AGENDA:

Announcement:-

Saturday → 7:00 AM to 10:00 AM. IST

Weekdays → 7:00 AM IST to 9:30 AM IST. (From Tuesday)

1) Schema Design (few things that we will talk about in Schema Design)
   - → How to decide what tables
   - → How to model the cardinality between tables
   - → Normalization

2) SQL Data types

Database Schema

Design → Class Diagram
→ Usecase diagram
→ Api Design
→ Database Design.

Database Schema is how my database will be structured.

- → What are going to be different tables
- → Attributes inside every table
- → Cardinality of relation b/w 2 tables.

Pictorial representation of how your DB is going to be structured.

Schema Design is part of a design doc.

How To Approach Schema Design

Case Study:- Scaler.

Requirements:-

1) Scaler has students.
   Every student has name, grad-year, university, email, phone number.
2) There are several batches at scaler.
   Each batch has name, start month, instructor, students.
3) Every batch has classes.
   Each class has name, instructor.
4) Each student has a student buddy

5) A student may move from one batch to another.
   → Course Pause
   → Student Request
   Entry date of a student to a batch.

		Joined	Left
Priyanka	Aug 21 Batch.	22/08/2021	30/08/2021
	Oct 21 Batch.	30/10/2022	NULL.

6) Every student will have a mentor
   Each mentor will have a name, currentCompany, yoe, dob, education

Assignment

→ Design Netflix Schema.

Modelling Schema.

1) Find all entities in requirements.
   - anything for which we are storing details
   - anything causing some behaviour in the system.
     - real thing (Student, Mentor, instructor)
     - conceptual thing. (Batch, class)

How to find entities.

1) find all the nouns in the requirements.
2) for every noun, you see if you have to store any information about that noun.
   - If yes, you create a table for that.

1. Batches.
2. instructors.
3. classes
4. Students.
5. Mentors.

Batches

PK →
- id : INT
- name
- current_instructor
- classes

Instructors

- id
- name

Students

- id
- name
- grad_year
- university
- email
- phone number
- current_batch
- prev_batch

Classes

- id
- name
- instructor.
-

Mentors

- id
- name
- current company
- yoe
- email id.

Naming Conventions for Attributes.

- Camel Case.
- '_' delimeter. current_instructor.
- current instructor (Theoretically possible)
  - very wrong to do this.

id related conventions:

- id (common way)
- {table name}_id e.g. batch_id, class_id. (Helpful in natural joins)

How are we storing current_instructor in batches.

↓

Instructor entity.

There are rules around how to store them.

Cardinality of relationships b/w Entities.

If there is a relation b/w A and B

Cardinality → How many of A : How many of B.

1 : 1

1 : M

M : 1

M : M

M → many.

How to cardinality.

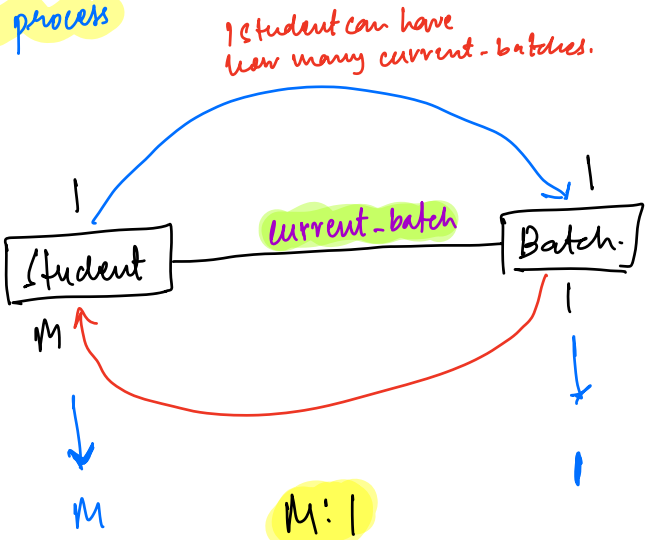
Step 1:- Find which relation b/w A and B are we talking about.

A: Student

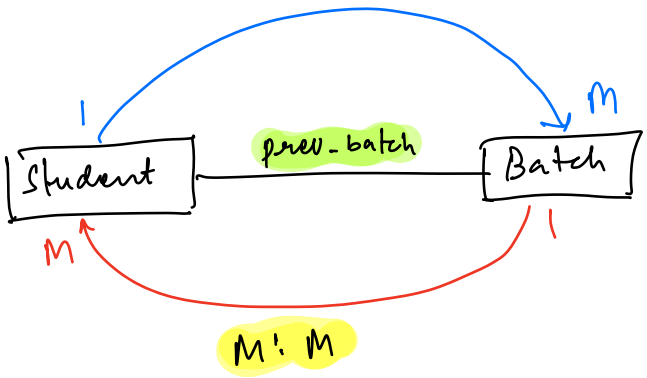
B: Batch.

