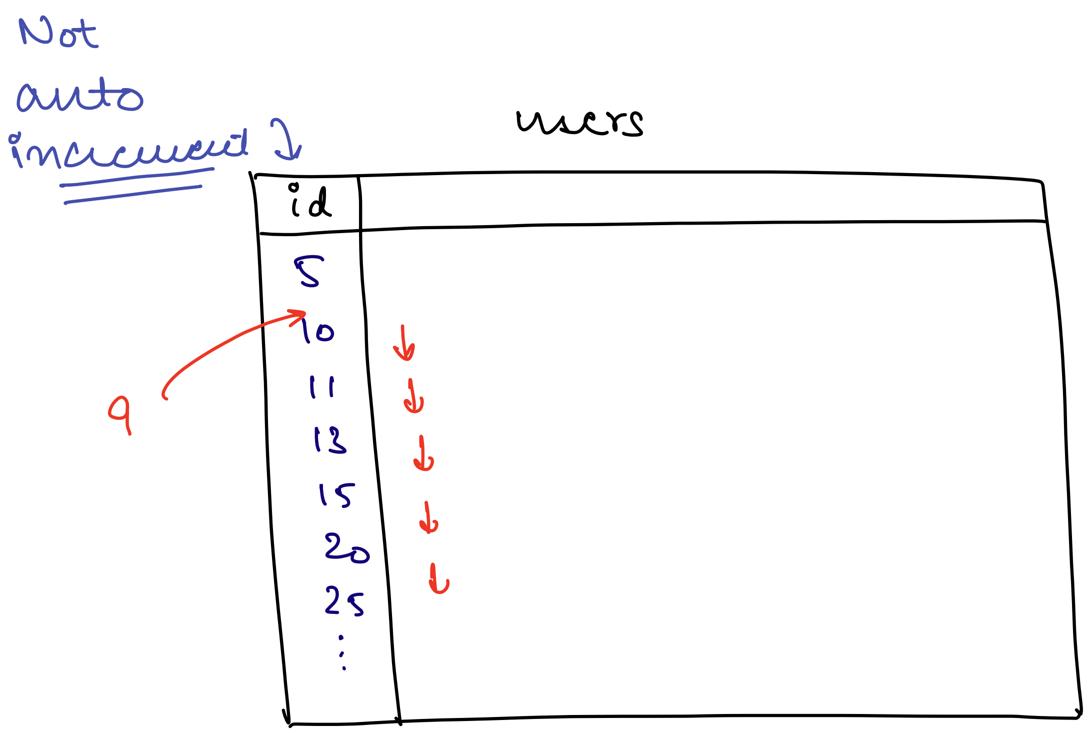
⇒



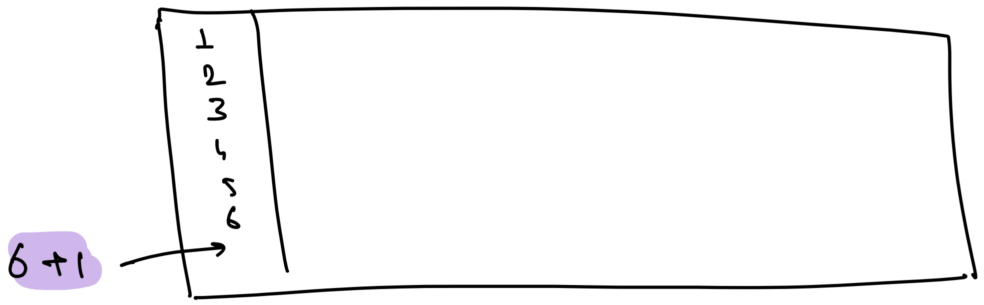
⇒ DB by default sorts the data based on the PK

⇒ Index is also created by default.

