu gồm các bảng dữ liệu sau:

Bảng DonVi(MaDV, TenDV)

Bảng NhanVien(MaNV, HoTen, NgaySinh, GioTinh, MaDV)

+ Tạo view để chứa các nhân viên có giới tính bằng 1

Create View vwgioitinh

As

Select MaNV, HoTen, NgaySinh, GioiTinh

From NhanVien

+ Câu lệnh dưới đây sẽ bị lỗi do tên của cột thứ 3 chưa được xác định

Create View vwnhanvien

As

Select MaNV, HoTen, Datediff(year,NgaySinh,Getdate()),TenDV

From NhanVien, DonVi

Where NhanVien.MaDV=DonVi.MaDV

+ Để câu lệnh trên có thể thực hiện được, bạn phải đặt tên cho các cột trong khung nhìn như sau:

Create View vwnhanvien(MaNV, HoTen, Tuoi, DonVi)

As

Select MaNV, HoTen, Datediff(year,NgaySinh,Getdate()) ,TenDV

From NhanVien, DonVi

Where NhanVien.MaDV=DonVi.MaDV

## 3.4. Các hạn chế của view

* Tên của view phải tuân theo qui tắc đặt tên, phải duy nhất và không được giống tên của bảng cơ sở.
* View chỉ được tạo nếu có quyền SELECT trên bảng cơ sở.
* View không thể lấy dữ liệu từ các table tạm.
* Định nghĩa View không thể bao gồm các mệnh đề ORDER BY, COMPUTER COMPUTER BY hoặc từ khoá INTO
* Không thể sử dụng ORDER BY cùng câu lệnh SELECT trong view

## 3.5. Thay đổi, hủy và đổi tên các View

**a) Thay đổi View**

Bạn có thể sửa lại view mà không cần phải hủy nó. Như thế sẽ đảm bảo rằng các quyền trên view không bị mất đi khi hủy view. Bạn cũng có thể sửa đổi view mà không làm ảnh hưởng tới các đối tượng phụ thuộc của view.

Bạn có thể sử dụng câu lệnh ALTER VIEW để sửa view.

Cú pháp

ALTER VIEW view\_name [(column\_name)]

[WITH ENCRYPTION]

AS select\_statement

[WITH CHECK OPTION]

+ view\_name: tên của view sẽ được thay đổi.

+ column\_name xác định tên của (các) cột được sử dụng trong view.

+ Nếu column\_name không được xác định, view được tạo sẽ có cùng các cột như đã xác định trong select\_statement.

+ WITH ENCRYPTION mã hoá định dạng text cho the view trong bảng syscomments.

+ select­\_ statement chỉ ra câu lệnh SELECT xác định view. View có thể sử dụng các view hoặc các bảng khác.

+ WITH CHECK OPTION buộc các lệnh sửa đổi dữ liệu phải thoã mãn tiêu chí mà câu lệnh SELECT xác định view đưa ra.

**b) Xóa View**

* Bạn có thể sử dụng câu lệnh DROP\_VIEW để hủy một view trong CSDL.
* Việc hủy view không gây ảnh hưởng gì đến các bảng cơ sở của view.
* Việc hủy view chỉ hủy các định nghĩa của view và các quyền được phân trên view.
* nếu bạn truy vấn bất kể một view nào tham chiếu đến một bảng đã bị xóa, bạn sẽ nhận được thông báo lỗi.

Cú pháp: DROP VIEW view\_name

Trong đó

+ view\_name là tên của view cần xóa.

+ Bạn có thể xóa nhiều view bằng một câu lệnh DROP VIEW. Tên của các view cần xóa được phân cách bởi dấu “,”

**c) Đổi tên View**

Bạn có thể đổi tên một view mà không cần xóa nó. Điều này đảm bảo cho các quyền được phân trên view không bị mất. Có thể được đổi tên view bằng cách sử dụng stored procedure hệ thống sp\_rename.

Cú pháp : sp\_rename old\_viewname, new\_viewname

Trong đó

old\_viewname là tên view cần đổi

new\_viewname là tên mới của view

**Chú ý**: Khi đổi tên view, bạn phải chắc chắn rằng:

* View phải nằm trong database hiện tại
* Tên mới phải được đặt đúng theo quy tắc đặt tên
* Chỉ người tạo view mới có quyền đổi tên
* Người tạo database cũng có quyền đổi tên view
* Định nghĩa View tồn tại ngay cả khi bảng tham gia đã bị xoá

## 3.6. Sử dụng View để cập nhật dữ liệu

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

Khi chúng ta thay đổi dữ liệu thông qua View đồng nghĩa với chúng ta thay đổi dữ liệu trên các bảng mà View đó đang tham chiếu. Tuy nhiên khi sử dụng View để cập nhật dữ liệu ta phải tuân theo các quy luật sau: Câu lệnh SELECT trong định nghĩa View không nên chứa:

* Các hàm nhóm dữ liệu hoặc biểu thức
* Các mệnh đề TOP, GROUP BY, UNION, DISTINCT.

Việc áp dụng View để cập nhật dữ liệu chỉ sử dụng với View đơn giản (tham chiếu đến duy nhất một Table). Nếu muốn cập nhất dữ liệu trên nhiều bảng (View phức tạp) ta phải sử dụng INSTEAD OF TRIGGER

# CHƯƠNG 4.

# THỦ TỤC LƯU TRỮ (STORED PROCEDURE – SP)

## 4.1. Khái niệm cơ bản về Stored Procedure

Ngoài những đối tượng trong SQL Server 2000 như đã giới thiệu như: table, view còn có Stored procedure (Thủ tục lưu trữ). Thủ tục lưu trữ là một đối tượng xây dựng bởi những phát biểu T-SQL.

Stored procedure được lưu trữ như một phần của CSDL, với cấu trúc như một văn bản Text được lưu trữ lại trong CSDL dưới dạng thủ tục lưu trữ. SP là một nhóm câu lệnh T-SQL đã được bien dịch sẵn được chữa trong SQL server với một tên gọi nào đó và được xử lý như một đơn vị.

## 4.2. Các ưu điểu của thủ tục lưu trữ

- Việc thực hiện tốt hơn do SQL không phải biên dịch lại thủ tục

- Giảm tắc nghẽn mạng do các ứng dụng không cần đưa các câu lệnh T-SQL tới máy chủ để thực hiện.

