CÂU LỆNH TRONG MYSQL - DinhNT

--- NỘI DUNG SLIDE 4 - PHẦN 1 (Ngôn ngữ định nghĩa dữ liệu DDL) ---

```
--Tạo database
CREATE DATABASE <tên_database>;
```

ví dụ:

```
CREATE DATABASE qlnhanvien;
```

```
--Tạo databse có charset
CREATE DATABASE <tên_database> CHARACTER SET <...> COLLATE <...>;
```

ví dụ:

```
CREATE DATABASE mydatabase CHARACTER SET utf8 COLLATE utf8_unicode_ci;
```

```
--Tạo bảng
CREATE TABLE  <tên_bảng> (
    <thuộc_tính_1>  <kiểu_dữ_liệu[miền_giá_trị]>,
    <thuộc_tính_2>  <kiểu_dữ_liệu[miền_giá_trị]>,
    ...
);
```

ví dụ:

```
CREATE TABLE NHAN_VIEN (
    ID_NHANVIEN  INT            NOT NULL,
    HO_NV        VARCHAR(20)    NULL,
    TEN_NV       VARCHAR(25)    NOT NULL,
    NGAY_SINH    DATE,
    LUONG        INT            NULL,
    PHG          CHAR(5)        NULL
);
```

```
--Tạo bảng có khóa chính (PK) và khóa ngoại (FK)
CREATE TABLE  <tên_bảng> (
    <thuộc_tính_1>  <kiểu_dữ_liệu[miền_giá_trị]>,
    <thuộc_tính_2>  <kiểu_dữ_liệu[miền_giá_trị]>,
    ...,
    PRIMARY KEY (<thuộc_tính_làm_khóa_chính>),
    FOREIGN KEY (<thuộc_tính_làm_khóa_ngoại>)
    REFERENCES <tên_bảng_liên_kết_khóa_ngoại> (<thuộc_tính>)
);
```

ví dụ:

```
CREATE TABLE NHAN_VIEN (
    ID_NHANVIEN  INT            NOT NULL,
    HO_NV        VARCHAR(20)    NULL,
    TEN_NV       VARCHAR(25)    NOT NULL,
    NGAY_SINH    DATE,
    LUONG        INT            NULL,
    PHG          CHAR(5)        NULL,
    PRIMARY KEY (ID_NHANVIEN),
    FOREIGN KEY (PHG) REFERENCES PHONG_BAN(MA_PHONG)
);
```

--- NỘI DUNG SLIDE 4 - PHẦN 2 (Các câu lệnh thay đổi và xoá bảng) ---

```
--Thêm MỘT cột vào bảng có sẵn
ALTER TABLE <tên_bảng>
ADD COLUMN <tên_thuộc_tính> <kiểu_dữ_liệu[miền_giá_trị]>;
```

ví dụ:

```
ALTER TABLE NHAN_VIEN
ADD COLUMN EMAIL VARCHAR(20);
```

```
--(*)Thêm NHIỀU cột vào bảng có sẵn
ALTER TABLE <tên_bảng>
ADD COLUMN <tên_thuộc_tính_1> <kiểu_dữ_liệu[miền_giá_trị]>,
ADD COLUMN <tên_thuộc_tính_2> <kiểu_dữ_liệu[miền_giá_trị]>,
...;
```

ví dụ:

```
ALTER TABLE NHAN_VIEN
ADD COLUMN EMAIL VARCHAR(20),
ADD COLUMN SDT VARCHAR(20);
```

```
--(*)Chỉnh sửa thông tin MỘT cột trong bảng có sẵn
ALTER TABLE <tên_bảng>
MODIFY COLUMN <tên_thuộc_tính> <kiểu_dữ_liệu[miền_giá_trị]>;
```

ví dụ:

```
ALTER TABLE PHONG_BAN
MODIFY COLUMN tenPB varchar(50);
```

```
--(*)Chỉnh sửa thông tin NHIỀU cột trong bảng có sẵn
ALTER TABLE <tên_bảng>
MODIFY COLUMN <tên_thuộc_tính_1> <kiểu_dữ_liệu[miền_giá_trị]>,
MODIFY COLUMN <tên_thuộc_tính_2> <kiểu_dữ_liệu[miền_giá_trị]>,
...;
```

ví dụ:

```
ALTER TABLE PHONG_BAN
MODIFY COLUMN email varchar(100),
MODIFY COLUMN tenPB varchar(50);
```

```
--Xóa MỘT cột trong bảng có sẵn
ALTER TABLE <tên_bảng>
DROP COLUMN <tên_cột_cần_xóa>;
```

ví dụ:

```
ALTER TABLE PHONG_BAN
DROP COLUMN EMAIL;
```

```
--(*)Xóa NHIỀU cột trong bảng có sẵn
ALTER TABLE <tên_bảng>
DROP COLUMN <tên_cột_cần_xóa_1>,
DROP COLUMN <tên_cột_cần_xóa_2>,
...;
```

ví dụ:

```
ALTER TABLE PHONG_BAN
DROP COLUMN EMAIL,
DROP COLUMN DIACHI;
```

```
--Thêm ràng buộc KIỂM TRA vào bảng
ALTER TABLE <tên_bảng>
ADD CONSTRAINT <tên_ràng_buộc>
CHECK (<điều_kiện>);
```

ví dụ:

```
ALTER TABLE NHAN_VIEN
ADD CONSTRAINT CHK_SALARY_MIN
CHECK (LUONG >= 100);
```

(*) ***Ràng buộc trong khoảng:***

```
ALTER TABLE NHAN_VIEN
ADD CONSTRAINT CHK_SALARY_MIN
CHECK (contact_id BETWEEN 100 AND 200);
```

```
--LOẠI BỎ ràng buộc
ALTER TABLE <tên_bảng>
DROP CONSTRAINT <tên_ràng_buộc_cần_xóa>;
```

