

# Chương 4

## **Ngôn ngữ hỏi SQL**

Phạm Thị Ngọc Diễm  
ptndiem@ctu.edu.vn  
Bộ môn HTTT - ĐHCT

# Nội dung

- Giới thiệu ngôn ngữ hỏi SQL
- Các lệnh SQL căn bản
- Các lệnh SQL nâng cao

# Nội dung

- Giới thiệu ngôn ngữ hỏi SQL
- Các lệnh SQL căn bản
- Các lệnh SQL nâng cao


# Ngôn ngữ quan hệ

- Ngôn ngữ được xây dựng trên đại số quan hệ
- Ví dụ ngôn ngữ quan hệ
  - QBE (Query By Example, Zloof 1977)
  - QUEL (Query Language) d'INGRES (1975)
  - SQL (Structured Query Language)
    - Exemples: SQL IBM, SQL ORACLE,...
- Đơn giản, chỉ cần biết cấu trúc của các quan hệ để xây dựng các câu truy vấn.

# Lịch sử SQL

- Phiên bản gốc là Sequel do IBM phát triển trong những năm đầu 1970
- Sau được đổi tên thành SQL
- SQL là ngôn ngữ CSDL quan hệ chuẩn duy nhất
- Chuẩn ANSI và ISO cho SQL :
  - SQL-86, SQL-89
  - SQL-92, SQL:1999, SQL:2003, SQL:2008

# SQL

- Ngôn ngữ hoàn chỉnh định nghĩa trên đại số quan hệ
- Ngôn ngữ phi thủ tục được sử dụng để :
  - định nghĩa,
  - thao tác,
  - truy vấn và
  - kiểm soát việc truy cập

thông tin trong cơ sở dữ liệu
- Tài liệu chuẩn bao gồm hơn 600 trang

# Sử dụng SQL

- Sử dụng tương tác: trực tiếp từ bàn phím
- Sử dụng trong một ngôn ngữ lập trình (SQL nhúng):
  - COBOL,
  - PASCAL,
  - JAVA, ...
- SQL sử dụng các từ bảng, dòng và cột thay vì quan hệ bộ, thuộc tính.
- Các thành phần chính của SQL: DDL, DML và DCL

# Ngôn ngữ định nghĩa dữ liệu DDL

- Cho phép đặc tả các thông tin về các quan hệ, bao gồm:
  - Lược đồ quan hệ
  - Khóa,
  - Các ràng buộc toàn vẹn,
  - Các thuộc tính và miền giá trị của chúng, ...
- DDL cũng bao gồm các lệnh cho phép :
  - Định nghĩa và sửa đổi view



# Ngôn ngữ thao tác dữ liệu DML

- Gồm các lệnh cho phép:
  - Truy vấn thông tin từ CSDL
  - Cập nhật dữ liệu :
    - Thêm các bộ vào CSDL
    - Sửa thông tin các bộ có trong CSDL
    - Xoá bộ khỏi CSDL

# Ngôn ngữ điều khiển dữ liệu DCL

- Hỗ trợ DDL và DML
- Gồm các lệnh cho phép:
  - Định nghĩa người dùng và
  - Định nghĩa các quyền truy xuất của họ trên dữ liệu

# Tóm tắt các thành phần SQL

DDL	DML	DCL
CREATE DROP ALTER	SELECT INSERT DELETE UPDATE	GRANT REVOKE

# Qui ước câu lệnh SQL

- Có thể viết trên nhiều dòng
- Kết thúc bằng dấu chấm phẩy (;)
- Không phân biệt chữ hoa chữ thường
- **1 lệnh SQL** còn gọi là **1 câu truy vấn**
- Qui ước cú pháp lệnh:
  - **<...>** phần bắt buộc
  - **[ ... ]** phần tùy chọn
- **Lưu ý:** *Các câu lệnh trình bày trong phần tiếp theo có thể không thể thực thi trong 1 HQT CSDL cụ thể*

# CSDL minh họa

PHICONG(MPC, hoten, dchi,nuoc)

CONGTY (MCT, tencty, nuoc)

LOAIMAYBAY(loai, NSX, socho)

MAYBAY(MMB, *loai*, *MCT*)

CHUYENBAY(SOCB, ngaybay, *MPC*, *MMB*, noidi, noiden, khoangcach, giodi, gioden)

LAMVIEC(MPC, MCT, ngayBD, songay)

# Nội dung

- Giới thiệu ngôn ngữ hỏi SQL
- Các lệnh SQL căn bản
- Các lệnh SQL nâng cao

# SQL

## Data Definition Language



# Ngôn ngữ định nghĩa dữ liệu

- Tạo bảng & Thiết lập các ràng buộc
  - CREATE TABLE
- Thay đổi cấu trúc của bảng:
  - ALTER TABLE
- Xóa một bảng
  - DROP TABLE



# Các kiểu dữ liệu SQL

- char(n)
- varchar(n)
- int
- bit
- smallint
- numeric(p,d)
- real, double
- float(n)
- date
- time
- timestamp

# CREATE TABLE

- Lệnh tạo bảng đơn giản
  - CREATE TABLE <ten\_bang> (  
    <ten\_cot> <kieudulieu> [*rangbuoc\_cot* [...]]  
    / *rangbuoc\_bang*  
    [,...] )
  - *rangbuoc\_cot* có thể là
    - NOT NULL
    - UNIQUE
    - PRIMARY KEY
    - CHECK (<dieukien>)
    - DEFAULT <giatri>
  - *rangbuoc\_bang* là ràng buộc khoá ngoại, có 2 cách :
    - + FOREIGN KEY(<cot\_thamchieu>) REFERENCES <bang\_thamchieu>(<cot\_thamchieu>)
    - + REFERENCES <bang\_thamchieu>(<cot\_thamchieu>)

# CREATE TABLE - Ví dụ

- Ví dụ đơn giản

```
CREATE TABLE PHICONG(  
    MPC smallint ,  
    hoten varchar(30),  
    dchi varchar(30));
```

```
CREATE TABLE CONGTY(  
    MCT smallint ,  
    tencty varchar(30),  
    nuoc varchar(20));
```

=> Bảng **chưa** định nghĩa khóa chính và các ràng buộc khác

# CREATE TABLE - Ví dụ

- Thêm các ràng buộc **PRIMARY KEY** , **NOT NULL**, **UNIQUE**

```
CREATE TABLE PHICONG(  
    MPC smallint PRIMARY KEY, -- Khoá chính  
    hoten varchar(30) NOT NULL,  
    dchi varchar(30) );
```

```
CREATE TABLE CONGTY(  
    MCT smallint NOT NULL,  
    tencty varchar(30) UNIQUE, -- Khoá ứng viên, duy nhất  
    nuoc varchar(20),  
    PRIMARY KEY (MCT) );
```

