

HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

1

TỔNG QUAN VỀ HỆ QUẢN TRỊ CSDL

3

Nguyễn Đình Hoàng Sơn
BM Hệ thống thông tin – Khoa Công nghệ thông tin
Trường Đại học Nha Trang
Email: sonndh@ntu.edu.vn - ĐĐ: 083 8705124

2

DỮ LIỆU

- ❖ Dữ liệu là các giá trị phản ánh về sự vật, hiện tượng trong thế giới khách quan và được biểu diễn vật lý dưới nhiều dạng khác nhau.
- ❖ Ví dụ: một nốt nhạc, một từ trong văn bản, một con số, hình ảnh...
- ❖ Dữ liệu về đối tượng có thể khác nhau, tùy thuộc vào ngữ cảnh.

4

CƠ SỞ DỮ LIỆU

- ❖ **CSDL (Database)** = Tập hợp dữ liệu được tổ chức có cấu trúc liên quan với nhau và được lưu trữ trong máy tính.
- ❖ CSDL được thiết kế, xây dựng cho phép người dùng **lưu trữ, truy xuất** hoặc **cập nhật** dữ liệu.

5

YÊU CẦU VỀ DỮ LIỆU TRONG CSDL

- ❖ Dữ liệu trong CSDL phải được thể hiện ở các mức độ khác nhau:
 - **Mức ngoại (External level)**
 - Mô tả một phần của CSDL mà một đối tượng / một nhóm người dùng được quyền tiếp cận
 - **Mức luận lý (Logic level)**
 - Mô tả những thông tin gì được lưu trữ trong CSDL và những mối quan hệ giữa những thông tin đó
 - **Mức vật lý (Physical level)**
 - Dữ liệu được lưu trữ như thế nào trên thiết bị lưu trữ.

6

YÊU CẦU VỀ DỮ LIỆU TRONG CSDL

7

MÔ HÌNH DỮ LIỆU

- ❖ Mô tả cách tổ chức dữ liệu bên trong CSDL.
- ❖ Mô tả mối quan hệ dữ liệu và các ràng buộc được định nghĩa trên dữ liệu đó.
 - Mô hình dữ liệu file phẳng (Flat-file data model)
 - Mô hình dữ liệu mạng (Network model)
 - Mô hình dữ liệu phân cấp (Hierarchical model)
 - **Mô hình dữ liệu quan hệ (Relational model)**
 - Mô hình dữ liệu hướng đối tượng (Object Oriented model)

8

MÔ HÌNH DỮ LIỆU QUAN HỆ

- ❖ Dữ liệu được biểu diễn dưới dạng bảng với các hàng và các cột
 - CSDL là tập hợp các bảng (còn gọi là quan hệ)
 - Mỗi hàng là một bản ghi (record), còn được gọi là bộ (tuple)
 - Mỗi cột là một thuộc tính, còn được gọi là trường (field)
- ❖ Dữ liệu trong hai bảng liên hệ với nhau thông qua các cột chung.
- ❖ Có các toán tử để thao tác trên các hàng của bảng.



9

MÔ HÌNH DỮ LIỆU QUAN HỆ

Ví dụ: mô hình dữ liệu quan hệ trong CSDL Northwind
gồm 3 bảng: Customer, Order, Employee

Bảng Customer						
Customer ID	Company Name	Contact First Name	Contact Last Name	Job Title	City	State
6	Company F	Francisco	Perez-Olivero	Purchasing Manager	Milwaukee	WI
26	Company Z	Kun	Liu	Accounting Assistant	Miami	FL

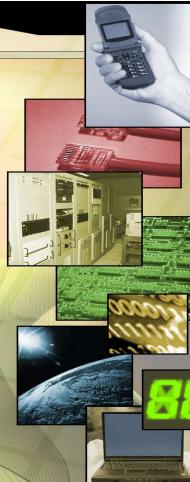
Bảng Order						
Order ID	Customer ID	Employee ID	Order Date	Shipped Date	Shipping Fee	
51	26	9	4/5/2006	4/5/2006	\$40.00	
56	6	2	4/3/2006	4/3/2006	\$ 0.00	
79	6	2	6/23/2006	6/23/2006	\$ 0.00	

Bảng Employee			
Employee ID	First Name	Last Name	Title
2	Andrew	Garcia	Vice President, Sales
3	Steven	Thorp	Sales Manager
9	Anne	Hollingsworth	Sales Representative

10

QUẢN LÝ DỮ LIỆU

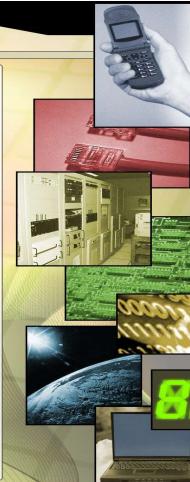
- ❖ Quản lý lượng lớn dữ liệu bao gồm cả việc lưu trữ dữ liệu và cơ chế thao tác trên các dữ liệu đó.
- ❖ Hai hệ thống quản lý dữ liệu khác nhau:
 - Hệ thống quản lý dựa trên tập tin (File-based systems)
 - Hệ thống CSDL (Database systems)



11

HỆ QUẢN TRỊ CSDL

- ❖ Là một **hệ thống phần mềm** cung cấp các công cụ để xây dựng, khai thác và quản lý CSDL.
- **Xây dựng** (Sử dụng ngôn ngữ DDL): Định nghĩa cấu trúc CSDL, lưu trữ dữ liệu
- **Khai thác** (Sử dụng ngôn ngữ DML): Truy vấn dữ liệu, cập nhật dữ liệu
- **Quản lý**:
 - Quản lý an toàn và bảo mật
 - Điều khiển truy xuất đồng thời
 - Khôi phục khi có sự cố
 - ...



12

LÝ DO SỬ DỤNG HQTCSDL

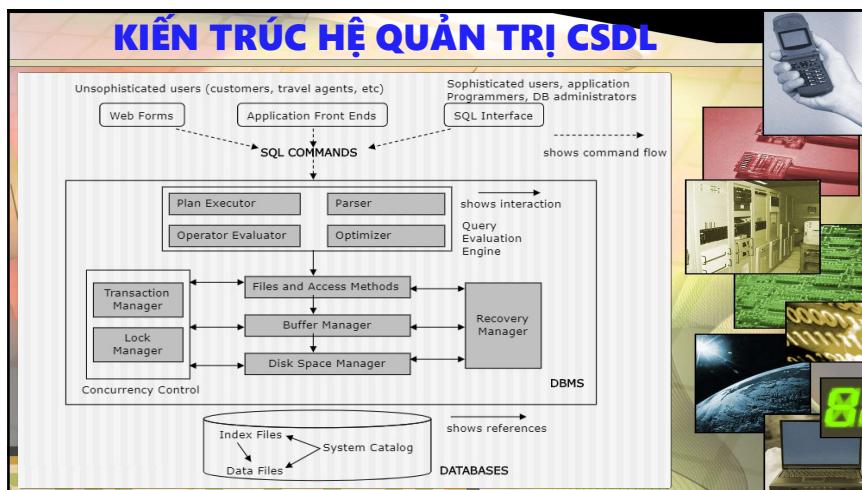
- ❖ Đọc lập dữ liệu
- ❖ Truy cập dữ liệu hiệu quả
- ❖ Toàn vẹn dữ liệu
- ❖ An ninh dữ liệu
- ❖ Truy xuất đồng thời
- ❖ Khôi phục sau sự cố
- ❖ Giảm thời gian phát triển ứng dụng

13

LỊCH SỬ PHÁT TRIỂN CỦA HQTCSDL

Decade of RDBMS			
1960s	1970s	1980s - 1990s	2000s
Mô hình mạng CODASYL Mô hình phân cấp IMS	Mô hình quan hệ QUEL SEQUEL	Mô hình đối tượng SQL	No SQL Database
SABRE system	Ingres PostgreSQL dBASE Ingres Corp Sybase MS SQL Server	System R Non-Stop SQL SQL/DS DB2 Allbase Oracle	Prototypes for ODBMS
			MongoDB, Oracle NoSQL Database, Apache Cassandra, ...

14



15



16

KIẾN TRÚC HQTCSQL – GIAO DIỆN LẬP TRÌNH

❖ HQTCSQL cung cấp giao diện lập trình dễ sử dụng và ngôn ngữ tương tác với dữ liệu được lưu trữ bên trong CSDL

- **Giao diện:** tương tác dòng lệnh (command line), đồ họa (GUI)
- **Ngôn ngữ:** SQL, SQL mở rộng (tùy thuộc HQTCSQL)

Ví dụ: MS SQL Server cung cấp ngôn ngữ Transaction SQL (T-SQL)



17

KIẾN TRÚC HQTCSQL – GIAO DIỆN LẬP TRÌNH

❖ Các loại ngôn ngữ truy vấn dữ liệu (SQL)

- **Ngôn ngữ định nghĩa dữ liệu (Data Definition Language – DDL):** giúp người dùng ra lệnh cho HQTCSQL tạo ra các cấu trúc dữ liệu của CSDL (cách tổ chức dữ liệu và mối liên hệ giữa các đối tượng dữ liệu).
- **Ngôn ngữ xử lý dữ liệu (Data Manipulation Language – DML):** giúp người dùng thao tác dữ liệu (xem, thêm, xóa, sửa) trong CSDL.
- **Ngôn ngữ điều khiển dữ liệu (Data Control Language – DCL):** thiết lập quyền truy cập của người dùng trên các đối tượng CSDL.
- **Ngôn ngữ kiểm soát giao dịch (Transaction control language – TCL):** xử lý các thay đổi có ảnh hưởng đến dữ liệu trong CSDL.

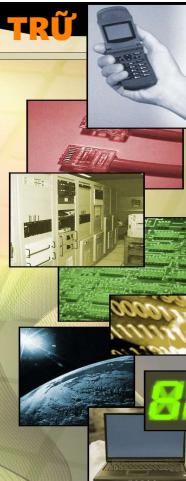


18

KIẾN TRÚC HQTCSQL – BỘ QUẢN LÝ LƯU TRỮ

❖ Bộ quản lý lưu trữ có nhiệm vụ lưu trữ, rút trích và cập nhật dữ liệu vào cơ sở dữ liệu.

- *Điều khiển tương tranh – xử lý truy xuất đồng thời.*
- *Quản lý file và bộ nhớ lưu trữ*
- *Quản lý khôi phục sau sự cố*

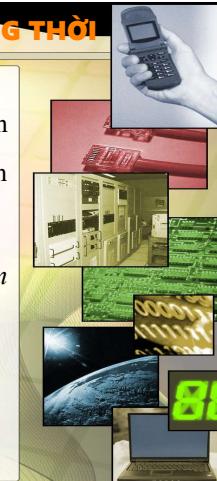


19

KIẾN TRÚC HQTCSQL – XỬ LÝ TRUY XUẤT ĐỒNG THỜI

❖ **Mục tiêu:** đảm bảo các xử lý có thể được thực hiện đồng thời mà làm không làm cho dữ liệu bị mất tính nhất quán (vi phạm các ràng buộc toàn vẹn).

❖ **Các thành phần con:** quản lý giao tác (*Transaction manager*) và quản lý khóa (*Lock manager*)

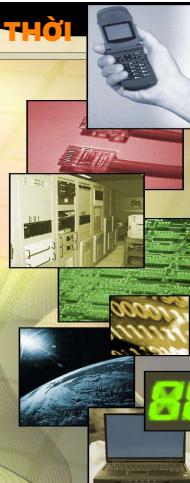


20

KIẾN TRÚC HQTCSQL – XỬ LÝ TRUY XUẤT ĐỒNG THỜI

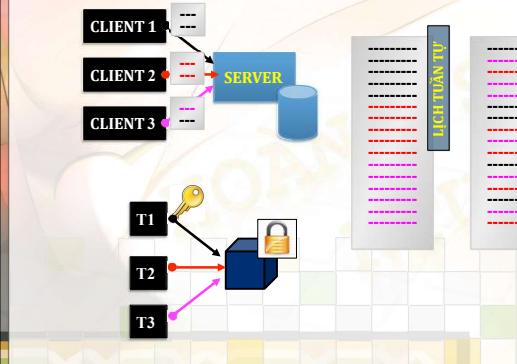
❖ Phương pháp:

- Sử dụng khái niệm giao tác (*transaction*) để biểu diễn một đơn vị xử lý, một giao tác bao gồm các hành động mà được thực hiện toàn bộ hoặc không có hành động nào được thực hiện.
- Bộ lập lịch (*scheduler*) có nhiệm vụ lập 1 lịch thực hiện từ n giao tác không tách biệt về thời gian sao cho kết quả không vi phạm tính nhất quán của CSDL.
- Sử dụng cơ chế khóa (*lock*) để khóa các đơn vị dữ liệu nào đó khi cần → ngăn 2 giao tác cùng thao tác lên 1 đơn vị dữ liệu tại cùng 1 thời điểm → Hỗ trợ để lập lịch.



21

KIẾN TRÚC HQTCSQL – XỬ LÝ TRUY XUẤT ĐỒNG THỜI

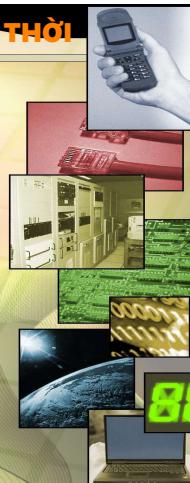


22

KIẾN TRÚC HQTCSQL – XỬ LÝ TRUY XUẤT ĐỒNG THỜI

❖ Vấn đề deadlock

- Do sử dụng cơ chế khóa nên các giao tác sẽ phải chờ khi cần truy xuất 1 đơn vị dữ liệu đang bị khóa.
- Tình huống chờ vĩnh viễn mà vẫn không được truy xuất đơn vị dữ liệu bị khóa gọi là **Deadlock** (khoá chết)
 - Các giao tác chờ đợi lẫn nhau để được cấp phát tài nguyên và không giao tác nào có thể hoàn tất.
- Thành phần quản lý giao tác sẽ phải can thiệp vào:
 - Hoặc hủy bỏ một trong các giao tác gây deadlock
 - Hoặc ngăn chặn từ trước để không bao giờ xảy ra deadlock



23

KIẾN TRÚC HQTCSQL – AN NINH VÀ BẢO MẬT

- **Bảo mật dữ liệu:** HQTCSQL hỗ trợ các tính năng về chứng thực, phân quyền giúp kiểm soát tốt những người dùng hợp pháp của hệ thống.
- **An ninh dữ liệu:** HQTCSQL hỗ trợ các phương pháp mã hóa dữ liệu để ngăn chặn các tấn công của những đối tượng tin tặc (đánh cắp thông tin trên đường truyền, đánh cắp nội dung CSDL).



24

KIẾN TRÚC HQTCSQL – QUẢN LÝ FILE & BỘ NHỚ

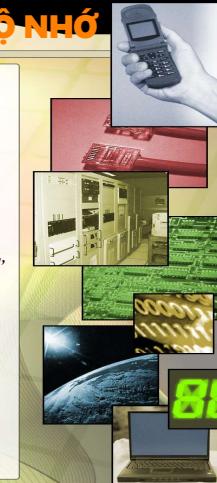
- ❖ **Files and access methods:** tổ chức dữ liệu, hỗ trợ truy cập nhanh đến tập dữ liệu mong muốn.
- ❖ **Buffer Manager:** đọc dữ liệu vào bộ nhớ để xử lý, và viết vào đĩa để lưu trữ lâu dài theo yêu cầu của chương trình.
- ❖ **Disk Space Manager:** tính toán và quản lý khoảng trống trên đĩa.



25

KIẾN TRÚC HQTCSQL – QUẢN LÝ FILE & BỘ NHỚ

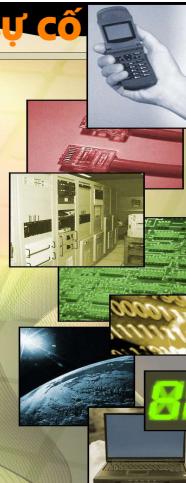
- ❖ **Các khái niệm**
 - Tập tin dữ liệu
 - Từ điển dữ liệu
 - Lưu trữ các metadata (siêu dữ liệu) về cấu trúc của CSDL, đặc biệt là lược đồ của CSDL
 - Chỉ mục
 - Giúp cho việc tìm kiếm dữ liệu được nhanh chóng



26

KIẾN TRÚC HQTCSQL – KHÔI PHỤC SAU SỰ CỐ

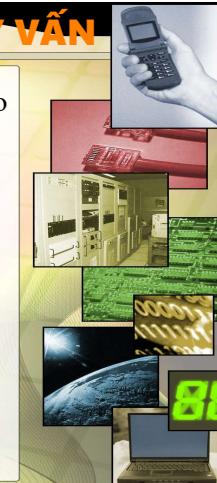
- ❖ **Mục tiêu:** đảm bảo sự tồn thât, sai sót về mặt dữ liệu là ít nhất có thể.
- ❖ **Cách tiếp cận:** mọi thay đổi trong CSDL phải được ghi nhận lại trong nhật ký (Log)
- ❖ **Các thành phần hỗ trợ quá trình khôi phục sau sự cố:**
 - *Quản lý nhật ký (Log manager):* đảm bảo ghi nhận đầy đủ và chính xác mọi thay đổi trên CSDL vào nhật ký.
 - *Quản lý khôi phục sự cố (Recovery Manager):* dựa vào nhật ký để phục hồi lại CSDL về trạng thái nhất quán trước đó (trạng thái thỏa mãn tất cả ràng buộc toàn vẹn của CSDL)



27

KIẾN TRÚC HQTCSQL – XỬ LÝ TRUY VẤN

- ❖ Biểu diễn câu truy vấn ở dạng ngôn ngữ cấp cao (SQL) và thực hiện câu truy vấn có hiệu quả.
- ❖ Các giai đoạn xử lý truy vấn
 - Phân tích (*parser*)
 - Tối ưu hóa câu hỏi (*query optimizer*)
 - Lập kế hoạch thực hiện (*plan executor*)
 - Thực hiện tính toán (*operator evaluator*)



28

PHÂN LOẠI

- ❖ Theo mô hình dữ liệu
 - Phân cấp
 - Mạng
 - Quan hệ
 - Đối tượng
- ❖ Theo kiến trúc tính toán
 - Tập trung (Centralized database system)
 - Khách / chủ (Client server database system)
 - Phân tán (Distributed database system)



29

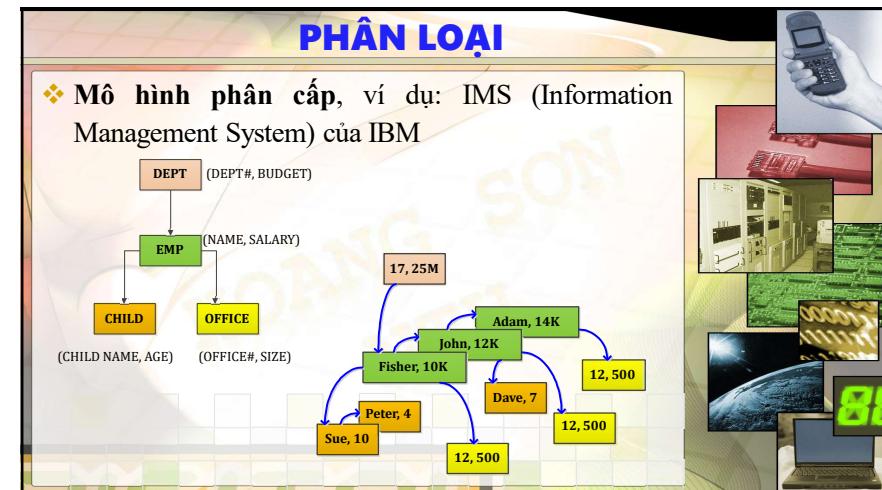
PHÂN LOẠI

- ❖ Mô hình phân cấp, ví dụ: IMS (Information Management System) của IBM

Diagram illustrating the hierarchical structure of the IMS system:

```

graph TD
    DEPT[DEPT (DEPT#, BUDGET)] --> EMP[EMP (NAME, SALARY)]
    EMP --> CHILD[CHILD (CHILD NAME, AGE)]
    EMP --> OFFICE[OFFICE (OFFICE#, SIZE)]
    CHILD --> Fisher[Fisher, 10K]
    CHILD --> John[John, 12K]
    CHILD --> Sue[Sue, 10]
    CHILD --> Peter[Peter, 4]
    OFFICE --> Adam[Adam, 14K]
    OFFICE --> Dave[Dave, 7]
    OFFICE --> 17_25M[17, 25M]
    Fisher --> 12_500[12, 500]
    Fisher --> 12_500[12, 500]
    John --> 12_500[12, 500]
    John --> 12_500[12, 500]
    Sue --> 12_500[12, 500]
    Peter --> 12_500[12, 500]
    Adam --> 12_500[12, 500]
    Adam --> 12_500[12, 500]
    Dave --> 12_500[12, 500]
    
```