ví dụ:

```
ALTER TABLE NHAN_VIEN
DROP CONSTRAINT CHK_SALARY_MIN;
```

```
--Thêm ràng buộc KHÓA CHÍNH vào bảng
ALTER TABLE <tên_bảng>
ADD CONSTRAINT <tên_ràng_buộc>
PRIMARY KEY (<thuộc_tính_khóa_chính>,[<thuộc_tính_khóa_chính_2>]);
```

ví dụ:

```
ALTER TABLE PHONG_BAN
ADD CONSTRAINT PRI_PHONGBAN
PRIMARY KEY (MaPB);

ALTER TABLE CTHD
ADD CONSTRAINT PRI_CTHD
PRIMARY KEY (MaHD, MaSP);
```

```
--LOẠI BỎ ràng buộc KHÓA CHÍNH trong bảng
ALTER TABLE <tên_bảng_cần_xóa_khóa_chính>
DROP PRIMARY KEY;
```

ví dụ:

```
ALTER TABLE NHAN_VIEN
DROP PRIMARY KEY;
```

```
--Thêm ràng buộc KHÓA NGOẠI vào bảng
ALTER TABLE <tên_bảng>
ADD CONSTRAINT <tên_ràng_buộc>
FOREIGN KEY (<thuộc_tính_làm_khóa_ngoại>)
REFERENCES <tên_bảng_liên_kết>(<khóa_chính_liên_kết_với_khóa_ngoại>);
```

ví dụ:

```
ALTER TABLE NHAN_VIEN
ADD CONSTRAINT FK_PHONGBAN_NHANVIEN
FOREIGN KEY (PHG)
REFERENCES PHONG_BAN(MAPB);
```

```
--LOẠI BỎ ràng buộc KHÓA NGOẠI trong bảng
ALTER TABLE <tên_bảng_cần_xóa_khóa_chính>
DROP FOREIGN KEY <tên_ràng_buộc_cần_xóa>;
và
ALTER TABLE <tên_bảng_cần_xóa_khóa_chính>
DROP INDEX <tên_ràng_buộc_cần_xóa>;
```

ví dụ:

```
ALTER TABLE NHAN_VIEN
DROP FOREIGN KEY FK_PHONGBAN_NHANVIEN;
và
ALTER TABLE NHAN_VIEN
DROP INDEX FK_PHONGBAN_NHANVIEN;
```

```
--Thêm ràng buộc UNIQUE
ALTER TABLE <tên_bảng>
ADD CONSTRAINT <tên_ràng_buộc>
UNIQUE (<tên_cột_cần_ràng_buộc>);
```

ví dụ:

```
ALTER TABLE NHAN_VIEN
ADD CONSTRAINT NHANVIEN_UNQ_EMAIL
UNIQUE (EMAIL);
```

```
--LOẠI BỎ ràng buộc UNIQUE
ALTER TABLE <tên_bảng>
DROP CONSTRAINT <tên_ràng_buộc_cần_xóa>;
```

ví dụ:

```
ALTER TABLE NHAN_VIEN
DROP CONSTRAINT NHANVIEN_UNQ_EMAIL;
```

```
--(*) Thay đổi tên bảng
RENAME TABLE `<tên_bảng_cần_thay_đổi>` TO  `<tên_bảng_mới>`;

    ví dụ:
    RENAME TABLE `PHOGNN_BAN` TO `PHONG_BAN`;

--(*) Thay đổi thông tin của MỘT cột trong bảng
ALTER TABLE <tên_bảng>
CHANGE COLUMN `<tên_cột_cần_thay_đổi>` `<tên_mới>` <kiểu_dữ_liệu>;

    ví dụ:
    ALTER TABLE NHANVIEN
    CHANGE COLUMN `CMND` `CCCD` int;

--Xóa bảng
DROP TABLE <tên_bảng>;

    ví dụ:
    DROP TABLE NHAN_VIEN;

--Xóa bảng và tự động loại bỏ các ràng buộc tham chiếu trong bảng
(chạy từng câu lệnh bên dưới)
SET GLOBAL FOREIGN_KEY_CHECKS = 0;
DROP TABLE <tên_bảng>;
SET GLOBAL FOREIGN_KEY_CHECKS = 1;

    ví dụ:
    SET GLOBAL FOREIGN_KEY_CHECKS = 0;
    DROP TABLE NHAN_VIEN;
    SET GLOBAL FOREIGN_KEY_CHECKS = 1;

--Xóa database
DROP DATABASE <tên_database_cần_xóa>;

    ví dụ:
    DROP DATABASE QLNHANVIEN;
```

--- NỘI DUNG SLIDE 5 - PHẦN 1 (Truy vấn dữ liệu trên MỘT bảng) ---

Cú pháp:

```
SELECT [DISTINCT] Column(s)
FROM <tên_bảng>, <views>
[WHERE  Conditions]
[ORDER BY Column(s) [asc|desc]]
[GROUP BY Row(s)]
```

(Các mệnh đề trong cặp dấu [] không bắt buộc)

Trong đó:

