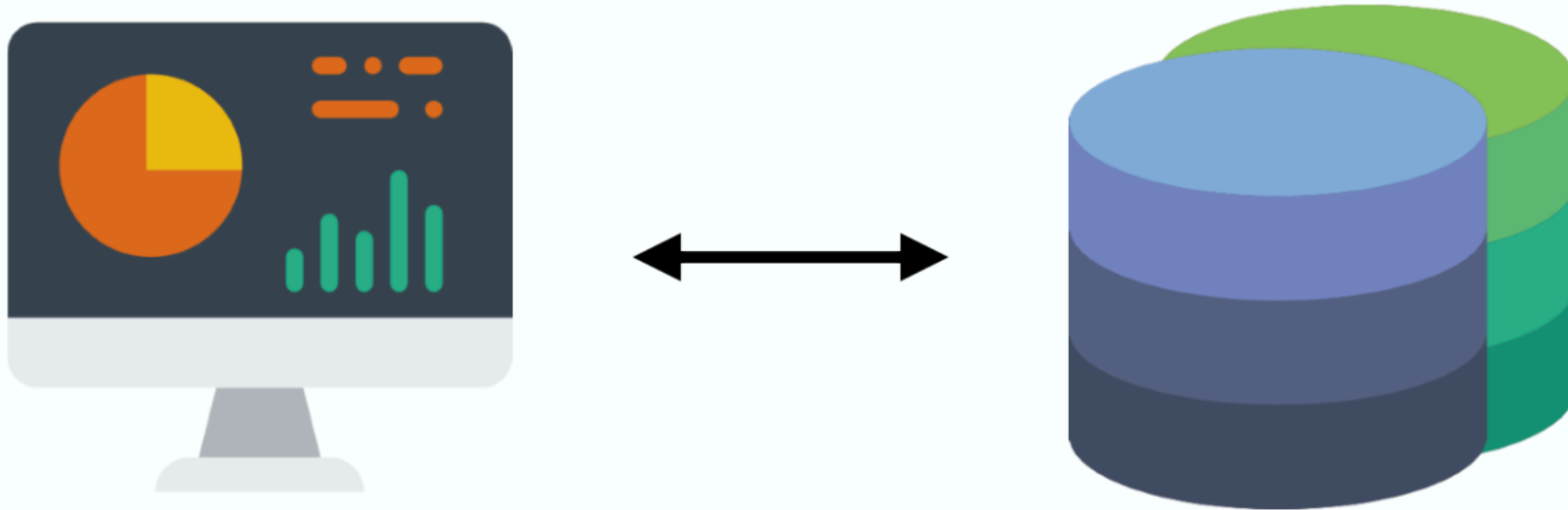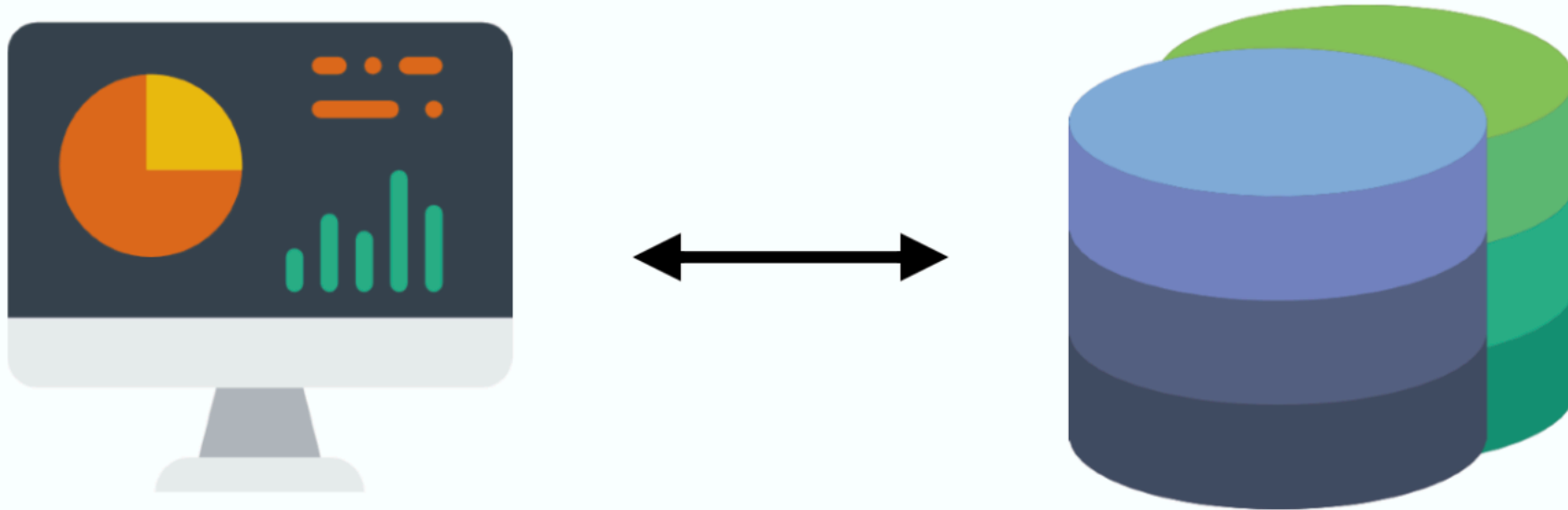# Lab
Advanced SQL

