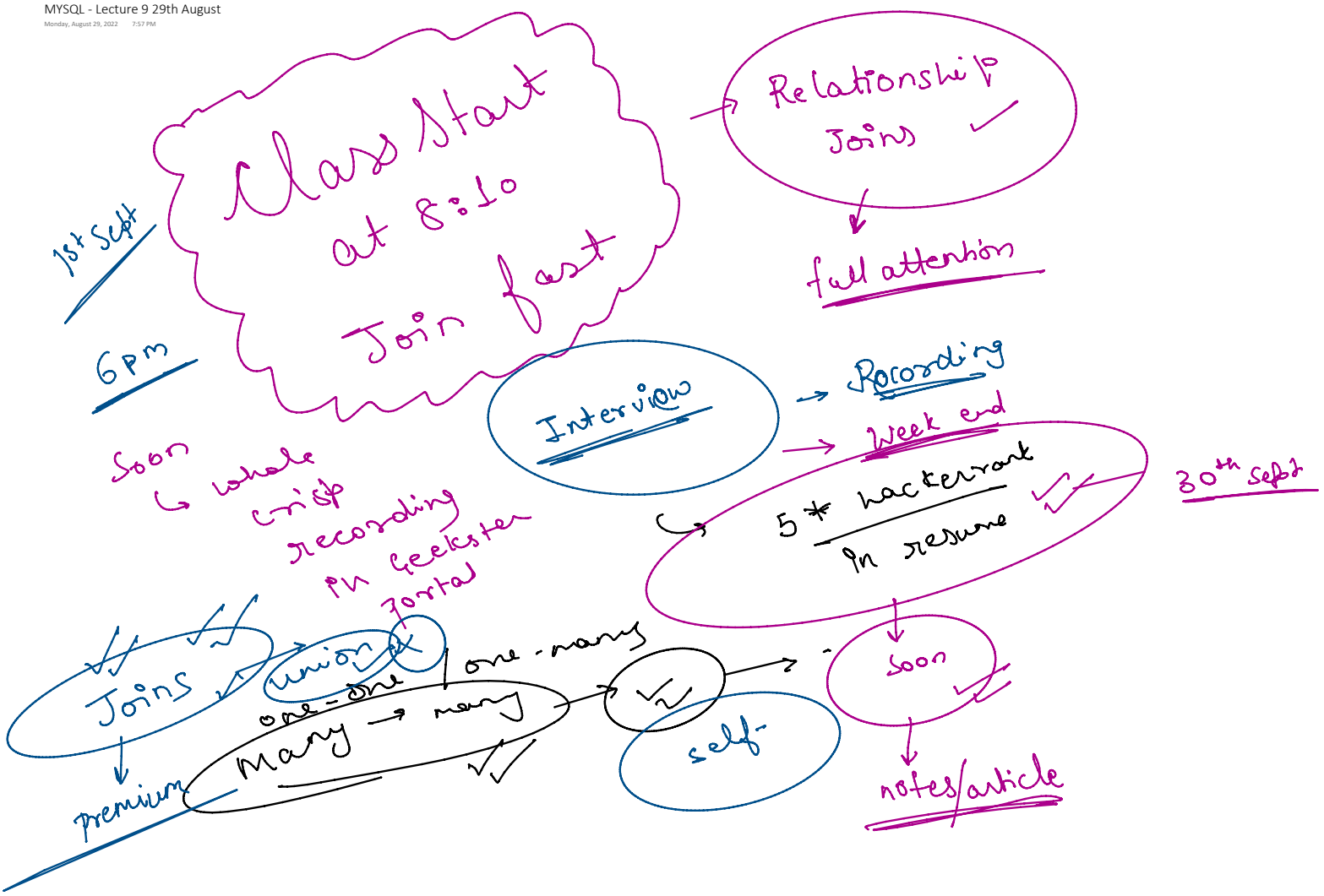
MYSQL - Lecture 9 29th August

Monday, August 29, 2022 7:57 PM



Relationship

↳ 3 types of relationship.

