



Trường Đại Học Bách Khoa Tp.HCM  
Hệ Đào Tạo Từ Xa  
Khoa Khoa Học và Kỹ Thuật Máy Tính

## TIN HỌC QUẢN LÝ

### Chương 5

### QUERY

(Phần 2)

Trần Quang  
quangt@cse.hcmut.edu.vn

# Ngôn ngữ SQL

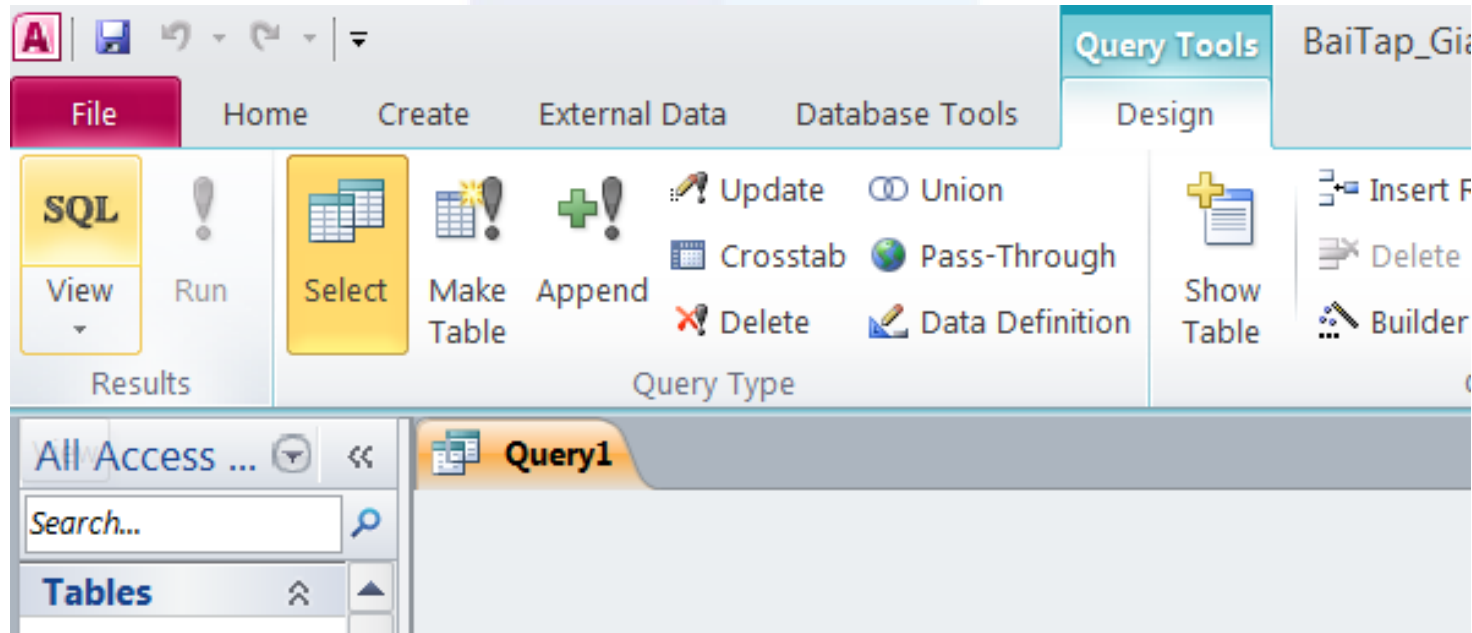


# Giới thiệu

- SQL (Structured Query Language) là ngôn ngữ truy vấn dữ liệu có cấu trúc
- Dùng rộng rãi trong các hệ quản trị CSDL quan hệ
- 1986, ANSI và ISO công bố chuẩn đầu tiên cho SQL: SQL-86 hay SQL1
- 1992, ISO công bố phiên bản sửa đổi gọi là SQL2 hay SQL-92
- 1999, SQL3 hay SQL-99 ra đời hỗ trợ quản trị dữ liệu hướng đối tượng
- Cuối năm 2003, chuẩn SQL-2003 ra đời
- Ghi chú: sinh viên sẽ được học đầy đủ về ngôn ngữ này trong môn “Cơ sở dữ liệu”

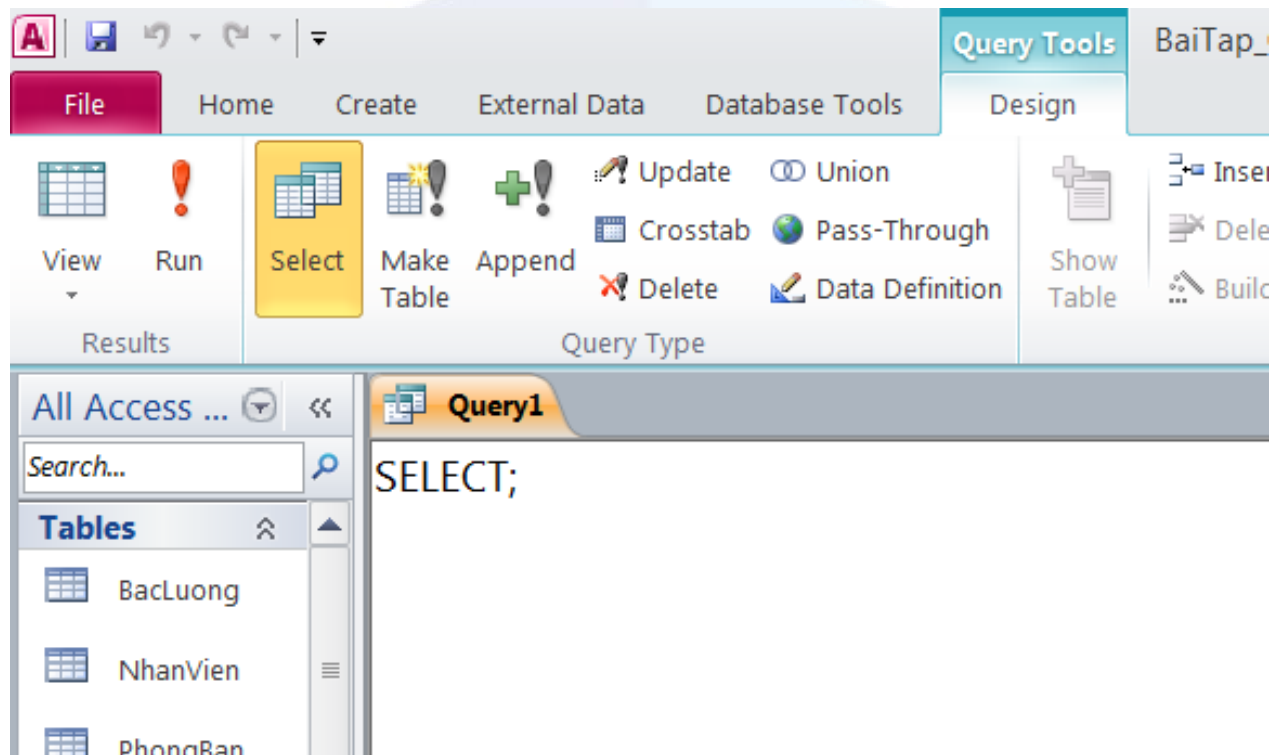
# Mở cửa sổ thiết kế query bằng SQL

- Chọn **Create** → **Query Design**
- Khi cửa sổ Show Table hiện lên ta chọn **Close**
- Bấm vào nút **SQL**



