**Behavirol Questios:**

<https://www.youtube.com/watch?v=WEx4HR0FBYQ&list=PLLucmoeZjtMTarjnBcV5qOuAI4lE5ZinV>

**STAR:**

Tell me about a situation where you had to solve a difficult problem

**Situation:**

Machine Learning algo for classification of System Anomaly for PATENT

Didn’t turned for meeting as AdHoc Task

One-to-one meeting: help with communication skill

**Task:**

**Action:**

**Result:** Publish Rating

# Tell me about a time you failed

To address story telling

Master build failed and had to revert as the think went in prod build(**WRONG**)

Make good context to story and make it a big story

**Explain about eGain Communication/ dATAsTRORE FAILED WITH aws CONFIGURATION; ELASTIC ip CHANGED IN aws(LEADER WAVEMAKER DOC ON aws; WENT TO MAINTENANCE WINDOW)**

**Given leadership role and on day of release got failed due to not proper testing**

**Learning: to have though long term view of the role and impact analysis**

**System design:**

**How to approach System Design:**

**Usecase 🡪 Functional - > non functional 🡪 constraints 🡪 bottleneck🡪 HLD 🡪 services(client protocol) 🡪 Load Balancer-🡪 Caching(!00 GB RAM for 500Byte data each request) 🡪 DB(MYSQL vs NOSQL)**

* **Fail Safe system**

**Compare Cache with CPU computation tradeOFF**

**Does cache maintains real time/changing data? Cache only maintains traffic**

**Consider SOLID PRINCIPLE also in design as ; keep concerned separated**

**Also, explain the implemented DataStructure Algotithm**

**Auto Increment Vs features based Sharding**

1. **Design google Map:**

<https://www.youtube.com/watch?v=lOcqPEOROwQ>

queues; used for Unsynchronous tasks vs PIPELINE

<https://github.com/donnemartin/system-design-primer>

1. **Target audience**

Interviewer: Evaluate how to deal ambiguity

Functional and Non Functional Requirements(**Usecase: Youtube**)

**Classify if:** SQL ,NOSQL(Cassandra),Cache(Redis) or stream(Kafka+Sparak) or batch processing(Hadoop and Map Reduce)

**4 categories:**

**Users/Customers**

Who and How system to be used

All Users, for individual users or Marketing

**Scale(Read and write)**

How many reads/sec, requests/sec

Are traffic spikes expected

**Performance**

Read-to-write delay accepted??

If no delay than no batch option but streaming only

**Expected latency for read queries?**

**Cost**

If dev is costly; got fot open source

If maintenance;; go for public cloud domain installation

**Functional Req:**

System behaviour, APIs

Non-Functional:

System qualities as fast, performance, fault tolerance

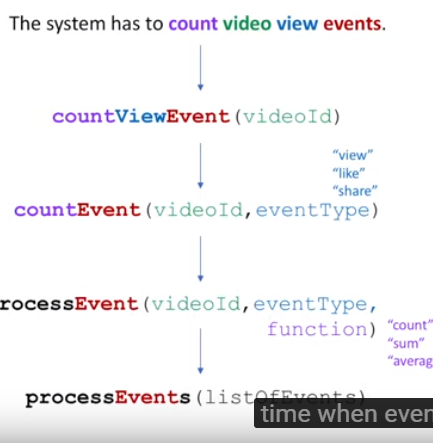
Ask functional req. first.

**Functional req:**

Write points on whiteboard

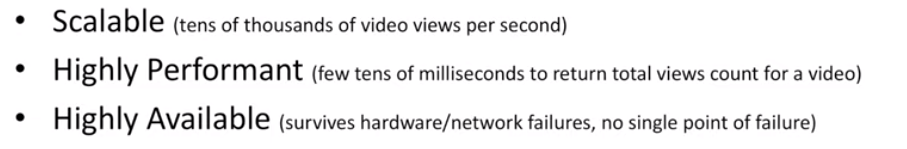
Make the most generalized API:

Identify name and parameters of the method



**NON-Functional:**

**Decise on CAP Theroem :** bwteeen availability and consistency



**STEP2:**

Make high level components and start explaining and drive conversation.

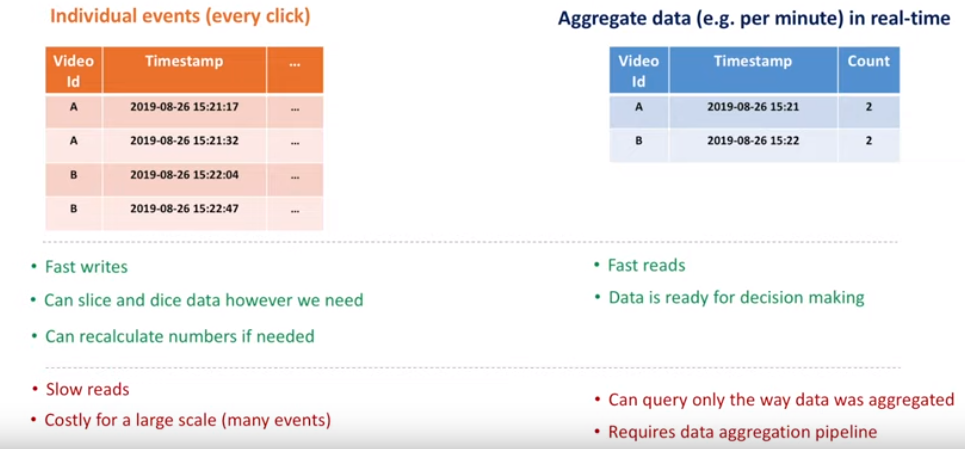
**Step1:** Start with data Model

**Compare the two approached for the persistence:**

**In fig:** First approach is stream processing and Later is Batch Processing

**(ASK INTERVIWER FOR APPROACH)**

**Discuss Both the approaches:**



**Discuss TradeOFFs in BOTH Approaches:**

**Step2: Where to store Data**

Evaluate both SQL and NOSQL

Compare for scalability, availability

**Scale for both read/write**

* **Not to loose data if hardware failure**
* **Strong consistency and tradeOffs**
* **Data security**
* **Recover data in outage**
* **Extensible for data model change in future**
* **Cost**
* **On Premises and cloud data centers**

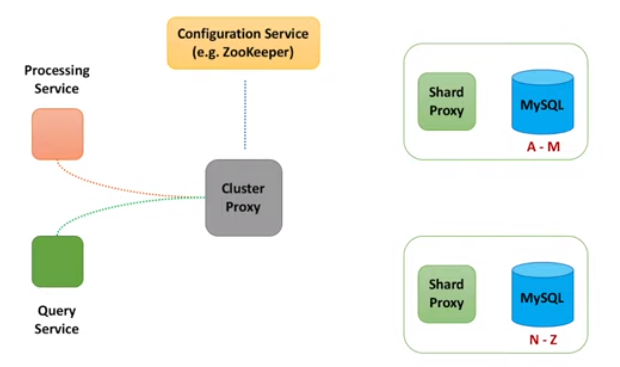
**For MSQL:**

Database sharding ? how it works

Maintain the proxy server/Load balancer

Shard Proxy before Database:

How shard proxy helps??