Chú ý: Thuộc tính được khai báo khoá chính mặc định là NOT NULL

# CREATE TABLE - Ví dụ

- Thêm khoá ngoại (**FOREIGN KEY**) và các ràng buộc khác, đặt tên cho một ràng buộc (**CONSTRAINT**)

```
CREATE TABLE CHUYENBAY(  
    SOCB varchar(10) NOT NULL,  
    MPC smallint NOT NULL,  
    MMB smallint NOT NULL,  
    noidi varchar(20) DEFAULT 'Paris', -- Giá trị mặc định  
    noiden varchar(20),  
    khoangcach int CHECK(khoangcach>0), -- Ràng buộc miền giá trị  
    giodi time,  
    gioden time ,  
    ngaybay date NOT NULL,  
    CONSTRAINT fk_MPC FOREIGN KEY(MPC) REFERENCES PHICONG(MPC),  
    FOREIGN KEY(MMB) REFERENCES MAYBAY(MMB) ); -- Khoá ngoại
```

=> *Khóa chính nhiều cột ? → PRIMARY KEY (SOCB, ngaybay)*

# CREATE TABLE - RB tham chiếu CASCADE

- Mệnh đề **REFERENCES** của lệnh **CREATE TABLE** và **ALTER TABLE** hỗ trợ mệnh đề ON DELETE và ON UPDATE
- CASCADE có thể định nghĩa cho cập nhật và xóa dữ liệu.
- 4 tùy chọn sau có thể dùng:

- SQL Server:

[ **ON DELETE | UPDATE** { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]

- Oracle :

[ **ON DELETE | UPDATE** { RESTRICT | CASCADE | SET NULL | NO ACTION } ]

# CREATE TABLE - RB tham chiếu CASCADE

1. **SET NULL**: cột sẽ nhận giá trị NULL nếu cột tham chiếu bị xoá hoặc cập nhật
2. **CASCADE**: cột sẽ được cập nhật khi cột tham chiếu được cập nhật và dòng sẽ bị xoá khi dòng tham chiếu bị xoá.
3. **SET DEFAULT**: cột sẽ nhận giá trị mặc định khi thao tác cập nhật/xoá được thực hiện trên dòng tham chiếu.
4. **NO ACTION/RESTRICT**: *tuỳ chọn mặc định*. Nếu thao tác cập nhật hay xoá được thực hiện trên dòng tham chiếu, thao tác này sẽ bị cấm.

# CREATE TABLE - RB tham chiếu CASCADE

```
CREATE TABLE Albums
```

```
(  
    AlbumID    INT    PRIMARY KEY,  
    Name       VARCHAR(50)  
);
```

```
CREATE TABLE Tracks
```

```
(  
    TrackID     INT    PRIMARY KEY,  
    Title       VARCHAR(50),  
    AlbumID     INT    REFERENCES Albums(AlbumID) -- khoá ngoại  
                ON DELETE SET NULL  
                ON UPDATE CASCADE  
);
```



# CREATE TABLE - RB tham chiếu CASCADE

**CREATE TABLE** Albums

```
(  
    AlbumID    INT    PRIMARY KEY,  
    Name       VARCHAR(50)  
);
```

**CREATE TABLE** Tracks

```
(  
    TrackID    INT    PRIMARY KEY,  
    Title      VARCHAR(50),  
    AlbumID    INT    DEFAULT 1 REFERENCES Albums(AlbumID)  
                ON DELETE SET DEFAULT  
                ON UPDATE CASCADE  
);
```

# ALTER TABLE

- Cho phép thay đổi cấu trúc bảng, thêm hoặc xóa các RBTV
- Thay đổi cấu trúc một bảng, gồm :

- Cú pháp thêm cột

```
ALTER TABLE <ten_bang>
```

```
ADD <ten_cot> <kieudulieu>
```

- Cú pháp xóa cột

```
ALTER TABLE <ten_bang>
```

```
DROP COLUMN <ten_cot>
```

- Cú pháp đổi kiểu dữ liệu một cột trong **SQL Server/Oracle**

```
ALTER TABLE <ten_bang>
```

```
ALTER COLUMN/MODIFY <ten_cot> <kieudulieu>
```

# ALTER TABLE - Ví dụ

- Thêm cột

- `ALTER TABLE LAMVIEC ADD songay int;`
- `ALTER TABLE LAMVIEC ADD nuoc varchar(20) UNIQUE;`

- Thêm khoá chính

- `ALTER TABLE PHICONG ADD PRIMARY KEY (MPC);`
- Hoặc:

`ALTER TABLE PHICONG ADD CONSTRAINT pk_MPC PRIMARY KEY (MPC);`

- **Chú ý:** cột MPC phải là **NOT NULL**

# ALTER TABLE - Ví dụ

- Xoá cột
  - `ALTER TABLE LAMVIEC DROP COLUMN nuoc ;`
- Xoá ràng buộc
  - SQL Server / Oracle
    - `ALTER TABLE PHICONG DROP CONSTRAINT pk_MPC ;`
  - My SQL
    - `ALTER TABLE PHICONG DROP PRIMARY KEY`
- Thay đổi kiểu dữ liệu một cột
  - **SQL Server :**
    - `ALTER TABLE LAMVIEC ALTER COLUMN songay smallint ;`
  - **Oracle :** `ALTER TABLE LAMVIEC MODIFY songay smallint ;`

# DROP TABLE

- Xóa cấu trúc một bảng
- Cú pháp

DROP TABLE <*ten\_bang*>

- Ví dụ

DROP TABLE LAMVIEC;

# SQL

## Data Manipulation Language



# DML - Ngôn ngữ thao tác dữ liệu

Ngôn ngữ thao tác dữ liệu (DML) cho phép:

- Hiển thị: **SELECT**
- Xóa : **DELETE**
- Thêm : **INSERT**
- Cập nhật : **UPDATE**

dữ liệu trong các bảng

# Lệnh INSERT

- Thêm một dòng dữ liệu vào bảng
- Cú pháp:
  - Không chỉ ra tên cột

**INSERT INTO** <ten\_bang>  
**VALUES** (giatri1, giatri2, giatri3,...);

- Chỉ ra tên cột

**INSERT INTO** <ten\_bang> (cot1, cot2, cot3,...)  
**VALUES** (giatri1, giatri2, giatri3,...);



# INSERT - Ví dụ

```
INSERT INTO CONGTY(MCT, tencty, nuoc)
```

```
VALUES (1, 'Air France', 'Phap');
```

```
INSERT INTO CONGTY VALUES (3, 'Qantas', 'Uc');
```

```
INSERT INTO CONGTY VALUES (2, 'British Airways', 'Anh');
```

```
INSERT INTO CONGTY VALUES (4, 'Easy Jet', 'EU');
```