DISTINCT trả về các bản ghi **không trùng lặp nhau**

WHERE cho phép truy vấn lựa chọn **theo hàng**

ORDER BY cho phép sắp xếp dữ liệu **theo cột**

GROUP BY cho phép nhóm dữ liệu **theo hàng**

```
--Truy vấn lựa chọn tất cả các hàng và cột
SELECT * FROM <tên_bảng>;
```

ví dụ:

```
SELECT * FROM NHAN_VIEN;
```

```
--Truy vấn lựa chọn một số cột
SELECT <tên_cột_1>, <tên_cột_2>, ... FROM <tên_bảng>;
```

ví dụ:

```
SELECT MaNV, HoTenNV, LUONG FROM NHAN_VIEN;
```

```
--Thay đổi tên cột hiển thị trong tập kết quả trả về
SELECT <tên_cột_1> AS `<tên_cần_đổi_cột_1>`,
       <tên_cột_2> AS `<tên_cần_đổi_cột_2>`,
       ...
FROM <tên_bảng>;
```

ví dụ:

```
SELECT MaNV AS `Mã Nhân Viên`,
       HoTenNV AS `Tên Nhân Viên`,
       LUONG AS `Lương`
FROM NHAN_VIEN;
```

```
--Mệnh đề CONCAT - Nối nội dung của những cột với nhau
SELECT CONCAT(<tên_cột_1>,<tên_cột_2>,<tên_cột_3>,...)
FROM <tên_bảng>;
hoặc
SELECT <tên_cột_1> + <tên_cột_2> + <tên_cột_3> + ...
FROM <tên_bảng>;
```

ví dụ:

```
SELECT CONCAT(HoNV,' ',TenNV)
FROM NHAN_VIEN;
hoặc
SELECT HoNV + ' ' + TenNV
FROM NHAN_VIEN;
```

```
--Mệnh đề DISTINCT - Loại bỏ các hàng trùng nhau trong tập kết quả
SELECT DISTINCT <tên_cột>
FROM <tên_bảng>;
```

ví dụ:

```
SELECT DISTINCT HoNV
FROM NHAN_VIEN;
```

```
--Mệnh đề LIMIT - dùng để hiển thị N hàng đầu tiên trong bảng
SELECT * FROM <tên_bảng> LIMIT <N>;
hoặc
SELECT <tên_cột_1>, <tên_cột_2>, … FROM <tên_bảng> LIMIT <N>;
```

Trong đó **<N>** là số hàng đầu tiên cần hiển thị

ví dụ:

```
SELECT * FROM NHAN_VIEN LIMIT 5;
hoặc
SELECT MaNV, HoTenNV, LUONG FROM NHAN_VIEN LIMIT 5;
```

--Mệnh đề **WHERE** - Câu điều kiện

Cú pháp:

```
SELECT [DISTINCT] Column(s)
FROM <tên_bảng>
[WHERE  <điều_kiện>]
```

Một số toán tử (Operator) sử dụng trong biểu thức Conditions:

- Toán tử so sánh : **>** , **<** , **>=** , **<=** , **<>**
- Toán tử logic : **AND**, **OR**, **NOT**
- So sánh chuỗi dùng toán tử **LIKE**
- **BETWEEN** … **AND** , **IN**

So sánh dùng toán tử **LIKE**

Kí tự đại diện	Mô tả	Ví dụ
_	Đại diện cho 1 kí tự	SELECT * FROM NHAN_VIEN WHERE HO_NV LIKE 'H_'
%	Đại diện cho một chuỗi kí tự có độ dài bất kì	SELECT * FROM NHAN_VIEN WHERE TEN_NV LIKE 'B%'

```
--Làm việc với Ngày/Tháng/Năm (Date)
Hiển thị tất cả các thông tin dự án có ngày bắt đầu từ ngày 01/01/2017
        SELECT *
        FROM du_an
        WHERE DATE(ngay_batdau) >= '2017-01-01'


--Lấy ngày, tháng, năm trong cột
```

- DAY(<tên_cột_>) : lấy ngày
- MONTH(<tên_cột_>) : lấy tháng
- YEAR(<tên_cột_>) : lấy năm
- CURDATE() hoặc CURRENT_DATE() để lấy ngày hiện tại

ví dụ:

```
Hiển thị danh sách các hoá hơn có ngày mua hàng trong năm 2016
    SELECT *
    FROM HOA_DON
    WHERE YEAR(ngay_mua) = '2016'

Hiển thị danh sách các hoá hơn có ngày mua bằng với ngày hiện tại
    SELECT *
    FROM HOA_DON
    WHERE CURRENT_DATE() = ngay_mua
```

--- NỘI DUNG SLIDE 5 - PHẦN 2 (Truy vấn dữ liệu trên MỘT bảng) ---

1. **MAX** - Tìm giá trị **LỚN** nhất trong cột

    ví dụ:
    Hiển thị mức lương CAO NHẤT trong bảng nhân viên
    ```
    SELECT MAX(Luong)
    FROM NHAN_VIEN
    ```

2. **MIN** - Tìm giá trị **NHỎ** nhất trong cột

    ví dụ:
    Hiển thị mức lương THẤP NHẤT trong bảng nhân viên
    ```
    SELECT MIN(Luong)
    FROM NHAN_VIEN
    ```

3. **AVG** - Tìm giá trị trung bình của cột

    ví dụ:
    Hiển thị mức lương TRUNG BÌNH của nhân viên trong công ty
    ```
    SELECT AVG(Luong)
    FROM NHAN_VIEN
    ```

4. **SUM** - Tính tổng giá trị của cột

    ví dụ:
    Hiển thị tổng tất cả số lượng sản phẩm có trong bảng sản phẩm
    ```
    SELECT SUM(SoLuong)
    FROM SAN_PHAM
    ```

5. **COUNT** - Đếm số lượng giá trị trong cột

    ví dụ:
    Đếm số lượng nhân viên trong công ty
    ```
    SELECT COUNT(*)
    FROM NHAN_VIEN
    ```
hoặc
    ```
    SELECT COUNT(MaNV)
    FROM NHAN_VIEN
    ```

6. (*) **ROUND** - Làm tròn số

    ví dụ:
    Tính tổng lương của tất cả nhân viên trong mỗi phòng ban, **làm tròn 2 số thập phân**
    ```
    SELECT PHG, ROUND(SUM(Luong),2)
    FROM NhanVien
    GROUP BY PHG
    ```

