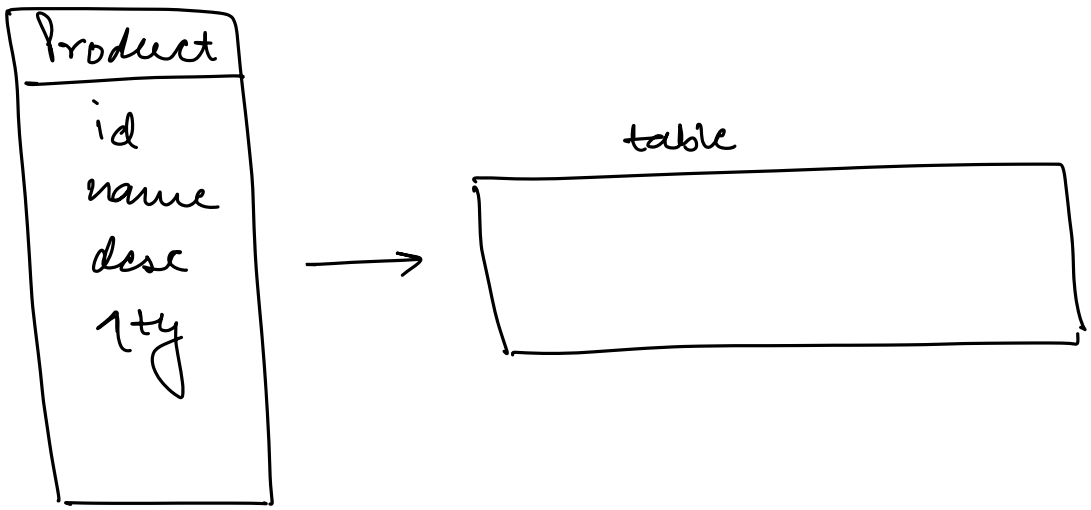
Agenda.

→ Represent inheritance in Database.

→ Setup our DB

→ Integerate DB with ProductService.



Extending relationship

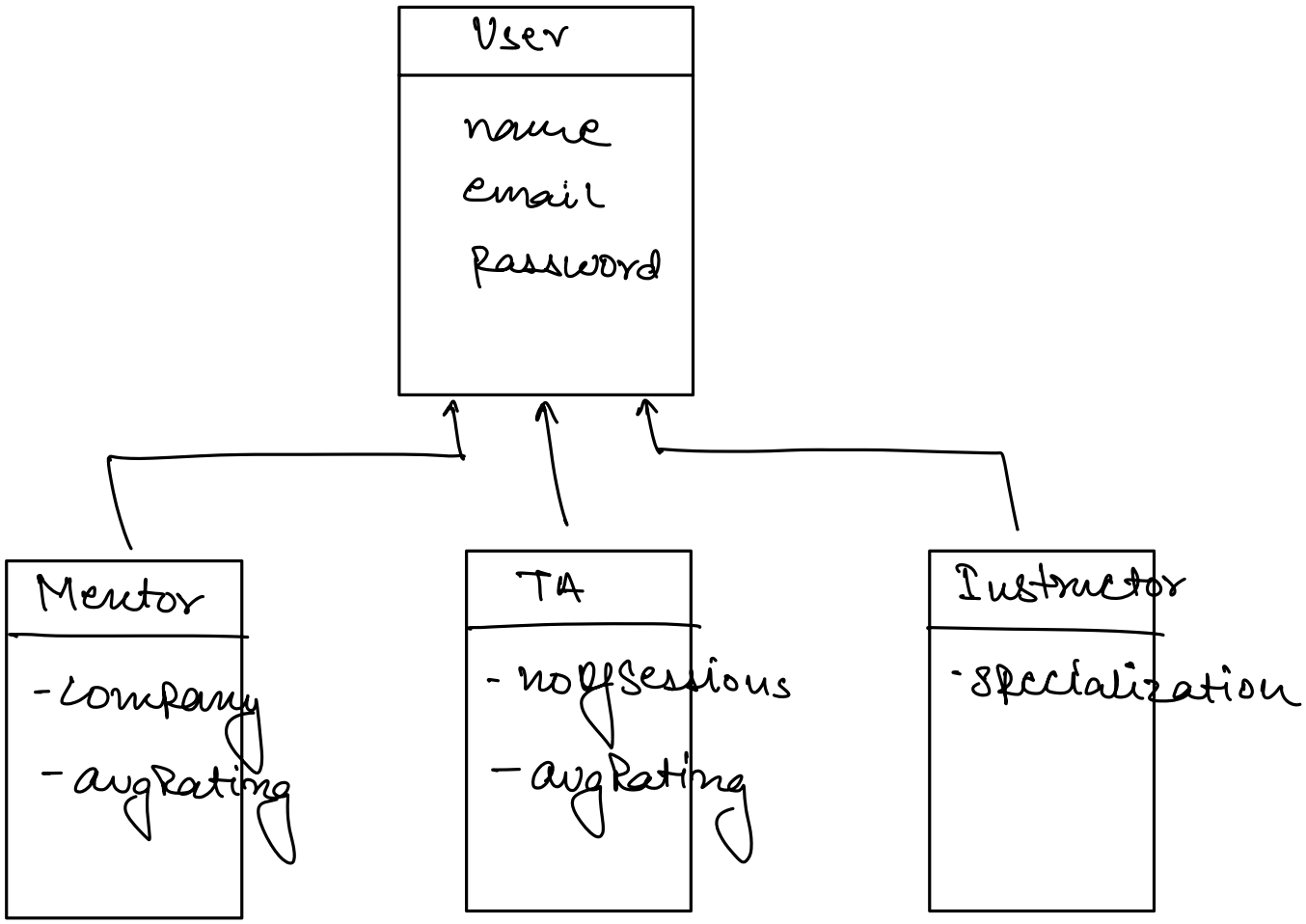
⇒ How to represent Inheritance in DB.

