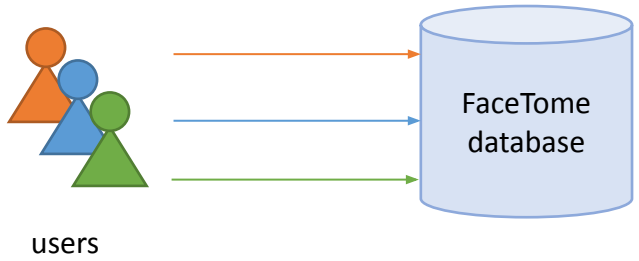
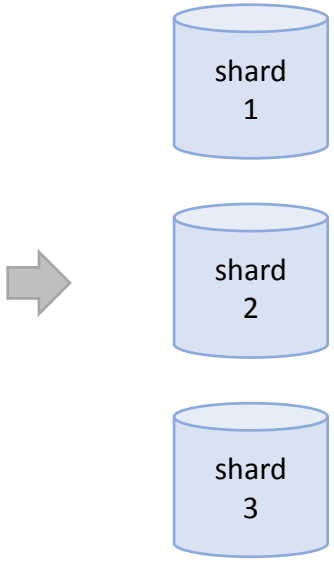
Partitioning strategies

lookup strategy



range strategy



hash strategy

How do we choose a shard for each user?

Partitioning strategies

create and store a mapping between keys and shards

lookup strategy

assign a shard randomly

assign based on shards utilization

lookup service

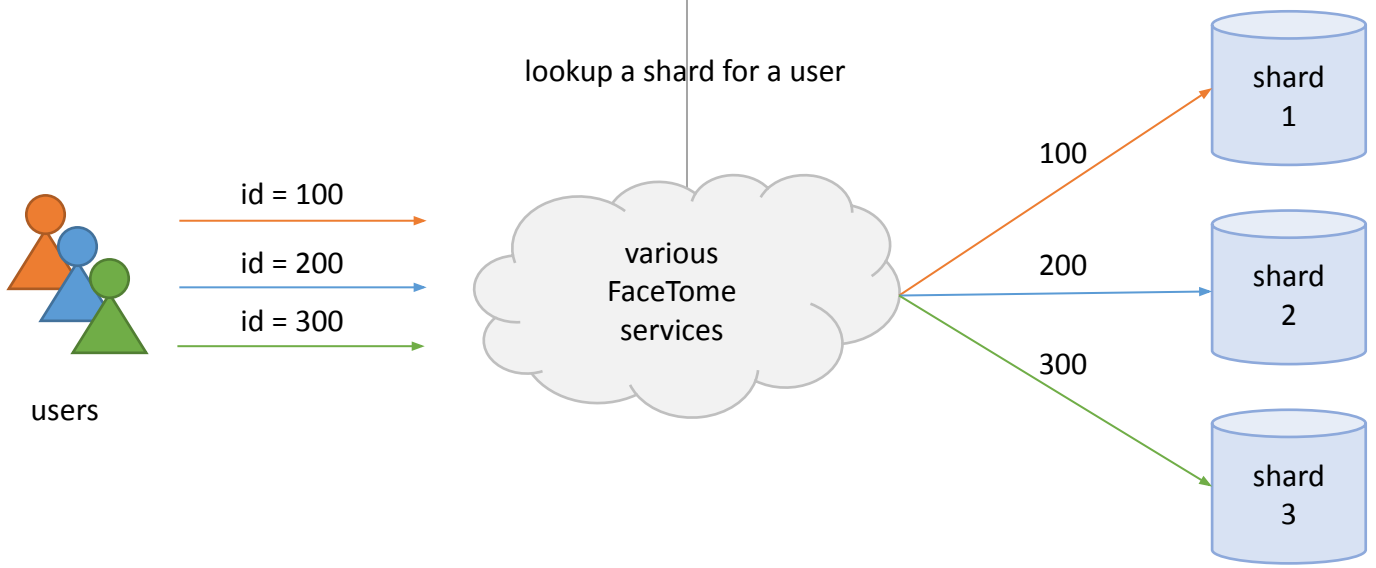
User Id	Last Name	Shard Id
100	Apple	shard 1
200	Apricot	shard 2
300	Banana	shard 3

- offers a lot of control over shard assignment logic

- hard dependency on the mapping
- requires highly-available and fast lookup service
- mapping can get really big over time (and needs to be partitioned)

How to make the lookup service

- highly available?
- fast?
- scalable?
- consistent?

