Mỗi phần nên tạo gồm đủ các phần

* Lý thuyết là cái j
* Cú pháp
* Ví dụ minh họa
* Câu hỏi tương tác
* Bài tập thực thành

# SQL

## **Slide 4 - DDL – Data Definiton Language Ngôn ngữ định nghĩa dữ liệu**

**[Slide4]**

Giới thiệu về SQL, SQL - Structured Query Language: Ngôn ngữ truy vấn có cấu trúc

* SQL cho phép Tạo CSDL, Thao tác trên dữ liệu (Lưu trữ dữ liệu, Sửa dữ liệu, Xóa dữ liệu)
* Được ANSI và ISO chuẩn hóa
* Đa số các DBMS hiện nay sử dụng SQL (MS SQL
* Server – T- SQL, Microsoft Access, Oracle –
* PL/SQL, DB2, MySQL…)

Có thể chia thành 4 nhóm lệnh SQL:

* Nhóm truy vấn dữ liệu (DQL): gồm các lệnh truy vấn lựa chọn

(Select) để lấy thông tin nhưng không làm thay đổi dữ liệu trong các bảng

* Nhóm định nghĩa dữ liệu (DDL): Gồm các lệnh tạo, thay đổi các bảng dữ liệu(Create, Drop, Alter, …)
* Nhóm thao tác dữ liệu (DML): Gồm các lệnh làm thay đổi dữ liệu

(Insert, Delete, Update,…) lưu trong các bảng

* Nhóm điều khiển dữ liệu (DCL): Gồm các lệnh quản lý quyền truy nhập vào dữ liệu và các bảng (Grant, Revoke, …)

SQL không phân biệt chữ hoa, chữ thường. Ví dụ **Create** hay **CREATE** được hiểu như nhau

Hướng dẫn sử dụng công cụ trực quan SQL trong MySQL và SQL Server

### **Bảng, trường (cột)**

* Các bảng trong CSDL đều được tạo thành từ các cột (trường).
* Các thuộc tính của các cột mô tả đặc điểm và hành vi dữ liệu được đưa vào cột đó
* Kiểu dữ liệu là thuộc tính quan trọng nhất vì nó xác định loại dữ liệu mà cột có thể lưu trữ
* Chỉ nên sử dụng kiểu và kích cỡ của cột mà bạn thực sự muốn sử dụng;

Ví dụ: đừng định nghĩa một cột với độ rộng là 10 ký tự nếu bạn chỉ sử dụng 2 ký tự.

### **Các kiểu dữ liệu**

Kiểu dữ liệu là thuộc tính xác định loại dữ liệu trong bảng mà cột có thể lưu trữ

Có nhiều loại dữ liệu khác nhau như:

* Kiểu chuỗi
* Kiểu số
* Kiểu ngày và giờ
* Kiểu bit
* Kiểu tham chiếu
* Kiểu đối tượng
* …

So sánh giữa Access, MySQL, SQL Server

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **Access** | **MySQL** | **SQL Server** |  |
| Chuỗi ký tự |  |  | Char  Varchar  Text  Long text |  |  |
|  |  |  |  |  |  |
| Số | Số nguyên |  | Smallint  Int  Bigint |  |  |
|  | Số thực |  | Float  Double |  |  |
| Thời gian | Ngày tháng năm |  | Datetime  Date  Time |  |  |
|  | Ngày tháng năm giờ phút giây |  |  |  |  |
| Bit |  |  |  |  |  |
|  |  |  |  |  |  |
| Tham chiếu |  |  |  |  |  |
|  |  |  |  |  |  |
| Đối tượng |  |  |  |  |  |
|  |  |  |  |  |  |

### **Tạo cơ sở dữ liệu**

Ngôn ngữ định nghĩa dữ liệu (**DDL – Data Definition Language**) gồm các lệnh cho phép tạo ra, thay đổi hoặc xóa các đối tượng trong cơ sở dữ liệu

Các đối tượng bao gồm: Database, Table, Index, Sequence, Function, Procedure, Trigger, View

Chúng ta cũng có thể định nghĩa các khoá (key), chỉ mục (index), chỉ định các liên kết giữa các bảng và thiết lập các quan hệ ràng buộc giữa các bảng trong CSDL

Nguyên tắc đặt tên:

* Kí tự đầu tiên của một định danh phải là một kí tự chữ cái theo chuẩn Unicode 2.0, hoặc dấu (\_), hoặc dấu @ (tên biến), hoặc # (bảng tạm).
* Không trùng với các từ khoá và từ dành riêng của ngôn ngữ T-SQL.
* Không chứa các kí tự đặt biệt +, -, \*, /, !, ~, | ....

