JOIN

JOIN

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

1	**INNER JOIN**	Chỉ lấy các bản ghi có điểm chung giữa các bảng
2	**LEFT JOIN**	Lấy tất cả bản ghi từ bảng bên trái, nếu không có khớp từ bảng phải thì trả về NULL
3	**RIGHT JOIN**	Lấy tất cả bản ghi từ bảng bên phải, nếu không có khớp từ bảng trái thì trả về NULL
4	**FULL OUTER JOIN**	Lấy tất cả bản ghi từ cả hai bảng, nếu không có khớp thì trả về NULL
5	**CROSS JOIN**	Nhân chéo hai bảng (tạo tổ hợp tất cả các bản ghi)

Chuan bi

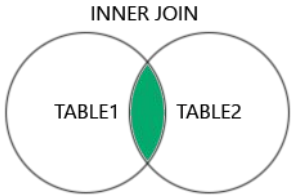
```sql
CREATE TABLE departments (
    id INT PRIMARY KEY AUTO_INCREMENT,
    department_name VARCHAR(100) NOT NULL
);

CREATE TABLE employees (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    salary DECIMAL(10,2) NOT NULL,
    department_id INT,
    FOREIGN KEY (department_id) REFERENCES departments(id)
);

INSERT INTO departments (department_name) VALUES
('Nhân sự'),
('Công nghệ thông tin'),
('Tài chính'),
('Marketing');

INSERT INTO employees (name, salary, department_id) VALUES
('Nguyễn Văn An', 12000000, 1),  -- Nhân sự
('Trần Thị Bình', 15000000, 2),  -- Công nghệ thông tin
('Lê Quốc Cường', 13000000, 2),  -- Công nghệ thông tin
('Phạm Hồng Đăng', 11000000, 3), -- Tài chính
('Hoàng Văn Nam', 14000000, NULL); -- Không có phòng ban
```

INNER JOIN



- **INNER JOIN** chọn các bản ghi có giá trị trùng khớp **trong cả hai bảng**
- VD: Danh sách các nhân viên và phòng ban của họ

#	id	name	salary	department_id
1	1	Nguyễn Văn An	12000000.00	1
2	2	Trần Thị Bình	15000000.00	2
3	3	Lê Quốc Cường	13000000.00	2
4	4	Phạm Hồng Đăng	11000000.00	3
5	5	Hoàng Văn Nam	14000000.00	NULL

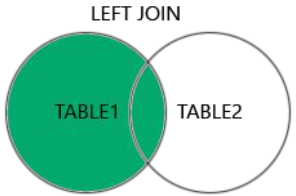
#	id	department_name
1	1	Nhân sự
2	2	Công nghệ thông tin
3	3	Tài chính
4	4	Marketing

INNER JOIN ⇒

#	name	salary	department_name
1	Nguyễn Văn An	12000000.00	Nhân sự
2	Trần Thị Bình	15000000.00	Công nghệ thông tin
3	Lê Quốc Cường	13000000.00	Công nghệ thông tin
4	Phạm Hồng Đăng	11000000.00	Tài chính

```
SELECT employees.name, employees.salary, departments.department_name
FROM employees
INNER JOIN departments
ON employees.department_id = departments.id;
```

LEFT JOIN



- LEFT JOIN trả về tất cả các bản ghi từ bảng bên trái (table1) và các bản ghi khớp (nếu có) từ bảng bên phải (table2)

#	id	name	salary	department_id
1	1	Nguyễn Văn An	12000000.00	1
2	2	Trần Thị Bình	15000000.00	2
3	3	Lê Quốc Cường	13000000.00	2
4	4	Phạm Hồng Đăng	11000000.00	3
5	5	Hoàng Văn Nam	14000000.00	NULL

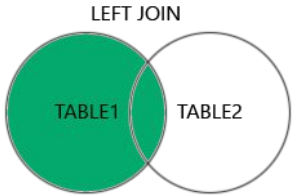
#	id	department_name
1	1	Nhân sự
2	2	Công nghệ thông tin
3	3	Tài chính
4	4	Marketing

LEFT JOIN ⇒

#	name	salary	department_name
1	Nguyễn Văn An	12000000.00	Nhân sự
2	Trần Thị Bình	15000000.00	Công nghệ thông tin
3	Lê Quốc Cường	13000000.00	Công nghệ thông tin
4	Phạm Hồng Đăng	11000000.00	Tài chính
5	Hoàng Văn Nam	14000000.00	NULL

```
SELECT employees.name, employees.salary, departments.department_name
FROM employees
LEFT JOIN departments
ON employees.department_id = departments.id;
```

LEFT JOIN



- Tìm **nhân viên** có liên quan đến đơn hàng của cty ‘RED'

