SCHEMA DESIGN :

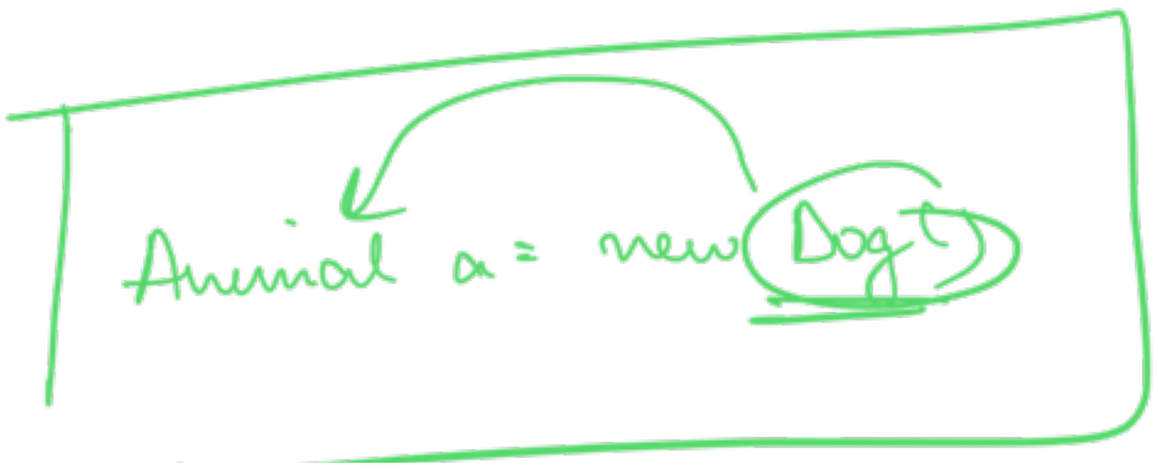
⇒ How will the database look

⇒ What will be diff tables in DB

→ diff columns inside them

UML Diagrams

Class Diagram : Classes in the system and rel^n b/w them



↳ Inheritance (IS-A)

↳ Association (HAS-A)

// ↳ Composition (Strong)

Weak

Aggregation

when you have attributes

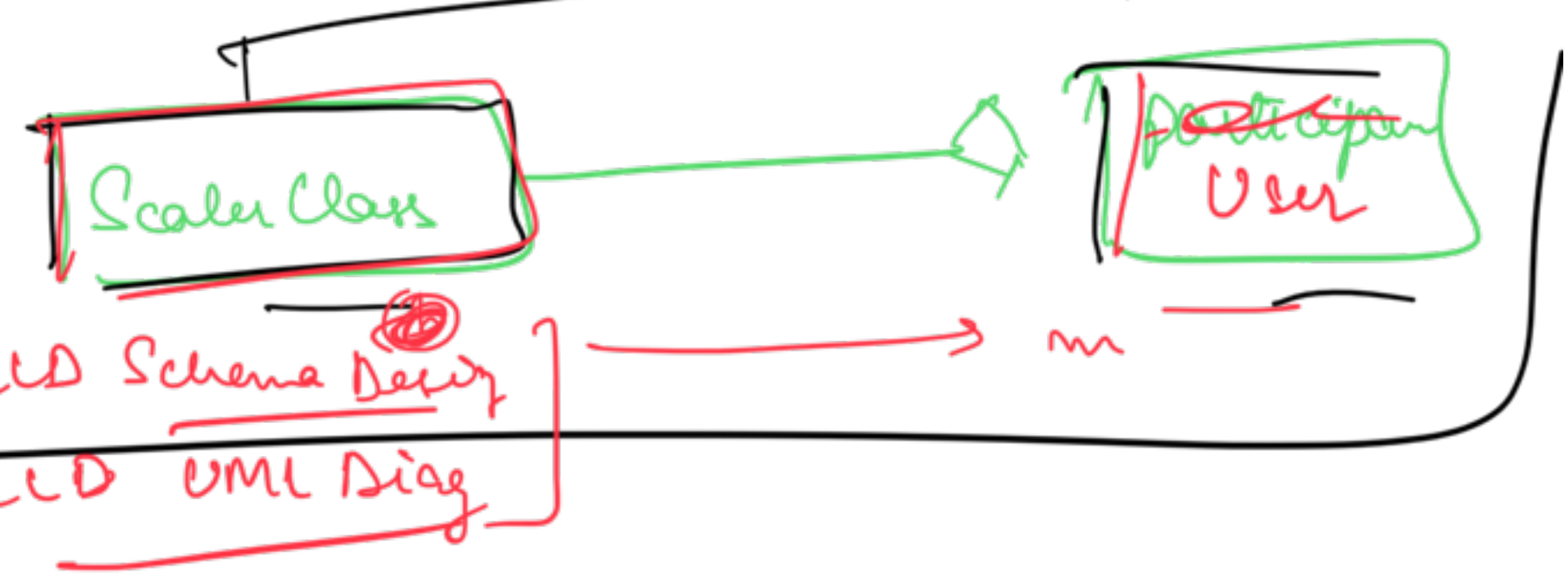
```
ScalerClass {

    List<User> participants;

}
```