# Mở cửa sổ thiết kế query bằng SQL

- Cửa sổ thiết kế hiện ra để ta gõ vào câu SQL.
- Để thực hiện câu query ta bấm vào nút **Run**

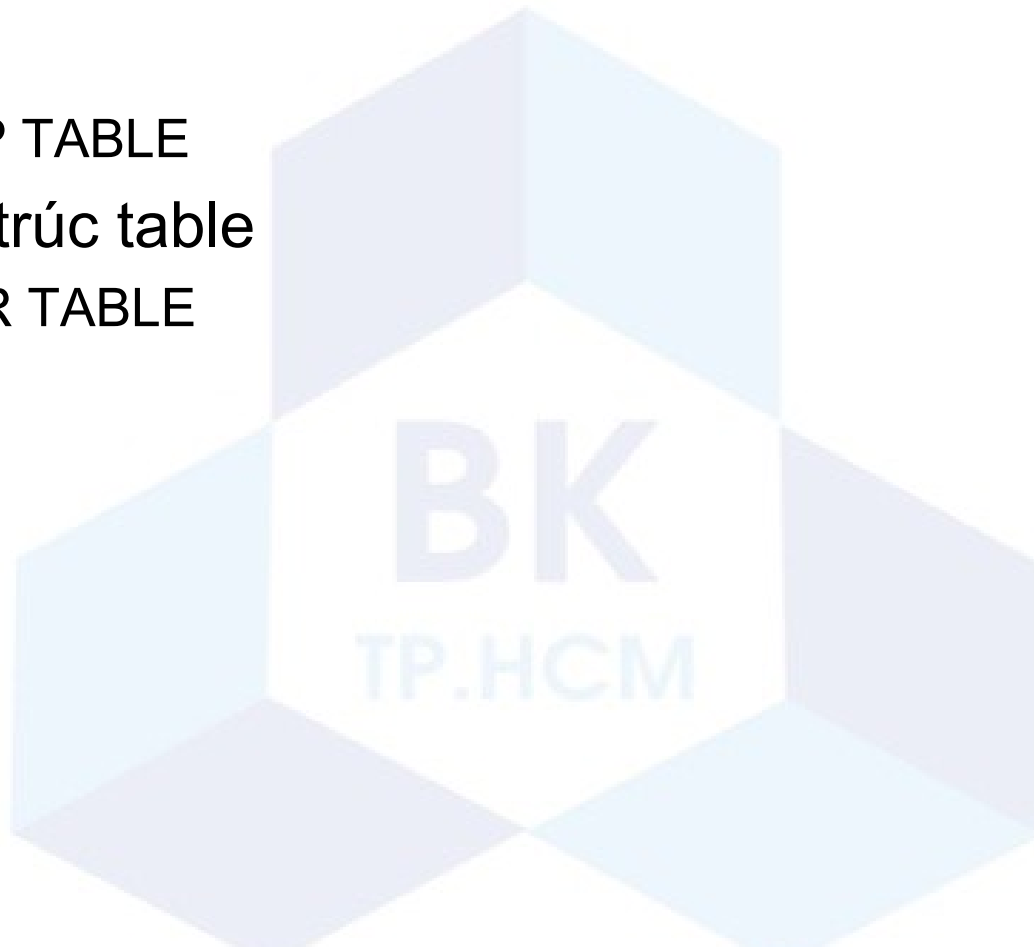


# Các ngôn ngữ con SQL

- DDL (Data Definition Language): liên quan đến việc thêm, sửa cấu trúc và xóa các đối tượng CSDL như table
  - CREATE, DROP, ALTER
- DML (Data Manipulation Language): dùng để thao tác trên dữ liệu trong cơ sở dữ liệu
  - INSERT, DELETE, UPDATE, SELECT
- DCL (Data Control Language): ngôn ngữ kiểm soát dữ liệu
  - GRANT, REVOKE, DENY

# DDL

- Tạo table:  
CREATE TABLE
- Xóa table  
DROP TABLE
- Sửa cấu trúc table  
ALTER TABLE



# CREATE TABLE

- Tạo bảng mới → đặt tên bảng và khai báo các thuộc tính (cột) và kiểu dữ liệu của từng cột
- Ràng buộc NOT NULL có thể khai báo cho từng cột

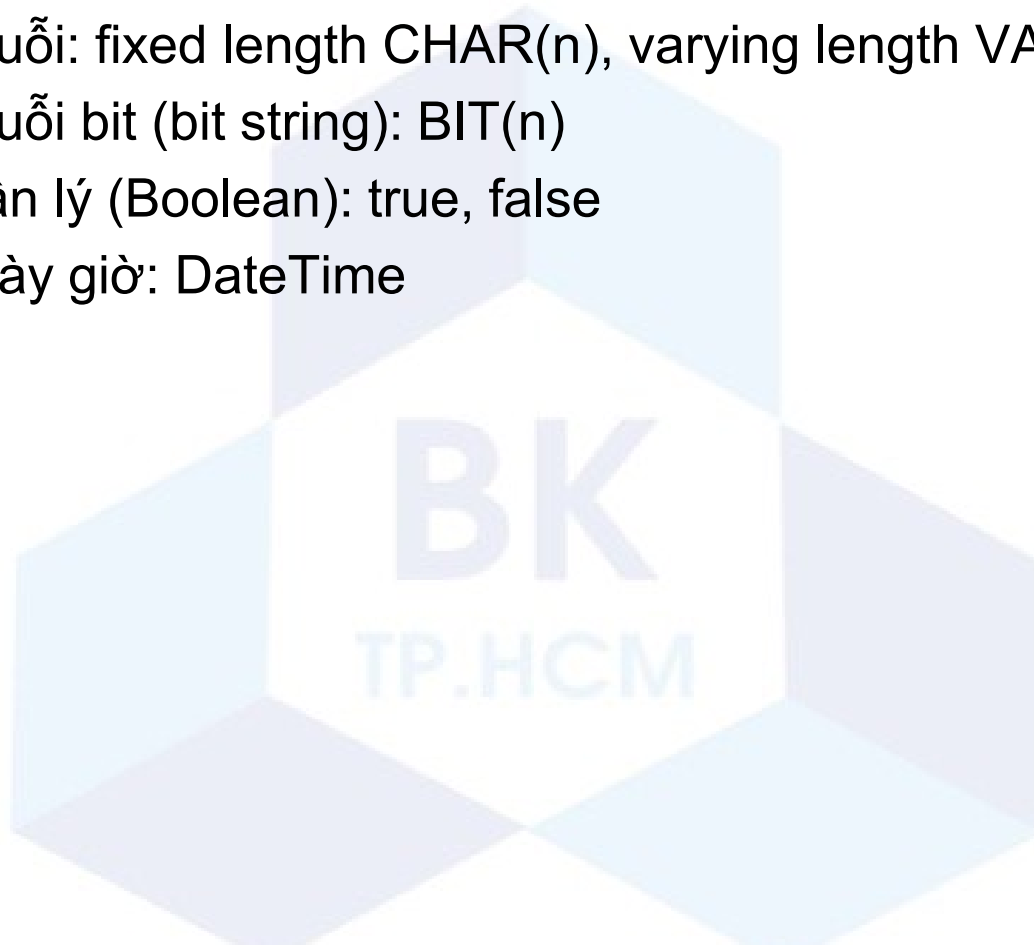
```
CREATE TABLE PhongBan (  
    MSPB    CHAR(2),  
    TenPB   VARCHAR(10) NOT NULL,  
    MSNQL   CHAR(4))
```

