Agenda → UUIDs
→ Inheritance in DB

UUIDs

↳ What can be selected as a data type for PK of your table?

int → 4 bytes

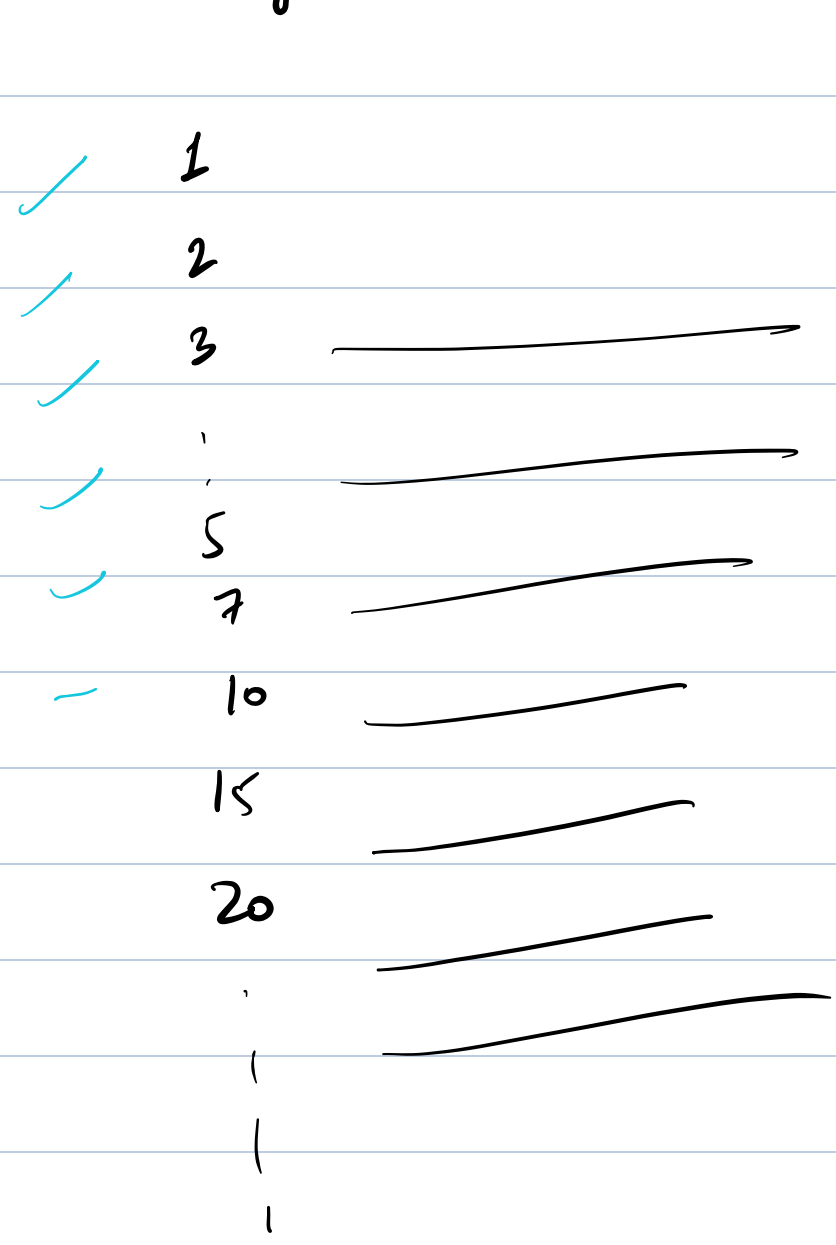
$$[0] - [2^{32} - 1]$$

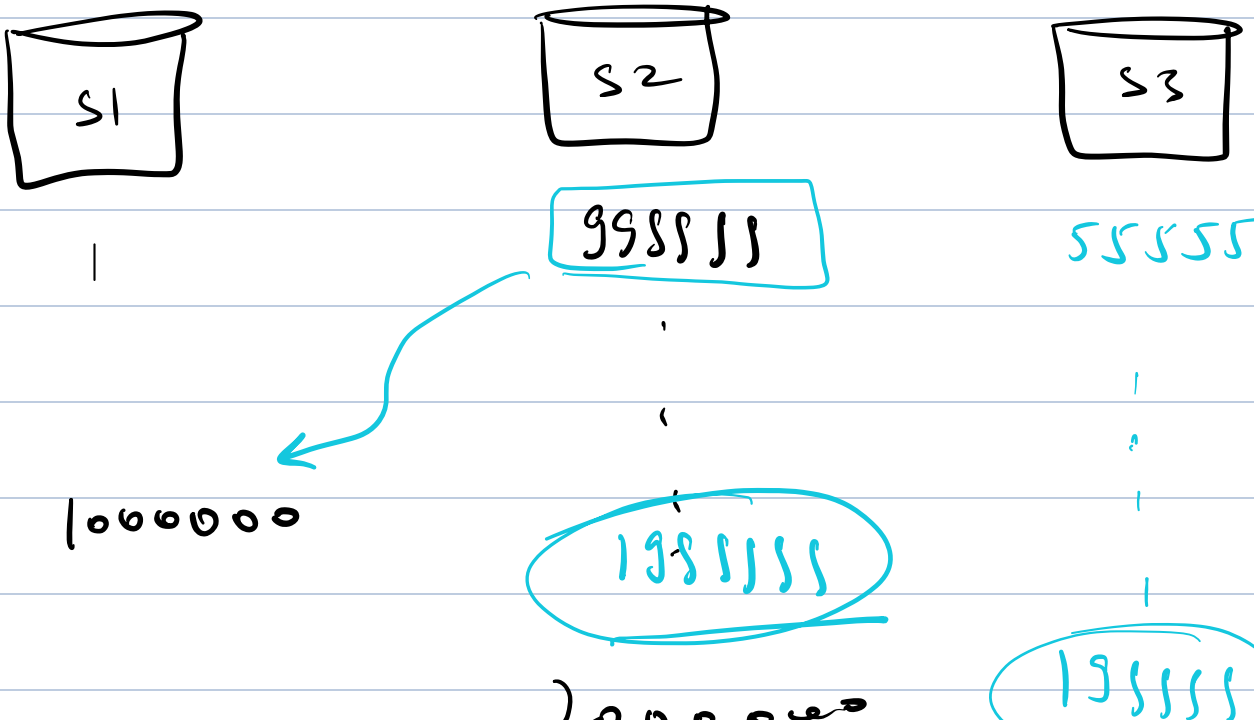