Software Studio

DataLab, CS, NTHU

Why using DBMS

Why using DBMS

From the client's point of view?
From the developer's point of view?
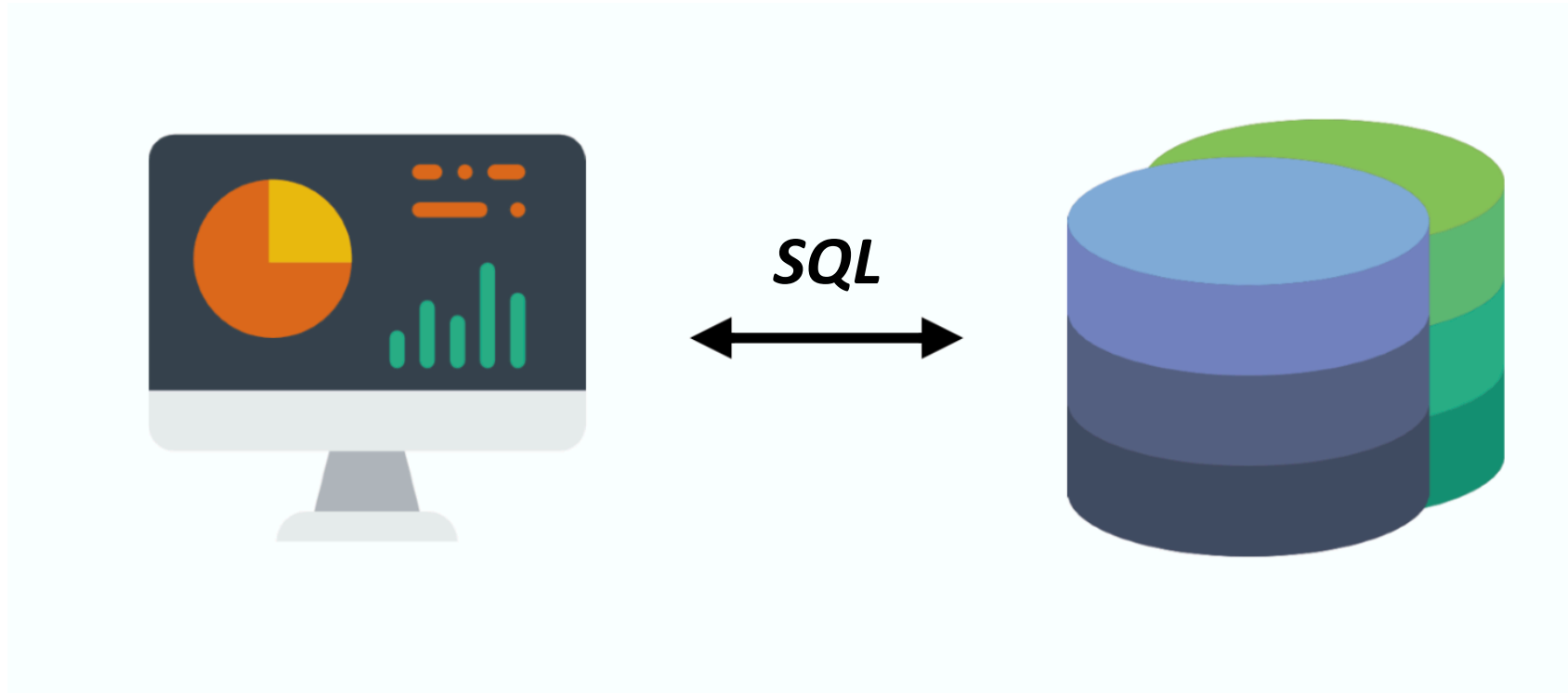
Using DB wisely saves plenty of time

- Database are written by some of biggest company in the world.

SQL

- To communicate to all database in the world, we need a standard language.