BK  
TP.HCM



# CREATE TABLE

- Kiểu dữ liệu (data type)
  - Kiểu số: INT (INTEGER), FLOAT (REAL), DOUBLE, ...
  - Kiểu chuỗi: fixed length CHAR(n), varying length VARCHAR(n), ..
  - Kiểu chuỗi bit (bit string): BIT(n)
  - Kiểu luận lý (Boolean): true, false
  - Kiểu ngày giờ: DateTime



# DROP TABLE

- Dùng để xóa một bảng

DROP TABLE PhongBan



# ALTER TABLE

- Thêm một cột vào bảng
  - Cột mới thêm sẽ chứa giá trị rỗng ở các mẫu tin đã có sẵn trong bảng → không thể đặt ràng buộc NOT NULL cho cột này
  - ALTER TABLE NhanVien  
ADD DiaChi Varchar(10)
- Sửa khai báo của một cột trong bảng
  - ALTER TABLE NhanVien  
ALTER COLUMN DiaChi Varchar(20)
- Xóa một cột trong bảng
  - ALTER TABLE NhanVien  
DROP COLUMN DiaChi

# PRIMARY KEY và UNIQUE KEY

- Cài đặt ràng buộc PRIMARY KEY và UNIQUE KEY

- Cách 1:

```
CREATE TABLE NhanVien (  
    MSNV      CHAR(4) PRIMARY KEY,  
    HOTEN     VARCHAR(20) NOT NULL,  
    LUONG     INTEGER,  
    SOCMND    CHAR(9) UNIQUE)
```

# PRIMARY KEY và UNIQUE KEY

- Cài đặt ràng buộc PRIMARY KEY và UNIQUE

- Cách 2:

```
CREATE TABLE NhanVien (  
    MSNV      CHAR(4),  
    HOTEN     VARCHAR(20) NOT NULL,  
    LUONG     INTEGER,  
    SOCMND    CHAR(9),  
    PRIMARY KEY(MSNV),  
    UNIQUE(SOCMND))
```

# PRIMARY KEY và UNIQUE KEY

- Ghi chú: Trường hợp PRIMARY KEY hoặc UNIQUE KEY bao gồm nhiều cột thì chỉ có thể sử dụng được cách 2

```
CREATE TABLE ChiTietHD (  
    MSHD      CHAR(4),  
    MSSP      CHAR(4),  
    SoLuong   INTEGER,  
    DonGia    INTEGER,  
    PRIMARY KEY(MSHD, MSSP))
```

# FOREIGN KEY

- Cài đặt ràng buộc FOREIGN KEY

- Cách 1:

```
CREATE TABLE NhanVien (  
    MSNV    CHAR(4)        PRIMARY KEY,  
    HOTEN   VARCHAR(20)    NOT NULL,  
    MSPB    CHAR(4)        REFERENCES PhongBan,  
    LUONG   INTEGER )
```

BK  
TP.HCM

# FOREIGN KEY

- Cài đặt ràng buộc FOREIGN KEY

- Cách 2:

```
CREATE TABLE NhanVien (  
    MSNV    CHAR(4)          PRIMARY KEY,  
    HOTEN   VARCHAR(20)      NOT NULL,  
    MSPB    CHAR(4),  
    LUONG   INTEGER,  
    FOREIGN KEY (MSPB) REFERENCES PhongBan)
```



# DDL (tóm tắt)

- Tạo table:

```
CREATE TABLE PhongBan (  
    MSPB CHAR(2) PRIMARY KEY,  
    TenPB VARCHAR(10),  
    MSNQLCHAR(4))
```

- Xóa table

- DROP TABLE PhongBan

- Sửa cấu trúc table

- ALTER TABLE NhanVien ADD DiaChi Varchar(10)
- ALTER TABLE NhanVien ALTER COLUMN DiaChi Varchar(20)
- ALTER TABLE NhanVien DROP COLUMN DiaChi

# DML

SQL cung cấp 3 lệnh để cập nhật dữ liệu trong các bảng:

- INSERT: Thêm dữ liệu vào bảng
- DELETE: Xóa dữ liệu trong bảng
- UPDATE: Sửa chữa dữ liệu trong bảng



# INSERT

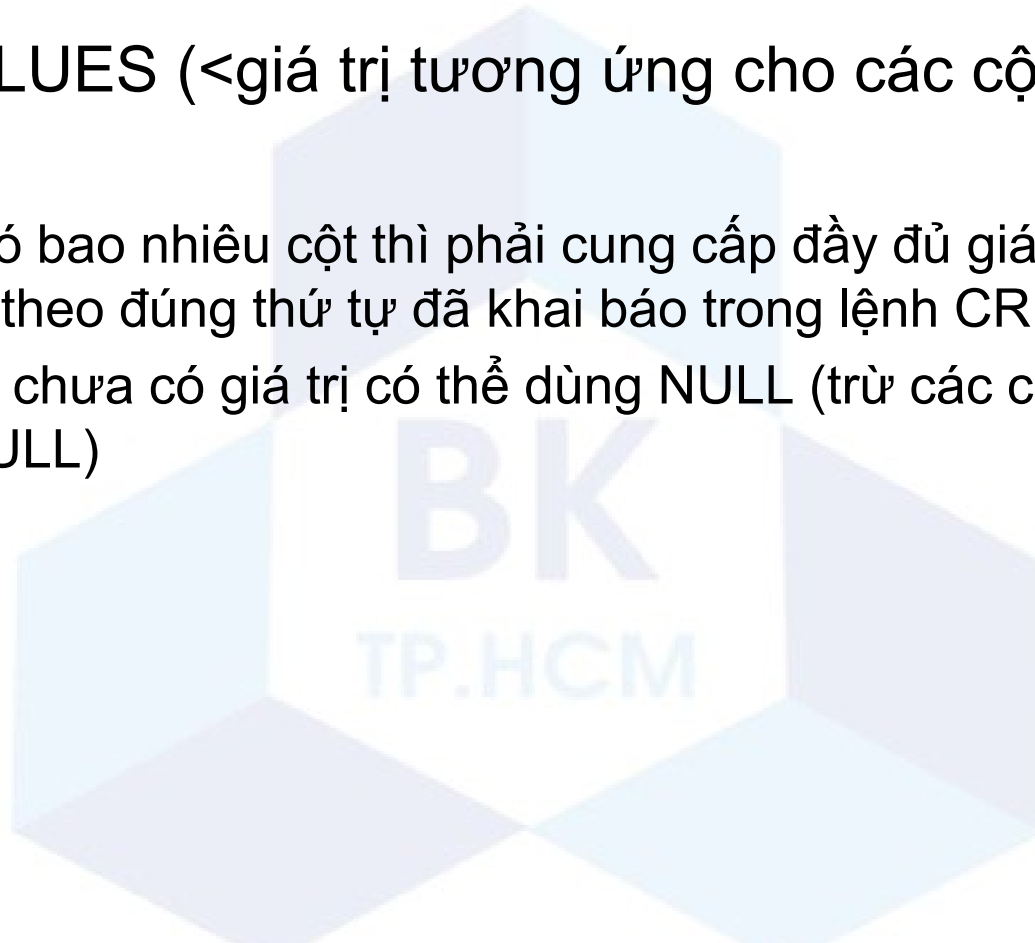
- Cú pháp 1:

INSERT INTO <table>

VALUES (<giá trị tương ứng cho các cột>)

- Ghi chú:

- Table có bao nhiêu cột thì phải cung cấp đầy đủ giá trị cho tất cả các cột theo đúng thứ tự đã khai báo trong lệnh CREATE TABLE
- Các cột chưa có giá trị có thể dùng NULL (trừ các cột có đặc tính NOT NULL)