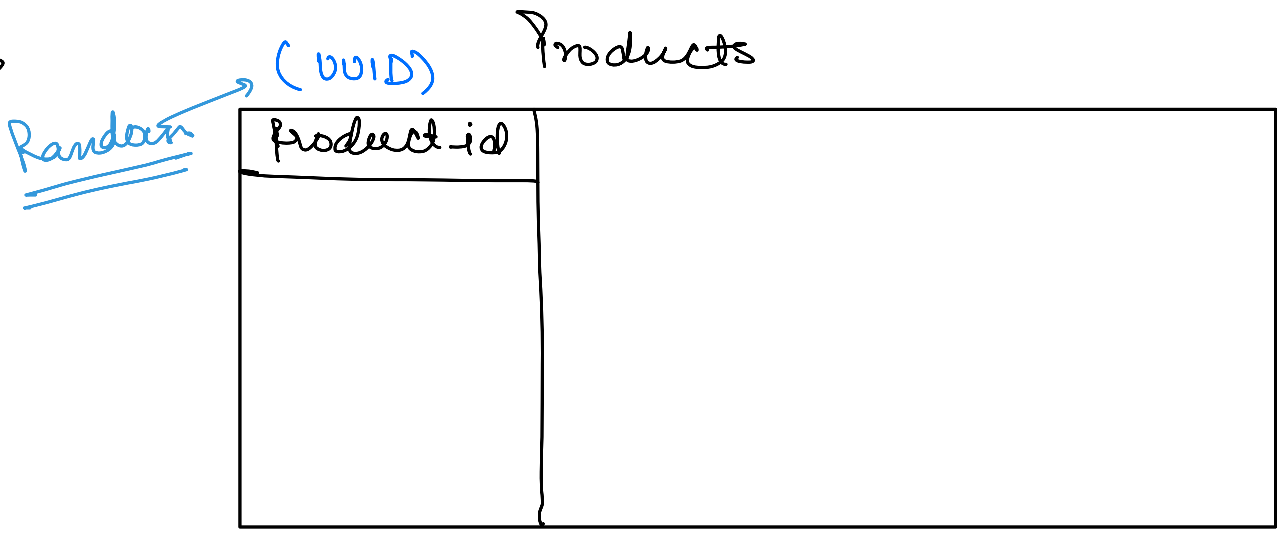
⇒ New id = $\boxed{9}$

⇒ To insert a row in between, we'll have to shift the data

⇒ If database is maintaining auto increment over PK, then shifting would not be required.