1 : m

m : m



LLD Schema Design

LLD UML Diag

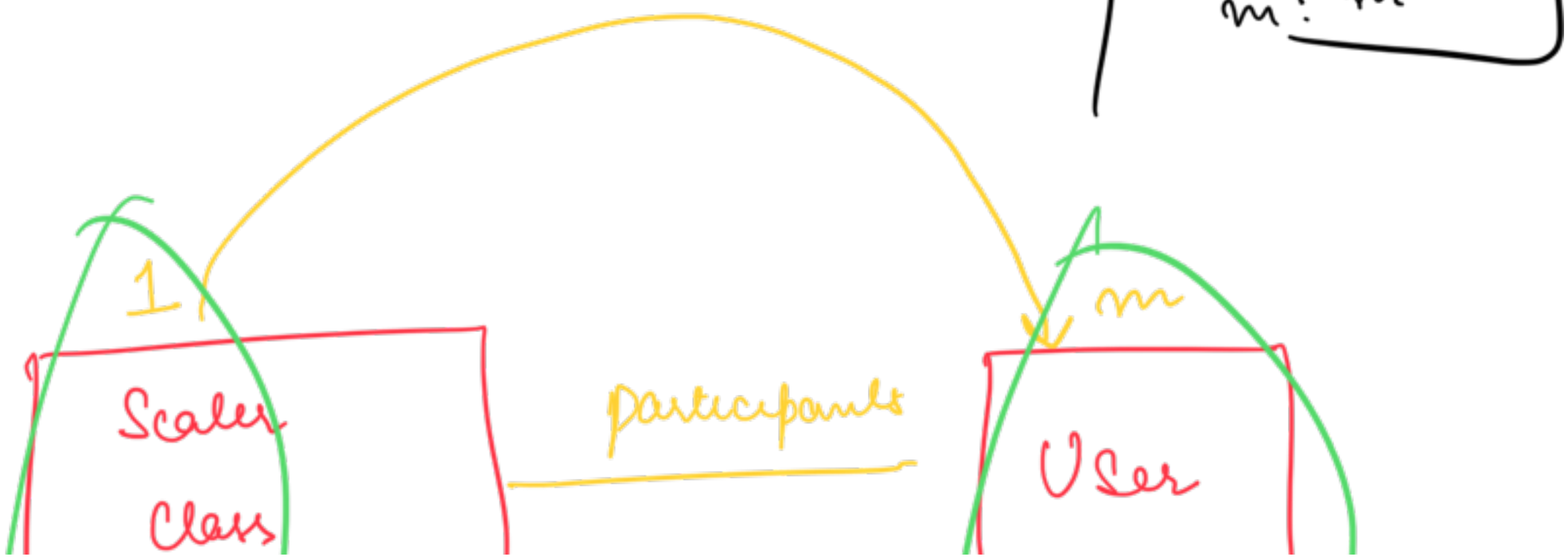
Cardinality

when 2 classes A and B have a

HAS-A Cardinality talks about

How Many of A associated to How Many of B

- 1 : 1
- 1 : m
- m : 1
- m : m



m

1

1

m

Tm : m

1

M

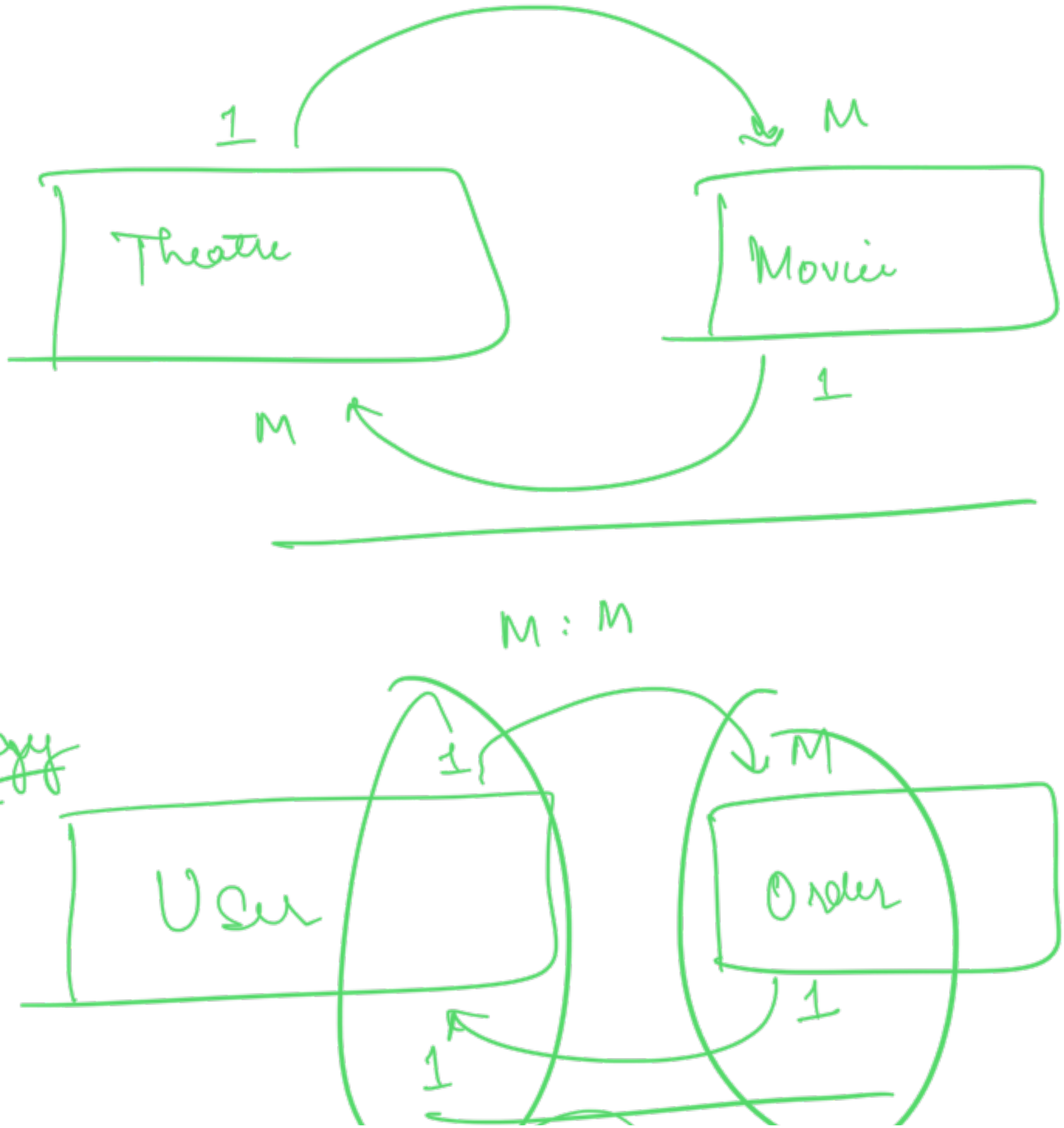
Asg ~

Question

m

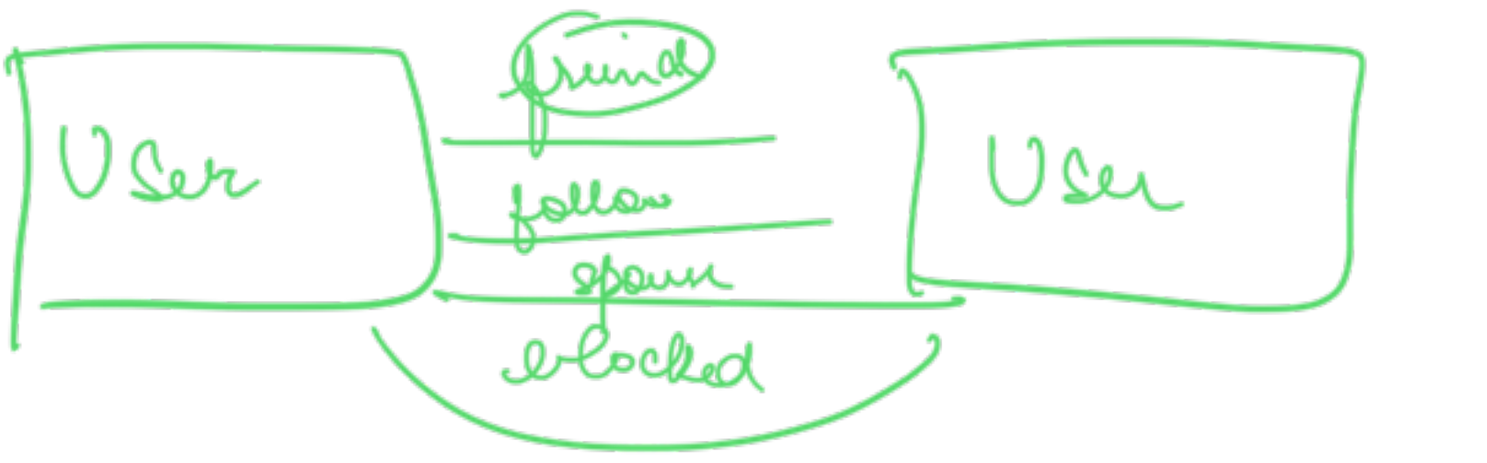
1

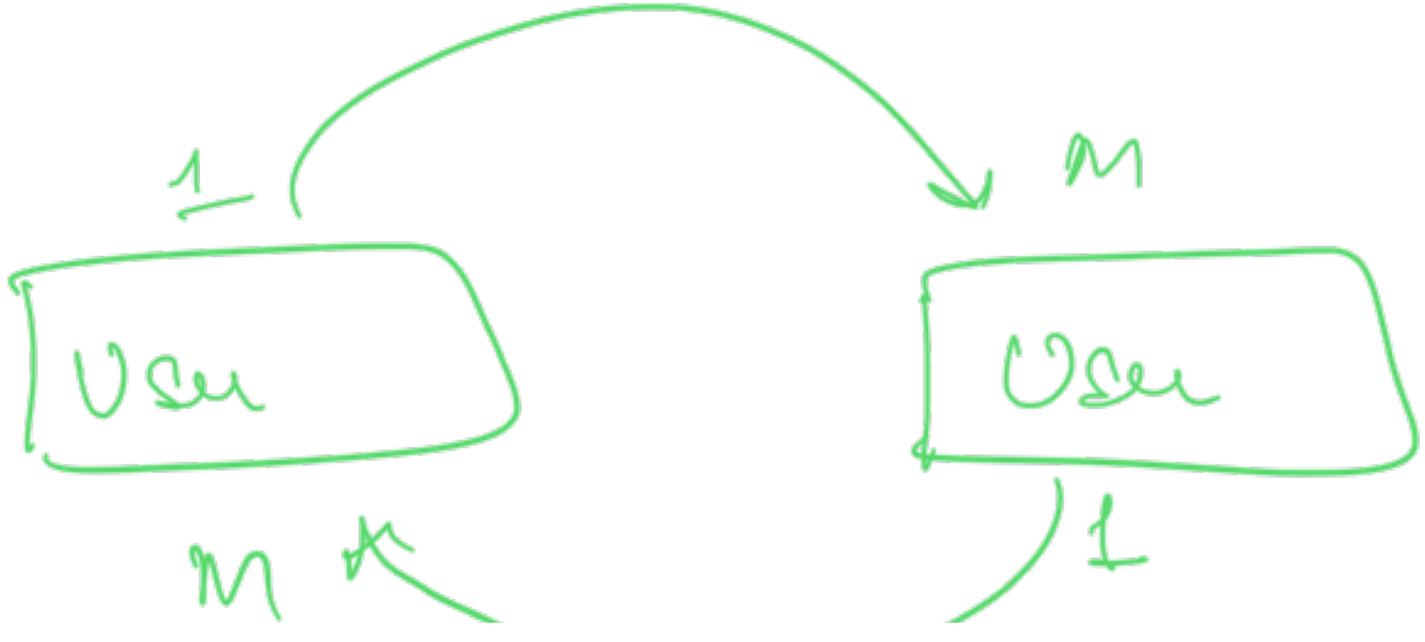
M : M


1
M
Theatre
Movie
M
1
M : M
Swiggy
User
Order
1
M
1
1

1 : M

FB





M : M

How to always get the correct answer for Cardinality :

(1) Clear what rel^n are we talking about

(2) 2 Step approach

?

1

M : M

1

M

Passenger

Ticket

1

M

M : M

BMR

1
M
Scaler
Offer
1
1
1 : M
1
1
Aadhaar
Phone
M
1

M : 1

1 : 1

Schema Design

1. What is Schema Design =
2. how to find what tables will be there in my Schema_
3. How to store relationships in the DB

Next class

④ Coding a real schema

→ We were never storing any data in my system till now

⇒ In real world systems → do need to store data ⇒ persistence

↓

Database

Assume SQL DB

Class : Blueprint of an entity

→ how every object of that type will look

Schema : Blueprint of a DB

→ how will a SQL DB to persist current req look

→ what will be diff table

→ what will be columns inside