Select Review

Student	
s_id	Primary key
s_name	名稱
s_level	等級
s_class	職業
s_lif	生命
s_atk	攻擊
s_def	防禦
s_mag	魔力
s_bs	伴侶

Select Review

Student	
s_id	Primary key
s_name	名稱
s_level	等級
s_class	職業
s_lif	生命
s_atk	攻擊
s_def	防禦
s_mag	魔力
s_bs	伴侶

What is primary key?

Select Review

Student	
s_id	Primary key
s_name	名稱
s_level	等級
s_class	職業
s_lif	生命
s_atk	攻擊
s_def	防禦
s_mag	魔力
s_bs	伴侶

- Which students' level more than 10?

```
SELECT * FROM student
WHERE s_level > 10
```

Select Review

Student	
s_id	Primary key
s_name	名稱
s_class	職業
s_level	等級
s_lif	生命
s_atk	攻擊
s_def	防禦
s_mag	魔力
s_bs	伴侶

Class	
c_id	Primary key
s_name	名稱
c_b_lif	生命加成
c_b_atk	攻擊加成
c_b_def	防禦加成
c_b_mag	魔力加成

Select Review

Student	
s_id	Primary key
s_name	名稱
s_level	等級
s_class	職業
s_b_lif	生命加成
s_b_atk	攻擊加成
s_b_def	防禦加成
s_b_mag	魔力加成
s_lif	生命
s_atk	攻擊
s_def	防禦
s_mag	魔力
s_bs	伴侶

Why is this schema design bad?

Query on multiple table

- Scenario:
  - How to query a student's information and class name at the same time?
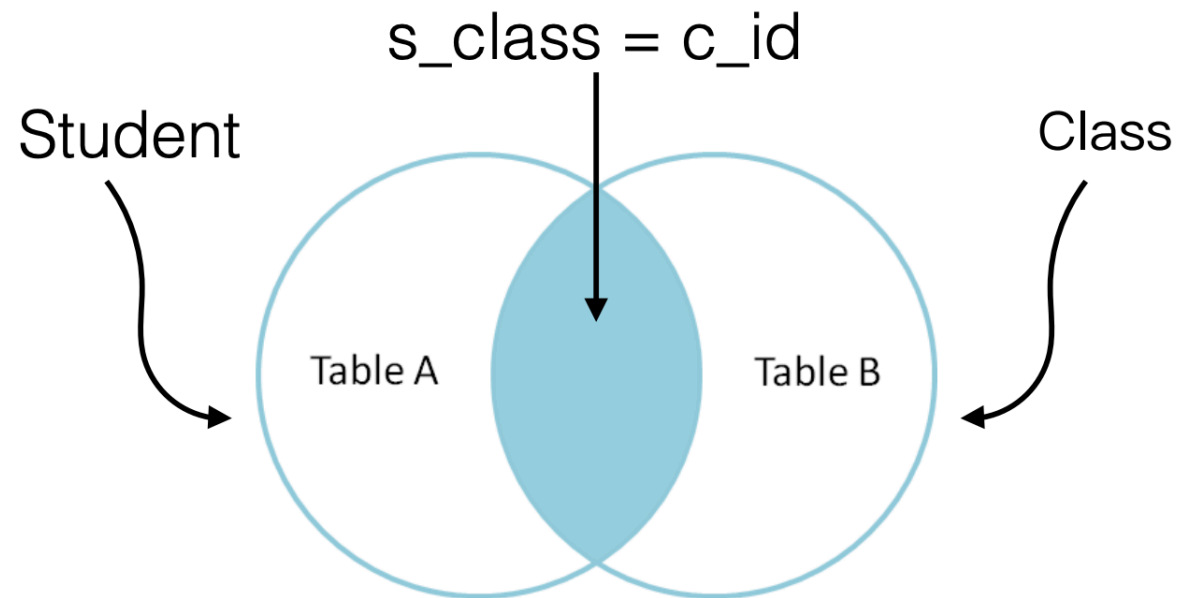
```
SELECT * FROM student, class
WHERE s_id = 10
AND s_class = c_id;
```
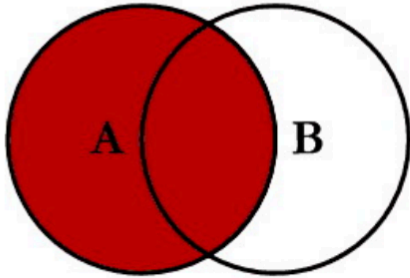
OR

```
SELECT * FROM student
JOIN class ON s_class = c_id
WHERE s_id = 10 ;
```
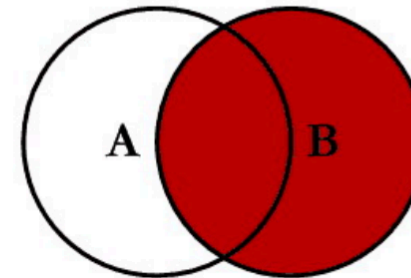
Join

```
SELECT * FROM student
JOIN class ON s_class = c_id
WHERE s_id = 10 ;
```
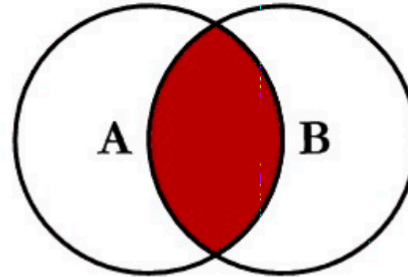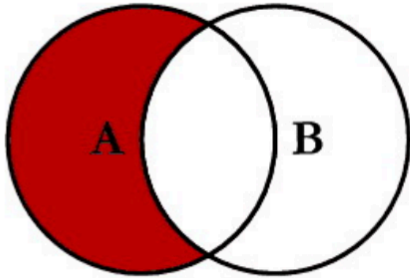