Ví dụ tên hợp lệ: Nhan\_vien, \_PhongBan

Tên không hợp lệ: [%], SELECT

#### **Tạo CSDL CREATE DATABASE <tên csdl>;**

#### **Tạo bảng CREATE TABLE <tên bảng> (**

<tên cột 1> <kiểu dữ liệu 1> [các ràng buộc],

<tên cột 2> <kiểu dữ liệu 2> [các ràng buộc],

.......);

* Có thể có một hoặc nhiều cột
* Mỗi cột chỉ có một kiểu dữ liệu duy nhất
* Mỗi cột có thể không hoặc có một hoặc nhiều ràng buộc: NULL NOT NULL PRIMARY KEY

Vẽ một bảng csdl đây và yêu cầu sinh viên tạo để làm quen, từ tạo bằng công cụ trực quan đến việc tạo bằng cú pháp SQL

### **Sửa ALTER TABLE**

Lệnh ALTER TABLE cho phép thay đổi các định nghĩa trên bảng như:

* Thêm/xóa cột trong bảng
* Thay đổi kiểu dữ liệu cho các cột trong bảng
* Thay đổi thuộc tính bộ nhớ cấp cho bảng
* Thêm/xóa/thay đổi các ràng buộc

#### **Thêm một cột ADD COLUMN:**

**ALTER TABLE** table\_name

**ADD COLUMN** column\_name data\_type;

#### **Xóa một cột DROP COLUMN:**

**ALTER TABLE** table\_name

**DROP COLUMN** column\_name;

Ví dụ:

ALTER TABLE NHAN\_VIEN

ADD COLUMN EMAIL VARCHAR(50) ;

#### **Thêm ràng buộc ADD CONSTRAINT**

Ràng buộc là các qui tắc để hạn chế các giá trị được lưu trữ vào bảng

* Các ràng buộc được kiểm tra trước khi một hàng mới thêm vào bảng hoặc cập nhật hàng

Cú pháp:

**ALTER TABLE** <tên bảng>

**ADD CONSTRAINT** <tên ràng buộc>

**CHECK** (điều kiện);

Ví dụ:

Thêm một ràng buộc kiểm tra - Check:

ALTER TABLE NHAN\_VIEN

ADD CONSTRAINT CHK\_SALARY\_MIN

CHECK (LUONG >= 100);

Thêm một ràng buộc Khoá chính- Primary key:

ALTER TABLE NHAN\_VIEN

ADD CONSTRAINT PRI\_NhanVien

PRIMARY KEY (ID\_NHANVIEN);

Định nghĩa một ràng buộc tham chiếu có tên EMP\_DEPT\_FK sẽ được thêm vào bảng NHAN\_VIEN để định nghĩa cột PHG là khóa ngoại tương ứng với cột khóa chính (MA\_PB) của bảng PHONG\_BAN

ALTER TABLE NHAN\_VIEN

ADD CONSTRAINT EMP\_DEPT\_FK

FOREIGN KEY (PHG )

REFERENCES PHONG\_BAN (MA\_PB);

Thêm ràng buộc UNIQUE

ALTER TABLE NHAN\_VIEN

ADD CONSTRAINT NHANVIEN\_UNQ\_EMAIL

UNIQUE (EMAIL);

Ràng buộc UNIQUE có thể được loại bỏ với lệnh ALTER

ALTER TABLE EMPLOYEE\_INPUT

DROP CONSTRAINT EMPLOYEES\_UNQ\_EMAIL;

Câu hỏi

* Thêm ràng buộc khoá chính và khoá ngoại cho các bảng trong csdl quản lý nhân viên
* Thêm 1 cột có tên Mo\_ta vào bảng phòng ban để chứa phần mô tả về phòng ban
* Thêm ràng buộc kiểm tra vào bảng QUANLY\_DUAN yêu cầu cột số giờ chỉ chứa giá trị lớn hơn 0

### **Xóa DROP**

Dùng để xóa các đối tượng không cần thiết khỏi CSDL: ràng buộc, cột, bảng, csdl

#### **Xóa CDSL DROP DATABASE**

**DROP DATABASE** <tên CSDL>

#### **Xóa bảng DROP TABLE**

**DROP TABLE** <tên bảng>

Lưu ý: Bảng muốn xóa phải đảm bảo không có các ràng buộc quan hệ

Ví dụ:

**DROP TABLE** NHAN\_VIEN **CASCADE CONSTRAINTS**;

(mệnh đề CASCADE CONSTRAINTS được thêm vào để tự động loại bỏ các ràng buộc tham chiếu trong bảng )

## **Slide 5 - DQL – Data Querry Language Ngôn ngữ truy vấn dữ liệu**

**[Slide5]**

Bao gồm các lệnh cho phép truy vấn dữ liệu mà không làm thay đổi dữ liệu hoặc các đối tượng trong CSDL. Là các truy vấn bắt đầu bằng từ khóa SELECT. Trả về một bộ các thuộc tính hoặc một tập hợp các bộ thuộc tính

* Ngôn ngữ truy vấn SQL được sử dụng để thao tác với hệ quản trị cơ sở dữ liệu
* Phân loại ngôn ngữ SQL: DML, DDL, DCL và DQL
* Ngôn ngữ truy vấn dữ liệu là các lệnh cho phép truy vấn dữ liệu mà không

làm thay đổi dữ liệu hoặc các đối tượng trong CSDL

### **SELECT – Truy vấn dữ liệu**

Lọc dữ liệu từ bảng theo điều kiện

**SELECT** [**ALL** | **DISTINCT** | **TOP**] <danh sách cột>

**FROM** <danh sách bảng>

[**WHERE <**điều kiện lọc>]

[**ORDER BY** <cột> **DESC | ASC**]

[**GROUP BY** <cột>]

[**HAVING** <điều điện nhóm dựa trên group by>;]

Các mệnh đề trong cặp dấu [] không bắt buộc, danh sách cột là tên các cột cách nhau bởi dấu ,

Mặc định là ALL, DISTINCT là trả về các bản ghi không trùng lặp nhau, TOP dùng với SQL Server trả về n (hay %) bản ghi từ trên xuống, MySQL dùng LIMIT ở cuối

WHERE là điều kiện lọc dữ liệu theo hàng

ORDER BY là sắp xếp dữ liệu theo cột, mặc định là ASC nếu bị bỏ trống là tăng dần, DESC là giảm dần

GROUP BY HAVING là nhóm dữ liệu theo hàng kết hợp theo điều kiện tại HAVING

Ví dụ,

**SELECT** city **FROM** sales.customers **ORDER BY** city;

**SELECT DISTINCT** city **FROM** sales.customers **ORDER BY** city;

#### **DISTINCT – Lọc dữ liệu loại bỏ trùng lặp**

Khi trong lệnh SELECT trả về có nhiều kết quả trùng lặp, mà bạn muốn loại bỏ, sử dụng từ khóa DISTINCT

**SELECT** **DISTINCT** column\_names

**FROM** table\_name;

Ví dụ, trong bảng customer, có nhiều khách hàng, trong đó có những khách hàng trùng thành phố. Nếu chỉ SELECT city trong bảng đó, ta sẽ nhận được về nhiều thành phố trùng lặp DISTINCT giải quyết vấn đề

**SELECT** city **FROM** sales.customers **ORDER BY** city;

**SELECT DISTINCT** city **FROM** sales.customers **ORDER BY** city;

#### **WHERE - Điều kiện lọc**

Sử dụng WHERE để đưa ra các điều kiện lọc CSDL giữ lại các bản ghi thỏa mãn

Tại WHERE sử dụng các toán tử điều kiện như

* Toán tử so sánh : > , < , >= , <= , <>
* Toán tử logic : AND, OR, NOT
* So sánh xâu dùng toán tử LIKE
* BETWEEN … AND
* IN : Tương đương nhiều toán tử OR, đúng nếu giá trị biểu thức nằm trong danh sách

**SELECT** product\_id, product\_name, category\_id, model\_year, list\_price

**FROM** production.products

**WHERE** category\_id = 1

**ORDER BY list\_price DESC;**

##### ***Toán tử LIKE - giống như***

Toán tử LIKE được sử dụng trong mệnh đề WHERE để tìm kiếm một mẫu được chỉ định trong một cột.