those were

Start with a class diagram

⇓

Do the Schema design

for every class that you see in the class diagram - create an associated table.

Case Study : Design DB of Scaler

V1

Use Cases

1. Student should be able to login (enroll, pay)
2. Students should be able to see their profile which contain Name, address and phone number

Loop 1

Approach → Do the class diagram

```
class Student {
    Name
```

addr

phoNo.

email -

password

2

Step 2 Map every class as it is to a table

every attr of that class becomes a column of the table

Students

id	Name	addr	phone No	email	password
1					
2					

Primary key

uniquely

every studen

⇒ int ids

||

Auto increment

→ All previous req are still there

→ Scaler now has multiple course

→ Every student can opt for ≤ 1 Course

UPTO 1

→ Every course has a name

Sol-1

Students

id	Name	address	phNo	email	password	course-Name	St-de
1	Naman	—	—	—	—	Aug 21 Jun	Oct 21 A
2	Manik	—	—	—	—	Aug 21 Jun	
3	Jayesh	—	—	—	—	Sep 21 Jun	

Cons

(1) Course Name is repeating
↳ wasting Space

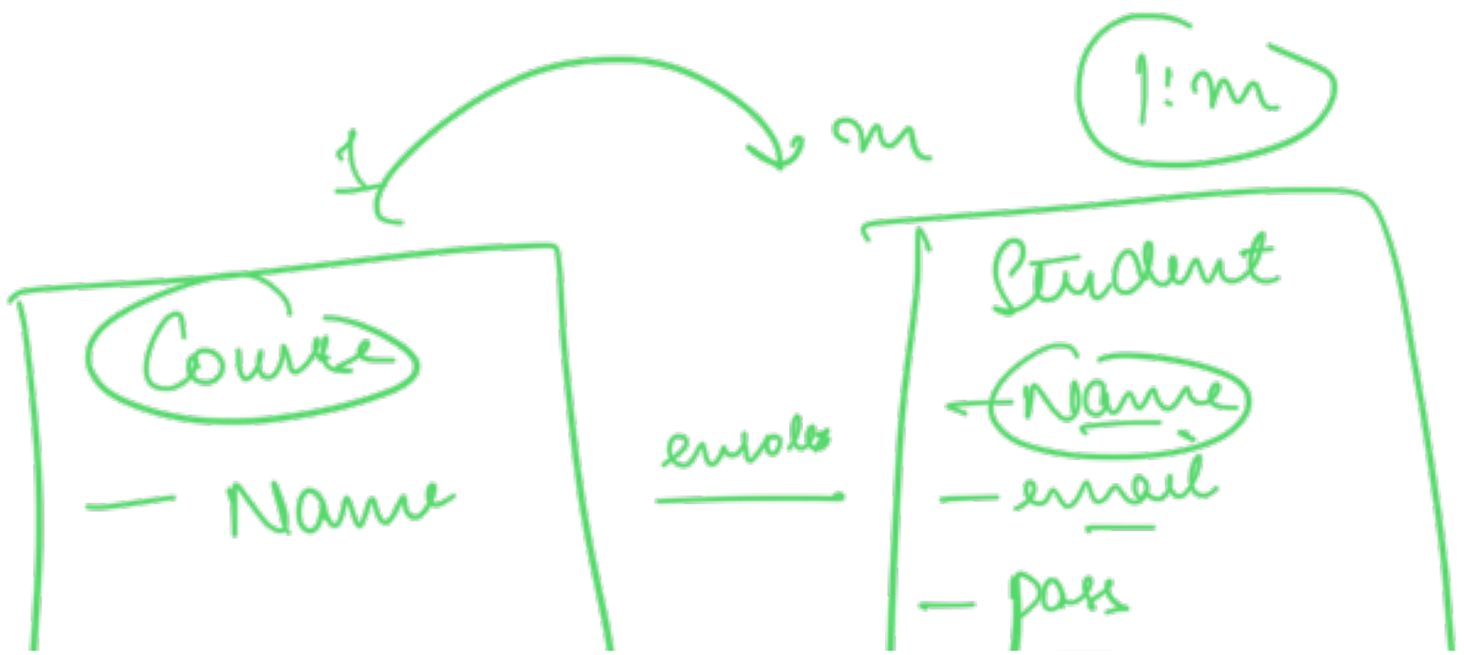
(2) Update a course Name
Aug 21 Jun → Oct 21 Adv

③ what if there are other/more attr
of course in future

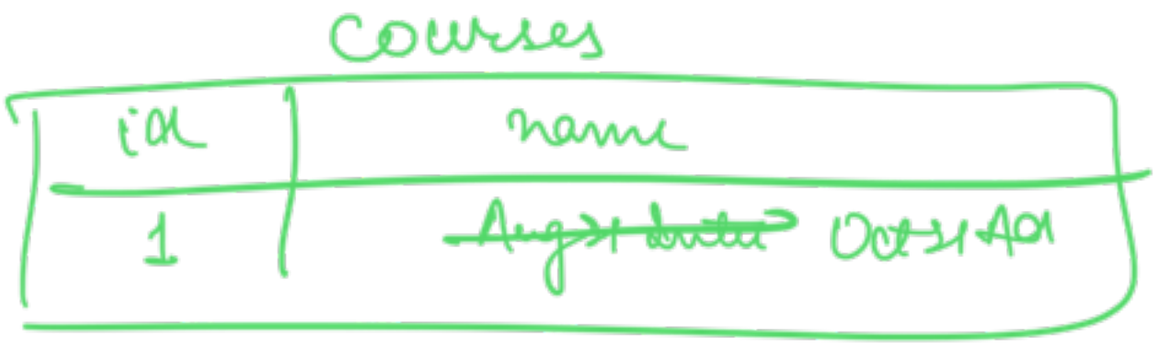
④ Let's today Scala creates a new cours
feb 22 Adv