s_class = c_id

Student

Class

Table A

Table B

SQL JOINS

SELECT <select_list>
FROM TableA A
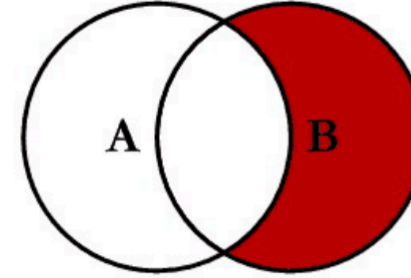LEFT JOIN TableB B
ON A.Key = B.Key

SELECT <select_list>
FROM TableA A
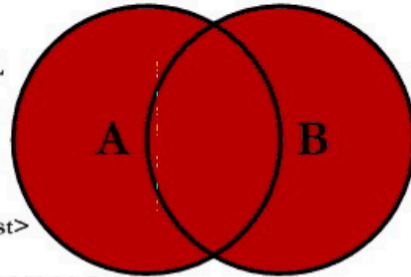RIGHT JOIN TableB B
ON A.Key = B.Key

SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
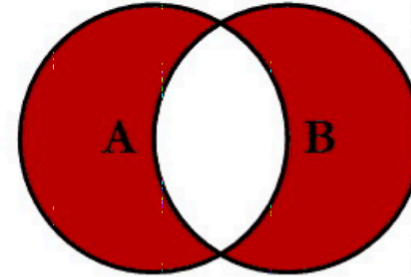
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL

SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL

SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key

SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
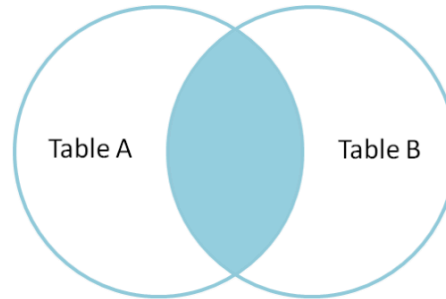
© C.L. Moffatt, 2008

Inner Join

- Scenario:
    - How to query a payment with its buyer names?

Payment	
p_id	Primary key
p_buy_id	買家
p_sel_id	賣家
p_name	名稱
p_price	價格

Inner Join

- Scenario:
  - How to query a payment with its **buyer names**?