--Mệnh đề **GROUP BY**

Mệnh đề **GROUP BY** cho phép nhóm các hàng dữ liệu có giá trị giống nhau thành một nhóm
Các tính toán (thường sử dụng các hàm tổng hợp) sẽ được tính trên mỗi nhóm.

ví dụ:

```
Đếm số lượng nhân viên trong mỗi phòng
SELECT COUNT(*) AS 'Số lượng',  PHG AS 'Mã PB'
FROM NHAN_VIEN
GROUP BY PHG

Tính tổng lương của tất cả nhân viên trong mỗi phòng ban
SELECT PHG, SUM(Luong)
FROM NhanVien
GROUP BY PHG
```

--Mệnh đề **HAVING**

Mệnh đề **HAVING** đi kèm với **GROUP BY** giúp **loại bỏ các nhóm không thoã mãn điều kiện**

ví dụ:

```
Hiển thị mức lương cao nhất của mỗi phòng ban. Chỉ hiển thị mức lương lớn hơn 1000$.
SELECT PHG AS 'Mã PB', MAX(LUONG) AS 'Lương cao nhất' FROM NHAN_VIEN
GROUP BY PHG
HAVING MAX(LUONG) > 1000

Hiển thị mã hoá đơn và số sản phẩm được mua trong từng hoá đơn. Yêu cầu chỉ hiển thị
hàng nào có số loại sản phẩm được mua >=5
SELECT MaHD, COUNT(MaSP) FROM ChiTietHoaDon
GROUP BY MaHD
HAVING COUNT(MaSP) >= 5
```

--Mệnh đề **ORDER BY** - Cho phép sắp xếp kết quả truy vấn theo cột

Có thể sắp xếp kết quả theo chiều:Tăng dần (**ASC**) Giảm dần (**DESC**)

ví dụ:

```
Hiển thị danh sách nhân viên theo thứ tự TĂNG DẦN của mã nhân viên
SELECT * FROM NHAN_VIEN
ORDER BY ID_NhanVien
```

hoặc

```
SELECT * FROM NHAN_VIEN
ORDER BY ID_NhanVien ASC

Hiển thị danh sách nhân viên theo thứ tự GIẢM DẦN của mã nhân viên
SELECT * FROM NHAN_VIEN
ORDER BY ID_NhanVien DESC
```

--- NỘI DUNG SLIDE 6 - PHẦN 1 (Truy vấn dữ liệu trên NHIỀU bảng) ---

--Sử dụng **PHÉP TÍCH**

Sử dụng điều kiện **kết bằng** trong mệnh đề WHERE

Nếu xuất hiện tên cột trùng nhau trong nhiều bảng thì bắt buộc phải sử dụng **tên bảng** hoặc **bí danh** bảng trước tên cột

Cú pháp:

```
SELECT <tên_bảng_1>.<tên_thuộc_tính>, <tên_bảng_2>.<tên_thuộc_tính>,...
FROM <tên_bảng_1>, <tên_bảng_2>
WHERE <tên_bảng_1>.<tên_thuộc_tính> = <tên_bảng_2>.<tên_thuộc_tính>
```

ví dụ:

Hiển thị dữ liệu của 2 bảng **Nhân viên** và **Phòng ban** bao gồm: *mã nhân viên, tên nhân viên, lương* và *tên phòng ban đang làm việc*

```
SELECT NHAN_VIEN.ID_NhanVien, NHAN_VIEN.Ten_NV, NHAN_VIEN.Luong, PHONG_BAN.Ten_PB
FROM NHAN_VIEN, PHONG_BAN
WHERE NHAN_VIEN.PHG = PHONG_BAN.MA_PB;
```

Hiển thị dữ liệu trong 3 bảng: **NHAN_VIEN, DU_AN** và **QL_DUAN** bao gồm: *tên dự án, tên nhân viên, ngày tham gia dự án* và *ngày kết thúc dự án*

```
SELECT DU_AN.Ten_DuAn, NHAN_VIEN.Ten_NV, QL_DUAN.Ngay_Tham_Gia, QL_DUAN.Ngay_Ket_Thuc
FROM NHAN_VIEN, DU_AN, QL_DUAN
WHERE DU_AN.MA_DUAN = QL_DUAN.MA_DUAN AND NHAN_VIEN.ID_NhanVien = QL_DUAN.MA_NHANVIEN;
```

--Dùng **BÍ DANH** cho tên bảng

Đơn giản hóa các câu truy vấn khi cần sử dụng tên bảng cho việc truy xuất các cột

ví dụ:

Hiển thị dữ liệu của 2 bảng **Nhân viên** và **Phòng ban** bao gồm: *mã nhân viên, tên nhân viên, lương* và *tên phòng ban đang làm việc*

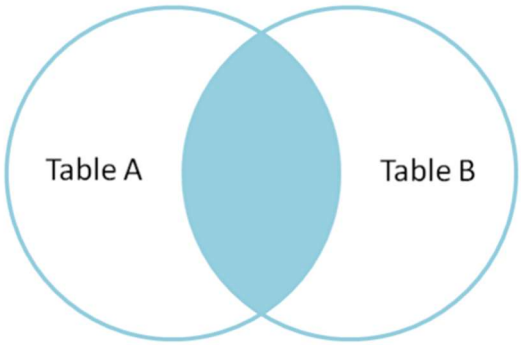
```
SELECT nv.ID_NhanVien, nv.Ten_NV, nv.Luong, pb.Ten_PB
FROM NHAN_VIEN nv, PHONG_BAN pb
WHERE nv.PHG = pb.MA_PB;
```

Hiển thị dữ liệu trong 3 bảng: **NHAN_VIEN, DU_AN** và **QL_DUAN** bao gồm: *tên dự án, tên nhân viên, ngày tham gia dự án* và *ngày kết thúc dự án*

```
SELECT da.Ten_DuAn, nv.Ten_NV, qlda.Ngay_Tham_Gia, qlda.Ngay_Ket_Thuc
FROM NHAN_VIEN nv, DU_AN da, QL_DUAN qlda
WHERE da.MA_DUAN = qlda.MA_DUAN AND nv.ID_NhanVien = qlda.MA_NHANVIEN;
```