# INSERT

- Giả sử ta có table NhanVien như sau:

```
CREATE TABLE NhanVien  
  (MSNV char(4) PRIMARY KEY,  
   HoTen varchar(20) NOT NULL,  
   DiaChi varchar(50),  
   Luong int)
```

- Ta có thể thực hiện các lệnh INSERT sau:

```
INSERT INTO NhanVien  
  VALUES ('01', 'An', '123 Le Loi', 300)  
INSERT INTO NhanVien  
  VALUES ('02', 'Binh', NULL , NULL)
```

# INSERT

- Cú pháp 2:

```
INSERT INTO <table>(<danh sách các cột>)  
VALUES (<giá trị tương ứng cho các cột>)
```

- Ghi chú:

- Danh sách các cột bắt buộc phải có các cột thuộc PRIMARY KEY và các cột NOT NULL
- Giá trị cung cấp phải theo đúng thứ tự
- Các cột không có trong danh sách, nhưng có khai báo DEFAULT sẽ lấy giá trị mặc định

- Ví dụ:

```
INSERT INTO NhanVien (MSNV, HoTen)  
VALUES ('03', 'Chi')
```

# DELETE

- Cú pháp:

```
DELETE FROM <table>  
WHERE <điều kiện xóa>
```

- Ghi chú:

- Chỉ xóa các mẫu tin thỏa điều kiện xóa
- Ràng buộc tham chiếu sẽ được kiểm tra.
- Nếu có DELETE CASCADE các mẫu tin ở các bảng tương ứng cũng bị xóa theo
- Nếu không có mệnh đề WHERE, toàn bộ dữ liệu trong table sẽ bị xóa sạch

- Ví dụ:

```
DELETE FROM NhanVien WHERE MSNV='03'
```

# UPDATE

- Cú pháp:

UPDATE <table>

SET <cột i> = <giá trị mới>,  
<cột j> = <giá trị mới>, ...

WHERE <điều kiện>

- Ghi chú:

- Nếu thiếu mệnh đề WHERE, tất cả các mẫu tin đều được sửa dữ liệu
- Ràng buộc tham chiếu sẽ được kiểm tra

# UPDATE

- Ví dụ 1: Sửa địa chỉ và lương của nhân viên có mã số '01'

```
UPDATE NV
```

```
SET DiaChi = '20 Nguyen Du',
```

```
    Luong = 500
```

```
WHERE MSNV = '01'
```

- Ví dụ 2: Tăng lương 10% cho tất cả nhân viên

```
UPDATE NV
```

```
SET Luong = Luong * 1.1
```

BK  
TP.HCM



# DML (tóm tắt)

## ■ Thêm dữ liệu vào table

- INSERT INTO NhanVien VALUES ('01', 'An', '123 Le Loi', 300)
- INSERT INTO NhanVien (MSNV, HoTen) VALUES ('03', 'Chi')

## ■ Xóa dữ liệu trong table

- DELETE FROM NhanVien WHERE MSNV='03'

## ■ Cập nhật dữ liệu trong table

- UPDATE NhanVien  
SET DiaChi = '20 Nguyen Du', Luong = 500  
WHERE MSNV = '01'

# SELECT

- Dùng câu lệnh SELECT để thực hiện việc truy vấn dữ liệu.  
Câu lệnh cơ bản:

<b>SELECT</b>	<danh sách các cột kết quả>
<b>FROM</b>	<danh sách table>
<b>WHERE</b>	<điều kiện lọc>
<b>GROUP BY</b>	<danh sách các nhóm>
<b>HAVING</b>	<điều kiện lọc trên nhóm>
<b>ORDER BY</b>	<các biểu thức sắp xếp>

# Dữ liệu mẫu

PHONGBAN	
MSPB	TENPB
P1	Hanh Chanh
P2	Ke Toan
P3	Kinh Doanh
P4	Nghiên cứu

# Dữ liệu mẫu

NHANVIEN								
MSNV	HO	TEN	NGAY SINH	NGAY VAOLAM	PHAI	MSPB	LUONG	MSNQL
0001	Le Van	An	12/5/1962	1/8/1980	Yes	P1	2200	
0002	Nguyen	Minh	20/6/1975	6/9/1995	Yes	P1	1000	0001
0003	Ly Thi	Nga	17/6/1980	26/8/2000	No	P2	2200	0001
0004	Tran Van	Tuan	1/1/1978	16/5/1996	Yes	P3	1000	0001
0005	Le Thi	Chi	12/3/1981	20/6/2003	No	P1	800	0002
0006	Ngo Thu	An	11/7/1978	15/5/2001	No	P1	300	0005
0007	Mai Kim	Chi	24/6/1970	15/8/1999	No	P2	400	0003
0008	Tran Tuan	Anh	16/5/1976	20/5/1999	Yes	P3	800	0004
0009	Le Ngoc	Mai	14/3/1979	17/4/1998	No	P3	400	0004

# Câu SELECT đơn giản

- Hai mệnh đề tối thiểu phải có trong câu truy vấn là SELECT và FROM

- Cú pháp:

SELECT <Danh sách các cột>

FROM <Table chứa dữ liệu>

- Ghi chú:

- Tương ứng với phép chiếu trong đại số quan hệ
- Các cột phân cách nhau bởi dấu phẩy (,)

- Ví dụ: Hiển thị mã số, họ, tên của tất cả nhân viên

```
SELECT MSNV, Ho, Ten  
FROM NhanVien
```

# Dấu \*

- Ký hiệu \* đại diện cho tất cả các cột
    - Ví dụ: Hiển thị tất cả thông tin của tất cả các nhân viên
- ```
SELECT *  
FROM NhanVien
```



# Cột tính toán (calculated column)

- Để tạo một cột tính toán từ 1 biểu thức nào đó. Ta dùng công thức:

<Biểu thức tính toán> AS <tên cột>

- Ví dụ:

Hiển thị mã số, họ tên (ghép chung 1 cột) của tất cả nhân viên.

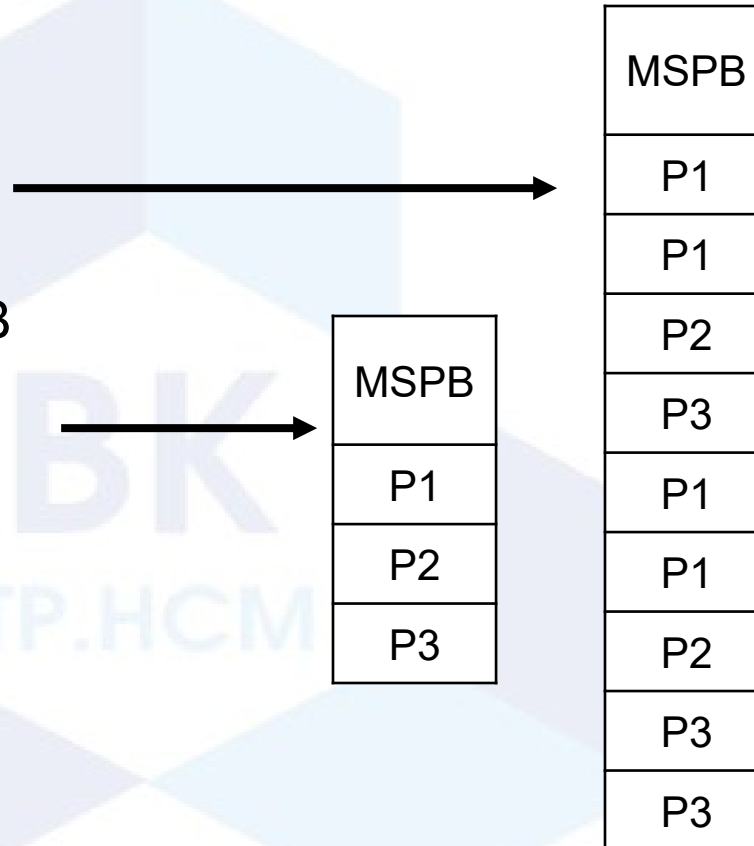
```
SELECT MSNV, Ho + ' ' + Ten AS "Ho Ten"  
FROM NhanVien
```

