Agenda.

1. What is LLD ?
2. Why LLD is important ?
3. How to approach any LLD problem ?
   - → Requirements Gathering
   - → Class Diagram
   - → Schema Design
4. Doubts.

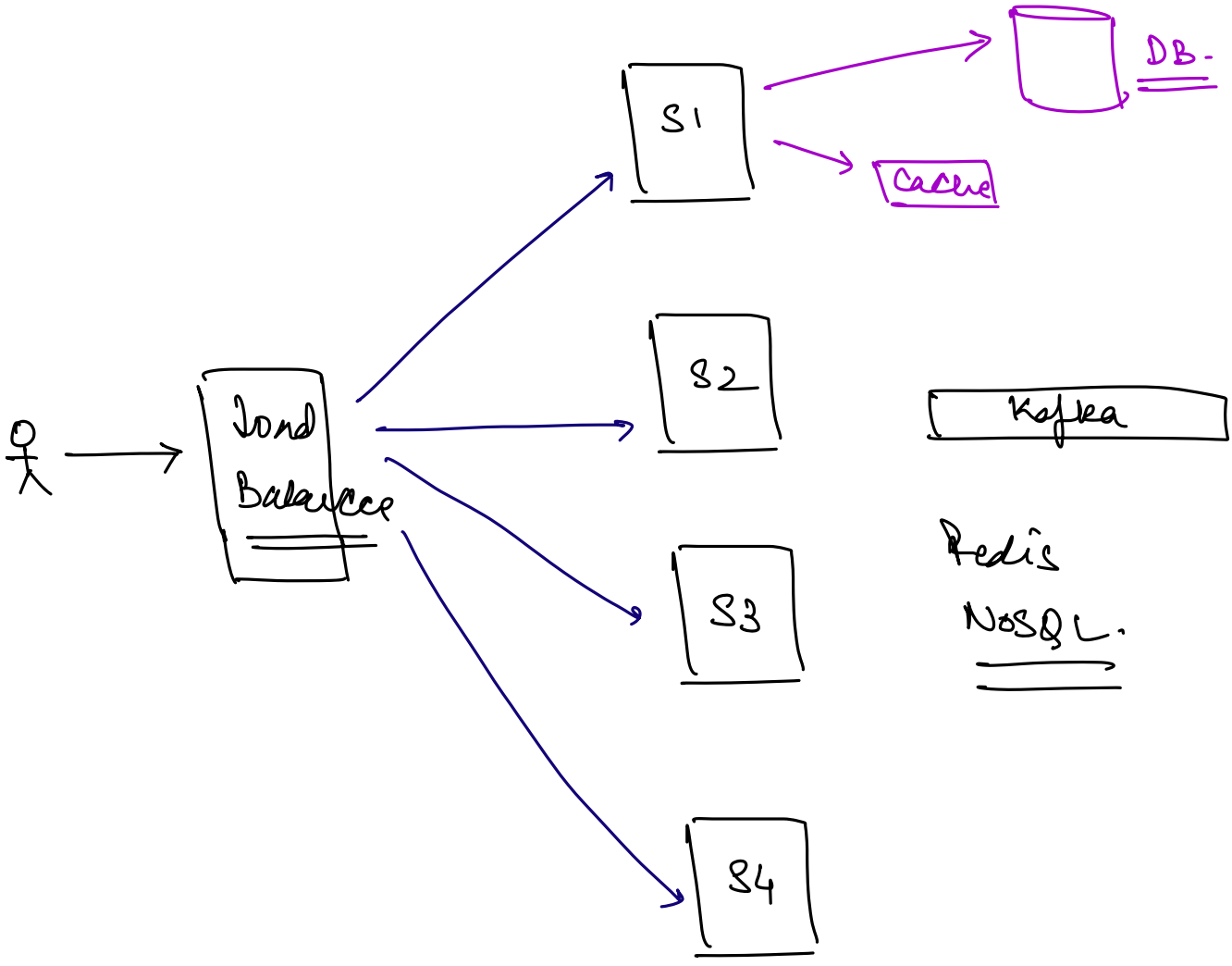
What is LLD.