Có hai ký tự đại diện thường được dùng kèm toán tử LIKE là % và \_

* Dấu phần trăm (%) đại diện cho không, một hoặc nhiều ký tự
* Dấu gạch dưới (\_) đại diện cho một ký tự đơn
* [] đại diện cho các ký tự đơn được liệt kê trong khoảng
* ^[] đại diện cho các ký tự đơn không được liệt kê trong khoảng

**SELECT** <danh sách cột>

**FROM** table\_name

**WHERE** columnN LIKE pattern;

|  |  |
| --- | --- |
| Toán tử LIKE | Mô tả |
| WHERE tenKhachHang LIKE 'a%' | Tìm bất kỳ giá trị bắt đầu với a |
| WHERE tenKhachHang LIKE '%a' | Tìm bất kỳ giá trị kết thúc bởi a |
| WHERE tenKhachHang LIKE '%or%' | Tìm bất kỳ giá trị có chứa or ở vị trí bất kỳ |
| WHERE tenKhachHang LIKE '\_r%' | Tìm bất kỳ giá trị có chứa r ở vị trí thứ 2 |
| WHERE tenKhachHang LIKE 'a\_%' | Tìm bất kỳ giá trị bắt đầu bởi a và có chiều dài ít nhất 2 ký tự |
| WHERE tenKhachHang LIKE 'a\_\_%' | Tìm bất kỳ giá trị bắt đầu bởi a và có chiều dài ít nhất 3 ký tự |
| WHERE tenKhachHang LIKE 'a%o' | Tìm bất kỳ giá trị bắt đầu bởi a và kết thúc bởi o |
| SELECT \* FROM NHAN\_VIEN WHERE TEN\_NV LIKE `[HT]%` | Danh sách nhân viên có tên bắt đầu bằng HT |
| SELECT \* FROM NHAN\_VIEN WHERE TEN\_NV LIKE `^[HT]%` | Danh sách nhân viên có tên không bắt đầu bằng HT |

Ví dụ: ***SELECT \* FROM Customers WHERE CustomerName NOT LIKE 'a%';***

#### **ORDER BY - sắp xếp kết quả trả về**

**ORDER BY** được sử dụng trong SELECT, kết quả trả về sẽ được sắp xếp theo cột chỉ định tăng dần (**ASC**) hoặc giảm dần (**DESC**). Nếu có nhiều thuộc tính sau ORDER BY, sắp xếp được thực hiện ưu tiên từ cột đầu tiên, sau đó mới đến các cột tiếp theo.

Mặc định ORDER BY sắp xếp theo thứ tự tăng dần, nên bạn có thể không chỉ định ASC cũng được.

**SELECT** <danh sách cột>

**FROM** <tên bảng>

**ORDER BY** <cột> DESC;

#### **GROUP BY - HAVING - gom nhóm**

Mệnh đề GROUP BY được dùng kết hợp với các hàm kết tập để nhóm kết quả theo một hoặc nhiều cột. (Kiểu như group by cái gì, phải có hàm tính toán liên quan đến cái đó)

**SELECT** <tên cột 1> [, tên cột 2> ,…], <hàm kết tập(tên cột 1)>, [, <hàm kết tập(tên cột 2)>, …]

**FROM** <tên bảng 1> [, <tên bảng 2>, …]

[ **WHERE** <điều kiện> ]

**GROUP BY** <tên cột 1> [, <tên cột 2>, …];

Ví dụ: Tìm số phi công của mỗi nước (nhóm theo từng nước)

**SELECT** nuoc, count(\*) so\_phi\_cong

**FROM** PHICONG

**GROUP BY** nuoc;

Mệnh đề WHERE không thể được dùng với các hàm kết tập -> Điều kiện trên nhóm với hàm kết tập -> dùng HAVING (WHERE : điều kiện trên dòng).

HAVING chỉ hiểu các điều kiện trên các hàm kết tập.

Ví dụ 9: Số phi công của mỗi nước lớn hơn 3 phi công

**SELECT** nuoc, count(\*) so\_phi\_cong

**FROM** PHICONG

**GROUP BY** nuoc **HAVING** count(\*) >= 3;

#### **Mệnh đề SELECT TOP (LIMIT) - giới hạn số bản ghi trả về**

Mệnh đề TOP (LIMIT mới MySQL) được sử dụng để chỉ định số bản ghi trả về, đặc biệt hữu ích khi làm việc trên CSDL có số lượng bản ghi lớn, nếu cùng một lúc truy vấn trả về số lượng lớn bản ghi sẽ ảnh hưởng đến tốc độ truy vấn