--INNER JOIN

INNER JOIN trả về kết quả là các bản ghi mà trường được join ở **hai bảng khớp nhau**, các bản ghi **chỉ xuất hiện ở một trong hai bảng sẽ bị loại**



Cú pháp:

```
SELECT <thuộc_tính_1>, <thuộc_tính_2>,...
FROM <tên_bảng_1> [INNER] JOIN <tên_bảng_2> ON <điều_kiện_join_1>
                  [[INNER] JOIN <tên_bảng_3> ON <điều_kiện_join_2>]
                  ...
```

ví dụ:

Sử dụng câu lệnh JOIN để hiển thị dữ liệu của 2 bảng **Nhân viên** và **Phòng ban**

```
SELECT ID_NhanVien, Ho_NV, Ten_NV, Ten_PB
FROM PHONG_BAN PB INNER JOIN NHAN_VIEN NV
ON PB.Ma_PB = NV.PHG
```

Sử dụng mệnh đề JOIN để hiển thị dữ liệu trong 3 bảng: **NHAN_VIEN, DU_AN** và **QUANLY_DUAN**

```
SELECT da.Ten_DaAn, nv.Ho_NV, nv.Ten_NV, qlda.Ngay_Tham_Gia, qlda.Ngay_Ket_Thuc
FROM NHAN_VIEN nv INNER JOIN QUANLY_DUAN qlda ON nv.ID_NhanVien = qlda.Ma_NhanVien
                  INNER JOIN DU_AN da ON da.Ma_DuAn = qlda.Ma_DuAn;
```

So sánh INNER JOIN với PHÉP TÍCH

- **Phép tích** sẽ nhân số lượng bản ghi 2 bảng, sau đó loại bỏ các bản ghi không thỏa mãn điều kiện

  Ví dụ: Bảng A có 3 bản ghi, bảng B có 4 bản ghi -> tích sẽ cho ra **12** bản ghi, sau đó sẽ loại bỏ các bản ghi không thoã mãn điều kiện: **A.MA_PB = B.PHG**
- Với **INNER JOIN** thì trong quá trình thực hiện tích 2 bảng nó sẽ kiểm tra điều kiện ở **ON** luôn, nếu đúng thì chọn, sai thì bỏ qua.
- Như vậy xét về tốc độ truy vấn thì trường hợp sử dụng **INNER JOIN** sẽ nhanh hơn rất nhiều so với sử dụng **phép tích**

ví dụ (NỘI DUNG SLIDE 6 - PHẦN 1 - PHÉP TÍCH VÀ INNER JOIN):

Sử dụng JOIN hoặc phép tích để hiển thị thông tin gồm: họ và tên nhân viên, lương, tên phòng ban mà nhân viên thuộc về, tên dự án, ngày bắt đầu tham gia vào dự án và số giờ làm của nhân viên trong dự án.

Phép tích:

```
SELECT Ho_NV, Ten_NV, Luong, Ten_PB, Ten_DuAn, Ngay_BatDau, So_Gio
FROM PHONG_BAN pb, NHAN_VIEN nv, DU_AN da, QUANLY_DUAN qlda
WHERE pb.Ma_PB = nv.PHG AND nv.ID_NhanVien = qlda.Ma_NhanVien
                        AND da.Ma_DuAn = qlda.Ma_DuAn
```

INNER JOIN:

```
SELECT Ho_NV, Ten_NV, Luong, Ten_PB, Ten_DuAn, Ngay_BatDau, So_Gio
FROM PHONG_BAN pb INNER JOIN NHAN_VIEN nv ON pb.Ma_PB = nv.PHG
                  INNER JOIN QUANLY_DUAN qlda ON nv.ID_NhanVien = qlda.Ma_NhanVien
                  INNER JOIN DU_AN da ON da.Ma_DuAn = qlda.Ma_DuAn
```

Viết câu truy vấn hiển thị các thông tin bao gồm họ, tên, lương của nhân viên, tên dự án với điều kiện nhân viên thuộc phòng ban có tên “Thiết kế”, tham gia vào các dự án có ngày bắt đầu 01/01/2016

Phép tích:

```
SELECT Ho_NV, Ten_NV, Ten_DuAn
FROM PHONG_BAN pb, NHAN_VIEN nv, DU_AN da, QUANLY_DUAN qlda
WHERE pb.Ma_PB = nv.PHG AND nv.ID_NhanVien = qlda.Ma_NhanVien
                        AND da.Ma_DuAn = qlda.Ma_DuAn
                        AND pb.Ten_PB LIKE 'Thiết kế'
                        AND DATE(da.Ngay_BatDau) >= '2016-01-01'
```

INNER JOIN:

```
SELECT Ho_NV, Ten_NV, Ten_DuAn
FROM PHONG_BAN pb INNER JOIN NHAN_VIEN nv ON pb.Ma_PB = nv.PHG
                  INNER JOIN QUANLY_DUAN qlda ON nv.ID_NhanVien = qlda.Ma_NhanVien
                  INNER JOIN DU_AN da ON da.Ma_DuAn = qlda.Ma_DuAn
WHERE pb.Ten_PB LIKE 'Thiết kế' AND DATE(da.Ngay_BatDau) >= '2016-01-01'
```