Reason - This range will not be enough

---

long → 10^{18} ✓

Range is fine


1
2
3
5
7
10
15
20



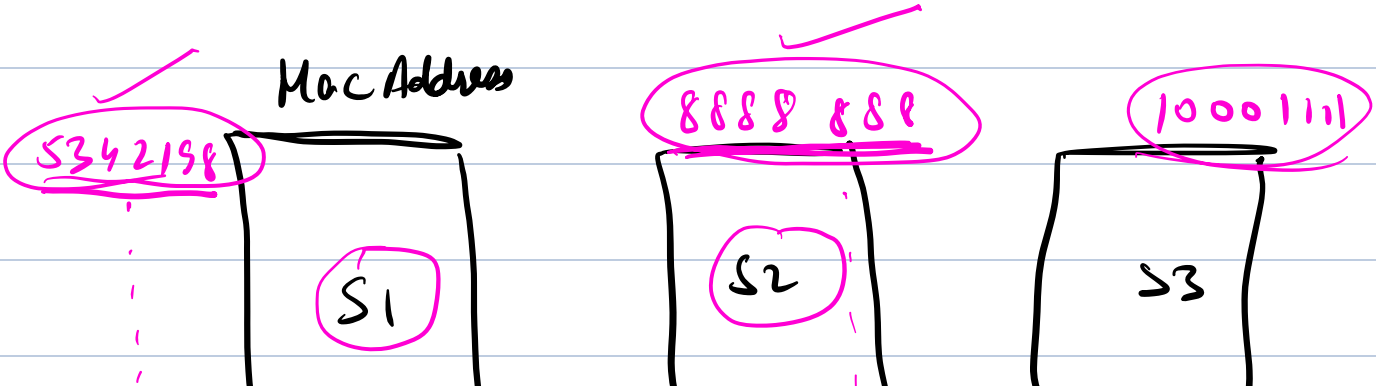
- **Collision** is there which will create ambiguity while reading