Bảng: SalesPerson

Column Name	Type
sales_id	int
name	varchar
salary	int
commission_rate	int
hire_date	date

Bảng: Company

Column Name	Type
com_id	int
name	varchar
city	varchar

Bảng: Orders

Column Name	Type
order_id	int
order_date	date
com_id	int
sales_id	int
amount	int

```
select p.name from salesperson p
left join orders o on p.sales_id=o.sales_id
left join company c
on c.com_id=o.com_id and c.name='RED';
```

Viet điều kien JOIN vao đau? ON hay WHERE

```sql
CREATE TABLE salesperson (
   sales_id INT PRIMARY KEY,
   name VARCHAR(100)
);

CREATE TABLE orders (
   order_id INT PRIMARY KEY,
   sales_id INT,
   region VARCHAR(50),
   FOREIGN KEY (sales_id) REFERENCES salesperson(sales_id)
);

-- Dữ liệu cho bảng salesperson
INSERT INTO salesperson (sales_id, name) VALUES
(1, 'Alice'),
(2, 'Bob'),
(3, 'Charlie'),
(4, 'Diana');

-- Dữ liệu cho bảng orders
INSERT INTO orders (order_id, sales_id, region) VALUES
(101, 1, 'North'),
(102, 1, 'South'),
(103, 2, 'North'),
(104, 2, 'East'),
(105, 3, 'South');
```

Viet dieu kien JOIN vao dau? ON hay WHERE

- Tìm **thông tin** về việc các nhân viên tham gia vào đơn hàng **phía Bắc** 'North'
- Sử dụng **LEFT JOIN** để biết thông tin những nhân viên nào **có/không** liên quan

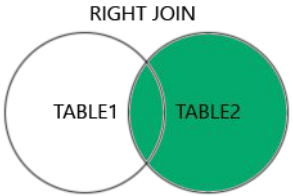
```
SELECT *
FROM salesperson s
LEFT JOIN orders o ON s.sales_id = o.sales_id
WHERE o.region = 'North';
```

#	sales_id	name	order_id	sales_id	region
1	1	Alice	101	1	North
2	2	Bob	103	2	North

```
SELECT *
FROM salesperson s
LEFT JOIN orders o
ON s.sales_id = o.sales_id
AND o.region = 'North';
```

#	sales_id	name	order_id	sales_id	region
1	1	Alice	101	1	North
2	2	Bob	103	2	North
3	3	Charlie	NULL	NULL	NULL
4	4	Diana	NULL	NULL	NULL

FULL OUTER JOIN



- FULL OUTER JOIN là kết hợp của LEFT JOIN và RIGHT JOIN

#	id	name	salary	department_id
1	1	Nguyễn Văn An	12000000.00	1
2	2	Trần Thị Bình	15000000.00	2
3	3	Lê Quốc Cường	13000000.00	2
4	4	Phạm Hồng Đăng	11000000.00	3
5	5	Hoàng Văn Nam	14000000.00	NULL

#	id	department_name
1	1	Nhân sự
2	2	Công nghệ thông tin
3	3	Tài chính
4	4	Marketing

FULL OUTER ⇒

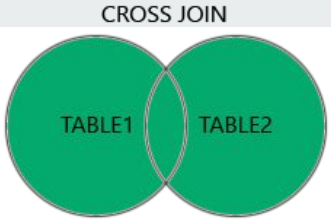
#	name	salary	department_name
1	Nguyễn Văn An	12000000.00	Nhân sự
2	Trần Thị Bình	15000000.00	Công nghệ thông tin
3	Lê Quốc Cường	13000000.00	Công nghệ thông tin
4	Phạm Hồng Đăng	11000000.00	Tài chính
5	Hoàng Văn Nam	14000000.00	NULL
6	NULL	NULL	Marketing

FULL OUTER JOIN

- MYSQL không hỗ trợ trực tiếp FULL OUTER JOIN ⇒ UNION (LEFT, RIGHT)

```
SELECT employees.name, employees.salary, departments.department_name
FROM employees
LEFT JOIN departments ON employees.department_id = departments.id
UNION
SELECT employees.name, employees.salary, departments.department_name
FROM employees
RIGHT JOIN departments ON employees.department_id = departments.id;
```

CROSS JOIN



- CROSS JOIN trả về tổ hợp các bản ghi từ các bảng ⇒ **Lưu ý tổ hợp không khớp**

```
SELECT employees.name,
departments.department_name
FROM employees
CROSS JOIN departments;
```