MCT	tencty	nuoc
1	Air France	Phap
2	British Airways	Anh
3	Qantas	Uc
4	Easy Jet	EU

# Lệnh UPDATE

- Sửa đổi nội dung của một hoặc nhiều dòng dữ liệu
- Cú pháp:

**UPDATE** <ten\_bang> **SET** cot1=giatri1, cot2=giatri2, ...  
[**WHERE** <dieu\_kien>];

- Ví dụ:

UPDATE CONGTY SET tencty= 'RYANAIR'  
WHERE tencty='Easy Jet';

- Nếu không có **WHERE**, sẽ cập nhật tất cả các dòng

# Lệnh DELETE

- Xóa hoặc nhiều dòng dữ liệu của một bảng
- Cú pháp:

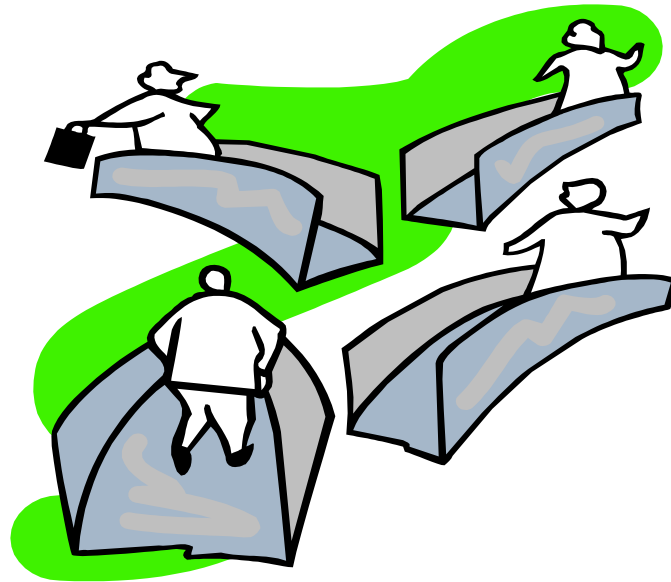
**DELETE FROM** <ten\_bang>

[**WHERE** <dieu\_kien>];

- Ví dụ:
  - DELETE FROM CONGTY ;
  - DELETE FROM CONGTY WHERE tencty='Easy Jet';
- *Nếu không có **WHERE**, xóa tất cả các dòng*

# Data Manipulation Language

## SELECT



# Lệnh SELECT

- Cho phép truy vấn và hiển thị dữ liệu từ các bảng
- Cú pháp đơn giản:

**SELECT** <ten\_cot|bieu\_thuc> [, ...]

[**FROM** <ten\_bang> [, ...] ]

[**WHERE** <điều kiện logic trên dòng/bộ>]

[**GROUP BY** <các thuộc tính gom nhóm>

[**HAVING** <điều kiện logic gom nhóm> ] ]

[**ORDER BY** <các thuộc tính sắp xếp>]

# Lệnh SELECT

- Ví dụ đơn giản: Liệt kê họ tên và địa chỉ của tất cả các phi công

```
SELECT hoten, dchi  
FROM PHICONG ;
```

	hoten	dchi
1	Dupond	Nice
2	Smith	London
3	Garratt	Perth
4	Durand	Paris
5	MacMachin	Glasgow

=> *Thứ tự các thuộc tính theo sau SELECT xác định thứ tự hiển thị của các cột trong bảng kết quả.*

=> *Có thể đặt lại tên thuộc tính*

# Lệnh SELECT - Từ khóa AS

- Sử dụng từ khoá **AS** để *đặt lại tên cột* hoặc *đặt bí danh cho bảng*.
- Ví dụ:** đặt lại tên cho cột hoten và dchi

```
SELECT hoten AS hoten_phicong, dchi AS diachi  
FROM PHICONG ;
```

	hoten	dchi
1	Dupond	Nice
2	Smith	London
3	Garratt	Perth
4	Durand	Paris
5	MacMachin	Glasgow

Đặt lại tên cột



	hoten_phicong	diachi
1	Dupond	Nice
2	Smith	London
3	Garratt	Perth
4	Durand	Paris
5	MacMachin	Glasgow

# Lệnh SELECT - Từ khóa AS

- **Bí danh (alias)** cho phép tạm thời đặt lại tên bảng

- Ví dụ:

```
SELECT p.hoten, p.dchi AS diachi  
FROM PHICONG AS p;
```

- AS là tùy chọn và có thể bỏ qua, nghĩa là:

- PHICONG AS p = PHICONG p
- p.dchi AS diachi = p.dchi diachi

- Lưu ý:

- Oracle bỏ qua AS
- SQL Server: sử dụng cả hai



# Lệnh SELECT - Chọn tất cả các cột

- Hiển thị tất cả các cột của một bảng, dùng \*
- Ví dụ: Tìm tất cả thông tin của các phi công

SELECT \*

FROM PHICONG ;

	MPC	hoten	dchi	nuoc
1	10	Dupond	Nice	Phap
2	20	Smith	London	Anh
3	30	Garratt	Perth	Uc
4	40	Durand	Paris	Phap
5	50	MacMachin	Glasgow	Scotland

# Phép chiếu

- Sử dụng SELECT và các thuộc tính cần chiếu. Các thuộc tính này cách nhau bởi dấu phẩy.
- Ví dụ:  $\pi_{NSX}(LOAIMAYBAY)$

SELECT NSX FROM LOAIMAYBAY;

- Kết quả:

	NSX
1	AirBus
2	AirBus
3	Boeing
4	Boeing

*=> Không loại bỏ những dòng trùng nhau*

# Phép chiếu - Loại các dòng trùng nhau

- Sử dụng **DISTINCT** để loại bỏ các **dòng** trùng nhau
- Ví dụ:  $\pi_{NSX}(LOAIMAYBAY)$

**SELECT DISTINCT NSX FROM LOAIMAYBAY;**

- Kết quả:

	NSX
1	AirBus
2	Boeing

- Ngược lại với **DISTINCT** là **ALL**. Giá trị mặc định là **ALL**.

# Phép chọn - Mệnh đề WHERE

- Mệnh đề WHERE là tùy chọn
- Sử dụng mệnh đề WHERE để chỉ ra một tập các điều kiện chọn.
- Nếu một bộ thỏa mãn các điều kiện, nó sẽ là một bộ trong kết quả

# Điều kiện chọn

- Các điều kiện chọn **chỉ có thể đặt sau WHERE hoặc HAVING** (xem phần GROUP BY)
- Các điều kiện chọn **trên dòng** là một biểu thức logic gồm
  - Các thuộc tính
  - Các phép toán số học
  - Các phép toán so sánh với các toán tử : <, <=, >, >=, != (<>)
  - Các toán tử AND, OR và NOT với thứ tự ưu tiên NOT, AND, OR
  - Các toán tử IN, LIKE, BETWEEN, IS NULL,...