⇒ Can't store a course enrolls 1 Student

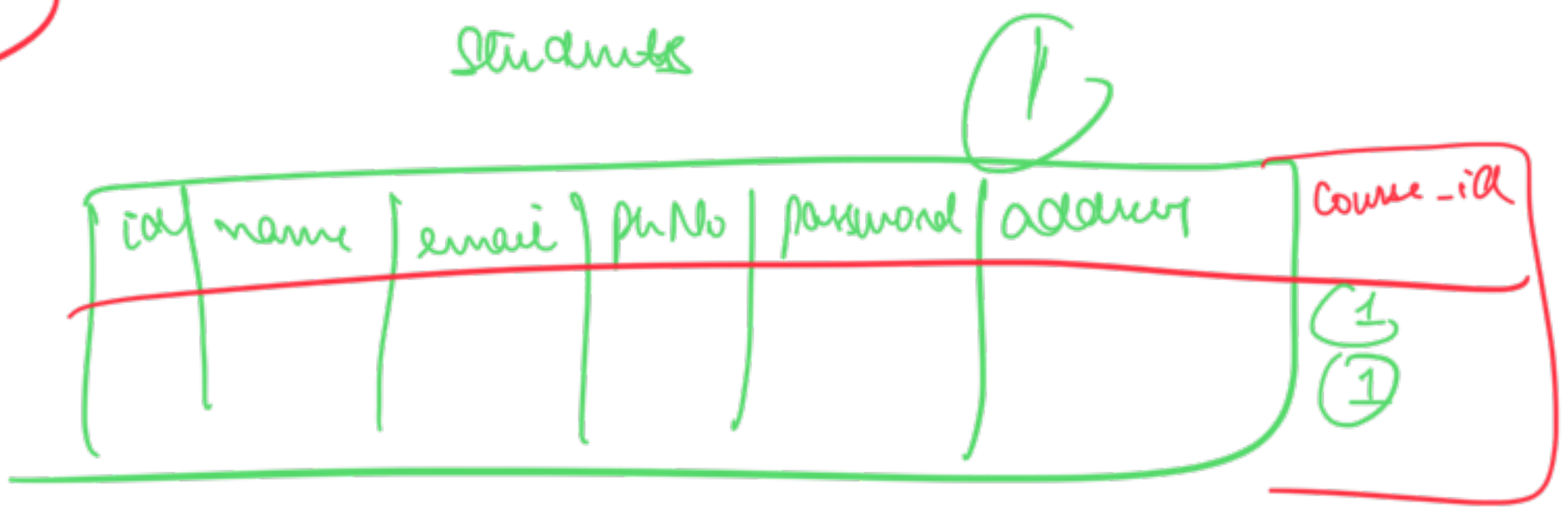
Soln 2




-phoneNo
- addr
1
1


Courses
id
name
1


1
1:M
1


Students
1
id
name
email
phNo
password
address
Course_id
1
1

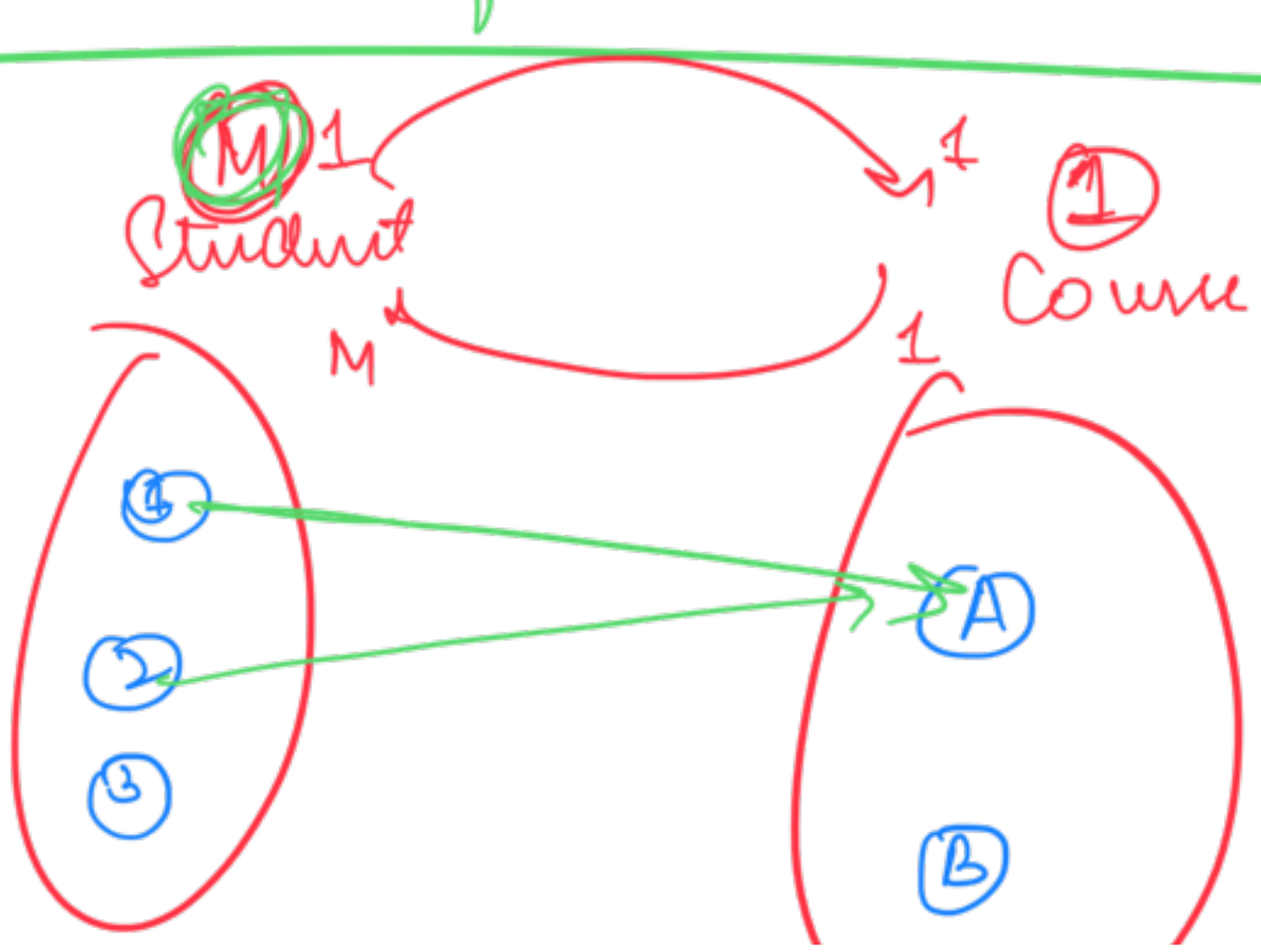
Once rep all the entities

→ find rel^n b/w all pairs of entities

Whenever we have 1: m rel^n

we rep it as a column in the table

of m side



In Relational DB

every column should have an atomic value

✓ int

double

boolean

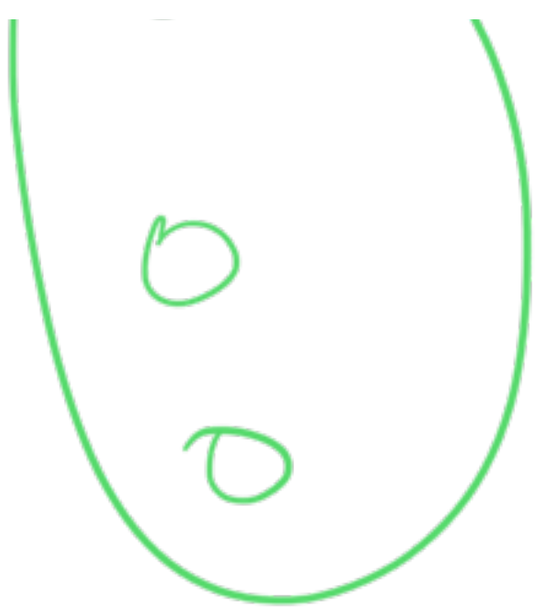
X

array

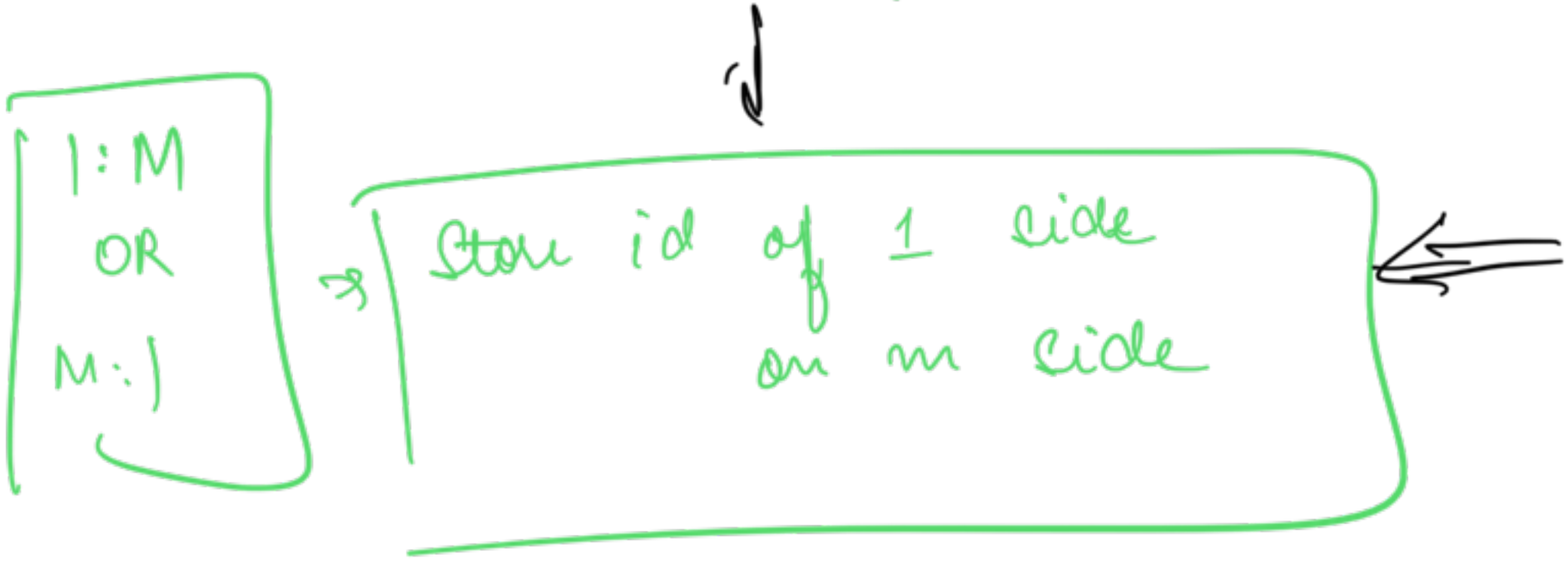
list

dictionary

Student → only 1 value to store for a student




Courses


1:M
OR
M:1
Store id of 1 side
on m side

Students

4B

Course id

20B

1 Sanath

2 Nikhil

3 Kartikeya

14 Nakul

NULL

1

1