When we use mapped super class ->

When a parent class is not a real entity and we are using parent class to encapsulate or store the common atrributes of child class

↳ 4 ways.

- Mapped Super Class
- JOINED Table
- Table Per Class
- Single Table class.



Store the data of instructors, ta & mentors in the DB, how?

1. MappedSuperClass.

Here we donot create parent class

=> When there's No object of parent class.

=> Parent class can be marked as an Abstract Class.

An abstract class is a class that defines a base class for other classes to inherit from.
To declare a class as abstract, add the keyword abstract before the class definition.
Subclasses inherit from abstract classes and provide implementations for the abstract methods.

In mapped superclass, we donot have object of parent class
One table for each child is created in Parent class

@MappedSuperclass indicates that a class is a superclass and is not associated with a specific database table, but its fields (or properties) can be inherited by child entity classes that are associated with tables. This helps avoid code duplication and ensures a cleaner object model structure.

Approach.

1) No table for parent class

2) One table for each of the class with their own attributes as well as attributes from parent class.

Child class will get all attributes from parent class -> name, attributes & password

Here parent class(mapped superclass) is not an real entity. It will not have atrributes of its own but only common atrributes from its child classesx`

mentors

Company	avgRating	name	email	Password

ta

noOfSessions	avgRating	name	email	password

instructor

specialization	name	email	Passwor

Q. Get email of all the users.

```
Select email from mentors
UNION
Select email from ta
UNION
```

Select email from instructors

(2) Joined Table. => 99.99% (Best Soln)

→ Every data wrt objects of parent class, we'll store in the parent table.

→ for each sub class also, we'll create a table with only their own attributes.

→ We'll get parent class attrs in child classes via foreign key.

Here we will have parent table -> users table
In this users table we have name, email & pass for Mentors, TA and Instructor and we will use Foreign Key on id to get the attributes of parent class into child class

users

id	name	email	password

mentor

company	avgRating	user_id

ta

wolfSessions	avgRating	user_id

instructor

specialization	user_id

Q. Get email ids of all the Mentors.

⇒ JOIN Mentor & User.

Q. Get email of all the users.

↳ Single query

select email from users.

Usecase

⇒ There can be some user which is neither ta, nor Instructor & nor Mentor?

↳ MappedSuperClass ✗

↳ Joined Table.

③ Table per Class.

→ Exactly similar to MappedSuperClass, only difference, here we'll also create table for parent class as well.

→ table for each class will have their own attributes as well as parent class attributes.

users

name	email	password
Suraj	—	—

Here in users class we will store name, email id and password for only those users which are not mentors, not TAs and not instructors

Here user class is a real entity

mentors

Company	avgRating	name	email	Password

If this Suraj has come to users table and later on become mentor, TA and Instructor, his name and details will duplicated in all these 3 tables as well -> disadvantage

ta

noOfSessions	avgRating	name	email	password

instructor

specialization	name	email	Passwo

Note

→ First define the inheritance rel^n in the Codebase

→ Then identify the query pattern. & go with one of the ways.

↓

Most frequently executed query.

④ Single Table. (Worst Soln)

⇒ Create one table with all the columns across the tables.

⇒ Add one extra column user_type to recognize the type of user.

Users-

name	email	password	company	avgRating	noOfSessions	avg	specialization
xyz	—	—	—	—	NULL	NULL	NULL
Peer	—	—	NULL	NULL	NULL	NULL	LLD

⇒ Too many Nulls. (Sparse table)

⇒ Waste of space.

⇒ If a user can have multiple roles.

⇒ 4 ways to represent inheritance in DB.

- Mapped Super Class. ⇒ No object for Parent Class.
- Joined Table. ⇒ Table for parent class + Child tables will refer the parent class via FK.
- Table per Class

  Exactly similar to Mapped Super Class, but table for parent class.
- Single Table. ✓