# Ví dụ

- Ví dụ 1: Tìm họ tên các phi công sống ở Paris

```
SELECT hoten  
FROM PHICONG  
WHERE dchi='Paris';
```

- Ví dụ 2: Tìm thông tin về các chuyến bay từ Hanoi đến Paris ngày 1/1/2014 ???

```
SELECT *  
FROM CHUYENBAY  
WHERE noidi='Ha noi' and noiden='Paris' and ngaybay='2014-01-01';
```

- Oracle : Hàm **TO\_DATE()** trả về giá trị ngày, **SYSDATE** : ngày mặc định

- Ví dụ: **TO\_DATE('01/01/2004', 'mm/dd/yyyy')**

# Toán tử IN

- Cho phép chỉ ra nhiều giá trị trong mệnh đề **WHERE**

```
SELECT <ten_cot> [, ...]  
FROM <ten_bang>  
WHERE <ten_cot> IN (value1,value2,...) ;
```

- Ví dụ: Tìm họ tên các phi công ở các nước Pháp, Anh hoặc Úc

```
SELECT hoten  
FROM PHICONG  
WHERE nuoc IN ( 'Phap' , 'Anh' , 'Uc' ) ;  
-- nuoc='Phap' or nuoc='Anh' or nuoc='Úc'
```

# Toán tử BETWEEN

- Cho phép chọn một giá trị trong một giới hạn. Giá trị có thể là số, chuỗi hoặc ngày

```
SELECT <ten_cot> [, ...]  
FROM <ten_bang>  
WHERE <ten_cot> BETWEEN value1 AND value2 ;
```

- Ví dụ: Tìm thông tin tất cả các chuyến bay từ 10000 km đến 15000 km

```
SELECT *  
FROM CHUYENBAY  
WHERE khoangcach BETWEEN 10000 AND 15000;  
-- khoangcach >= 10000 AND khoangcach <= 15000;
```



# Toán tử LIKE

- Được sử dụng trong mệnh đề WHERE để tìm một *mẫu* cụ thể trong một cột

```
SELECT <ten_cot> [, ...]  
FROM <ten_bang>  
WHERE <ten_cot> LIKE mẫu;
```

- Ví dụ: Tìm họ tên các phi công bắt đầu bằng chữ D

```
SELECT hoten  
FROM PHICONG  
WHERE hoten LIKE 'D%';
```

# Toán tử LIKE

- Các dạng của *mẫu*

Mẫu	Giải thích
%	Không hoặc nhiều ký tự
_	Một ký tự duy nhất
[danh sách các ký tự]	Tập các ký tự dùng so khớp
[^danh sách các ký tự]	Chỉ khớp với một ký tự không nằm trong [ ]

Ví dụ:

- Tìm họ tên các phi công bắt đầu bằng d hoặc m

```
SELECT hoten FROM PHICONG
```

```
WHERE hoten LIKE '[dm]%' -- hoten LIKE '___'
```

- Tìm họ tên các phi công không bắt đầu bằng d hoặc m

```
SELECT hoten FROM PHICONG
```

```
WHERE hoten LIKE '[^dm]%' -- hoten LIKE '%Lê%'
```

# Giá trị NULL

- Trong một dòng dữ liệu, một vài thuộc tính có thể nhận giá trị NULL. Giá trị NULL có thể là:
  - Giá trị không rõ hoặc
  - Giá trị không tồn tại
- Các kết quả của bất kỳ biểu thức số học với giá trị NULL sẽ là NULL
  - Ví dụ:  $5 + \text{null} \rightarrow \text{null}$
- Để kiểm tra giá trị NULL dùng toán tử **IS NULL** hoặc **IS NOT NULL**

# Toán tử **IS NULL** và **IS NOT NULL**

- Không thể kiểm tra giá trị null với các toán tử như =, >=, !=

=> Sử dụng **IS NULL** và **IS NOT NULL**

- Ví dụ: tìm họ tên các phi công có địa chỉ null

```
SELECT hoten  
FROM PHICONG  
WHERE dchi IS NULL;
```

# Các phép toán số học và hàm

- Sử dụng trong mệnh đề **SELECT, WHERE và HAVING**
- Các toán tử số học : **+, -, \*, /**
  - Độ ưu tiên \* hoặc /, + hoặc -
  - Sử dụng dấu ngoặc đơn () để thay đổi độ ưu tiên này
  - Nếu một toán hạng là **NULL**, kết quả sẽ là **NULL**
- Các hàm **SQL Server**
  - Các hàm chuỗi
    - **LEN, UPPER, LOWER, LEFT, RIGHT, LTRIM, RTRIM, ...**
  - Các hàm ngày
    - **GETDATE, DAY, MONTH, YEAR, DATEDIFF, ISDATE, ...**

# Các hàm kết tập

- Sử dụng trong mệnh đề **SELECT** hoặc **HAVING**
- Đầu vào: một tập các giá trị của một cột hoặc một *nhóm*
- Đầu ra: một giá trị duy nhất
- Các hàm trên các thuộc tính số:
  - SUM : tính tổng các giá trị của một thuộc tính của các bộ được chọn
  - AVG : tính giá trị trung bình
- Các hàm trên các thuộc tính số và kiểu dữ liệu khác:
  - MIN : giá trị nhỏ nhất
  - MAX : giá trị lớn nhất
  - COUNT : đếm số giá trị

# Các hàm kết tập

- Ví dụ:

**1.** Có tất cả bao nhiêu phi công ở nước Pháp

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

**2.** 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 COUNT(*) socb, SUM(khoangcach) Tong,  
       MIN(khoangcach) Nho_nhat,  
       MAX(khoangcach) Lon_nhat,  
       AVG(khoangcach) Trung_binh  
FROM CHUYENBAY  
WHERE MPC=20;
```

# Các hàm kết tập

- Hàm AVG = tổng các giá trị không NULL/ số các giá trị không NULL
- Các hàm MIN, MAX, COUNT có thể sử dụng cho CHAR, VARCHAR, DATE và các kiểu dữ liệu số
- Hàm SUM, AVG chỉ sử dụng duy nhất cho kiểu số
- **Chú ý:**