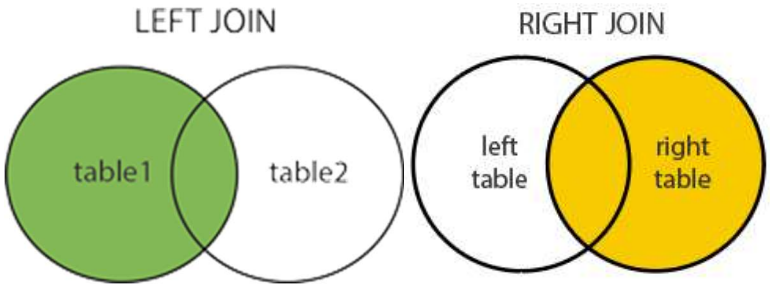
--- NỘI DUNG SLIDE 6 - PHẦN 2 (Truy vấn dữ liệu trên NHIỀU bảng) ---

MySQL không có **FULL OUTER JOIN

--LEFT/RIGHT JOIN

Bảng **A** **LEFT/RIGHT JOIN** với **bảng B** thì *kết quả gồm có bản ghi* trong **bảng A**, *với các bảng ghi không có mặt* trong **bảng B** thì **các cột từ B được ghi NULL**. Các bảng ghi *chỉ có trong bảng B mà không có trong bảng A* sẽ **không được trả về**

Sử dụng **LEFT JOIN** hay **RIGHT JOIN** phụ thuộc vào vị trí của bảng trong câu truy vấn



Cú pháp:

```
1. LEFT JOIN
SELECT <thuộc_tính_1>, <thuộc_tính_2>,...
FROM <tên_bảng_1> LEFT JOIN <tên_bảng_2> ON <điều_kiện_join_1>

2. RIGHT JOIN
SELECT <thuộc_tính_1>, <thuộc_tính_2>,...
FROM <tên_bảng_1> RIGHT JOIN <tên_bảng_2> ON <điều_kiện_join_1>
```

ví dụ:

Ta có dữ liệu 2 bảng **NHÂN VIÊN** và **PHÒNG BAN** như sau:

- Bảng **PHÒNG BAN**:

MA_PB	TEN_PB	MA_TRUONGPHONG
PB001	Phòng dự án Quận 9	1235
PB002	Phòng sản xuất sản phẩm	8561
PB003	Phòng dự án CNC	8793
PB004	Phòng dự án thiết kế Metro	*NULL*
PB005	Phòng thi công hạ tầng	*NULL*
PB006	Phòng quản lý chất lượng	*NULL*
PB007	Phòng đảm bảo chất lượng	*NULL*

- Bảng **NHÂN VIÊN**:

ID_NhanVien	HO_NV	TEN_NV	NAM_SINH	DIA_CHI	GIOI_TINH	LUONG	PHG
1235	Nguyễn Văn	Anh	2021-11-10 13:16:08.000	Bình Thuận	1	550.0000	PB001
3365	Trần Minh	Tú	2021-11-10 13:16:08.000	Phú Yên	1	790.0000	PB005
4568	Trần Hùng	Mạnh	2021-11-10 13:16:08.000	Bạc Liêu	1	670.0000	PB004
5654	Trần	Cường	2021-11-10 13:16:08.000	HCM	1	1400.0000	PB002
6324	Hồ	Minh	2021-11-10 13:16:08.000	HCM	1	495.0000	PB001
6378	Huỳnh Anh	Tá	1996-11-11 13:16:08.000	Nam Định	1	775.0000	PB004
6532	Lê Bé	Na	2021-11-10 13:16:08.000	Phú Yên	0	1400.0000	PB006
8561	Trần Văn	Bình	2021-11-10 13:16:08.000	HCM	1	1350.0000	PB002
8765	Cao Mỹ	Lệ	2021-11-10 13:16:08.000	Bạc Liêu	0	100.0000	PB004
8793	Huỳnh	Tuấn	2021-11-10 13:16:08.000	Phú Yên	1	875.0000	PB003

Khi sử dụng **RIGHT JOIN** ta được kết quả như sau:

Truy vấn:

Viết câu truy vấn hiển thị các thông tin bao gồm mã nhân viên, họ tên nhân viên, và tên phòng ban mà nhân viên trực thuộc. Nếu nhân viên chưa được phân bổ vào phòng nào thì cột tên phòng để trống

```
SELECT ID_NhanVien, Ho_NV, Ten_NV, Ten_PB
FROM PHONG_BAN PB RIGHT JOIN NHAN_VIEN NV
ON PB.Ma_PB = NV.PHG
```

Kết quả: Trên câu truy vấn ta thấy bảng **PHÒNG BAN** nằm bên trái và bảng **NHÂN VIÊN** nằm bên phải so với **JOIN**, nên khi sử dụng **RIGHT JOIN** ta phải lấy *toàn bộ dữ liệu* từ bảng **NHÂN VIÊN** (vì đang sử dụng **RIGHT JOIN** và bảng **NHÂN VIÊN** đang nằm bên phải). Khi đó những nhân viên không thuộc bất kỳ một phòng ban nào thì cột Ten_PB sẽ để NULL còn những nhân viên thuộc một phòng ban nào đó thì cột TenPB sẽ hiển thị ra tên phòng ban tương ứng.