- Tính nhất quán tốt hơn do các câu lệnh T-SQL được định nghĩa trong SP là đồng bộ được thực hiện qua tất cả các ứng dụng.

- Kỹ thuật bảo mật tốt hơn do các user có thể được thực hiện một thủ tục nhưng cho dù không sở hữu nó.

## 4.3. Phân loại Store Procedure

Các Thủ tục lưu trữ có thể được phân thành 5 loại:

+ Thủ tục lưu trữ được định nghĩa bởi người dùng : Các Thủ tục lưu trữ này được tạo bởi người dụng và được lưu trong CSDL hiện tại

+ Thủ tục lưu trữ hệ thống : Có tên tiền tố là sp\_. Và hỗ trợ chủ yếu cho Quản trị viên quản lý SQL Server. Các CSDL Hệ thống lưu trữ các thủ tục lưu trữ hệ thống là master và msdb.

+ Thủ tục lưu trữ tạm thời : Có tên tiền tố với biểu tượng #. Chúng được lưu trữ trong CSDL tempdb và được drop tự động khi các client connect tới đầu cuối của server

+ Thủ tục lưu trữ từ xa : Là các thủ tục được tạo và lưu trữ trong CSDL trên máy chủ từ xa. Nó có thể được truy nhập từ nhiều máy chủ, được cung cấp bởi các Người dùng có quyền tương đương

+ Thủ tục lưu trữ mở rộng : Đó là thư viện liên kết động (DLL’s) được thực hiện bên ngoài môi trường SQL Server. Chúng được xác định bởi tiền tố xp\_

## 4.4. Tạo thủ tục lưu trữ

CREATE PROC[EDURE] <Ten thu tuc luu tru>

[<@tenbien1> <kieu du lieu>[= <default>][out put]]

[,…[<@tenbienn> <kieu du lieu>[= <default>][out put]]

[WITH ENCRYPTION]

AS <Code>

**Trong đó:**

- Ten thu tuc luu tru: Là tên của thủ tục do người dùng đặt tên tuân theo quy tắc đặt tên.

- [<@tenbien1> <kieu du lieu>[= <default>][output]]: Khai báo tham số cho thủ tục. Có 2 kiểu tham số:

+ Tham số đầu vào: Cho phép người dùng chuyển dữ liệu tới thủ tục khi thực thi thủ tục.

Khai báo:

<@tenbien1> <kieu du lieu>[= <default>]

+ Tham số đầu ra: Cho phép chuyển dữ liệu trở lại người dùng khi thủ tục được thực thi.

Khai báo:

<@tenbien1> <kieu du lieu>[= <default>] Output

- Code: Phần mã lệnh T-SQL để thực hiện một nhiệm vụ nào đó

## 4.5. Ví dụ.

Cho CSDL quản lý sinh viên gồm các bảng dữ liệu:

LOP(Malop, Tenlop)

SV(Masv, Tensv, Malop)

**1. Tạo một SP đơn giản (không sử dụng tham số)**

- Tạo SP lưu trữ toàn bộ sinh viên có tên là ‘Nguyễn Thị Hà’

CREATE PROC SPDS

AS

SELECT SV.\*

FROM SV

WHERE TENSV= ‘ Nguyễn Thị Hà’

- Thực thi (Gọi thủ tục) không sử dụng tham số

[EXEC]<ten thu tuc>

Ví dụ:

EXEC SPDS

**2. Tạo thủ tục có sử dụng tham số**

***- Tạo thủ tục sử dụng tham số đầu vào có giá trị mặc nhiên***

Tạo SP có tham số đầu vào @Tensv để lưu lại các sinh viên có tên ở trên

CREATE PROC SPDSSV

@Tensv Nvarchar(30) = ‘Nguyễn Thị Hà’

AS

SELECT SV.\*

FROM SV

WHERE Tensv=@Tensv

***- Tạo thủ tục sử dụng tham số đầu vào không có giá trị mặc nhiên***

CREATE PROC SPDSSV1

@Tensv Nvarchar(30)

AS

SELECT SV.\*

FROM SV

WHERE Tensv=@Tensv

***- Thực thi thủ tục có sử dụng tham số.***

[EXEC]<ten thu tuc> <gia tri 1,…,gia tri n>

EXEC SPDSSV1 ‘Nguyễn Thanh Hải’

**Ghi chú:** Danh sách các giá trị truyền vào phải tương ứng với các tham biến đã khai báo khi định nghĩa thủ tục (tương ứng về số lượng biến, tương ứng với thứ tự biến, tương ứng với kiểu dữ liệu của biến đó )

***- Tạo thủ tục có tham số Output***

Tạo thủ tục DSLop có tham số đầu ra là số lượng sinh viên của lớp

CREATE PROC SPDSLOP

@malop char(3), @SL int output

AS

SELECT @SL=COUNT(MALOP)

FROM SV

WHERE MALOP=@MALOP

Để lưu giá trị được trả về từ thủ tục ta phải khai báo biến

DECLARE @SLSV INT

EXEC SPDSLOP ‘K5C’, @SLSV OUTPUT

PRINT CONVERT(CHAR(3),@SLSV)

## 4.6. Thay đổi cấu trúc store procedure

Để thay đổi Stored Procedure trong SQL Server, ta sử dụng lệnh ALTER PROCEDURE

ALTER PROC[EDURE] <Ten thu tuc luu tru>

[<@tenbien1> <kieu du lieu>[= <default>][out put]]

[,…[<@tenbienn> <kieu du lieu>[= <default>][out put]]

[WITH ENCRYPTION]

AS <Code>

**Ghi chú:**

Thủ tục đó phải tồn tại

Tùy thuộc vào quyền hạn của người sử dụng có thể thay đổi Store Procedure hay không.

Kiểm tra tất cả các thông tin có liên quan đến các đối tượng khác trong khi bị thay đổi

## 4.7. Xóa thủ tục

DROP PROC <Tên thủ tục>

Ví dụ: Xoá thủ tục SPDSLOP

DROP PROC SPDSLOP

# CHƯƠNG 5. BẪY LỖI - TRIGGER

## 5.1. Khái niệm cơ bản về Trigger.

- Trigger là một loại STORE PROCEDURE đặc biệt được thực thi một cách tự động khi có một sự kiện thay đổi dữ liệu xảy ra : UPDATE, INSERT hoặc DELETE. Trigger được dùng để đảm bảo ràng buộc dữ liệu, tính nhất quán, hoặc thực hiện các quy tắc dữ liệu phức hợp.