```
SELECT hoten, COUNT(*) FROM PHICONG  
WHERE nuoc='Phap' ;
```

⇒ Truy vấn không hợp lệ



# Sắp xếp kết quả - ORDER BY

- Từ khóa **ORDER BY** được dùng để sắp xếp kết quả trên một hoặc nhiều cột
- **Mặc định** ORDER BY sắp xếp kết quả theo thứ tự tăng dần (ASC)
- Từ khóa DESC cho phép sắp xếp kết quả theo thứ tự giảm dần
- Nếu có nhiều thuộc tính sau ORDER BY, sắp xếp được thực hiện ưu tiên trên cột đầu tiên, kể đến cột 2, ...
- Ví dụ

```
SELECT hoten, nuoc  
FROM PHICONG  
ORDER BY hoten, nuoc DESC;
```

# Các phép toán trên tập hợp

- Các phép toán trên tập hợp gồm:

- Phép hợp  $\cup \rightarrow$  UNION
- Phép trừ  $\setminus \rightarrow$  EXCEPT (\*)
- Phép giao  $\cap \rightarrow$  INTERSECT

- Cú pháp

*<Câu truy vấn 1>*

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

*<Câu truy vấn 2>*

(\*) Oracle dùng từ khoá **MINUS** cho phép trừ

# Các phép toán trên tập hợp

- Tìm các mã phi công làm cho công ty 1 và công ty 2

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

```
INTERSECT
```

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

- Tìm các mã phi công làm cho công ty 2 và không làm cho công ty 3

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

```
EXCEPT ALL
```

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

# Truy vấn đơn giản trên nhiều bảng

- Cho phép tìm kiếm dữ liệu từ nhiều bảng khác nhau trong CSDL.
- Liệt kê các bảng cần dùng trong mệnh đề FROM
- Ví dụ: Tìm mã và họ tên phi công có số ngày làm việc cho một công ty bất kỳ lớn hơn 20 ngày

```
SELECT DISTINCT p.MPC, hoten
```

```
FROM LAMVIEC l, PHICONG p
```

```
WHERE l.MPC = p.MPC
```

```
AND songay > 20
```

← Điều kiện nối kết

← Điều kiện chọn

# Truy vấn đơn giản trên nhiều bảng

- Điều kiện nối kết:
  - Nếu có **n bảng** thì có ít nhất **n-1 điều kiện nối kết**
  - Nối kết giữa **thuộc tính khóa chính** của bảng cha và **thuộc tính khóa ngoài** của bảng con
  - Liên kết các điều kiện nối kết bởi toán tử **AND**
- Ví dụ 1: Tìm khoảng cách bay mà phi công Nguyễn Tuấn đã thực hiện ngày 1/1/2014

```
SELECT khoangcach  
FROM PHICONG p, CHUYENBAY c  
WHERE c.MPC = p.MPC  
AND hoten = 'Nguyễn Tuấn'  
AND ngaybay = TO_DATE('2014/01/01', 'yyyy/mm/dd');
```

# Truy vấn đơn giản trên nhiều bảng

- Điều kiện nối kết:
  - Nếu có **n bảng** thì có ít nhất **n-1 điều kiện nối kết**
  - Nối kết giữa **thuộc tính khóa chính** của bảng cha và **thuộc tính khóa ngoài** của bảng con
  - Liên kết các điều kiện nối kết bởi toán tử **AND**
- Ví dụ 2: Tìm tên công ty mà phi công Patrick Cortier đã làm việc

```
SELECT DISTINCT tencty
FROM LAMVIEC l, PHICONG p, CONGTY c
WHERE l.MPC = p.MPC AND l.MCT = c.MCT
      AND hoten = 'Patrick Cortier'
```

# Truy vấn đơn giản trên nhiều bảng

- Nếu không có đk nối kết = Phép tích Descartes
- Tích Descartes trong SQL Server: **CROSS JOIN**
  - **CROSS JOIN** cho phép thực hiện tích Descartes giữa hai bảng

- Ví dụ:

```
SELECT *  
FROM PHICONG p CROSS JOIN LAMVIEC l
```

- Hoặc không dùng điều kiện nối kết :

```
SELECT *  
FROM PHICONG p, LAMVIEC l
```

# Nội dung

- Giới thiệu ngôn ngữ hỏi SQL
- Các lệnh SQL căn bản
- Các lệnh SQL nâng cao



# Các lệnh SQL nâng cao

- Phép kết nối
- Truy vấn lồng nhau
- Gom nhóm
- Các lệnh khác

# Truy vấn trên nhiều bảng

- Phép nối kết:
  - Kết nối tự nhiên
  - INNER JOIN
  - OUTER JOIN
- Truy vấn lồng nhau
  - Trả về một giá trị
  - Trả về nhiều giá trị

# Phép nối kết (join)

- Cho phép kết hợp các dòng giữa hai hay nhiều bảng dựa trên **các cột giống nhau** giữa các bảng này
- Ba kiểu nối kết :
  - Phép kết nối tự nhiên NATURAL JOIN
  - Phép nối kết đơn giản INNER JOIN
  - Các phép nối kết OUTER JOIN
    - LEFT JOIN (mở rộng trái)
    - RIGHT JOIN (mở rộng phải)
    - FULL JOIN (mở rộng hai bên)
- Khái niệm JOIN này có thể dễ hiểu hơn là việc gộp tất cả các đk chọn và đk nối kết trong mệnh đề WHERE.

# NATURAL JOIN

- Tương tự ĐSQH, nối kết tự nhiên cho phép nối kết giữa hai bảng
  - Không yêu cầu chỉ ra đk nối kết
  - Nối kết ngầm định giữa hai thuộc tính cùng tên giữa 2 bảng
  - Hai thuộc tính cùng tên này chỉ xuất hiện 1 lần trong quan hệ kết quả
- Ví dụ: Tìm mã và họ tên phi công có số ngày làm việc cho một công ty bất kỳ lớn hơn 20 ngày

```
SELECT DISTINCT p.MPC, hoten  
FROM LAMVIEC l NATURAL JOIN PHICONG p  
WHERE songay > 20
```

# NATURAL JOIN

- Nếu 2 thuộc tính không cùng tên → có thể đổi tên trước khi nối kết
- Có thể xảy ra lỗi khi có 2 thuộc tính cùng tên ở 2 bảng nhưng không phải là hai thuộc tính để nối kết (thuộc tính khoá chính và khoá ngoài)
- **Ví dụ:** hai bảng PHICONG và CONGTU cùng có thuộc tính 'nuoc'
- **SQL Server không hỗ trợ NATURAL JOIN,**
- **Oracle hỗ trợ NATURAL JOIN** (trên 1 hoặc nhiều cột cùng tên)

# INNER JOIN

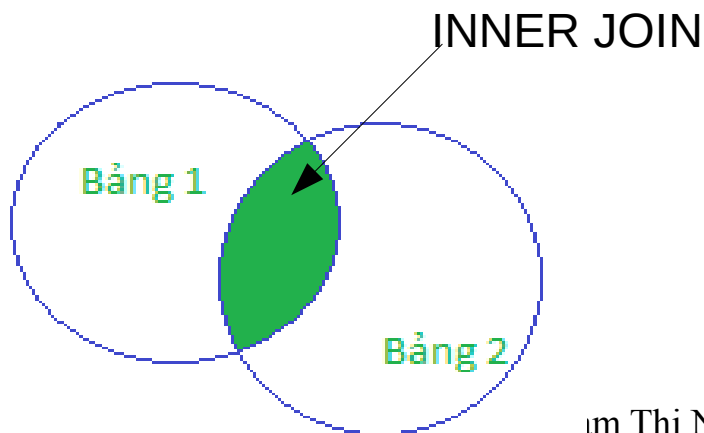
- Là loại nối kết đơn giản được dùng nhiều nhất
- Cú pháp

