Agenda

- → Joining multiple Tables
- → Compound Joins
- → Types of Joins
- → Using } syntax sugar.
- → Natural }
- → implicit Join
- → Union

Joining multiples Tables.

1 3 6
(1) + (2) + (3) + 4 + 5
+2

Join on Join
((A B) (C)
→ (AB)
Intermidiate

students

id	name	instructor_id	batch_id

batches

id name.

instructors

id name.

Print the name of the std, their batch and the instructor name.

Join (students, batches) intermidiate

Join (_____, instructors)

Pseudo

```
ans = []
for row in ~~filter~~ students
    for row1 in batches
        if (condit^n)
            ans.appen(row1 + row2)

ans2 = []
for row in ans
    for row3 in instructors
        if (conditions match)
            ans2.append(row + row3)
```

Join1

Join2

for each in arr 2

result print (filter col^m).

Compound Joins

For every film, name all the films that were released in the range of 2 year and their rental rate was more than the rate of the movie.

film

name	rate	release_year
3 idiots	3	1998
Animal	2	2000
KK111	1	1997.

Self Join.

2 > 3 ?

Animal Join 3 idiots

3 > 2 ✓ 1998 - 2000 → 2002

AND

rate > release_year Range(2) eg-2

release_year (release year,)

2 2

```
Select  f1.name , f2.name
from    film f1
Join    film f2          } self Join
on  ( f2.rental > f1.rental)  AND
      ( f2.year Between f1.year - 2 and
                        f1.year + 2)
```

join condition

Types of Joins

inner join : default in SQL.

"Join" → "inner join"

```
select  s1.name , s2.name
from    students s1
Join    students s2
on  s1.buddy_id = s2.id ✓
```

id	name	$buddy_id
1	A	2
2	B	Null

3	C	2
4	D	2

1 A 2 2 B Null.

→

outer join

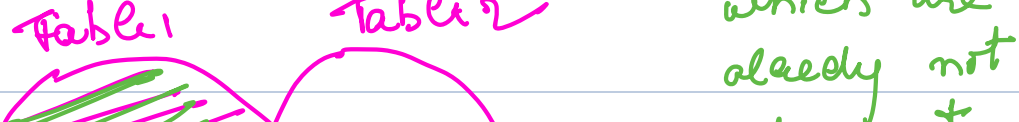
- → left outer join
- → right outer join
- → full outer join

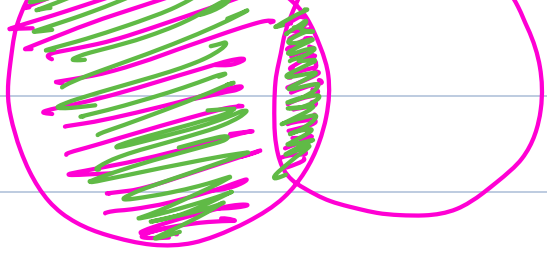
left outer join

include the rows that match the condition

+

include rows in left table.





present.

Students

	id	name	batch_id
present	1	John	1
present	2	Jim	1
	3	Jenny	null
present	4	Jack	3

stu.batchid = batches.id

name → batch

batches

id	name
1	A
2	B
3	C

left | right | inner join

John	A.
Jim	A
Jack	C
Jenny.	null.

left join

left (stu) right (batches)

1	John	1	1	A
2	Jim	1	1	A
3	Jenny	null	null	null.
4	Jack	3	3	C.

pseudo

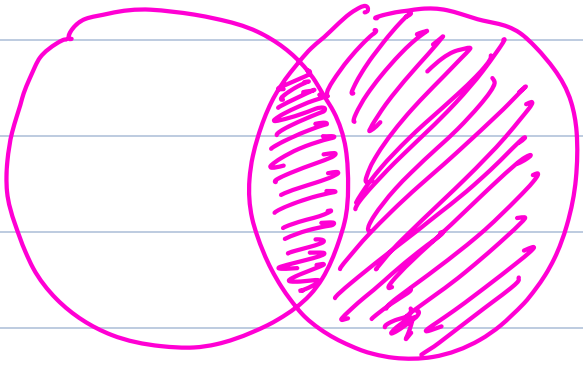
ans = []

intersect:
for row in Table1
  for row in Table2
    if (condition match)
      ans.append(row + row2)

left part:
for row in Table1
  if (row not in ans):
    ans.append(row + null)

lecture ans

right outer join



intersection + rows in right that are already not present.