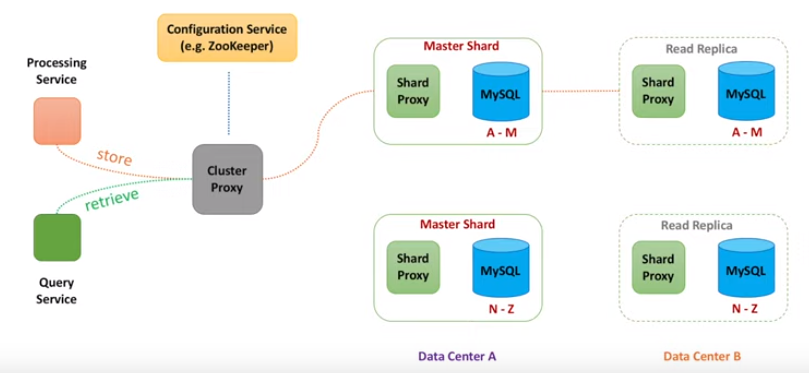
Thus attains scalabilty:

Q: **how to attain availability??**

What if DB shard dies?

Replication

Done synchronously OR unSyncronously



**Alternately, for NO SQL as Cassandra**

**Gossip Protocol;** used to sync data across multiple Nodes

*In SQL approach ; only proxy shard knew about all nodes BUT in NO SQL each node knows about other Nodes*

**Node can be chosen based on Round Robin algo or node closed to client in term of netword distance**

**Uses consistent hashing:**

**Starting node is coordinator Node**

**Has Quorum read and quoram wrte approach**

There might be the stale data as one not misses thus **; eventual Consistency**

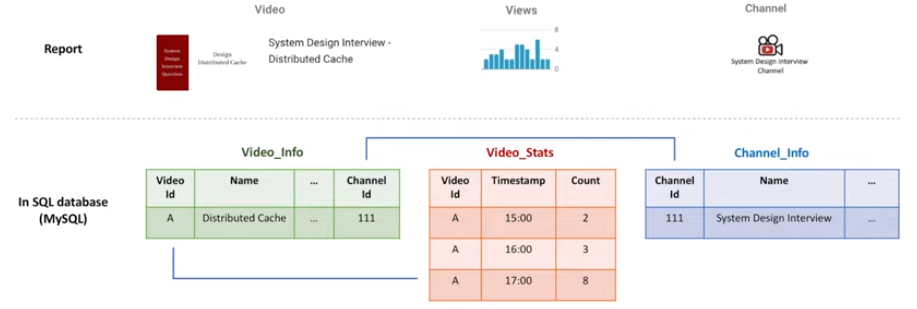
**For Non Functional req: Availabilty dominates consistency**

**As Data is replicated Asyncronously and inconsistency is temporary**

**Step 3: How to store Data**

**Report Building:**

Need join across if using RDBMS:



Advantage: Normalization hence, less data redundancy

**In case of NO SQL:**

Think as query

Add column for every other transaction / hour

Prefer Cassandra as; its best for Real time analysis

**Compare which NOSQL needed for which purpose?**

Types of NOSQL

* Column
* Doc
* Key-value
* Graph

**Data Processing:**

ProcessingService:

Write requirements on White Board:

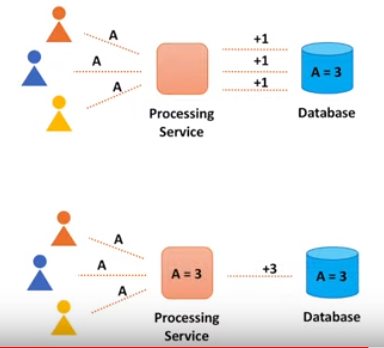
Scalable == portioning

Reliable == replica and checkpoint

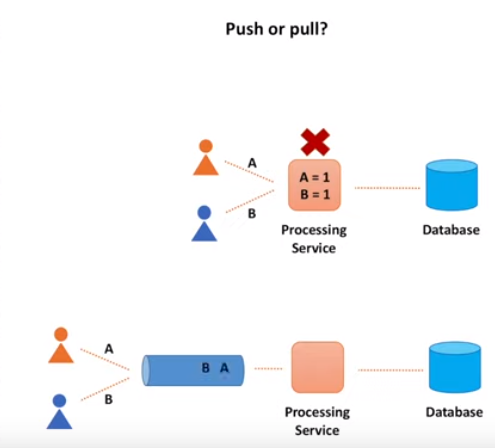
Fast == in memory

**Data Aggregation basics:**

Prefer Second option: In-Memory Counter

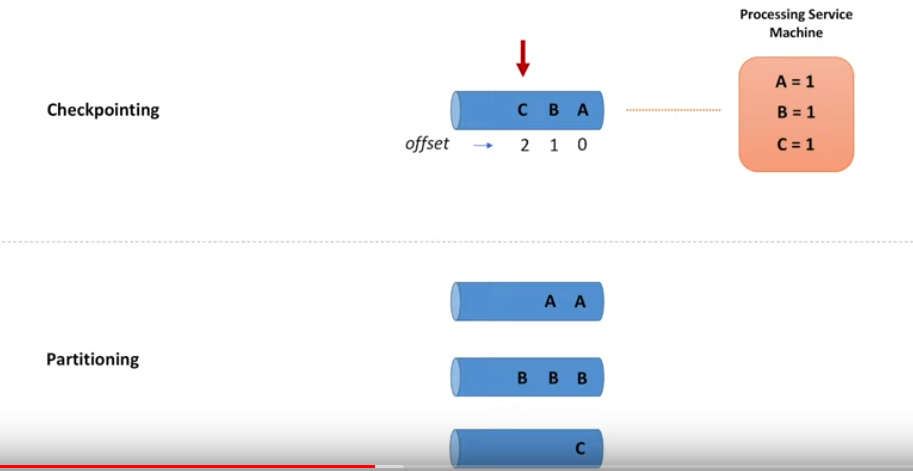


Push the changes from processService or PULL?