- Trigger gồm một đoạn mã được biên dịch sẵn và gắn vào bảng dữ liệu và được tự động thực thi khi có một hành động xảy ra tương ứng với Trigger được gán cho hành động đó.

- Trigger không sử dụng 2 đặc tính của thủ tục lưu trữ là tham số và giá trị trả về.

## 5.2. Ứng dụng của Trigger.

- Kiểm tra dữ liệu nhập vào có phù hợp với mối liên hệ giữa các bảng dữ liệu với nhau?

- Ràng buộc toàn vẹn dữ liệu cho phù hợp với mô hình quan hệ CSDL.

- Kiểm soát dữ liệu hiện tại khi có thay đổi đến giá trị trong mẩu tin của bảng.

- Kiểm chứng khi xóa mẩu tin trong bảng.

## 5.3. Phân loại Trigger.

Dựa vào ứng dụng của Trigger trên một bảng dữ liệu có thể phân loại Trigger như sau:

- Insert Trigger: Xảy ra khi thực hiện thêm dữ liệu vào bảng

- Update Trigger: Xảy ra khi thực hiện thay đổi dữ liệu vào bảng

- Delete Trigger: Xảy ra khi thực hiện xóa dữ liệu trong bảng.

- Inserted of Trigger: Là trigger cho phép view chấp nhận sự thay đổi dữ liệu trong các bảng liên quan trong View.

## 5.4. Tạo và quản lý Trigger

***5.4.1. Tạo Trigger***

CREATE TRIGGER <Ten Trigger>

ON <Ten bang>| <Ten view>

[WITH ENCRYPTION>

FOR [ AFTER] [DELETE],[INSERT],[UPDATE]

AS

<CODE>

Trong đó:

- Ten Trigger: Là tên của đối tượng do người dùng đặt tên theo nguyên tắc đặt tên.

- ON: Chỉ ra rằng Trigger đang viết có tác dựng đến bảng dữ liệu hay bảng ảo nào.

- FOR hoặc AFTER chỉ ra rằng Trigger sẽ áp dụng cho hành động nào trong 3 hành động INSERT, UPDATE, DELETE. Mệnh đề có dạng như sau:

FOR INSERT

Hay

FOR UPDATE

Hay

FOR DELETE

Hay

FOR INSERT,UPDATE,DELETE.

- CODE: Phần mã gồm tập câu lệnh T-SQL nhằm mục đích thực thi hành động khi Trigger được

***5.4.2. Ví dụ:***

1. Tạo Trigger thực thi hành động sau khi thực hiện thêm dữ liệu vào bảng thì in ra thông báo ‘Bảng đã được thêm dữ liệu’

--TAO TRIGGER IN RA THONG BAO DON GIAN--

DROP TRIGGER TBAO

[SET NOCOUNT ON]

GO

CREATE TRIGGER TBAO

ON HANGHOA

FOR INSERT

AS

PRINT 'DA THEM DU LIEU VAO BANG'

GO

**2.** Cho CSDL quản lý điểm gồm các bảng dữ liệu sau:

SV(MaSV, TenSV, NgayS)

MH(Mamh, Tenmh, DVHT)

DIEM(Mamh, MaSV, HK, Diemthi)

Hãy tạo Trigger kiểm tra việc nhập dữ liệu cho bảng DIEM. Yêu cầu điểm thi nhập vào phải >=0 và <=10.

create trigger KTNHAP

on DIEM

for insert

as

if exists (select Diemthi

from DIEM

where (Diemthi<0 ) or (Diemthi>10))

begin

print'ban ghi khong hop le, hay nhap lai'

rollback tran

end

else print'da nhap thanh cong'

***5.4.3. Inserted và Deleted***

Là 2 bảng tạm được tạo ra khi Trigger được thực thi. Chúng được tham khảo như bảng thật nhưng lưu trong bộ nhớ chứ không phải trên đĩa.

Hai bảng có cùng cấu trúc (cùng số cột và kiểu dữ liệu) với Table hoặc View mà Trigger tham chiếu đến

- Bảng Inserrted lưu trữ bản sao dữ liệu các bản ghi được thêm vào bảng mà Trigger tham chiếu đến bởi sự thực thi của phát biểu Insert hoặc Update (Với phát biểu Update thì Inserted lưu trữ bản ghi dữ liệu mới sau khi Update)

- Bảng Deleted lưu trữ các dòng bị xóa khỏi bảng mà Trigger đang tham chiếu đến bởi sự thực thi của phát biểu Delete hoặc Update (Với phát biểu Update thì Deleted lưu trữ bản ghi dữ liệu cũ bị xóa khỏi bảng )

**VD:**

**1. --TRIGGER KIEM TRA NHAP DU LIEU CHO BANG HANG HOA--**

DROP TRIGGER KTNDL

CREATE TRIGGER KTNDL

ON HANGHOA

AFTER INSERT

AS

IF (SELECT SL

FROM INSERTED)<0

BEGIN

PRINT 'dU LIEU KHONG HOP LE'

ROLLBACK TRAN

END

ELSE PRINT 'DU LIEU HOP LE'

**2.** + Bảng **MATHANG** để lưu trữ thông tin về các mặt hàng gồm các thuộc tính sau:

|  |  |
| --- | --- |
| **Tên Thuộc tính** | **Giải thích** |
| MaH | Mã hàng |
| TenH | Tên hàng |
| Soluong | Số lượng |

+ Bảng **NHATKYBAN** để lưu thông tin về các lần bán hàng gồm các thuộc tính sau:

|  |  |
| --- | --- |
| **Tên Thuộc tính** | **Giải thích** |
| STT | Số thứ tự |
| Ngay | Ngày bán |
| Nguoimua | Tên người mua hàng |
| MaH | Mã hàng |
| SL | Số lượng hàng được bán |
| Gia | Giá bán hàng |

**Ví dụ: Sử dụng Inserted Trigger**

Tạo Trigger có chức năng tự động giảm số lượng hiện có khi một mặt hàng nào đó được bán (tức là khi câu lệnh Insert được thực thi trên bảng NHATKYBAN).

