

1. Cấu trúc cơ bản

```
SELECT [ ALL | DISTINCT ]  
      [ TOP n [ PERCENT ] ]
```

```
      * | {column_name | expression [alias],...}
```

```
FROM tableName
```

```
[WHERE conditional expressions]
```

```
ORDER BY expression1 [ASC | DESC], expression2 [ASC | DESC] ...
```

- SELECT: Phép chiếu để chọn ra các cột của bảng. Có thể kết hợp với Distinct để loại trừ trùng lặp. Top / Percent để hiển thị n dòng / n% dòng đầu tiên của bảng.

- FROM: chọn bảng để thao tác

- WHERE: điều kiện áp dụng vào bảng

- ORDER có thể có hoặc không, hiển thị theo thứ tự sắp xếp được yêu cầu. Sắp xếp expression1 trước, nếu bằng nhau thì sắp xếp theo expression 2.

2. Đổi tên / Đặt tên: Sử dụng "as"

Ví dụ: SELECT Product as Productname (Sẽ hiển thị cột Product nhưng tên cột sẽ thay bằng Productname)

Một ví dụ khác:

```
(select top 5 UnitPrice  
from Products  
order by UnitPrice desc) as Top5
```

Tạo bảng mới Tên là Top5 (đây là câu lệnh con, đặt tên bảng là Top 5, sẽ sử dụng được trong câu lệnh From. Nếu không đặt tên thì không sử dụng được trong FROM - cái này thuộc về câu lệnh con, không hiểu thì cũng không sao có thể bỏ qua)

3. s LIKE p

Sử dụng để lọc ra các giá trị s giống p

"%" đại diện cho 1 chuỗi kí tự, "_" đại diện cho 1 kí tự.

VD: WHERE title LIKE 'man%' = title bắt đầu bằng 'man'

'%man%' = title có chứa 'man'

'-----' = title chỉ có 5 kí tự.

'[S,Q]%' = bắt đầu với S hoặc Q (lệnh gộp)

3. Các toán tử

<, >, <=, >=, =, <>

AND, OR

4. Quan hệ

$$A \bowtie B = \delta_{A.class_id = B.class_id}(A \times B)$$

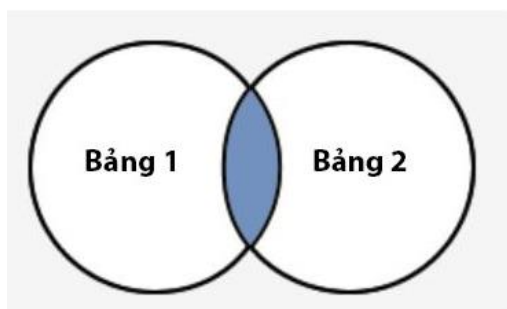
```
SELECT *  
FROM Student A, Class B  
WHERE A.class_id = B.class_id
```

```
SELECT *  
FROM Student A INNER JOIN Class B  
on A.class_id = B.class_id
```

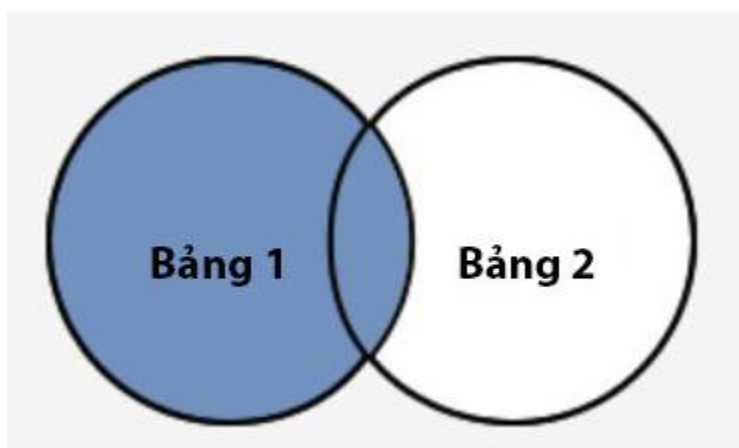
```
SELECT cot  
FROM bang1  
RIGHT [OUTER] JOIN bang2  
ON bang1.cot = bang2.cot;
```

