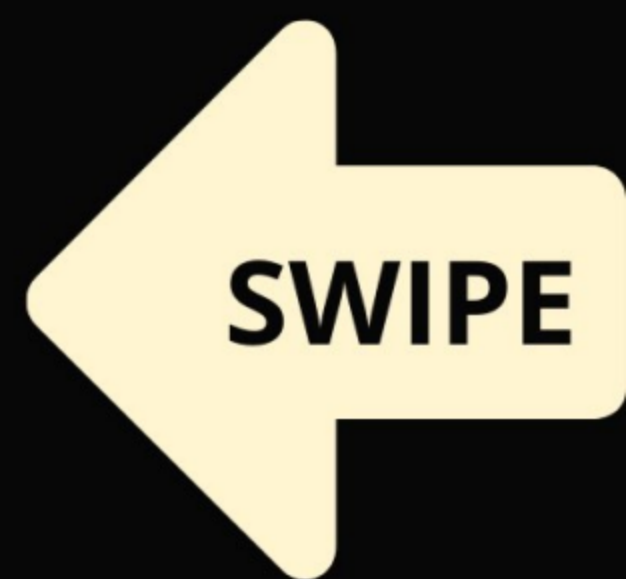




**#ASLI ENGINEERING**

# Database Scaling Techniques



**BY**

**ARPIT BHAYANI**

## Scaling Databases

Databases are the most important component of any system out there. It makes or breaks any system...



Hence, it is critical to understand how to scale them...

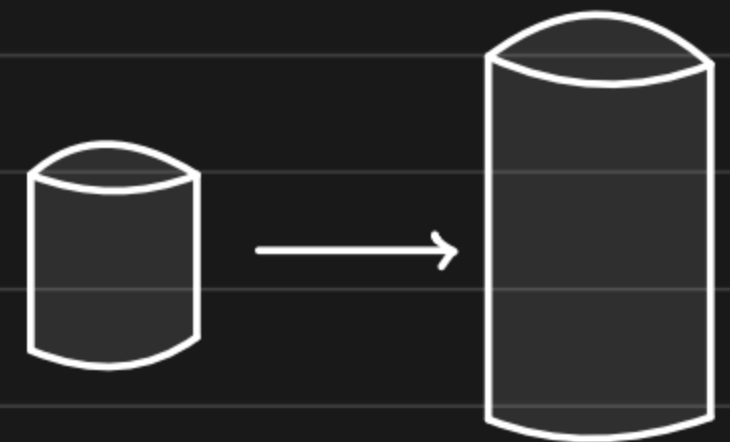
\* These techniques are applicable to most databases out there.

read your DB documentation.

↓  
relational and non-relational

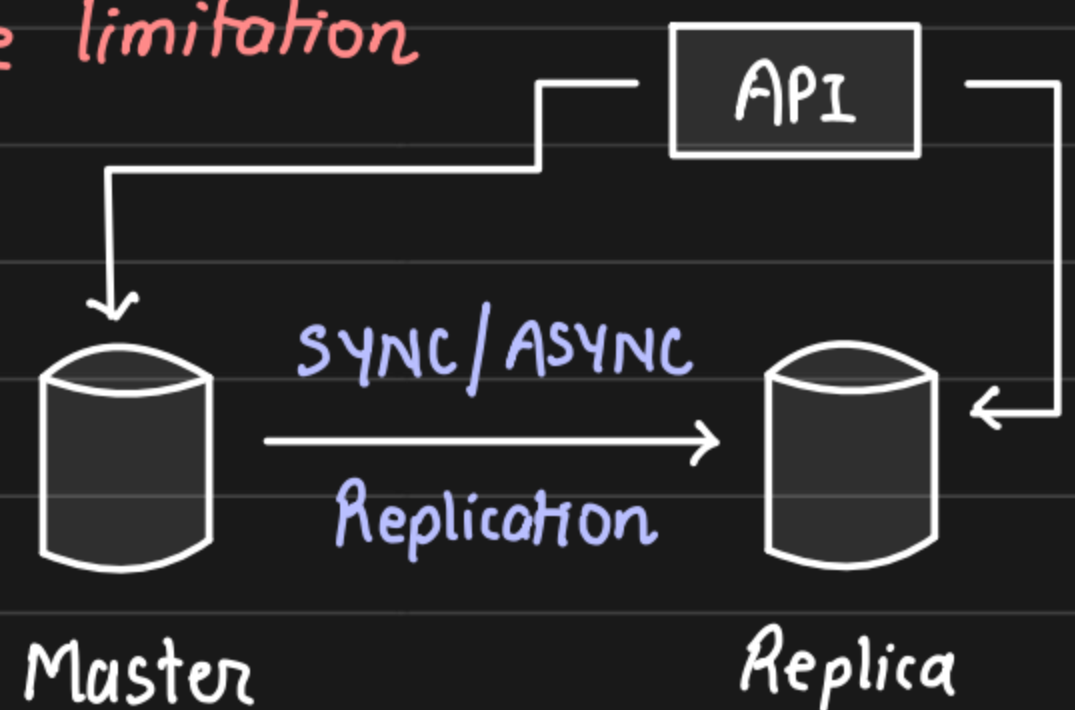
### Vertical Scaling

- add more CPU, RAM, Disk to the database
- requires downtime during reboot
- gives you ability to handle "scale", more load
- vertical scaling has a physical hardware limitation