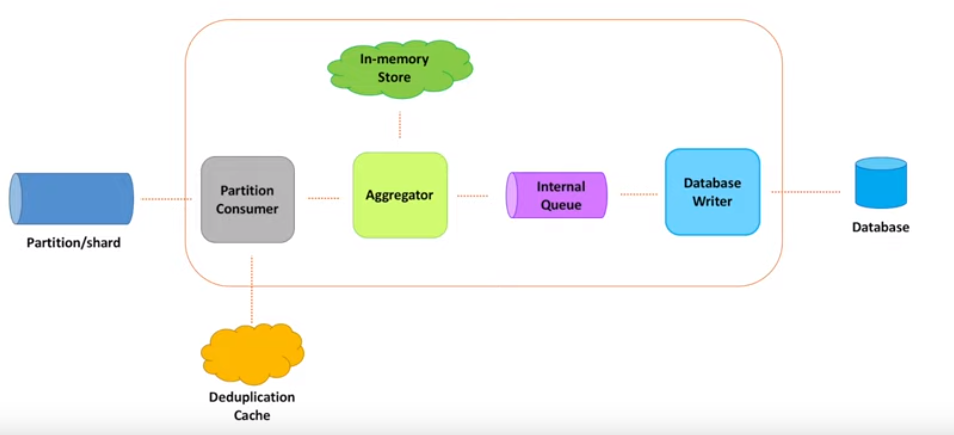
**CheckPointing:**

In StreamData Processing:



Uses Hashing to pick a queue to parse the message

**ThroughPut Vs MultipleThreads???**

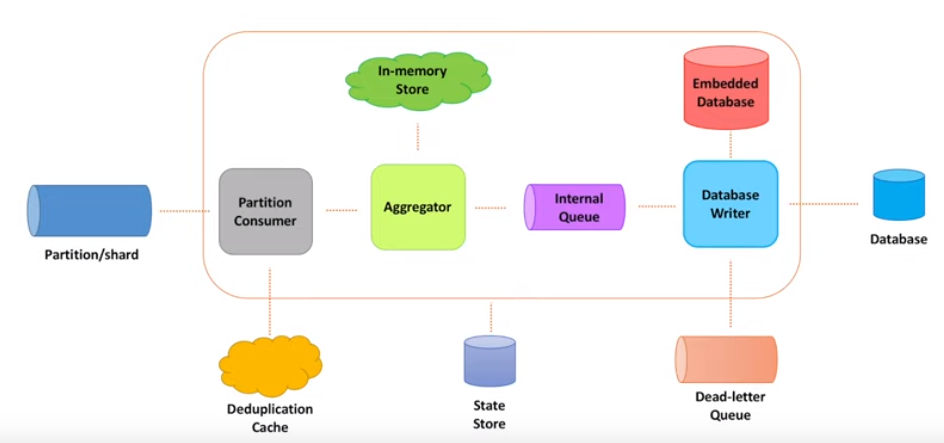


Use of Dead-letter Queue:

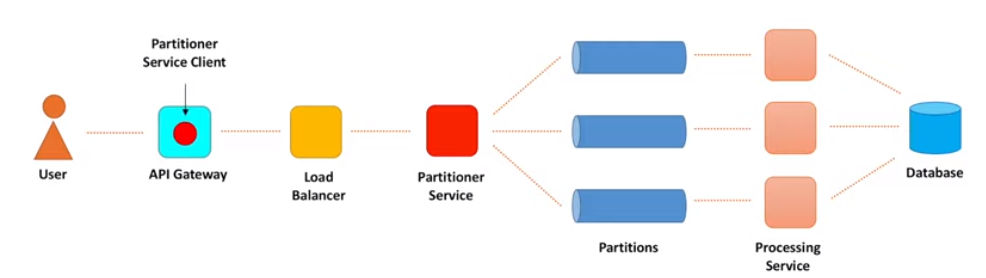
Database in same machine where processService is residing: called Embedded DB; thus lessens the need for remote call

**StateManagement for Objects stored in In-Memory: How to deal iff machine fails?**

**Thus use Store data.**



**Data Ingestion:**



Ingestion path component:

**Blocking I/O Vs Non-Blocking I/O**

The thread create a connection and blocks the socket as well

One thread/connection??

Rate Limiter: Important here

Non-Blocking Systems have: High ThroughPut

Than why so many Blocking I/O system??

Tracking is problematic in Non-Blocking BUT Blocking I/O has thread stack to track progress

And also Unefficient as makes system too huge and complex

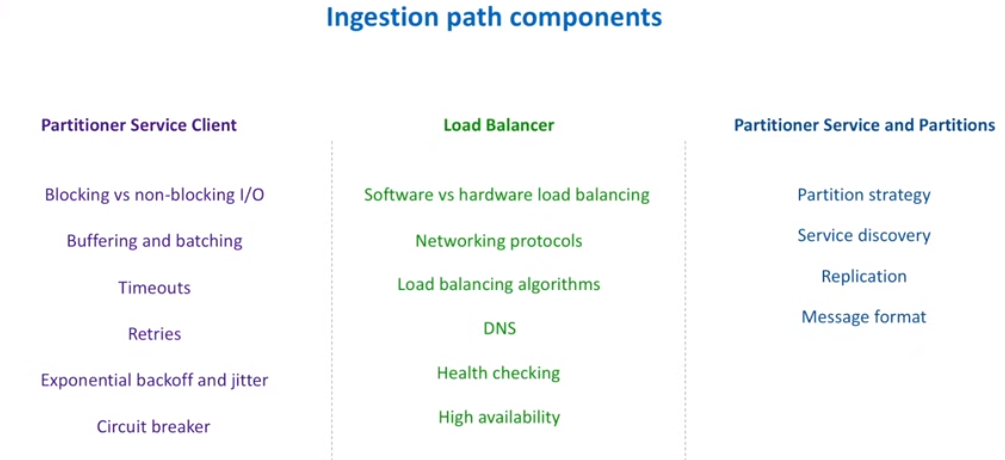
**Buffering vs Batching::???**

**Timeouts:**

Connection timeout: client willing to wait for connection to get established; time is usually too low < 3 ms

Request Timeout: when request process takes time; need to evaluate latency %tile(as 1 out of 1000 req. will get timeout); option is to Re-Try