# DISTINCT

- Từ khóa DISTINCT giúp loại các mẫu tin trùng nhau trong kết quả
- Ví dụ:

SELECT MSPB  
FROM NHANVIEN

SELECT DISTINCT MSPB  
FROM NHANVIEN





# TOP n

- Từ khóa TOP n, với n là một số nguyên dương sẽ chỉ định kết quả chỉ lấy ra n mẫu tin từ trên xuống

- Ví dụ:

```
SELECT TOP 3 MSNV, Ho, Ten, Luong  
FROM NHANVIEN
```

| MSNV | HO     | TEN  | LUONG |
|------|--------|------|-------|
| 0001 | Le Van | An   | 2200  |
| 0002 | Nguyen | Minh | 1000  |
| 0003 | Ly Thi | Nga  | 2200  |

# Mệnh đề WHERE

- Dùng để thiết lập các điều kiện lọc dữ liệu. Chỉ các mẫu tin thỏa điều kiện trong mệnh đề WHERE mới được lấy ra → tương ứng phép chọn (trích ngang) trong đại số quan hệ
- Cú pháp:

**WHERE <điều kiện lọc>**

- Trong biểu thức của điều kiện lọc ta có thể sử dụng các phép so sánh: **=, <>, >, >=, <, <=**

Ví dụ: Hiển thị các nhân viên có lương trên 800

```
SELECT *  
FROM NhanVien  
WHERE Luong > 800
```

# Mệnh đề WHERE

- Ta có thể dùng các phép toán logic như **NOT**, **AND**, **OR** để kết hợp các điều kiện

- Ví dụ:

Hiển thị các nhân viên nữ (Phai=No) của phòng có MSPB='P1'

```
SELECT *  
FROM NhanVien  
WHERE (Phai=No) And (MSPB='P1')
```

# Toán tử LIKE

- Toán tử LIKE: dùng để so sánh với 1 mẫu (pattern) cho trước. Thường dùng với các ký tự thay thế:
  - \* (đại diện cho 0 đến nhiều ký tự)
  - ? (đại diện cho 1 ký tự)

- Ví dụ:

Hiển thị các nhân viên mà tên bắt đầu bằng chữ 'A'

```
SELECT *  
FROM NhanVien  
WHERE Ten LIKE 'A*'
```

# BETWEEN ... AND ...

- Dùng BETWEEN ... AND ... để kiểm tra giá trị có nằm trong 1 khoảng nào

- Ví dụ:

Liệt kê các nhân viên có lương từ 500 đến 1000

```
SELECT *
```

```
FROM NhanVien
```

```
WHERE Lương BETWEEN 500 AND 1000
```



# Toán tử IN

- Dùng toán tử IN để kiểm tra giá trị có nằm trong 1 tập hợp nào đó.
- Ví dụ:

Liệt kê các nhân viên của phòng 'P1' và 'P3'

```
SELECT *  
FROM NhanVien  
WHERE MSPB IN ('P1', 'P3')
```

BK  
TP.HCM

# IS NULL

- Dùng IS NULL để kiểm tra 1 giá trị là rỗng, và IS NOT NULL để kiểm tra 1 giá trị là khác rỗng

- Ví dụ:

Hiển thị các nhân viên không có người quản lý (tức MSNQL để trống)

```
SELECT *  
FROM NhanVien  
WHERE MSNQL is null
```

# Phép kết

- Trong trường hợp dữ liệu của câu truy vấn lấy từ nhiều table, ta sử dụng phép kết để kết nối các table nguồn lại với nhau.
- Trong trường hợp này khi cần dùng 1 field nào đó ta phải viết theo cú pháp sau:

**<TênTable>.<TênField>**

BK  
TP.HCM



# Phép kết

- Cách 1: Điều kiện kết được đặt trong mệnh đề WHERE. Cú pháp như sau:

```
SELECT <danh sách các cột>  
FROM <table1>, <table2>, ..., <tablen>  
WHERE <đklk1> AND <đklk2> AND ...  
... AND đklk(n-1) AND <điều kiện lọc>
```

- Ví dụ: Hiển thị MSNV, Ho, Ten, MSPB, TenPB của các nhân viên thuộc phòng 'Ke Toan'

```
SELECT NhanVien.MSNV, NhanVien.Ho, NhanVien.Ten,  
       NhanVien.MSPB, PhongBan.TenPB  
FROM   NhanVien, PhongBan  
WHERE  (NhanVien.MSPB = PhongBan.MSPB)  
       AND (PhongBan.TenPB='Ke Toan')
```

# Phép kết

- Cách 2: Điều kiện kết được đặt trong mệnh đề FROM. Cú pháp như sau:

```
SELECT <danh sách các cột>  
FROM (<table1>INNER JOIN<table2>ON<đklk1>)  
     INNER JOIN <table3> ON <đklk2>...  
WHERE <điều kiện lọc>
```

- Ví dụ: Hiển thị MSNV, Ho, Ten, MSPB, TenPB của các nhân viên thuộc phòng 'Ke Toan'

```
SELECT NhanVien.MSNV, NhanVien.Ho, NhanVien.Ten,  
       NhanVien.MSPB, PhongBan.TenPB  
FROM   NhanVien INNER JOIN PhongBan  
       ON (NhanVien.MSPB = PhongBan.MSPB)  
WHERE  PhongBan.TenPB='Ke Toan'
```

# Phép kết

- Trong trường tên field không bị trùng trong các table sử dụng trong câu truy vấn, thì ta có thể không cần ghi tên table phía trước tên field

- Ví dụ:

```
SELECT MSNV, Ho, Ten, NhanVien.MSPB, TenPB
FROM NhanVien INNER JOIN PhongBan
      ON (NhanVien.MSPB = PhongBan.MSPB)
WHERE TenPB='Ke Toan'
```

- Ghi chú:

Trong câu truy vấn này field MSPB là không thể bỏ tên table phía trước, vì nó có trong cả 2 table NhanVien và PhongBan

# ALIAS

- Table có thể được đặt bí danh (alias). Thường là tên tắt để thuận tiện khi viết lệnh

- Cú pháp:

FROM <tên table> **AS** <bí danh>

- Ví dụ:

```
SELECT MSNV, Ho, Ten, nv.MSPB, TenPB
```

```
FROM NhanVien AS nv INNER JOIN
```

```
PhongBan AS pb ON (nv.MSPB = pb.MSPB)
```

```
WHERE TenPB='Ke Toan'
```

# Hàm thống kê

- Có thể sử dụng các hàm như SUM, COUNT, MIN, MAX, AVG, ... để thống kê dữ liệu

- Ví dụ:

```
SELECT Count(MSNV) AS TongSoNV  
FROM NhanVien
```

**TongSoNV**

9

- Chú ý:

```
SELECT Count(MSPB) AS KQ  
FROM NhanVien
```

**KQ**

9

```
SELECT Count(Distinct MSPB) AS KQ  
FROM NhanVien
```

**KQ**

3

```
SELECT Count(MSNQL) AS KQ  
FROM NhanVien
```

**KQ**

8

# Mệnh đề GROUP BY

- Dùng để nhóm dữ liệu. Thường dùng để nhóm dữ liệu lại trước khi thực hiện các hàm thống kê trên từng nhóm.
- Cú pháp:

**GROUP BY <nhóm cấp 1>, <nhóm cấp 2>, ...**

- Ví dụ: Thống kê xem mỗi phòng ban có bao nhiêu nhân viên. Chỉ hiển thị 2 cột: MSPB và số lượng nhân viên của phòng đó

```
SELECT MSPB, Count(MSNV) AS TongSoNV  
FROM NhanVien  
GROUP BY MSPB
```

# Mệnh đề GROUP BY

- Ví dụ: Thống kê xem mỗi phòng ban có bao nhiêu nhân viên. Hiện thị 3 cột: MSPB, tên phòng ban và số lượng nhân viên của phòng đó

```
SELECT PhongBan.MSPB, TenPB, Count(MSNV) AS TongSoNV  
FROM NhanVien INNER JOIN PhongBan  
    ON (NhanVien.MSPB = PhongBan.MSPB)  
GROUP BY PhongBan.MSPB, TenPB
```

# Mệnh đề HAVING

- Đi kèm với mệnh đề GROUP BY khi cần đặt điều kiện lọc trên các dữ liệu đã thống kê theo nhóm.
- Ví dụ: Liệt kê các phòng có số lượng nhân viên từ 3 người trở lên

```
SELECT MSPB, Count(MSNV) AS TSNV  
FROM NhanVien  
GROUP BY MSPB  
HAVING Count(MSNV) >=3
```

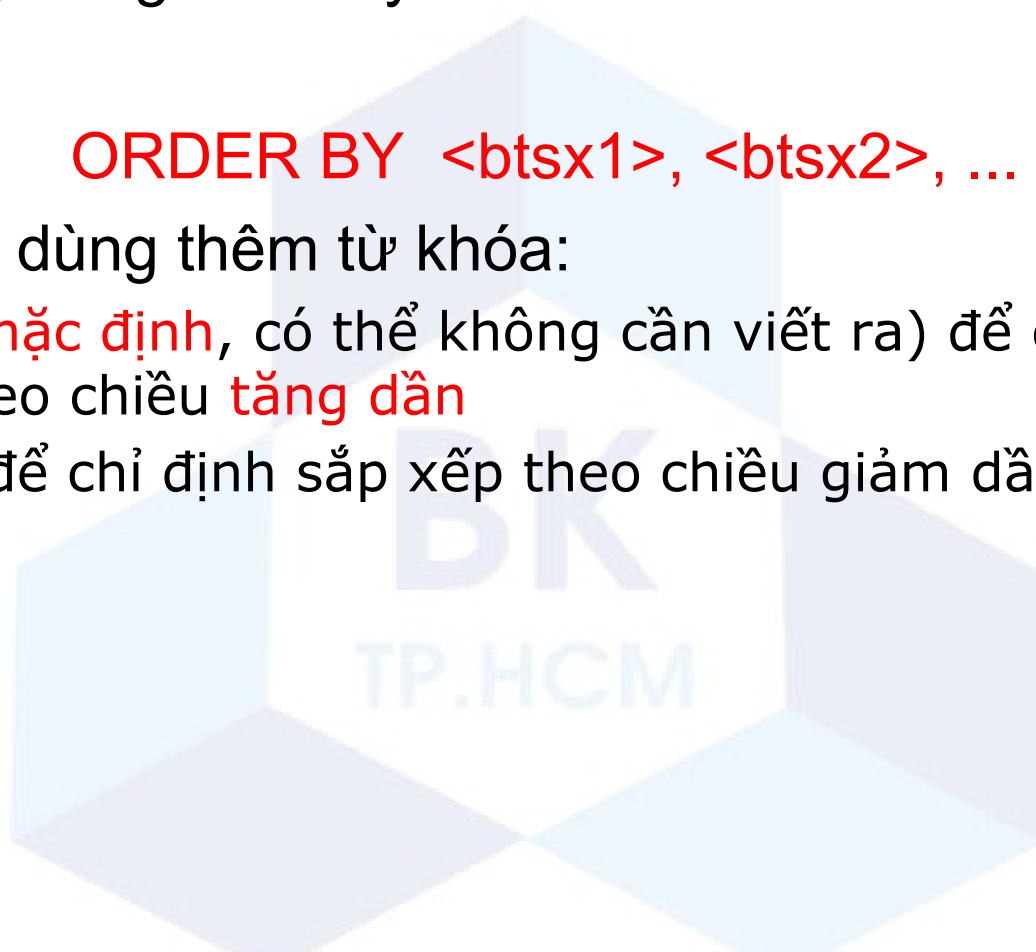


# Mệnh đề ORDER BY

- Dùng để sắp xếp kết quả hiển thị. Mệnh đề này được đặt cuối cùng trong câu truy vấn
- Cú pháp:

**ORDER BY** <btsx1>, <btsx2>, ...

- Ta có thể dùng thêm từ khóa:
  - **ASC** (mặc định, có thể không cần viết ra) để chỉ định sắp xếp theo chiều **tăng dần**
  - **DESC** để chỉ định sắp xếp theo chiều giảm dần



# Mệnh đề ORDER BY

- Ví dụ: Hiển thị danh sách nhân viên, sắp thứ tự theo tên, cùng tên xếp theo họ  

```
SELECT *  
FROM NhanVien  
ORDER BY Ten, Ho
```
- Ví dụ: Hiển thị danh sách nhân viên, sắp thứ tự theo phòng ban, cùng phòng ban xếp theo mức lương giảm dần  

```
SELECT *  
FROM NhanVien  
ORDER BY MSPB, Luong DESC
```