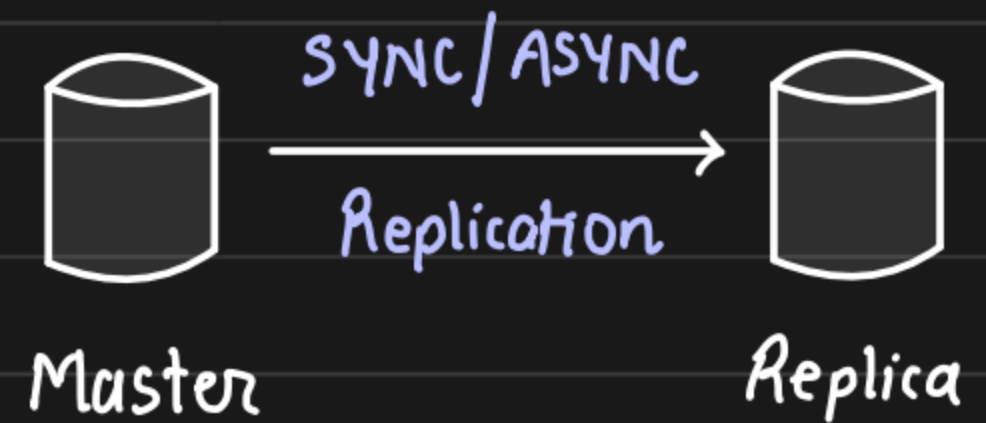
### Horizontal Scaling : Read Replicas

- when read : write = 90 : 10
- you move reads to other database so that "master" is free to do writes
- API servers should know which DB to connect to get things done



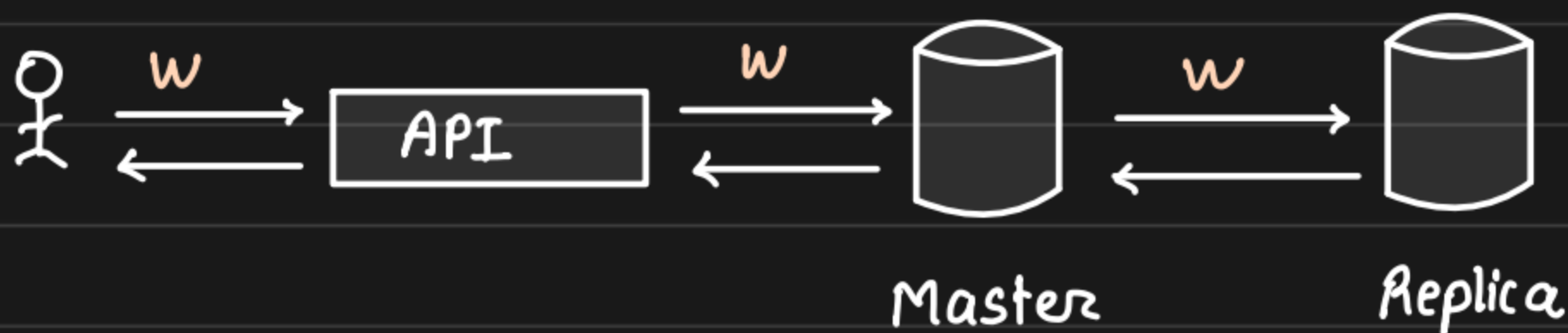
## Replication

Changes on one database (Master) needs to be sent to Replica to maintain consistency