```
SELECT <ten_cot> [,...]
```

```
FROM <table1>
```

```
INNER JOIN | JOIN <table2>
```

```
ON <table1>.<ten_cot>= <table2>.<ten_cot>;
```



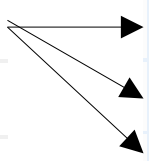
# INNER JOIN

- Cách nối kết:
  - Đối với mỗi dòng của **table1**, tìm các dòng tương ứng trong **table2**,  
→ sự tương ứng này được thực hiện dựa trên các cột chung của 2 bảng, đó là các cột dùng để nối kết
    - Nếu **không tìm được**, dòng này (table1) không được thêm vào kết quả
    - Nếu **tìm được**, một dòng sẽ được thêm vào kết quả (*dòng này bao gồm sự kết hợp các cột ở cả hai bảng*)
    - Nếu **tìm được nhiều dòng** tương ứng ở **table2**, nhiều dòng sẽ được thêm vào kết quả (giá trị các cột của **table1** lặp lại nhiều lần).

# INNER JOIN

- Ví Dụ: Tìm họ tên các phi công có số ngày làm việc cho một công ty nào đó là 20 ngày

MPC	hoten	dchi	nuoc
10	Dupond	Nice	Phap
20	Smith	London	Anh
30	Garratt	Perth	Uc
40	Durand	Paris	Phap
50	MacMachin	Glasgow	Scotland
60	Patrick	NULL	Phap
70	Patrick	vietnam	anh



MPC	MCT	songay
10	1	20
20	1	30
30	1	40
30	2	200
30	3	300
40	2	40
50	3	20



# INNER JOIN

- Ví Dụ: Tìm họ tên các phi công có số ngày làm việc cho một công ty nào đó là 20 ngày

	MPC	hoten	dchi	nuoc	MPC	MCT	songay
1	10	Dupond	Nice	Phap	10	1	20
2	20	Smith	London	Anh	20	1	30
3	30	Garratt	Perth	Uc	30	1	40
4	30	Garratt	Perth	Uc	30	2	200
5	30	Garratt	Perth	Uc	30	3	300
6	40	Durand	Paris	Phap	40	2	40
7	50	MacMachin	Glasgow	Scotland	50	3	20

# INNER JOIN

- Ví Dụ: Tìm họ tên các phi công có số ngày làm việc cho một công ty nào đó là 20 ngày

1. SELECT hoten

FROM PHICONG p JOIN LAMVIEC l

ON p.MPC=l.MPC

WHERE songay=20;

2. SELECT tencty

FROM LAMVIEC l JOIN PHICONG p ON l.MPC = p.MPC

JOIN CONGTY c ON l.MCT = c.MCT

WHERE hoten = 'Patrick Cortier'

# LEFT JOIN

- Cú pháp

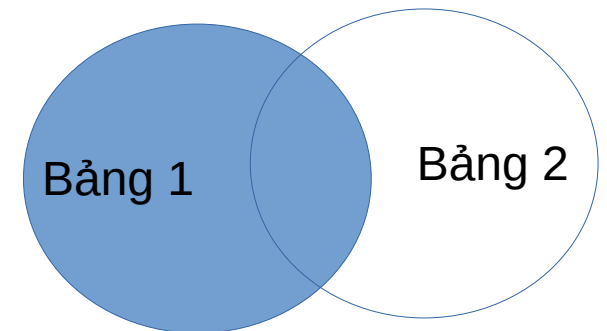
```
SELECT <ten_cot> [,...]  
FROM <table1>
```

```
LEFT JOIN | LEFT OUTER JOIN <table2>
```

```
ON <table1>.<ten_cot>= <table2>.<ten_cot>;
```

- Kết quả bao gồm :

- Các dòng tương ứng ở cả 2 bảng (kết quả INNER JOIN)
- Và các dòng thuộc bảng **table1** không tương ứng với **table2**. Các cột tương ứng của **table2** sẽ mang giá trị NULL



# LEFT JOIN

- Ví Dụ: Tìm họ tên các phi công, mã công ty và số ngày làm việc cho công ty nào đó kể cả các phi công chưa làm việc cho công ty nào

MPC	hoten	dchi	nuoc
10	Dupond	Nice	Phap
20	Smith	London	Anh
30	Garratt	Perth	Uc
40	Durand	Paris	Phap
50	MacMachin	Glasgow	Scotland
60	Patrick	NULL	Phap
70	Patrick	vietnam	anh

MPC	MCT	songay
10	1	20
20	1	30
30	1	40
30	2	200
30	3	300
40	2	40
50	3	20

# LEFT JOIN

- Ví Dụ: Tìm họ tên các phi công, mã công ty và số ngày làm việc cho công ty nào đó kể cả các phi công chưa làm việc cho công ty nào

```
SELECT  hoten, MCT, songay
FROM PHICONG p
LEFT JOIN LAMVIEC l
ON p.MPC=l.MPC
```

	hoten	MCT	songay
1	Dupond	1	20
2	Smith	1	30
3	Garratt	1	40
4	Garratt	2	200
5	Garratt	3	300
6	Durand	2	40
7	MacMachin	3	20
8	Patrick	NULL	NULL
9	Patrick	NULL	NULL

# RIGHT JOIN

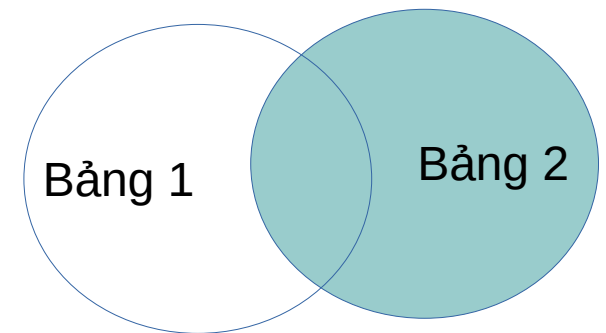
- Cú pháp