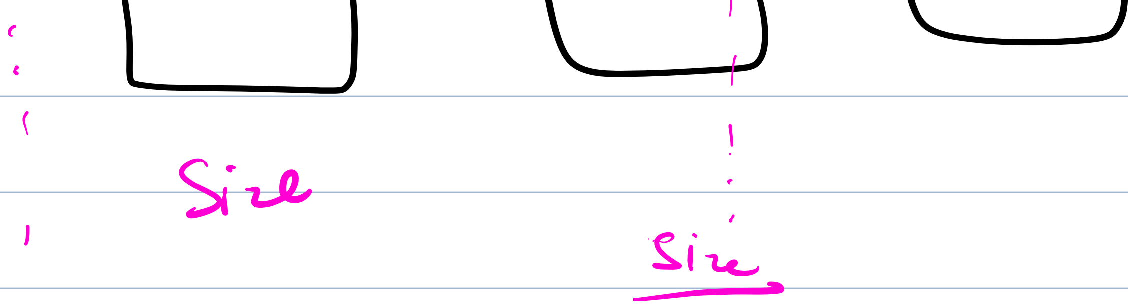
We can go to any youtube video if we have link available

{ youtube.com/10000
  youtube.com/33333

If any person can download these, its problem [security concern]

Mac Address

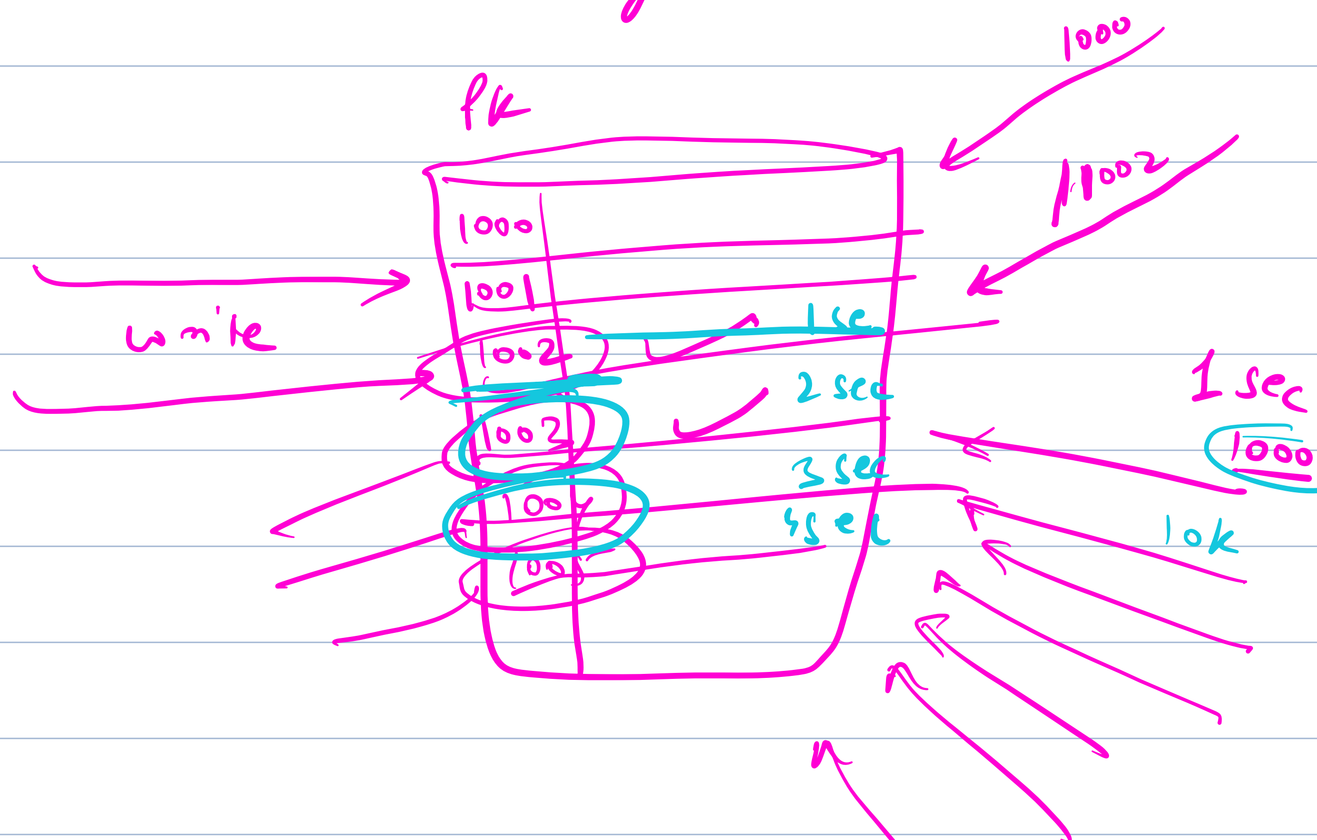




last 534218 + size

1. Problem of collision lil bit improved

2. Mac Addresses are random, so not very straight forward to guess PK & Hence easily download it.



Delays introduced in the system
(locks)

Better Alternative ?

Mac Address + epoch timestamp

1980

YYYY · MM · DD · HH · nn · ss · mmm

UTC

↓

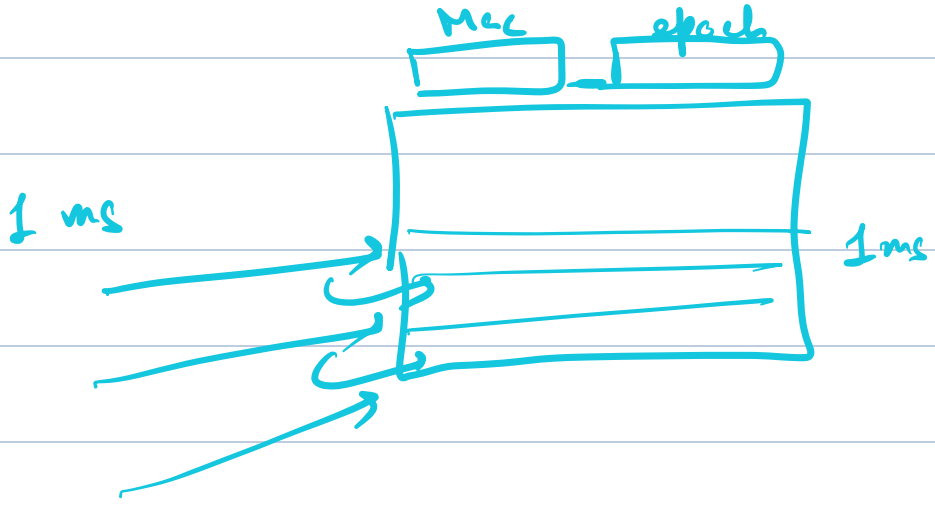
10-12 digit Number

Mac Address + Epoch time stamp

↳ collision ✓

↳ security ✓

↳ Delays (1 ms) ✓



Can we further improve here?

[Mac Address _ Epoch Time stamp _ Random No.]

↳ collision Never Ever

↳ Security (Very High)

↳ No more 1 ms delays because we are having random No. also.

↓

UUIDs

universally unique Identifier

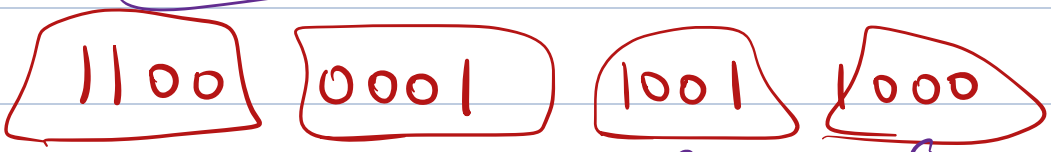
(128 bit)

00011100 001111 000 11000
1000

↳ It would require lot of size if we go with string

↳ Searching, joins will become very expensive if we choose string

HEX ADECIMAL [0-F] 0-9 A-F]16



128 bit = $\frac{128}{4}$ = 32 characters

32 ≪ 128

mid

UUID key;