SELECT TOP number|percent column\_name(s)

FROM table\_name

WHERE condition;

Với MySQL

SELECT column\_name(s)

FROM table\_name

WHERE condition

LIMIT number;

## **Slide 6 - Truy vấn trên nhiều bảng**

**[Slide6]**

Đơn giản nhất của truy vấn trên nhiều bảng là đặt nhiều bảng trong FROM, vẫn còn được gọi là phép tích. Kết quả phép truy vấn là tổ hợp các phần tử giữa các bảng

Cú pháp:

**SELECT** <danh sách cột>

**FROM** <danh sách bảng>

Ví dụ

Nếu được đặt điều kiện truy vấn bằng WHERE theo khóa chính khóa ngoại, kết quả là phần giao hai tập hợp

Cú pháp:

**SELECT** <danh sách cột>

**FROM** <danh sách bảng>

**WHERE** <tên bảng 1>.<tên cột> = <tên bảng 2>.<tên cột>

Nếu có nhiều bảng thì cần có nhiều điều kiện tương ứng dưới WHERE, các điều kiện là phép giao dùng toán tử **AND**

Ví dụ

**SELECT** DU\_AN.TEN\_DUAN, NHAN\_VIEN.HO\_NV, NHAN\_VIEN.TEN\_NV, QUANLY\_DUAN.NGAY\_THAM\_GIA, QUANLY\_DUAN.NGAY\_KET\_THUC

**FROM** NHAN\_VIEN , DU\_AN , QUANLY\_DUAN

**WHERE** (DU\_AN.MA\_DUAN = QUANLY\_DUAN.MA\_DUAN )

**AND** (NHAN\_VIEN.ID\_NhanVien = QUANLY\_DUAN.MA\_NHANVIEN)

### **JOIN – kết nối bảng**

Mệnh đề **JOIN** được sử dụng để kết hợp các hàng từ hai hoặc nhiều bảng, dựa trên một cột có liên quan giữa chúng.

Ví dụ:

**SELECT** \* **FROM** phicong **JOIN** lamviec **ON** phicong.mpc = lamviec.mpc;

**SELECT** phicong.hoten, lamviec.songay **FROM** phicong **JOIN** lamviec **ON** phicong.mpc = lamviec.mpc;

Các kiểu **JOIN** khác nhau

**(INNER) JOIN**: Trả về các bản ghi có giá trị khớp trong cả hai bảng

**LEFT (OUTER) JOIN**: Trả về tất cả các bản ghi từ bảng bên trái và các bản ghi phù hợp từ bảng bên phải

**RIGHT (OUTER) JOIN**: Trả về tất cả các bản ghi từ bảng bên phải và các bản ghi phù hợp từ bảng bên trái

**FULL (OUTER) JOIN**: Trả về tất cả các bản ghi khi có sự trùng khớp trong bảng bên trái hoặc bên phải

Ví dụ: Tìm họ tên, MPC, địa chỉ và số ngày làm việc của các phi công sống ở Anh làm việc cho công ty mã số 1 kể cả trường hợp các phi công ở Anh không làm việc cho công ty 1 và cả trường hợp công ty này không có phi công ở Anh làm việc.

**SELECT** lv.MPC, hoten,dchi, songay

**FROM** (**SELECT** \* **FROM** PHICONG **WHERE** nuoc **LIKE** 'Anh') **AS** pc

**FULL JOIN** (**SELECT** \* **FROM** LAMVIEC **WHERE** MCT = 1) **AS** lv

**ON** pc.MPC = lv.MPC ;

Ví dụ kết nối ba bảng

**SELECT** Orders.OrderID, Customers.CustomerName, Shippers.ShipperName

**FROM** ((Orders

**INNER JOIN** Customers **ON** Orders.CustomerID = Customers.CustomerID)

**INNER JOIN** Shippers **ON** Orders.ShipperID = Shippers.ShipperID);

Một ví dụ kết nối một bảng vào chính nó: >>> những khách hàng đến từ cùng một thành phố:

**SELECT** A.CustomerName **AS** CustomerName1, B.CustomerName **AS** CustomerName2, A.City

**FROM** Customers A, Customers B

**WHERE** A.CustomerID <> B.CustomerID

**AND** A.City = B.City

**ORDER** **BY** A.City;

#### **AS – Bí danh**

Bí danh SQL được sử dụng để đặt tên tạm thời cho **bảng** hoặc **cột** trong bảng.