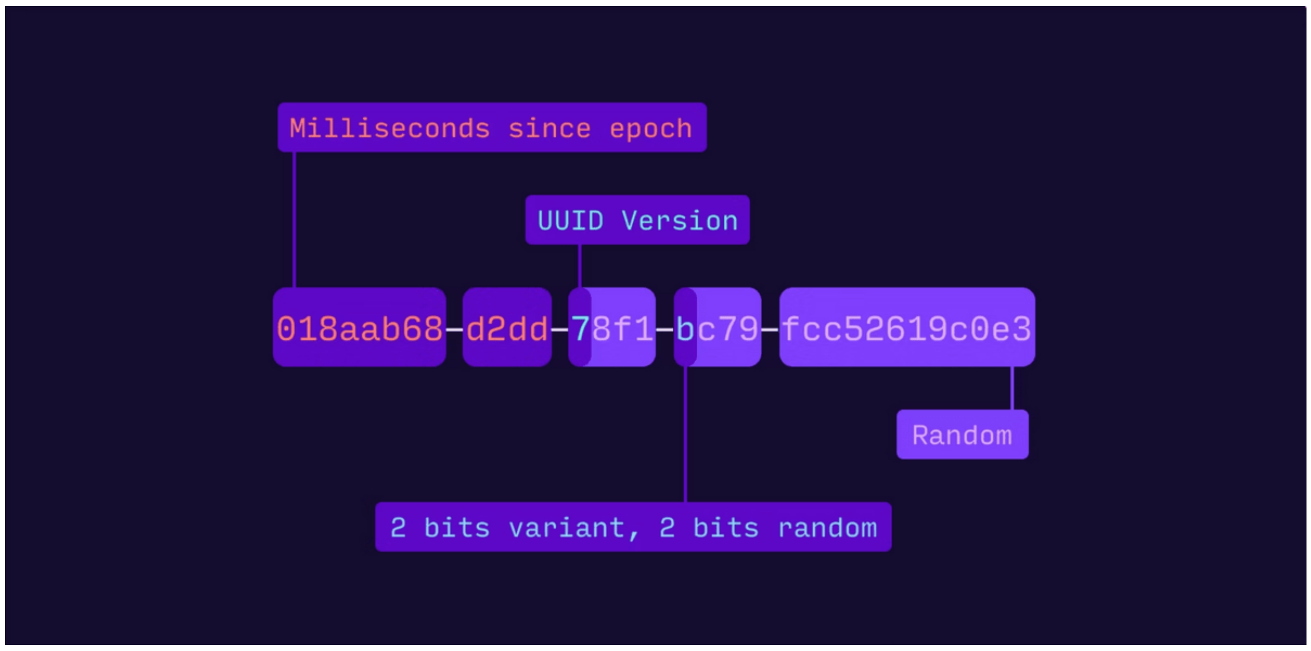


⇒



⇒ UUID V7 makes sure that the newly generated UUID is greater than the previously generated UUID

⇒ timestamp.