1. One to One
2. One to many
3. Many to Many

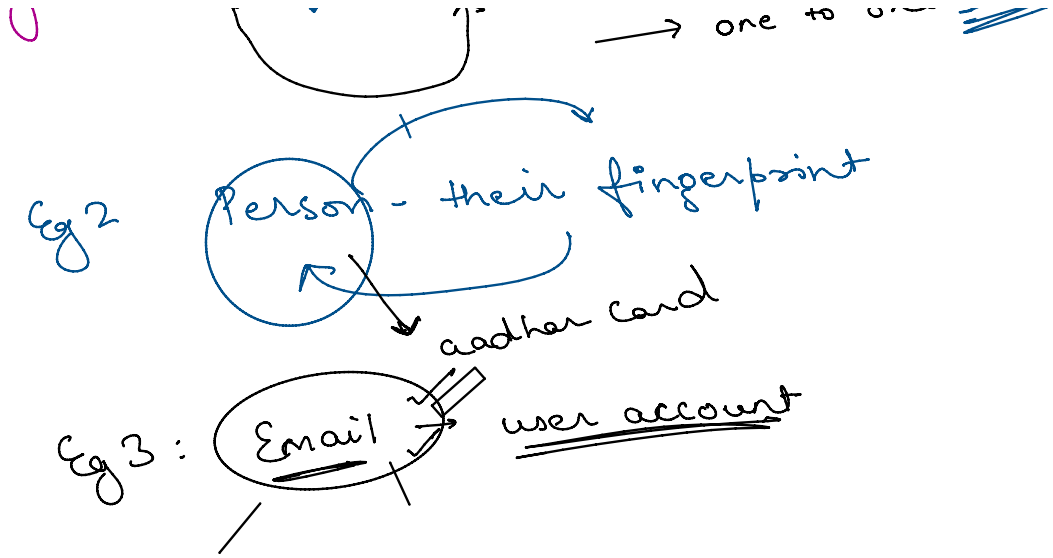
} today

→ recording (easy)

1 One to One

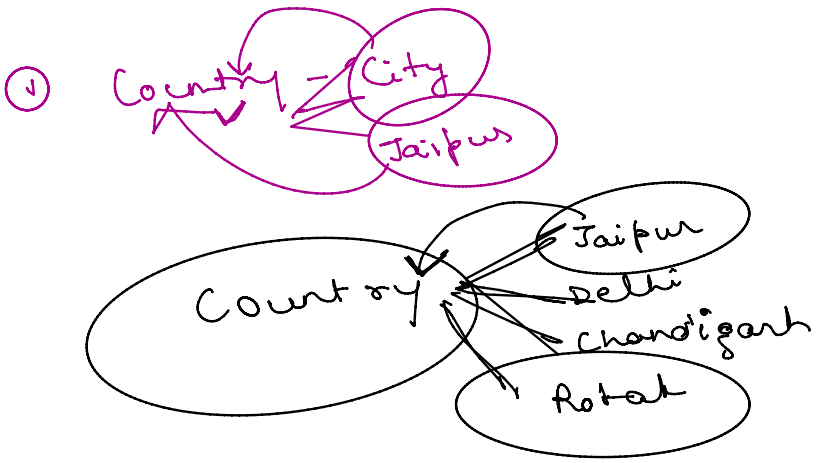
Eg 1. Country - Capital City : Delhi → India

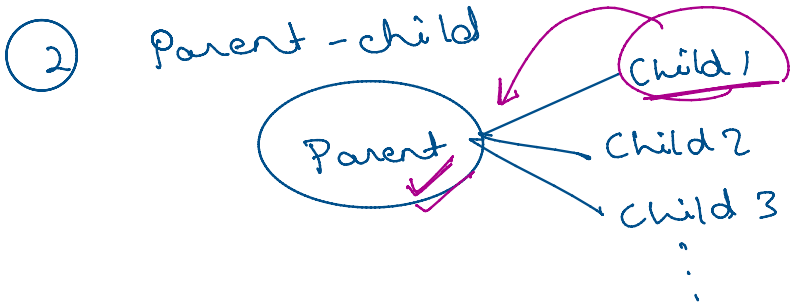
→ one to one

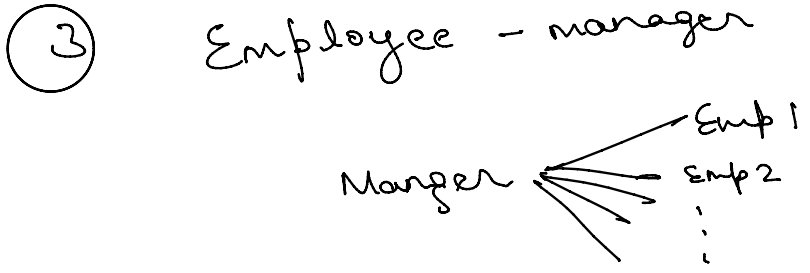


Contrast

One to Many







One to Many

Customers & Orders

Join

We Want To Store...