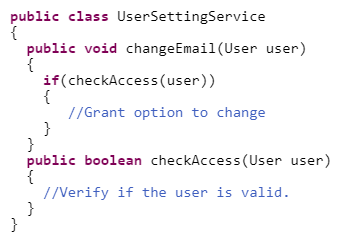
**Exponential Backoff time and jitter: value to fix min. time between re-tries**



**DISTRIBUTED CACHE:**

**SOLID PRINCIPLE:**

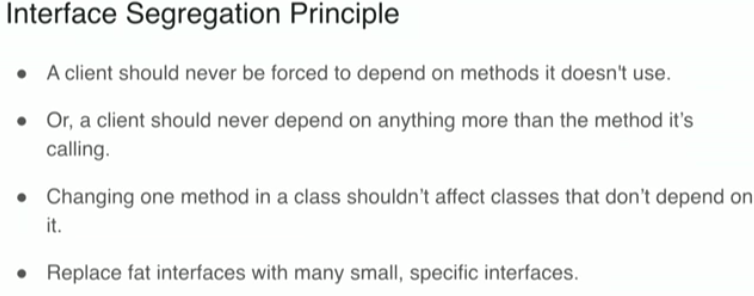
**Single responsibility:**



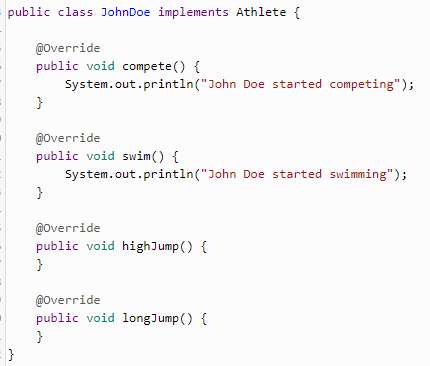
**Open and closed:**

Code to interface; so no if-else rather extend to create new class instead

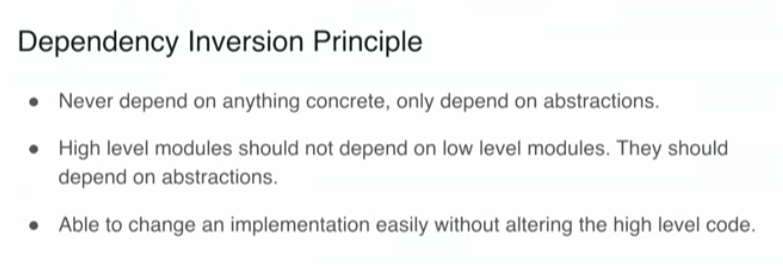
**Interface segregation Principle:**

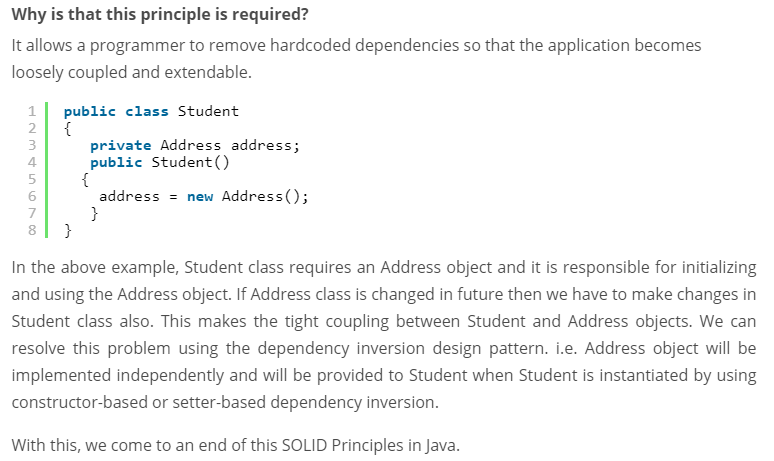


Unnessaray forced implementation