30

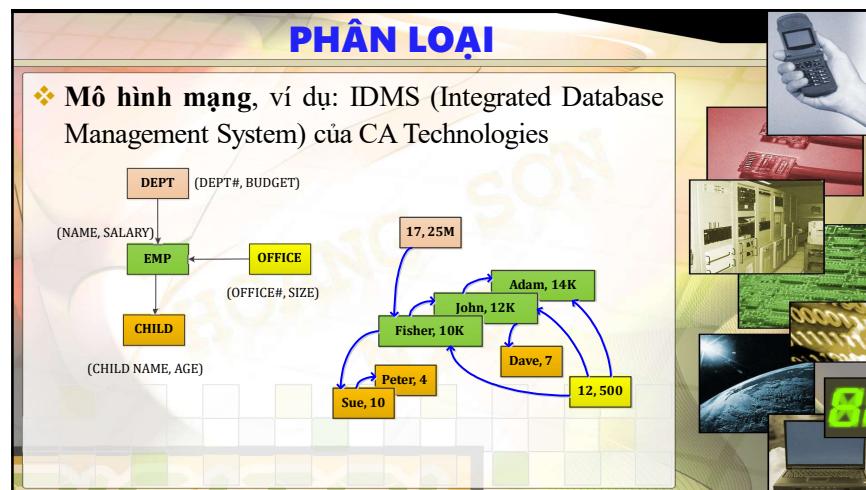
PHÂN LOẠI

- ❖ Mô hình mạng, ví dụ: IDMS (Integrated Database Management System) của CA Technologies

Diagram illustrating the networked structure of the IDMS system:

```

graph TD
    DEPT[DEPT (DEPT#, BUDGET)] --> EMP[EMP (NAME, SALARY)]
    EMP --> CHILD[CHILD (CHILD NAME, AGE)]
    EMP --> OFFICE[OFFICE (OFFICE#, SIZE)]
    CHILD --> Fisher[Fisher, 10K]
    CHILD --> John[John, 12K]
    CHILD --> Sue[Sue, 10]
    CHILD --> Peter[Peter, 4]
    OFFICE --> Adam[Adam, 14K]
    OFFICE --> Dave[Dave, 7]
    OFFICE --> 17_25M[17, 25M]
    Fisher --> 12_500[12, 500]
    Fisher --> 12_500[12, 500]
    John --> 12_500[12, 500]
    John --> 12_500[12, 500]
    Sue --> 12_500[12, 500]
    Peter --> 12_500[12, 500]
    Adam --> 12_500[12, 500]
    Adam --> 12_500[12, 500]
    Dave --> 12_500[12, 500]
    
```



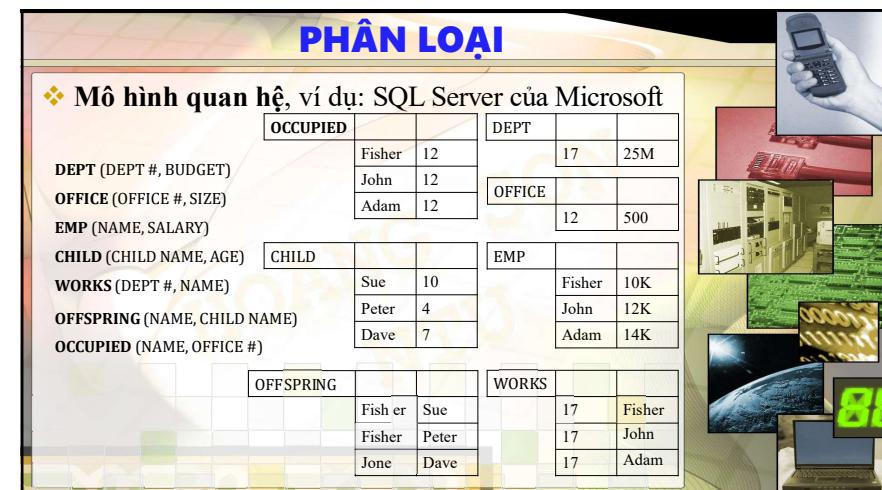
31

PHÂN LOẠI

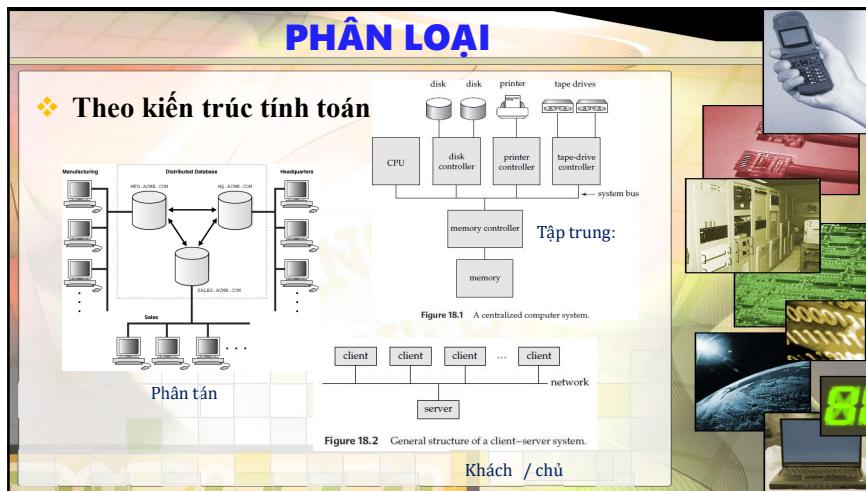
- ❖ Mô hình quan hệ, ví dụ: SQL Server của Microsoft

OCCUPIED			DEPT		
DEPT (DEPT #, BUDGET)	Fisher	12		17	25M
OFFICE (OFFICE #, SIZE)	John	12			
EMP (NAME, SALARY)	Adam	12			
CHILD (CHILD NAME, AGE)			OFFICE		
WORKS (DEPT #, NAME)	Sue	10			
OFFSPRING (NAME, CHILD NAME)	Fisher	10K			
OCCUPIED (NAME, OFFICE #)	John	12K			
	Dave	7			

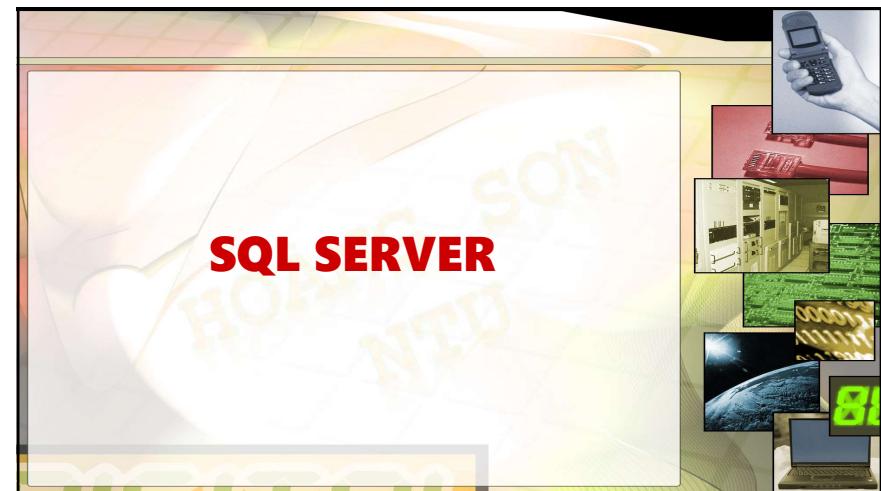
OFFSPRING			WORKS		
	Fisher	Sue		17	Fisher
	Fisher	Peter		17	John
	Jone	Dave		17	Adam



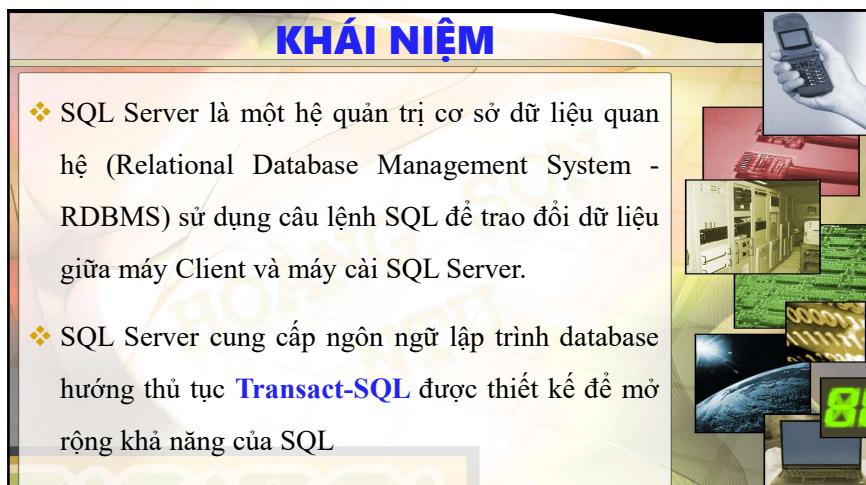
32



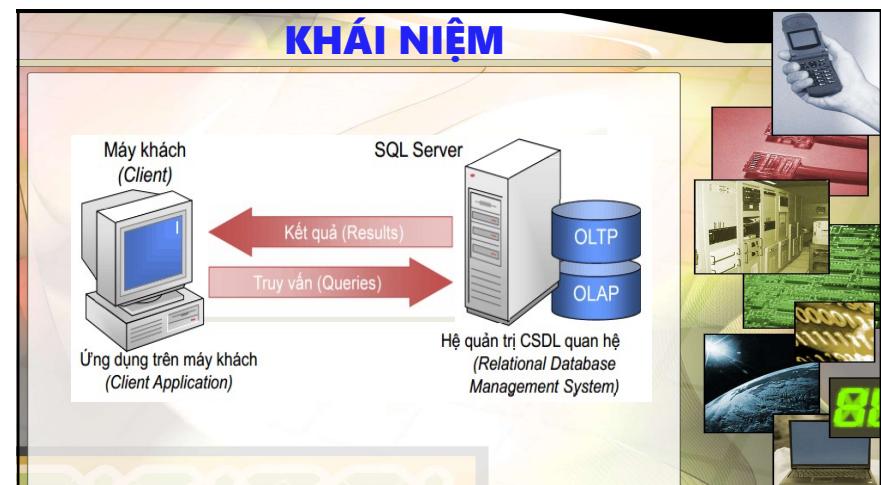
33



34



35

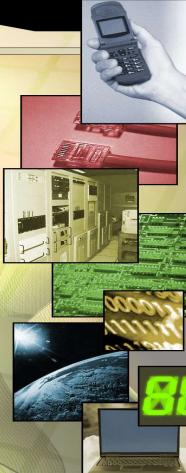


36

KHÁI NIỆM

❖ OLTP (online transaction processing): **hệ thống xử lý giao dịch trực tuyến.**

- Đặc trưng bởi một số lượng lớn giao dịch (**insert, update, delete**) trong một thời gian ngắn.
- Truy vấn OLTP **đơn giản và ngắn hơn** và do đó đòi hỏi **ít thời gian hơn trong quá trình xử lý** và cũng cần **ít không gian hơn**.
- Được thiết kế nhằm mục đích **thu thập dữ liệu**, phải duy trì **ràng buộc toàn vẹn** dữ liệu.

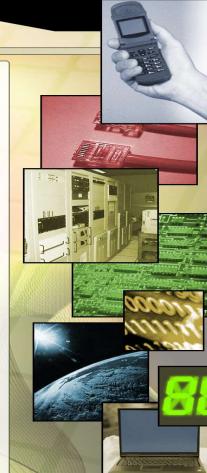


37

KHÁI NIỆM

❖ OLAP (online analytical processing): **hệ thống xử lý phân tích trực tuyến.**

- Tổ chức và thống kê **khối lượng lớn dữ liệu** được dùng để **phân tích, báo cáo, hỗ trợ ra quyết định**.
- Cho phép người dùng thực hiện **các truy vấn phức tạp** để trích xuất dữ liệu đa chiều.
- Giao dịch trong OLAP **dài** và do đó mất **nhiều thời gian hơn** để xử lý và yêu cầu không gian lớn.
- Các giao dịch trong OLAP **ít thường xuyên** hơn so với OLTP.



38

PHIÊN BẢN

❖ **Standard**: phiên bản chuẩn, tính năng cơ bản đầy đủ, chạy tốt trên hệ thống lên tới 4 CPU và 2 GB RAM

❖ **Developer**: dành cho nhà phát triển, chứa đầy đủ tính năng nhưng chỉ phân quyền cho một người duy nhất.

❖ **Enterprise**: phiên bản cao cấp, có đầy đủ các tính năng

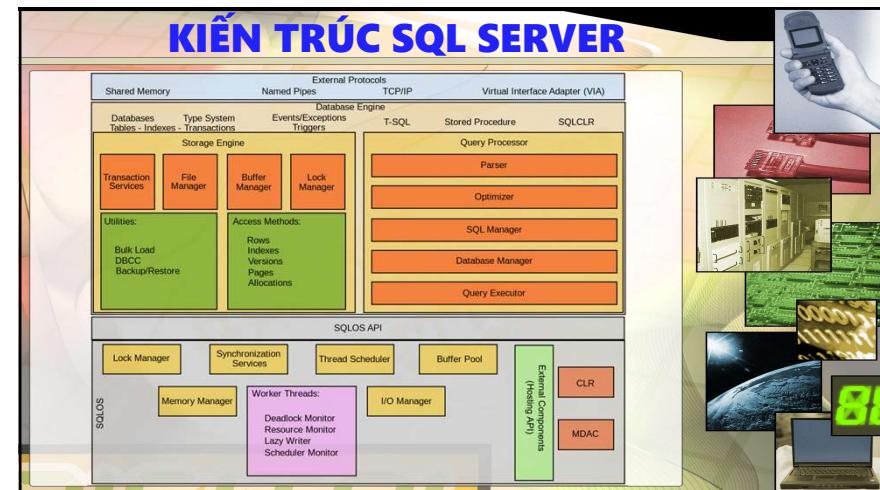
❖ **Web**: dành cho các ứng dụng web

❖ **Express**: là bản cơ bản, sử dụng tối đa 1 CPU và 1GB RAM, dung lượng lưu trữ tối đa là 10GB.



39

KIẾN TRÚC SQL SERVER



40

KIẾN TRÚC SQL SERVER

- ❖ **Protocols:** có 4 giao thức
 - **Shared memory:** Dành cho các kết nối cục bộ và mục đích khắc phục sự cố.
 - **Named pipes:** Dành cho những kết nối trong mạng LAN.
 - **TCP/IP:** Dành cho các kết nối với mạng WAN.
 - **VIA-Virtual Interface Adapter:** Yêu cầu phần cứng đặc biệt được thiết lập bởi nhà cung cấp và cũng không được hỗ trợ từ bản SQL 2012.



41

KIẾN TRÚC SQL SERVER

- ❖ **Database Engine:** thành phần chính của SQL Server quản lý các đối tượng vật lý và logic của CSDL.
- ❖ **Query Processor (Relational Engine):** chứa bộ phân tích Query, tối ưu hóa Query và bộ thực thi Query.
 - **Query Parser (Command Parser) và Compiler (Translator):** chịu trách nhiệm kiểm tra cú pháp của truy vấn và chuyển đổi truy vấn sang ngôn ngữ của máy.



42

KIẾN TRÚC SQL SERVER

- ❖ **Query Processor (Relational Engine) - tt**
 - **Query Optimizer:** chuẩn bị output là *Execution Plan* bằng cách lấy input là truy vấn, các thông kê và cây *Algebrizer* (tiến trình xác minh các đối tượng CSDL).
 - **Execution Plan:** tương tự như bản đồ chỉ đường, chứa thứ tự các bước thực hiện như là một phần của việc thực hiện các truy vấn.
 - **Query Executor:** thực hiện truy vấn từng bước một dưới sự chỉ dẫn của Execution Plan và cũng là nơi Storage Engine sẽ được liên lạc.



43

KIẾN TRÚC SQL SERVER

- ❖ **Storage Engine:** Chịu trách nhiệm lưu trữ, truy xuất dữ liệu trong hệ thống lưu trữ, thao tác dữ liệu, khóa và quản lý các transaction.
- ❖ **SQL OS:** Cung cấp các dịch vụ hệ điều hành khác nhau, ví dụ như hoạt động quản lý bộ nhớ với buffer pool, log buffer, phát hiện deadlock (khóa chết) bằng cách sử dụng cấu trúc block và lock.



44

KIẾN TRÚC SQL SERVER

SQL OS

- Checkpoint:** là một tiến trình nội bộ, ghi tất cả các trang đã sửa đổi (Dirty Page) từ Buffer Cache vào ổ đĩa vật lý. Ngoài ra, nó cũng ghi các bản log từ Log Buffer vào file vật lý.
- Lazy Writer:** đẩy các Dirty Page và ổ cứng để giải phóng bộ nhớ trong Buffer Pool. Điều này xảy ra khi SQL Server đang bị thiếu bộ nhớ.

45

DỊCH VỤ TRONG SQL SERVER

Service	Tasks
MSSQLServer Service	Data Management, Transaction and Query Processing, Data Integrity
SQLServerAgent Service	Jobs, Alerts, Operators
Microsoft Distributed Transaction Coordinator	Distributed Transaction Management
Microsoft Search	Full-Text Catalogs, Full-Text Indexes

46

DỊCH VỤ TRONG SQL SERVER

MSSQL Server Services

- Cấp phát tài nguyên máy tính cho nhiều user đồng thời.
- Quản lý dữ liệu.
- Xử lý truy vấn và giao dịch
- Bảo đảm toàn vẹn dữ liệu.

47

DỊCH VỤ TRONG SQL SERVER

SQLServerAgent Services

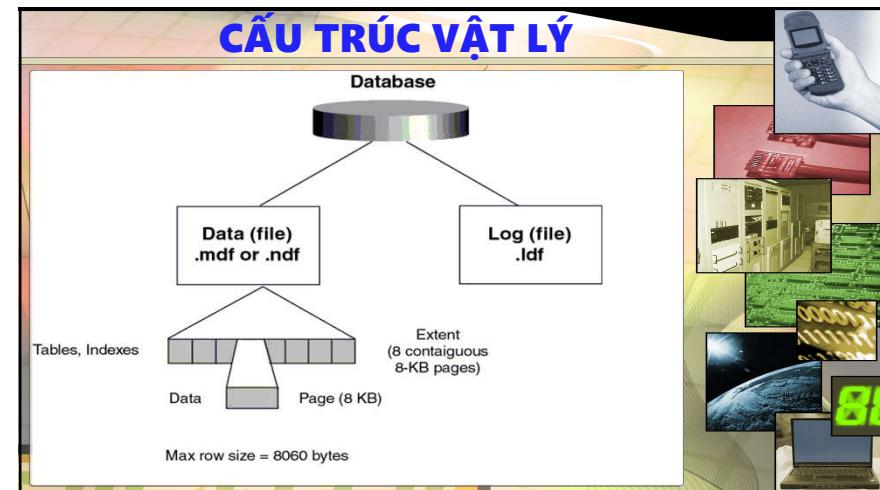
- Cảnh báo về trạng thái của một process, ví dụ như một công việc nào đó được hoàn thành hoặc xảy ra lỗi.
- Tạo ra một công việc mới và lập lịch để tự động hóa các nhiệm vụ như Backup dữ liệu, Replication, ...

48

DỊCH VỤ TRONG SQL SERVER

- Microsoft Distributed Transaction Coordinator (MS DTC)**
 - Cho phép các clients làm việc với nhiều nguồn dữ liệu khác nhau trong một transaction.
- Microsoft Search**
 - Là một full-text engine cho phép đánh chỉ mục và truy vấn cho dữ liệu văn bản không cấu trúc

49

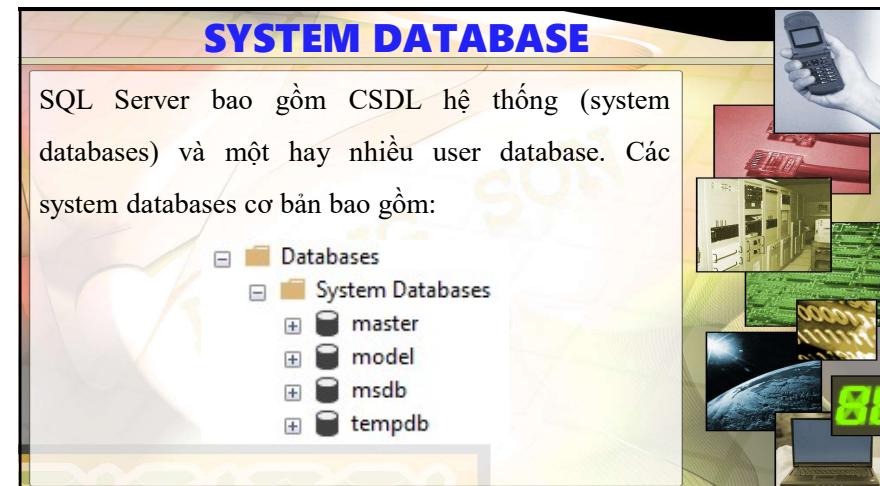


50

CẤU TRÚC VẬT LÝ

- Primary data file (.mdf):** 1 file chính chứa data và những system tables.
- Secondary data file (.ndf):** 1 hay nhiều file phụ chỉ sử dụng khi database được phân chia để chứa trên nhiều đĩa.
- Transaction log file (.ldf):** 1 file ghi lại tất cả những thay đổi diễn ra trong một database và chứa đầy đủ thông tin để có thể rollback hay rollforward.