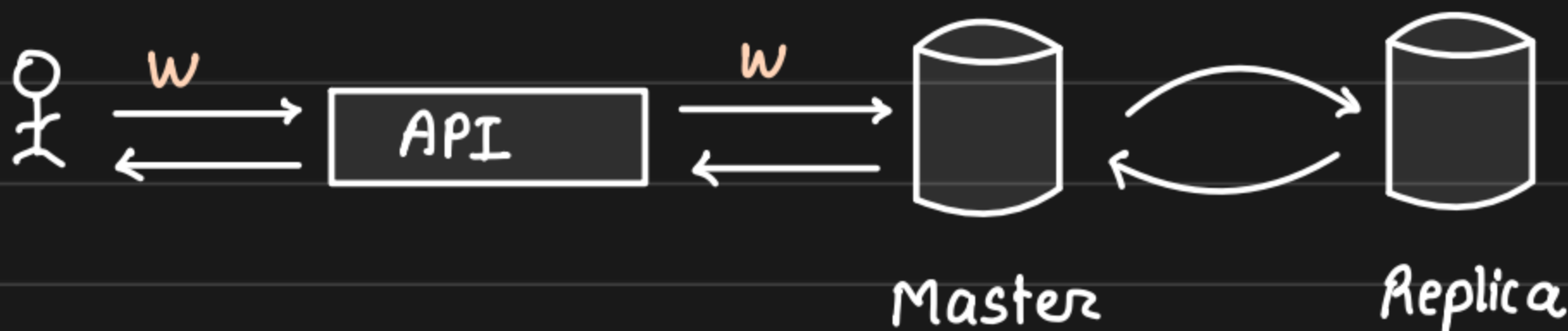
### Two modes of replication

#### 1. Synchronous replication



- Strong consistency
- zero replication lag
- slower writes

#### 2. Asynchronous replication



- eventual consistency
- some replication lag
- faster writes

## Horizontal Scaling : Sharding

Because one node cannot handle the data/load

we split it into multiple exclusive subsets

Writes on a particular row/document will go to one particular shard.

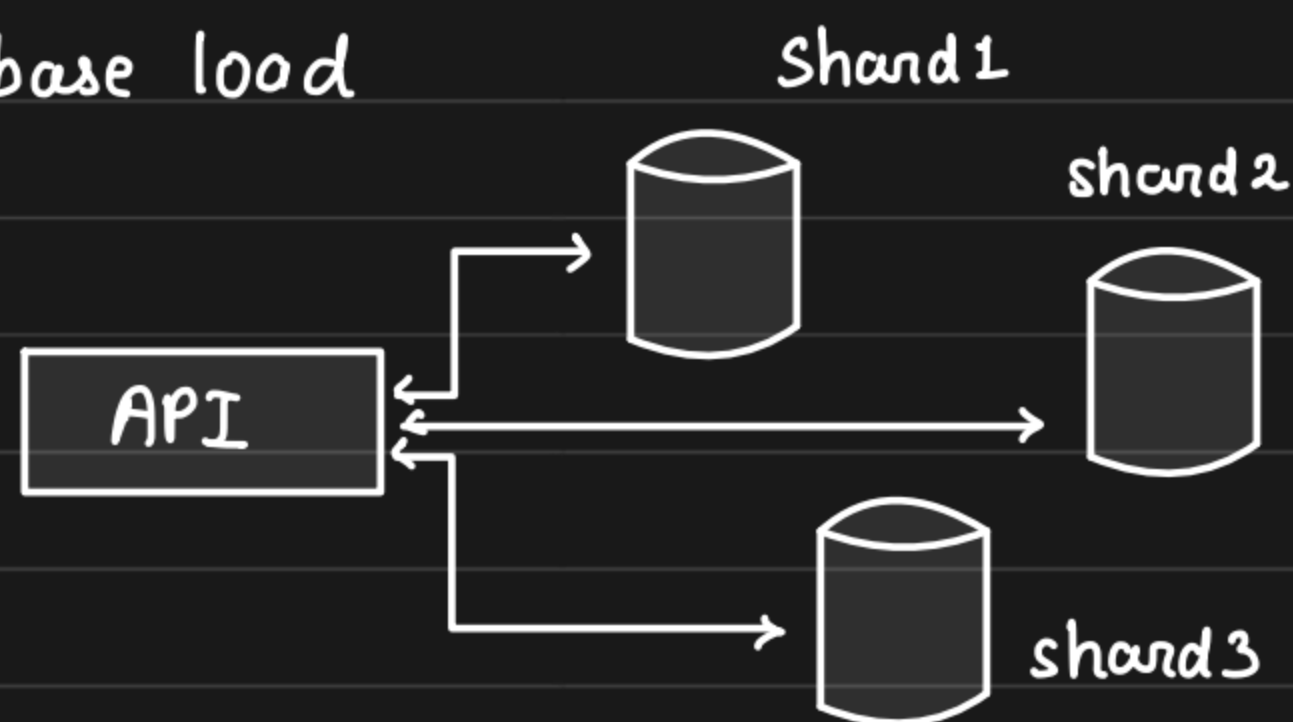


This way, we scale our overall database load

Note: shards are independent

no replication b/w them

API server needs to know whom to connect to, to get things done.



Note: some databases has a proxy that takes care of routing

Each shard can have its own replica (if needed)

## Exercise

1. configure one MySQL as replica of another
2. put some data and see the replication happening
3. write a small API service that has two connection obj
  - ↳ one master and one replica
4. depending on the request, make call to either master or replica
5. implement sharding by spinning Two DB
  - ↳ one handling keys (a to m)
  - ↳ second handling keys (n to z)
6. write API service that routes request to one of them depending on key