- A customer's first and last name
- A customer's email
- The date of the purchase
- The price of the order

We Could Use One Table...

first_name	last_name	email	order_date	amount
Boy	George	george@gmail.com	'2016/02/10'	99.99
Boy	George	george@gmail.com	'2017/11/11'	35.50
George	Michael	gm@gmail.com	'2014/12/12'	800.67
George	Michael	gm@gmail.com	'2015/01/03'	12.50
David	Bowie	david@gmail.com	NULL	NULL
Blue	Steele	blue@gmail.com	NULL	NULL

Multiple table

one table

messy

1000 colun million

difficult to update & time consuming

facebook

Customers

- customer_id
- first_name
- last_name
- email

Orders

- order_id
- order_date
- amount
- customer_id

P.K

P.K

schema

Create

foreign key

USA

India

indian embassy

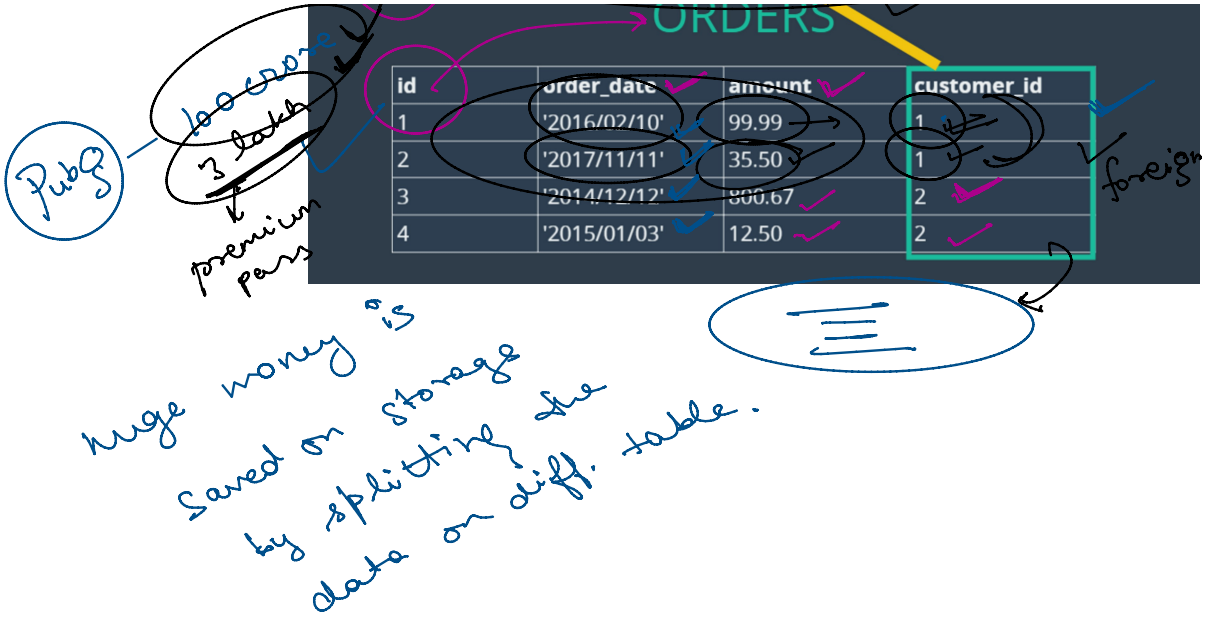
foreign

CUSTOMERS

id	first_name	last_name	email
1	Boy	George	george@gmail.com
2	George	Michael	gm@gmail.com
3	David	Bowie	david@gmail.com
4	Blue	Steele	blue@gmail.com

ORDERS

id	order_date	amount	customer_id




```sql
CREATE TABLE customers(
    id int AUTO_INCREMENT PRIMARY KEY ,
    first_name varchar(100),
    last_name varchar(100),
    email varchar(100)
);
```

Date
YYYY-MM-DD

```sql
CREATE TABLE orders(
    id int AUTO_INCREMENT PRIMARY KEY ,
    order_date DATE,
    amount DECIMAL(8 , 2),
    customer_id int,
    FOREIGN KEY(customer_id) REFERENCES customers(id)
);
```

decimal (8, 2)
scale
precision

15.29 → decimal

Shirts
99%
char
limited set of char
faster
varchar VS
varchar(100)
Krishna → 7 char
14 bytes
flexible
100 bytes
20 bytes
M/F
Y/N
char(1);
faster
char(1)
char(100);
(1)
first_name
7
2 bytes

flexible

50 → 100 bytes 20 bytes

first_nm (1)

