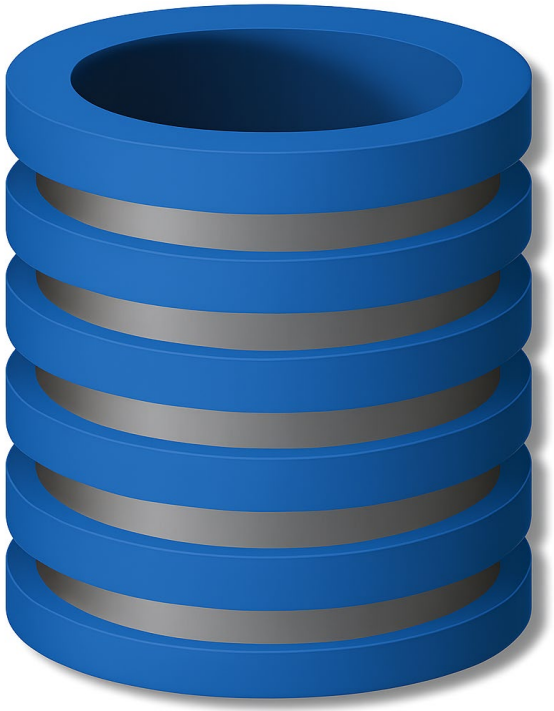


Sharding

Predmet: Administriranje Baze Podataka

Predavač: dr Dušan Stefanović

Problem

- ✓ Velika baza, tabele sa velikim brojem redova (reda milion)
- ✓ indeksiranjem nećemo postići željene performase jer se indeksna tabela konstatno uvećava
- ✓ Rešenje je **Sharding**

SELECT URL
FROM URL_TABLE
WHERE URLID = "5FTOJ"

URL tabela sa URL adresama koja sadrži preko milion redova

id	Url	urlid*
1	https://www.canva.com/design/DADrSCuKg4I/5sKekxVdctoGGq7Ri9O5GQ/edit	5FTOJ
2	https://en.wikipedia.org/wiki/Shard_(database_architecture)#Database_architecture	CeG0z
..	....	....
..	....	....
1M	https://www.quora.com/How-does-base64-encoding-work	J9COp

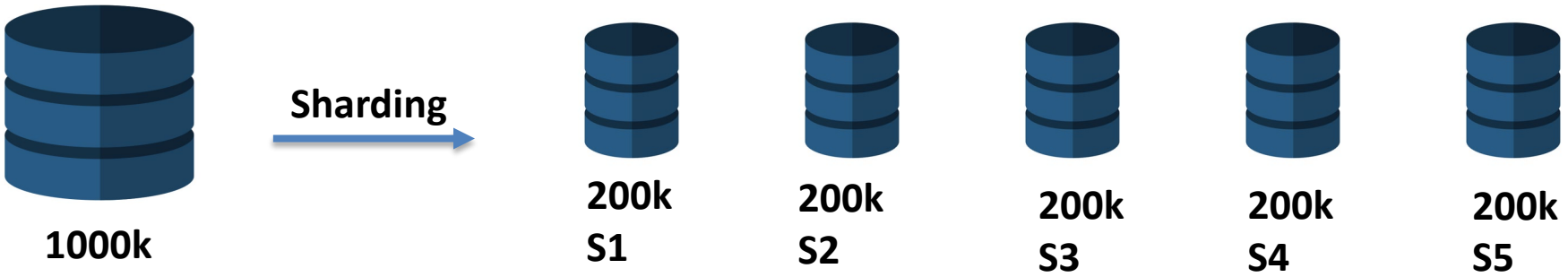
Problem

✓ indeksiranjem nećemo postići željene performase jer se indeksna tabela konstatno uvećava

Karakteristika	Problem
Velika indeksna tabela	Kako broj redova raste, tako raste i veličina indeksa. To može dovesti do: povećane latencije pri pretrazi i ažuriranju, veće potrošnje memorije i I/O operacija.
Ažuriranje podataka	Indeks mora da se ažurira pri svakom INSERT, UPDATE ili DELETE, što dodatno usporava performanse.
Jedan čvor	Indeksiranje ne rešava problem skalabilnosti ako je sve pohranjeno na jednom serveru.