1	Nguyễn Văn An	Marketing
2	Nguyễn Văn An	Tài chính
3	Nguyễn Văn An	Công nghệ thông tin
4	Nguyễn Văn An	Nhân sự
5	Trần Thị Bình	Marketing
6	Trần Thị Bình	Tài chính
7	Trần Thị Bình	Công nghệ thông tin
8	Trần Thị Bình	Nhân sự
9	Lê Quốc Cường	Marketing
10	Lê Quốc Cường	Tài chính
11	Lê Quốc Cường	Công nghệ thông tin
12	Lê Quốc Cường	Nhân sự
13	Phạm Hồng Đăng	Marketing
14	Phạm Hồng Đăng	Tài chính
15	Phạm Hồng Đăng	Công nghệ thông tin
16	Phạm Hồng Đăng	Nhân sự

SELF JOIN

- SELF JOIN là JOIN bình thường nhưng kết hợp với chính nó
- Ví dụ tìm các cặp nhân viên cùng phòng ban với nhau

```
SELECT e1.name, e2.name
FROM employees e1
JOIN employees e2
ON e1.id < e2.id
AND e1.department_id = e2.department_id
```

#	name	name	department_id	(select d.department
1	Trần Thị Bình	Lê Quốc Cường	2	Công nghệ thông tin

```
SELECT e1.name, e2.name, e1.department_id,
     (select d.department_name from departments as d where d.id = e1.department_id)
FROM employees e1
JOIN employees e2
WHERE e1.id < e2.id AND e1.department_id = e2.department_id
```

JOIN - Ví dụ

- Tìm những nhân viên chưa tham gia vào phòng ban nào

```
SELECT e.name, d.department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.id
WHERE d.department_name IS NULL;
```

#	name	department_name
1	Hoàng Văn Nam	NULL

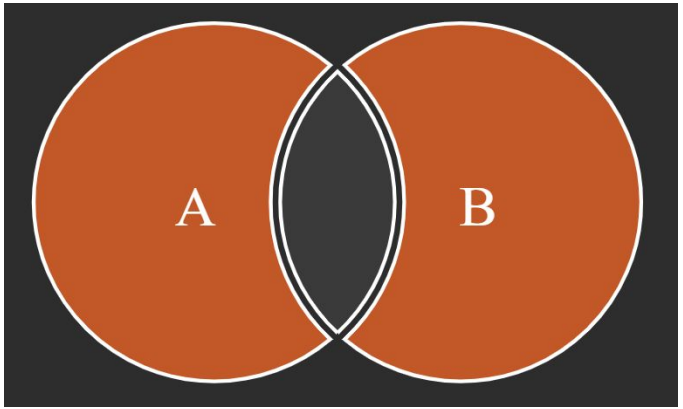
JOIN - Ví dụ

- Sử dụng **LEFT JOIN** để tìm những nhân viên **có / không** thuộc phòng **IT**
  - Nhận xét code sau:

```
SELECT e.name, d.department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.id
WHERE d.department_name = 'IT';
```