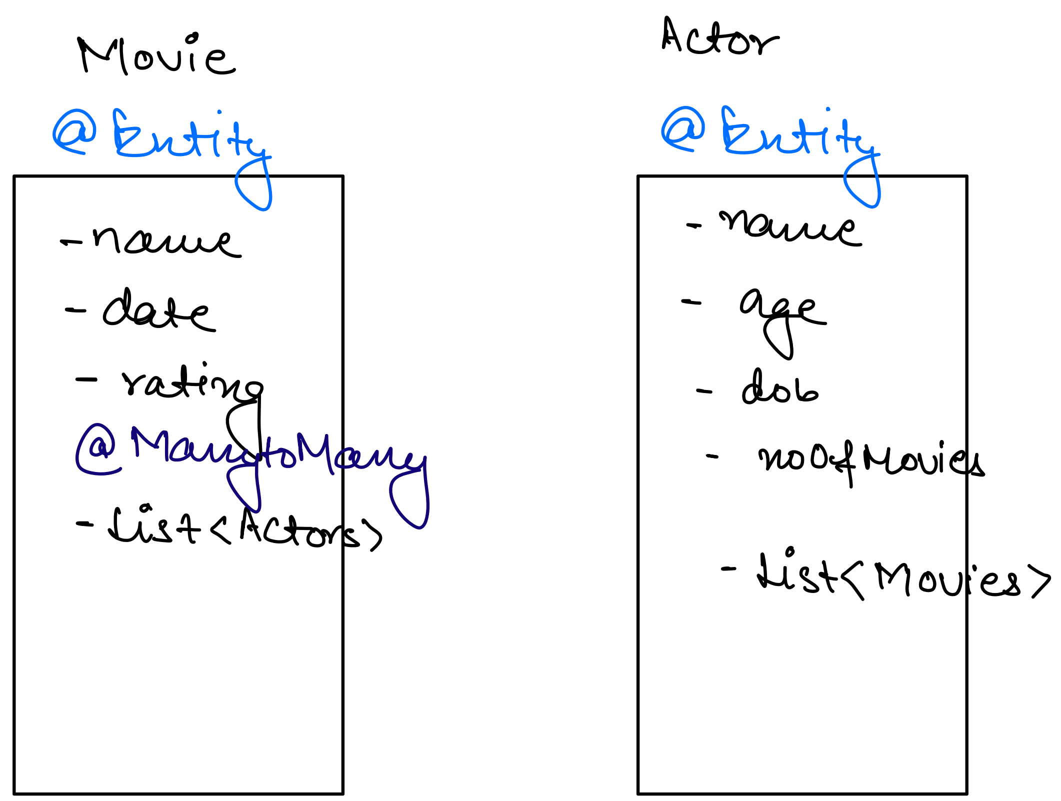

Milliseconds since epoch
UUID Version
018aab68-d2dd-78f1-bc79-fcc52619c0e3
Random
2 bits variant, 2 bits random