Student	
s_id	Primary key
s_name	名稱
s_level	等級
s_class	職業
…	…

Payment	
p_id	Primary key
p_buy_id	買家
p_sel_id	賣家
p_name	名稱
p_price	價格

Inner Join

- Scenario:
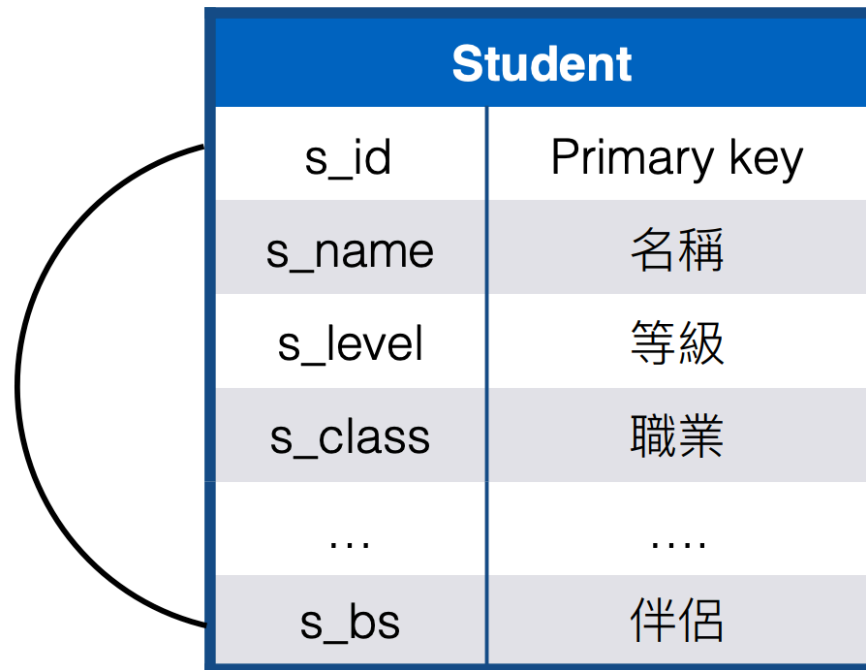 - How to query a payment with its **buyer names**?
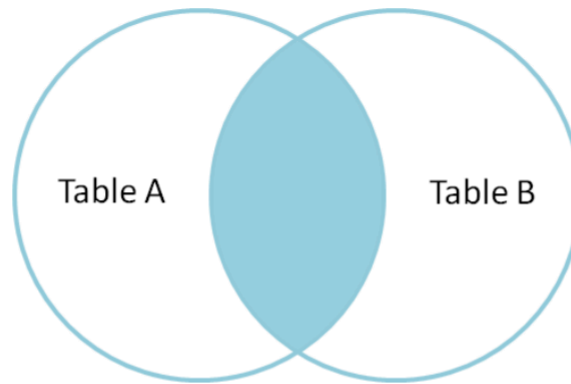

```
SELECT s_name, p_name FROM student
INNER JOIN payment on s_id = p_buy_id;
```

Self Join

- Scenario :
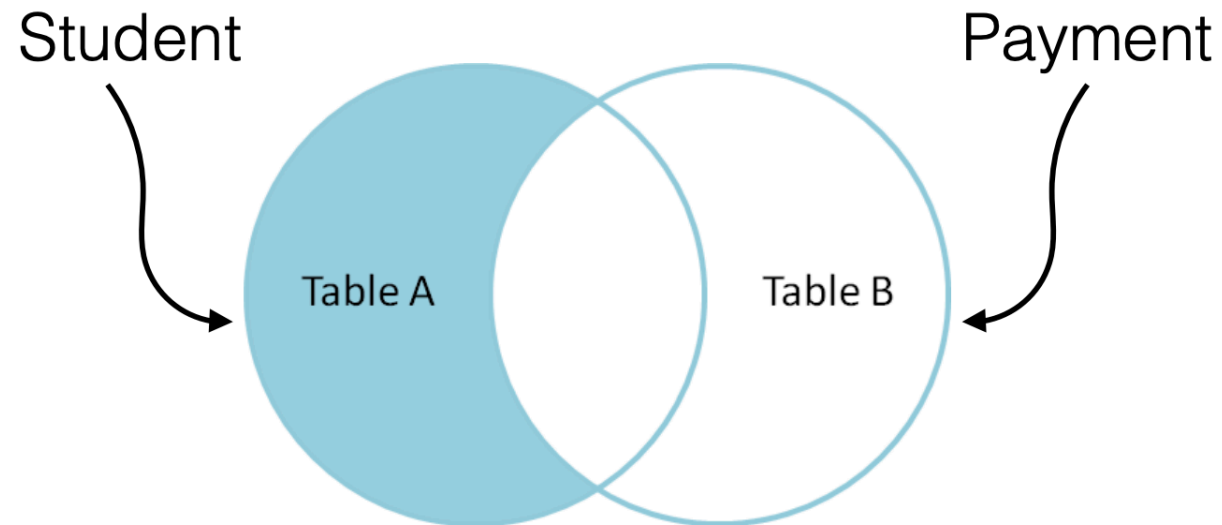 - How to get best friends pairs in student?