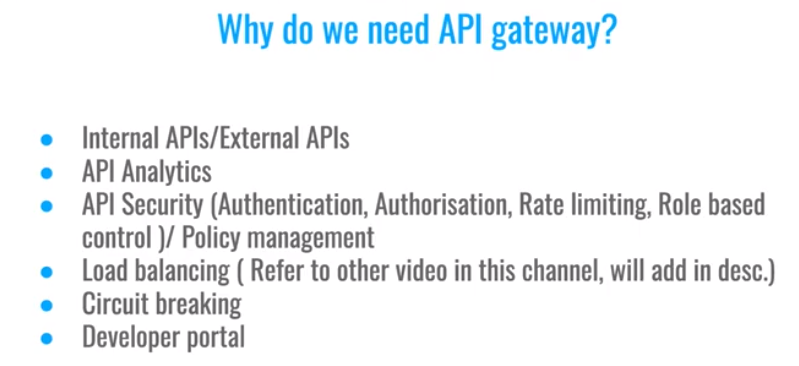


Dependency Inversion

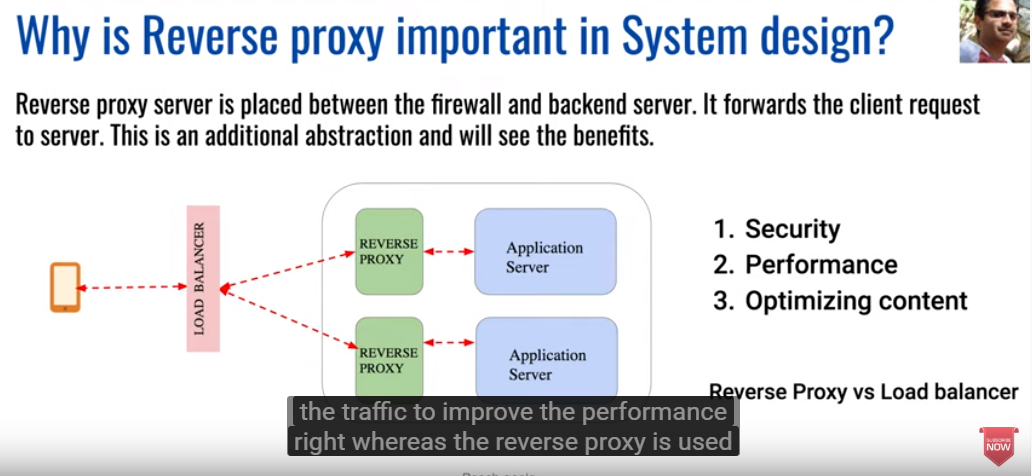


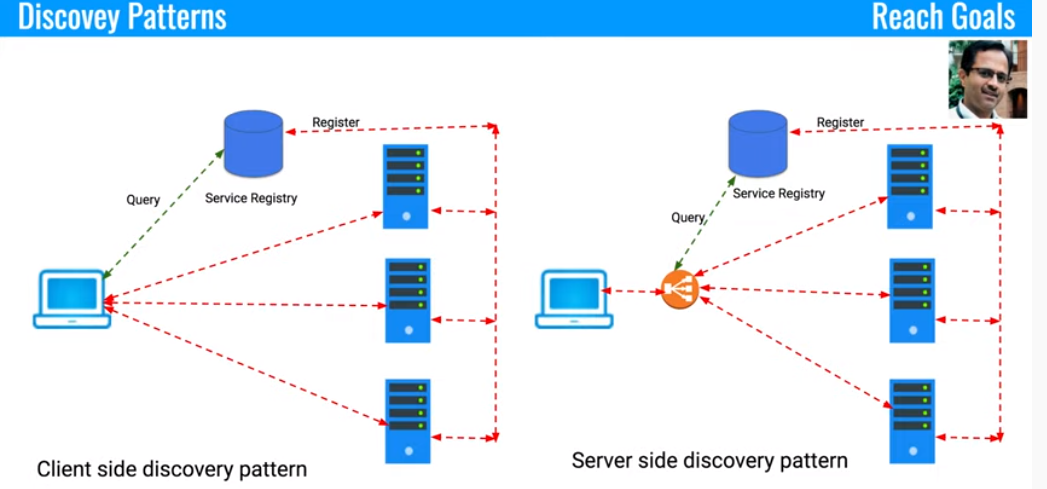


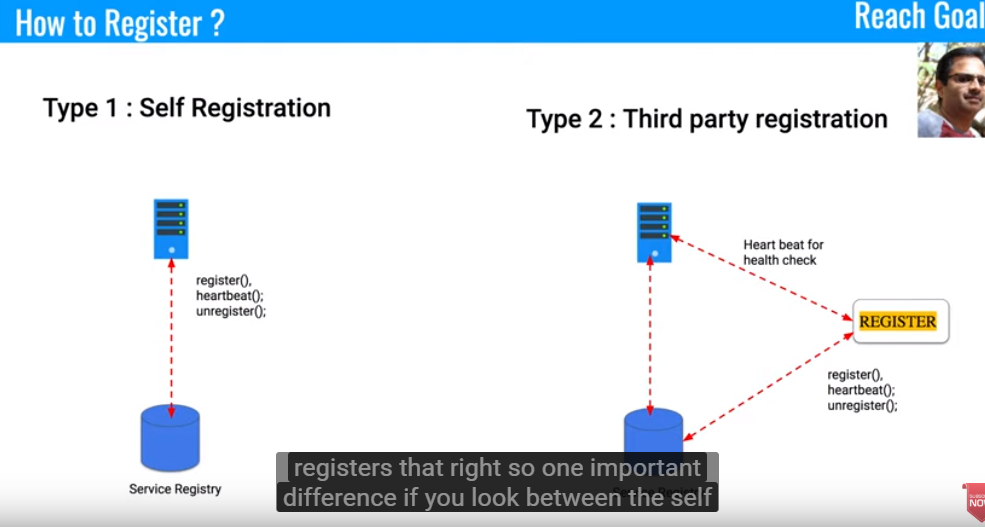
**API Gateway:**



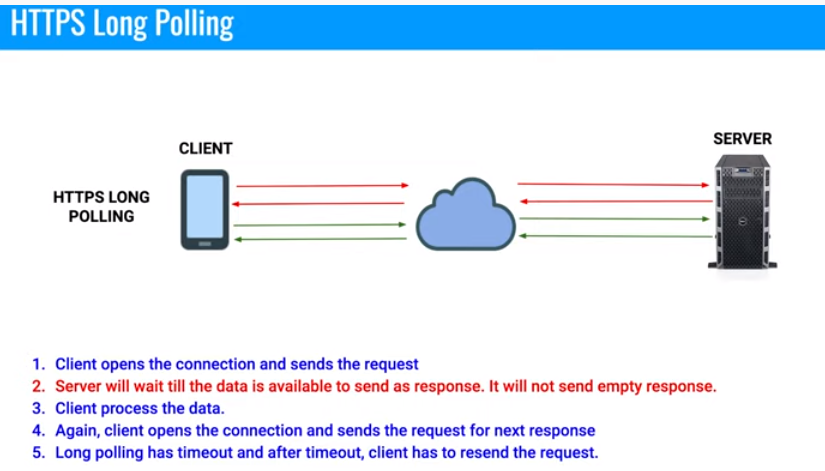
**Reverse Proxy:**

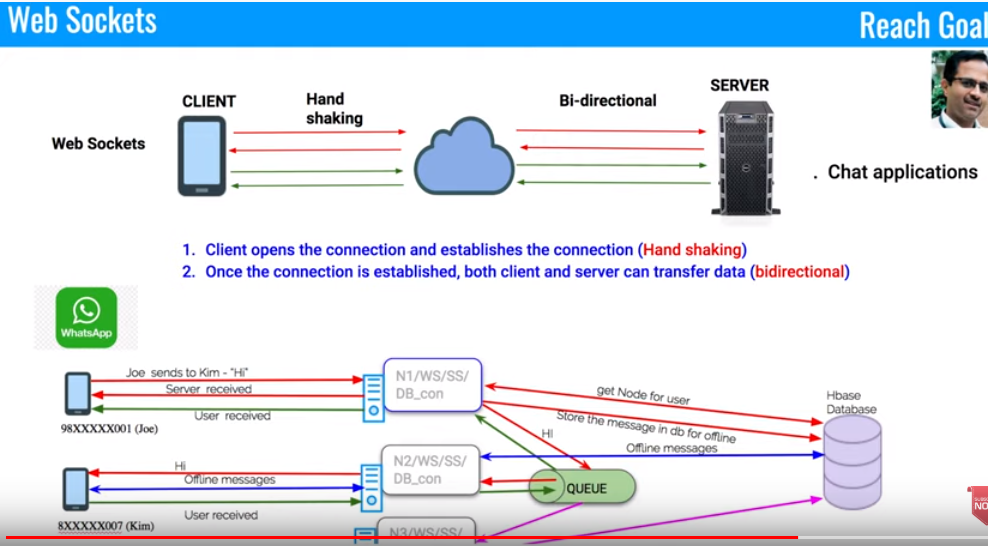




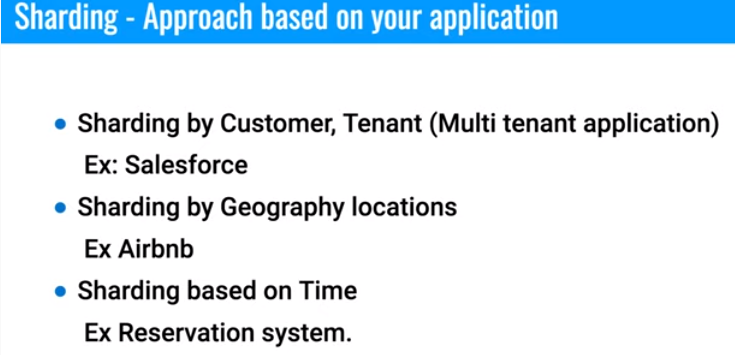


**HTTP Polling:**

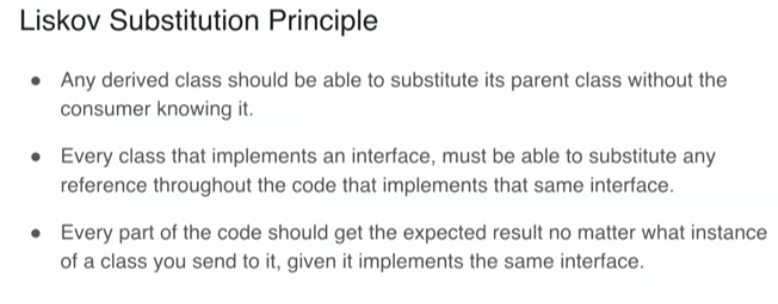




**DataBase Sharding:**

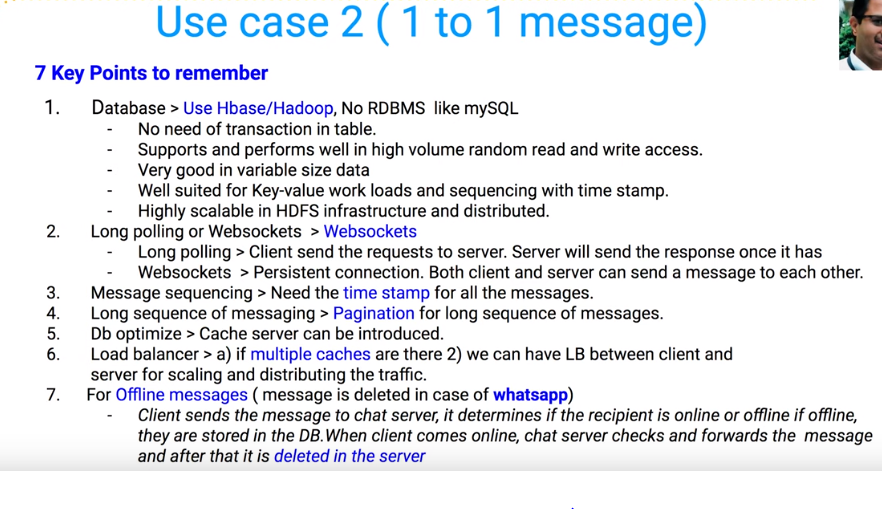


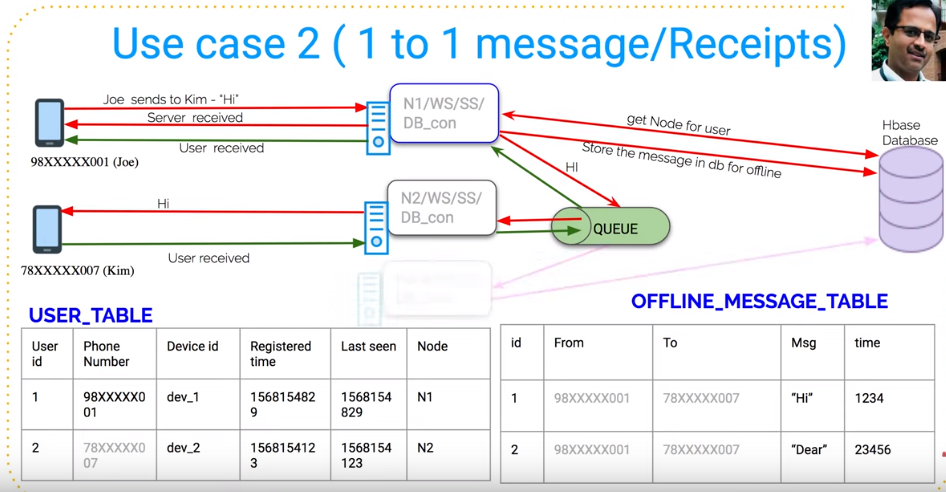