ID_NhanVien	ho_nv	Ten_NV	Ten_PB
1235	Nguyễn Văn	Anh	Phòng dự án Quận 9
3365	Trần Minh	Tú	Phòng thi công hạ tầng
4568	Trần Hùng	Mạnh	Phòng dự án thiết kế Metro
5654	Trần	Cường	Phòng sản xuất sản phẩm
6324	Hồ	Minh	Phòng dự án Quận 9
6378	Huỳnh Anh	Tá	Phòng dự án thiết kế Metro
6532	Lê Bé	Na	Phòng quản lý chất lượng
8561	Trần Văn	Bình	Phòng sản xuất sản phẩm
8765	Cao Mỹ	Lệ	Phòng dự án thiết kế Metro
8793	Huỳnh	Tuấn	Phòng dự án CNC

Khi sử dụng **LEFT JOIN** ta được kết quả như sau:

Truy vấn:

Viết câu truy vấn hiển thị các thông tin bao gồm mã nhân viên, họ tên nhân viên và tên phòng ban mà nhân viên trực thuộc. Nếu phòng ban nào chưa có nhân phân bổ vào phòng thì cột mã nhân viên, họ tên nhân viên và tên phòng ban để trống

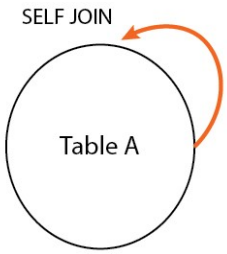
```
SELECT ID_NhanVien, Ho_NV, Ten_NV, Ten_PB
FROM PHONG_BAN PB LEFT JOIN NHAN_VIEN NV
ON PB.Ma_PB = NV.PHG
```

Kết quả: Trên câu truy vấn ta thấy bảng **PHÒNG BAN** nằm bên trái và bảng **NHÂN VIÊN** nằm bên phải so với **JOIN**, nên khi sử dụng **LEFT JOIN** ta phải lấy ***toàn bộ dữ liệu*** từ bảng **PHÒNG BAN** (vì đang sử dụng **LEFT JOIN** và bảng **PHÒNG BAN** đang nằm bên trái). Khi đó những phòng ban không có bất kỳ nhân viên nào thì các cột ID_NhanVien, Ho_NV, Ten_NV sẽ để NULL còn những phòng ban có nhân viên làm việc trong đó thì các cột ID_NhanVien, Ho_NV, Ten_NV sẽ hiển thị ra thông tin nhân viên tương ứng làm việc trong phòng ban đó.

ID_NhanVien	ho_nv	Ten_NV	Ten_PB
1235	Nguyễn Văn	Anh	Phòng dự án Quận 9
6324	Hồ	Minh	Phòng dự án Quận 9
5654	Trần	Cường	Phòng sản xuất sản phẩm
8561	Trần Văn	Bình	Phòng sản xuất sản phẩm
8793	Huỳnh	Tuấn	Phòng dự án CNC
4568	Trần Hùng	Mạnh	Phòng dự án thiết kế Metro
6378	Huỳnh Anh	Tá	Phòng dự án thiết kế Metro
8765	Cao Mỹ	Lệ	Phòng dự án thiết kế Metro
3365	Trần Minh	Tú	Phòng thi công hạ tầng
6532	Lê Bé	Na	Phòng quản lý chất lượng
NULL	*NULL*	*NULL*	Phòng đảm bảo chất lượng

--SELF JOIN

Một bảng kết nối với chính nó



ví dụ:

Ta có bảng dữ liệu sau:

MA_NV	TEN_NV	MA_QUANLY
1	Nga	
2	An	1
3	Van	2
4	Hoang	2

```
Hiển thị tên nhân viên và tên người Quản Lý của nhân viên đó
SELECT nv1.Ma_NV, nv1.Ten_NV, nv2.Ten_NV
FROM NHAN_VIEN nv1 INNER JOIN NHAN_VIEN nv2
ON nv2.Ma_NV = nv1.Ma_QuanLy;
```

--TRUY VẤN CON

- Là câu truy vấn **SELECT** nằm lồng bên trong một câu truy vấn khác
- Câu truy vấn con có thể được sử dụng:
  - Trong mệnh đề **WHERE** như một điều kiện tìm kiếm
  - Trong mệnh đề **HAVING** như một điều kiện tìm kiếm
  - Trong mệnh đề **FROM** như một đặc tả bảng
  - Trong mệnh đề **SELECT** như một đặc tả cột

ví dụ:

```
Hiển thị thông tin nhân viên có mức lương lớn hơn mức lương trung bình trong công ty
SELECT * FROM NHAN_VIEN
WHERE LUONG > (SELECT AVG(LUONG) FROM NHAN_VIEN)
```

SO SÁNH JOIN VÀ CÂU TRUY VẤN CON

JOIN	CÂU 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

--IN / NOT IN

ví dụ: MỆNH ĐỀ IN

Hiển thị toàn bộ thông tin của những khách hàng có họ thuộc 1 trong 3 họ sau: *Smith, Anderson, Johnson*

```
SELECT *
FROM NHAN_VIEN
WHERE Ho_NV IN ('Smith','Anderson','Johnson');
```

Câu truy vấn trên tương đương với:

```
SELECT *
FROM NHAN_VIEN
WHERE Ho_NV LIKE 'Smith' OR Ho_NV LIKE 'Anderson' OR Ho_NV LIKE 'Johnson';
```

Hiển thị toàn bộ thông tin nhân viên thuộc phòng ban chứa cụm từ “Sản xuất”

```
SELECT *
FROM NHAN_VIEN
WHERE PHG IN (SELECT MA_PB FROM PHONG_BAN WHERE TEN_PB LIKE '%Sản xuất%')
```

ví dụ: MỆNH ĐỀ NOT IN

Hiển thị toàn bộ thông tin của những khách hàng có họ **KHÔNG** thuộc những họ sau: *Sarah, Johnny, Dale*

```
SELECT *
FROM NHAN_VIEN
WHERE Ho_NV NOT IN ('Sarah','Johnny','Dale');
```

Câu truy vấn trên tương đương với:

```
SELECT *
FROM NHAN_VIEN
WHERE Ho_NV <> 'Sarah' AND Ho_NV <> 'Johnny' AND Ho_NV <> 'Dale';
```