Relation:- current_batch.

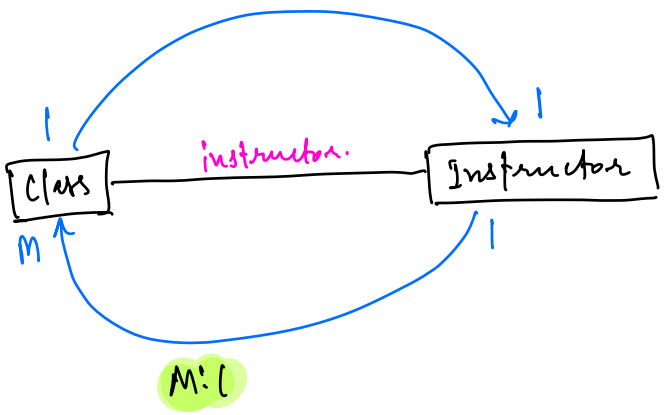
Step 2:- 2 step process



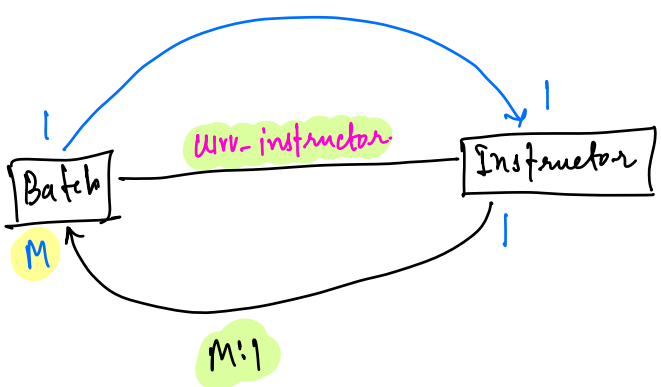
Step 3:- If M on one side, you put M there else 1 on that side.


1
M
Student
prev_batch
Batch
M
1
M : M

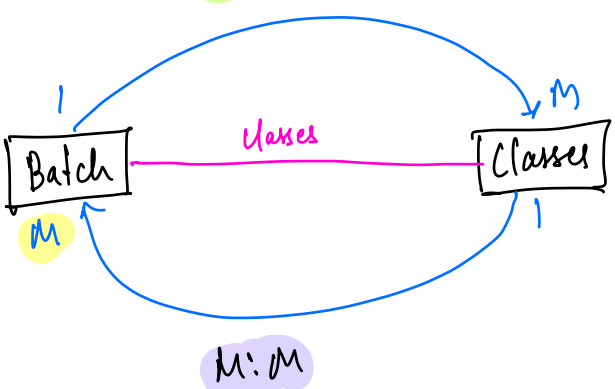
e.g.


1
1
Class
instructor
Instructor
M
1
M : 1

e.g.


1
1
Batch
curr_instructor
Instructor
M
1
M : 1

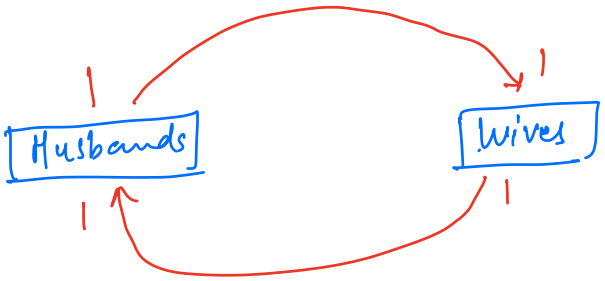
e.g.


1
M
Batch
classes
Classes
M
1
M : M

How to represent different cardinalities in a database.

1) 1:1

e.g.



1:1 cardinality.

Husbands.

id	name	wife-id

OR

Wives.

id	name	hus-id

In 1:1 relation, put the id of other (ANY) side on the diff. side.

hus-id in wives table as fk

OR

wife-id in husbands table as fk.

2) 1:M or M:1

Class

instructor-id

instructors.

class ids
[1, 2, 3, 4..]

1 class can have 1 instructor

1 instructor can have M classes

→ Store instructor-id in class table ✓

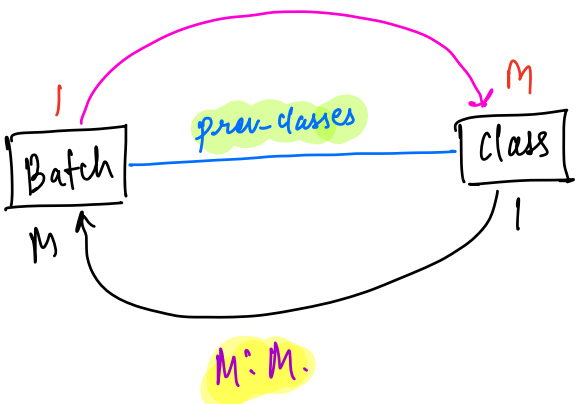
→ Store class_ids in in instructors table. X

2 options:-

1) The table which is related to many can have a list of ids ✗
   (Multivalued attributes not recommended) (Atomic values).

