Chapter 8: Design a URL Shortener.

Functional Requirements

① longURL → ShortURL getShortURL(longURL)

② ShortURL → longURL getLongURL(shortURL).

Additionally if we want to add expiretime, user who added it.

getShortURL(longURL, expiretime, Usermetadata);

Non Functional Requirement :-

① Availability :- Application should be always available

② low latency → we want low wait time of requests

③ Durability → Data gets persisted and data is durable.

Estimation : Here we will decide on the traffic and the length of tinyURL.

* We will serve 1000 req/second.
* We will persist the mapping for 10 year.

⇒ (1000 × 60 × 60 × 24 × 365) = 320 Billion records

(60 × 60: seconds, minutes)

So the String or tinyURL should be so unique that we are able to identify 320 billion records uniquely

Base 62 → [a-z A-Z 0-9] 62 different characters we can use in each place.

$\{62^{\wedge 7} \geq 320 \text{ billion}\}$

So 7 letters we can have the length of tiny URL.

(Data Storage)

Long URL → 100 length (100 bytes).

ShortURL → 7 (7 bytes).

expiryTime → (10 bytes)

User metadata → { User Name
User phoneNo
User location, ⇒ (2KB)
User language
}

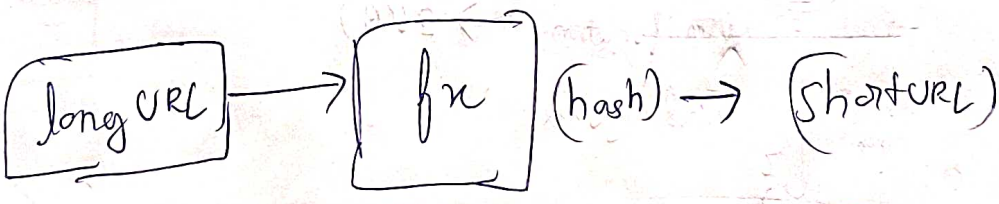
[320 Billion Records × (2KB for each record)]

= 640 GB.

(Deciding on database) [Sql or NoSql]

Properties	SQL	NoSql
Relational	✓	✓
ACID properties.		✓
Durability we ~~dont~~ need,	✗	
Eventual Consistency is fine,		
No hard and fast rule on Consistency		
Analytics we want maybe	✗	✓
Complex queries we don't need	✗	✓
so sql is not winning		

Algorithm.



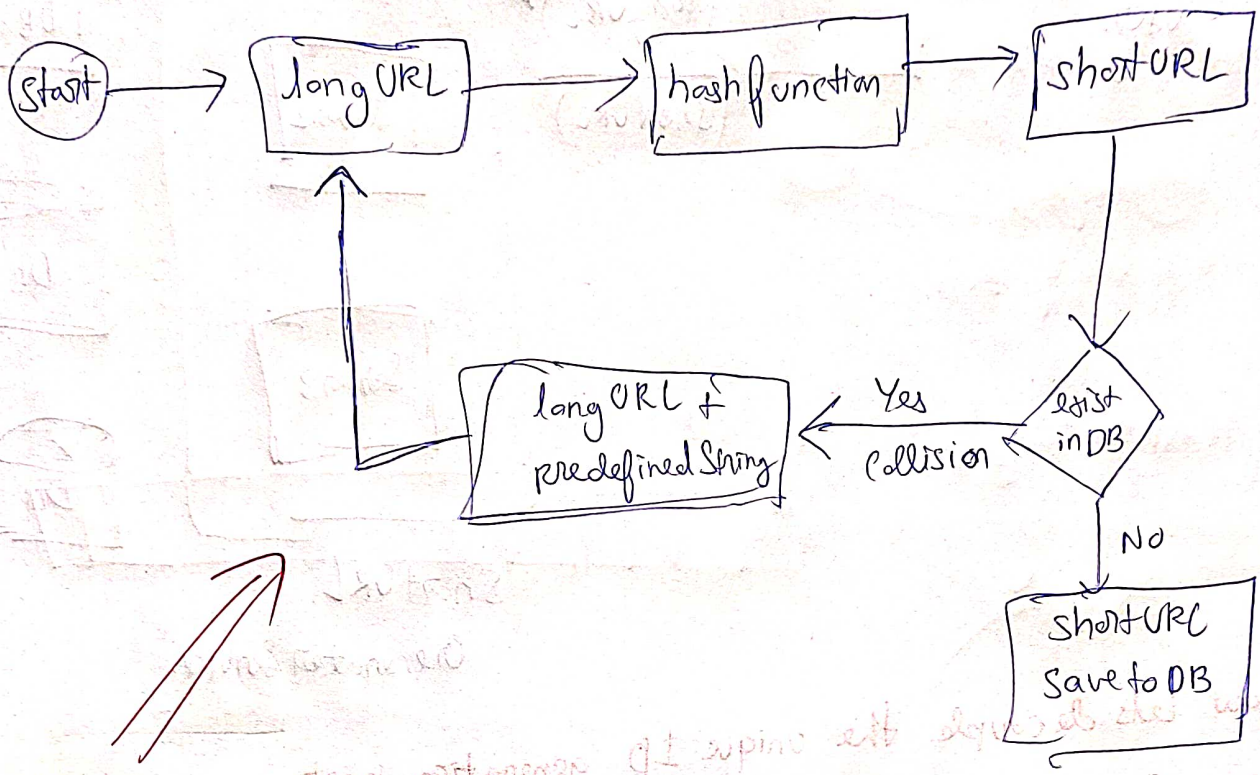
Multiple HashFunctions are available.