Guid

Uniqueness is within our System

↓

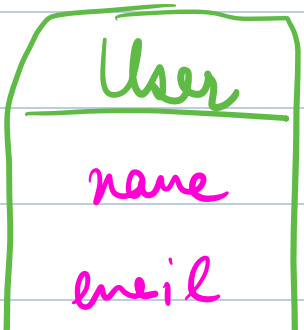
DB System

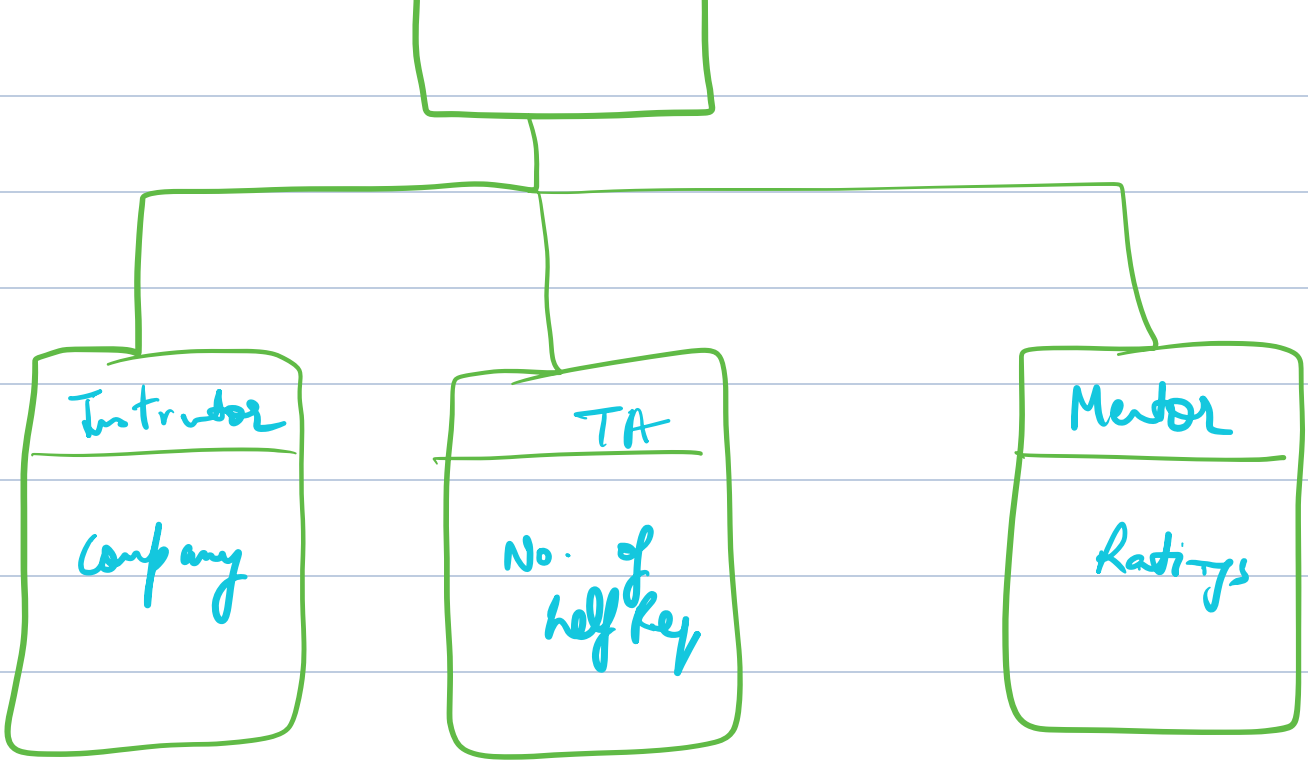
[per Microservice

Representing Inheritance in DB

Instructor
TA
Mentor
] - Contractual Employees

Code





How can we store this relation in DB?

App 1 Table per class

User	name	email	
Instructor	name	email	company
TA	name	email	HelpRequests
Mentor	name	email	Ratings

We are storing data of people who are not TA, not Instructor & not Mentor

Problems →

If we have to get data of all users we need to union

↓

latency & Not opt

No duplicate record

App 2 — Foreign key Relationship (Joined class)

user	name	email
TA	Help Req count	user Id
Instructor	Company	user Id
Mentor	Ratings	user Id

Cons →

→ Join among all tables in order to get details of all users

→ If one of write operation fails,

it can lead to data inconsistency.

