⇒ **BOOLEAN:-** (BOOL or BOOLEAN)

⇒ Boolean is nothing but a TINYINT 1 Byte

↓

0 → false

or

1 → true

⇒ **ENUMS:-**

```
class Car {
    String name;  → anything is fine
    String brand; → anything is fine
    CarType ~~String~~ carType; →
}
```

↓

SEDAN

SUV

HATCHBACK

Constant values which can be used as a type in attributes ⇒ ENUM

CarType (SEDAN, SUV, HATCHBACK, HUV)

Car

Cars

id	name	brand	carType
1	Nano	Tata	Hatchback
2	730D	BMW	SEDAN
3	XUV700	Mahindra	SUV
4	Scorpio	Mahindra	SUV
5	Thar	Mahindra	SUV

- Redundancy => SUV => 3 times
- If we want to change SUV to HUV, we will have to update query over the entire table

Practical advice:

Cars

id	name	brand	carType_id
1	Nano	Tata	1
2	730D	BMW	2
3	XUV700	Mahindra	3
4	Scorpio	Mahindra	3
5	Thar	Mahindra	3

fk

pk

CarTypes

id	type
1	HATCHBACK
2	SEDAN
3	SUV

Date and Time

↳ A sep class is going to be there on working SQL queries where we will use Date/Time functions.

⇒ Date ⇒ only allows us to store date.

ex ⇒ 28/12/2023

⇒ Time ⇒ only allows us to store time

ex ⇒ 06:00PM

⇒ DateTime ⇒ allows us to store both of them together

28/12/2023 06:00PM

⇒ Timestamp

⇒ no. of milliseconds since

01/01/1970 UTC

9:38 PM ⇒ 10/03/2023 ⟶ 1678464535

=> Dont use specially with MySQL

4B

2038

01/01/2039 => no. of milliseconds > storage of 4B

2038 problem

Year-de => Year 2000

Y2K => 2000 => for year just store 2 digits

96 — 1996

1989 — 89

1999 — 99

47 — 1947

63 — 1963

00 — 2000

1900

Y Y

0 0

1900 2000

99 => 1979

00 => 1900

2000

BLOBS

↳ Binary large Object

Storing files in DB is done by storing binary of those files.

- TINYBLOBS — 255 B
- BLOB — 65 KB
- MEDIUM BLOB — 16 MB
- LONG BLOB — 4 GB.

* Until or unless a very specific requirement is present, don't use these blobs

* Generally file storage system is used to store files. ⇒ HLD

⇒ NORMALIZATION:-

* A technique to reduce redundancy of data in a database.

* Helps to create a good | organised | optimised schema.

* By mostly creating extra tables, to reduce the redundancy

⇒ Problems / Anomalies due to redundancy :

Student

id	name	phone no	gender	batch id	batch Name	Instructor Name
1	Rakesh	1234	M	1	B1	Sandeep
2	Shubham	4632	M	2	B2	Naman
3	Shivam	9652	M	1	B1	Sandeep
4	Dipali	0012	F	3	B3	Mohit
5	Pankaj	0132	M	2	B2	Naman
~~null~~	null	null	null	4	B4	Satya

* Insertion Anomaly

Can we create a fresh batch without adding a Student ?

* Deletion Anomaly.

* Deleting some entries might lead to deletion of data as for other entity.

⇒ delete Student with id = 4

* Updation Anomaly :

Student

id	name	phoneNo	gender	batch id	batchName	InstructorN
1	Rakesh	1234	M	1	B1	Sandeep
2	Shubham	4632	M	2	B2	Naman
3	Shivam	9652	M	1	B1	Sandeep
4	Dipali	0012	F	~~3~~ 2	~~B3~~ B2	~~Mohit~~ Naman
5	Pankaj	0132	M	2	B2	Naman
6	Rishabh	0152	M	~~4~~ 2	~~B4~~ B2	~~Satya~~ Naman

1) name => Student 4 | 6 to 0 new batch => B2

2) instructor

id	batch id	name
1	1	Sandeep
2	2	Naman
3	3	Mohit

Student

id	name	phoneNo	gender	batch id	batchName	InstructorN
1	Rakesh	1234	M	1	B1	Sandeep
2	Shubham	4632	M	2	B2	~~Naman~~ Sandeep
3	Shivam	9652	M	1	B1	Sandeep
4	Dipali	0012	F	3	B3	Mohit
5	Pankaj	0132	M	2	B2	~~Naman~~ Sandeep
6	Rishabh	0152	M	4	B4	Satya

⇒ update instructor of B2 to Sandeep in Students table

updation anomaly ⇒ leads to high no of data inconsistency

DB Normalization :

⇒ Process to organize data in DB

⇒ Reducing the redundancy is the biggest motive

There are many forms of normalization, each of the form are stricter than the earlier.

1) 1NF – FNF – First NF
2) 2NF – SNF – Second NF
   3NF – TNF – Third NF
   3.5NF – BCNF – Boyce Codd NF
   4NF
   5NF
   6NF

incrementally practically ⇒ no advantage of using them

⇒ 1NF ⇒ 1st Normalization form

- Every cell should only have atomic values
- No multivalued attribute

ex ⇒ Student

id	name	phoneNo.
1	Shubham	[1234, 5678]
2	Rakesh	[0011, 0022, 0033]
3	Kranti	[1122, 1133, 1144, 1551, 1662]

violating 1NF

Sol

Assume a person have max^m 4 phone no.

id	name	ph1	ph2	ph3	ph4
1	Shubham	1234	5678	null	null
2	Shivam	0011	null	null	null
3	Rakesh	0010	0020	0030	null
4	Madhav	1111	null	null	null

too many null

1NF 2

Simply create another table for phone no. data will be sorted by student

Student

id	Name
1	Shubham
2	Rakesh
3	Kranthi

PhoneNo.