# ORDER BY và TOP n

- Ví dụ: Tìm nhân viên có lương cao nhất tại công ty

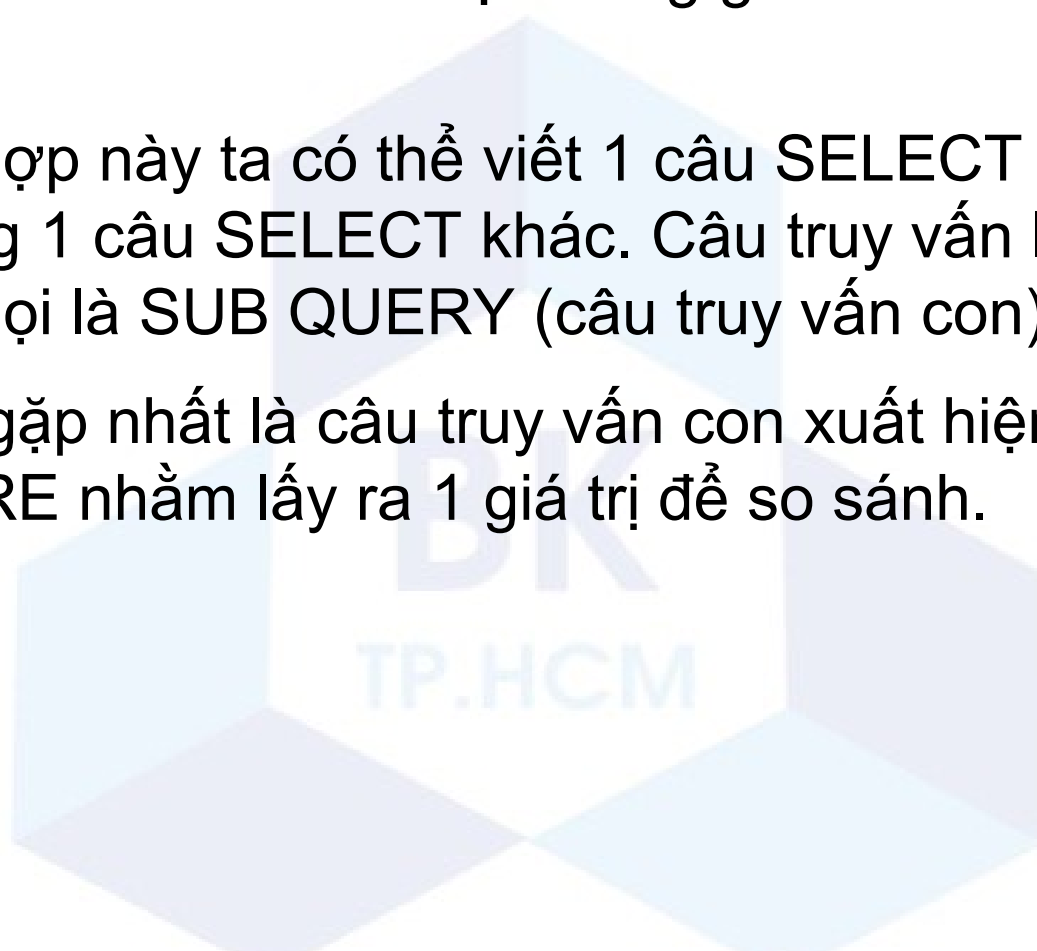
```
SELECT TOP 1 MSNV, Ho, Ten, Luong  
FROM NHANVIEN  
ORDER BY Luong DESC
```

| MSNV | HO     | TEN | LUONG |
|------|--------|-----|-------|
| 0001 | Le Van | An  | 2200  |
| 0003 | Ly Thi | Nga | 2200  |

- Giải thích: Hồ sơ nhân viên sẽ được sắp xếp theo lương giảm dần, vì vậy người có lương cao nhất sẽ nằm trên cùng. Dùng SELECT TOP 1 sẽ lấy ra được người trên cùng (tức người có lương cao nhất)

# SUB QUERY

- Trong một số trường hợp ta cần lấy kết quả của một câu truy vấn khác để làm dữ liệu trung gian cho câu truy vấn chính.
- Trường hợp này ta có thể viết 1 câu SELECT mà bên trong lại có lồng 1 câu SELECT khác. Câu truy vấn lồng bên trong ta gọi là SUB QUERY (câu truy vấn con).
- Thường gặp nhất là câu truy vấn con xuất hiện trong mệnh đề WHERE nhằm lấy ra 1 giá trị để so sánh.



# SUB QUERY

## Trường hợp 1:

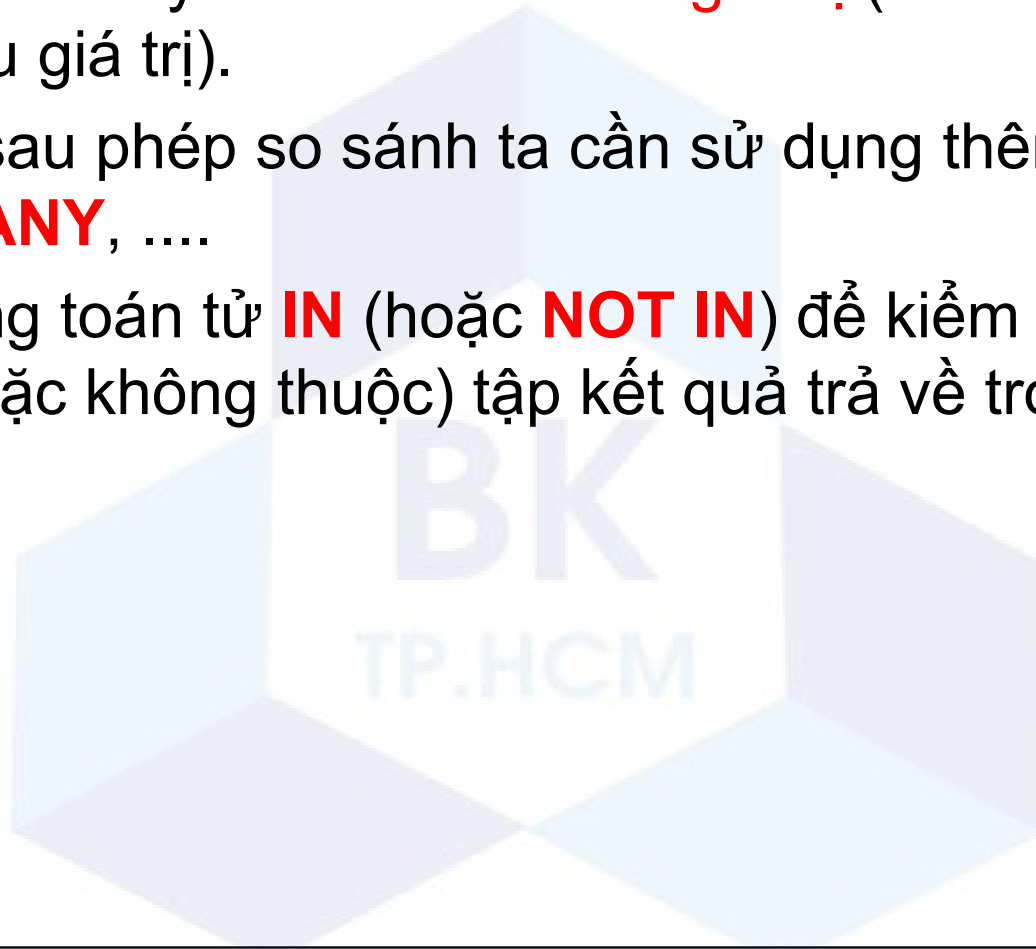
- Kết quả của câu truy vấn con **trả về 1 giá trị** → ta có thể dùng trực tiếp các phép so sánh **=, >, >=, <, <=, <>**
- Ví dụ: Hiển thị các nhân viên hưởng lương cao nhất trong công ty.

```
SELECT *  
FROM NhanVien  
WHERE Luong = (SELECT MAX(Luong)  
               FROM NhanVien )
```

# SUB QUERY

## Trường hợp 2:

- Kết quả câu truy vấn trả về **nhiều giá trị** (ta xem như 1 tập hợp nhiều giá trị).
- Lúc này sau phép so sánh ta cần sử dụng thêm các lượng từ **ALL, ANY, ...**
- Hoặc dùng toán tử **IN** (hoặc **NOT IN**) để kiểm tra giá trị có thuộc (hoặc không thuộc) tập kết quả trả về trong câu truy vấn con



# SUB QUERY

- Ví dụ 1: Hiển thị các nhân viên hưởng lương cao nhất trong công ty.

```
SELECT *  
FROM NhanVien  
WHERE Luong >= ALL (SELECT Luong  
                     FROM NhanVien)
```

- Ví dụ 2: Cho biết các phòng nào hiện không có danh sách nhân viên.

```
SELECT *  
FROM PhongBan  
WHERE MSPB NOT IN (SELECT MSPB  
                  FROM NhanVien)
```