NULL

1M

⇒ If I am storing so many null ⇒ wasting space

10 lakh

10 k

A table with a particular column having a lot of NULL ⇒ Sparse table

10L - 10K NULL in my system

4B * 10L

4B 10^6

4MB

$\frac{10^6}{10^3} \approx 10^3$

Student

sid	Name	addr	email	ph No	password

Course

id	Name

Mapping Table

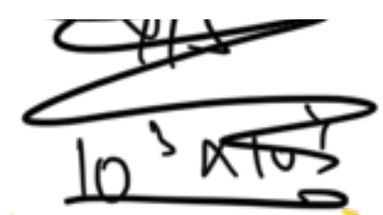
make also student

1 M

Student - course

St_id NON-NULL	C_id NON-NULL
1	3
4	3
5	1

10^3 10^1



1 : M

OR

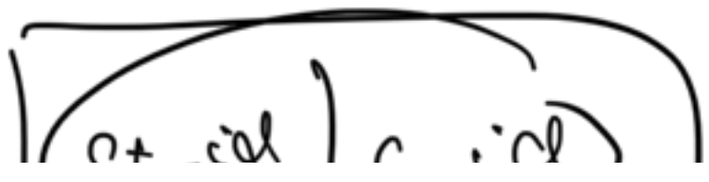
M : 1

⇒ Store the rel^{n} by adding fkey on M side

But if the relation is sparse

⇒ Separate Mapping table

Student_course



(13) Every student will enroll for PRACTICAL 4 Courses

List Courses

Students

Course_ID

ID	Name	add	phNo	email	password	list courses
						[ ]