51



52

SYSTEM DATABASE

- master:** là CSDL kiểm soát tất cả các hoạt động trên SQL Server, chứa thông tin về hệ thống; các tài khoản đăng nhập, cấu hình hệ thống, thông tin về các CSDL đã tạo, các thủ tục hệ thống thực hiện các tác vụ quản trị hệ thống, các thủ tục của người dùng tạo thêm...
- model:** CSDL mẫu để tạo ra các CSDL người dùng
- msdb:** được sử dụng cho SQL Server Agent để lập lịch các công việc và các cảnh báo.
- tempdb:** lưu trữ các thông tin tạm thời của các hoạt động trên SQL



53

ĐỐI TƯỢNG CSDL

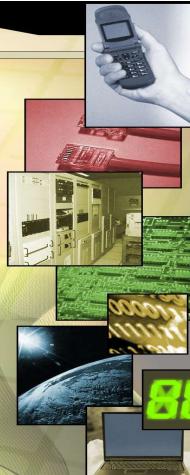
Đối tượng	Mô tả
Table	Đối tượng lưu trữ dữ liệu của CSDL
Data type	Kiểu dữ liệu
Constraint	Các qui tắc để kiểm tra ràng buộc dữ liệu
Default	Các giá trị mặc định nếu giá trị của một field không được nhập vào
Rule	Các thông tin mà định nghĩa các giá trị hợp lệ trong một field
Index	Là một cấu trúc lưu trữ nhằm truy xuất nhanh dữ liệu



54

ĐỐI TƯỢNG CSDL

Đối tượng	Mô tả
View	Là bảng ảo, lưu trữ dữ liệu từ các table hoặc các view khác thông qua câu lệnh SQL SELECT
User-defined function	Là các hàm do programmer định nghĩa
Stored procedure	Là các thủ tục do programmer định nghĩa
Trigger	Là loại thủ tục lưu trữ đặc biệt, được thực thi khi dữ liệu trong bảng thay đổi

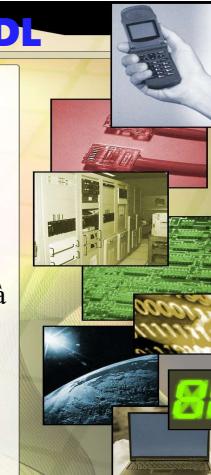


55

THAM CHIẾU ĐỐI TƯỢNG CSDL

- Tên đầy đủ:
`[server].[database].[schema.]]object`
- Tên ngắn gọn
Được hiểu là server default, database hiện hành và schema hiện hành (dbo)

Ví dụ: `SELECT * FROM dbo.nhanvien`



56

THỦ TỤC TRUY VẤN ĐỔI TƯỢNG CSDL

System Stored Procedure	Ứng dụng
<code>sp_help ['object']</code>	Cung cấp thông tin về một database object (table, view...) hay một data type.
<code>sp_helpdb ['database']</code>	Cung cấp thông tin về một database cụ thể nào đó.
<code>sp_monitor</code>	Cho biết độ "bận rộn" của SQL Server
<code>sp_spaceused ['object']</code>	Cung cấp thông tin về các khoảng trống đã được sử dụng cho một object nào đó
<code>sp_who ['login']</code>	Cho biết thông tin về một SQL Server user



57

THỦ TỤC TRUY VẤN ĐỔI TƯỢNG CSDL

Ví dụ: Giả sử trong SQL Server có CSDL QLSV.

Thực thi lệnh `sp_helpdb 'QLSV'` ta có kết quả

```
Connect...  HS SQL Server 14.0.2027.2 - son
          + System Databases
          + Database Snapshots
          + DMVConfiguration
          + DMVDiagnostic
          + DMVQueue
          + mydb
          + QLSV
          + Security

Results  Messages
+-----+-----+-----+-----+-----+-----+-----+-----+
| name | db_size | owner | dbid | created | status | compatibility_level |
+-----+-----+-----+-----+-----+-----+-----+
| QLSV | 16.00 MB | son | 9 | Sep 11 2019 | Status=ONLINE, Updatedatability=READ_WRITE, UserAcc... | 140 |
+-----+-----+-----+-----+-----+-----+-----+
| name | file | filename | filegroup | size | maxsize | growth | usage |
|      | id   |           |            |     |       |       |        |
+-----+-----+-----+-----+-----+-----+-----+-----+
| QLSV | 1 | D:\MSSQLServer2017\Data\QLSV\mdf | PRIMARY | 6192 KB | Unlimited | 65536 KB | data only |
| QLSV | 2 | D:\MSSQLServer2017\Log\QLSV\log.ldf | NULL | 8192 KB | 2147483648 KB | 65536 KB | log only |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

58

TRUY VẤN THÔNG TIN VỀ DATABASE

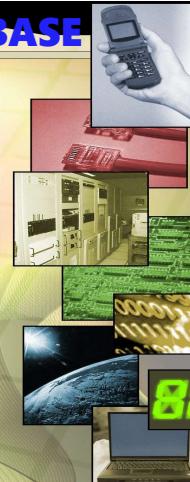
❖ Hiển thị danh sách database

```
SELECT * FROM sys.databases;
```

❖ Đếm số lượng bảng trong Database

```
USE MyDatabase;
```

```
SELECT COUNT(*) FROM
    INFORMATION_SCHEMA.TABLES
    WHERE TABLE_TYPE = 'BASE TABLE';
```



59

TRUY VẤN THÔNG TIN VỀ DATABASE

❖ Lấy danh sách các Stored Procedure

```
SELECT * FROM sys.objects WHERE type = 'P';
```

hoặc

```
SELECT * FROM INFORMATION_SCHEMA.ROUTINES
WHERE ROUTINE_TYPE = 'PROCEDURE';
```

❖ Tìm kiếm các Stored Procedure

```
SELECT p.name FROM sys.sql_modules AS m
INNER JOIN sys.procedures AS p
    ON m.object_id = p.object_id
WHERE definition LIKE '%keyword%';
```



60

NGÔN NGỮ ĐỊNH NGHĨA DỮ LIỆU DATA DEFINITION LANGUAGE - DDL



61

DDL

- ❖ **CREATE:** Tạo mới đối tượng CSDL
- ❖ **ALTER:** Thay đổi định nghĩa của đối tượng CSDL
- ❖ **DROP:** Xoá đối tượng CSDL



62

TẠO MỚI CSDL

- ❖ Tao database mặc định

```
CREATE DATABASE Tên_CSDL;
```

Ví dụ:

```
IF EXISTS (SELECT * FROM sys.DATABASES
            WHERE NAME = N'QLBanHang' )
    --Xóa Database
    DROP DATABASE QLBanHang;
GO
    --Tạo database mới
    CREATE DATABASE QLBanHang;
```



63

TẠO MỚI CSDL

- ❖ Tao database theo yêu cầu

```
CREATE DATABASE Tên_CSDL
ON PRIMARY
( Name = tên_file_logic
  , FileName = 'tên_file_vật_lý'
  , Size = kích_thước_ban_dầu [KB|MB|GB|TB]
  , MaxSize = kích_thước_tối_da [KB|MB|GB|TB]
  , FileGrowth=kích_thước_tăng_trưởng
)
LOG ON      -- Tạo tập tin log
( Name = tên_file_logic
  , FileName = 'tên_file_vật_lý'
  , Size = kích_thước_ban_dầu [KB|MB|GB|TB]
  , MaxSize = kích_thước_tối_da [KB|MB|GB|TB]
  , FileGrowth=kích_thước_tăng_trưởng
);
```



64

TẠO MỚI CSDL

Ví dụ:

```
create database QuanLyBanHang
on primary
(
    name = QuanLyBanHang_Data
    ,filename = 'c:\QuanLyBanHang.mdf'
    ,size = 200
    ,maxsize = Unlimited
    ,filegrowth = 10%      -- Tăng 10%
)
log on
(
    name = QuanLyBanHang_Log
    ,filename = 'c:\QuanLyBanHang.ldf'
    ,size = 100
    ,maxsize = 1GB
    ,filegrowth = 10        -- Tăng 10MB
);
```

65

TẠO MỚI CSDL

Ví dụ:

```
create database Products ON
(
    name = prods_dat,
    filename = 'D:\BTSQL\prods.mdf',
    size = 5,
    maxsize = 500,
    filegrowth = 10
);
```

Lưu ý: Nếu không chỉ định transaction log file thì SQL Server sẽ tự động tạo ra 1 log file với kích thước ban đầu là 1MB

66

SỬ DỤNG CSDL

- ❖ Sử dụng (mở) CSDL để làm việc

USE *Tên_CSDL*;

67

SỬA ĐỔI CSDL

ALTER DATABASE *Tên_CSDL* **MODIFY FILE**

```
(  
    Name      = tên_file_logic  
    [, NewName = 'tên_file_logic_sửa_đổi' ]  
    [, FileName = 'tên_file_vật_lý_sửa_đổi' ]  
    [, Size    = kích_thước_ban_dầu_sửa_đổi ]  
    [, MaxSize = kích_thước_tối_da_sửa_đổi ]  
    [, FileGrowth = kích_thước_tăng_trưởng_sửa_đổi ]  
)
```

68

SỬA ĐỔI CSDL

Lưu ý: Trong trường hợp sửa đổi file vật lý (đổi tên hoặc di chuyển file) thì cần làm theo các bước sau:

- ❖ Sửa đổi thông số của CSDL như trên
`ALTER DATABASE Tên_CSDL MODIFY FILE (...);`
- ❖ Ngắt kết nối, thiết lập offline CSDL
`ALTER DATABASE Tên_CSDL`
--đóng hết các connection tới database
`SET SINGLE_USER WITH ROLLBACK IMMEDIATE;`
--đặt database thành offline
`ALTER DATABASE Tên_CSDL SET OFFLINE;`



69

SỬA ĐỔI CSDL

❖ Sau khi sửa đổi CSDL (đổi tên, di chuyển file) thì kết nối CSDL với người dùng trở lại

- đặt database trở lại online
`ALTER DATABASE Tên_CSDL SET ONLINE;`
- đặt trở lại chế độ nhiều người dùng
`ALTER DATABASE Tên_CSDL SET MULTI_USER;`



70

SỬA ĐỔI CSDL

Ví dụ: Di chuyển CSDL file (data file hoặc log file) sang vị trí mới.



71

THU HẸP CSDL

❖ Thu hẹp toàn bộ CSDL

DBCC SHRINKDATABASE

```
(Tên_CSDL | id_CSDL | 0      -- 0 là CSDL mặc định
 [ ,ty_lệ ]                  -- % khoảng trống còn lại
 [ , { NOTRUNCATE | TRUNCATEONLY } ]
 );
```

Ví dụ: `dbcc shrinkdatabase(tempdb, 40);`



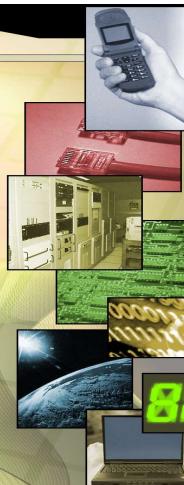
72

THU HẸP CSDL

- ❖ Thu hẹp file (cụ thể) trong CSDL

```
USE Tên_CSDL;
DBCC SHRINKFILE
({tên_file | id_file }
[, EMPTYFILE ]
| [[, kích_thước_file] |,{NOTRUNCATE|TRUNCATEONLY} ]]
);
```

Ví dụ: use QLSV;
dbcc shrinkfile (QLSV_Log, 1);



73

THÊM FILE VÀO CSDL

ALTER DATABASE *Tên_CSDL*

ADD FILE [TO FILEGROUP *Tên_nhóm*] -- Thêm file dữ liệu .ndf
| LOG FILE -- Thêm file log

```
(  
    Name      = tên_file_logic  
    , FileName = 'tên_file_vật_lý'  
    , Size     = kích_thước_ban_đầu [KB|MB|GB|TB]  
    , MaxSize  = kích_thước_tối_đa [KB|MB|GB|TB]  
    , FileGrowth = kích_thước_tăng_trưởng  
)
```

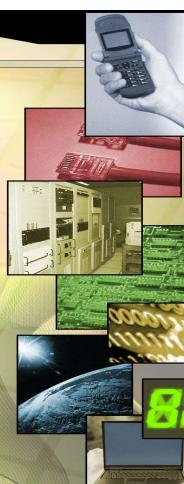


74

THÊM FILE VÀO CSDL

Ví dụ:

```
ALTER DATABASE QLSV
ADD FILE
(
    NAME = QLSV_data2,
    FILENAME = 'd:\qlsv_data2.ndf'
);
```



75

THÊM FILE VÀO CSDL

Ví dụ:

```
ALTER DATABASE QLSV
ADD LOG FILE
(  
    NAME      = qlsvlog2,  
    FILENAME  = 'D:\SQLServer\qlsvlog2.ldf',  
    SIZE      = 5MB,  
    MAXSIZE   = 100MB,  
    FILEGROWTH = 5%  
)  
(  
    NAME      = qlsvlog3,  
    FILENAME  = 'D:\SQLServer\qlsvlog3.ldf',  
    SIZE      = 5MB,  
    MAXSIZE   = 100MB,  
    FILEGROWTH = 5MB  
)
```



76

GỠ BỎ FILE CSDL

```
USE Tên_CSDL;
-- Làm trống file
DBCC SHRINKFILE (Tên_file_logic, EMPTYFILE);
-- Gỡ bỏ file
ALTER DATABASE Tên_CSDL
    REMOVE FILE Tên_file_logic;
```

77

ĐỔI TÊN CSDL

- ❖ Cách 1


```
ALTER DATABASE Tên_CSDL
        MODIFY NAME = Tên_CSDL_mới;
```

Ví dụ: alter database QLSV modify name = QuanLySinhVien;
- ❖ Cách 2: sử dụng thủ tục **sp_renamedb** của hệ thống
`sp_renamedb 'Tên_CSDL_cũ', 'Tên_CSDL_mới'`

Ví dụ: sp_renamedb N'QLSV', N'QuanLySinhVien'

Lưu ý:

 - CSDL phải ở chế độ một người dùng
 - Phải làm việc trong CSDL chính để thực thi thủ tục

78

GỠ BỎ (XÓA) CSDL

- ❖ Gỡ bỏ thông tin của CSDL từ các bảng hệ thống và gỡ bỏ các tập tin nhật ký và dữ liệu từ hệ thống.
- ❖ CSDL bị xóa có thể khôi phục lại từ bản sao.
- ❖ Không một người dùng nào đang sử dụng CSDL tại thời điểm nó bị xóa.
- ❖ Phải là thành viên của db_owner (hoặc vai trò máy chủ sysadmin) để xóa CSDL
- ❖ **Cú pháp:**

```
DROP DATABASE Tên_CSDL [, Tên_CSDL_2] [...];
```

79

BẢN CHỤP CSDL (DATABASE SNAPSHOTS)

- ❖ Là bản chỉ đọc, khung nhìn tĩnh của 1 CSDL

```
CREATE DATABASE Tên_bản_chụp_CSDL ON
(
    Name      = tên_file_logic,
    FileName = 'tên_file_vật_lý',
    ) [, ...] -- Các file bản chụp tiếp theo
AS SNAPSHOT OF Tên_CSDL_gốc;
```

80

BẢN CHỤP CSDL (DATABASE SNAPSHOTS)

Ưu điểm

- Bản chụp cung cấp một bản copy chỉ đọc của dữ liệu.
- Khi một bản chụp được truy vấn, không làm giảm hiệu suất của đối tượng được quan sát.
- Các tập tin dữ liệu của bản chụp là nhỏ và được tạo ra rất nhanh. Nó chỉ lớn khi cơ sở dữ liệu là chủ thể thay đổi thường xuyên.
- Sử dụng chủ yếu cho mục đích báo cáo, bảo vệ dữ liệu khỏi các lỗi quản trị, các lỗi người dùng và quản lý các dữ liệu thử nghiệm

Nhược điểm

- Bản sao của bản chụp không thể được tạo
- Bản chụp phải tồn tại trên cùng máy chủ cơ sở dữ liệu cùng với cơ sở dữ liệu gốc.
- Một người sử dụng mới không thể được quyền truy cập vào dữ liệu trong bản chụp. Quyền được thừa kế từ cơ sở dữ liệu gốc khi nó đã tồn tại tại thời điểm tạo ra bản chụp.

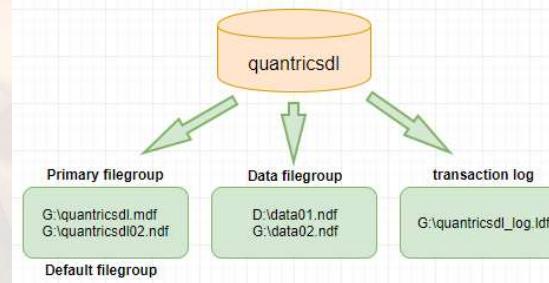
81

NHÓM TẬP TIN (FILEGROUP) CSDL

- ❖ Filegroup là tên đặt cho một nhóm data file trong SQL Server.
- ❖ Filegroup không chứa dữ liệu, mà chỉ định nghĩa ở mức logic các data file nằm trong đó (các data file mới thực sự chứa dữ liệu).
- ❖ Tương tự như tổ chức thư mục trong Windows.
- ❖ Filegroup mặc định trong SQL Server là **PRIMARY**

82

NHÓM TẬP TIN (FILEGROUP) CSDL



83

NHÓM TẬP TIN (FILEGROUP) CSDL

- ❖ Filegroup có thể được tạo cùng lúc với tạo CSDL hoặc thêm vào sau.
- ❖ Một file không thể thuộc nhiều hơn 1 filegroup tại cùng thời điểm.
- ❖ Không thể thay đổi filegroup của 1 file trong CSDL
- ❖ Filegroup chỉ có thể chứa file dữ liệu, KHÔNG chứa file log.

84

NHÓM TẬP TIN (FILEGROUP) CSDL

- Thêm filegroup khi tạo CSDL

CREATE DATABASE *Tên_CSDL*

[ON

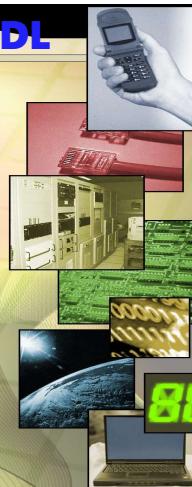
[PRIMARY] <định_nghĩa_file_data> [, ...n]

[, <FILEGROUP> [, ...n]]

[LOG ON <định_nghĩa_file_log> [, ...n]]

]

[;]



85

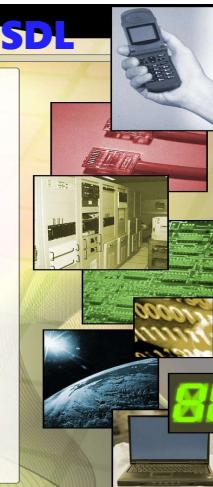
NHÓM TẬP TIN (FILEGROUP) CSDL

Ví dụ: CREATE DATABASE MySQLDB ON PRIMARY

```

    (
        NAME      = 'MySQLDB_Data',
        FILENAME = 'd:\baitap\mysql ldb_data.mdf',
        SIZE     = 10,
        MAXSIZE  = UNLIMITED,
        FILEGROWTH = 10%
    )
    FILEGROUP my_Group
    (
        NAME      = 'Ext_Data',
        FILENAME = 'd:\baitap\ext_data.ndf',
        SIZE     = 2 ,
        MAXSIZE  = UNLIMITED,
        FILEGROWTH = 1
    )
    LOG ON
    (
        NAME      = 'MySQLDB_Log',
        FILENAME = 'd:\baitap\mysql ldb_log.ldf',
        SIZE     = 1 ,
        MAXSIZE  = 10,
        FILEGROWTH = 1
    );

```



86

NHÓM TẬP TIN (FILEGROUP) CSDL

- Thêm filegroup vào CSDL đã có

ALTER DATABASE *Tên_CSDL*

ADD FILEGROUP *Tên_nhóm*

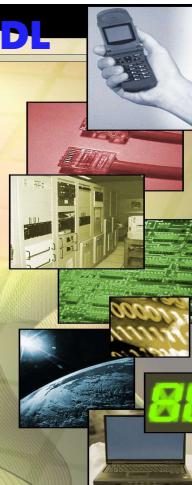
- Thêm file vào filegroup trong CSDL

ALTER DATABASE *Tên_CSDL*

ADD FILE

(<định_nghĩa_file_data> [, ...n])

TO FILEGROUP *Tên_nhóm*;



87

NHÓM TẬP TIN (FILEGROUP) CSDL

Ví dụ:

```
ALTER DATABASE QLSV
ADD FILEGROUP DaiHoc;
```

```
ALTER DATABASE QLSV
ADD FILE
```

```

    (
        NAME      = 'DaiHoc2019_Data',
        FILENAME = 'd:\baitap\daihoc2019_data.ndf',
        SIZE     = 100,
        MAXSIZE  = UNLIMITED,
        FILEGROWTH = 10%
    )
    TO FILEGROUP DaiHoc;

```



88

SCHEMA

- ❖ Một schema thuận lợi cho việc tách, quản lý, và quyền sở hữu của các đối tượng CSDL, nâng cao quản trị an ninh của các đối tượng CSDL.
- ❖ Có trong SQL Server 2005 trở lên
- ❖ Schema là một namespace đối với các đối tượng CSDL



89

SCHEMA

- ❖ Trong một CSDL, tên của schema là duy nhất, luôn có tên với đường đày đủ dạng **server.database.schema.object**
- ❖ Schema mặc định của các đối tượng trong CSDL là **dbo**
- ❖ **Ưu điểm**
 - Nhóm các đối tượng CSDL lại với nhau cho dễ quản lý
 - Cho phép phân quyền ở schema tăng tính bảo mật



90

SCHEMA

- ❖ Tạo schema
CREATE SCHEMA *Tên_schema*;
- ❖ Xóa schema
DROP SCHEMA *Tên_schema*;
- ❖ Thay đổi schema cho đối tượng CSDL
 - Dành cho kiểu dữ liệu người dùng định nghĩa
ALTER SCHEMA *Tên_schema_mới*
TRANSFER TYPE :: *Tên_schema_cũ.User_Type*
 - Dành cho các đối tượng CSDL khác
ALTER SCHEMA *Tên_schema_mới*
TRANSFER OBJECT :: *Tên_schema_cũ.Đối_tượng*



91

SCHEMA

Ví dụ:

```

CREATE SCHEMA S1;
CREATE SCHEMA S2;
-- Tạo kiểu dữ liệu thuộc S1
CREATE TYPE S1.TestType FROM [varchar](10) NOT NULL;
-- Kiểm tra thông tin schema
SELECT sys.types.name, sys.types.schema_id, sys.schemas.name
  FROM sys.types JOIN sys.schemas
    ON sys.types.schema_id = sys.schemas.schema_id
   WHERE sys.types.name = 'TestType';
GO
-- Thay đổi kiểu dữ liệu TestType từ S1 sang S2
ALTER SCHEMA S2 TRANSFER type::S1.TestType;

```



92

KIỂU DỮ LIỆU		
KIỂU DỮ LIỆU	KÍCH THƯỚC	MIỀN GIÁ TRỊ LUU TRỮ
❖ Các kiểu dữ liệu dạng số nguyên		
BigInt	8 bytes	từ -2^{63} đến $2^{63} - 1$
Int	4 bytes	từ -2^{31} đến $2^{31} - 1$
SmallInt	2 bytes	từ -2^{15} đến $2^{15} - 1$
TinyInt	1 byte	từ 0 đến 255
Bit	1 byte	0, 1 hoặc Null
❖ Các kiểu dữ liệu số thực		
Decimal[(p , s)]	5 – 17 bytes	<ul style="list-style-type: none"> p là tổng số chữ số (mặc định là 18). s là số chữ số thập phân (mặc định là 0) Tối đa $-10^{38} + 1$ đến $10^{38} - 1$ (khi sử dụng 17 bytes)
Dec[(p , s)]		
Numeric[(p , s)]		
Float[(n)]	4 – 8 bytes	Số thực dấu chấm (phẩy) đồng với phần định trị n bit (mặc định là 53 – tối đa)
Real	4 bytes	Tương đương với Float(24) . Số thực chính xác đơn.

93

KIỂU DỮ LIỆU		
KIỂU DỮ LIỆU	KÍCH THƯỚC	MIỀN GIÁ TRỊ LUU TRỮ
❖ Các kiểu dữ liệu tiền tệ (số)		
Money	8 bytes	từ $-922,337,203,685,477.5808$ đến $922,337,203,685,477.5807$
SmallMoney	4 bytes	từ $-214,748.3648$ đến $214,748.3647$

94

KIỂU DỮ LIỆU		
KIỂU DỮ LIỆU	KÍCH THƯỚC	MIỀN GIÁ TRỊ LUU TRỮ
❖ Các kiểu dữ liệu ký tự		
Char [(n)]	Tối đa 8000 ký tự (mỗi ký tự 1 byte)	<ul style="list-style-type: none"> n là số ký tự lưu trữ (mặc định 1) Độ dài cố định Thêm dấu cách về bên phải để bù phần trống cho dù số ký tự. Không chứa ký tự Unicode.
NChar [(n)]	Tối đa 4000 ký tự (mỗi ký tự 2 bytes)	Tương tự như Char nhưng chứa được ký tự Unicode
VarChar [(n max)]	Tối đa 8000 ký tự (mỗi ký tự 1 byte) hoặc tính theo max	<ul style="list-style-type: none"> n là số ký tự lưu trữ (mặc định 1) Độ dài tùy biến. Nếu chỉ định max thì tối đa là 2GB Không chứa ký tự Unicode.
NVarChar [(n max)]	Tối đa 4000 ký tự (mỗi ký tự 2 byte) hoặc tính theo max	Tương tự như VarChar nhưng chứa được ký tự Unicode
Binary [(n)]	Tối đa 8000 ký tự (mỗi ký tự 1 byte)	Tương tự như Char / VarChar nhưng chỉ chứa giá trị nhị phân (mã của ký tự)
NBinary [(n)]		

95

KIỂU DỮ LIỆU		
KIỂU DỮ LIỆU	KÍCH THƯỚC	MIỀN GIÁ TRỊ LUU TRỮ
❖ Các kiểu dữ liệu thời gian		
Date	3 bytes	<ul style="list-style-type: none"> Hiển thị dưới dạng YYYY-MM-DD Ngày từ 0001-01-01 đến 9999-12-31
DateTime	8 bytes	<ul style="list-style-type: none"> Hiển thị dưới dạng YYYY-MM-DD hh:mm:ss[.mmm] Ngày từ 1753-01-01 đến 9999-12-31 Giờ từ 00:00:00 đến 23:59:59.999999
DateTime2 [(ps)]	Tối đa 8 bytes	<ul style="list-style-type: none"> Hiển thị dưới dạng YYYY-MM-DD hh:mm:ss[.số_giây_thập_phân] Ngày từ 0001-01-01 đến 9999-12-31 Giờ từ 00:00:00 đến 23:59:59.999999 ps – số chữ số chính xác của giây (mặc định là 7 – tối đa)

96

KIỂU DỮ LIỆU		
KIỂU DỮ LIỆU	KÍCH THƯỚC	MÌN GIÁ TRỊ LUU TRỮ
❖ Các kiểu dữ liệu thời gian (tt)		
SmallDatetime	4 bytes	<ul style="list-style-type: none"> Hiển thị dưới dạng YYYY-MM-DD Ngày từ 0001-01-01 đến 2079-06-06 Giờ từ 00:00:00 đến 23:59:59
Time [(ps)]	5 bytes	<ul style="list-style-type: none"> hh:mm:ss[.số_giây_thập_phân] Giờ từ 00:00:00 đến 23:59:59.999999 ps – số chữ số chính xác của giây (mặc định là 7 – tối đa)
DateTimeOffset [(ps)] [(múi_giờ)]	10 bytes	<ul style="list-style-type: none"> Hiển thị dưới dạng YYYY-MM-DD hh:mm:ss[.số_giây_thập_phân][{+/-}hh:mm] Ngày từ 0001-01-01 đến 9999-12-31 Giờ từ 00:00:00 đến 23:59:59.999999 Múi giờ từ -14:00 đến +14:00 ps – số chữ số chính xác của giây (mặc định là 7 – tối đa)

97

GIÁ TRỊ NULL		
<ul style="list-style-type: none"> Giá trị NULL: giá trị dữ liệu tồn tại trong CSDL có thể không xác định được do: <ul style="list-style-type: none"> Giá trị đó có tồn tại nhưng không biết. Không xác định được giá trị đó có tồn tại hay không. Tại một thời điểm nào đó giá trị chưa có nhưng rồi có thể sẽ có. Giá trị bị lỗi do tính toán (tràn số, chia cho không...). <p>→ Được biểu diễn bởi các giá trị NULL.</p> <p>Lưu ý: Phân biệt NULL với giá trị rỗng (kiểu ký tự - chuỗi) hoặc giá trị 0 (kiểu số)</p>		

98

TẠO KIỂU DỮ LIỆU		
<ul style="list-style-type: none"> Tạo kiểu dữ liệu đơn tự định nghĩa 		
CREATE TYPE [<i>Tên_schema.</i>] <i>Tên_kiểu</i> FROM <i>Kiểu_dữ_liệu_gốc</i> [NONE NOT NULL]; <i>Ví dụ:</i> <pre>CREATE TYPE AgeType FROM TinyInt NOT NULL; CREATE TYPE ntu.LocationType FROM VARCHAR(500);</pre>		

99

TẠO KIỂU DỮ LIỆU		
<ul style="list-style-type: none"> Tạo kiểu dữ liệu dạng bảng 		

100

TẠO KIỂU DỮ LIỆU

Ví dụ:

```
CREATE TYPE Item_Trans AS TABLE
(
    ItemId NVARCHAR(10) NOT NULL,
    SupplierId INT NOT NULL,
    Price DECIMAL (18, 4) NULL,
    PRIMARY KEY (
        ItemId ASC,
        SupplierId DESC
    ),
    INDEX IX_Item_Price ( Price ),
    CHECK (Price >= 0)
);
```

101

XÓA KIỂU DỮ LIỆU

DROP TYPE [IF EXISTS] [Tên_schema.] Tên_kiểu ;

Lưu ý: Không sử dụng ALTER để sửa đổi thông số trong kiểu dữ liệu tự định nghĩa. Để thay đổi kiểu dữ liệu thì ta phải XÓA kiểu dữ liệu cũ và tạo lại kiểu dữ liệu với thông số mới.

102

TẠO BẢNG DỮ LIỆU

CREATE TABLE tên_bảng

```
(  
    tên_cột_1 kiêu_dữ_liệu [thuộc_tính_cột_1] [các_ràng_buộc_cột_1],  
    tên_cột_2 kiêu_dữ_liệu [thuộc_tính_cột_2] [các_ràng_buộc_cột_2],  
    ...,  
    tên_cột_n kiêu_dữ_liệu [thuộc_tính_cột_n] [các_ràng_buộc_cột_n]  
    [, các_ràng_buộc_trên_bảng]  
);
```

103

TẠO BẢNG DỮ LIỆU

Ví dụ:

```
CREATE TABLE NHANVIEN  
(  
    manv      VARCHAR(10) PRIMARY KEY,  
    honv      NVARCHAR(30) NOT NULL,  
    tennv      NVARCHAR(20) NOT NULL,  
    gioitinh  BIT DEFAULT(1),  
    ngaysinh   DATE NOT NULL,  
    diachi     NVARCHAR(100) NOT NULL,  
    dienthoai  NVARCHAR(15) NULL  
)
```

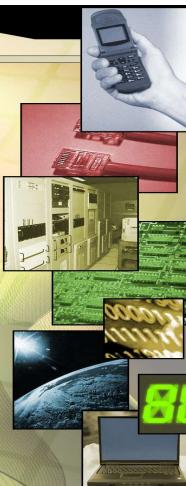
104

RÀNG BUỘC KHÓA CHÍNH

❖ **PRIMARY KEY** : định nghĩa **khoá chính** của bảng
[CONSTRAINT tên_ràng_buộc]
PRIMARY KEY (danh_sách_cột)

Ví dụ:

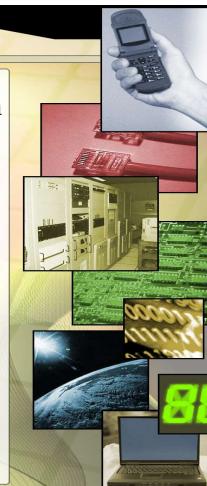
```
create table KetQuaHocTap
(
    masv varchar(10) not null,
    mamh varchar(10) not null,
    diem tinyint,
    constraint kequaHT_pk primary key(masv, mamh)
);
```



105

RÀNG BUỘC KHÓA NGOẠI

❖ **FOREIGN KEY** : được sử dụng để tạo ra mối liên kết dữ liệu giữa hai bảng.
[CONSTRAINT tên_ràng_buộc] FOREIGN KEY [(danh_sách_cột)] REFERENCES tên_bảng_tham_chiếu (danh_sách_cột_tham_chiếu)
[ON DELETE CASCADE | NO ACTION | SET NULL | SET DEFAULT]
[ON UPDATE CASCADE | NO ACTION | SET NULL | SET DEFAULT]



106

RÀNG BUỘC KHÓA NGOẠI

❖ Giá trị thêm vào cột FOREIGN KEY chỉ có thể nằm trong tập giá trị của PRIMARY KEY hoặc cột có ràng buộc UNIQUE của bảng mà FOREIGN KEY tham chiếu tới.

❖ Mặc định SQL Server không cho phép xóa dữ liệu trong bảng cha khi giá trị khóa chính đang được tham chiếu trong bảng con.



107

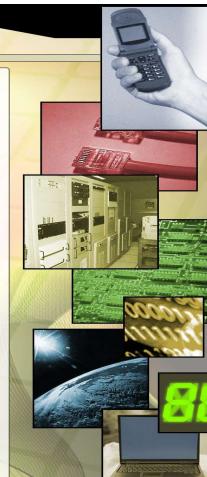
RÀNG BUỘC KHÓA NGOẠI

❖ **CASCADE** : Tự động xoá | cập nhật bản ghi trong bảng con nếu bản ghi được tham chiếu trong bảng cha bị xoá | cập nhật.

❖ **NO ACTION** : Tùy chọn **mặc định** khi tạo khóa ngoại trong SQL Server. Một dòng trong bảng cha đang được tham chiếu sẽ không thể bị xoá hay cập nhật.

❖ **SET NULL** : Cập nhật lại giá trị khoá ngoại của dòng trong bảng con thành NULL nếu dòng được tham chiếu trong bảng cha bị xoá.

❖ **SET DEFAULT** : Cập nhật lại giá trị khoá ngoại của dòng trong bảng con thành giá trị mặc định của cột nếu dòng được tham chiếu trong bảng cha bị xoá



108

RÀNG BUỘC KHÓA NGOẠI

Ví dụ:

```
create table dondathang
(
    mahoadon varchar(10) primary key,
    makh varchar(10) not null,
    manv varchar(10) not null,
    ngaydathang datetime,
    constraint fk_dondathang_khachhang foreign key (makh)
        references khachhang (makh),
    constraint fk_dondathang_nhanvien foreign key (manv)
        references nhanvien (manv)
)
```



109

RÀNG BUỘC GIỚI HẠN DỮ LIỆU

❖ **CHECK** : chỉ định điều kiện hợp lệ đối với dữ liệu dữ liệu được **thêm** vào hoặc **cập nhật** trong bảng.

[CONSTRAINT tên_ràng_buộc] CHECK (*điều_kiện_logic*)

Ví dụ:

```
create table KetQuaHocTap
(
    masv varchar(10) not null,
    mamh varchar(10) not null,
    diem tinyint,
    constraint kequaHT_pk primary key(masv, mamh),
    constraint diem_ck check(diem >= 0 and diem <=10)
)
```



110

RÀNG BUỘC DỮ LIỆU DUY NHẤT

❖ **UNIQUE** : đảm bảo giá trị của một dòng tại một hay nhiều cột tham gia ràng buộc là **duy nhất**.

[CONSTRAINT tên_ràng_buộc] UNIQUE (*danh_sách_cột*)

Ví dụ:

```
create table hanghoa
(
    mahang varchar(10) primary key,
    tenhang nvarchar(100) not null,
    donvitinh nvarchar(10) not null,
    dongia money check(dongia >= 0),
    constraint tenhang_unq unique (tenhang)
);
```



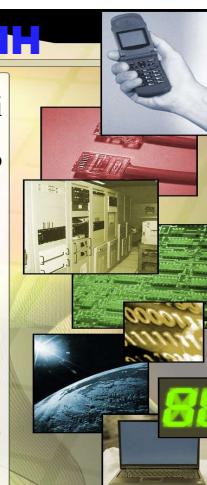
111

RÀNG BUỘC DỮ LIỆU MẶC ĐỊNH

❖ **DEFAULT** : cung cấp giá trị mặc định cho cột khi thêm dòng mới hoặc cập nhật dữ liệu mà giá trị cho cột này không được cung cấp.

Ví dụ:

```
CREATE TABLE MyTable
(
    column_a INT,
    column_b INT DEFAULT 50
);
CREATE TABLE MyTable
(
    column_a INT,
    column_b INT
    CONSTRAINT column_b_def DEFAULT 50
);
```



112

SỬA ĐỔI BẢNG	
CÔNG VIỆC	CÚ PHÁP
Thêm cột vào bảng	<code>ALTER TABLE tên_bảng ADD định_nghĩa_cột_1 [, ...n]</code>
Sửa cột trong bảng	<code>ALTER TABLE tên_bảng ALTER COLUMN tên_cột kiểu_dữ_liệu [NULL NOT NULL]</code>
Xóa cột trong bảng	<code>ALTER TABLE tên_bảng DROP COLUMN tên_cột_1 [, ...n]</code>
Thêm ràng buộc vào bảng	<code>ALTER TABLE tên_bảng ADD CONSTRAINT tên_ràng_buộc <i>định_nghĩa_ràng_buộc</i></code>
Xóa ràng buộc	<code>ALTER TABLE tên_bảng DROP CONSTRAINT tên_ràng_buộc</code>

113

SỬA ĐỔI BẢNG

Ví dụ:

- ❑ Thêm cột EMAIL, DIACHI vào bảng NHANVIEN
`alter table nhanvien
add email nvarchar(50) not null,
diachi nvarchar(200);`
- ❑ Thay đổi định nghĩa cột EMAIL
`alter table nhanvien
alter column email nvarchar(40) null ;`
- ❑ Xóa cột EMAIL
`alter table nhanvien
drop column email ;`

114

SỬA ĐỔI BẢNG

Ví dụ:

- ❑ Thêm cột EMAIL và xây dựng ràng buộc cho cột này
`alter table nhanvien
add column email nvarchar(40) null,
constraint chk_email check
(email like N'%@%.com');`

115

XÓA BẢNG

DROP TABLE Tên_bảng_1 [, ...n];

Ví dụ:

- `drop table hanghoa, MyTable;`
- ❖ Lệnh DROP TABLE không thể thực hiện nếu bảng cần xoá đang được tham chiếu bởi một ràng buộc FOREIGN KEY → cần xóa ràng buộc FOREIGN KEY hoặc bảng tham chiếu trước

116

VIEW

- ❖ Khung nhìn (view) là một thành phần của một bảng hoặc nhiều bảng được tạo thành bởi kết quả của một truy vấn SQL.
- ❖ View là bảng ảo cho phép
 - Cấu trúc dữ liệu theo cách của người dùng.
 - Giới hạn truy cập tới dữ liệu người dùng có thể thấy hoặc sửa đổi.
 - Tổng kết dữ liệu từ các bảng khác nhau để tạo ra bảng báo cáo.

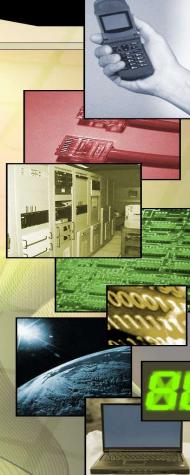


117

TẠO VIEW

**CREATE [OR ALTER] VIEW [tên_schema .] tên_view
[(tên_cột_đại_diện_1 [, ...n])]
AS
< lệnh SELECT >
[WITH CHECK OPTION] ;**

Lưu ý: Lệnh SELECT **không** chứa mệnh đề ORDER BY, OPTION, từ khóa INTO; **không** chứa tham chiếu đến bảng tạm hoặc biến kiểu bảng



118

TẠO VIEW

Ví dụ:

```
create view sinhvien_view as
select ten, diachi
from sinhvien;
```

Ví dụ:

```
create or alter view myview (khoa_name, tongsv) as
select k.khoa_name, COUNT(s.sv_id)
from khoa k , sinhvien s
where k.khoa_id = s.khoa_id
group by (k.khoa_name);
```



119

TẠO VIEW – CHECK OPTION

❖ **WITH CHECK OPTION** bảo đảm rằng tất cả UPDATE và INSERT thỏa mãn các điều kiện trong định nghĩa VIEW.

Ví dụ: `create view sinhvien_view as
select ten, diachi
from sinhvien
where diachi is not null
with check option;`

Với ví dụ trên, khi thực hiện insert hay update view thì trường *diachi* mới không được có giá trị NULL.



120

CẬP NHẬT VIEW

Điều kiện để cập nhật được view

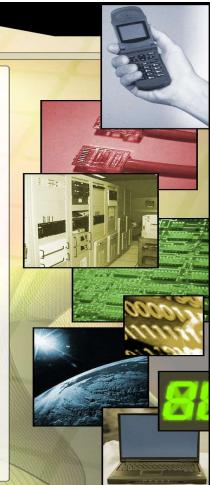
- ❖ Mệnh đề SELECT không thể chứa từ khóa DISTINCT.
- ❖ Mệnh đề SELECT không thể chứa các hàm tổng.
- ❖ Mệnh đề SELECT không thể chứa các hàm và các toán tử tập hợp.
- ❖ Mệnh đề SELECT không thể chứa một mệnh đề ORDER BY.



CẬP NHẬT VIEW

Điều kiện để cập nhật được view (tt)

- ❖ Mệnh đề FROM không thể chứa nhiều bảng.
- ❖ Mệnh đề WHERE không thể chứa các truy vấn phụ.
- ❖ Truy vấn không thể chứa GROUP BY hoặc HAVING.
- ❖ Các cột được ước lượng không thể bị cập nhật.
- ❖ Tất cả các cột NOT NULL từ bảng ban đầu phải được bao trong view để cho truy vấn INSERT vận hành.



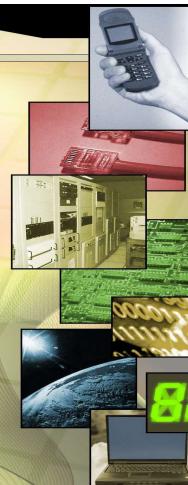
121

122

CẬP NHẬT VIEW

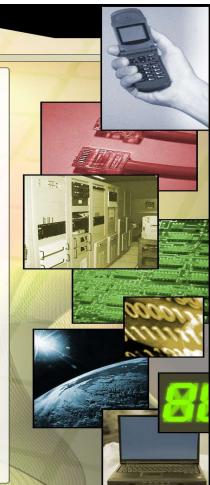
Điều kiện để cập nhật được view (tt)

- ❖ Mệnh đề FROM không thể chứa nhiều bảng.
- ❖ Mệnh đề WHERE không thể chứa các truy vấn phụ.
- ❖ Truy vấn không thể chứa GROUP BY hoặc HAVING.
- ❖ Các cột được ước lượng không thể bị cập nhật.
- ❖ Tất cả các cột NOT NULL từ bảng ban đầu phải được bao trong view để cho truy vấn INSERT vận hành.



CẬP NHẬT VIEW

- ❖ INSERT
- ❖ UPDATE
- ❖ DELETE



123

124

NGÔN NGỮ THAO TÁC DỮ LIỆU

DATA MANIPULATION LANGUAGE - DML

125

SELECT