App 3 Single Table

user

name	email	Helper	Ratings	Company	user Type
aa	aa@	NULL	NULL	Microsoft	I
bb	bb	52	NULL	NULL	TA

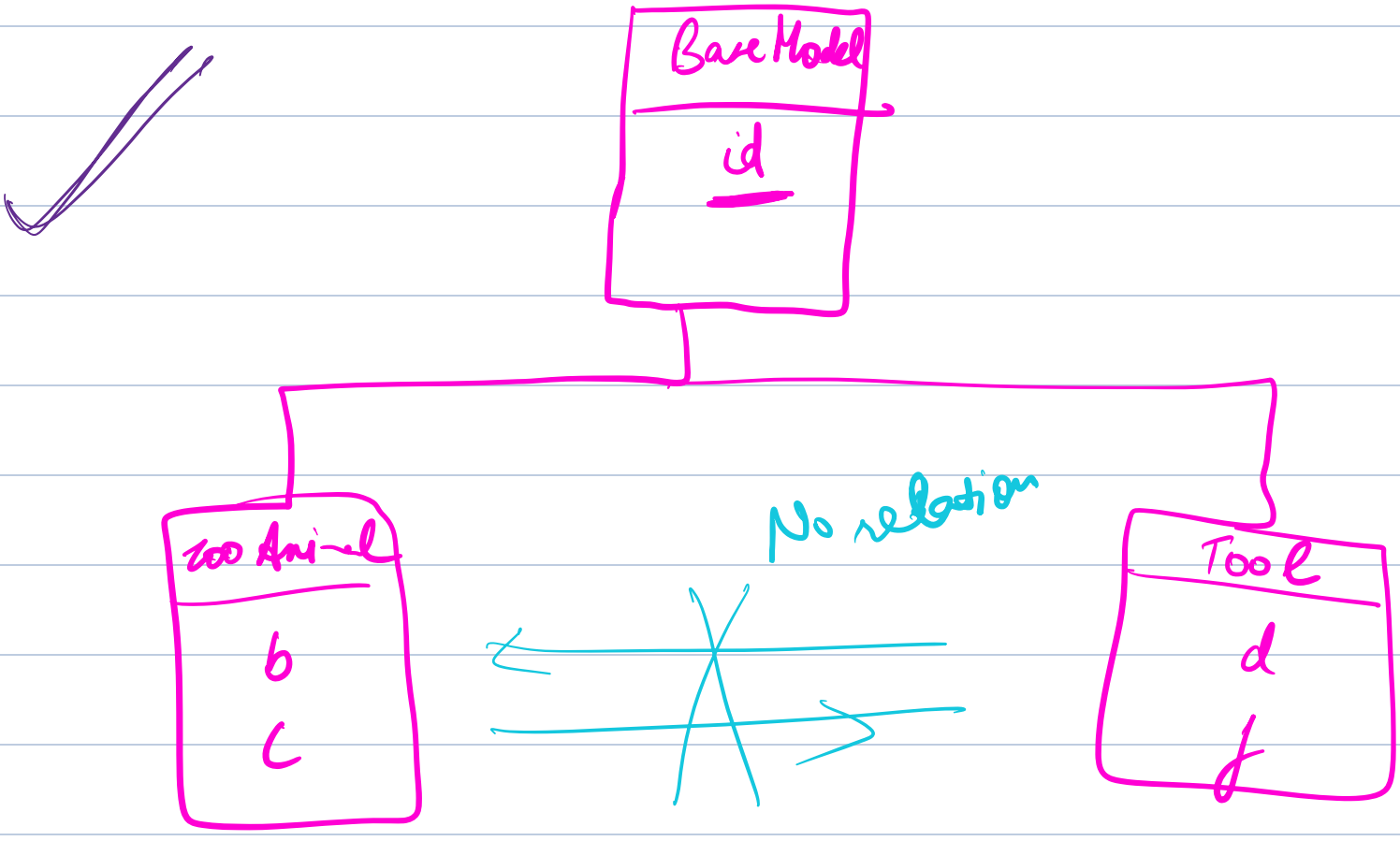
Lot of NULLS, Sparse Table

Memory Waste

Approach - 4 [Only use this approach when you have Base Model]

↓

Mapped Super class



No joins will be needed

zoo Animal Tool	id id	b d	c f

Extra App