Thay lệnh này bằng các kiểu join phù hợp với yêu cầu

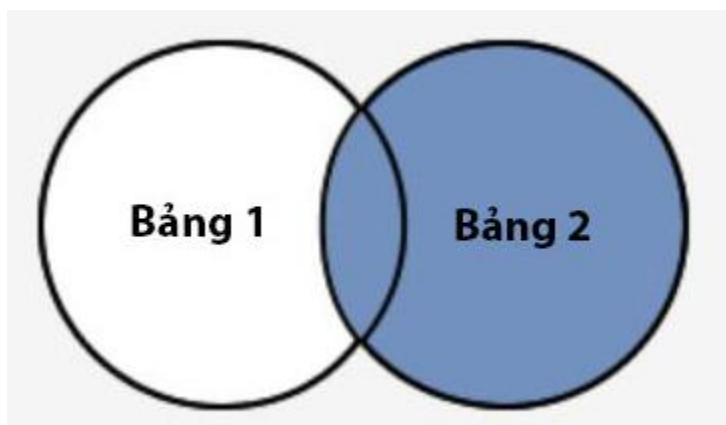
A. INNER JOIN



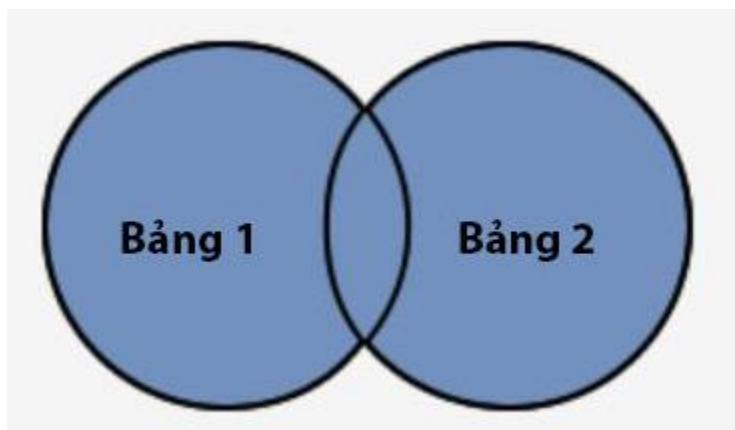
B. LEFT JOIN



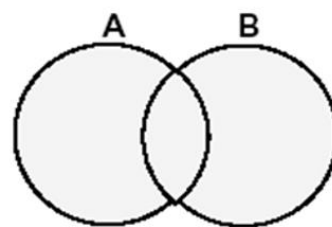
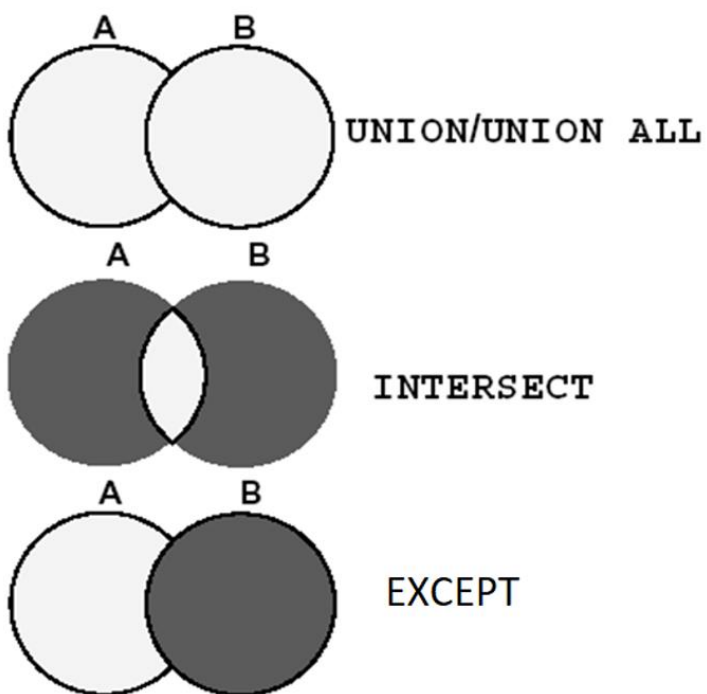
C. RIGHT JOIN