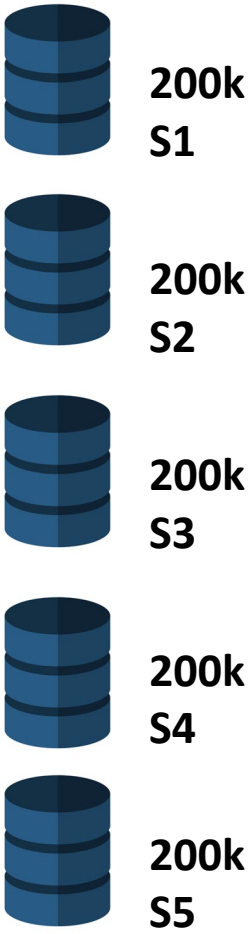
Rešenje

Sharding je tehnika **horizontalne podele baze** na više delova koji se nazivaju **shard-ovi**, a svaki shard sadrži deo ukupnih podataka.



Rešenje

- ✓ Podeliti milion redova jedne tabele u 5 instance baza sa istom šemom na različitim serverima
- ✓ Tabela UserLogs sa miliona redova može se podeliti:
  - ✓ Shard 1: korisnici sa ID 1–200k
  - ✓ Shard 2: korisnici sa ID 201 –400k
  - ✓ Shard 3: korisnici sa ID 401 –600k
  - ✓ itd.



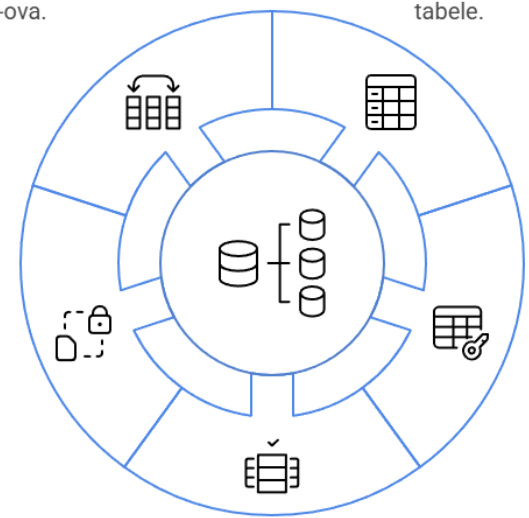
KARAKTERISTIKE SHARDING-a

Paralelna obrada upita

Upiti se izvršavaju paralelno nad više shard-ova.

Horizontalna podela

Svaki shard sadrži podskup redova iz tabele.



Izolacija grešaka

Pad jednog shard-a ne utiče na ostale.

Distribucija podataka

Podaci se dele prema određenom ključu (npr. ID korisnika, geografska lokacija ...)

Skalabilnost

Opterećenje se raspoređuje na više servera.

KADA KORISTITI SHARDING

Paralelni zahtevi

Kada su zahtevi za čitanje/pisanje obimni i paralelni

Nedovoljne performanse

Kada indeksiranje i replikacija ne mogu da poboljšaju brzinu

Brz rast baze podataka

Kada baze rastu brže nego što jedan server može da ih podrži.

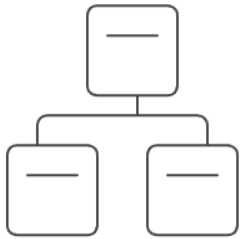
Rešenje

- ✓ Podeliti milion redova jedne tabele u 5 instance baza sa istom šemom na različitim serverima
- ✓ Primenjuje se row based particionisanje

Na koji način se određuje server za ključ **5FTOJ** ?
Koriste se hash funkcije za određivanje servera gde se nalazi podatak

```
SELECT URL
FROM URL_TABLE
WHERE URLID = "5FTOJ"
```

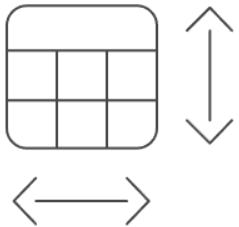
SLOŽENOST VREMENSKE PRETRAGE



O(log n)

Binarna pretraga

Uređena lista
Uslov: Lista mora biti sortirana.
Algoritam: Binarno prepolavljanje – poredi se srednji element sa ciljnim.
Vremenska složenost: O(log n)
Implementacija: najčešće kroz Binary Search Tree (BST)