Ví dụ:

**SELECT** column\_name **AS** alias\_name **FROM** table\_name;

**SELECT** column\_name(s) **FROM** table\_name **AS** alias\_name;

Bí danh thường được sử dụng để làm cho tên cột dễ đọc hơn.

Bí danh chỉ tồn tại trong khoảng thời gian của truy vấn đó.

Bí danh được tạo bằng từ khóa **AS**.

Lưu ý: Nó yêu cầu dấu ngoặc kép, nháy đơn hoặc dấu ngoặc vuông với SQL Server nếu tên bí danh chứa dấu cách:

**SELECT** CustomerName **AS** “Họ tên”, ContactName **AS** [Contact Person]

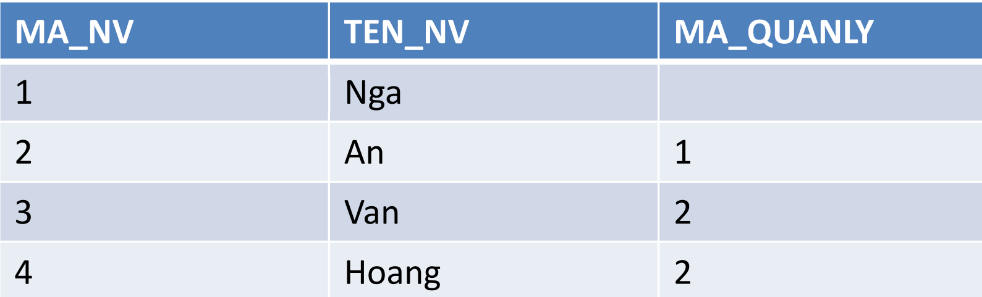
**FROM** Customers;

Bí danh đặc biệt cần thiết trong các quan hệ đệ quy

Một bảng kết nối với chính nó

Ví dụ:

Hiển thị tên nhân viên và tên người Quản Lý của anh ấy:



**SELECT** A.MA\_NV, A.TEN\_NV, B.TEN\_NV AS `Quản lý`

**FROM** NHAN\_VIEN **AS** A **INNER JOIN** NHAN\_VIEN **AS** B

**ON** B.MA\_NV = A.MA\_QUANLY

### **Truy vấn con, truy vấn trong truy vấn**

Một truy vấn con là một câu lệnh SELECT được lồng trong một SELECT, INSERT, DELETE, UPDATE hoặc một SELECT con khác.

**SELECT** … **FROM** ...

**WHERE** [… AND] <điều kiện với truy vấn con>

(**SELECT** con)

Ví dụ: danh sách sách có số lượng nhỏ hơn số lượng trung bình

**SELECT** sach.ma\_sach, sach.ten\_sach, sach.giatien, sach.ghichu

**FROM** Sach **JOIN** donhang

**ON** sach.ma\_sach = donhang.ma\_sach

**WHERE** donhang.soluong <

(**SELECT AVG**(donhang.soluong) **FROM** donhang ); -- trả về 1 giá trị trung bình

Truy vấn con có ràng buộc:

* Mệnh đề ORDER BY không được dùng trong truy vấn con
* Câu truy vấn con phải được bao trong cặp dấu ngoặc đơn ()
* Mệnh đề SELECT của truy vấn con chỉ bao gồm một thuộc tính duy nhất trừ trường hợp dùng EXISTS
* Thuộc tính trong điều kiện với truy vấn con và thuộc tính trong mệnh đề SELECT của truy vấn con phải tương thích trừ trường hợp dùng EXISTS
* Các thuộc tính được định nghĩa trong SELECT chính có thể được sử dụng trong SELECT con
* Nhưng các thuộc tính được định nghĩa trong SELECT con không thể được sử dụng trong SELECT chính

Ví dụ: Tìm họ tên của các phi công có số ngày làm việc lớn hơn 20?