```

SELECT [ ALL | DISTINCT ]  

[ Tên_dai_diện_bảng_1 . ] Mục_chọn_1 [ AS Tên_cột_hiển_thị_1 ]  

[ , ...n ]  

FROM Bảng_1 [ Tên_dai_diện_bảng_1 ] [, ...n ]  

[ WHERE Điều_kiện_1 [ AND | OR Điều_kiện_2... ] ]  

[ GROUP BY Nhóm_cột_1 [, ...n ] ]  

[ HAVING Điều_kiện_nhóm ]  

[ UNION [ ALL ] Lệnh_SELECT ]  

[ ORDER BY Cột_sắp_xếp [ ASC | DESC ] [, ... n ] ];

```

126

SELECT

Ví dụ:

```

select *  

from sinhvien;

```

Ví dụ:

```

select khoa_name, COUNT(sv_id)  

from khoa k , sinhvien s  

where k.khoa_id = s.khoa_id  

group by (khoa_name)  

having COUNT(sv_id) <= 50;

```

127

SELECT - ALIAS

ALIAS trong SQL Server là cách đổi tên một bảng hoặc một cột tạm thời bằng tên khác cho mục đích truy vấn (hiển thị hoặc rút gọn tên...). Tên ALIAS sẽ không ảnh hưởng (thay đổi) đến đối tượng trong CSDL và không thể dùng lại cho mục đích truy vấn hoặc tính toán

tên_cột | tên_bảng [AS] tên_ALIAS

Lưu ý: Nếu tên ALIAS có chứa khoảng trắng thì phải đặt trong dấu trích dẫn '' hoặc []

128

SELECT - ALIAS

Ví dụ:

```
select manv as [Mã nhân viên],
       honv + ' ' + tennv 'Họ tên nhân viên',
       diachi as 'Địa chỉ'
  from nhanvien;
```

129

SELECT - ALIAS

Ví dụ:

```
select tmp.honv + ' ' + tmp.tennv as hoten,
       tmp.gioitinh, tmp.diachi
  from ( select * from nhanvien
    where manv = 'nv001' or manv = 'nv002'
  ) as tmp;
```

130

SELECT – THAY ĐỔI KẾT QUẢ HIỂN THỊ

Cấu trúc CASE...WHEN... được sử dụng nhằm thay đổi kết quả hiển thị của truy vấn tùy mục đích khác nhau nhưng KHÔNG làm thay đổi dữ liệu trong CSDL

❖ Simple CASE

CASE biểu_thức

```
WHEN giá_trị_1 THEN kết_quả_1
WHEN giá_trị_2 THEN kết_quả_2
...
[ ELSE kết_quả_còn_lại ]
END
```

131

SELECT – THAY ĐỔI KẾT QUẢ HIỂN THỊ

❖ Searched CASE

CASE

```
WHEN biểu_thức_logic_1 THEN kết_quả_1
WHEN biểu_thức_logic_2 THEN kết_quả_2
...
[ ELSE kết_quả_còn_lại ]
END
```

132

SELECT – THAY ĐỔI KẾT QUẢ HIỂN THỊ

Ví dụ:

```
select manv, honv, tennv,
       case gioitinh
         when 1 then N'nữ'
         else N'nam'
       end as gioitinh, diachi
from nhanvien;
```

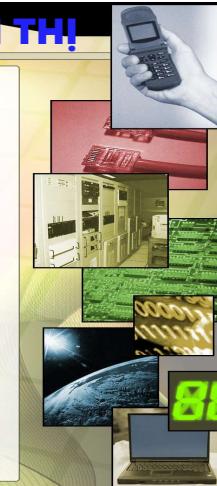


133

SELECT – THAY ĐỔI KẾT QUẢ HIỂN THỊ

Ví dụ:

```
select manv, honv, tennv,
       case
         when gioitinh=1 then N'nữ'
         else N'nam'
       end as gioitinh, diachi
from nhanvien;
```

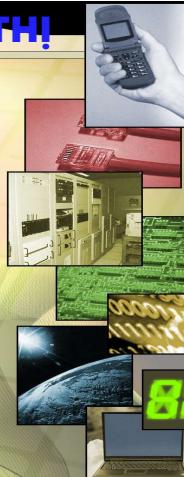


134

SELECT – THAY ĐỔI KẾT QUẢ HIỂN THỊ

Ví dụ:

```
select customername, city, country
from customers
order by
(case
  when city is null then country
  else city
end);
```



135

SELECT – HIỂN THỊ HỮU HẠN DÒNG

Ví dụ:

```
select TOP 2 manv, honv, tennv, diachi
from nhanvien;
```

Ví dụ:

```
select TOP 10 PERCENT manv, honv, tennv, diachi
from nhanvien;
```



136

SELECT – TOÁN TỬ ĐIỀU KIỆN

- ❖ Toán tử kết nối điều kiện: **AND, OR**
- ❖ Toán tử so sánh giá trị: **= , > , < , >= , <= , != , <> , !< , !>**
- ❖ Toán tử so sánh giá trị tập hợp: **ALL | SOME | ANY**
- ❖ Toán tử kiểm tra giới hạn của dữ liệu: **BETWEEN | NOT BETWEEN**
- ❖ Toán tử làm việc với tập hợp: **IN | NOT IN**
- ❖ Toán tử kiểm tra khuôn dạng dữ liệu: **LIKE | NOT LIKE**
- ❖ Toán tử làm việc với giá trị NULL: **IS NULL | IS NOT NULL**
- ❖ Toán tử kiểm tra tồn tại dữ liệu: **EXISTS**



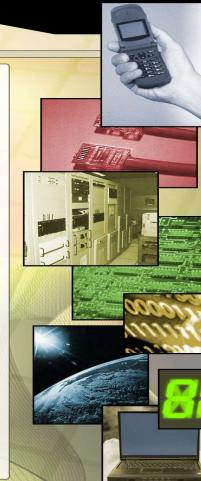
137

SELECT – TOÁN TỬ ĐIỀU KIỆN

Ví dụ: Liệt kê các nhân viên nữ có tuổi lớn hơn 40.

Ngày sinh hiển thị theo định dạng dd/MM/yyyy

```
select manv, honv + ' ' + tennv as hoten,
convert(varchar(10), ngaysinh, 103) as ngaysinh
from nhanvien
where gioitinh = 0
and datediff(year, ngaysinh, getdate()) >= 40;
```

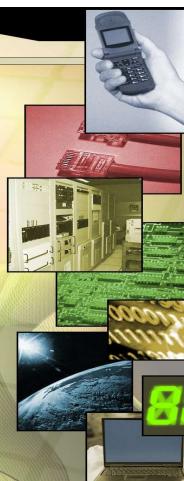


138

SELECT – TOÁN TỬ ĐIỀU KIỆN

Ví dụ: Liệt kê các nhân viên có tuổi thuộc tập hợp {50,55,60}

```
select *
from nhanvien
where datediff(year, ngaysinh, getdate())
in (50,55,60);
```

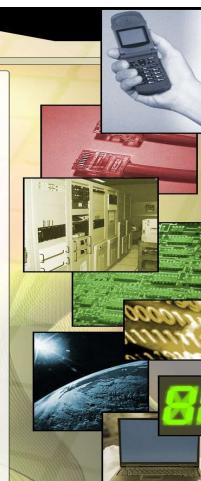


139

SELECT – TOÁN TỬ ĐIỀU KIỆN

Ví dụ: Liệt kê các nhân viên có lập hóa đơn cho khách hàng

```
select *
from nhanvien
where manv in (select manv from dondathang);
```



140

SELECT – TOÁN TỬ ĐIỀU KIỆN

Ví dụ: Liệt kê các phòng ban có nhân viên tên Châu

```
select * from phongban pb
where exists (
    select nv.manv
    from nhanvien nv
    where (nv.mapb = pb.mapb)
        and (nv.tenvn like N'%Châu%')
);
```



141

SELECT – TOÁN TỬ ĐIỀU KIỆN

Ví dụ: Liệt kê nhân viên có tuổi lớn nhất

```
select * from nhanvien
where datediff(year, ngaysinh, getdate())
>=all (select datediff(year, ngaysinh, getdate())
        from nhanvien);
```



142

SELECT – KÝ TỰ ĐẠI DIỆN (WILDCARD)

Ký tự đại diện	Ý nghĩa
%	Chuỗi ký tự bắt kí bao gồm cả chuỗi rỗng
_	Một ký tự
[]	Ký tự nằm trong giới hạn được chỉ định. <u>Ví dụ:</u> [a-f] hay [abcdef] hàm ý chỉ một trong các ký tự: a, b, c, d, e, f.
[^]	Ký tự không nằm trong giới hạn được chỉ định. <u>Ví dụ:</u> [^a-f] hay [^abcdef] hàm ý chỉ một ký tự khác tất cả các ký tự: a, b, c, d, e, f.



❖ **Ký tự thoát (escape character):** là ký tự đặt trước ký tự đại diện để đánh dấu rằng ký tự đại diện sau nó được xem là một ký tự bình thường.

143

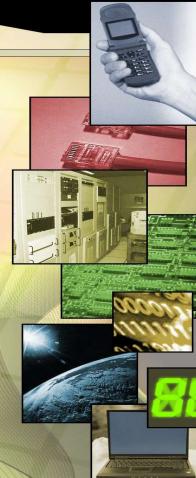
SELECT – WILDCARD

Ví dụ: Liệt kê những nhân viên họ Nguyễn

```
select *
from nhanvien
where honv like N'Nguyễn%';
```

Ví dụ:

```
select *
from tintuc
where noidung like N'%tăng trưởng 10-15!%'  
ESCAPE '!' ;
```



144

SELECT – PHÉP HỢP (UNION)

- ❖ Kết quả truy vấn của hai hay nhiều câu truy vấn

SELECT <Truy vấn Bảng 1>

UNION [ALL]

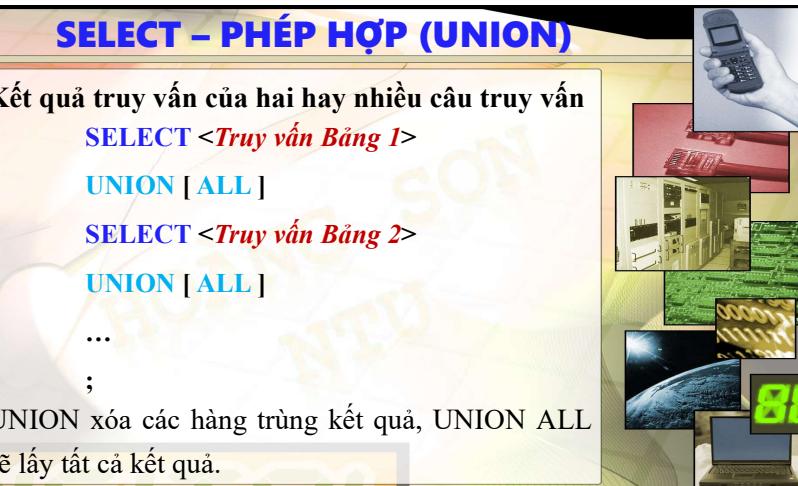
SELECT <Truy vấn Bảng 2>

UNION [ALL]

...

;

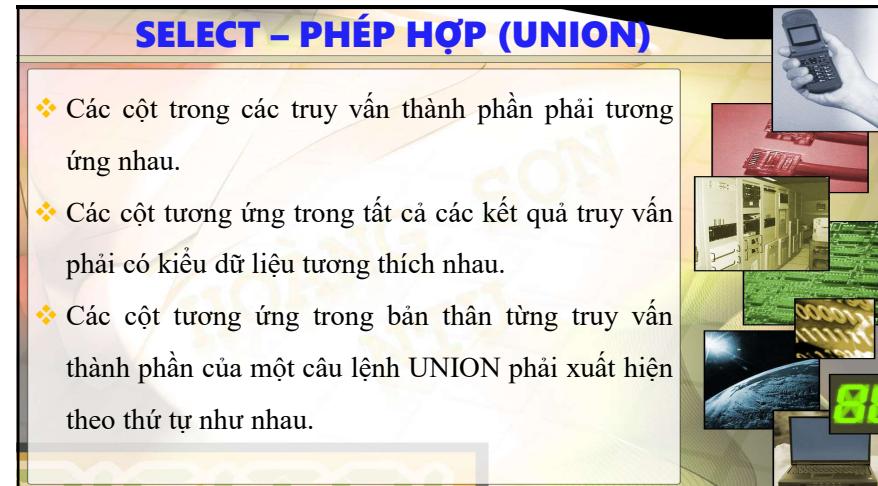
- ❖ UNION xóa các hàng trùng kết quả, UNION ALL sẽ lấy tất cả kết quả.



145

SELECT – PHÉP HỢP (UNION)

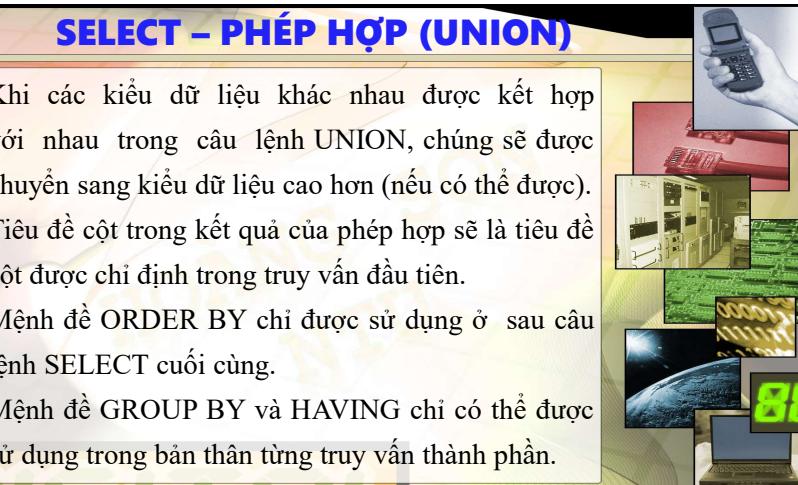
- ❖ Các cột trong các truy vấn thành phần phải tương ứng nhau.
- ❖ Các cột tương ứng trong tất cả các kết quả truy vấn phải có kiểu dữ liệu tương thích nhau.
- ❖ Các cột tương ứng trong bản thân từng truy vấn thành phần của một câu lệnh UNION phải xuất hiện theo thứ tự như nhau.



146

SELECT – PHÉP HỢP (UNION)

- ❖ Khi các kiểu dữ liệu khác nhau được kết hợp với nhau trong câu lệnh UNION, chúng sẽ được chuyển sang kiểu dữ liệu cao hơn (nếu có thể được).
- ❖ Tiêu đề cột trong kết quả của phép hợp sẽ là tiêu đề cột được chỉ định trong truy vấn đầu tiên.
- ❖ Mệnh đề ORDER BY chỉ được sử dụng ở sau câu lệnh SELECT cuối cùng.
- ❖ Mệnh đề GROUP BY và HAVING chỉ có thể được sử dụng trong bản thân từng truy vấn thành phần.



147

SELECT – PHÉP HỢP (UNION)

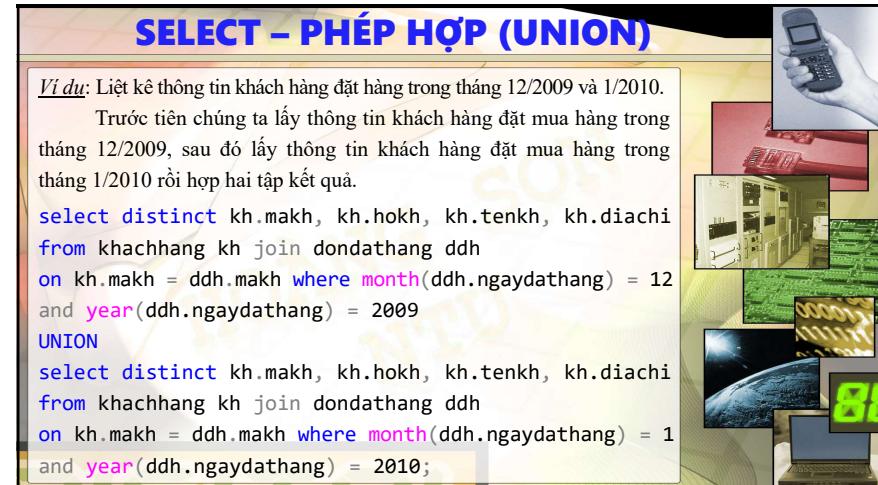
Ví dụ: Liệt kê thông tin khách hàng đặt hàng trong tháng 12/2009 và 1/2010.

Trước tiên chúng ta lấy thông tin khách hàng đặt mua hàng trong tháng 12/2009, sau đó lấy thông tin khách hàng đặt mua hàng trong tháng 1/2010 rồi hợp hai tập kết quả.

```

select distinct kh.makh, kh.hokh, kh.tenkh, kh.diachi
from khachhang kh join dondathang ddh
on kh.makh = ddh.makh where month(ddh.ngaydathang) = 12
and year(ddh.ngaydathang) = 2009
UNION
select distinct kh.makh, kh.hokh, kh.tenkh, kh.diachi
from khachhang kh join dondathang ddh
on kh.makh = ddh.makh where month(ddh.ngaydathang) = 1
and year(ddh.ngaydathang) = 2010;

```



148

SELECT – PHÉP GIAO (INTERSECT)

- ❖ Tìm những dòng dữ liệu giống nhau trong nhiều kết quả truy vấn.

```

SELECT <Truy vấn Bảng 1>
INTERSECT
SELECT <Truy vấn Bảng 2>
INTERSECT
...
;
    
```

❖ Các điều kiện của INTERSECT tương tự như UNION

149

SELECT – PHÉP GIAO (INTERSECT)

Ví dụ: Liệt kê thông tin khách hàng đặt hàng trong cả 2 tháng 12/2009 và 1/2010.

```

select distinct kh.makh, hokh, tenkh, diachi
from khachhang kh join dondathang ddh
on kh.makh = ddh.makh where month(ngaydathang) = 12
and year(ngaydathang) = 2009
INTERSECT
select distinct kh.makh, hokh, tenkh, diachi
from khachhang kh join dondathang ddh
on kh.makh = ddh.makh where month(ngaydathang) = 1
and year(ngaydathang) = 2010;
    
```

150

SELECT – PHÉP TRỪ (EXCEPT)

- ❖ Xác định các bản ghi thuộc kết quả câu truy vấn này nhưng không thuộc kết quả câu truy vấn khác.

```

SELECT <Truy vấn Bảng 1>
EXCEPT
SELECT <Truy vấn Bảng 2>
EXCEPT
...
;
    
```

❖ Các điều kiện của EXCEPT tương tự như UNION

151

SELECT – PHÉP TRỪ (EXCEPT)

Ví dụ: Liệt kê thông tin khách hàng đặt hàng trong tháng 12/2009 không đặt hàng trong tháng 1/2010.