John A
Jim A
} intersect

Jack C

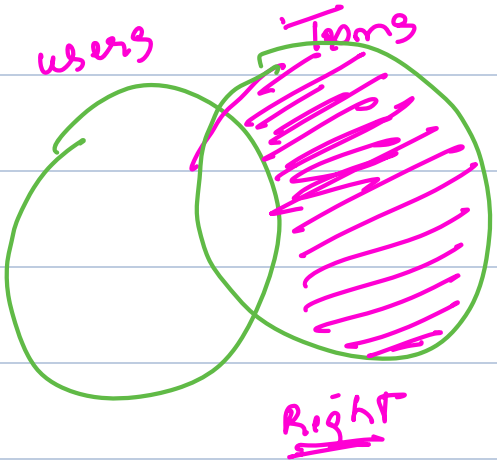
null B

batches

present 1 A

2 B

present 3 C



users left. null. → transact^n right. users.id != null

full outer join

intersect^n + rows in right (not present already) + rows in left (")

John A.

Jim A

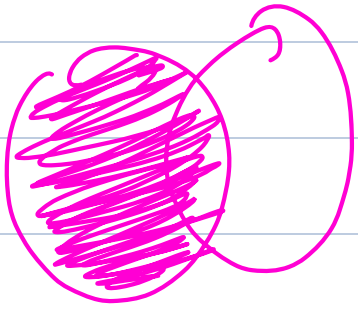
Jack C.

3 | Jenny | null. | left.

null B. right

Trick :

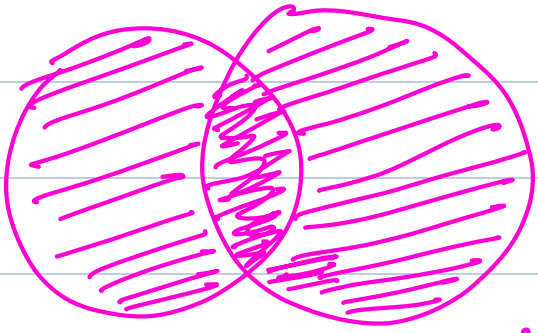
① if Left join, left cannot null



② for right join, right $ can't be null.

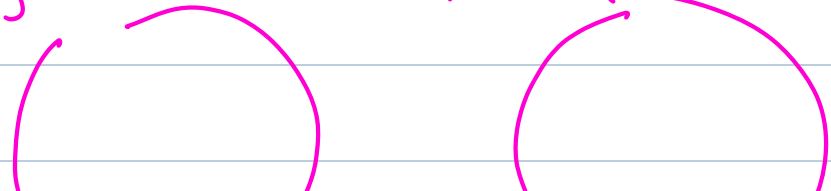
③ for inner, no nulls

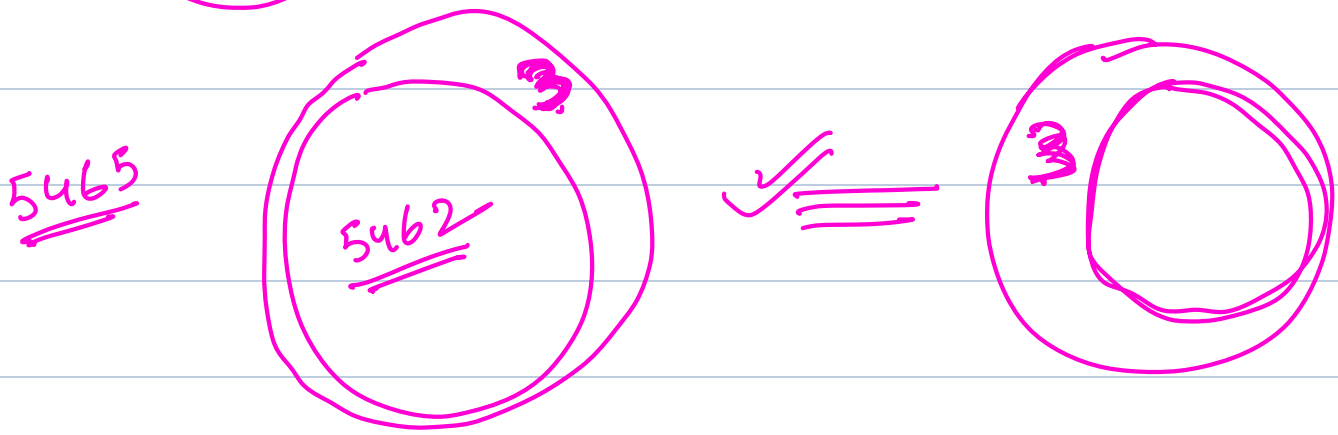
Break Till 10:38.