CREATE TRIGGER CNHAT

ON NKBAN

FOR INSERT

AS

SELECT \* FROM INSERTED

UPDATE HANGHOA

SET HANGHOA.SL=HANGHOA.SL-INSERTED.SLB

FROM INSERTED

WHERE HANGHOA.MAH=INSERTED.MAH

**3. Tạo Deleted Trigger**

Giả sử cần tạo bảng để lưu lại những mặt hàng đã bị xóa dùng cho mục đích phân tích sau này. Mỗi mặt hàng trong bảng MATHANG khi bị xóa sẽ được lưu lại trong bảng này.

Ta tạo bảng TTHANGXOA có cấu trúc giống hệt với bảng MATHANG

Tạo Trigger để lưu lại thông tin mỗi khi xóa

CREATE TRIGGER TTHANGXOA

ON HANG

FOR DELETE

AS

INSERT INTO HANGXOA SELECT \*

FROM DELETED

GO

**4. Tạo Update Trigger**

Tạo Trigger kiểm tra cột Giá trong bảng NHATKYBAN thực hiện hành động sau: Giá chỉ cập nhật lại khi Giá mới tăng không quá 10% so với giá cũ.

CREATE TRIGGER CN

ON BAN

FOR UPDATE

AS

DECLARE @DGC INT, @DGM INT

SELECT @DGC=DGB FROM DELETED

PRINT 'GIA CU ='

PRINT CONVERT(CHAR(10),@DGC)

SELECT @DGM =DGB FROM INSERTED

PRINT 'GIA MOI ='

PRINT CONVERT(CHAR(10),@DGM)

IF (@DGM>(@DGC\*1.10))

BEGIN

PRINT 'GIA MOI TANG QUA 10%, KHONG CAP NHAT'

ROLLBACK TRAN

END

ELSE

PRINT 'GIA MOI CHAP NHAN DUOC'

GO

***5.4.4. Quản lý Trigger***

**- Xem mã Trigger**

SP\_helptext <Tên Trigger>

SP\_helptext CNHAT

**- Xem những Trigger nào đang tồn tại trên bảng**

SP\_Helptrigger <Tên bảng>

SP\_helptrigger BAN

**- Xóa Trigger**

Drop Trigger <Tên Trigger>

Drop Trigger CNHAT

**- Sửa đổi Trigger:**

Alter Trigger <Tên Trigger>

On <Ten bảng>| <Tên view>

For Insert | [Update] | [Delete]

AS <Code>

**- Vô hiệu hoặc làm cho có hiệu lực một Trigger**

Dùng phát biểu ALTER TABLE để vô hiệu hoặc cho có hiệu lực một Trigger thay vì phải xóa định nghĩa của nó khỏi bảng.

+ Vô hiệu một Trigger:

ALTER TABLE NKBAN

Disable Trigger CNHAT

+ Làm cho Trigger có hiệu lực trở lại

ALTER TABLE NKBAN

Enable Trigger CNHAT

# CHƯƠNG 6. KIỂU DỮ LIỆU CON TRỎ (CURSOR)

## 6.1. Khái niệm cơ bản về CURSOR

Khi sử dụng câu select để truy vấn dữ liệu thì kết quả của câu select trả về dưới dạng bảng tuy nhiên nếu muốn truy xuất từng giá trị trong một dòng dữ liệu thì các câu lệnh select trong SQL server không thực hiện được, điều này được giải quyết bằng cách sử dụng kiểu dữ liệu con trỏ.

***Cursor là kiểu dữ* liệu *dùng để lấy một tập hợp mẩu tin từ bảng hay từ một câu truy vấn Select. Cursor được sử dụng khi cần truy vấn cà xử lý những mẩu tin.***

Kết quả của một tập hợp mẩu tin được gán bởi Cursor phải thực hiện theo các bước sau:

* Khai báo tên với kiểu Cursor
* Cursor sau khi khai báo được gán vào một tập hợp mẩu tin
* Mẩu tin được truy cập bằng cách tham chiếu đến tên của Cursor
* Khi mở Cursor, Cursor sẽ có hiệu lực cho đến khi gặp lệnh đóng Cursor

## 6.2. Các bước thực hiện một Cursor

* Khai báo Cursor
* Mở Cursor
* Xử lý mẩu tin trên Cursor
* Đóng Cursor
* Hủy bỏ Cursor

***6.2.1. Khai báo Cursor***

DECLARE *cursor\_name* CURSOR   
[ LOCAL | GLOBAL ]   
[ FORWARD\_ONLY | SCROLL ]   
[ STATIC | KEYSET | DYNAMIC ]   
FOR *select\_statement*

Trong đó:

+ LOCAL | GLOBAL : Xác định phạm vi hoạt động của Cursor

* Nếu khai báo LOCAL (mặc định), Cursor chỉ có hiệu lực trong bó lệnh có khai báo nó.
* Nếu khai báo GLOBAL, Cursor đó sẽ có phạm vi ảnh hưởng đến toàn bộ hoạt động của một kết nối.

FORWARD\_ONLY | SCROLL : Dùng để điều khiển việc dịch chuyển trỏ bản ghi

* Nếu chọn Forward\_only chỉ cho phép di chuyển trỏ bản ghi theo chiều tiến.
* Nếu chon SCROLL cho phép di chuyển trỏ bản ghi theo cả hai chiều là tiến hoặc lùi.( *Forward\_only là mặc định nếu không chỉ rõ STATIC | KEYSET | DYNAMIC khi khai báo kiểu Cursor. Ngược lại khi khai báo có chỉ rõ STATIC | KEYSET | DYNAMIC thì mặc định là SCROLL)*

STATIC | KEYSET | DYNAMIC: Dùng để phân loại Cursor.