Low level Design. ⟺ High level Design.

- → Bird's Eye View.
- → Brief
- → Not going into much detail

Arij's.

- → Single point of failure.
- → Bottleneck.



⇒ HLD.

→ Study of Different infrastructure layers interacting with each other in order to serve a request.

Who decides how a m/c will behaves

⇒ Software Systems.

⇒ It depends on what kind of Software you have installed in your m/c.

Software → Code.

⇒ LLD is the study of the code written to build Software system.

→ Details of how the code is structured.

~8 hrs.

↳ 2-3

→ Team.

1

Properties of a good S/w System.

- Understandable ✓
- Readable. ✓
- Easily testable ✓
- Maintainable.
- Extensible

↓

It should be easy to add new features

⇒ Why LLD is important ?

- SE : Code everyday.
- Interviews.

Atleast one LLD round

Flipkart | Prowoke | Swiggy ....

↓

1st round will be LLD round.

- Machine Coding. ⇒ 2-3 hours.

Amazon | MS | Apple | Atlassian | ...

- LLD round
  - ⇒ SOLID design principles
  - ⇒ Design Patterns
  - ⇒ MVC
  - ⇒ Schema Design.

Machine Coding ⇒ LLD + Working Code.

- Requirements
- Class Diagram
- Schema Design

→ UML

→ Working Code + Test cases.

⇒ Apart from writing code, what other activities we do in office

1. Meeting
2. Debugging ⇒ Reading code
3. Testing ⇒
4. Code Review ⇒
5. Requirement Gathering ⇒
6. Documentation ⇒

⇒ 12% of time writing code.

⇒ LLD makes you 88% of time, more productive

How to approach LLD problem.

1. Requirement Gathering
   - Understand.
   - Ask clarification questions
   - Edge cases.
2. Class Diagram.
   - What all the classes/interfaces
   - What all the methods
   - ——— attrs in a class.
3. Schema Design.
   - DB design
   - What all the tables.
   - Mapping b/w the tables
     - 1:1
     - 1:M
     - M:M
     - M:1
4. Code.
   - Implement. + Test cases } MVC architecture + DRY.

⇒ LLD of Payment Apps.

① Set of Requirements. (8-10 mins).

Payment:

↳ Buy iPhone worth 1L ₹

→ ① Pay the full amount via CC.
Single Transaction.

→ ② 12 Months EMI.

$\frac{100000}{12} = x$ (6th EMI)

12 transactions

→ ③ 50000 Rs ⇒ Amazon Pay ⇒ T1

50000 Rs ⇒ CC. ⇒ T2

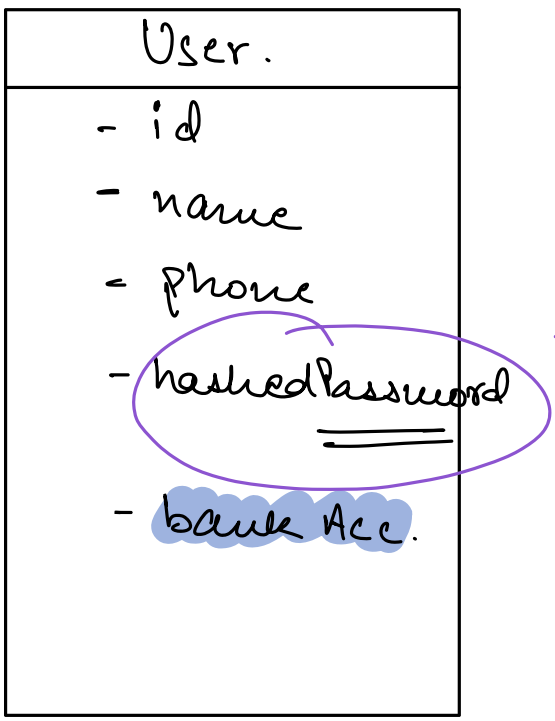
Every payment can have one or more than one transactions.