# CORRELATED SUBQUERY

- Trong các câu sub query thông thường, câu truy vấn con chỉ thực hiện một lần để lấy kết quả, sau đó dùng kết quả này để thực hiện tiếp cho câu truy vấn cha.
- Có một số trường hợp câu truy vấn con khi thực hiện cần tham chiếu đến dữ liệu của table nằm ở câu truy vấn bên ngoài. Trường hợp này câu truy vấn con sẽ được thực hiện nhiều lần tương ứng với từng mẫu tin trong câu truy vấn cha.
- Câu truy vấn dạng này gọi là **correlated subquery** (câu truy vấn con tương quan)



# CORRELATED SUBQUERY

- Ví dụ: Cho biết nhân viên có lương cao nhất của từng phòng

Để tìm ra các nhân viên này, ta cần lấy lương của nhân viên đó so sánh với lương cao nhất của phòng mà nhân viên đó đang làm việc.

```
SELECT *  
FROM NhanVien a  
WHERE Luong = (SELECT Max(Luong)  
                FROM NhanVien b  
                WHERE b.MSPB = a.MSPB)
```

# EXISTS

- Từ khóa EXISTS dùng để kiểm tra câu truy vấn con có trả về ít nhất một mẫu tin hay không. Thường dùng với correlated query.
- Ví dụ: Liệt kê các phòng hiện không có danh sách nhân viên

```
SELECT *  
FROM PhongBan  
WHERE NOT EXISTS  
    (SELECT *  
     FROM NhanVien  
     WHERE NhanVien.MSPB=PhongBan.MSPB)
```

# SUB QUERY ở mệnh đề FROM

- Ví dụ: Lấy ra các nhân viên có lương cao nhất của mỗi phòng

```
SELECT MSNV, Ho, Ten, NhanVien.MSPB, Luong
FROM NHANVIEN,
    ( SELECT MSPB, MAX(LUONG) AS LgMAX
      FROM NHANVIEN
    GROUP BY MSPB) AS Temp
WHERE (NHANVIEN.MSPB = Temp.MSPB)
      AND (NHANVIEN.LUONG= Temp.LgMAX)
```

# OUTER JOIN

- Phép kết mà ta đã khảo sát gọi là phép kết trong (INNER JOIN). Trong phép kết này, chỉ những mẫu tin thỏa điều kiện liên kết mới được lấy ra.
- Giả sử ta có 2 table NHANVIEN và PHONGBAN, với số liệu như sau:

| PHONGBAN |            |
|----------|------------|
| MSPB     | TenPB      |
| P1       | Hanh Chanh |
| P2       | Ke Toan    |
| P3       | Kinh Doanh |

| NHANVIEN |       |      |
|----------|-------|------|
| MSNV     | HoTen | MSPB |
| 0001     | An    | P1   |
| 0002     | Minh  | P1   |
| 0003     | Tuan  | P2   |
| 0004     | Long  | P2   |

# OUTER JOIN

- Nếu thực hiện câu truy vấn:  
SELECT PhongBan.\*, NhanVien.\*  
FROM PhongBan INNER JOIN NhanVien ON  
PhongBan.MSPB=NhanVien.MSPB
- Ta được kết quả như sau:

| PhongBan.MSPB | TenPB      | MSNV | HoTen | NhanVien.MSPB |
|---------------|------------|------|-------|---------------|
| P1            | Hanh Chanh | 0001 | An    | P1            |
| P1            | Hanh Chanh | 0002 | Minh  | P1            |
| P2            | Ke Toan    | 0003 | Tuan  | P2            |
| P2            | Ke Toan    | 0004 | Long  | P2            |

# OUTER JOIN

- Nếu ta đổi lại câu lệnh trên thành:

SELECT PhongBan.\*, NhanVien.\*

FROM PhongBan **LEFT JOIN** NhanVien ON  
PhongBan.MSPB=NhanVien.MSPB

- Thì kết quả sẽ như sau:

| PhongBan.MSPB | TenPB             | MSNV | HoTen | NhanVien.MSPB |
|---------------|-------------------|------|-------|---------------|
| P1            | Hanh Chanh        | 0001 | An    | P1            |
| P1            | Hanh Chanh        | 0002 | Minh  | P1            |
| P2            | Ke Toan           | 0003 | Tuan  | P2            |
| P2            | Ke Toan           | 0004 | Long  | P2            |
| <b>P3</b>     | <b>Kinh Doanh</b> |      |       |               |

# OUTER JOIN

- Ta có thể áp dụng tính chất này để thực hiện câu truy vấn “Liệt kê các phòng hiện không có danh sách nhân viên”

```
SELECT PhongBan.*  
FROM PhongBan LEFT JOIN NhanVien  
      ON PhongBan.MSPB=NhanVien.MSPB  
WHERE MSNV IS NULL
```

- Việc thực hiện hoàn toàn tương tự cho **RIGHT JOIN**

# SELF JOIN

- Trong một số trường hợp, một table có thể được sử dụng hai hay nhiều lần trong cùng một câu truy vấn. Bản thân table này lại liên kết với chính nó, ta gọi phép kết này là tự liên kết.
- Ví dụ: Giả sử ta có table NhanVien

| MSNV | HoTen | MSNQL |
|------|-------|-------|
| 0001 | An    |       |
| 0002 | Binh  | 0001  |
| 0003 | Tuan  | 0001  |
| 0004 | Long  | 0002  |



# SELF JOIN

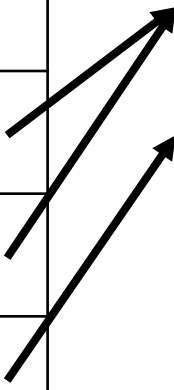
- Trong đó MSNQL là “Mã số người quản lý”. Như vậy nhân viên ‘0002’ và ‘0003’ là do nhân viên ‘0001’ quản lý. Nhân viên ‘0004’ là do nhân viên ‘0002’ quản lý.
- Ta muốn tạo câu truy vấn để lấy ra kết quả

| MSNV | HoTen | MSNQL | HoTenNQL |
|------|-------|-------|----------|
| 0001 | An    |       |          |
| 0002 | Binh  | 0001  | An       |
| 0003 | Tuan  | 0001  | An       |
| 0004 | Long  | 0002  | Binh     |

# SELF JOIN

- Trong trường hợp này ta cần sử dụng table NhanVien 2 lần trong câu truy vấn:

| Alias: <b>nv</b> |       |       | Alias: <b>ql</b> |       |       |
|------------------|-------|-------|------------------|-------|-------|
| MSNV             | HoTen | MSNQL | MSNV             | HoTen | MSNQL |
| 0001             | An    |       | 0001             | An    |       |
| 0002             | Binh  | 0001  | 0002             | Binh  | 0001  |
| 0003             | Tuan  | 0001  | 0003             | Tuan  | 0001  |
| 0004             | Long  | 0002  | 0004             | Long  | 0002  |



# SELF JOIN

- Câu truy vấn thực hiện như sau:

```
SELECT nv.MSNV, nv.HoTen,  
       nv.MSNQL, ql.HoTen  
FROM   NhanVien AS nv LEFT JOIN NhanVien AS ql  
       ON nv.MSNQL = ql.MSNV
```



# UNION QUERY

- Ta có thể lấy dữ liệu từ nhiều câu truy vấn và nối lại kết quả trả về của các câu truy vấn này bằng mệnh đề UNION

```
SELECT MSNV, Ho, Ten  
FROM NhanVienCN1  
UNION  
SELECT MSNV, Ho, Ten  
FROM NhanVienCN2
```

BK  
TP.HCM

# TÓM TẮT

- Giới thiệu về query
- Thiết kế select query bằng lưới QBE
- Thiết kế các loại query khác:
  - Crosstab query
  - Action query
  - Union query
- Giới thiệu ngôn ngữ SQL