St_id	NO.
1	1234
1	5678
2	0011
2	0022
2	0033
3	____
3	____
3	____
3	____
3	____
3	____

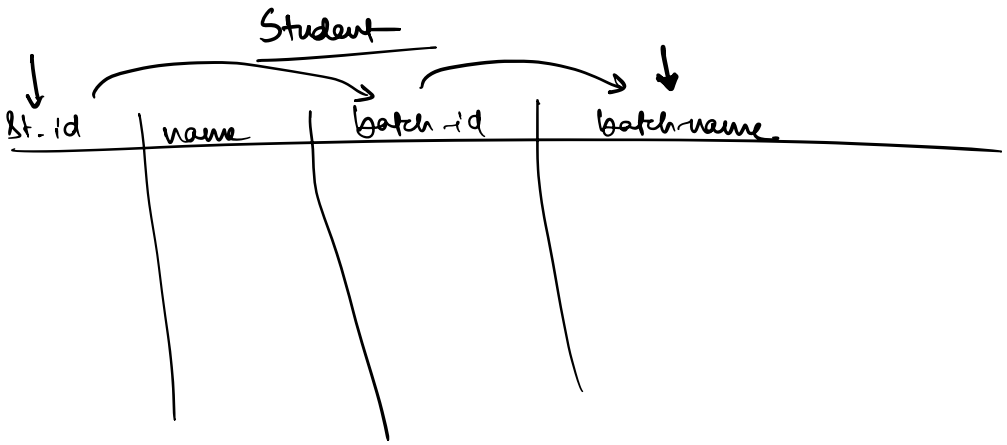
2NF ⇒ Second normalization form

mentor sessions :-

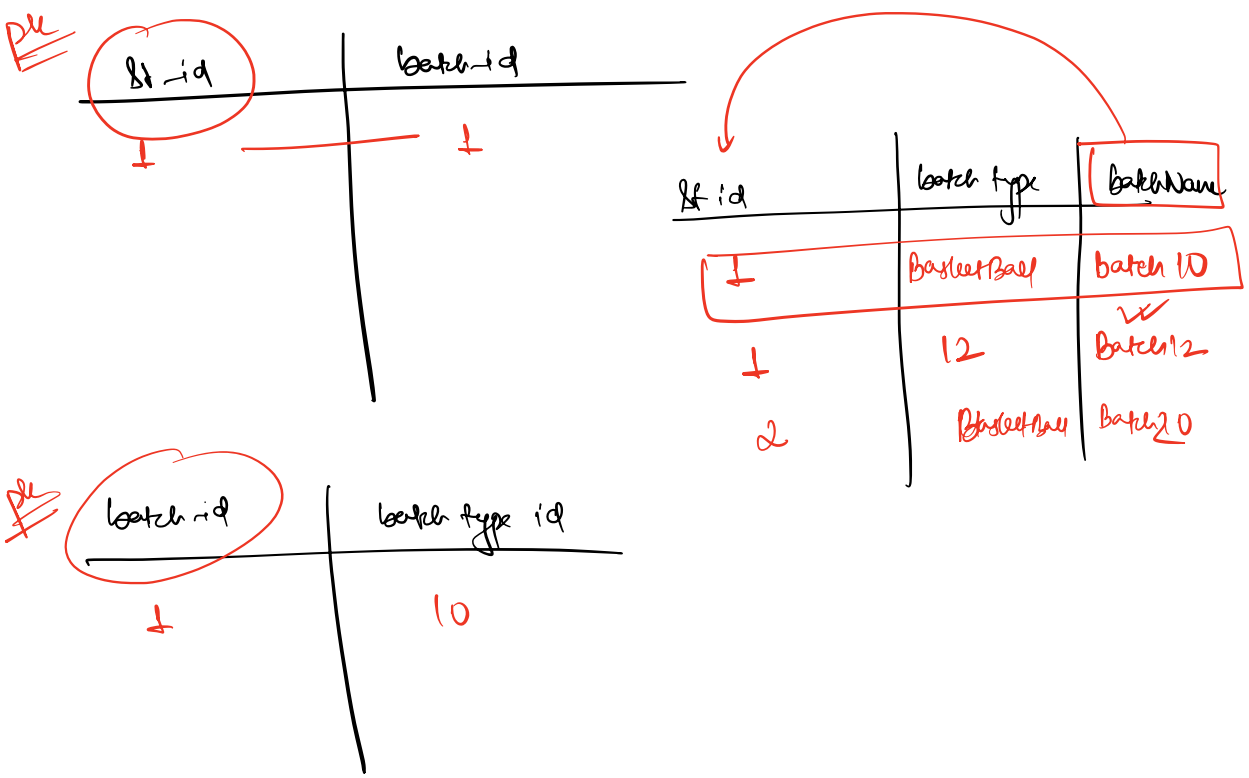
id	st_id	mentor_id	session_date_time	feedback	mentorName

NO non-prime attribute (attribute not part of primary key) should depend on subset

3NF → Third normalization form



find batch name for a student Pd = 1



for a student find the batch name

St_id ⟶ batch_id ⟶ batchtype

St_id ⟶ batchtype_id → batchname

batchname is a non-prime attribute which is indirectly dependent St_id & batch_id

Violation of 3NF

3NF ⇒ no non-prime attribute should indirectly dependent upon pk

Assg

1) Ensure MySQL & ur workBench are ready on your system

2) try to create | update | delete | read ⇒ CRUD operations from tables