Class Diagram.

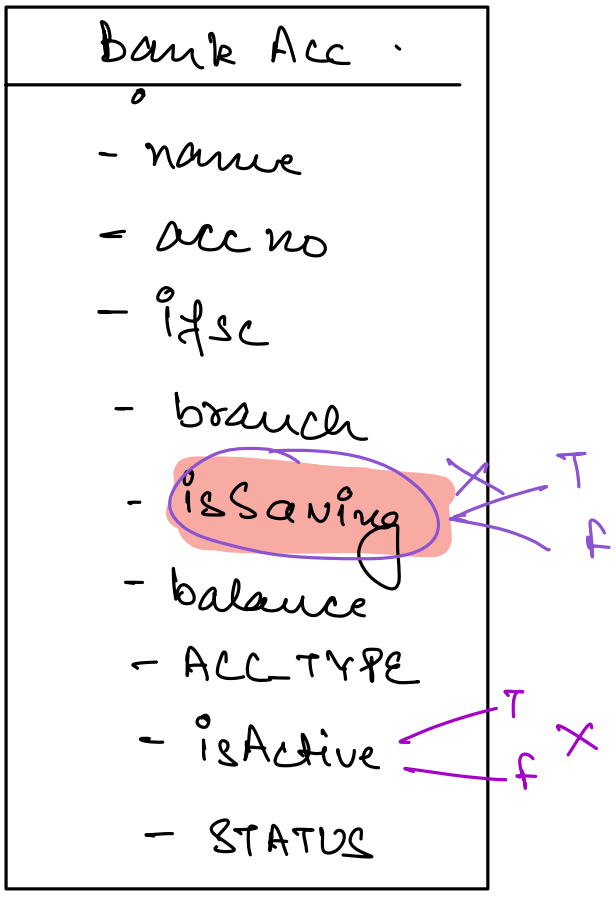
→ Go through the requirements.

→ Find out Nouns in each requirement & Check if you want to store data for this NOUN.

- if yes ⇒ Create a separate Class.
- else ⇒ Move on.



⇒ Blcrypt. Password Encoder.