* Nếu chọn STATIC ( kiểu trỏ tĩnh), Khi loại trỏ này được tạo ra hầu như không liên quan gì đến việc thay đổi cơ sở dữ liệu (mọi sự thay đổi dữ liệu trên các bảng nguồn của câu lệnh select sẽ không tác động đến kết quả mà con trỏ đang duyệt (*vùng làm việc của con trỏ không bị ảnh hưởng khi dữ liệu nguồn bi thay đổi*)).
* Ngược lại nếu chọn DYNAMIC (Cursor động): mọi sự thay đổi dữ liệu trên các bảng nguồn của câu lệnh select sẽ tác động đến kết quả mà con trỏ đang duyệt (vùng làm việc của con trỏ bị ảnh hưởng khi dữ liệu vùng nguồn bi thay đổi).

Phát biểu Select trả về một tập hợp mẩu tin hợp lệ. Cursor có số mẩu tin bằng số lượng mẩu tin trả về của phát biểu Select.

VD:

DECLARE THU

CURSOR FOR

SELECT SV.\*

FROM SV

WHERE MALOP= ‘K5C’

***6.2.2. Mở Cursor***

CP: OPEN <Ten Cursor>

VD:

OPEN THU

***6.2.3. Lấy mẩu tin từ Cursor***

Sau khi mở Cursor ta cần khai báo lệnh để lấy giá trị của các cột từ mẩu tin trong Cursor bằng cú pháp:

CP: FETCH [Next | Prior | First | Last] FROM <Ten Cursor>

INTO @<Ten bien 1>, @<Ten bien 2>,…, @<Ten bien n>

Trong đó:

* FETCH: Sử dụng phát biểu FETCH để lấy dữ liệu từ Cursor
* NEXT: Nhảy đến mẩu tin kế tiếp
* PRIOR: Nhảy về mẩu tin trước
* FIRST: Nhảy đến mẩu tin đầu tiên
* LAST: Nhảy đến mẩu tin cuối cùng
* @<Ten bien>: Là các biến cục bộ

**Chú ý:** Nếu sử dụng FETCH NEXT cho mẩu tin đầu tiên trong Cursor sẽ tương đương với FETCH FIRST.

**VD: Khai báo Cursor, mở và lấy dữ liệu từ Cursor**

/\* Khai báo biến cục bộ\*/

DECLARE @ MASV CHAR(5), @TENSV NVARCHAR(30), @MALOP CHAR(5)

/\* Khai báo biến Cursor\*/

DECLARE DSSV

CURSOR FOR

SELECT MASV, TENSV, MALOP

FROM SINHVIEN

/\*Mở Cursor\*/

OPEN DSSV

/\* Lấy dữ liệu vào biến cục bộ\*/

FETCH NEXT FROM DSSV

INTO @MASV, @TENSV, @MALOP

***6.2.4. Kiểm tra trạng thái Cursor***

Kiểm tra cuối mẩu tin bằng hàm hệ thống @@Fetch\_Status, với hàm này có thể trả về giá trị trong khi lấy dữ liệu từng mẩu tin.

0: Nếu lấy mẩu tin thành công

-1: Nếu lấy mẩu tin thất bại (Do cursor đang ở BOF hoặc EOF)

-2: Nếu lấy mẩu tin thất bại, với lý do không tồn tại.

VD:

/\* Lấy dữ liệu vào biến cục bộ\*/

FETCH NEXT FROM DSSV

INTO @MASV, @TENSV, @MALOP

WHILE @@FETCH\_STATUS = 0

BEGIN

PRINT ‘KET QUA’

PRINT @MASV+ ‘---‘ + @TENSSV+ ‘---‘ + @MALOP

FETCH NEXT FROM DSSV

INTO @MASV, @TENSV, @MALOP

END

***6.2.5. Đóng, giải phóng Cursor***

- Đóng Cursor

Close <Tên Cursor>

* Hủy bỏ Cursor

DEALLOCATE <Tên Cursor>

***6.2.6. Ví dụ.***

**1. Tạo Cursor đơn giản đưa ra thông tin của những mặt hàng có trong cửa hàng**

**--KHAI BAO CURSOR--**

DECLARE THU

CURSOR FOR

SELECT \*

FROM HANGHOA

**--MO CURSOR--**

OPEN THU

**--KHAI BAO BIEN CUC BO--**

DECLARE @MAH CHAR(3), @TENH NVARCHAR(20), @SL INT

**--LAY DU LIEU TU CURSOR--**

FETCH NEXT FROM THU

INTO @MAH,@TENH,@SL

WHILE @@FETCH\_STATUS =0

BEGIN

PRINT '--'+@MAH+'---'+@TENH+'---'+STR(@SL)

FETCH NEXT FROM THU

INTO @MAH,@TENH,@SL

END

**--DONG CURSOR--**

CLOSE THU

**--HUY BO CURSOR--**

DEALLOCATE THU

2. + Bảng **DMKhach** để lưu các danh mục các khách hàng gồm các thuộc tính sau:

|  |  |
| --- | --- |
| **Tên Thuộc tính** | **Giải thích** |
| MaKhach | Mã khách hàng |
| TenKhach | Tên khách hàng |
|  |  |
| DienThoai | Điện thoại |

+ Bảng **DMHang** để lưu danh mục hàng hoá gồm các thuộc tính sau:

|  |  |
| --- | --- |
| **Tên Thuộc tính** | **Giải thích** |
| MaHang | Mã hàng |
| TenHang | Tên hàng |
| DVT | Đơn vị tính |

+ Bảng **HoaDonBan** để lưu danh sách các đơn hàng gồm các thuộc tính sau:

|  |  |
| --- | --- |
| **Tên Thuộc tính** | **Giải thích** |
| SoHD | Số hoá đơn bán |
| MaKhach | Mã khách hàng |
| NgayHD | Ngày hoá đơn |
| DienGiai | Diễn giải |

+ Bảng **ChiTietHoaDon** để lưu chi tiết các hoá đơn bán hàng gồm các thuộc tính sau:

|  |  |
| --- | --- |
| **Tên Thuộc tính** | **Giải thích** |
| SoHD | Số hoá đơn bán |
| MaHang | Mã hàng |
| SoLuong | Số lượng bán |
| DonGia | Đơn giá |