Self Join

- Scenario :
  - How to get best friends pairs in student?
  - Same as the previous join

```
SELECT s1.s_name, s2.s_name
FROM student s1
INNER JOIN student s2
ON s1.s_bs = s2.s_id;
```
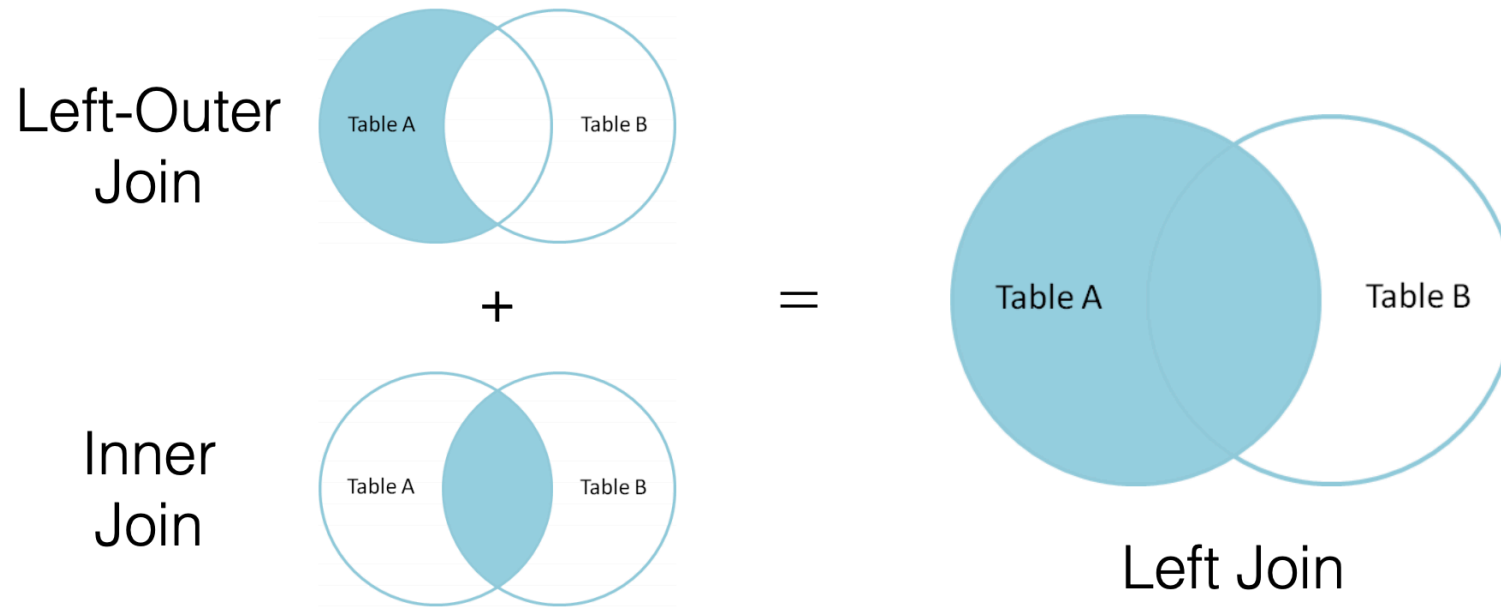
Left outer join

- Scenario :
  - Who haven't buy an item?

Left outer join

- Unfortunately, SQL don't have native left outer join
- But SQL have left join!

Left outer join

- Scenario :
  - Who haven't buy an item?

```
SELECT * FROM student
LEFT JOIN payment on s_id = p_buy_id
WHERE payment.p_buy_id is NULL;
```

Left outer join

- Scenario :
  - Who haven't buy an item?