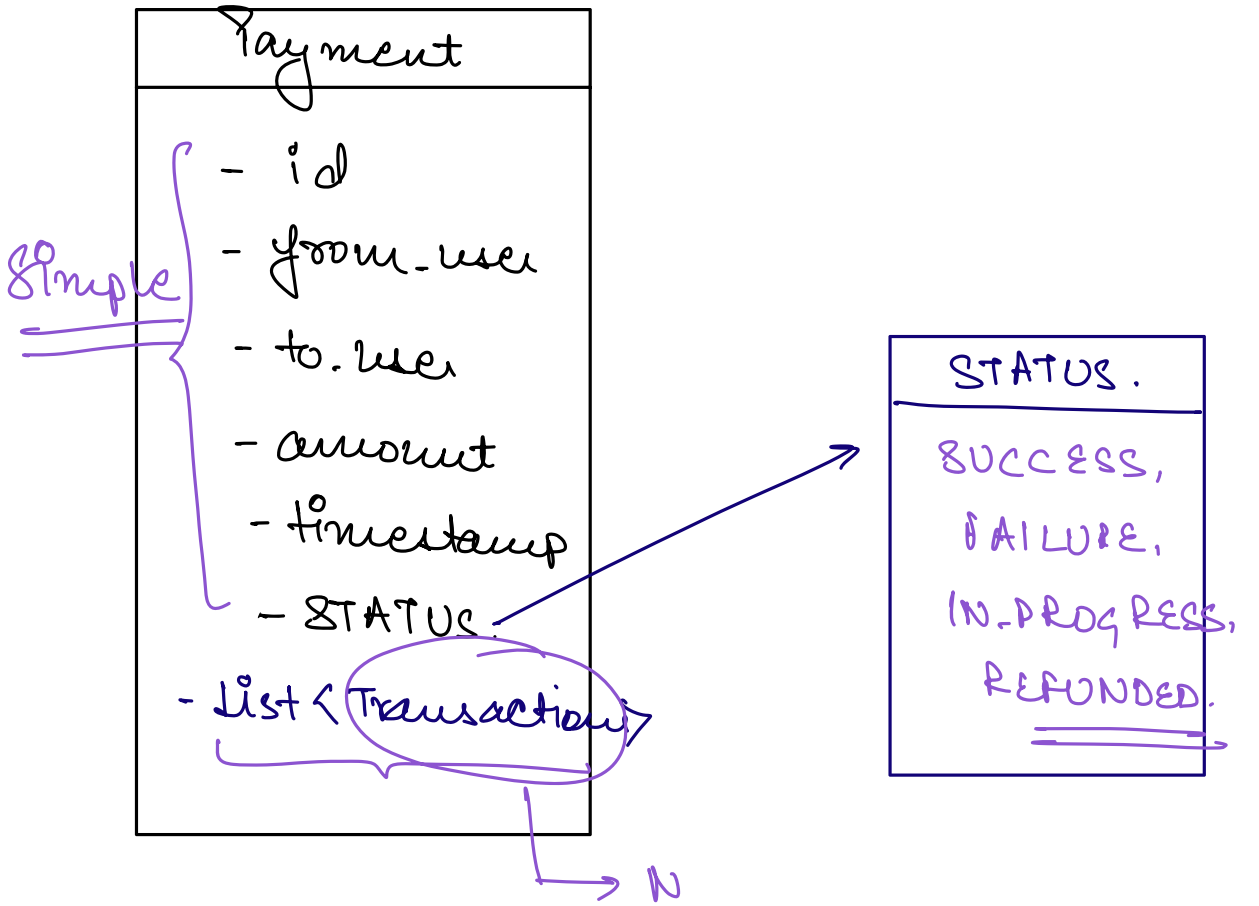


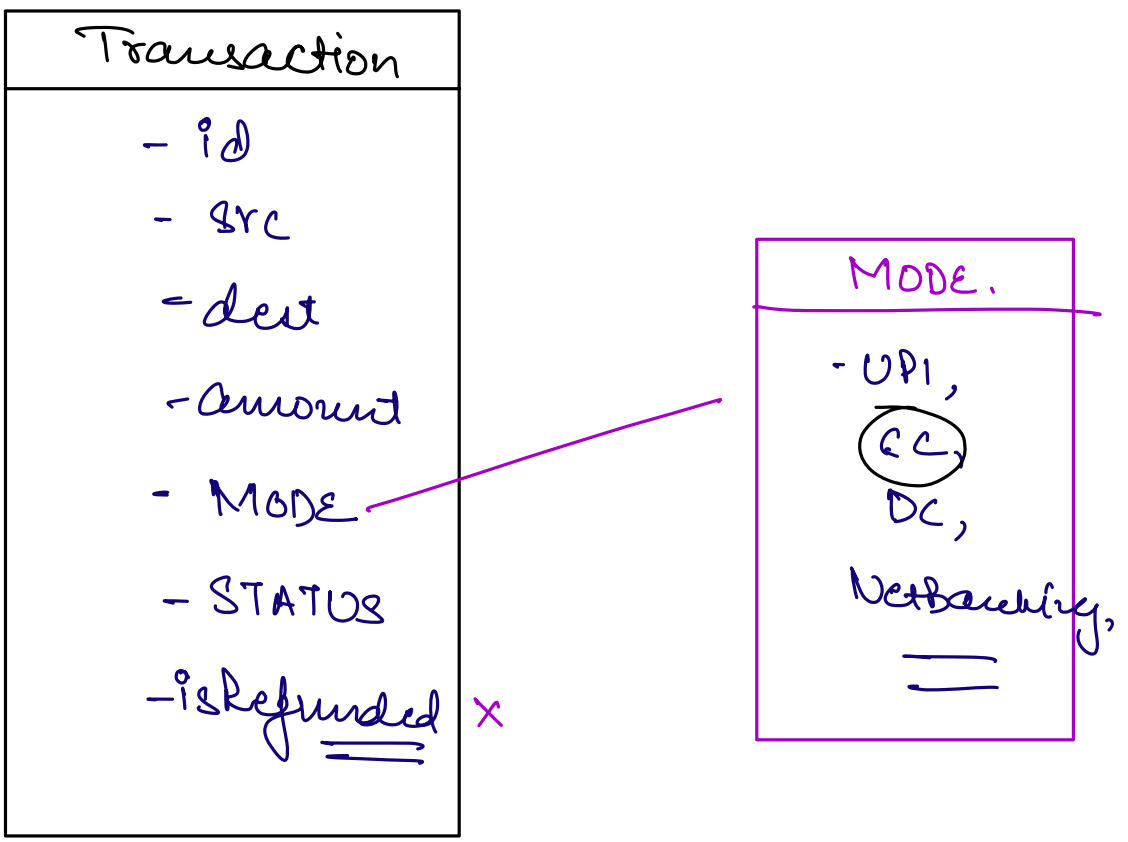
Try Not to use Booleans in your class diagram.