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **PHICONG** |  |  |  | **LAMVIEC** |  |  |  |  |
| **MPC** | **hoten** |  |  | **MPC** | **songay** |  |  |  |
| 1 | An |  |  | 1 | 10 |  |  |  |
| 2 | Sỹ |  |  | 2 | 21 |  |  |  |
| 3 | Long |  |  | 3 | 22 |  |  |  |
| 4 | Vinh |  |  |  |  |  |  |  |

**SELECT** hoten

**FROM** PHICONG

**WHERE** MPC **IN**

(**SELECT** MPC **FROM** LAMVIEC **WHERE** songay > 20) ; -- trả về nhiều giá trị

SQL mẫu [Bai giang\ketnoijoin.sql](Bai%20giang/ketnoijoin.sql)

Truy vấn con có điều kiện và toán tử:

* Nếu truy vấn con trả về một giá trị, các toán tử như >, >=, <, … có thể được sử dụng trong điều kiện với truy vấn con
* Nếu truy vấn con trả về một tập các giá trị, phải sử dụng các toán tử như ANY, ALL, EXISTS trong điều kiện với truy vấn con:
* ANY: Trả về true nếu một trong các giá trị của truy vấn con đúng
* ALL: Trả về true nếu tất cả các giá trị của truy vấn con đúng
* EXISTS: Trả về true nếu truy vấn con trả về ít nhất một dòng

Ví dụ: Tìm các loại máy bay của hãng Boeing mà có số chỗ lớn hơn ít nhất một loại nào đó của hãng Airbus

**SELECT** loai **FROM** LOAIMAYBAY

**WHERE** NSX = 'Boeing'

**AND** socho > **ANY**

**(SELECT** socho **FROM** LOAIMAYBAY **WHERE** NSX = 'Airbus');

SQL mẫu [Bai giang\ketnoijoin.sql](Bai%20giang/ketnoijoin.sql)

Ví dụ 4: Tìm các kiểu máy bay của hãng Airbus mà có số chỗ lớn hơn số chỗ của tất cả các kiểu của hãng Boeing

**SELECT** loai **FROM** LOAIMAYBAY

**WHERE** NSX = 'Airbus'

**AND** socho > ALL

(**SELECT** socho **FROM** LOAIMAYBAY **WHERE** NSX = 'Boeing');

SQL mẫu [Bai giang\ketnoijoin.sql](Bai%20giang/ketnoijoin.sql)

Ví dụ dụ 5: Tìm các máy bay thực hiện ít nhất một chuyến bay đến Paris?

**SELECT** \* **FROM** MAYBAY

**WHERE EXISTS**

(**SELECT** MMB **FROM** CHUYENBAY

**WHERE** MAYBAY.MMB = CHUYENBAY.MMB

**AND** noiden='Paris');

-- Cũng có thể sử dụng IN hoặc = ANY thay cho EXISTS

SO SÁNH JOIN VÀ CÂU TRUY VẤN CON

|  |  |
| --- | --- |
| JOIN | Truy vấn con |
| Kết quả có thể bao gồm các cột của cả 2 bảng | Không thể bao gồm các cột của câu truy vấn con |
| Sử dụng mối quan hệ giữa 2 bảng |  |
| Chạy nhanh hơn |  |
|  | Có thể chuyển 1 giá trị tính toán ra câu truy vấn bên ngoài |
|  | Dễ viết code và dễ hiểu |

#### **Toán tử IN**

Cú pháp:

**WHERE** <biểu thức> **[NOT] IN** (câu truy vấn con)

Ví dụ:

**SELECT** \* **FROM** NHAN\_VIEN

**WHERE** PHG **IN**

(**SELECT** MA\_PB **FROM** PHONG\_BAN **WHERE** TEN\_PB LIKE 'San xuat ?')

### **Hàm COUNT(), AVG(), SUM() - đếm, trung bình, tổng**

Hàm COUNT() trả về số bản ghi thỏa mãn điều kiện

**SELECT** **COUNT**(column\_name)

**FROM** table\_name

**WHERE** condition;

Ví dụ: Tìm xem có bao nhiêu phi công ở nước Pháp

**SELECT** **COUNT**(\*) **FROM** PHICONG **WHERE** nuoc = 'Phap';

Hàm AVG() trả về giá trị trung bình của cột dữ liệu kiểu số

**SELECT** **AVG**(column\_name)

**FROM** table\_name

**WHERE** condition;

Hàm SUM() trả về giá trị tổng của cột dữ liệu kiểu số

**SELECT** **SUM**(column\_name)

**FROM** table\_name

**WHERE** condition;

Ví dụ: Tính tổng khoảng cách đã bay, đường bay ngắn nhất, dài nhất và độ dài trung bình các chuyến bay của phi công mã số 20.