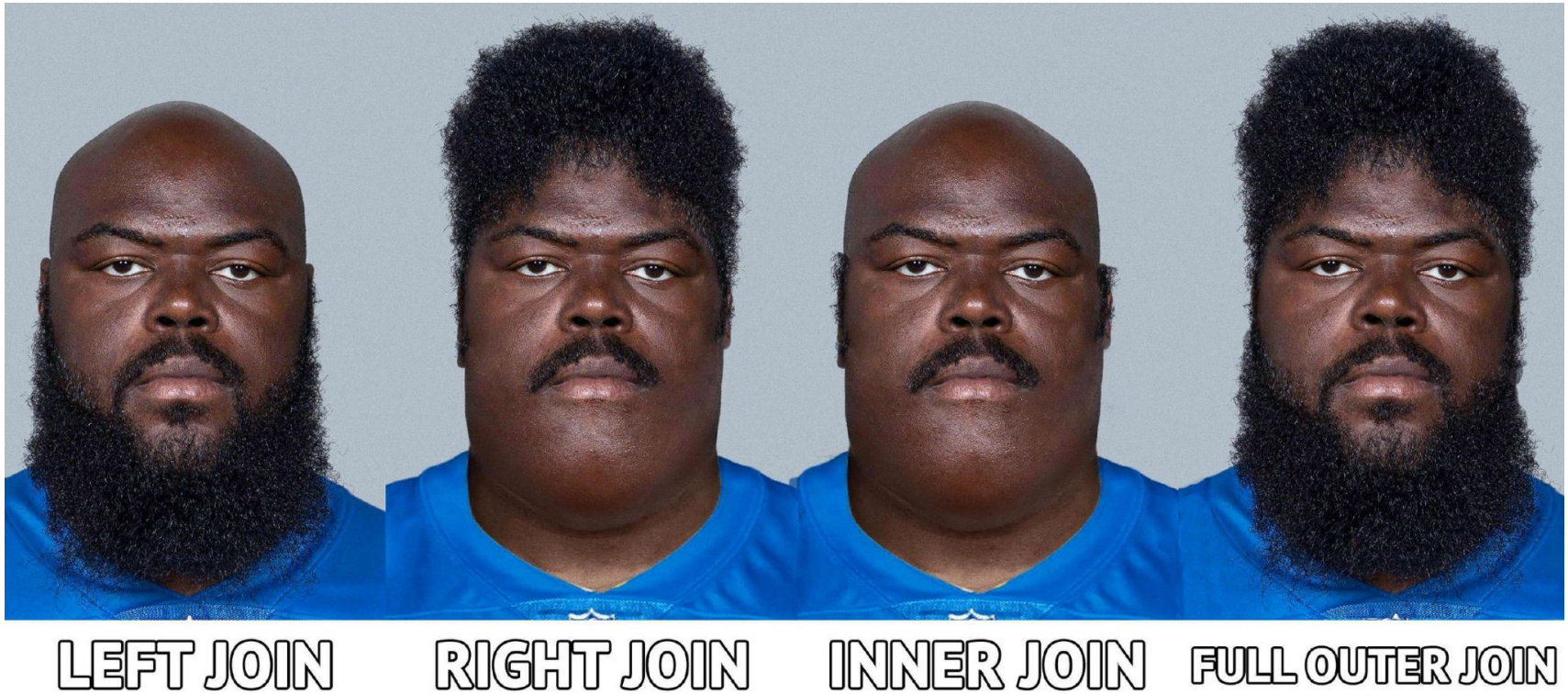
JOIN - Vi du

- Thay WHERE thành ON thì sao ?
- `ON A.key = B.key WHERE A.key IS NULL OR B.key IS NULL`




```
SELECT * FROM TableA A
FULL OUTER JOIN TableB B ON A.key = B.key
WHERE A.key IS NULL OR B.key IS NULL
```

JOIN



Bai tap JOIN

Self JOIN:

https://sql.zcode.vn/contests/CSDL_SQL/questions/cd56fbaef80a4a17aaadc0988f6939e6

https://sql.zcode.vn/contests/CSDL_SQL/questions/757629fbb6ed4a0cbc8436b775ae8ea7

https://sql.zcode.vn/contests/CSDL_SQL/questions/51fd72425da047afa8cfedaa5439017b

https://sql.zcode.vn/contests/CSDL_SQL/questions/e2919fc087ce4790a32b890a75d85a9d

https://sql.zcode.vn/contests/CSDL_SQL/questions/d9ba0cf5cfd14808b1f3686325ee3bef

LEFT JOIN:

https://sql.zcode.vn/contests/CSDL_SQL/questions/7d60a8c84e2e4a1f8083afd1f7b124b2

https://sql.zcode.vn/contests/CSDL_SQL/questions/6865e56862284a958973d9b7b2aeb0ab

USE CASE Tim kiem ton tai

https://sql.zcode.vn/contests/CSDL_SQL/questions/ddb45d3de1b3469e99dc8d44b4cac521

```sql
select distinct C1.seat_id
from Cinema as C1
join Cinema as C2
on ((C1.seat_id = C2.seat_id + 1) or (C1.seat_id = C2.seat_id - 1))
where C1.free = 1 and C2.free = 1
order by C1.seat_id
```

```sql
select seat_id
from Cinema as C1
where C1.free  = 1 and exists (
   select seat_id from  Cinema as C2
   where C2.free  = 1 and ((C2.seat_id = C1.seat_id - 1) or (C2.seat_id = C1.seat_id +1))
)
```

USE CASE Tim kiem ton tai

Database Northwind: Tìm thông tin những khách hàng đã từng đặt hàng trong năm khác 2007

```
select C.*
from Customer C
where C.custId IN (
   select custId from SalesOrder
   where YEAR(orderDate) != 2007
);
```

```
select C.*
from Customer C
where exists (
   select 1 from SalesOrder SO
   where SO.custId = C.custId
   and YEAR(orderDate) != 2007
);
```

```
select distinct C.*
from Customer C
join SalesOrder SO
where SO.custId = C.custId
and YEAR(orderDate) != 2007
```

USE CASE Tim kiem ton tai

Database Northwind: Tìm thông tin những khách hàng đã từng đặt hàng trong năm khác 2007

1/ **Distinct** khiến cho chạy chậm do cần so sánh toàn bộ các dòng, không tối ưu

2/ Trong bài toán kiểm tra tồn tại, nếu join thì tạo ra **nhiều bản ghi dư thừa**

3/ EXISTS chỉ cần tìm thấy **1 đáp án đúng** trong subquery là sẽ **dừng**