Why? Because Booleans aren't extensible.

ACC_TYPE
SAVINGS, CURRENT, HNI.

STATUS
ACTIVE, INACTIVE. BLOCKED, HOLD.


Payment
- id
- from_user
- to_user
- amount
- timestamp
- STATUS
- List<Transaction>
Simple
N
STATUS.
SUCCESS,
FAILURE,
IN_PROGRESS,
REFUNDED.


Transaction
- id
- src
- dest
- amount
- MODE
- STATUS
- isRefunded
MODE.
- UPI,
CC,
DC,
NetBanking,

⇒ Schema Design.

- Tables.
- Columns in the table
- PK, FK..
- Relationships b/w the tables.

⇒ for every class that we have come up in class, create a table.

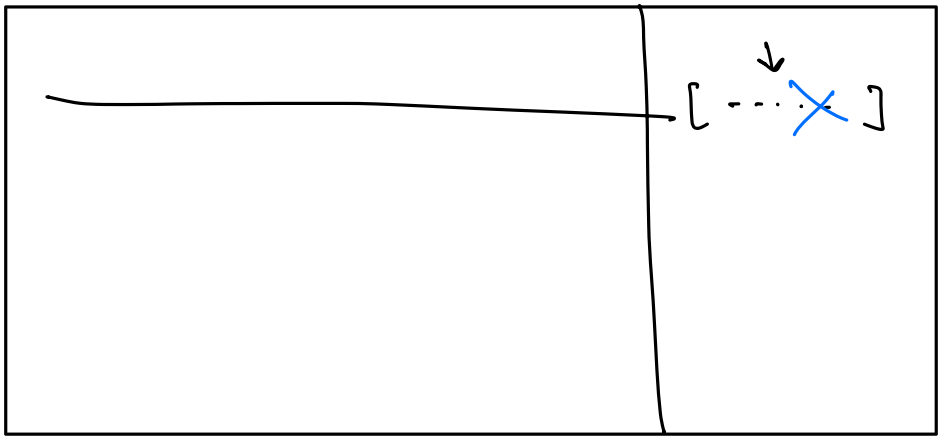
⇒ Class can have 2 type of attributes

- Primitive / Simple: int, bool, double, String
- Non Primitive / Complex.

⇒ Primitive attrs we can directly add as columns in the table but for Non primitive we need to find their relation/mapping.

MySQL

Users



⇒ Every column should only contain atomic value.

Single

Payments

id	fr	to	amount	transactions.
1	A	B	1008	[T1, T2, T3]

} X

⇒ Mapping / Cardinality.

- 1:1
- 1:M / M:1
- M:M.

-

User. ⇒ 10M → lot of space.