7 → 14 bytes 2 bytes → 200 bytes

Char — insert — no. of character

" — all the time

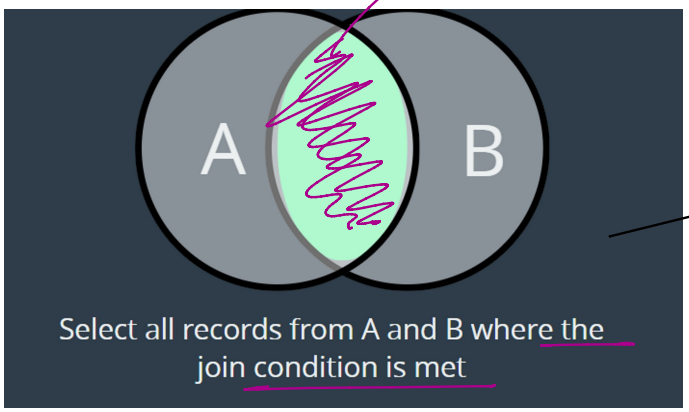
Parent → those that f.k belongs to

Child → table having f.k in it

Joins → Inner Join
Left Join
Right Join

Inner Join

→ Inner Join

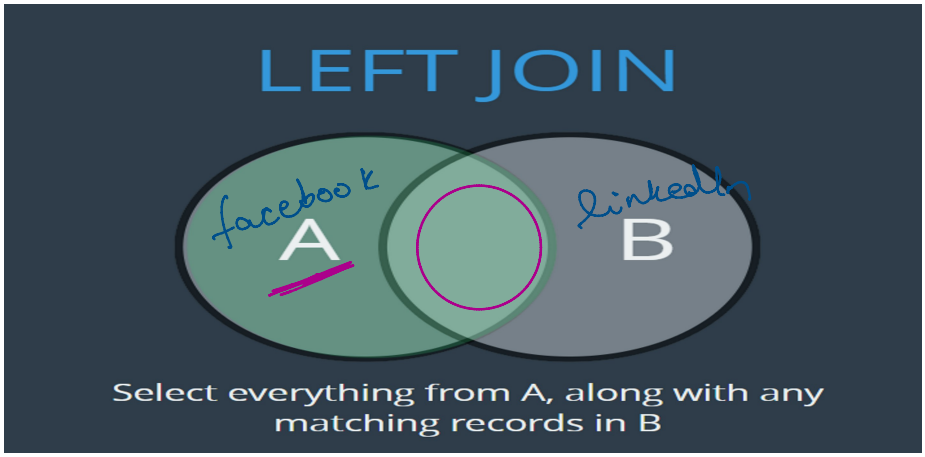


inner join
join

whenever you don't fill null in the output by inner join

left join

left join



left join

```
SELECT *
FROM facebook
LEFT JOIN linkedin
ON facebook.name = linkedin.name
```

facebook

Name	# of Friends
Matt	300
Lisa	500
Jeff	600
Sarah	400

linkedin

Name	# of connections
Matt	500
Lisa	200
Sarah	100
Louis	300

Trick

facebook and linkedin JOINed

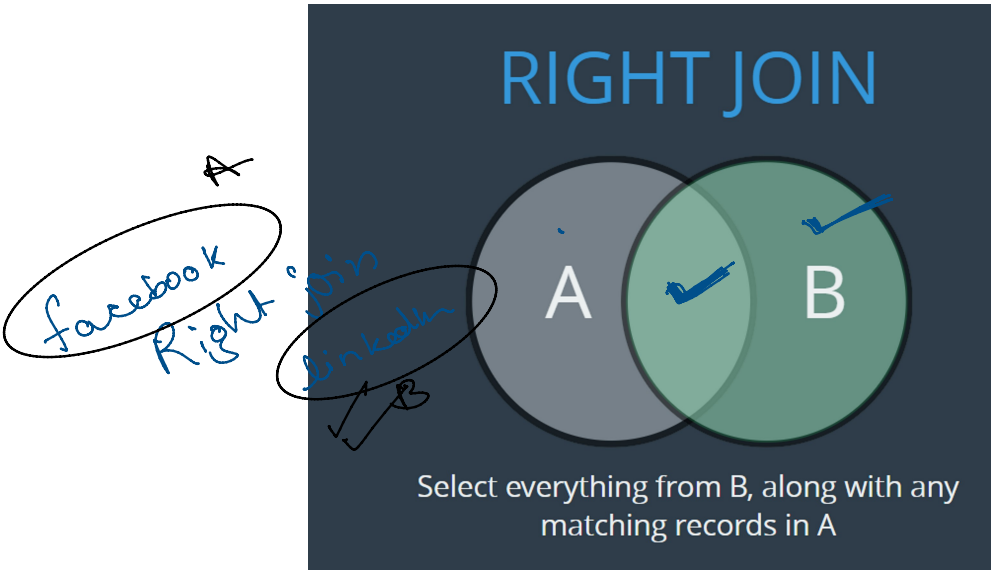
facebook.Name	facebook.# of Friends	linkedin.Name	linkedin.# of connections
Matt	300	Matt	500
Lisa	500	Lisa	200
Jeff	600	*Null*	*Null*
Sarah	400	Sarah	100