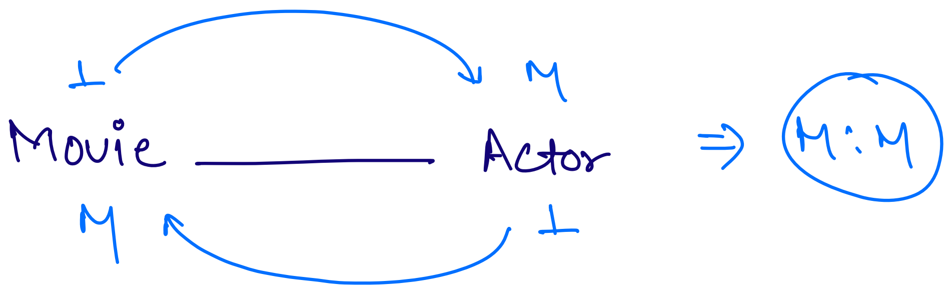
⇒ Timestamp.

↳ Milliseconds passed since 01/01/1970.

LLD Class.

Class Diagram. ⇒ Schema Design.





movies

id	name	date	rating

actors

id	name	age	noofmovies

movie_actors

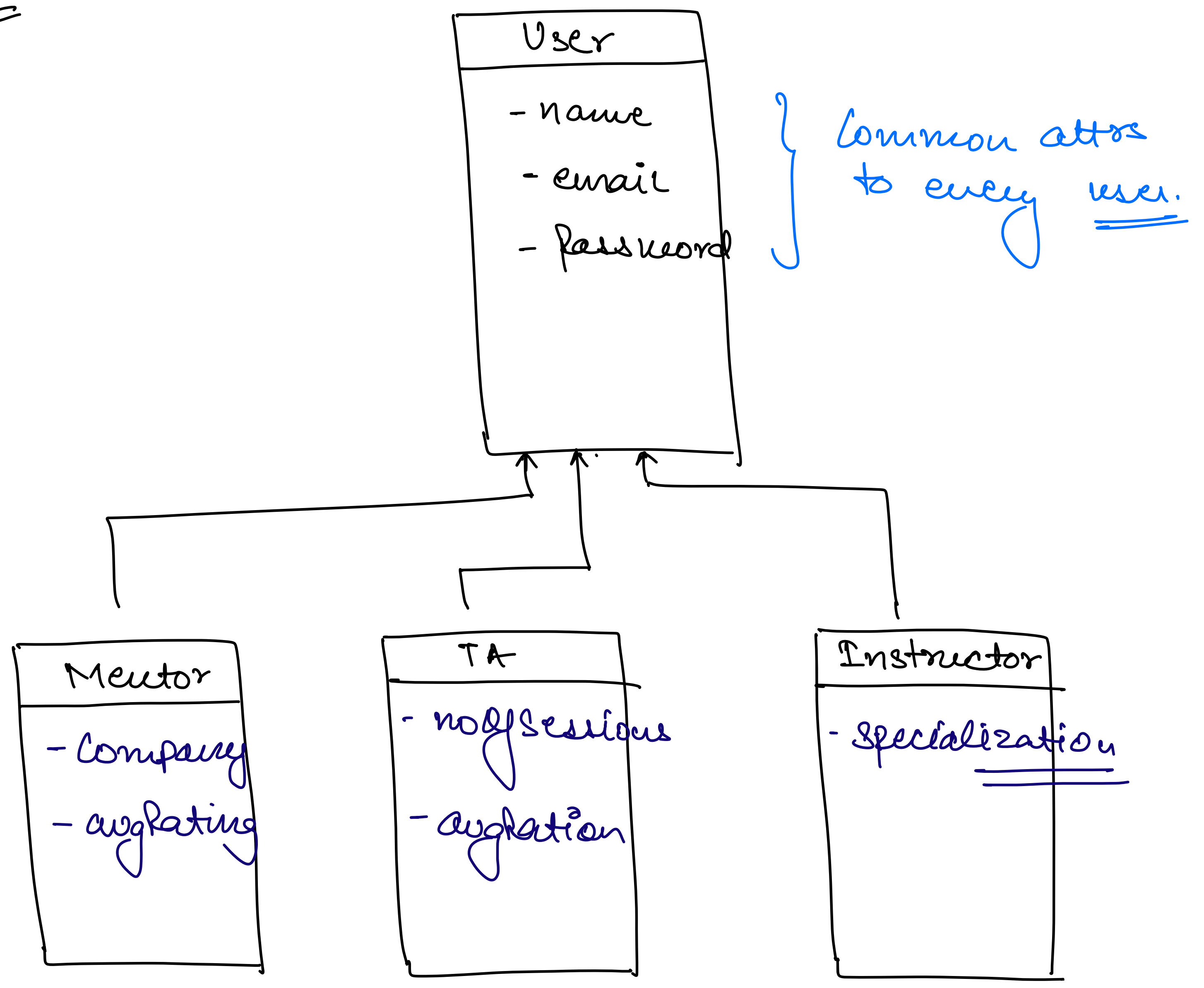
movie_id	actor_id

⇐ Mapping table.

Q. Given a movie_id, get me details of all actors in that movie.

⇒ join mapping table & actors table.

Ex:



⇒ Represent the above inheritance relation in Database.

4 ways to represent Inheritance relations in Databases.

① MappedSuperClass.

⇒ When there will be No objects of Parent Class.

⇒ Technically parent class is an abstract class.

Approach.

1) No table for parent class.

2) 1 table for each child class with their own attributes as well as their parent's attributes.

mentors

name	email	Password	Company	avgRating

tas

name	email	Password	noOfSess	avgRating

instructors

name	email	Password	Specialization

Q. Get email for every user.

```
select email from mentors
        UNION
select email from tas
        UNION
select email from instructors.
```

⇒ lot of queries.