	Car-type
	1
	2
	3
	1
	2
	3

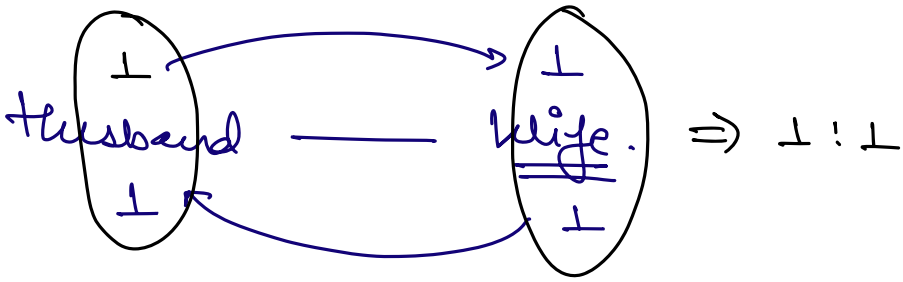
SUV → XUV.

Car-type

id	Value
1	SUV → XUV
2	SEDAN
3	HB

⇒ Cardinality.

1 : 1



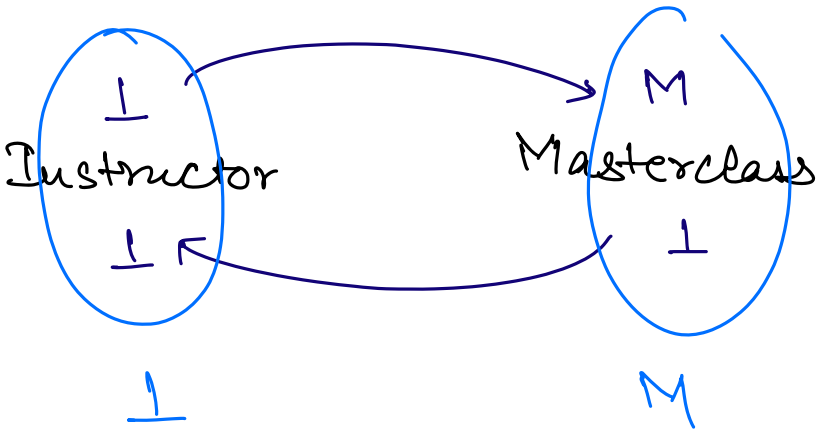
Husbands

	wife id

wives.

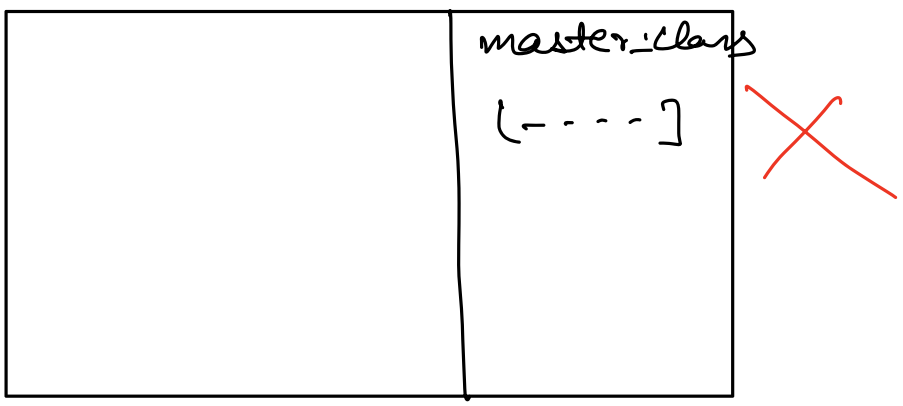
	Hus-id.

1 : M / M : 1

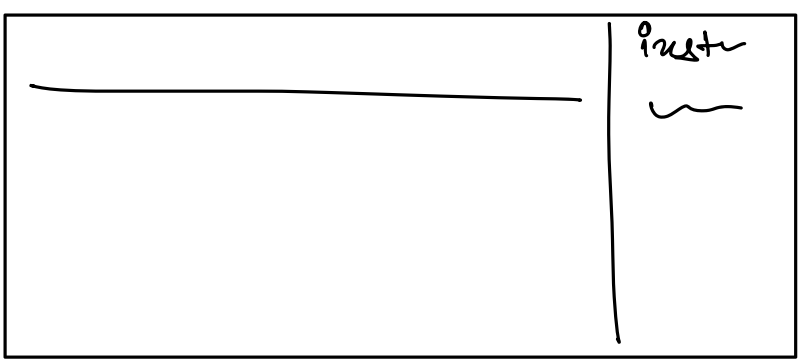


$\Rightarrow$ 1 : M.