```
SELECT <ten_cot> [,...]  
FROM <table1>
```

```
    RIGHT JOIN | RIGHT OUTER JOIN <table2>  
    ON <table1>.<ten_cot>= <table2>.<ten_cot>;
```

- Kết quả bao gồm :

- Các dòng tương ứng ở cả 2 bảng (kết quả INNER JOIN)
- Và các dòng thuộc bảng **table2** không tương ứng với **table1**. Các cột tương ứng của **table1** sẽ mang giá trị NULL



# RIGHT JOIN

- 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 công ty này không có phi công ở Anh làm việc

	MPC	hoten	dchi	nuoc
1	20	Smith	London	Anh
2	70	Patrick	vietnam	anh

Phi công sống ở Anh

	MPC	MCT	songay
1	10	1	20
2	20	1	30
3	30	1	40

Phi công làm việc cho công ty số 1

# RIGHT JOIN

- 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 công ty này không có phi công ở Anh làm việc

```
SELECT l.MPC, hoten, dchi, songay
FROM (SELECT * FROM PHICONG WHERE nuoc LIKE 'anh') p
RIGHT JOIN (SELECT * FROM LAMVIEC where MCT=1) l
ON p.MPC=l.MPC
```

	MPC	hoten	dchi	songay
1	10	NULL	NULL	20
2	20	Smith	London	30
3	30	NULL	NULL	40



# FULL JOIN

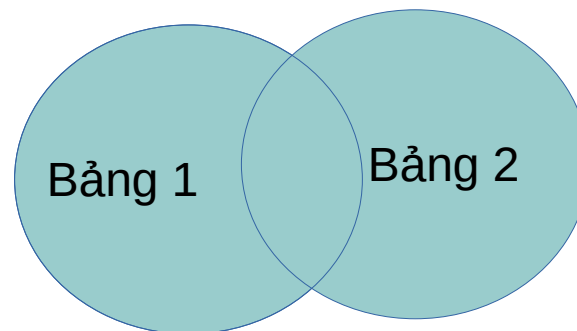
- Cú pháp

```
SELECT <ten_cot> [,...]
```

```
FROM <table1>
```

```
FULL JOIN | FULL OUTER JOIN <table2>
```

```
ON <table1>.<ten_cot>= <table2>.<ten_cot>;
```



# FULL JOIN

- Kết quả bao gồm :
  - Các dòng tương ứng ở cả 2 bảng (kết quả INNER JOIN)
  - Và các dòng thuộc bảng **table1** không tương ứng với **table2**. Các cột tương ứng của **table2** sẽ mang giá trị NULL
  - Và các dòng thuộc bảng **table2** không tương ứng với **table1**. Các cột tương ứng của **table1** sẽ mang giá trị NULL

# FULL JOIN

- 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

	MPC	hoten	dchi	nuoc
1	20	Smith	London	Anh
2	70	Patrick	vietnam	anh

Phi công sống ở Anh

	MPC	MCT	songay
1	10	1	20
2	20	1	30
3	30	1	40

Phi công làm việc cho công ty số 1

# FULL JOIN

- 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 l.MPC, hoten, dchi, songay
FROM (SELECT * FROM PHICONG WHERE nuoc LIKE 'anh') p
FULL JOIN (SELECT * FROM LAMVIEC where MCT=1) l
ON p.MPC=l.MPC
```

	MPC	hoten	dchi	songay
1	10	NULL	NULL	20
2	20	Smith	London	30
3	30	NULL	NULL	40
4	NULL	Patrick	vietnam	NULL

# Truy vấn con (subquery/ nested query)

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

***SELECT ... FROM ...***

***WHERE [... **AND**] <dk với truy vấn con>***

***(SELECT con)***

- Một truy vấn con có thể chứa một truy vấn con khác
- Kết quả của truy vấn con có thể là **một giá trị** hoặc **nhiều giá trị**
- Truy vấn con sinh ra **một điều kiện trong mệnh đề WHERE** của truy vấn chính

# Truy vấn con – Ví dụ

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

```
SELECT hoten  
FROM PHICONG  
WHERE MPC IN  
    (SELECT MPC FROM LAMVIEC  
     WHERE songay>20)
```

*=> trả về nhiều giá trị*

- Tìm họ tên của các phi công có số ngày làm việc nhiều nhất ?

```
SELECT hoten  
FROM PHICONG JOIN LAMVIEC ON p.MPC=l.MPC  
WHERE songay =  
    (SELECT MAX (songay) FROM LAMVIEC);
```

*=> trả về một giá trị*

# Truy vấn con – 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

# Truy vấn con – Điều kiện & 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 IN, ANY/SOME, ALL, EXISTS trong **điều kiện với truy vấn con**
  - **IN** : **where** <tên\_cột> IN/NOT IN (select <tên\_cột>...)
  - **ANY** : trả về true nếu một trong các giá trị của truy vấn con đúng:
    - **where** <tên\_cột> <toán tử so sánh> ANY (select <tên\_cột>...)
  - **ALL**: trả về true nếu tất cả các giá trị của truy vấn con đúng
    - **where** <tên\_cột> <toán tử so sánh> ALL (select <tên\_cột>...)
  - **EXISTS**: true nếu truy vấn con trả về ít nhất một dòng
    - where exists** (select...) ;



# Truy vấn con – Điều kiện & Toán tử

- Ví dụ

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

```
SELECT loại FROM LOAIMAYBAY
WHERE NSX='Boeing'
AND socho > ANY
  (SELECT socho FROM LOAIMAYBAY WHERE NSX='Airbus')
```

```
-- AND socho >
  (SELECT min(socho) FROM LOAIMAYBAY WHERE NSX='Airbus')
```

\* Tìm họ tên các phi công ở Pháp có số ngày làm việc lớn hơn ít nhất một phi công ở Anh

# Truy vấn con – Điều kiện & Toán tử

- Ví dụ

2. 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')

-- AND socho >
    (SELECT max(socho) FROM LOAIMAYBAY WHERE NSX='Boeing')
```

# Truy vấn con – Điều kiện & Toán tử

- Ví dụ

3. 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**

```
SELECT * FROM MAYBAY
WHERE MMB IN (SELECT MMB FROM CHUYENBAY
              WHERE noiden='paris')
```

# Sử dụng OUTER JOIN

- LEFT JOIN, RIGHT JOIN sử dụng thay cho truy vấn con trong trường hợp sau (phép trừ):

```
SELECT t1.* FROM Table1 t1
WHERE t1.ID NOT IN (SELECT t2.ID FROM Table2 t2)
```

Được thay bằng