Relational Models are not allowed to have multivalued

attributes

Sol^n 1

Students

id	Name	addr	email	password	phNo	C1_id	C2_id	C3_id
3						1	~~4~~ 5	3	6

Cons

(1) Query

find all students who are enrolled

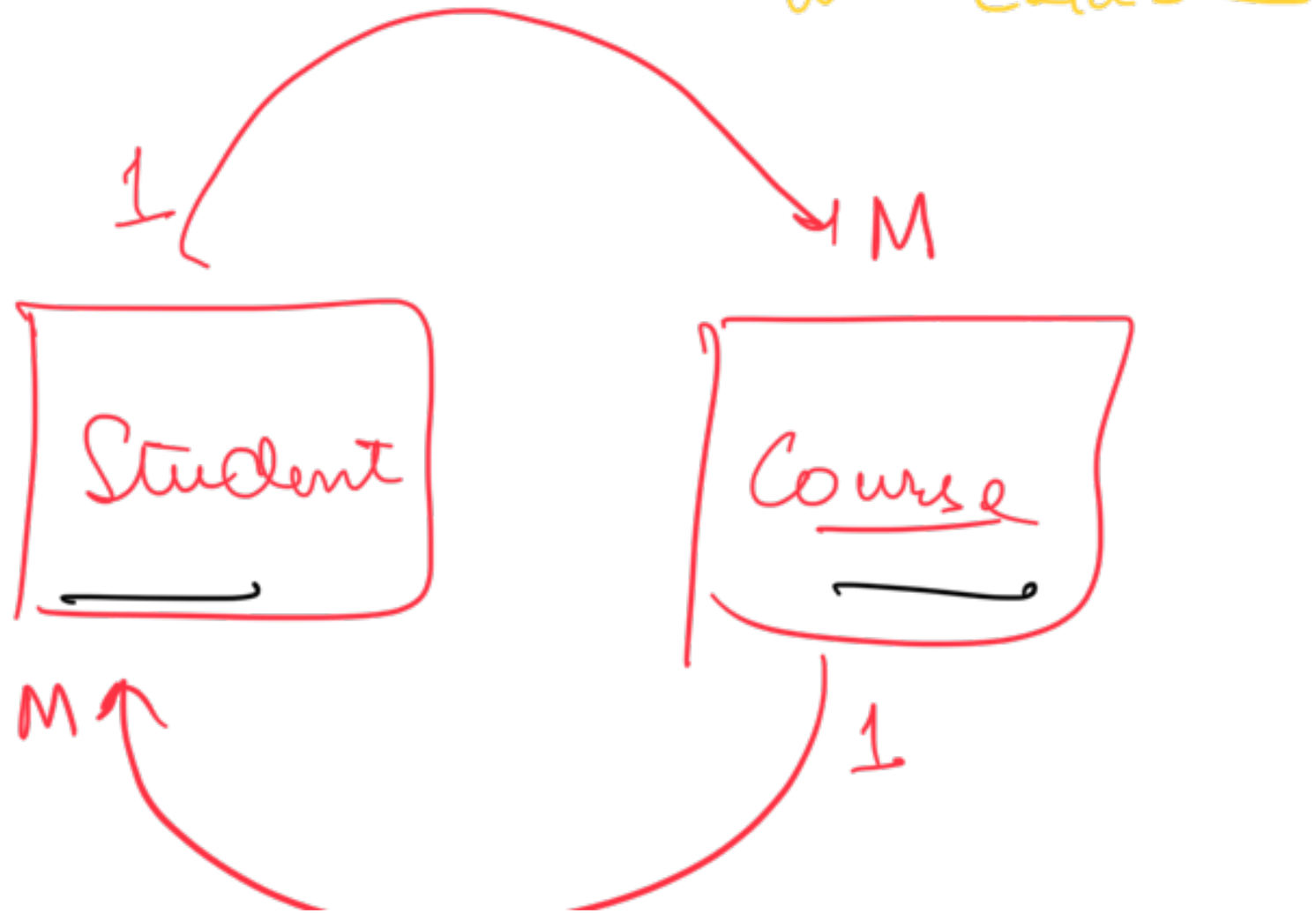
for course with id = 5

② update

for student with id 3

change replace c_id 7

with c_id 5



M : M

M:M ⇒ Always use a mapping table

Course — Students

C_id	Student_id

	NON-SPARSE	SPARSE
1:1	ID on 1 side of the other side	Mapping table
1:M M:1	FK on M side	Mapping Table
M:M	Mapping Table	