**a. Thêm mới một trường TongTien vào bảng HoaDonBan**

ALTER TABLE HDB

ADD TONGTIEN INT

**b. Dùng kiểu dữ liệu CURSOR để cập nhập dữ liệu cho trường TongTien**

**--KHAI BAO CURSOR--**

DECLARE TONG

CURSOR FOR

SELECT SHD,MH,SL,DGIA,TONGTIEN=SL\*DGIA

FROM CTHD

**--MO CURSOR--**

OPEN TONG

**--KHAI BAO BIEN CUC BO--**

DECLARE @SHD INT,@MH NVARCHAR(10),@sl INT,@dgia INT,@TT INT

**--LAY DU LIEU TU CURSOR--**

FETCH NEXT FROM TONG

INTO @SHD,@MH,@SL,@DGIA,@TT

WHILE @@FETCH\_STATUS=0

BEGIN

UPDATE CTHD

SET TONGTIEN=@TT

WHERE (SHD=@SHD) AND (MH=@MH)

FETCH NEXT FROM TONG

INTO @SHD,@MH,@SL,@DGIA,@TT

END

**--DONG CURSOR--**

CLOSE TONG

**--HUY BO CURSOR--**

DEALLOCATE TONG

**3. Dùng kiểu dữ liệu CURSOR để đưa ra màn hình danh sách các đề tài có kết quả thực tập cao nhất**

declare @MaDT char(10),@TenDT varchar(30)

Declare @diachi varchar(100), @NS datetime

declare p cursor scroll dynamic

for

select SinhVien.\*

from SinhVien,Sv\_DeTai

Where Sinhvien.MaSV=SV\_DeTai.MaSv And Kq IN (Select Max(Kq) from SV\_detai)

open p

fetch first from p into @MaSV ,@HoTenSV ,@NS ,@diachi

while @@fetch\_status =0

begin

print @MaSV +''+ @HoTenSV +' '+convert(char(30),@NS) +' '+@diachi

fetch next from p into @MaSV ,@HoTenSV ,@NS ,@diachi

end

close p

deallocate p

## 6.3. Loại Cursor

Có 4 loại Cursor sau:

* Static: Loại Cursor tĩnh
* Keyset-driven: Loại Cursor keyset
* Dynamic: Loại Cursor động
* Fast-forward-only: Loại Cursor Fast Forward
* Static Cursor: Là loại Cursor tĩnh khi được tạo ra hầu như không liên quan gì đến việc thay đổi dữ liệu. Khi Cursor này được tạo ra những mẩu tin được copy chuyển vào bảng tạm thời lưu trong CSDL Tempdb.
* Cursor Keyset: Quan tâm đến dữ liệu có định dạng duy nhất

Một số vấn đề cần lưu ý đối với Cursor Keyset:

* + - Yêu cầu phải có khai báo Unique index trong bảng dữ liệu
    - Có thể thay đổi dữ liệu bằng cách cho phép xóa mẩu tin trên bảng dữ liệu mà Cursor đã gán
    - Có thể thêm dữ liệu khi Cursor gán vào bảng đó đã được tạo ra
    - Chỉ có Keyset lưu trữ trong Tempdb
    - Keyset cursor có thể cập nhật dữ liệu vào bảng khi Cursor tạo ra.
* Dynamic Cursor: Cursor động cho phép thêm mẩu tin, xóa mẩu tin, cập nhật dữ liệu.

**--KHAI BAO CURSOR--**

DECLARE THU

CURSOR Keyset FOR

SELECT \*

FROM HANGHOA

**--MO CURSOR--**

OPEN THU

**--KHAI BAO BIEN CUC BO--**

DECLARE @MAH CHAR(3), @TENH NVARCHAR(20), @SL INT

**--LAY DU LIEU TU CURSOR--**

FETCH NEXT FROM THU

INTO @MAH,@TENH,@SL

WHILE @@FETCH\_STATUS =0

BEGIN

PRINT '--'+@MAH+'---'+@TENH+'---'+STR(@SL)

FETCH NEXT FROM THU

INTO @MAH,@TENH,@SL

END

insert into HANGHOA

VALUES('H13','BANH',400)

**--CHUYEN VI TRI CURSOR VE VI TRI DAU TIEN--**

FETCH FIRST FROM THU

INTO @MAH,@TENH,@SL

WHILE @@FETCH\_STATUS =0

BEGIN

PRINT '--'+@MAH+'---'+@TENH+'---'+STR(@SL)

FETCH NEXT FROM THU

INTO @MAH,@TENH,@SL

END

**--DONG CURSOR--**

CLOSE THU

**--HUY BO CURSOR--**

DEALLOCATE THU

## 6.4. Thay đổi dữ liệu tại vị trí Cursor

Sử dụng các phát biểu Update hay Delete để cập nhật hoặc xóa mẩu tin tại vị trí Cursor đang sử dụng với mệnh đề WHERE CURRENT OF <Tên Cursor>

--**KHAI BAO CURSOR--**

DECLARE THUMOI CURSOR DYNAMIC FOR

SELECT MAH,TENH

FROM HANGHOA

WHERE TENH='BANH'

**--KHAI BAO BIEN CUC BO--**

DECLARE @MAH CHAR(3), @TENH NVARCHAR(20)

--MO CURSOR--

OPEN THUMOI

**--LAY DU LIEU TU CURSOR--**

FETCH NEXT FROM THUMOI

INTO @MAH,@TENH

SELECT \* FROM HANGHOA WHERE TENH='BANH'

WHILE @@FETCH\_STATUS =0

BEGIN

IF @MAH='H07'

BEGIN

UPDATE HANGHOA

SET TENH='KEO'

WHERE CURRENT OF THUMOI **--UPDATE DU LIEU TAI VI TRI CURSOR--**

END

FETCH NEXT FROM THUMOI

INTO @MAH,@TENH

END

**--THU NGHIEM DU LIEU SAU KHI DA THAY DOI DL--**

SELECT \* FROM HANGHOA WHERE TENH='KEO'

**-- DONG CURSOR--**

CLOSE THUMOI

**--HUY BO CURSOR--**

DEALLOCATE THUMOI

# CHƯƠNG 7

# QUẢN LÝ BẢO MẬT VÀ NGƯỜI DÙNG

## 7.1. Khái niệm cơ bản về bảo mật

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.