**SOLID PRINCIPLE:**

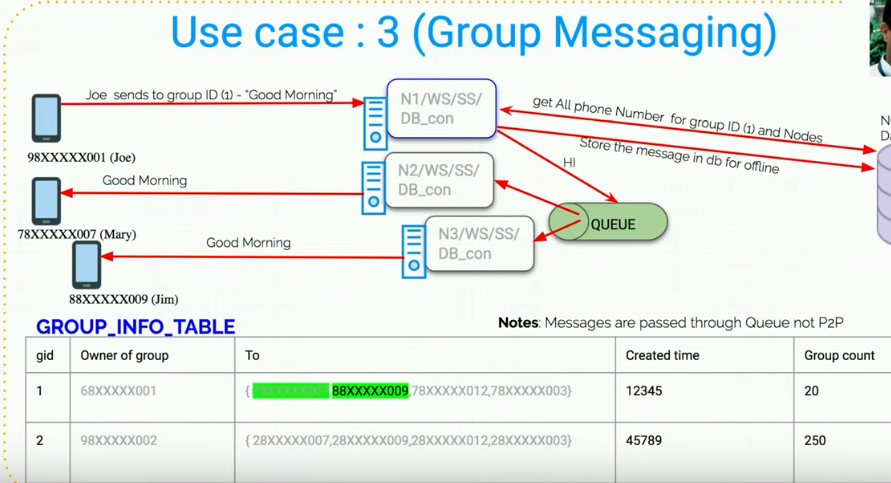


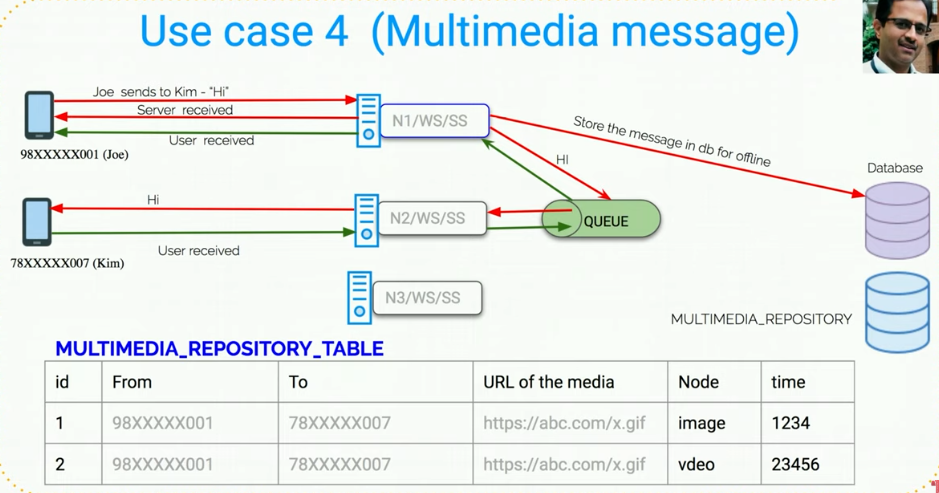
**Circuit Breakers:**

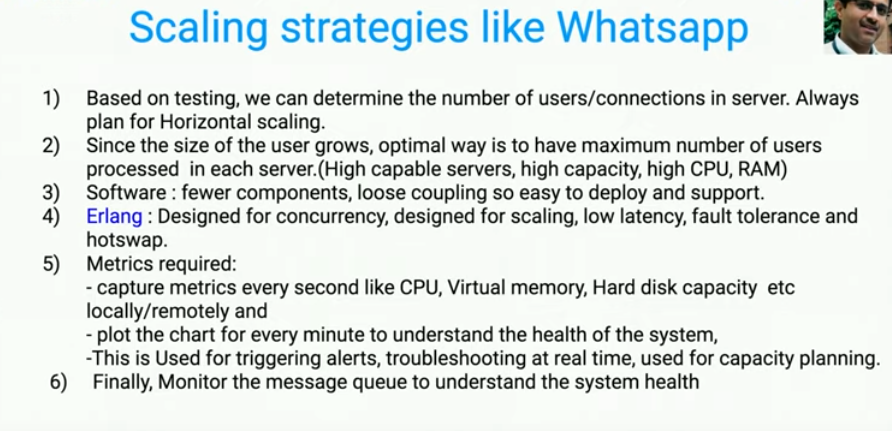
**Design whatssup:**



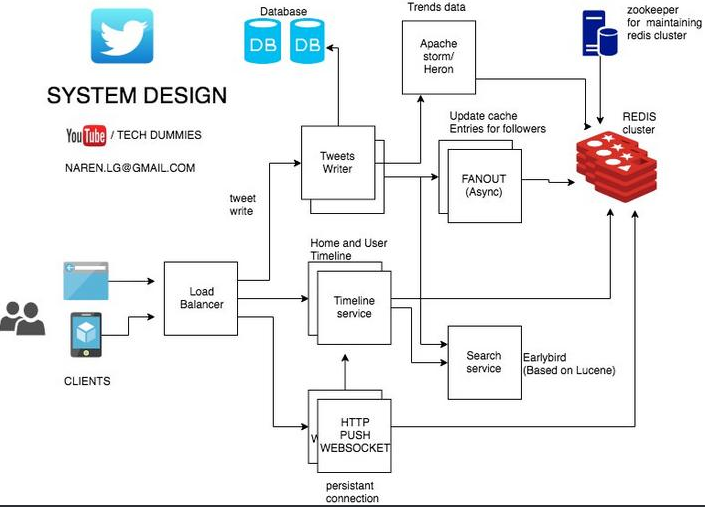








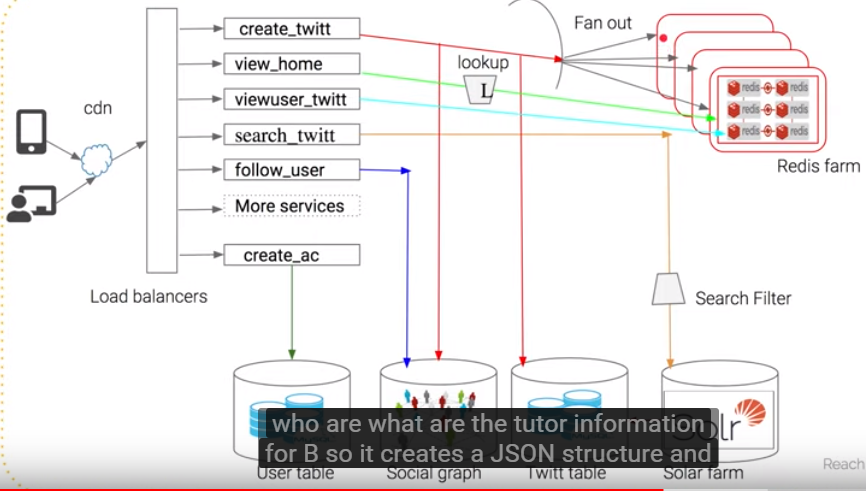
**Twitter System Design:**

****



**Solr Farm or Elastic Search**





**Features:**

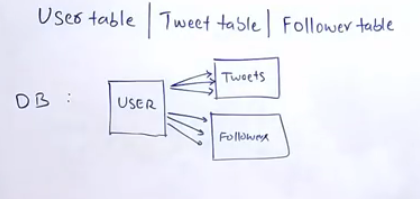
* **Tweet**
* **Timeline : Home timeline / User timeline / Search Timeline**
* **Trends**

Read heavy

**Eventual consistency**: consistency not major factor

Storage cost??

User Table | Tweets Table | Follower Table



Caching layer needed for a read heavy Twitter

In redis the structure to be as :

User\_ID : USER\_treet IDS[1,2,3………….] as list

Tweet\_ID: User\_id

BottleNeck is multiple DB queries: Alternatively: NoSQL/Nodes

Data need to create the Hometimeline? Using **FANOUT** to push the tweet made by person to his followers into there individual Home Timeline Cache

Get the tweets from the Cache rather than from Database based on User ID

how much latency is acceptable in database query?