1 : 1

husbands

id	Name	age
	A	24
	B	28

wives

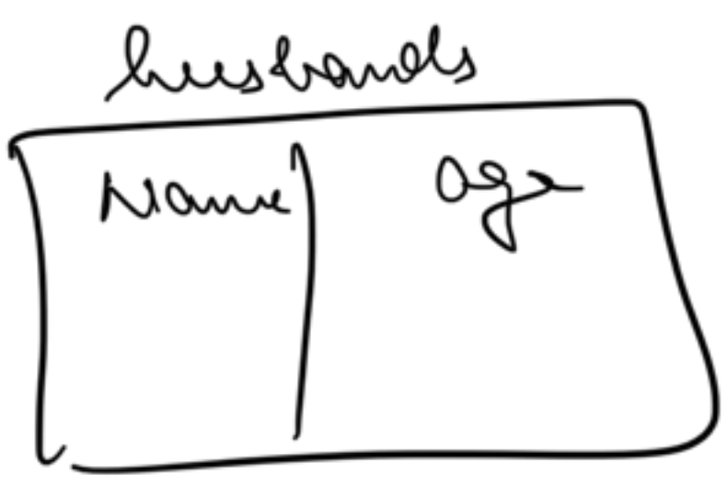
id	Name	age
	C	24
	D	28

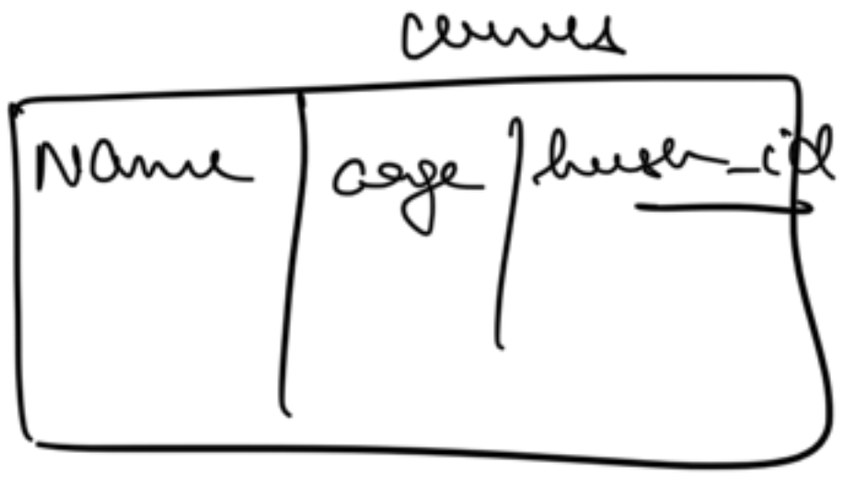
Opn 1 id on left side

husbands

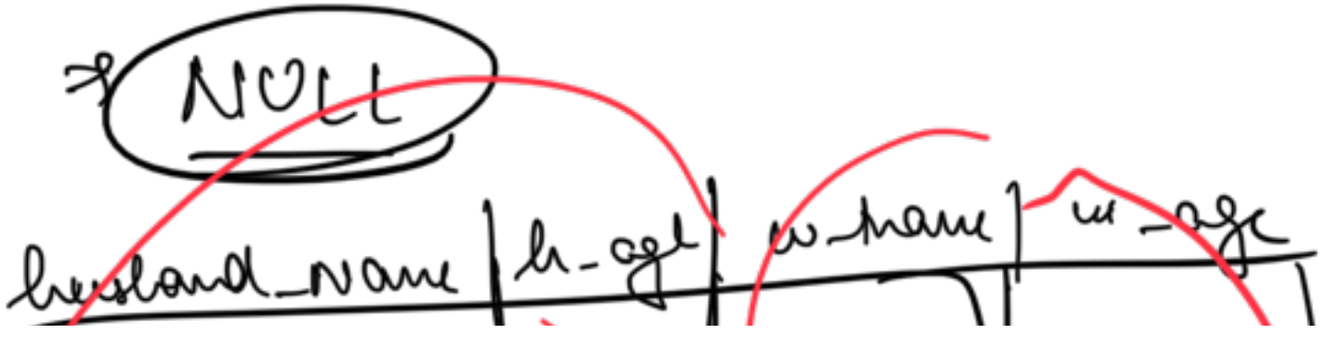
wives

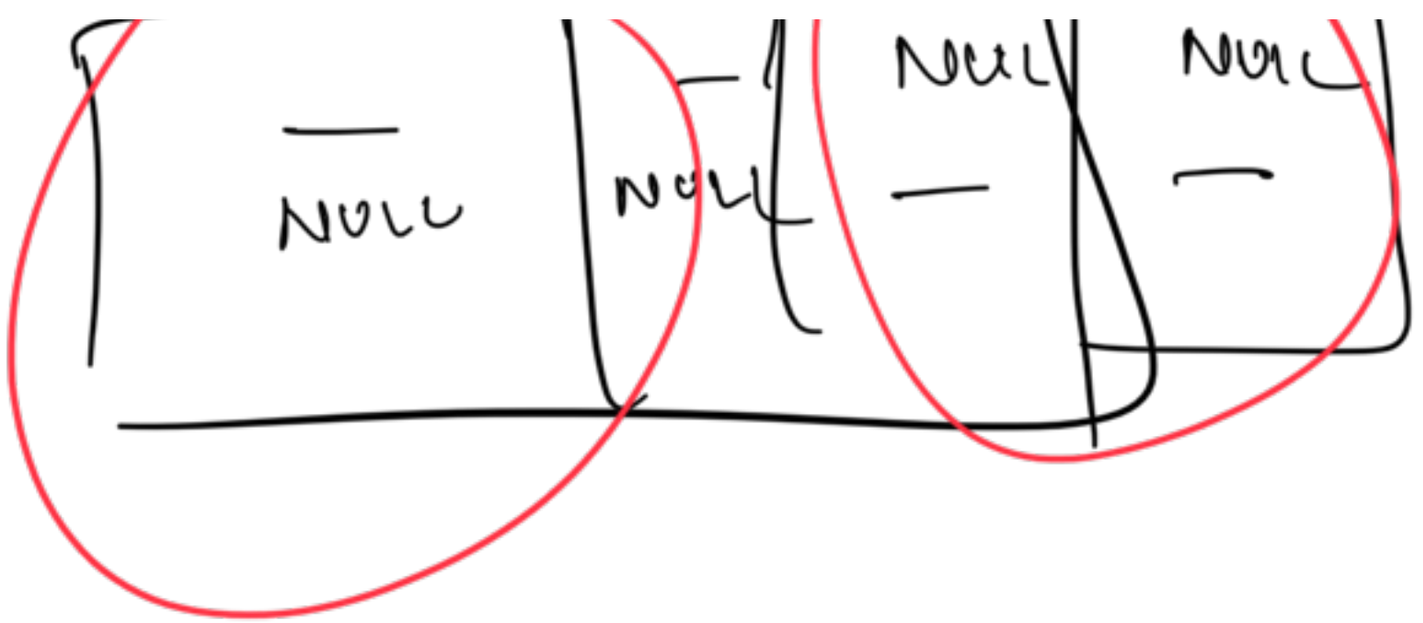
Op.n 2 id on right side

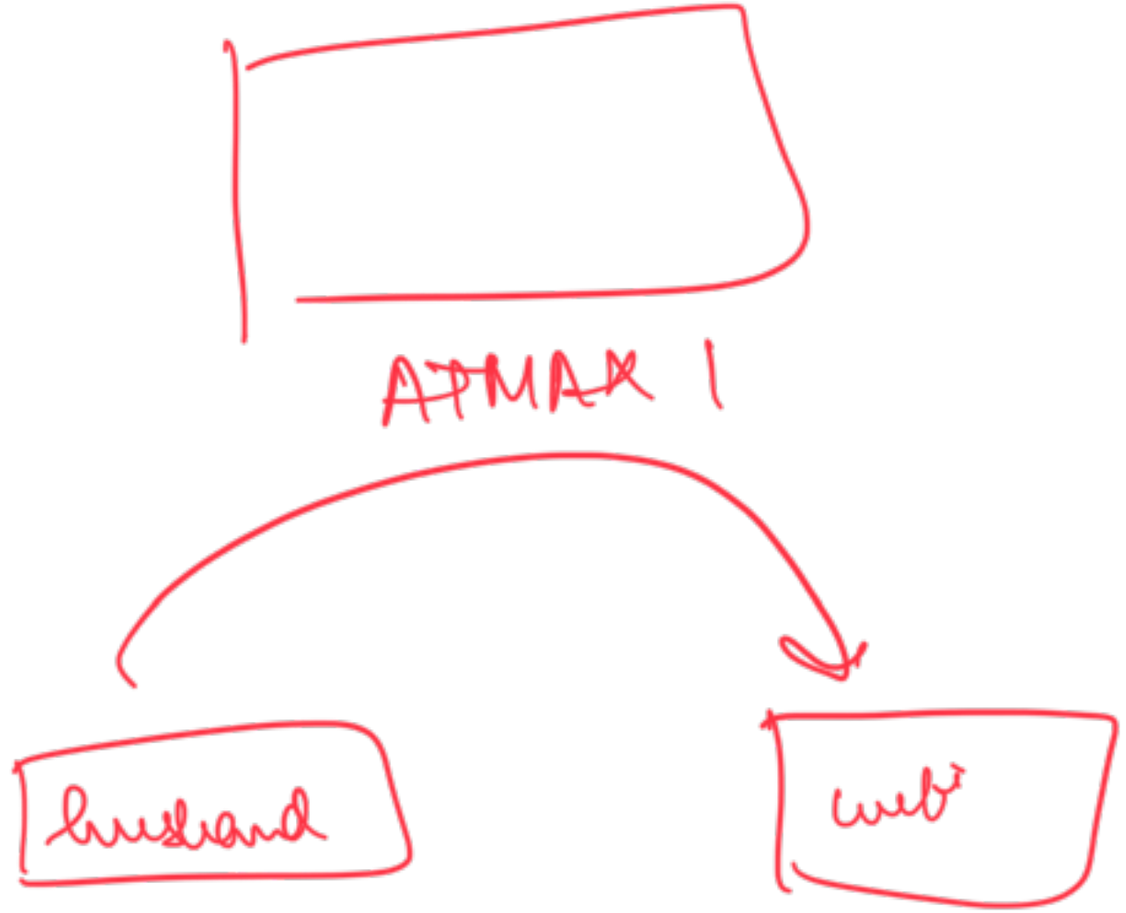




Sparse table ⇒ NULL




NULL
NULL
NULL
NULL


husband
wife

1 : 1

1

1 : 1

Users

op1

loadbeam_cards

op2

id	Name	phone	address	a_number	ph_NO	email	vkyc

up —

u_id	a_id

V4

1. Students
2. ~~Courses~~ Modules → DSA → 2 hrs → Feb → 5th Nov
   → CC → 30 hrs → E1 = 28 Oct
   → LLD → 1 [illegible] → E1 = 3 Nov
3. 1 student → Any # of ~~Courses~~ Modules
4. Every module will have Multiple exam