Left Outer Join

Left Join

```
SELECT * FROM student
LEFT JOIN payment on s_id = p_buy_id
WHERE payment.p_buy_id is NULL;
```
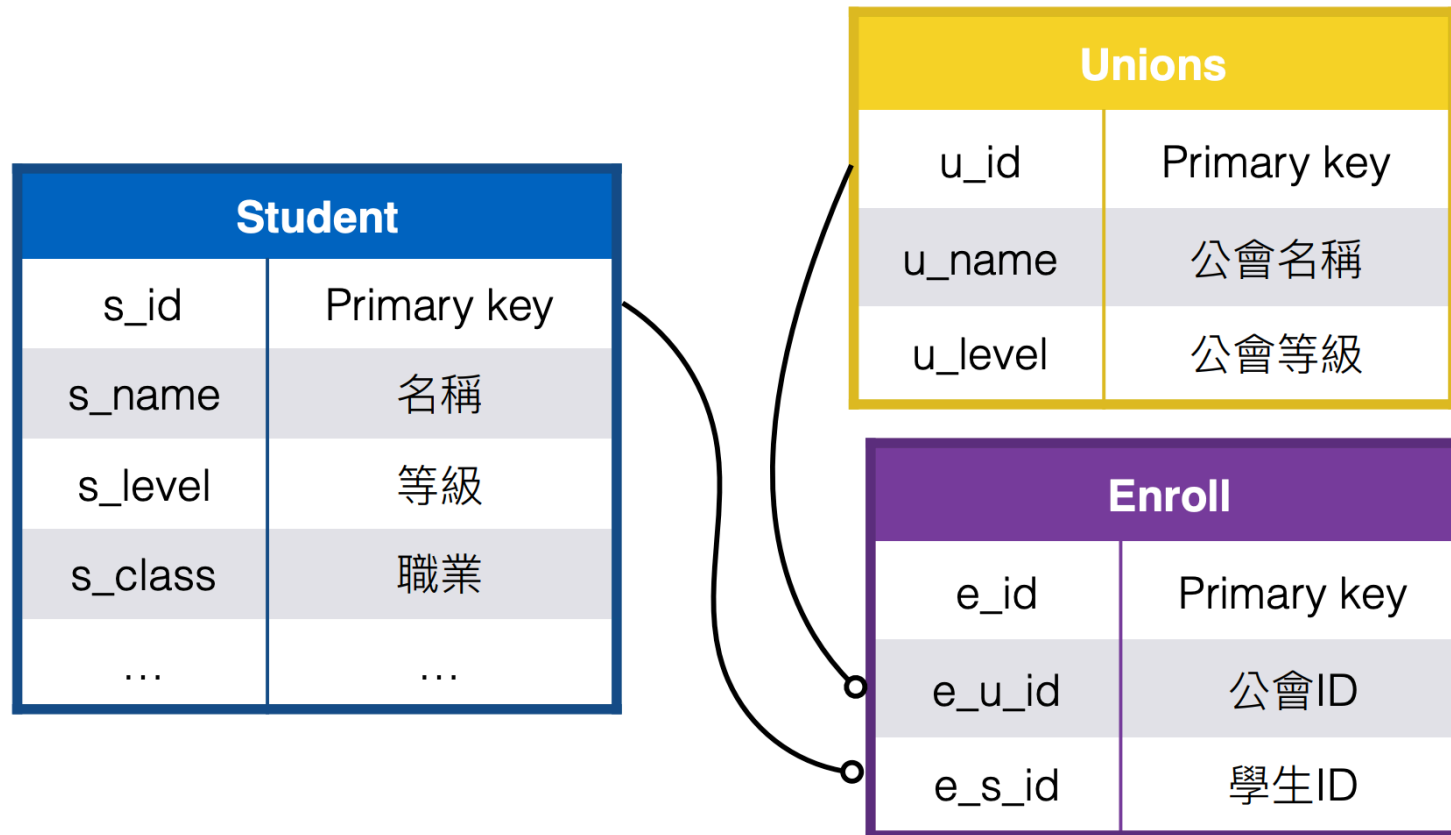
Only select students that ~~don't~~ have NULL p_buy_id

Why not store multiple key in one field ?

Student	
s_id	Primary key
s_name	名稱
s_level	等級
s_class	職業
…	….
s_unions	1,2…

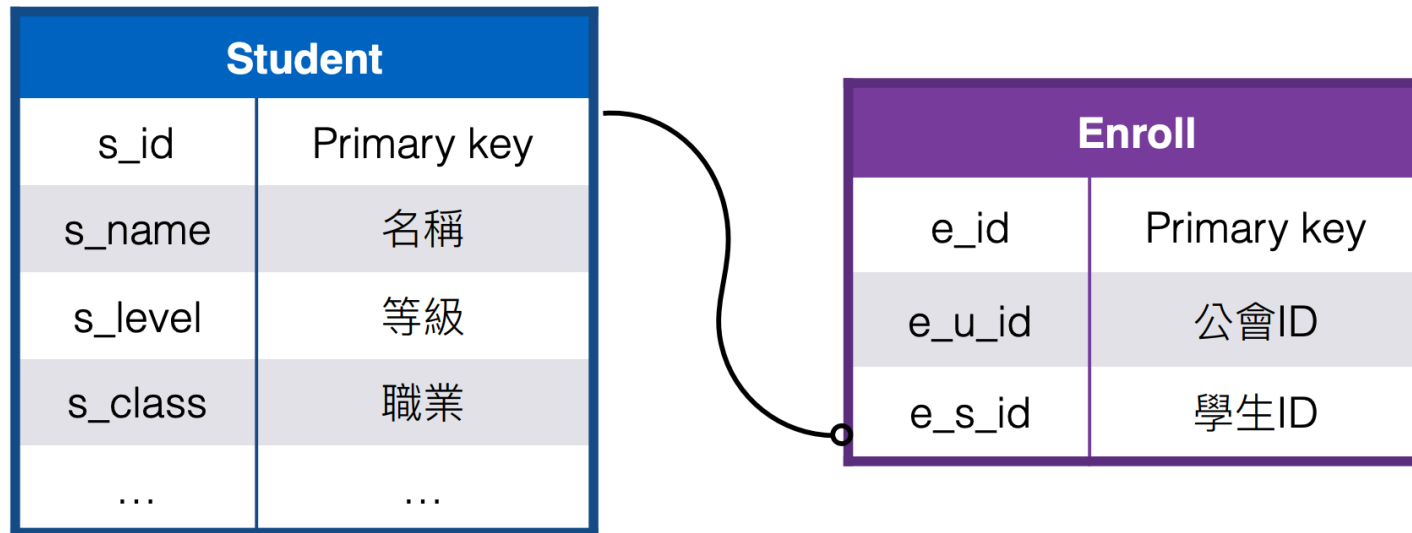| Unions	
u_id	Primary key
u_name	公會名稱
u_level	公會等級