D. FULL JOIN



E. Phép giao, phép hợp, phép loại trừ: UNION sẽ loại bỏ trùng lặp trong khi UNION ALL giữ trùng lặp.



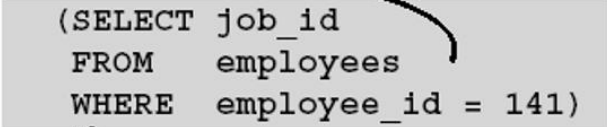
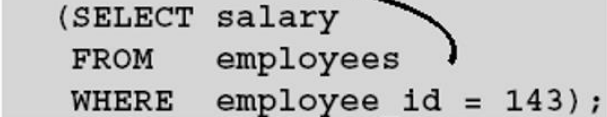
Ví dụ:

```
SELECT employee_id, job_id
FROM employees
UNION
SELECT employee_id, job_id
FROM job_history;
```

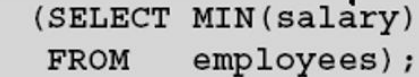
```
SELECT employee_id, job_id, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

5. Truy vấn con: Cái nào trong ngoặc thì làm trước. Như làm toán.

VD:

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id = 
AND salary > ;
```

The first example shows a query with two subqueries. The first subquery, `(SELECT job_id FROM employees WHERE employee_id = 141)`, is highlighted in a grey box and has an arrow pointing to the `job_id` column in the `WHERE` clause, with the label "ST CLERK" above it. The second subquery, `(SELECT salary FROM employees WHERE employee_id = 143)`, is also highlighted in a grey box and has an arrow pointing to the `salary` column in the `WHERE` clause, with the value "2600" above it.

```
SELECT last_name, job_id, salary
FROM employees
WHERE salary = ;
```

The second example shows a query with a single subquery, `(SELECT MIN(salary) FROM employees)`, highlighted in a grey box. An arrow points from this subquery to the `salary` column in the `WHERE` clause, with the value "2500" above it.

Đây là ví dụ khi nó ở where, nên sẽ là trả về 1 cột để so sánh nhưng cũng có thể đặt ở Select hoặc From miễn sao ở Select thì đó phải là tên cột, còn ở From thì là tên bảng, để làm được điều này thì truy vấn con sẽ dùng thêm 'as' để đặt tên. Xem lại ví dụ mục đổi tên/đặt tên (Mục 2).

Thêm 1 vài toán tử để dùng trong câu lệnh điều kiện:

- **EXISTS R**: is a condition that is true if and only if R is not empty
- **s IN R**: is true if and only if s is equal to one of the values in R
- **s > ALL R**: is true if and only if s is greater than every value in unary relation R
- **s > ANY R**: is true if and only if s is greater than at least one value in unary relation R

VD: WHERE MathPoint > ALL PhysicPoint v...v...

6. Các toán tử quan hệ khác:

❖ **AVG**
❖ **COUNT**
❖ **MAX**
❖ **MIN**
❖ **SUM**
❖ **STDDEV**
❖ **VARIANCE**

AVG([distinct <u>ALL</u>] column expression)
COUNT({ * [distinct <u>ALL</u>] column expression})
MAX([distinct <u>ALL</u>] column expression)
MIN([distinct <u>ALL</u>] column expression)
STDDEV([distinct <u>ALL</u>] column expression)
SUM([distinct <u>ALL</u>] column expression)
VARIANCE([distinct <u>ALL</u>] column expression)

Ngoài ra có, GROUP BY - GROUP các giá trị giống nhau

```
SELECT column, aggregate_function (column)
FROM table
[WHERE conditions]
[GROUP BY grouping_attributes]
[HAVING conditions]
[ORDER BY {column [ASC | DESC] ,...} ]
```

HAVING: cách để đặt điều kiện cho GROUP BY

```
SELECT bophan, SUM (soluong) AS "Tong so luong"  
FROM sanpham  
GROUP BY bophan  
HAVING SUM (soluong) > 100;
```

copy

```
SELECT thanhpho, COUNT (*) AS "So nhanvien"  
FROM nhanvien  
WHERE bang = 'California'  
GROUP BY thanhpho  
HAVING COUNT (*) > 20;
```

copy