MD5 → 21 letters,

if we trim and keep only the first 7, there is a good chance of collision.

To remove collision,

Lets consider a predefined string = "suffix".



Bad Design, very high load on DB and network I/O, making Short URL generation very slow.

Base 62 Hashing Conversion.

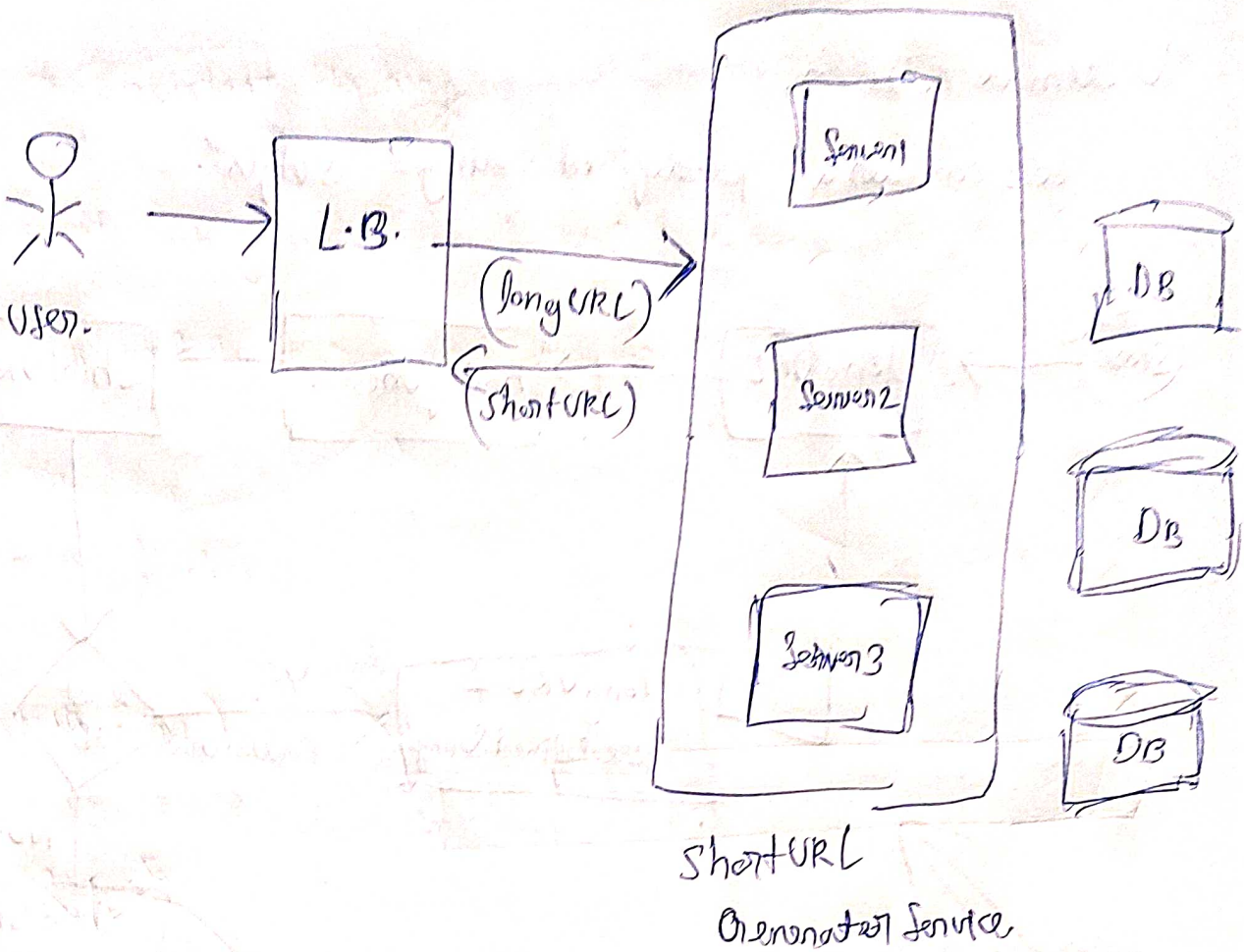
For each request lets consider that we got unique Identifier for it. (Decimal all integers of base 10).

longURL → Get a unique 64 bit decimal number → Use base62 encoding on it to convert to ShortURL.

(0, 61) → [0...9a...zA...Z]

$11517_{10} = 2 \times 62^2 + 55 \times 62^1 + 59 \times 62^0$

$= [2, 55, 59] = [2, T, X]$



Now lets decouple the unique ID generation part.

Remember how we generated in Twitter Snowflake Algo. of Incremental Integer IDs (Chapter 7). [We decouple the ID generation service]