# Design Twitter

we know that

people reading >>> people tweeting

Tip

↳ when asking which features we do require to implement, initially ask for those which you feel you can easily implement. Because it wont be possible in a 45 min interview to implement all features.

Let say, our priorities are as follow :-

functional req^m.

1) follow others } Simple
2) Create tweets }
3). View feeds.

↳ we will for now show only those whom we follow.

Also, how to rank the tweets in the feeds ?

# Non-functional reqm.

↳ Total users :- 500 M

Daily active :- 200 M

Storage

↳ Each tweet will have some

- text ( 1000 char) → 1 kB
- image :- 1 MB
- video :- 50 MB

lets avg out to 10 MB.

Now, suppose an active user reads about 100 tweets per day.

Total ⇒ 200 M users X 100 X 10 MB

$\Rightarrow$ 20PB reads / day.

writes

↳ 200 M total users

Out of which

50 M tweets daily.

Total writes $\Rightarrow$ 50 M X 10MB

= 500 TB.

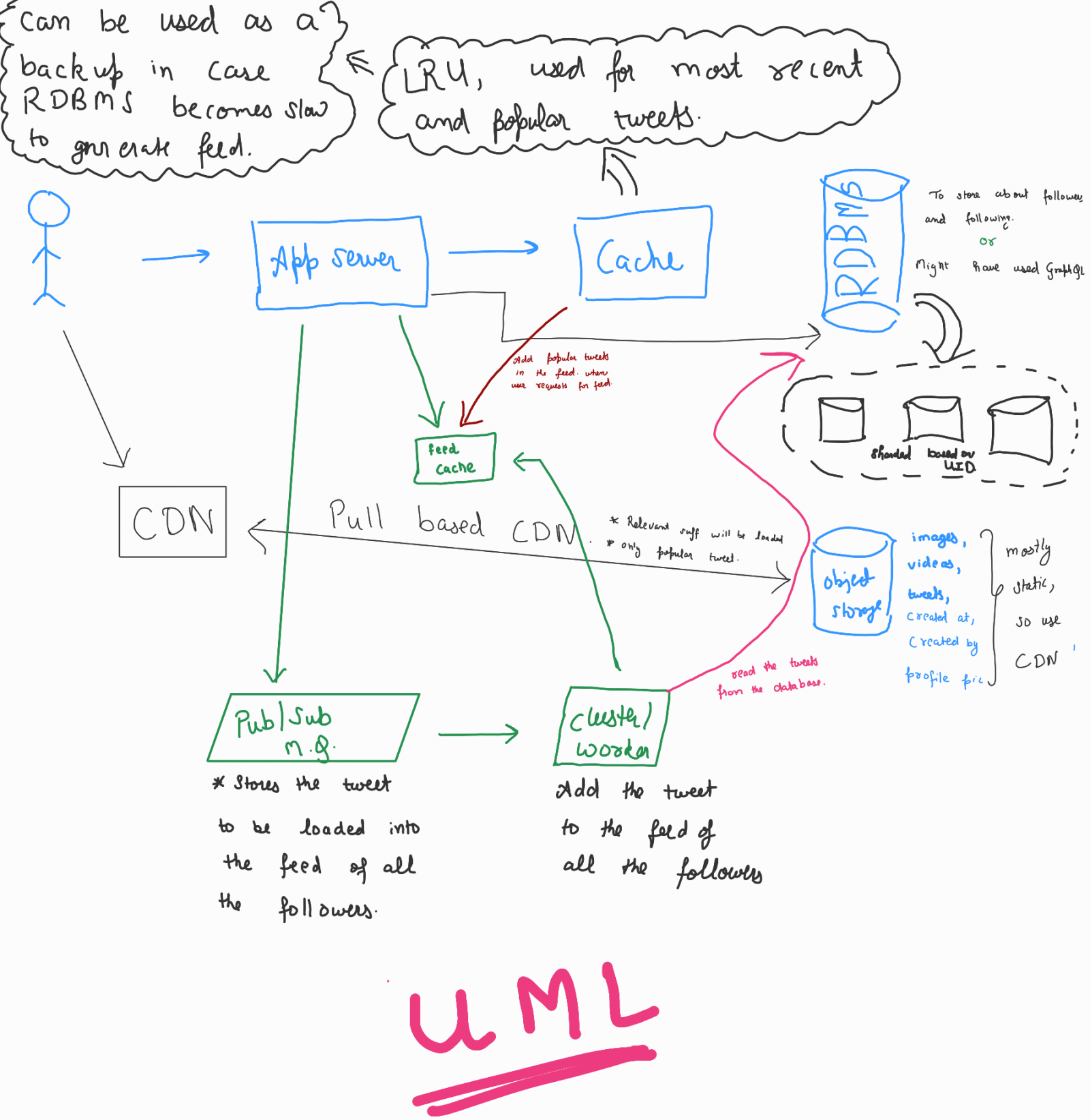
& a user on an average follows about 100 people.

But issue arises with popular people on twitter having

followers > 1 million.

How, we are going to notify them? or update their feed when someone posts!

So, wherever we store their tweets, it is going to overload up quickly. (read)




# UML

create Tweet ( text , media )

↳ Header => UID, Authorization token

↳ Created at, Tweet ID } will be generated at the server side.

get Feeds (UID)

↳ But make sure to check for authorization so that 'A' can't view 'B's feed.

follow (UID, username).

---

# RDBMS

Follow table.

followee : string (UID)

follower : string (UID)

# Since, this table will be used to fetch feed based on the followers.

So, it should be indexed on the basis of 'follower'

Tweet

Time stamp.

UID

Reference to object storage. (for media)

# main problem

we write 50 GB / day
1.5 TB / month
18 TB / year.

& reads >>> writes.

So, we might go with read-only replica's of the RDBMS. One leader will using message queue update the replicas with the data. & worst we get 5 sec stale data.

But issue is on a avg, we have

500 writes / sec which can go upto 10X in the peak hours.

So, we have to scale up the writes as well.

So, we may use sharding. But how to implement sharding?

↳ By using User ID.

↳ Because one particular user worries about only a subset of user base.

## Latency = problem

↳ Suppose tweets load into feed at the rate of 20 (limit).

Now 10 are from cache. (fast)
10 are from storage (sharded) (slow)

⇓

So, at the end our system got slow.

## Soln (Pre-working using async)

Generate the news feed in advance. for all 500 M users. or top 200 M users who are mostly active.

and store it in the 'Feed Cache'.

But this can't be in one memory

⇓

have to shard it.

We are still left with things like pagination and more.

Also this is not how original twitter works.