**SELECT** **SUM**(khoangcach) **AS** Tong, **MIN**(khoangcach) Nho\_nhat,

**MAX**(khoangcach) AS Lon\_nhat, **AVG**(khoangcach) AS Trung\_binh

**FROM** CHUYENBAY

**WHERE** MPC = 20;

*Ví dụ: … đi kèm khi mở file cơ ở dữ liệu …*

### **Hàm MIN(), MAX() - lớn nhất, nhỏ nhất**

Hàm MIN() trả về giá trị nhỏ nhất của cột dữ liệu lựa chọn

SELECT MIN(column\_name)

FROM table\_name

WHERE condition;

Hàm MAX() trả về giá trị lớn nhất của cột dữ liệu lựa chọn

SELECT MAX(column\_name)

FROM table\_name

WHERE condition;

Ví dụ: ***SELECT MAX(Price) AS LargestPrice FROM Products;***

### **UNION, INTERSECT, EXCEPT - hợp, giao, trừ trên tập hợp kết quả**

***[Phần dạy thêm]***

**ORDER BY** được sử dụng trong SELECT, kết quả trả về sẽ được sắp xếp theo cột chỉ định tăng dần (**ASC**) hoặc giảm dần (**DESC**). Nếu có nhiều thuộc tính sau ORDER BY, sắp xếp được thực hiện ưu tiên từ cột đầu tiên, sau đó mới đến các cột tiếp theo

<câu truy vấn 1>

**UNION [ALL] | INTERSECT [ALL] | EXCEPT [ALL]**

<câu truy vấn 2>

* Với UNION, là hợp, muốn cho phép trùng lặp thì dung UNION ALL

Ví dụ 1, hợp, hiển thị danh sách phi công làm việc cho cả hai công ty

**SELECT** MPC **FROM** LAMVIEC **WHERE** MCT = 1

**UNION**

**SELECT** MPC **FROM** LAMVIEC **WHERE** MCT = 2;

Ví dụ 2, giao, hiển thị danh sách phi công làm việc cho cả hai công ty

**SELECT** MPC **FROM** LAMVIEC **WHERE** MCT = 1

**INTERSECT**

**SELECT** MPC **FROM** LAMVIEC **WHERE** MCT = 2;

Ví dụ 3, trừ, hiển thị danh sách phi công làm việc cho công ty 2, nhưng không làm việc cho công ty 3

**SELECT** MPC **FROM** LAMVIEC **WHERE** MCT = 2

**EXCEPT**

**SELECT** MPC **FROM** LAMVIEC **WHERE** MCT = 3;

## **Slide 7 - DML – Data Manipulation Language Ngôn ngữ truy vấn dữ liệu**

**[Slide7] chưa làm**

Kết thúc bài học này bạn có khả năng

* Tạo 1 bảng mới từ 1 câu truy vấn
* Thêm 1 dòng dữ liệu vào bảng
* Thay đổi dữ liệu trong bảng
* Xoá các dòng
* Mở rộng hàng trong bảng

### **SELECT INTO - INSERT INTO - Select có lưu kết quả, chèn vào bảng mới**

***[Phần dạy thêm]***

SELECT INTO Cho phép lưu kết quả truy vấn vào một bảng mới hoặc một biến (1)

**SELECT** <tên cột> [, …] **INTO** <tên bảng mới>

**FROM** <tên bảng> [Đầy đủ các điều kiện như SELECT tiêu chuẩn];

Đặc biệt, SQL Server còn cho phép lưu kết quả vào biến tạm (2) SELECT <tên cột> [, ... ] INTO <#Tên biến tạm> FROM <tên bảng>;

**SELECT** \* **INTO** temp **FROM** CONGTY; -- trường hợp (1)

**SELECT** \* **INTO** #temp **FROM** CONGTY; -- trường hợp (2)

INSERT INTO cho phép thêm các dòng vào bảng từ kết quả truy vấn SELECT:

**INSERT** **INTO** <tên bảng> [ ( <tên cột> [, …]) ] <Lệnh SELECT>;

Ví dụ 11: Thêm dữ liệu phi công Pháp vào một bảng mới.

**CREATE** **TABLE** PHICONG\_PHAP(

MPC smallint PRIMARY KEY, hoten varchar(30), dchi varchar(30));

**INSERT** **INTO** PHICONG\_PHAP

**SELECT** MPC, hoten, dchi **FROM** PHICONG **WHERE** nuoc = 'Pháp';

## **Slide 8 - TÌM HIỂU CHỈ MỤC - SAO LƯU VÀ BACKUP CSDL**

**[Slide8] chưa làm**