## 7.2 Các chế độ bảo mật

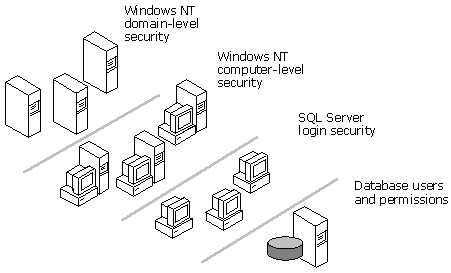
SQL Server có 2 chế độ bảo mật:

+ Windows Authentication Mode (Windows Authentication)

+ Mixed Mode (Windows Authentication and SQL Server Authentication)

**a) Windows Authentication**

Là chế độ bảo mật mà những User truy nhập SQL Server phải là những User của Windows. Khi Server đặt ở chế độ bảo mật này, những User phải là những User được Windows quản lý mới được truy nhập.



Nhìn trên hình ta thấy khi thực hiện chế độ này người sử dụng muốn khai thác SQL Server phải thông qua 4 bước xác thực (1- Domain, 2- Computer, 3- SQL Server, 4- Database).

**b) SQL Server Authentication**

Khi thiết lập ở chế độ bảo mật này, những User được quyền khai thác phải là những User do quản trị SQL Server tạo ra, mà những user của Windows không được khai thác.

Tuy nhiên, SQL Server cho phép thiết lập hai chế độ Windows Authentication Mode (Windows Authentication) và Mixed Mode (Windows Authentication and SQL Server Authentication), chế độ Mixed Mode là sự kết hợp của Windows Authentication và SQL Server Authentication, ở chế độ này cả user của Windows và SQL Server đều có thể thiết lập để truy nhập SQL Server.

## 7.3. Quản lý người dùng

Người dùng trong SQL Server được chia thành 2 mức: Người truy nhập vào SQL Server gọi là Login, người khai thác CSDL gọi là User.

Login là đối tượng được quyền truy nhập vào SQL Server, tùy theo chế độ bảo mật của SQL Server mà những login là account của Windows NT hay của SQL Server, login do Server quản lý trực tiếp.

User là đối tượng khai thác CSDL, nếu login chỉ xác định truy nhập vào SQL Server thì User là login ID tham gia khai thác CSDL, user do CSDL quản lý trực tiếp.

**7.3.1 Tạo tài khoản đăng nhập cho người dùng (Login)**

Tạo tài khoản người dùng, ta dùng thủ tục với cú pháp sau

Sp\_addlogin [@login=]<’Tên người dùng’>

[,[@Password=]<’Mật khẩu’>]

[,[@defdb=]<’Tên tệp CSDL’>]

**Ví dụ 7.1:** Tạo tài khoản người dùng có các tham số sau:

EXEC SP\_ADDLOGIN ‘HA’, ‘HA1’, ‘QL\_BanHag’

Để thay đổi mật khẩu người dùng ta dùng thủ tục với cú pháp sau:

SP\_Password[[@old=]<’Mật khẩu cũ>,]

[@New=]<‘Mật khẩu mới’>

[,[@login=]<‘Tên người dùng’>]

**Ví dụ 7.2**: Thay đổi mật khẩu người dùng

SP\_Password ‘HA1’, ‘HAI’, ‘HA’

**7.3.2 Cấp phát quyền truy cập vào CSDL**

Để cấp phát quyền truy cập cho người dùng vào CSDL ta sử dụng thủ tục có cú pháp sau:

SP\_grantdbaccess [@login=]< ‘Tên người dùng’>

[,@Name\_in\_db=]<‘Tên bí danh’>

Tham số @Name\_in\_db là bí danh tên của tài khoản người dùng thành một tên khác khi truy cập vào CSDL chỉ định.

Ví dụ 7.3: Cấp phát quyền cho người dùng

SP\_grantdbaccess ‘HA’, ‘HABD’

***Loại bỏ quyền sử dụng***

Cú pháp:

SP\_revokedbaccess [@Name\_in\_db =]<‘Tên bí danh’>

**Ví dụ 7.4**: Hủy bỏ quyền của người dùng

SP\_revokedbaccess ‘HABD’

## 7.4. Quyền người dùng

Bảng danh sách các quyền thao tác trên các đối tượng của CSDL

|  |  |
| --- | --- |
| **Quyền** | **Diễn giải** |
| SELECT | Cho phép người dùng nhìn thấy dữ liệu, nếu người dùng có quyền này thì họ chỉ có thể thực thi những phát biểu Select để truy vấn dữ liệu trên các bảng hay View được cho phép |
| INSERT | Cho phép người sử dụng thêm dữ liệu, nếu người sử dụng có quyền này họ có thể thực hiện phát biểu Insert. |
| UPDATE | Cho phép người sử dụng chỉnh sửa dữ liệu trong CSDL, với quyền Update người sử dụng có thể cập nhật dữ liệu bằng phát biểu Update |
| DELETE | Quyền này cho phép xóa dữ liệu trong CSDL với quyền Delete người sử dụng có thể xóa dữ liệu bằng phát biểu Delete. |
| REFERENCE | Cho phép người sử dụng thêm dữ liệu và bảng có khóa ngoài, nếu người dùng có quyền này họ có thể thực hiện phát biểu INSERT, trong trường hợp SQL Server thì quyền này không liên quan đến quyền truy vấn SELECT. |
| EXECUTE | Quyền này cho phép người sử dụng thực thi thủ tục trong CSDL, Với quyền này người sử dụng có thể xóa mẩu tin bằng phát biểu thực thi thủ tục |

**7.4.1 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. 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 mà anh ta đang sở hữu.
* Người sở hữu cơ sở dữ liệu cấp phát quyền thực thi các câu lệnh (như CREATE TABLE, CREATE VIEW,...) cho những người dùng khác.

**a) Cấp phát quyền cho người dùng trên các đối tượng cơ sở dữ liệu**

Chỉ có 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 GRANT trong trường hợp này có cú pháp như sau:

GRANT ALL | 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: Cấp phát tất cả các quyền cho người dùng trên đối tượng cơ sở dữ liệu được chỉ định. 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.
* Đối với cột trong bảng, khung nhìn: SELECT và UPDATE.
* Đối với thủ tục lưu trữ: EXECUTE.
* Trong các quyền  được  đề  cập  đến  ở  trên,  quyền REFERENCES  được  sử  dụng  nhằm  cho  phép  tạo khóa ngoài tham chiếu đến bảng cấp phát.

|  |  |
| --- | --- |
| 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 7.5**: Cấp phát cho người dùng có tên Ha quyền thực thi các câu lệnh SELECT, INSERT và UPDATE trên bảng Hang.