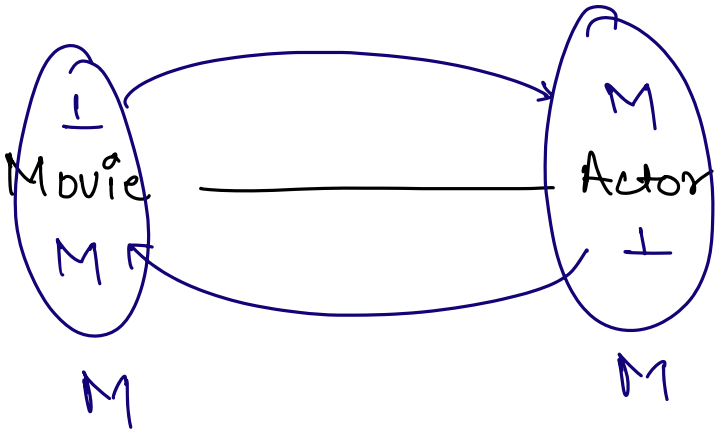
instructors



masterclass

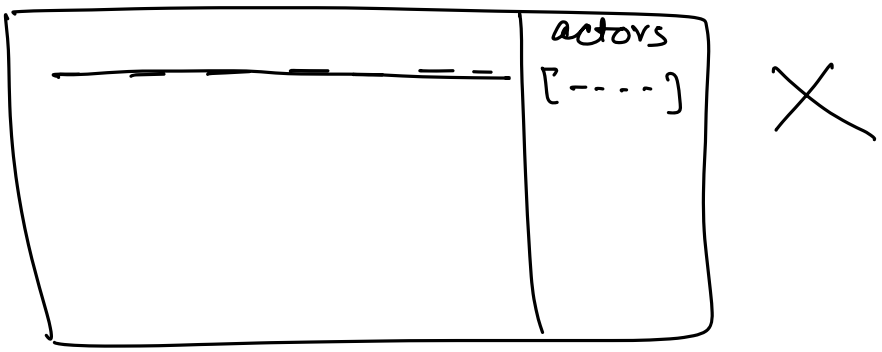


⇒ M:M.

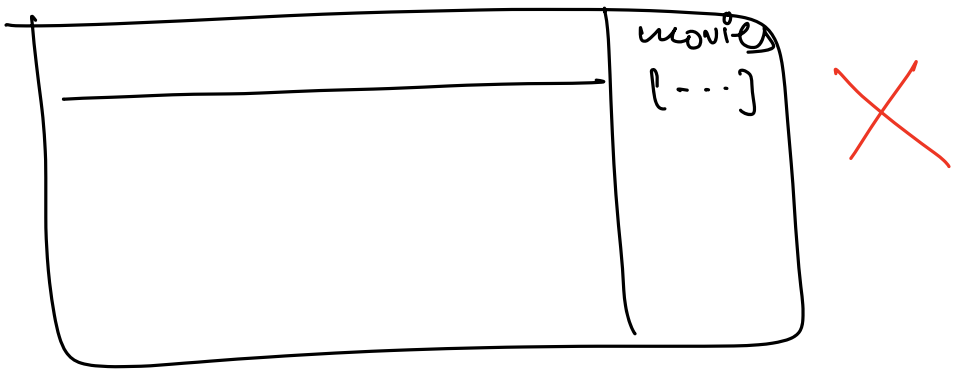


⇒ M:M.

movies



actors



Mapping

movie_actor.

movie	actor
—	—
—	—