```

select distinct kh.makh, kh.hokh, kh.tenkh, kh.diachi
from khachhang kh join dondathang ddh
on kh.makh = ddh.makh where month(ddh.ngaydathang) = 12
and year(ddh.ngaydathang) = 2009
EXCEPT
select distinct kh.makh, kh.hokh, kh.tenkh, kh.diachi
from khachhang kh join dondathang ddh
on kh.makh = ddh.makh where month(ddh.ngaydathang) = 1
and year(ddh.ngaydathang) = 2010;
    
```

152

SELECT – PHÉP NỐI (JOIN)

❖ Xác định kết quả truy vấn từ hai hay nhiều bảng

```

SELECT ...
FROM <Bảng_1> [ [ AS ] Alias_bảng_1 ]
<Kiểu_JOIN> <Bảng_2> [ [ AS ] Alias_bảng_2 ]
ON <Điều_kiện_JOIN>
...
;

```

153

SELECT – PHÉP NỐI TRONG (INNER JOIN)

❖ Kết quả trả về là những bản ghi chung giữa các bảng

```

SELECT ...
FROM <Bảng_1> [ [ AS ] Alias_bảng_1 ]
[ INNER | JOIN <Bảng_2> [ [ AS ] Alias_bảng_2 ]
ON <Điều_kiện_JOIN> ...
WHERE ...
;

```

154

SELECT – PHÉP NỐI TRONG (INNER JOIN)

❖ Có thể sử dụng câu lệnh SQL sau thay cho INNER JOIN

```

SELECT ...
FROM <Bảng_1> [ [ AS ] Alias_bảng_1 ],
<Bảng_2> [ [ AS ] Alias_bảng_2 ]
WHERE <Điều_kiện_JOIN> AND ...
...
;

```

155

SELECT – PHÉP NỐI TRONG (INNER JOIN)

Ví dụ: Truy xuất thông tin các nhân viên đã lập hóa đơn trong tháng 12/2019

```

select nv.manv, honv, tennv, diachi,
ngaydathang as ngaylaphoadon
from nhanvien nv join dondathang ddh
on nv.manv = ddh.manv
where month(ngaydathang) = 12
and year(ngaydathang) = 2019;

```

156

SELECT – PHÉP NỐI TRONG (INNER JOIN)

- ❖ Phép tự nối : điều kiện nối được chỉ định liên quan đến các cột của cùng một bảng. Các bảng phải được đặt tên alias

Ví dụ: Tìm các cặp nhân viên có cùng ngày sinh.

```
select nv1.manv, nv1.honv + ' ' + nv1.tennv
      as hoten
  from nhanvien nv1 join nhanvien nv2
    on nv1.manv <> nv2.manv
   and nv1.ngaysinh = nv2.ngaysinh;
```



157

SELECT – PHÉP NỐI NGOÀI (OUTER JOIN)

- ❖ Xác định kết quả truy vấn từ hai hay nhiều bảng kể cả những kết quả không thỏa mãn điều kiện nối

```
SELECT ...
FROM <Bảng_1> [ [ AS ] Alias_bảng_1 ]
LEFT | RIGHT | FULL | OUTER | JOIN
      <Bảng_2> [ [ AS ] Alias_bảng_2 ]
ON <Điều_kiện_JOIN> ...
WHERE ...
;
```



158

SELECT – PHÉP NỐI NGOÀI (OUTER JOIN)

- Ví dụ: Xem tất cả thông tin nhân viên đã và chưa lập hóa đơn trong tháng 12/2019

```
select nv.manv, nv.honv, nv.tennv, nv.diachi,
       ddh.ngaydathang as ngaylaphoaddon
  from nhanvien nv left join dondathang ddh
    on nv.manv = ddh.manv
   where month(ddh.ngaydathang) = 12
  and year(ddh.ngaydathang) = 2019;
```



159

SELECT GROUP BY – THỐNG KÊ DỮ LIỆU

- ❖ Sử dụng mệnh đề GROUP BY để gom nhóm và thực hiện việc thống kê dữ liệu trên từng nhóm

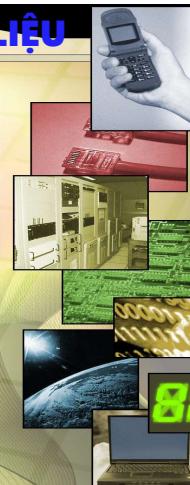
```
SELECT [ <Danh_sách_chọn> ],
       <Hàm_gộp_1> [ ,<Hàm_gộp_2>... ]
  FROM ...
 WHERE ...
[ GROUP BY <Danh_sách_chọn> ]
[ HAVING <Điều_kiện_nhóm> ]
;
```



160

SELECT GROUP BY – THỐNG KÊ DỮ LIỆU

HÀM GỘP	CHỨC NĂNG
SUM([ALL DISTINCT] <i>biểu_thức</i>)	Tính tổng các giá trị.
AVG([ALL DISTINCT] <i>biểu_thức</i>)	Tính trung bình của các giá trị
COUNT([ALL DISTINCT] <i>biểu_thức</i>)	Đếm số các giá trị trong biểu thức
COUNT(*)	Đếm số các dòng được chọn
MAX(<i>biểu_thức</i>)	Tính giá trị lớn nhất
MIN(<i>biểu_thức</i>)	Tính giá trị nhỏ nhất

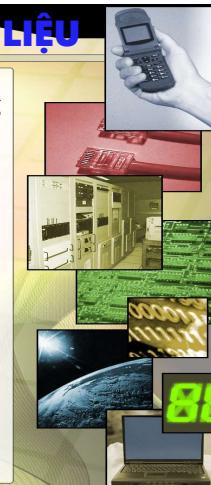


161

SELECT GROUP BY – THỐNG KÊ DỮ LIỆU

Ví dụ: Truy xuất tuổi lớn nhất, nhỏ nhất và tuổi trung bình của các nhân viên trong công ty.

```
select max(datediff(year, ngaysinh, getdate())) as tuailonnhat,
       min(datediff(year, ngaysinh, getdate())) as tuoinhonhat,
       avg(datediff(year, ngaysinh, getdate())) as tuoitrungbinh
  from nhanvien
```



162

SELECT GROUP BY – THỐNG KÊ DỮ LIỆU

Ví dụ: Liệt kê thông tin nhân viên và số đơn đặt hàng của các nhân viên đó trong tháng 12/2019

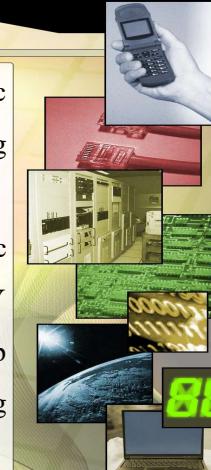
```
select nv.manv, nv.honv, nv.tenvn, nv.diachi,
       count(ddh.manv) as N'Số đơn hàng'
  from nhanvien nv left join dondathang ddh
    on nv.manv = ddh.manv
 where month(ddh.ngaydathang) = 12
   and year(ddh.ngaydathang) = 2019
 group by nv.manv, nv.honv, nv.tenvn, nv.diachi;
```



163

SELECT GROUP BY – HAVING

- ❖ Mệnh đề **WHERE** dùng để chỉ định điều kiện lọc áp dụng trên bất cứ dòng dữ liệu trong các bảng tham gia truy vấn.
- ❖ Mệnh đề **HAVING** chỉ có thể chỉ định điều kiện lọc đối với các dòng dữ liệu được liệt kê tại GROUP BY
- ❖ Mệnh đề HAVING cho phép sử dụng các hàm gộp trong biểu thức điều kiện còn WHERE thì không cho phép.



164

SELECT GROUP BY – HAVING

Ví dụ: Liệt kê thông tin nhân viên tên Châu có số đơn đặt hàng lớn hơn 10 trong năm 2019

```
select nv.manv, nv.honv, nv.tennv, nv.diachi
from nhanvien nv join dondathang ddh
    on nv.manv = ddh.manv
where year(ddh.ngaydathang) = 2019
group by nv.manv, nv.honv, nv.tennv, nv.diachi
having count(ddh.manv) > 10
and (nv.tennv like N'Châu');
```

165

SELECT INTO

- Tạo một bảng mới có cấu trúc và dữ liệu được xác định từ kết quả của câu lệnh SELECT

```
SELECT ... INTO [# | ##] tên_bảng
FROM ...
;
```

166

SELECT INTO

Ví dụ: Lưu thông tin các nhân viên đã lập hóa đơn trong tháng 12/2019 vào bảng có tên **baocao12_2019**

```
select nv.manv, honv, tennv, diachi,
       ngaydathang as ngaylaphoadon
into baocao12_2019
from nhanvien nv join dondathang ddh
    on nv.manv = ddh.manv
where month(ngaydathang) = 12
and year(ngaydathang) = 2019;
```

167

INSERT

```
INSERT INTO tên_bảng [ (danh_sách_cột) ]
VALUES (danh_sách_giá_trị_1) [, ...n];
```

- Nếu không khai báo **danh_sách_cột** thì phải nhập **danh_sách_giá_trị** có đầy đủ các cột của bảng.
- Có thể sử dụng truy vấn SELECT để nhập **danh_sách_giá_trị** vào bảng.

```
INSERT INTO tên_bảng [ (danh_sách_cột) ]
truy_vấn_SELECT ;
```

168

INSERT

Ví dụ: Thêm giá trị vào bảng NhanVien (trường DiaChi có giá trị NULL)

```
insert into nhanvien (manv, honv, tennv,
                      gioitinh, ngaysinh)
values ('nv001', N'Nguyễn Văn', N'An',
       1, '1990-09-19'),
       ('nv002', N'Trần Thị', N'Bình', 0, '1995-11-09'),
       ('nv003', N'Lê Hoàng', N'Châu', 1, '1993-02-24');
```

169

INSERT

Ví dụ: Tạo bảng tên **KhachHang_Backup** có cấu trúc giống hệt bảng **KhachHang**.

```
insert into khachhang_backup
select * from khachhang;
hoặc
select * into khachhang_backup
from khachhang;
```

170

UPDATE

UPDATE tên_bảng

```
SET tên_cột_1 = biểu_thức_1
[ ,...n ]
[ FROM danh_sách_bảng_tác_động ]
[ WHERE điều_kiện ]
```

❖ Nếu không chỉ định **điều_kiện** thì toàn bộ các bản ghi trong bảng sẽ được cập nhật.

171

UPDATE

Ví dụ: Cập nhật tên hàng hóa

```
update hanghoa
set tenhh = N'Dell XPS 13'
where mahh = 'hh005';
```

172

DELETE

```
DELETE [ TOP số_lượng [ PERCENT ] ]
FROM tên_bảng
[ FROM danh_sách_bảng_tác_động ]
[ WHERE điều_kiện ]
;
```

❖ Nếu không chỉ định **điều_kiện** thì toàn bộ các bản ghi trong bảng sẽ bị xóa.



173

DELETE

Ví dụ: Xóa 2 bản ghi đầu tiên trong bảng **dondathang** của các nhân viên tên Châu

```
delete top 2 from dondathang
from dondathang as ddh join nhanvien as nv
on ddh.manv = nv.manv
where nv.tenvn like N'Châu';
```



174

TRUNCATE

❖ TRUNCATE chỉ được sử dụng khi muốn xóa toàn bộ dữ liệu của bảng

```
TRUNCATE TABLE tên_bảng;
```

❖ Dữ liệu bị xóa bởi DELETE có thể phục hồi được còn TRUNCATE thì không do dữ liệu bị xóa bởi TRUNCATE sẽ bị reset ở **Transaction Log** (do đó khi tạo 1 bản ghi mới, thứ tự bản ghi sẽ bắt đầu từ 1)

❖ TRUNCATE thực hiện nhanh hơn DELETE



175

NGÔN NGỮ ĐIỀU KHIỂN DỮ LIỆU DATA CONTROL LANGUAGE - DCL



176

NGÔN NGỮ ĐIỀU KHIỂN DỮ LIỆU

- ❖ Thiết lập quyền truy cập trên các đối tượng CSDL.
- ❖ Sử dụng để bảo mật CSDL.
- ❖ Câu lệnh: **GRANT, REVOKE** và **DENY**



177

BẢO MẬT TRONG SQL SERVER

3 lớp

- ❖ **Login security:** đăng nhập vào SQL Server
- ❖ **Database access security:** user có thể truy cập vào một CSDL cụ thể trên server.
- ❖ **Permission security:** user có thể thực hiện thao tác gì trên CSDL



178

BẢO MẬT TRONG SQL SERVER

- ❖ SQL Server sử dụng **Permission** và **Role** để bảo mật CSDL
 - **Permission:** Quy định các hành động mà người dùng thực hiện trên các đối tượng CSDL.
 - **Role:** tập các quyền được gán cho người dùng.
- ❖ SQL server dựa vào Permission và Role để xác định các đối tượng, hành động mà người dùng được phép thực hiện trên CSDL



179

TÀI KHOẢN ĐĂNG NHẬP (LOGIN)

- ❖ Lệnh CREATE LOGIN được dùng để tạo tài khoản đăng nhập (Login) kết nối tới SQL Server. Tài khoản đăng nhập sau đó sẽ được ánh xạ vào tài khoản người dùng.
- ❖ 2 loại tài khoản đăng nhập chính:
 - Tài khoản đăng nhập sử dụng xác thực SQL Server Authentication.
 - Tài khoản đăng nhập sử dụng xác thực Windows Authentication.



180

TÀI KHOẢN ĐĂNG NHẬP (LOGIN)

❖ Tạo tài khoản đăng nhập sử dụng xác thực SQL Server Authentication

```
CREATE LOGIN tên_login
WITH PASSWORD = 'mật_khẩu'
[ MUST_CHANGE ,CHECK_EXPIRATION=ON ]
[, DEFAULT_DATABASE = tên_CSDL ]
[, DEFAULT_LANGUAGE = tên_ngôn_ngữ ]
;
```

181

TÀI KHOẢN ĐĂNG NHẬP (LOGIN)

Lưu ý: Để khai báo CSDL mặc định (DEFAULT_DATABASE) khác với master cần phải chuyển owner của CSDL cho tên_login bằng câu lệnh

```
ALTER AUTHORIZATION
ON DATABASE::tên_CSDL TO tên_login ;
```

182

TÀI KHOẢN ĐĂNG NHẬP (LOGIN)

❖ Tạo tài khoản đăng nhập sử dụng xác thực Windows Authentication

Trong Windows, trong **CMD (Admin)** hoặc **Windows PowerShell (Admin)**:

- Xem tài khoản Windows
`net user`
- Tạo tài khoản Windows
`net user tên_tk mật_khẩu /add`
- Xem hostname
`hostname`

183

TÀI KHOẢN ĐĂNG NHẬP (LOGIN)

❖ Tạo tài khoản đăng nhập sử dụng xác thực Windows Authentication

Sử dụng **SQL Server Management Studio**

```
CREATE LOGIN "hostname\姓名"
FROM WINDOWS
[ WITH DEFAULT_DATABASE = tên_CSDL ]
[, DEFAULT_LANGUAGE = tên_ngôn_ngữ ]
;
```

184

TÀI KHOẢN ĐĂNG NHẬP (LOGIN)

❖ Sửa đổi tài khoản đăng nhập

```
ALTER LOGIN tên_login [ ENABLE | DISABLE ]
| [ WITH [ PASSWORD = 'mật_khẩu_mới'
          [ OLD_PASSWORD = 'mật_khẩu_cũ' ]
          [ MUST_CHANGE ] [ UNLOCK ]
      ]
      [ ,DEFAULT_DATABASE = tên_CSDL ]
      [ ,DEFAULT_LANGUAGE = tên_ngôn_ngữ ]
      [ ,NAME = tên_login_mới ]
      [ ,CHECK_EXPIRATION = ON | OFF ]
];

```

185

TÀI KHOẢN ĐĂNG NHẬP (LOGIN)

❖ Xóa tài khoản đăng nhập

```
DROP LOGIN tên_login ;
```

186

TÀI KHOẢN ĐĂNG NHẬP (LOGIN)

❖ Chuyển đổi tài khoản đăng nhập

```
EXECUTE | EXEC AS LOGIN = 'tên_login'
[ WITH NO REVERT ];
```

187

TÀI KHOẢN ĐĂNG NHẬP (LOGIN)

Ví dụ: Liệt kê danh sách tài khoản đăng nhập

```
SELECT name AS Login_Name,
       type_desc AS Account_Type
  FROM sys.server_principals
 WHERE TYPE IN ('U', 'S', 'G')
   and name not like '%##%'
 ORDER BY name, type_desc;
```

188

TÀI KHOẢN ĐĂNG NHẬP (LOGIN)

Ví dụ: Hiển thị tài khoản đăng nhập hiện tại

```
SELECT
SUSER_NAME() AS ProcessLoginName,
SUSER_NAME(1) AS AdminLoginName,
SUSER_SNAME(SUSER_SID()) AS SystemLoginName;
```

189

NGƯỜI DÙNG (USER)

- ❖ SQL Server cho phép truy nhập vào hệ thống thông qua các login nhưng chưa truy nhập được vào các CSDL chứa trong đó.
- ❖ Mỗi CSDL duy trì một danh sách các user, các user ánh xạ (mapped) với một login ở mức server. Khi đăng nhập vào SQL Server thông qua login thì sẽ có quyền truy nhập vào CSDL thông qua user với các quyền hạn xác định.

190

NGƯỜI DÙNG (USER)

❖ Tạo người dùng

```
CREATE USER tên_user
[ <FOR | FROM> LOGIN tên_login | WITHOUT LOGIN ]
[ WITH [ PASSWORD = 'mật khẩu' ]
[, DEFAULT_SCHEMA = tên_schema ]
; ]
```

Lưu ý: Nếu sử dụng Windows Authentication thì tên_user và tên_login sẽ là "hostname\Tên_login"

191

NGƯỜI DÙNG (USER)

Ví dụ: Liệt kê danh sách người dùng

```
select * from sys.database_principals
where type not in ('A', 'G', 'R', 'X')
and sid is not null
and name != 'guest';
```

192

NGƯỜI DÙNG (USER)

Ví dụ: Hiển thị tên người dùng hiện tại

```
select CURRENT_USER;
hoặc
select USER_NAME();
```

193

NGƯỜI DÙNG (USER)

Ví dụ:

```
create login HS
    with password = '8fdkj13$nlv';
use QLSV2019;
create user user1 for login HS
    with default_schema = marketing;
```

194

NGƯỜI DÙNG (USER)

Ví dụ:

```
create user [HS\MyLogin]
for login [HS\MyLogin];
```

195

NGƯỜI DÙNG (USER)

Ví dụ:

```
use QLSV2019 ;
create user CustomApp without login ;
grant impersonate on user::CustomApp
to [HS\MyLogin] ;
```

196

NGƯỜI DÙNG (USER)

Ví dụ:

```
use QLSV2019 ;
create user user1
with password = 'RN92piTCh%$'
, default_schema = [dbo];
```

197

NGƯỜI DÙNG (USER)

- ❖ Sửa đổi user

```
ALTER USER tên_user
WITH [ NAME = tên_user_mới ]
[ , DEFAULT_SCHEMA = tên_schema | NULL ]
[ , LOGIN = tên_login ]
[ , PASSWORD = 'mật_khẩu'
[ OLD_PASSWORD = 'mật_khẩu_cũ' ]
];
```

198

NGƯỜI DÙNG (USER)

- ❖ Thiết lập / chuyển đổi user

```
EXECUTE | EXEC AS USER = 'tên_user' ;
```

199

NGƯỜI DÙNG (USER)

- ❖ Xóa user

```
DROP USER [ IF EXISTS ] 'tên_user'
```

200

GRANT

❖ Lệnh GRANT được sử dụng để cấp thêm đặc quyền mới cho người dùng cơ sở dữ liệu.

```

GRANT < privileges >  

ON < object >  

TO < users | roles | PUBLIC >  

[ WITH GRANT OPTION ]
```



201

GRANT

❖ **PRIVILEGES:** quyền thực hiện những thao tác được cấp phát cho người dùng trên các đối tượng CSDL.
Ví dụ: CREATE DATABASE, SELECT, INSERT, UPDATE, DELETE, EXECUTE, CREATE VIEW và **ALL**

❖ **OBJECT:** *một* đối tượng chịu tác động của các quyền.
Ví dụ: DATABASE, FUNCTION, STORED PROCEDURE, TABLE, VIEW.

❖ **USERS:** các người dùng hoặc nhóm người dùng của CSDL.

❖ **WITH GRANT OPTION:** cho phép USERS cấp các quyền trong PRIVILEGES cho người dùng khác.



202

GRANT

Ví dụ: Hiển thị các quyền hiện hành

- Quyền trên SERVER


```
select *  
from fn_my_permissions(NULL, 'SERVER');
```
- Quyền trên DATABASE


```
select *  
from fn_my_permissions(NULL, 'DATABASE');
```



203

GRANT

Ví dụ: Cấp phát cho người dùng có tên user1 quyền thực thi các câu lệnh SELECT, INSERT và UPDATE trên bảng NHANVIEN

```
grant SELECT, INSERT, UPDATE  
on NHANVIEN  
to user1;
```



204

GRANT

Ví dụ: Cho phép người dùng user1 quyền xem họ, tên, ngày sinh và giới tính của các nhân viên.

```
grant SELECT (honv, tennv, ngaysinh, gioitinh)
on NHANVIEN
to user1;
```

hoặc

```
grant SELECT
on object::NHANVIEN(honv, tennv, ngaysinh,
gioitinh)
to user1;
```

205

GRANT

Ví dụ: Cấp phát cho người dùng user1 quyền tạo bảng và khung nhìn

```
grant CREATE TABLE, CREATE VIEW
to user1;
```

206

GRANT

Ví dụ: Cho phép người dùng user1 quyền xem, thêm, sửa dữ liệu trên schema user1Schema

```
grant SELECT, INSERT, UPDATE
on schema::user1Schema
to user1;
```

207

GRANT

Ví dụ: Cấp phát cho người dùng user1 quyền CONTROL cho cơ sở dữ liệu QLSV2019

```
use QLSV2019;
grant control on database::QLSV2019
TO user1;
```

208

REVOKE

- ❖ Lệnh REVOKE được dùng để loại bỏ các quyền truy xuất CSDL mà trước đó được gán thông qua lệnh GRANT.