GRANT SELECT, INSERT, UPDATE

ON Hang

TO thuchanh

Cho phép người dùng Haquyền xem các thuộc tính (MaH,TenH,SoLuong)

GRANT SELECT

ON Hang(MaH,TenH,SoLuong) ON Hang) TO thuchanh

Với quyền được cấp phát như trên, người dùng ***Ha***có thể thực hiện câu lệnh sau trên bảng **Hang**

SELECT MaH, TenH, SoLuong

FROM Hang

Nhưng câu lệnh dưới đây lại không thể thực hiện được

SELECT \* FROM Hang

**Ví dụ 7.6:** Câu lệnh dưới đây cấp phát cho người dùng *Ha* các quyền SELECT, INSERT, UPDATE, DELETE VÀ REFERENCES trên bảng DIEMTHI

GRANT ALL ON DIEMTHI TO Ha

Khi ta cấp phát quyền nào đó cho một người dùng trên một đối tượng cơ sở dữ liệu, người dùng đó có thể thực thi câu lệnh được cho phép trên đối tượng đã cấp phát. Tuy nhiên, người dùng đó không có quyền cấp phát những quyền mà mình được phép cho những người sử dụng khác.Trong một số trường hợp, khi ta cấp phát quyền cho một người dùng nào đó, ta có thể cho phép người đó chuyển tiếp quyền cho người dùng khác bằng cách chỉ định tuỳ chọn WITH GRANT OPTION trong câu lệnh GRANT.

**Ví dụ 7.7**: Cho phép người dùng ***Ha***quyền xem dữ liệu trên bảng **Hang** đồng thời có thể chuyển tiếp quyền này cho người dùng khác

GRANT SELECT On Hang

WITH GRANT OPTION

**b) 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:

•    Tạo cơ sở dữ liệu: CREATE DATEBASE.

•    Tạo bảng: CREATE RULE

•    Tạo khung nhìn: CREATE VIEW

•    Tạo thủ tục lưu trữ: CREATE PROCEDURE

•    Tạo hàm: CREATE FUNCTION

•    Sao lưu cơ sở dữ liệu: BACKUP DATABASE

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

GRANT ALL | *danh\_sách\_câu\_lênh*

TO *danh\_sách\_người\_dùng*

**Ví dụ 7.8:** Để cấp phát quyền tạo bảng và khung nhìn cho người dùng có tên là *Ha* ta sử dụng câu lệnh như sau:

GRANT CREATE TABLE,CREATE VIEW TO Ha

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 tuỳ chọn WITH GRANT OPTION, tức là người dùng không thể chuyển tiếp được các quyền thực thi các câu lệnh đã được cấp phát.

**7.4.2 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.

**a) Thu hồi quyền trên đối tượng cơ sở dữ liệu**

 Cú pháp câu lệnh REVOKE sử dụng để thu hồi quyền đã cấp phát trên đối tượng cơ sở dữ liệu có cú pháp như sau:

REVOKE ALL| *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]

Câu lệnh REVOKE có thể sử dụng để thu hồi một số quyền đã cấp phát cho người dùng hoặc là thu hồi tất cả các quyền (ALL).

**Ví dụ 7.9:** Thu hồi quyền thực thi lệnh INSERT trên bảng Hang đối với người dùng *Ha*.

REVOKE INSERT ON Hang

FROM Ha

Giả sử người dùng *Ha* đã được cấp phát quyền xem dữ liệu trên các cột MaH, TenH, SoLuong của bảng Hang, câu lệnh dưới đây sẽ thu hồi quyền đã cấp phát trên cột SoLuong (chỉ cho phép xem dữ liệu trên cột MaH, TenH)

REVOKE SELECT

ON sinhvien(SoLuong) FROM Ha

Khi ta sử dụng câu lệnh REVOKE để thu hồi quyền trên một đối tượng cơ sở dữ liêu 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. Nói cách khác, nếu hai người dùng khác nhau cấp phát cùng các quyền trên cùng một đối tượng cơ sở dữ liệu cho một người dùng khác, sau đó người thứ nhất thu hồi lại quyền đã cấp phát thì những quyền mà người dùng thứ hai cấp phát vẫn có hiệu lực.

**Ví dụ 7.10:** Giả sử trong cơ sở dữ liệu ta có 3 người dùng là *A*, *B* và *C*. A và B đều có quyền sử dụng và cấp phát quyền trên bảng R. A thực hiện lệnh sau để cấp phát quyền xem dữ liệu trên bảng R cho C:

GRANT SELECT ON R TO C

và B cấp phát quyền xem và bổ sung dữ liệu trên bảng R cho C bằng câu lệnh:

GRANT SELECT, INSERT ON R TO C

Như vậy, C có quyền xem và bổ sung dữ liệu trên bảng R. Bây giờ, nếu B thực hiện lệnh:

REVOKE SELECT, INSERT ON R FROM C

Người dùng C sẽ không còn quyền bổ sung dữ liệu trên bảng R nhưng vẫn có thể xem được dữ liệu của bảng này (quyền này do A cấp cho C và vẫn còn hiệu lực).

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

**Ví dụ 7.11:** Ta cấp phát cho người dùng A trên bảng R với câu lệnh GRANT như sau:

GRANT SELECT ON R TO A

WITH GRANT OPTION

sau đó người dùng A lại cấp phát cho người dùng B quyền xem dữ liệu trên R với câu lệnh:

GRAN SELECT ON R TO B

Nếu muốn thu hồi quyền đã cấp phát cho người dùng A, ta sử dụng câu lệnh REVOKE như sau:

REVOKE SELECT ON NHANVIEN FROM A CASCADE

Câu lệnh trên sẽ đồng thời thu hồi quyền mà A đã cấp cho B và như vậy cả A và B đều không thể xem được dữ liệu trên bảng R.

**b) Thu hồi quyền thực thi các câu lênh**

  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 7.12:** Để không cho phép người dùng *Ha* thực hiện lệnh CREATE TABLE trên cơ sở dữ liệu, ta sử dụng câu lệnh:

REVOKE CREATE TABLE FROM Ha