(5) Every exam has a duration

(6) One exam can be a part of multiple modules

(7) For every module, a exam has a fixed date

(8) One exam can be given only once for 1 Module

(9) We have to store marks of every student in every exam they attempted

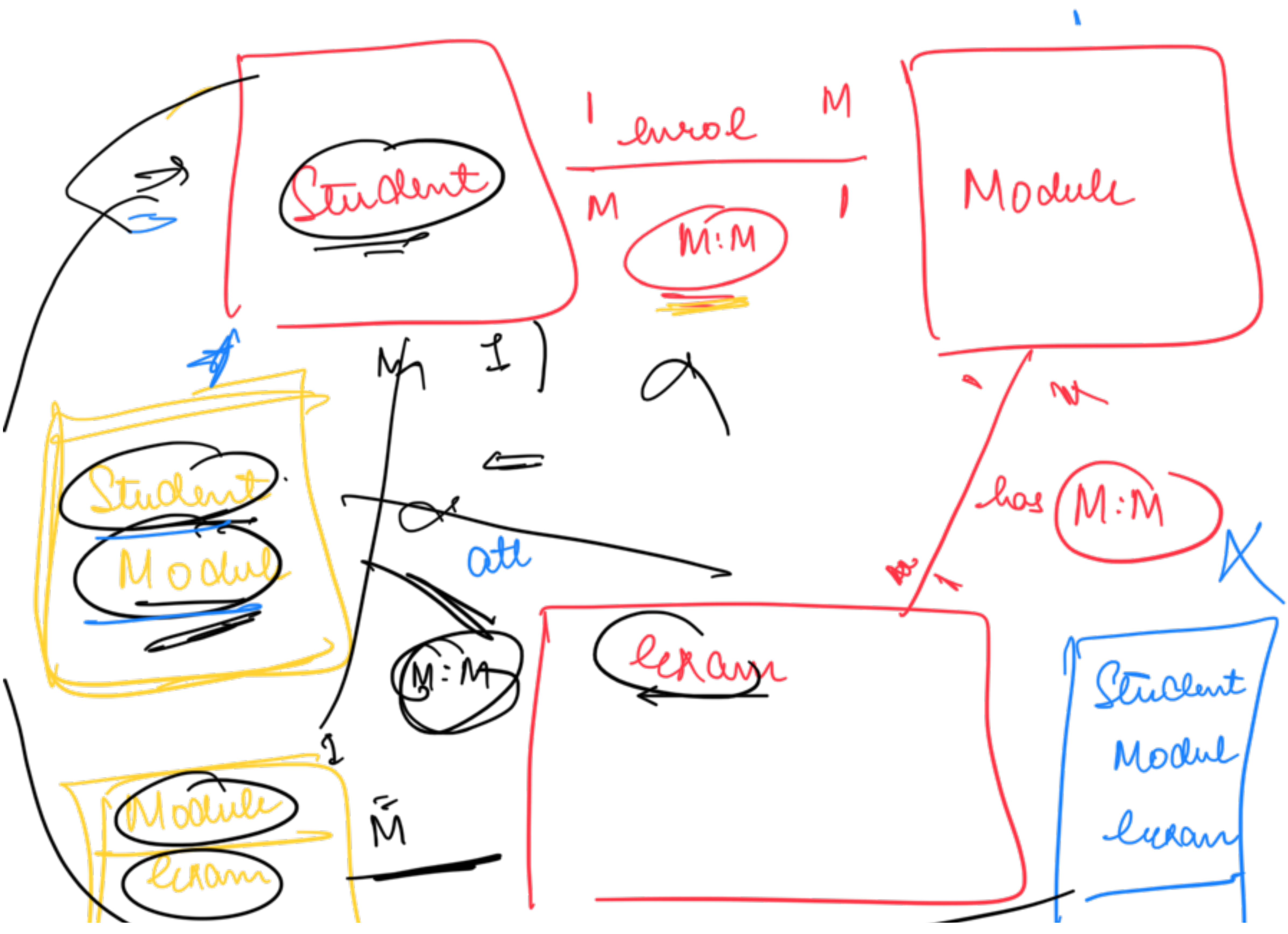
Step 1 :) Write all the classes

Student
- Name
- Add
- email
- phNo
- pass

Module
- Name

Exam
- duration
- Name

Step 2


Student
enrol
1
M
M
1
M:M
Module
has
M:M
M:M
att
Student
Module
Student
Modul

Schema

Students

id	Name	email	Pts	pwd

Modules

id	Name

Exams

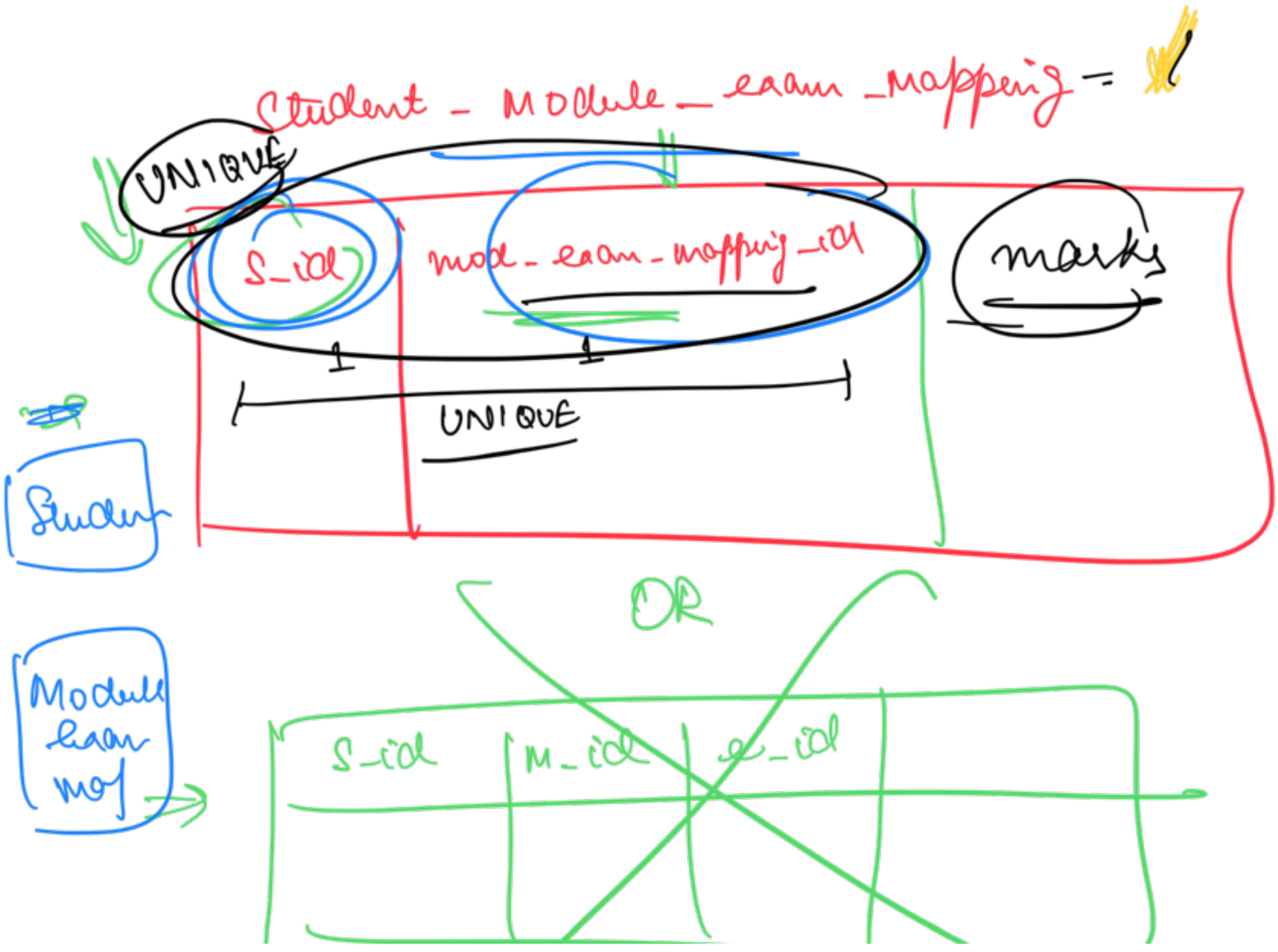
id	Name	duration

Student_Module_Mapping

S_id	m_id
1	

module_exam_Mapping

id	M_id	e_id	date


Student - Module - eaam - mapping =
UNIQUE
S_id
mod - eaam - mapping_id
marks
1
1
UNIQUE
Student
Module eaam map
OR
S_id
M_id
e_id

D
n

⇒ Even Mapping Tables can have all relations can have attr

Step 1: Identify all classes

Step 2: Make a table for each class

Step 3: For all the classes that exist till Now find cardinality of rel^{n} b/w them

Step 4: Rep all those rel^{n} in tables

Step 5: For every mapping table we created we should also add a class of them

repeat till no new rel^n

Step Last: Go through all the rel^n and see if any rel^n has attribute

draw.io

2 peers
1TA

Use Case : Every exam has q's

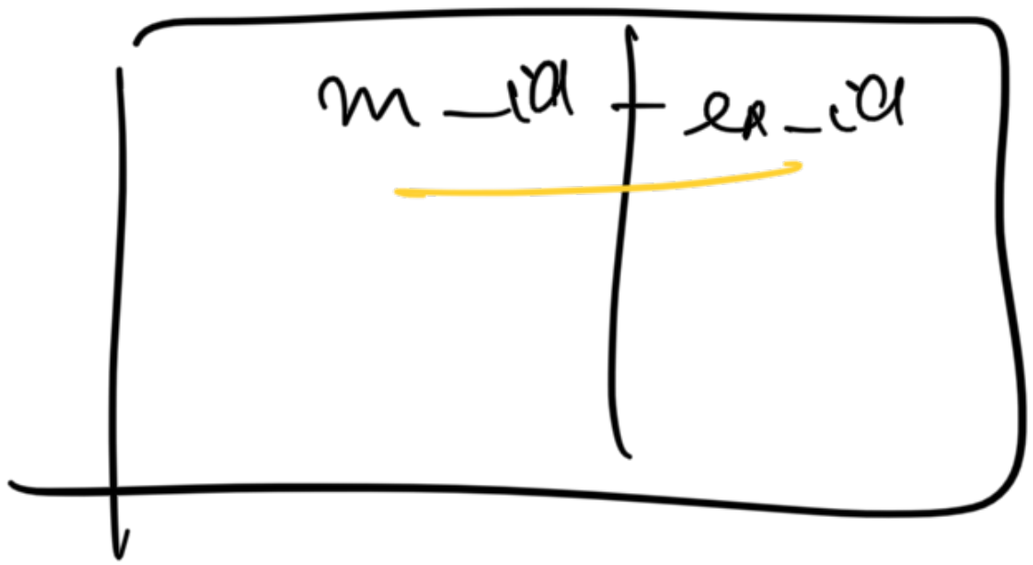
Student-module mapping

sm_id	st_id	m_id

exam-questions mapping

e_q_id	exam_id	q_id

student_exam_mapping



S_m_e_q_mapping