--ANY

ĐIỀU KIỆN	MÔ TẢ	Ví dụ
X > ANY(truy_vấn_con)	X phải **lớn hơn** ít nhất 1 giá trị trả về từ câu truy vấn con	*Điều kiện*: **X > ANY(1,2,0)** *Kết quả*: **X > 0** **~> Lấy giá trị nhỏ nhất**
X < ANY(truy_vấn_con)	X phải **nhỏ hơn** ít nhất 1 giá trị trả về từ câu truy vấn con	*Điều kiện*: **X < ANY(1,2,-1)** *Kết quả*: **X < 2** **~> Lấy giá trị lớn nhất**
X = ANY(truy_vấn_con)	X IN (1,2,5)	*Điều kiện*: **X = ANY(1,2,5)** *Kết quả*: **X=1 or X=2 or X=5**

ví dụ:

```
SELECT *
FROM NHAN_VIEN
WHERE LUONG > ANY (SELECT LUONG
                   FROM NHAN_VIEN
                   WHERE PHG LIKE 'PB002')
```

--EXISTS / NOT EXISTS

ví dụ: MỆNH ĐỀ EXISTS - tồn tại

```
SELECT *
FROM NHAN_VIEN
WHERE EXISTS (SELECT *
              FROM PHONG_BAN
              WHERE NHAN_VIEN.PHG = PHONG_BAN.MA_PB)
```

ví dụ: MỆNH ĐỀ NOT EXISTS - không tồn tại

```
SELECT *
FROM NHAN_VIEN
WHERE NOT EXISTS (SELECT *
                  FROM PHONG_BAN
                  WHERE NHAN_VIEN.PHG = PHONG_BAN.MA_PB)
```

--- NỘI DUNG SLIDE 7 - PHẦN 1 (Ngôn ngữ thao tác dữ liệu) ---

```
--Sao chép TOÀN BỘ thông tin (thuộc tính, ràng buộc, dữ liệu,...) từ một bảng
CREATE TABLE <tên_bảng_mới> LIKE <tên_bảng_cần_sao_chép>;
INSERT INTO <tên_bảng_mới> SELECT * FROM <tên_bảng_cần_sao_chép>;
```

ví dụ:

```
CREATE TABLE NHAN_VIEN_CAO_CAP LIKE NHAN_VIEN;
INSERT INTO NHAN_VIEN_CAO_CAP SELECT * FROM NHAN_VIEN;
```

```
--Sao chép thông tin (chỉ sao chép thuộc tính và dữ liệu) từ một bảng
CREATE TABLE <tên_bảng_mới> AS SELECT * FROM <tên_bảng_cần_sao_chép>;
```

ví dụ:

```
CREATE TABLE NHAN_VIEN_CAO_CAP AS SELECT * FROM NHAN_VIEN;
```

```
--Thêm một hàng mới vào bảng KHÔNG LIỆT KÊ danh sách các cột (đầy đủ thông tin)
INSERT INTO <tên_bảng>
VALUES (<giá_trị_1>, <giá_trị_2>, <giá_trị_3>,...)
```

Lưu ý: Giá trị các cột truyền vào phải theo thứ tự các cột trong bảng

ví dụ:

```
INSERT INTO PHONG_BAN
VALUES ('PB007', 'Truyen Thong', null);
```

```
--Thêm một hàng mới vào bảng LIỆT KÊ danh sách các cột
INSERT INTO <tên_bảng>(<tên_cột_1>, <tên_cột_2>, <tên_cột_3>,...)
VALUES (<giá_trị_cột_1>, <giá_trị_cột_2>, <giá_trị_cột_3>,...)
```

ví dụ:

```
INSERT INTO PHONG_BAN(MA_PB, TEN_PB, MA_TP)
VALUES ('PB007', 'Truyen Thong', null);
```

--- NỘI DUNG SLIDE 7 - PHẦN 2 (Ngôn ngữ thao tác dữ liệu) ---

```
--Cập nhật dữ liệu ở trong bảng
UPDATE <tên_bảng>
SET <tên_cột_1> = <biểu_thức_1> [, <tên_cột_2> = <biểu_thức_2>] . . .
[WHERE  <điều_kiện>]
```

ví dụ:

Thay đổi tên phòng ban có mã phòng ban PB001 thành Phòng phát triển công nghệ

```
UPDATE PHONG_BAN
SET TEN_PB = 'Phòng phát triển công nghệ'
WHERE MA_PB = 'PB001'
```

Cập nhật lại lương tăng thêm 10% cho những nhân viên thuộc phòng ban Thiết kế

```
UPDATE NHAN_VIEN
SET LUONG = LUONG + LUONG * 0.1
WHERE PHG = (SELECT MA_PB FROM PHONG_BAN WHERE TEN_PB LIKE 'Thiết kế')
```

Cập nhật lại lương cho các nhân viên có tham gia vào dự án có mã DA001 lên 50$

```
UPDATE nhan_vien
SET LUONG = LUONG + 50
WHERE ID_NhanVien IN (SELECT ma_nhanvien
                      FROM quanly_duan
                      WHERE ma_duan LIKE 'DA001')
```

Cập nhật lại cột mã trưởng phòng cho Phòng ban có tên “Sản xuất” với giá trị mới là mã nhân viên có tên “Huỳnh Anh Tuấn”

```
UPDATE phong_ban
SET ma_truongphong = (SELECT id_nhanvien
                      FROM nhan_vien
                      WHERE ho_nv = 'Huỳnh' AND ten_nv = 'Anh Tuấn')
WHERE tenn_pb = 'Sản xuất'
```

```
--Xóa dữ liệu ở trong bảng
DELETE FROM <tên_bảng>
[WHERE <điều_kiện>]
```

ví dụ:

Xóa toàn bộ dữ liệu có trong bảng phòng ban

```
DELETE FROM PHONG_BAN
```

Xóa phòng ban có mã PB007

```
DELETE FROM PHONG_BAN
WHERE MA_PB LIKE 'PB007'
```

Xoá các Phòng ban không có nhân viên nào

```
DELETE FROM PHONG_BAN
WHERE MA_PB NOT IN (SELECT DISTINCT PHG FROM NHAN_VIEN)
```