```

REVOKE [ GRANT OPTION FOR ] <privileges>  

ON <object>  

FROM | TO <users | PUBLIC >  

[ CASCADE ]
```



209

REVOKE

- ❖ **PRIVILEGES:** các đặc quyền muốn loại bỏ từ người dùng. Lệnh này cũng sẽ loại bỏ việc chuyển giao hoặc từ chối (DENY) của người dùng đối với quyền bị loại bỏ (REVOKE)
- ❖ **GRANT OPTION FOR:** loại bỏ khả năng chuyển giao quyền và các quyền được gán cho người dùng khác nhưng đặc quyền được gán trước đó của user sẽ không bị loại bỏ.
- ❖ **CASCADE:** loại bỏ đặc quyền của các người dùng được cấp bởi user có tùy chọn WITH GRANT OPTION



210

REVOKE

Ví dụ: Thu hồi quyền thực thi lệnh INSERT trên bảng NHANVIEN đối với người dùng user1.

```

revoke INSERT  

on NHANVIEN  

from user1;
```



211

REVOKE

Ví dụ: Thu hồi quyền đã cấp phát trên cột NGAYSINH (chỉ cho phép xem dữ liệu trên cột HODEM và TEN) trên bảng NHANVIEN đối với người dùng user1.

```

revoke SELECT  

on NHANVIEN(NGAYSINH)  

from user1;
```



212

REVOKE

Ví dụ: Người dùng A cấp phát quyền xem dữ liệu trên bảng R cho C:

```
grant SELECT on R to C
```

Người dùng B cấp phát quyền xem và bổ sung dữ liệu trên bảng R cho C:

```
grant SELECT, INSERT on R to C
```

Người dùng B thu hồi các quyền đã cấp trên bảng R của C: **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 do A cấp cho C và vẫn còn hiệu lực).

213

REVOKE

Ví dụ: Cấp phát quyền xem dữ liệu cho người dùng A trên bảng R

```
grant SELECT on R to A with grant option
```

Người dùng A cấp cho người dùng B quyền xem dữ liệu trên R (vì A trước đó có *WITH GRANT OPTION*)

```
grant SELECT on R to B
```

Thu hồi quyền đã cấp phát cho người dùng A

```
revoke SELECT on R from A CASCADE
```

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

214

REVOKE

Ví dụ: Với ví dụ trước, nếu thay câu lệnh

```
revoke SELECT  
on R from A CASCADE
```

bằng câu lệnh

```
revoke GRANT OPTION FOR SELECT  
on R from A CASCADE
```

thì B sẽ không còn quyền xem dữ liệu trên bảng R đồng thời A không thể chuyển tiếp quyền cấp phát cho những người dùng khác (tuy nhiên A vẫn còn quyền xem dữ liệu trên bảng R).

215

DENY

❖ Lệnh DENY được sử dụng để ngăn chặn việc người dùng có thể được cấp đặc quyền trong CSDL

```
DENY < privileges >  
ON < object >  
TO < users | PUBLIC >  
[ CASCADE ]
```

216

ROLE

- ❖ Role là tập hợp các đặc quyền hoặc quyền truy cập.
- ❖ Được sử dụng trong việc cấp hoặc thu hồi quyền cho nhiều người dùng.
- ❖ Phân loại:
 - *Server Role*
 - *Database Role*
 - *Application Role*

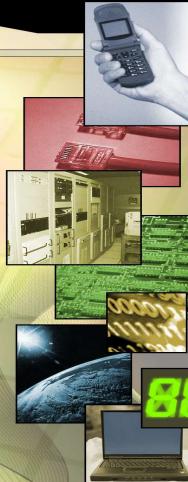


217

SERVER ROLE

- ❖ Nhóm các quyền ở mức server mà login khi được cấp sẽ có thể thực hiện một số thao tác xác định ở mức server.
- ❖ Server Roles cố định trong SQL Server

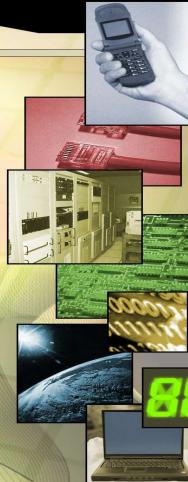
<ul style="list-style-type: none"> ▪ sysadmin ▪ bulkadmin ▪ dbcreator ▪ diskadmin 	<ul style="list-style-type: none"> ▪ processadmin ▪ securityadmin ▪ serveradmin ▪ setupadmin
---	--



218

SERVER ROLE

- ❖ Tạo Server Role mới
`use master;`
`CREATE SERVER ROLE tên_server_role`
`[AUTHORIZATION tên_login_cấp_quyền] ;`
- ❖ Gán thêm quyền cho role
`GRANT các_quyền_server TO tên_server_role`
`[WITH GRANT OPTION] ;`
- ❖ Gán role cho login | role khác
`ALTER SERVER ROLE tên_server_role`
`ADD MEMBER tên_login | tên_role_con ;`



219

SERVER ROLE

- ❖ Loại bỏ role của login | role khác
`ALTER SERVER ROLE tên_server_role`
`DROP MEMBER tên_login | tên_role_con ;`
- ❖ Đổi tên role
`ALTER SERVER ROLE tên_server_role`
`WITH NAME = tên_server_role_mới ;`
- ❖ Xóa role
`DROP SERVER ROLE tên_server_role;`



220

SERVER ROLE

Ví dụ: Tạo role cấp quyền kết nối, tạo, xem CSDL cho đăng nhập xyz

```
use master;
create server role myRole;
alter server role [myRole] add member [xyz];
grant alter trace to [myRole];
grant connect sql to [myRole];
grant create any database to [myRole];
grant view any database to [myRole];
grant view any definition to [myRole];
grant view server state to [myRole];
GO
```

221

SERVER ROLE

Ví dụ: Cấp quyền xử lý CSDL theo role dbcreator của SQL Server (CREATE, ALTER, DROP và RESTORE) cho đăng nhập xyz

```
use master;
create server role myRole;
alter server role [myRole]
    add member [xyz];
alter server role [dbcreator]
    add member [myRole];
GO
```

222

DATABASE ROLE

- ❖ Tập hợp các quyền truy xuất CSDL thành từng nhóm và được đại diện bởi một tên dùng để cấp phát quyền truy cập CSDL cho các users.
- ❖ Database Roles cố định trong SQL Server

<ul style="list-style-type: none"> ▪ db_owner ▪ db_securityadmin ▪ db_accessadmin ▪ db_backupoperator ▪ db_ddladmin 	<ul style="list-style-type: none"> ▪ db_datawriter ▪ db_datareader ▪ db_denydatawriter ▪ db_denydatareader
---	--

223

DATABASE ROLE

- ❖ **Tạo Database Role mới**
CREATE ROLE tên_database_role
[AUTHORIZATION tên_user_cấp_quyền] ;
- ❖ **Gán thêm quyền cho role**
GRANT các_quyền_database TO tên_database_role
[WITH GRANT OPTION] ;
- ❖ **Gán role cho user | role khác**
ALTER ROLE tên_database_role
ADD MEMBER tên_user | tên_role_con ;

224

DATABASE ROLE

- ❖ Loại bỏ role của user | role khác

```
ALTER ROLE tên_database_role
```

```
DROP MEMBER tên_user | tên_role_con ;
```

- ❖ Đổi tên role

```
ALTER ROLE tên_database_role
```

```
WITH NAME = tên_database_role_mới ;
```

- ❖ Xóa role

```
DROP ROLE tên_database_role;
```

225

APPLICATION ROLE

- ❖ Cho phép các ứng dụng thực thi trên CSDL, tương tự như một user với các quyền hạn được gán. Ta có thể sử dụng APPLICATION ROLE để cho phép truy cập tới các dữ liệu riêng biệt mà chỉ có một số user mới có quyền kết nối đến thông qua Application.

- ❖ Cú pháp tạo APPLICATION ROLE

```
CREATE APPLICATION ROLE tên_app_role
```

```
WITH PASSWORD = 'mật_khẩu'
```

```
[ , DEFAULT_SCHEMA = tên_schema ] ;
```

226

APPLICATION ROLE

- ❖ Gán quyền cho APPLICATION ROLE

```
GRANT ... TO tên_app_role ;
```

- ❖ Sửa đổi APPLICATION ROLE

```
ALTER APPLICATION ROLE tên_app_role
```

```
WITH [ NAME = tên_app_role_mới ]
```

```
[ , PASSWORD = 'mật_khẩu' ]
```

```
[ , DEFAULT_SCHEMA = tên_schema ] ;
```

- ❖ Xóa APPLICATION ROLE

```
DROP APPLICATION ROLE tên_app_role ;
```

227

APPLICATION ROLE

- ❖ Kích hoạt APPLICATION ROLE trong CSDL hiện hành

```
EXECUTE | EXEC sp_setapprole
```

```
[ @rolename = ] 'tên_app_role' ,
```

```
[ @password = ] 'mật_khẩu' ;
```

- ❖ Thủ tục sp_setapprole chỉ có thể được gọi trực tiếp bằng lệnh EXECUTE (EXEC). Nó không thể được thực thi bên trong 1 thủ tục khác hay từ 1 transaction của người dùng

228

SAO LƯU VÀ KHÔI PHỤC DỮ LIỆU BACKUP & RESTORE

229

SAO LƯU VÀ KHÔI PHỤC DỮ LIỆU

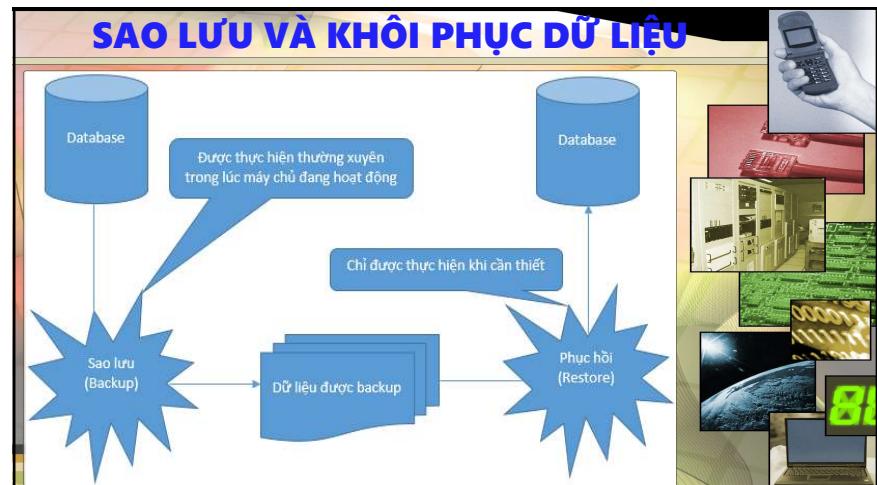
- ❖ Sao lưu (backup) một CSDL là tạo một bản sao CSDL và có thể dùng bản sao để khôi phục lại CSDL nếu CSDL bị mất hoặc hư hỏng. Bản sao gồm tất cả những file có trong CSDL kể cả transaction log.
- ❖ **Transaction log:** được sử dụng trong suốt quá trình khôi phục để roll forward những transaction hoàn thành và roll back những transaction chưa hoàn thành.

230

SAO LƯU VÀ KHÔI PHỤC DỮ LIỆU

- ❖ **Roll back:** là hủy bỏ giao dịch chưa hoàn thành khi hệ thống xảy ra sự cố (hoặc trong trường hợp sao lưu, khi đã thực hiện xong việc sao lưu mà giao dịch chưa hoàn thành).
- ❖ **Roll forward:** là khôi phục tất cả giao dịch đã hoàn thành khi hệ thống xảy ra sự cố (hoặc trong trường hợp sao lưu, những giao dịch đã hoàn thành khi đã thực hiện xong việc sao lưu)
- ❖ **Checkpoint:** là thời điểm ghi lại tất cả những trang dữ liệu thay đổi lên đĩa.

231



232

BACKUP

- ❖ **Full backup:** backup toàn bộ dữ liệu tại thời điểm đó và là backup dữ liệu đầy đủ nhất.

BACKUP DATABASE Tên_CSDL

TO DISK = 'Đường_dẫn\Tên_file_backup.bak' ;

- ❖ **Differential backup:** backup dữ liệu mới được cập nhật kể từ lần full backup trước đó.

BACKUP DATABASE Tên_CSDL

TO DISK = 'Đường_dẫn\Tên_file_backup.bak'

WITH DIFFERENTIAL ;



233

BACKUP

- ❖ **Transaction log backup:** backup các log record hiện có trong log file, nghĩa là nó sao lưu các hành động (các thao tác xảy ra đối với CSDL) chứ không sao lưu dữ liệu. Đồng thời nó cũng cắt bỏ (truncate) log file, loại bỏ các log record vừa được backup ra khỏi log file.

BACKUP LOG Tên_CSDL

TO DISK = 'Đường_dẫn\Tên_file_backup.trn' ;



234

BACKUP

- ❖ Một nguyên tắc chung để giảm bớt lượng dữ liệu mất mát khi có sự cố là tăng tần suất backup.
- ❖ Cần kết hợp full backup, differential backup và transaction log backup để tạo lập các phương án sao lưu thích hợp, đảm bảo dữ liệu được backup thường xuyên hơn mà không chiếm nhiều tài nguyên của hệ thống.



235

RESTORE

- ❖ Khôi phục là quá trình sao chép dữ liệu đã sao lưu và đưa các giao dịch được ghi lại vào dữ liệu của SQL Server.
- ❖ **Cú pháp**

RESTORE DATABASE | LOG Tên_CSDL

FROM DISK = 'Đường_dẫn\Tên_File'

[WITH NORECOVERY] ;



236

RESTORE

Ví dụ: Giả CSDL QLSV2019 bị hư hỏng, tiến hành khôi phục dữ liệu.

- Khôi dữ liệu từ full backup
`restore database QLSV2019
from disk = 'D:\bk\QLSV2019_full.bak'
with norecovery ;`
- Khôi phục dữ liệu từ different backup
`restore database QLSV2019
from disk = 'D:\bk\QLSV2019_Diff.bak'
with norecovery ;`
- Khôi phục dữ liệu từ log backup
`restore log QLSV2019
from disk = 'D:\bk\QLSV2019.trn';`



237

RESTORE

- ❖ Để có thể thực hiện khôi phục dữ liệu từ nhiều bản backup, phải chỉ định tùy chọn **WITH NORECOVERY** ở các câu lệnh restore **ngoại trừ** câu lệnh restore cuối cùng.
- ❖ Trong trường hợp file backup có nhiều phiên bản, ta có thể sử dụng tùy chọn **WITH FILE = position** để chọn phiên bản.
position có thể tìm được thông qua lệnh
RESTORE HEADERONLY
FROM DISK = 'Đường_dẫn\Tên_File';

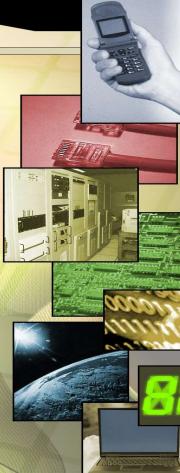


238

RESTORE

Ví dụ: Giả CSDL QLSV2019 bị hư hỏng, tiến hành khôi phục dữ liệu.

- Khôi dữ liệu từ full backup
`restore database QLSV2019
from disk = 'D:\bk\QLSV2019_full.bak'
with file = 3 norecovery ;`
- ...



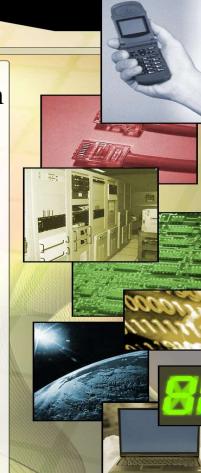
239

RESTORE

- ❖ Có thể thực hiện khôi phục dữ liệu tại một thời điểm backup bằng cách dùng tùy chọn **STOPAT = 'thời_gian'**

Ví dụ:

```
restore log QLSV2019  
from disk = 'D:\bk\QLSV2019.trn'  
with file = 3, norecovery,  
stopat = '2010-10-02 15:10:00';
```



240

RESTORE

❖ Nếu thực hiện khôi phục dữ liệu thay thế CSDL hiện tại ta sử dụng tùy chọn **WITH REPLACE**

Ví dụ:

```
restore database QLSV2019
from disk = 'D:\bk\QLSV2019_full.bak'
with replace ;
```

241

RESTORE

❖ Nếu thực hiện khôi phục dữ liệu và di chuyển CSDL đến nơi khác (copy CSDL), ta sử dụng tùy chọn **WITH MOVE 'file_logic' TO 'file_vật_lý'**

Ví dụ:

```
restore database QLSV2019
from disk = 'D:\bk\QLSV2019_full.bak'
with move 'QLSV2019_Data'
to 'D:\Data\QLSV2019.mdf',
move 'QLSV2019_Log'
to 'D:\Data\QLSV2019_Log.ldf';
```

242

VÍ DỤ BACKUP & RESTORE

❖ Kịch bản backup được thực hiện trên cơ sở dữ liệu QLSV2019

- Thực hiện **full backup** vào lúc **3h sáng ngày chủ nhật** (1 lần trong ngày)
- Thực hiện **different backup** vào lúc **2h sáng ngày thứ 4** (1 lần trong ngày)
- Thực hiện **log backup** vào các thời điểm **2h10', 2h20', 2h40' và 2h50'** (4 lần trong một ngày sau thời điểm different backup)

243

VÍ DỤ BACKUP & RESTORE

❖ Giả sử máy chủ bị hư ổ cứng vào lúc **2h55' ngày thứ 4**. Kịch bản phục hồi và tình trạng dữ liệu sau khi khôi phục:

- Đầu tiên, thực hiện khôi phục dữ liệu về thời điểm **3h sáng ngày chủ nhật** bằng cách sử dụng bản **full backup**.
- Tiếp theo, khôi phục dữ liệu về thời điểm **2h sáng thứ 4** sử dụng bản **different backup**.
- Cuối cùng khôi phục dữ liệu từ các bản **log backup** theo thứ tự thời điểm **2h10', 2h20', 2h40' và 2h50'** của ngày thứ 4 để đưa dữ liệu về thời điểm 2h50'.
- ❖ Như vậy dữ liệu chỉ được khôi phục trở lại ở thời điểm 2h50' ngày thứ 4 và sẽ bị mất dữ liệu từ 2h51' trở về sau.

244

LẬP TRÌNH T-SQL

245

GIỚI THIỆU

- ❖ Transact-SQL (T-SQL) là một ngôn ngữ lập trình CSDL hướng thủ tục độc quyền của Microsoft sử dụng trong SQL Server.
- ❖ Được thiết kế để mở rộng khả năng của SQL

GIỚI THIỆU

❖ T-SQL tổ chức theo từng khối lệnh, bắt đầu bởi **BEGIN** và kết thúc bởi **END**, các lệnh trong khối ngăn cách nhau bởi dấu ";"

BEGIN

- Khai báo biến
- Các câu lệnh T-SQL

END;

247

246

BIẾN CỤC BỘ

❖ Dùng để lưu trữ các giá trị tạm thời trong quá trình tính toán.

❖ Biến phải có kiểu dữ liệu.

❖ Biến muốn sử dụng trong một batch phải khai báo trước.

❖ Một biến cục bộ chỉ có phạm vi hoạt động trong một Batch, Stored Procedure hay Trigger

BIẾN CỤC BỘ

❖ Khai báo

```
DECLARE @Tên_biến Kiểu_dữ_liệu [=Giá_trị]
[ , ... ] ;
```

Ví dụ:

```
DECLARE @TenNV CHAR(50), @Ngaysinh DATE;
DECLARE @Tong INT = 0;
DECLARE @v_table TABLE (
    Định nghĩa bảng
);
```

249

BIẾN CỤC BỘ

❖ Gán giá trị cho biến

```
SET @Tên_biến = Giá_trị ;
SELECT @Tên_biến = Tên_Cột [...]
FROM Tên_Bảng [...] ;
```

Ví dụ:

```
SET @TenNV = N'Sơn';
SELECT
    @product_name = product_name,
    @list_price = list_price
FROM PRODUCTS
WHERE product_id = 100;
```

250

BIẾN CỤC BỘ

❖ Xem giá trị hiện hành của biến: sử dụng hàm CAST hoặc CONVERT

```
CAST(Biểu_thức AS Kiểu_dữ_liệu)
CONVERT(Kiểu_dữ_liệu, Biểu_thức [, Định_dạng])
```

Ví dụ:

```
declare @Ngay datetime = '2019-11-26';
declare @Soluong int = 5;
PRINT N'Ngày: ' + CONVERT(nvarchar(20),@Ngay,103);
PRINT N'Số lượng: ' + CAST(@Soluong as char(4));
```

251

BIẾN HỆ THỐNG

❖ Cung cấp các thông tin hệ thống

- @@Version - Phiên bản SQL Server
- @@ServerName - Tên Server
- @@Language - Ngôn ngữ sử dụng
- @@Connections - Tổng kết nối vào SQL Server
- @@Error - Mã lỗi của lệnh lỗi gần nhất
- @@Fetch_Status - Tình trạng đọc con trỏ (đọc thành công = 0)
- @@RowCount - Tổng số mẫu tin tác động bởi câu lệnh truy vấn gần nhất.

❖ Không cần khai báo

252

RẼ NHÁNH IF...ELSE...

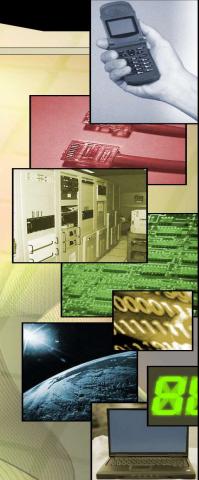
IF Biểu_thúc_logic
BEGIN
 `< Code xử lý khi Biểu_thúc_logic TRUE >`
END
[
ELSE
BEGIN
 `< Code xử lý khi Biểu_thúc_logic FALSE >`
END
]



253

LẶP WHILE

WHILE Biểu_thúc_logic
BEGIN
 `< Code xử lý khi Biểu_thúc_logic TRUE >`
END
 ♦ Vòng lặp sẽ dừng khi **Biểu_thúc_logic** có giá trị
FALSE



254

XỬ LÝ LỖI TRY...CATCH

- Thực hiện các lệnh trong khối TRY, nếu gặp lỗi sẽ chuyển qua xử lý bằng các lệnh trong khối CATCH

BEGIN TRY
 `< Code xử lý >`
END TRY
BEGIN CATCH
 `< Code xử lý khi phần TRY có lỗi >`
END CATCH



255

XỬ LÝ LỖI TRY...CATCH

- TRY và CATCH phải cùng lô xử lý
- Sau khối TRY phải là khối CATCH
- Có thể lồng nhiều cấp



256

XỬ LÝ LỖI TRY...CATCH

Ví dụ:

```
begin try
    select * from BangKhongTonTai;
end try
begin catch
    select
        ERROR_NUMBER() as ErrorNumber,
        ERROR_MESSAGE() as ErrorMessage;
end catch
```

257

XỬ LÝ LỖI TRY...CATCH

- ❖ Một số hàm ERROR thường dùng
 - **ERROR_NUMBER()** - Mã lỗi
 - **ERROR_MESSAGE()** - Thông báo lỗi
 - **ERROR_SEVERITY()** - Mức độ lỗi
 - **ERROR_STATE()** - Tình trạng lỗi
 - **ERROR_LINE()** - Dòng code gây ra lỗi
 - **ERROR_PROCEDURE()** - Tên thủ tục/ trigger gây ra lỗi

258

CON TRỎ (CURSOR)

- ❖ Các lệnh của SQL Server làm việc trên một nhóm nhiều mẫu tin
- ❖ Cursor là cấu trúc giúp làm việc từng mẫu tin tại một thời điểm
 - Khai báo như một câu lệnh SELECT
 - Di chuyển giữa các mẫu tin trong cursor để làm việc
 - Cập nhật dữ liệu (Update, Delete)

259

SỬ DỤNG CON TRỎ

- ❖ Định nghĩa biến kiểu cursor bằng lệnh **DECLARE**
 - Có hai loại cursor: **LOCAL, GLOBAL**
 - Cách di chuyển mẫu tin trong cursor: **FORWARD_ONLY, SCROLL**
 - Cách quản lý dữ liệu của cursor: **STATIC, DYNAMIC, KEYSET**

260

SỬ DỤNG CON TRỎ

- ❖ Sử dụng lệnh **OPEN** để mở ra cursor
- ❖ Đọc và xử lý trên từng dòng dữ liệu trong cursor:
 - Sử dụng biến **@@Fetch_Status**, các lệnh **FETCH** và cấu trúc **WHILE**
- ❖ Đóng cursor lại bằng lệnh **CLOSE** và **DEALLOCATE**
 - Sau khi CLOSE, có thể mở lại
 - DEALLOCATE: hủy cursor khỏi bộ nhớ



261

SỬ DỤNG CON TRỎ

```

DECLARE Tên_cursor CURSOR
[ LOCAL | GLOBAL ]
[ FORWARD_ONLY | SCROLL ]
[ STATIC | DYNAMIC | KEYSET ]
[ READ_ONLY | SCROLL_LOCK ]
FOR Câu_lệnh_SELECT
[ FOR UPDATE [ OF Danh_sách_cột_cập_nhật ] ]
;
  
```



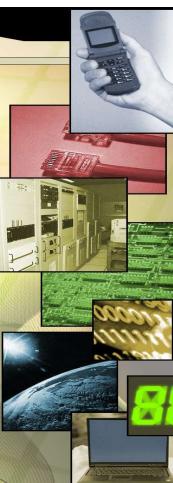
262

SỬ DỤNG CON TRỎ

```

FETCH [ NEXT | PRIOR | FIRST | LAST
      | ABSOLUTE n | RELATIVE n ]
FROM Tên_cursor
[ INTO Danh_sách_biến ] ;
  
```

- **Absolute n:** Đọc dòng thứ n trong cursor
- **Relative n:** Đọc dòng thứ n kể từ vị trí hiện hành



263

SỬ DỤNG CON TRỎ

Ví dụ:

```

--1. Khai báo cursor
declare cur_Vattu cursor keyset
for select * from VATTU
where MAVTU like 'TV%'
order by MAVTU ;

--2. Mở cursor
open cur_Vattu ;
  
```



264

SỬ DỤNG CON TRỎ

```
--3. Đọc dữ liệu
fetch next from cur_Vattu ;
while @@FETCH_STATUS = 0
begin
    -- Xử lý dòng mới vừa đọc được
    -- Thực hiện đọc tiếp các dòng kế
    fetch next from cur_Vattu ;
end ;

--4. Đóng cursor
close cur_Vattu ;
deallocate cur_Vattu ;
```

265

SỬ DỤNG CON TRỎ

Ví dụ: Khai báo con trỏ dạng biến

```
--1. Khai báo biến cursor
declare @cur_Vattu cursor ;
SET @cur_Vattu = CURSOR
for select * from VATTU
where MAVTU like 'TV%'
order by MAVTU ;
```

--2. Mở cursor

```
open @cur_Vattu ;
```

266

SỬ DỤNG CON TRỎ

```
--3. Đọc dữ liệu
fetch next from @cur_Vattu ;
while @@FETCH_STATUS = 0
begin
    -- Xử lý dòng mới vừa đọc được
    -- Thực hiện đọc tiếp các dòng kế
    fetch next from @cur_Vattu ;
end ;

--4. Đóng cursor
close @cur_Vattu ;
deallocate @cur_Vattu ;
```

267

THỦ TỤC (STORED PROCEDURE)

- ❖ Là tập hợp một hoặc nhiều câu lệnh T-SQL thành một nhóm đơn vị xử lý logic và được lưu trữ trên Database Server.
- ❖ Khi gọi stored procedure lần đầu tiên thì SQL Server sẽ thực thi và lưu trữ vào bộ nhớ đệm, gọi là **plan cache**, những lần gọi tiếp theo SQL Server sẽ sử dụng lại plan cache nên tốc độ xử lý tối ưu.

268

THỦ TỤC (STORED PROCEDURE)

CREATE PROCEDURE <Tên_thủ_tục>

```

    (
        @tham_số_1 Kiểu_dữ_liệu_1 [ OUTPUT ] ,
        @tham_số_2 Kiểu_dữ_liệu_2 [ OUTPUT ] ,
        ...
    )
    AS
    BEGIN
        -- Khai báo biến sử dụng
        -- Nội dung của thủ tục
    END;

```

269

THỦ TỤC (STORED PROCEDURE)

Ví dụ:

```

create procedure FindProductByModel (
    @model_year smallint,
    @product_count int output
) as
begin
    select product_name, list_price
    from production.products
    where model_year = @model_year;

    select @product_count = @@ROWCOUNT;
end;

```

270

THỦ TỤC (STORED PROCEDURE)

❖ **Gọi thủ tục**

```

EXECUTE | EXEC schema.Tên_thủ_tục
[ Giá trị_tham_số_1, Giá trị_tham_số_2, ... ]
hoặc
EXECUTE | EXEC schema.Tên_thủ_tục
[ @tham_số_1 = Giá trị_tham_số_1,
  @tham_số_2 = Giá trị_tham_số_2,
  ...
]

```

271

THỦ TỤC (STORED PROCEDURE)

Ví dụ:

```

declare @count int;

exec FindProductByModel
    @model_year = 2018
    @product_count = @count;

if (@count > 0)
begin
    print 'Number of products found';
end;

```

272

THỦ TỤC (STORED PROCEDURE)

- Thay đổi thủ tục
ALTER PROCEDURE <Tên_thủ_tục>
[
 @tham_số_1 Kiểu_dữ_liệu_1 [OUTPUT] ,
 @tham_số_2 Kiểu_dữ_liệu_2 [OUTPUT] ,
 ...
]
AS ...
- Xóa thủ tục
DROP PROCEDURE <Tên_thủ_tục>;

273

HÀM(FUNCTION)

- Tương tự như procedure nhưng Function có giá trị trả về
CREATE FUNCTION <Tên_hàm>
([
 @tham_số_1 Kiểu_dữ_liệu_1 [OUTPUT] ,
 @tham_số_2 Kiểu_dữ_liệu_2 [OUTPUT] ,
 ...
])
RETURNS Kiểu_dữ_liệu AS
BEGIN
 -- Khai báo biến sử dụng
 -- Nội dung của hàm
 RETURN Giá_trị_hàm ;
END;

274

HÀM(FUNCTION)

- Gọi hàm: gọi trực tiếp trong câu lệnh
schema.Tên_hàm (
 Giá_trị_tham_số_1,
 Giá_trị_tham_số_2,
 ...
)

275

HÀM(FUNCTION)

- Thay đổi hàm
ALTER FUNCTION <Tên_thủ_tục>
([
 @tham_số_1 Kiểu_dữ_liệu_1 [OUTPUT] ,
 @tham_số_2 Kiểu_dữ_liệu_2 [OUTPUT] ,
 ...
])
RETURNS ...
- Xóa hàm
DROP FUNCTION <Tên_hàm>;

276

HÀM(FUNCTION)

❖ Các hàm chuỗi cơ bản

- Đổi một số thành chuỗi
STR(Số_thực, Số_ký_tự [, Số_lè])

Ví dụ:

```
select str(12345.6789,8,2);
```

277

HÀM(FUNCTION)

❖ Các hàm chuỗi cơ bản

- Cắt chuỗi
LEFT(Chuỗi_nguồn, Số_ký_tự)
RIGHT(Chuỗi_nguồn, Số_ký_tự)
SUBSTRING(Chuỗi_nguồn, Vị_trí, Số_ký_tự)
- Tính chiều dài chuỗi
LEN(Chuỗi)
- Tìm và thay thế chuỗi
REPLACE(Chuỗi_nguồn, Chuỗi_tìm, Chuỗi_thay_thế)

279

HÀM(FUNCTION)

❖ Các hàm chuỗi cơ bản

- Đổi kiểu dữ liệu và định dạng
CONVERT(Kiểu_dữ_liệu, Biểu_thức [, Định_dạng])

yy	yyyy	Chuỗi kết quả
-	0 hoặc 100	mon dd yyyy hh:miAM (or PM)
1	101	mm/dd/yyyy
2	102	yy.mm.dd
3	103	dd/mm/yy
5	105	dd-mm-yy
6	106	dd mon yy
8	108	hh:mm:ss
9	109	mon dd yyyy hh:mi:ss:mmmAM (or PM)
10	110	mm-dd-yy
11	111	yy/mm/dd
12	112	yymmdd

278

HÀM(FUNCTION)

❖ Các hàm chuỗi cơ bản

- Cắt khoảng trắng
LTRIM(Chuỗi_nguồn, Số_ký_tự)
RTRIM(Chuỗi_nguồn, Số_ký_tự)
- Tạo chuỗi khoảng trắng
SPACE(Số_ký_tự)
- Đảo chuỗi
REVERSE(Chuỗi)
- Đổi số mã ASCII thành ký tự và ngược lại
CHAR(Số) **ASCII(Ký_tự)**

280

HÀM(FUNCTION)

Các hàm ngày giờ

- Lấy thời điểm hiện hành
GETDATE()
- Tính số chênh lệch giữa 2 mốc thời gian
DATEDIFF(Đơn_vị, Thời_gian_1, Thời_gian_2)
- Lấy tên đơn vị thời gian (cho kiểu chuỗi)
DATENAME(Đơn_vị, Thời_gian)
- Lấy đơn vị thời gian (cho kiểu số)
DATEPART(Đơn_vị, Thời_gian)

281

HÀM(FUNCTION)

Đơn_vị (Thành phần của ngày)	Chữ viết tắt
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

282

HÀM(FUNCTION)

Các hàm toán học (số)

- Làm tròn số
ROUND(Biểu_thức_số, Vị_trí_làm_tròn)
- Lấy số ngẫu nhiên >0 và <1
RAND([seed])
(Nếu có số seed thì số ngẫu nhiên sẽ cố định)

283

TRIGGER

Là thủ tục đặc biệt tự động thực hiện khi có sự kiện diễn ra trong CSDL server

Có 3 loại Trigger: DML, DDL, Logon

- DML Trigger: xảy ra khi có lệnh INSERT, UPDATE, DELETE trên bảng hoặc view.
- DDL Trigger: thực thi khi xảy ra lệnh CREATE, ALTER hoặc DROP.
- Logon Trigger: xảy ra khi session người dùng được thiết lập.

284

DML TRIGGER

SQL Server cung cấp 2 bảng ảo dành riêng cho trigger tên là INSERTED và DELETED, hai bảng này sẽ lưu trữ dữ liệu của các row trước hoặc sau khi hành động xảy ra.

Event	INSERTED	DELETED
INSERT	Dữ liệu của row vừa insert	Rỗng
UPDATE	Dữ liệu mới của row vừa update	Dữ liệu cũ của row vừa update
DELETE	Rỗng	Dữ liệu của row bị xóa

285

DML TRIGGER

```
CREATE [ OR ALTER ] TRIGGER [tên_schema.] tên_trigger
ON tên_table | tên_view
FOR | AFTER | INSTEAD OF
[ INSERT ] [,] [ UPDATE ] [,] [ DELETE ]
AS
BEGIN
...
END ;
```

286

DML TRIGGER

Ví dụ:

```
/* cập nhật hàng trong kho sau khi đặt hàng */
CREATE TRIGGER trg_DatHang ON tbl_DatHang
AFTER INSERT AS
BEGIN
    UPDATE tbl_KhoHang
    SET SoLuongTon = SoLuongTon - (
        SELECT SoLuongDat
        FROM inserted
        WHERE MaHang = tbl_KhoHang.MaHang
    )
    FROM tbl_KhoHang JOIN inserted ON
        tbl_KhoHang.MaHang = inserted.MaHang
END
```

287

DML TRIGGER

Ví dụ:

```
/* Gửi mail sau khi thao tác trên CSDL */
CREATE TRIGGER reminder2 ON Sales.Customer
AFTER INSERT, UPDATE, DELETE
AS
EXEC msdb.dbo.sp_send_dbmail
    @profile_name = 'James Bond',
    @recipients = 'jbond@mi5.com',
    @body = 'Mission completed',
    @subject = 'Reminder';
```

288

DML TRIGGER

Ví dụ:

```
CREATE TRIGGER trg_InsteadOfInsert ON tbl_Origin
INSTEAD OF INSERT
AS
INSERT INTO tbl_MyTable VALUES
((SELECT TOP 1 inserted.ID FROM inserted), N'Dã thêm')
```

289

DDL TRIGGER

**CREATE [OR ALTER] TRIGGER tên_trigger
ON ALL SERVER | DATABASE
FOR | AFTER tên_sự_kiện [,n]
AS
BEGIN
...
END ;**

290

DDL TRIGGER

Ví dụ:

```
CREATE TABLE index_logs (
    log_id INT IDENTITY PRIMARY KEY,
    event_data XML NOT NULL,
    changed_by SYSNAME NOT NULL
);
```

291

DDL TRIGGER

Ví dụ:

```
CREATE TRIGGER trg_index_changes
ON DATABASE
FOR
    CREATE_INDEX, ALTER_INDEX, DROP_INDEX
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO index_logs(event_data, changed_by)
    VALUES (EVENTDATA(), USER);

END;
```

292

DDL TRIGGER

Ví dụ:

```

CREATE TRIGGER trRestrictDDEvents
ON DATABASE
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
BEGIN
    PRINT 'You cannot create, alter or drop a table'
    ROLLBACK TRANSACTION
END

```

293

LOGON TRIGGER

CREATE [OR ALTER] TRIGGER tên_trigger
ON ALL SERVER [WITH EXECUTE AS tên_login]
FOR | AFTER LOGON
AS
BEGIN
...
END ;

294

LOGON TRIGGER

Ví dụ: Giới hạn giờ đăng nhập tài khoản test_user từ 10AM đến 6PM

```

CREATE TRIGGER Logon_from_10AM_to_6PM
ON ALL SERVER
FOR LOGON
AS
BEGIN
    IF ((ORIGINAL_LOGIN() = 'test_user') and
        ( (DATEPART(hour, GETDATE()) between 18 and 24)
        or (DATEPART(hour, GETDATE()) between 0 and 9)
        ))
        BEGIN
            ROLLBACK
        END
    END

```

295

LOGON TRIGGER

Ví dụ: Giới hạn tài khoản đăng nhập vào server

```

CREATE TRIGGER Prevent_login
ON ALL SERVER WITH EXECUTE AS 'sa'
FOR LOGON
AS
BEGIN
    DECLARE @LoginName sysname
    DECLARE @LoginType sysname

    SET @LoginName = ORIGINAL_LOGIN()
    IF(@LoginName NOT IN ('sa', 'son'))
        BEGIN
            ROLLBACK; --Disconnect the session
        END
    END

```

296

LOGON TRIGGER

Ví dụ: Giới hạn ứng dụng đăng nhập vào server

```
CREATE TRIGGER Prevent_login
ON ALL SERVER WITH EXECUTE AS 'sa'
FOR LOGON
AS
BEGIN
    DECLARE @AppName varchar(max)
    DECLARE @LoginName sysname
    DECLARE @LoginType sysname

    SET @AppName = APP_NAME()
    SET @LoginName = ORIGINAL_LOGIN()
    IF NOT (@LoginName = 'son' AND @AppName like 'SQLCMD')
    BEGIN
        ROLLBACK;
    END
END
```

297

LOGON TRIGGER

Ví dụ: Giới hạn kết nối server

```
CREATE TRIGGER Limit_Connection
ON ALL SERVER
FOR LOGON
AS
BEGIN
    IF(
        select COUNT(*) from sys.dm_exec_sessions
        where is_user_process = 1
    ) > 5
    BEGIN
        ROLLBACK; --Disconnect the session
    END
END
```

298

TRIGGER

- ❖ Kích hoạt | vô hiệu hóa Trigger
**ENABLE | DISABLE TRIGGER [tên_schema .]
tên_trigger [,...n] | ALL
ON tên đối tượng | DATABASE | ALL SERVER
[;]**
- ❖ Xóa Trigger
**DROP TRIGGER [IF EXISTS]
[tên_schema .] tên_trigger [,...n]
ON DATABASE | ALL SERVER
[;]**

299

TRANSACTION

- ❖ Transaction được dùng để đảm bảo tính toàn vẹn dữ liệu khi xảy ra cập nhật (INSERT, UPDATE, DELETE...), ngăn chặn tình huống dữ liệu được cập nhật nửa chừng.
- ❖ Cấu trúc transaction
 - *Bắt đầu transaction:* **begin tran | begin transaction**
 - *Kết thúc transaction:* **commit | commit tran | commit transaction**

300

TRANSACTION

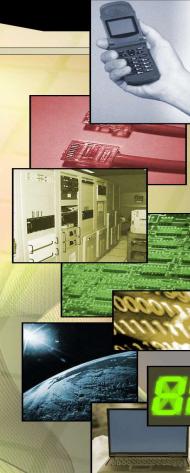
- *Quay lui transaction (Rollback transaction):*
rollback | rollback tran | rollback transaction
- *Đánh dấu một savepoint trong transaction:*
save transaction tên_savepoint



301

TRANSACTION

- Lệnh **rollback tên_savepoint** có tác dụng quay lui (rollback) giao dịch đến vị trí đặt savepoint tương ứng (không có tác dụng kết thúc transaction).
- Biến **@@trancount**: cho biết số transaction hiện đang thực hiện (chưa được kết thúc với rollback hay commit)



302

TRANSACTION

❖ Cú pháp

```

SET XACT_ABORT ON
BEGIN TRAN
  BEGIN TRY
    ...
    COMMIT
  END TRY
  BEGIN CATCH
    ROLLBACK
    DECLARE @ErrorMessage VARCHAR(2000)
    SELECT @ErrorMessage = ERROR_MESSAGE()
    RAISERROR(@ErrorMessage, 16, 1)
  END CATCH

```



303

GO

- ❖ SQL Server thực thi các lệnh theo từng nhóm lệnh (batch)
- ❖ Các nhóm lệnh phân cách bởi lệnh GO
- ❖ Các lệnh nằm đầu gói và không kết hợp với các lệnh khác: **CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE RULE, CREATE SCHEMA, CREATE TRIGGER, CREATE VIEW**
- ❖ Các lệnh trong gói lệnh phải phù hợp với thứ tự logic



304

GO

Ví dụ:

```
use SON
IF OBJECT_ID('cities') IS NOT NULL
    DROP TABLE cities;
CREATE TABLE cities (
    name VARCHAR(90) PRIMARY KEY
)
GO
DROP TABLE cities
```



305