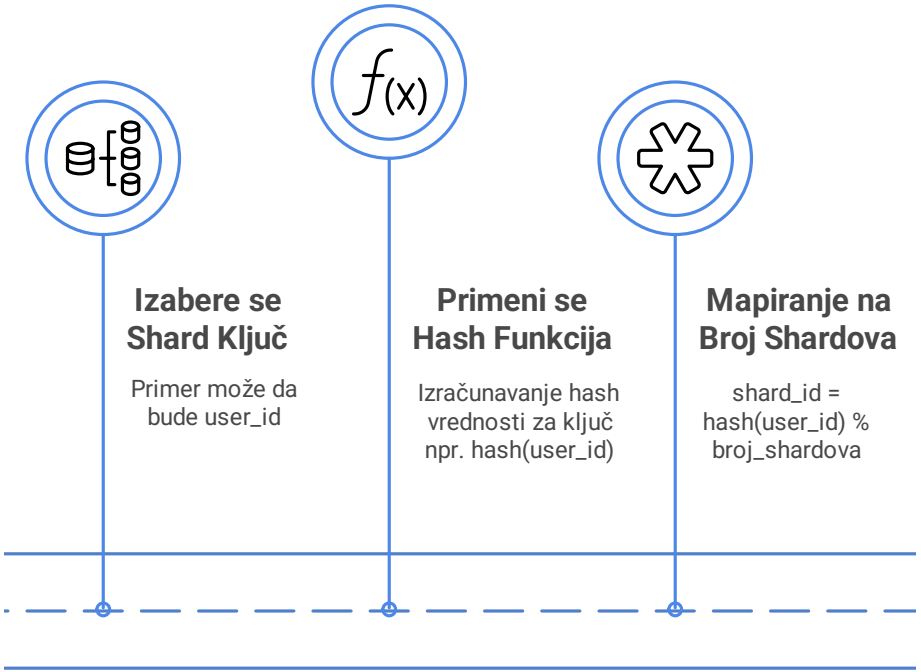


O(n)

Linearna pretraga

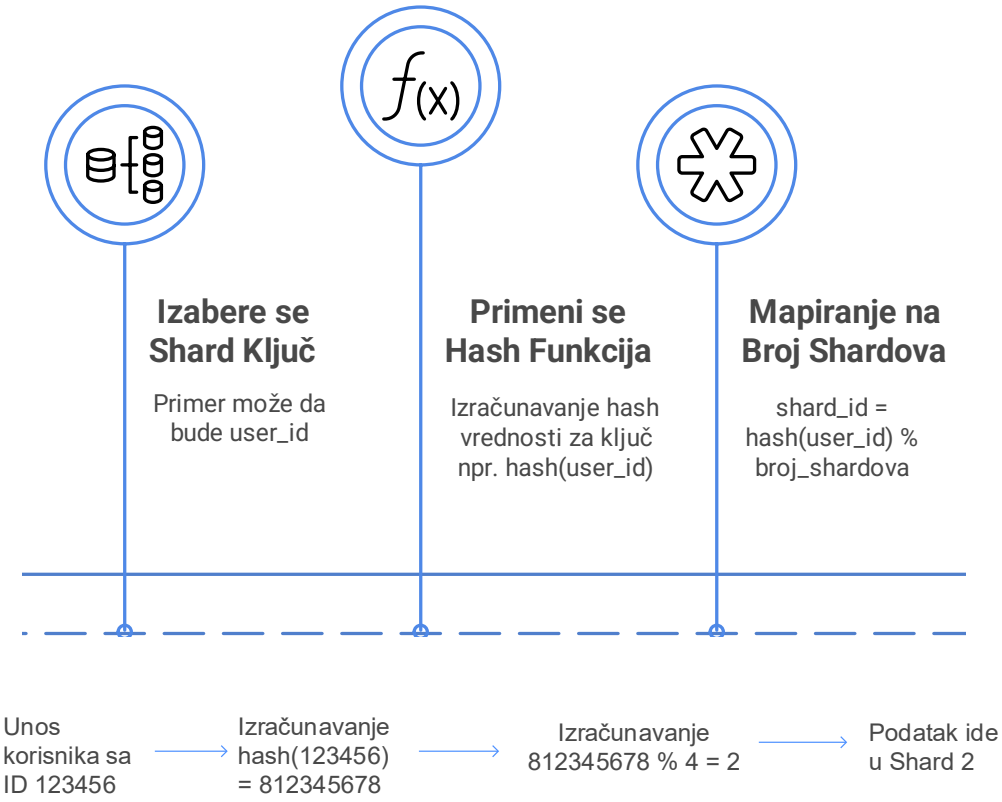
Neuređena lista
Uslov: Lista nije sortirana.
Algoritam: Prolazak kroz svaki element dok se ne pronađe traženi.
Vremenska složenost: O(n)
Rešenje: U takvim slučajevima, za ubrzanje pretrage koristi se hash tabela (hash mapa).