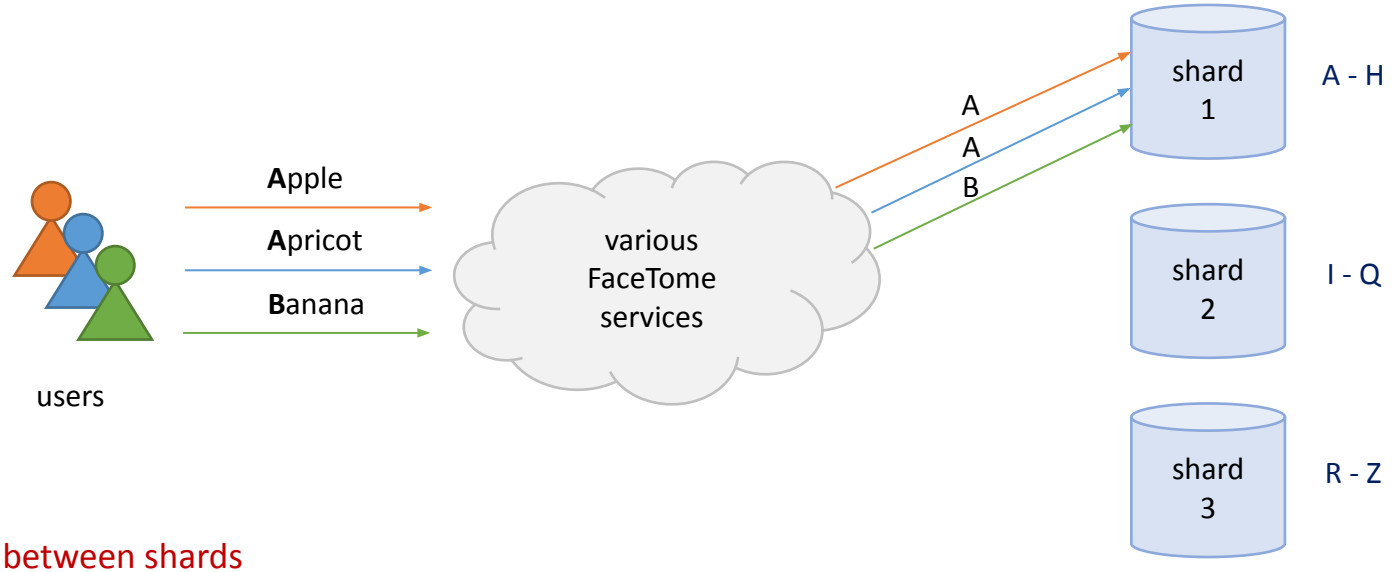


Partitioning strategies

each shard is responsible for a continuous range of keys

range strategy

- easy to implement
- works well with range queries



- provides suboptimal balancing between shards (which may lead to the hot shard problem)

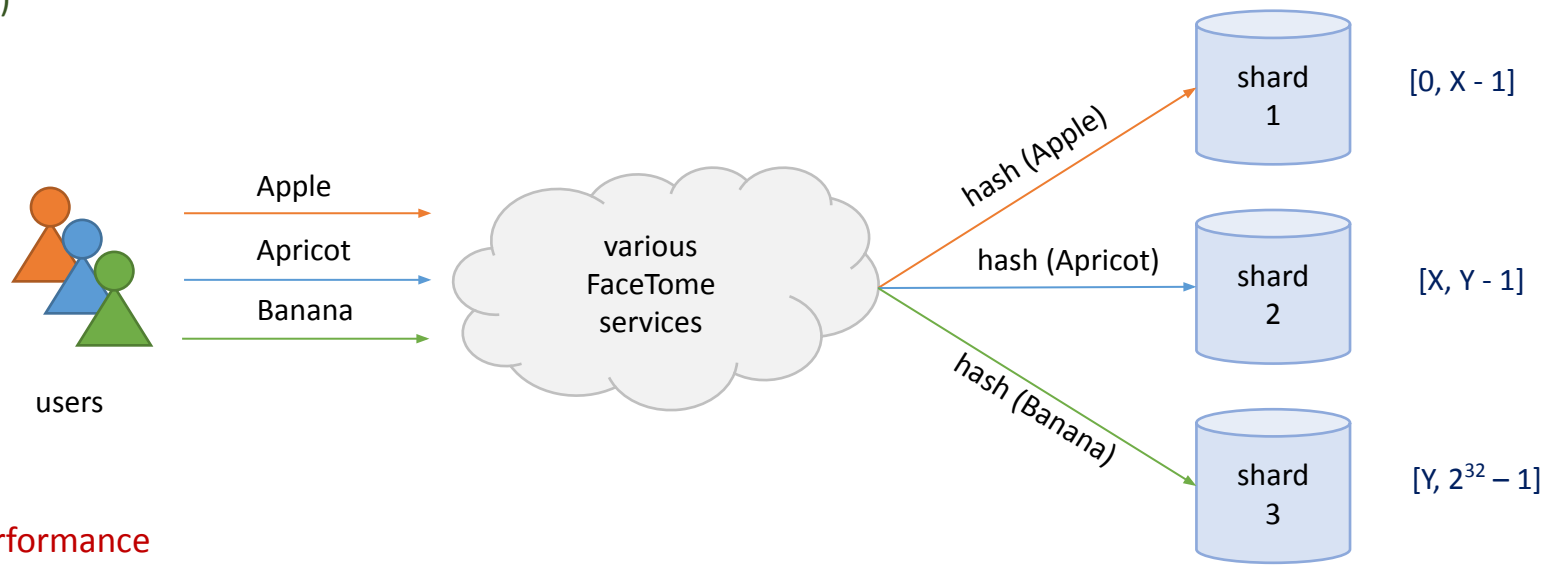
Partitioning strategies

each shard is responsible for a continuous range of hashes

hash strategy

hash (user last name) ---> $[0, 2^{32} - 1]$

- provides more even data distribution across shards (reduces the chance of hot spots)



- range queries may have poor performance
- computing the hash imposes an additional overhead

define shard boundaries by
- spacing them evenly
- using consistent hashing