Why not store multiple key in one field ?

Group By and Aggregation

- Scenario:
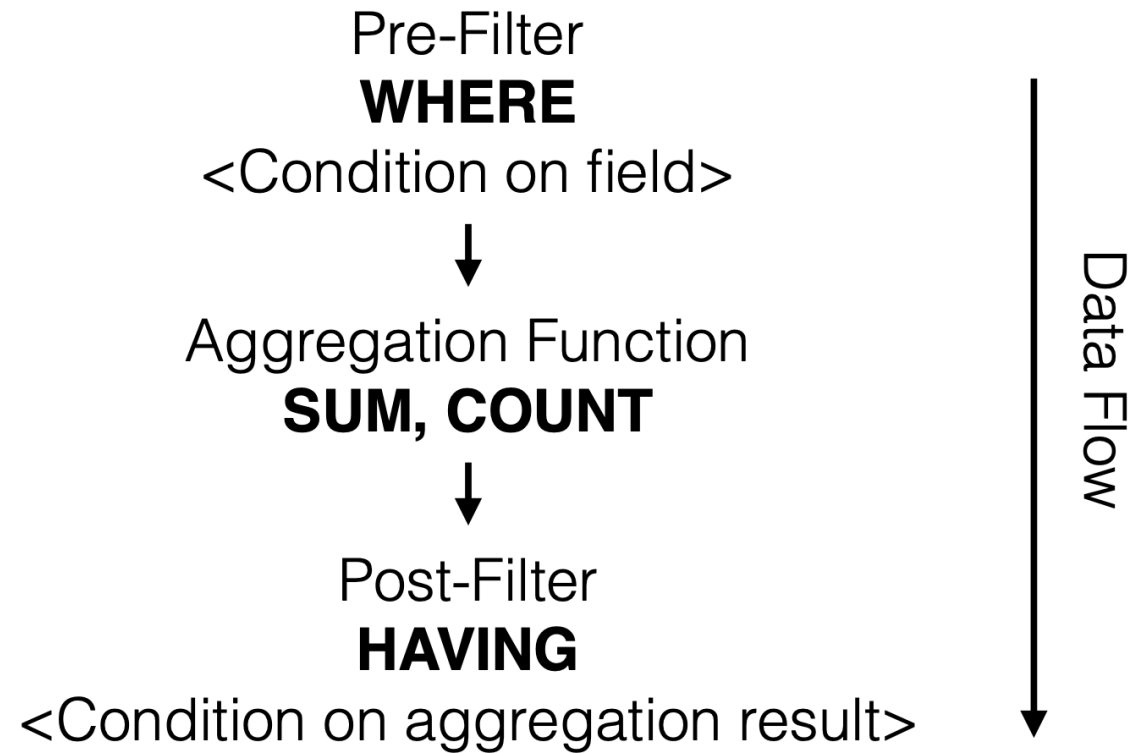 - What is sum of attack in a union?

Group By and Aggregation

- Scenario:
  - What is sum of attack in a union?

```
SELECT e_u_id, sum(s_atk) FROM student
INNER JOIN enroll on s_id = e_s_id
GROUP BY e_u_id;
```

Enroll	
e_id	Primary key
e_u_id	公會ID
e_s_id	學生ID

Having ? Where?

Pre-Filter
WHERE
<Condition on field>

↓

Aggregation Function
SUM, COUNT

↓

Post-Filter
HAVING
<Condition on aggregation result>

Data Flow →

Having ? Where?

- Scenario :
  - Which unions that sum of attack more than 300?

```
SELECT e_u_id , sum(s_atk) FROM student
INNER JOIN enroll on s_id = e_s_id
GROUP BY e_u_id HAVING sum(s_atk) > 300;
```