2) The table which is related to 1 can have a single id. ✓

3) M:M.



Batches.

	prev_classids
	[list of ids] ✗

Classes.

	batch_ids.
	[id1, id2,] ✗

Create a new table to store relationship b/w these 2.

→ mapping table / lookup table.

batch_classes.

id	batch-id	class-id.
	1	2
	1	5
	1	7
	2	2
	3	2

→ PK for mapping table.

7:00 AM, 10:00 AM

↳ 3 hours

→ PK for mapping table.

1) Combination of both fk (batch_id, class_id) ✓

2) A new column (id column)

↓

when will you not go with this approach?

① If the mapping table can also have relationship with some other table.

② there are attributes of relation and the relation itself acts like an entity.

e.g. Batch and class will have an assignment

batch_classes.

id

batch_id	class_id.
1	2
1	5
1	7
2	2
3	2

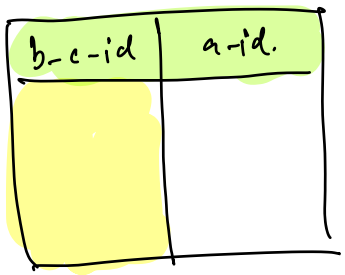
Assignments.

id	questions.

If the relation b/w these 2 tables is M:M, then I will want to have a mapping table for storing that relationship.

batch_class_assignment

batch_id	class_id	assignment_id



Why prefer multivalued or composite PK

Small note:- By default, your table is sorted according to PK

PK is (batch_id, class_id)

Caveats

1:1 / 1:M / M:1 cardinality.

e.g.

Husbands

id	name	wife_id
1	—	1
2	—	2
3	—	NULL
4	—	NULL
5	—	NULL

fk

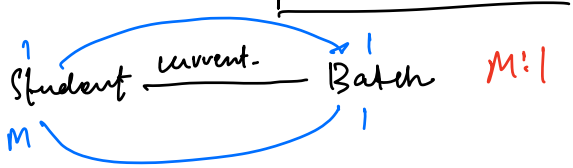
Wives.

id	name

5 million husbands → 3 million don't have wives.

Students.

Batches.



25K joined master class → 500 enrolled.

↓

24.5K NULLs.

We are wasting a lot of storage here.

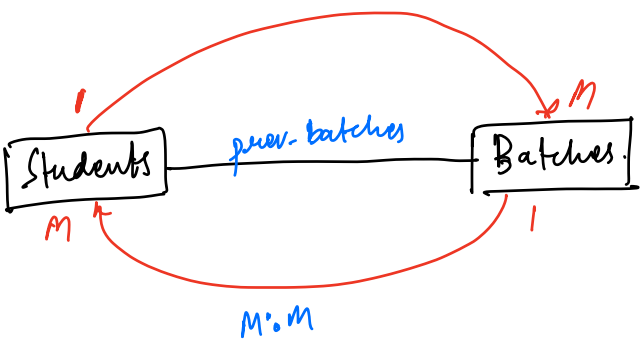
Whenever there are NULLs in the fk, what to do?

→ A new table is your ans. (mapping table)

* If relation representation can have a lot of NULLs (sparse table), consider representing the relation using a mapping/look up table.

Caveat 2

A relation can also have attributes.



Mapping table.

id	stud_id	batch_id	st_date	end

Attributes for this relation

→ start_date, end_date.

Ques 1:1

Husbands-

id	name	wife_id.

Wives.

id	name.

Where to put anniversary?

Husbands-

id	name	wife_id.	anniversary.	engagement date	no. of kids.	honeymoon desti
			X	X	X	X

Marriage Entity.

husb_id	wif_id	anniver	engage

* Whenever any relation (1:1 / M:1 / 1:M / M:M) starts acting like an entity, consider it a new entity and breaking it out of the table and forming a new mapping table with Pd.

Summary.