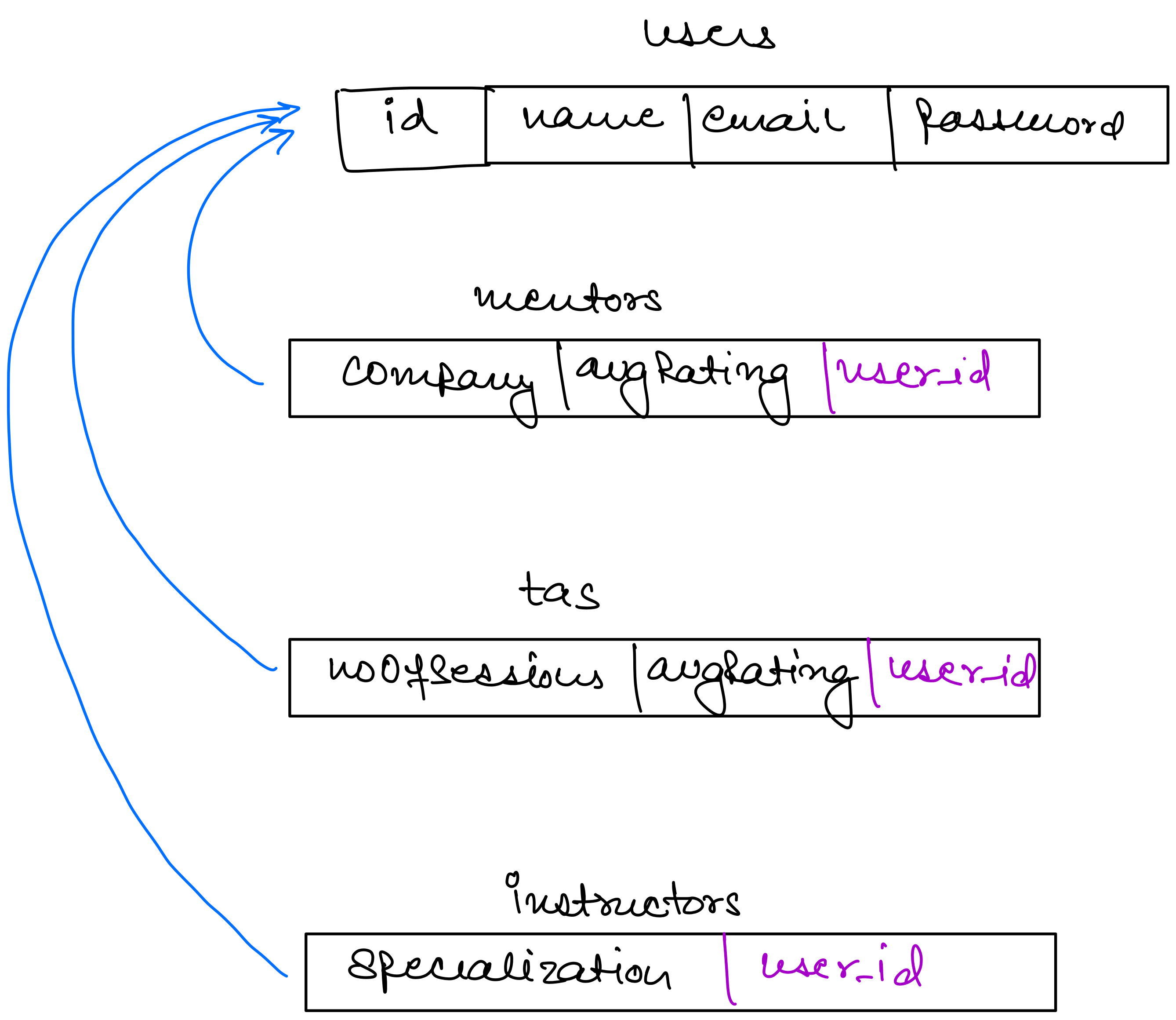
⇒ Not a very good approach as if we have multiple type of users then there's a chance of miss.

② JOINED TABLE. (Most frequently used).

→ Every data sort objects of parent class, we'll have in the parent table.

→ For each child class also, we'll create a table with only their specific attrs.

→ We'll get parent's (common) attrs for each table via foreign key.



Q. Get email for every user.

⇒ select email from users. ✓

③ Table per Class.

→ It is exactly same as mapped super class.

→ We'll also have table for parent class.

→ It is used when we can object for parent class as well.

→ Table for each class will be with its own attributes as well it's parent attrs.

users

name	email	password

⇐ Contains only users data.

mentors

name	email	password	Company	avgRating

tas

name	email	password	noofsess	avgRating

instructors

name	email	password	specialization

users

name	email	password
Deepak	deepak@scaler.com	1234

4 Single Table.

→ Create one table with all the columns across all the child classes.

users.

user_type	name	email	password	company	avgRating	no	avg	Specia
Mentor	deepak	deepak@scaler.c	—	Amazon	4.8	Null	Null	Null
TA	Suraj.	—	—	NULL	NULL	100	4.9	Null

→ Lot of Nulls. (Sparse table)

→ Space wastage

→ Add one new column (user_type) to recognise the type of user.

→ If a user can have multiple roles.

4 <u>ways.</u>

1) Mapped Super Class

2) Joined table

3) Table Per Class

4) Single <u>Table.</u>

——— * ———