How much fast does a Cache makes the operation as compared to DB query?

Also maintain HomeTimeLine in Redis Cache

Do we need to precompile the HT and UT for user who haven’t logged to their twitter account?

No,We can avoid the processing

**#Trends:**

*Streaming????*

Stream processing Framwork as **Apache Storm**

**Or Kakfa**

ZooKeeper??

Use cache against distributed DB / nodes using FANOUT

Redis a cache db??

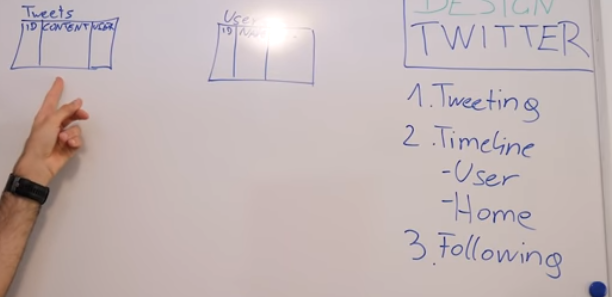
API are exposed form Redis to be consumed further

**In memory database?? How Redis maintains persistence of data when machine crashes**

Trends: volume of data/ **stream processing: storm or kafka**

Validation

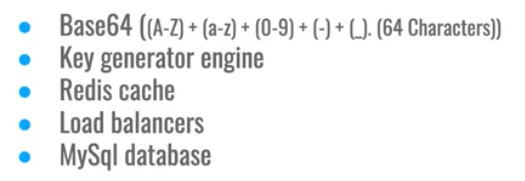
Indexinf for seach/ lucene

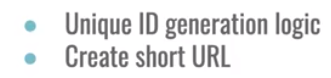


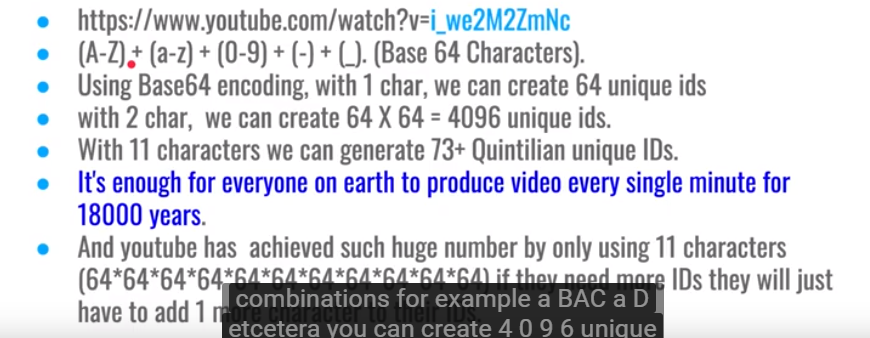
**Sending tweets**

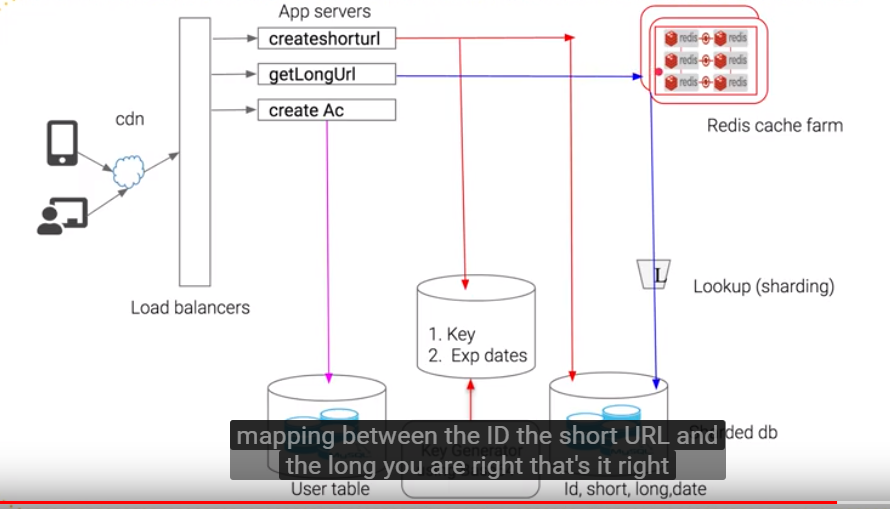
**Eventual consistency:**

**URL Shortener:**









**DataBase sharding: Based on expiration date**

**Tiny URL**

1. Load of requests
2. Character of bytes for short url
3. 1 million user / month
4. Mapping between Actual Url and tiny URL
5. Prequeisite:
6. The size of data to be inserted into Database:
7. Called :DataCapacity Model

**Details to persist in database:**

**LongURL | ShortURL | created time(epoch) | expiry at**

**The above is entry of data per URL Shortener**

Now, to decide the database: DBMS or NOSQL need to understand the number of read/ write operations

The TinyURL have a ***domain.name/uniqueCode***

The Unique Code is generated:

Base 62 / Md5 Hash

MD5 takes string as input and might give outputs with collision

So use Base 62 and Base 10

As Base 62 has values from A\_ Z , a-z and 0-9

Hence, 62 pow 7 will give 3.5 trillion combinations of UniqueIds

**Database:**

RDBMS: ACID but not scalable as per month 30 Million read/write

NO SQL: Highly Available, scalable BUT *Eventual Consistency*

Use base 62 libraray to convert ramdomnumber to base 62

<https://www.youtube.com/watch?v=bUHFg8CZFws>

BEHAVIROL QUESTIONS:

<https://leetcode.com/discuss/interview-experience/468811/Amazon-or-SDE2-or-Seattle-or-Dec-2019-Offer>

Tell me about a time when you had to work on something after you have delivered it  
How did you handle a tight deadline?  
What aspects would you improve if you had to do it again?  
Tell me about a time you had to handle conflict  
Tell me about a time you disagreed with your team  
Tell me about a time when you went out of your way in your company

<https://leetcode.com/problems/group-anagrams/>

<https://github.com/donnemartin/system-design-primer>

**>>>>>>>>>>>>>>>>>>>>>>>>**

**LEETCODE DESIGN QUESTIONS:**

Design a system that checks the health of 10,000 nodes. Each of 10,000 modes have an API exposed that you have to call and check the health. This API takes 1-5 seconds to respond. There is UI which tells which nodes are healthy OR not healthy. Your system should provide API for this UI. This UI refreshes every 10 mins. So, every 10 mins your system should be able to update data about those 10,000 nodes