Type of Relation → / Cardinality ↓	Normal	Sparse	Like an entity
1 : 1	FK of any side one other side	Mapping table	Mapping table
1 : M or M : 1	FK of ONE's side on M's side.	Mapping table	Mapping table
M : M	Mapping table	Mapping table	Mapping table

SCHEMA MODEL

Batches

- id : INT
- name
- curr_ins_id — M:1
- batch_start_date — M:M

mapping table: Batch_classes

- batch_id
- class_id

Instructors

- id
- name

Students

- id
- name
- grad_year
- university
- email
- phone number
- curr_batch_id — M:1, M:M
- curr_batch_start_date
- mentor_id

mapping table: Students_batch_prev_batches

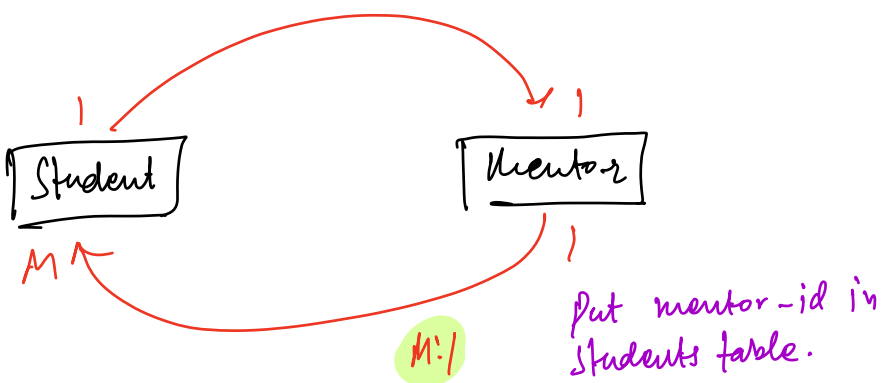
- id
- stu_id
- batch_id
- start_date
- end_date

Classes

- id
- name
- instructor_id M:1

Mentors

- id
- name
- current company
- yoe
- email id



Break:- 8:27

SQL Data Types-

Types of Data Types in SQL (MySQL 8 Datatypes)

1) String
2) Integers
3) Floating Point
4) Boolean
5) Enum
6) DateTime
7) JSON
8) Blob

#Strings.

1) CHAR
2) VARCHAR
3) ~~*~~TEXT

Char(x)
→ strings of fixed max length.
→ x → [0, 255]

eg. CHAR(4)
"abc" → Yes → "abc␣" (space.)
"abcd" → Yes.
"abcde" → No
↳ Error.
↳ abcd.

Where to use CHAR?
→ Phone numbers
→ State codes
→ Pin code
→ GST Number

② VARCHAR (x)

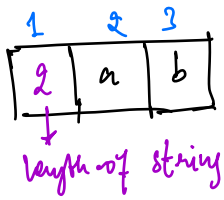
x can be from 0 to 65535 (2^{16})

→ String of variable length.

→ What is x here?

→ 1-2 bytes are used to store length of string.

e.g. "ab"



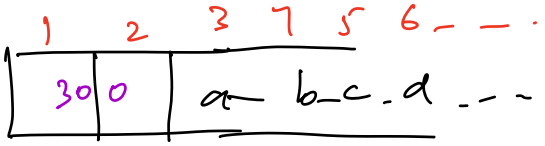
VARCHAR (5)

"abcde."

"abcdef"

"abc.

e.g. String of length = 300



i):

String		In memory.
""	→	1 byte (To store size. which is 0 here)
"a"	→	2 bytes 1 a
"ab"	→	3 bytes 2 a b
"abc"	→	4 bytes 3 a b c
"abcd"	→	5 bytes. 4 a b c d.

* Name, email, etc.

3) #TEXT

→ Different types of text.

Not much use.

TINY TEXT	TEXT	MEDIUM TEXT	LONG TEXT
255 B	64KB (Blog post)	16MB	4GB

TEXT columns cannot be indexed.

e.g. select * from students where name = 'Neha'

Integer Data types.

TINY INT → 1 byte [-128, 127]

User/byte
UNSIGNED TINY INT → [0, 255]

SMALL INT → 2 bytes [-32K, 32K]

MEDIUM INT → 3 bytes [-8M, 8M]

(int) INT → 4 bytes [-2B, 2B]

BIGINT → 8 bytes. [-9Z, 9Z]

(long long int)

Floating Data Types.

a) DECIMAL(p,s) → How many values you want to store after decimal.
↓
precision / total length

e.g. 302.04 p=5, s=2

If I know p and s, I can store the exact values in my SQL DB and put the decimal.

$1.9 - 0.9 == 1$

1.89999 0.89999

$1 - \text{epsilon} \leq 1.9 - 0.9 \leq 1 + \text{epsilon}$.

epsilon → 0.02

Actual	Close approximation.
0.2	0.20001
0.01	0.00099
0.5	0.50101
0.001	→

10^6 stocks. → 10^{-3} in each stock.

$10^6 * 10^{-3} \rightarrow 10^3$ stocks.

1000 stocks.

fee, amount, tax.

300095
↓ Rupee, ↓ paisa

3000.95

Why use floating points?

→ They allow you to store a huge range of numbers

They are used

→ Where having approx values is okay.

→ You can add extra logic to handle approximation.

② float

③ Double.