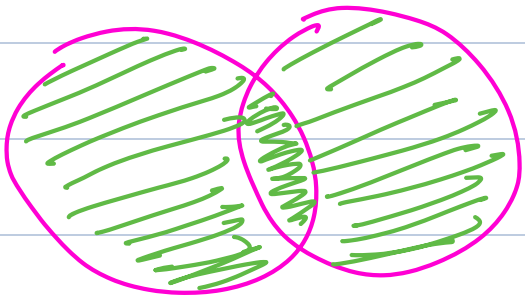


fa

film

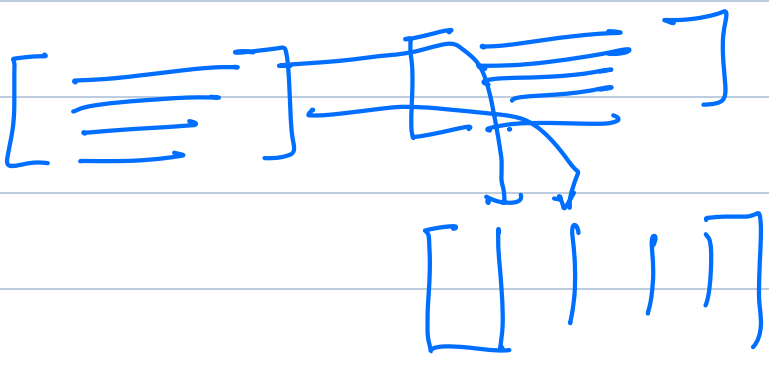






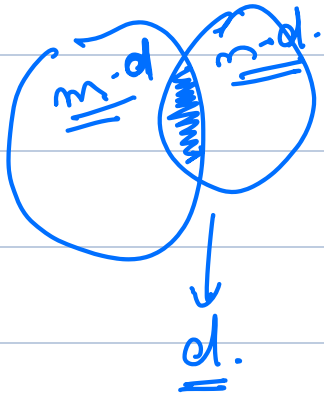
$A \cup B \equiv (A + B) - (A \cap B)$

Cross Join



select * from film_actor fa
cross join film f.

cross Join = M x n
outer Join = M + n



$(m-d) + (n-d')$

Syntax sugar

+2 QL

using

1. Joining should be on equality (equi-join).
2. Name of column should be same on both sides.

```
select *
from students s
join batches b
on s.batch_id = b.batch_id
```

```
select *
from students s
join batches b
using (batch_id)
```

Natural Join

Join 2 Tables on equality with same col^n name in both.

using (A, B, C, D, E, ...).

Implicit Join

int x = 4; float a = 12.4, b = a/x.

a/float(x)

implicit conversion b/w data types:

b : float
a : float
x : int.

```
select * from a, b.
```

||

```
select * from a
join b
```

cross join (m*n)

```
select * from a, b where a.id
                = b.ref-id.
```

Implicit join

||

```
select * from cross join where.
    a.id = b.ref-id.
```

pseudo

```
ans = []
```

```
for row in a
    for row2 in b:
        ans.append(ans)

filtered_ans = []

for row in ans
    if (where condition)
        row.append(filtered_ans)

return filtered
```

cross join with "where" ≡ inner join

```
for row1 in a
    for row2 in b
        if (condition match)
            append.
```

speed

inner > implicit (where).

space.

inner (n*m - d).

implicit (n*m).

Unions

students
id name

employees
id name

investors
id name.

union allows you to combine the output of one query after another

① select name from students
union
② select name from employees
union
③ " " investors

constraints

① each query should output the same number of col^m.

② you can't use order-by.

→ These queries are executed independently.

[≡] sorted [≡] sorted

[≡] sorted ??

unions work on sets

union all work on list.

left union right

≡ full outer join