```
SELECT t1.*
FROM Table1 t1 LEFT JOIN Table2 t2
                ON t1.ID = t2.ID
WHERE t2.ID IS NULL
```

– Hoặc

```
SELECT t1.*
FROM Table1 t1 RIGHT JOIN Table2 t2
                ON t1.ID = t2.ID
WHERE t1.ID IS NULL
```

# Sử dụng OUTER JOIN

- Ví dụ: Tìm phi công chưa làm việc ngày nào

- **Dùng truy vấn con**

```
SELECT hoten FROM PHICONG  
WHERE MPC NOT IN (SELECT MPC FROM LAMVIEC)
```

- **Tập hợp**

```
SELECT MPC FROM PHICONG  
EXCEPT  
SELECT MPC FROM LAMVIEC
```

- **Dùng kết nối mở rộng**

```
SELECT hoten  
FROM PHICONG p LEFT JOIN LAMVIEC l ON p.MPC=l.MPC  
WHERE songay IS NULL
```

# Các hàm kết tập

- Sử dụng trong mệnh đề **SELECT** hoặc **HAVING**
- Đầu vào: một tập các giá trị của một cột hoặc một *nhóm*
- Đầu ra: một giá trị duy nhất
- Các hàm trên các thuộc tính số:
  - SUM : tính tổng các giá trị của một thuộc tính của các bộ được chọn
  - AVG : tính giá trị trung bình
- Các hàm trên các thuộc tính số và kiểu dữ liệu khác:
  - MIN : giá trị nhỏ nhất
  - MAX : giá trị lớn nhất
  - COUNT : đếm số giá trị

# Các hàm kết tập

- Ví dụ:

**1.** Có tất cả bao nhiêu phi công ở nước Pháp

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

**2.** 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) Lon_nhat,  
          AVG(khoangcach) as Trung_binh  
FROM CHUYENBAY  
WHERE MPC=20;
```

# GROUP BY - Gom nhóm

- Mệnh đề GROUP BY được dùng kết hợp với các hàm kết tập để nhóm kết quả theo một hoặc nhiều cột

```
SELECT <ten_cot 1>[, <ten_cot 2>, ... ], <ham_ket_tap(ten_cot  
1i)>[, <ham_ket_tap(ten_cot 2i)>, ... ]
```

```
FROM <ten_bang 1>[, <ten_bang 2>, ... ]
```

```
[WHERE <dieu_kien>]
```

```
GROUP BY <ten_cot 1> [, <ten_cot 2>, ... ];
```

- Luật**: mỗi thuộc tính trong mệnh đề SELECT phải bao hàm trong hàm kết tập hoặc trong mệnh đề GROUP BY



# GROUP BY - Gom nhóm

- Ví dụ 1: Số phi công của mỗi nước

```
SELECT nuoc, count(*) so_phi_cong  
FROM PHICONG  
GROUP BY nuoc ;
```

	nuoc	so_phi_cong
1	Anh	1
2	Phap	3
3	Scotland	1
4	Uc	1

- Ví dụ 2: Tổng số ngày làm việc của mỗi phi công

```
SELECT l.MPC, hoten, sum(songay) tong_ngay  
FROM LAMVIEC l JOIN PHICONG p ON l.MPC=p.MPC  
GROUP BY l.MPC, hoten ;
```

# GROUP BY - HAVING

- 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: 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 ;
```

	nuoc	so_phi_cong
1	Phap	3

# SELECT – Lưu kết quả

- Cho phép lưu kết quả truy vấn vào một bảng mới
- **SQL Server**

```
SELECT <ten_cot>[,...] INTO <bang_moi>  
FROM <ten_bang>;
```

- *SQL Server còn cho phép lưu kết quả vào biến tạm với cú pháp:*

```
SELECT <ten_cot>[,...] INTO <#ten_bien> FROM <ten_bang>;
```

- Ví dụ:

```
SELECT * INTO temp  
FROM CONGTY
```

```
SELECT * INTO #temp  
FROM CONGTY
```

# SELECT – Lưu kết quả

- Cho phép lưu kết quả truy vấn vào một bảng mới
- **Oracle**

***CREATE TABLE <bang\_moi> AS***

*SELECT <ten\_cot>[,...]*

*FROM <ten\_bang>;*

- Ví dụ:

**CREATE TABLE Temp AS**

**SELECT \***

**FROM CONGTU**

# SELECT – Lưu kết quả

- Cho phép lưu kết quả truy vấn vào một bảng mới

- **MySQL**

- ***CREATE TABLE <bang\_moi> [AS]***

- SELECT <ten\_cot>[,...]*

- FROM <ten\_bang>;*

- Ví dụ:

- CREATE TABLE Temp**

- SELECT \***

- FROM CONGTU**

# SELECT – Thêm dữ liệu cho bảng

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

*INSERT INTO <ten\_bang>[(<ten\_cot>[,...]) ]  
<Lệnh SELECT>;*

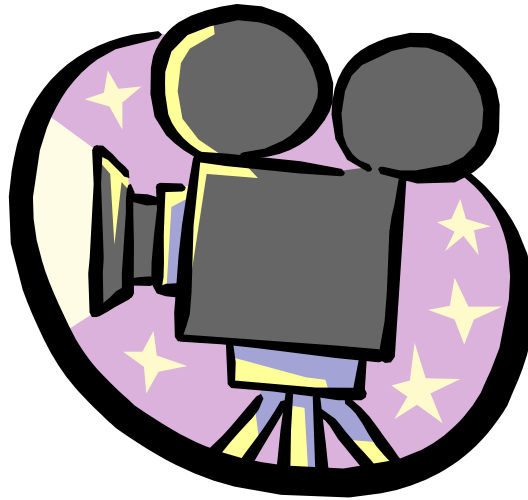
- Ví dụ:

```
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='Phap'
```

# SQL

## Data Control Language - DCL



# DCL

- Ngôn ngữ điều khiển dữ liệu cho phép điều khiển việc truy xuất đến các đối tượng của CSDL
- Các lệnh cơ bản
  - CREATE USER : tạo người dùng
  - GRANT : gán quyền trên các đối tượng
  - REVOKE: gỡ bỏ quyền



# DCL

- Ví dụ

- **CREATE USER** user1 IDENTIFIED BY '123@456Monkey' ;
- **GRANT ALL PRIVILEGES TO** user1
- **GRANT SELECT, INSERT ON PHICONG TO** user1  
**WITH GRANT OPTION**
- REVOKE SELECT, INSERT ON PHICONG FROM** user1  
**CASCADE**
- Oracle :
- **GRANT CREATE SESSION, CREATE TABLE, CREATE PROCEDURE TO** user1;