null

if null

wherever you find null value on right side it simply mean we have used left join

Right Join



facebook
Right join
linkedin

```
SELECT *
FROM facebook
JOIN linkedin
ON facebook.name = linkedin.name
```

facebook

Name	# of Friends
Matt	300
Lisa	500
Jeff	600
Sarah	400

linkedin

Name	# of connections
Matt	500
Lisa	200
Sarah	100
Louis	300

4

facebook and linkedin JOINed

facebook.Name	facebook.# of Friends	linkedin.Name	linkedin.# of connections
Matt	300	Matt	500
Lisa	500	Lisa	200
Sarah	400	Sarah	100
Null	Null	Louis	300

Trick

→ wherever you find null on the left side that means you are using right join

WRITE THIS SCHEMA

STUDENTS

- id
- first_name

PAPERS

- title
- grade
- student_id

p.k

f.k

INSERT THIS DATA

(COPY AND PASTE IT)

```
INSERT INTO students (first_name) VALUES
('Caleb'), ('Samantha'), ('Raj'), ('Carlos'), ('Lisa');

INSERT INTO papers (student_id, title, grade ) VALUES
(1, 'My First Book Report', 60),
(1, 'My Second Book Report', 75),
(2, 'Russian Lit Through The Ages', 94),
(2, 'De Montaigne and The Art of The Essay', 98),
(4, 'Borges and Magical Realism', 89);
```

on

1 PRINT THIS

inner

```
+------------+-------------------------------------+-------+
| first_name | title                               | grade |
```

2 PRINT THIS

left

```
+------------+-------------------------------------+-------+
| first_name | title                               | grade |
```

first_name	title	grade
Samantha	De Montaigne and The Art of The Essay	98
Samantha	Russian Lit Through The Ages	94
Carlos	Borges and Magical Realism	89
Caleb	My Second Book Report	75
Caleb	My First Book Report	60

first_name	title	grade
Caleb	My First Book Report	60
Caleb	My Second Book Report	75
Samantha	Russian Lit Through The Ages	94
Samantha	De Montaigne and The Art of The Essay	98
Raj	NULL	NULL
Carlos	Borges and Magical Realism	89
Lisa	NULL	NULL

3 PRINT THIS

first_name	title	grade
Caleb	My First Book Report	60
Caleb	My Second Book Report	75
Samantha	Russian Lit Through The Ages	94
Samantha	De Montaigne and The Art of The Essay	98
Raj	MISSING	0
Carlos	Borges and Magical Realism	89
Lisa	MISSING	0

4 PRINT THIS

first_name	average
Samantha	96.0000
Carlos	89.0000
Caleb	67.5000
Raj	0
Lisa	0

5 PRINT THIS

first_name	average	passing_status
Samantha	96.0000	PASSING
Carlos	89.0000	PASSING
Caleb	67.5000	FAILING
Raj	0	FAILING
Lisa	0	FAILING

```
SELECT first_name ,
ifnull(avg(grade) , 0) as 'Average'
from students LEFT JOIN
papers on students.id = papers.student_id
GROUP BY students.id
ORDER BY Average DESC;
```

Students

id	first_name
1	Caleb
2	Samantha
3	Raj
4	Carlos
5	Lisa

Papers

title	grade	student_id
My First Book Report	60	1
My Second Book Report	75	1
Russian Lit Through The Ages	94	2
De Montaigne and The Art of The Essay	98	2
Borges and Magical Realism	89	4

```sql
SELECT first_name ,
ifnull(avg(grade) , 0) as 'Average' ,
    CASE
        WHEN AVG(grade) >= 75 THEN 'PASSING'
        ELSE 'FAILING'
    END as passing_status
from students LEFT JOIN
papers on students.id = papers.student_id
GROUP BY students.id
ORDER BY Average DESC;
```