SLOŽENOST VREMENSKE PRETRAGE



Heš mape

- ✓ Ukoliko lista nije uređena potrebno je u tom slučaju proći kroz svaki element liste
  - ✓ U ovakvim slučajevima se koriste hash mape ili hash tabele (logika je slična telefonskom imeniku gde je ime (ključ) povezan sa brojem telefona (vrednost)).
- ✓ Komponente su:
  - ✓ ključ
  - ✓ Vrednost
  - ✓ hash funkcija je funkcija koja iz ulaznog podatka generiše broj fiksne dužine
  - ✓ Hash tabela se sastoji iz dve kolone ključa i vrednosti za svaki ključ



Kreiranje Heš tabela

Ključ se šalje hash funkciji
Hash funkcija generiše neki veliki broj.
Broj ne može da se direktno mapira sa brojem redova hash tabele.
Rešenje je da se iz hash vrednosti odredi moduo broja koji odgovara broju redova

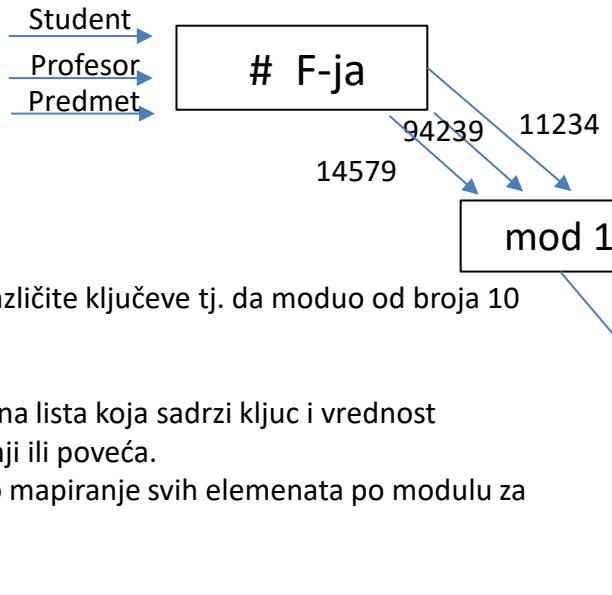
Key : Value Par

Student : Marko

Profesor : Dušan

Predmet : Fizika

Hash funkcija



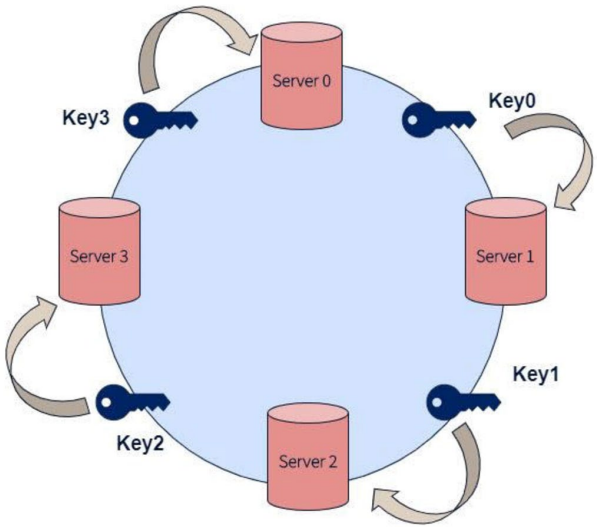
Hash tabela

Key	Bucket	
0		
1		
2		
3		
4	Marko	
5		
6		
7		
8		
9	Profesor: Dusan	Predmet: Fizika

Hash funkcija može da vrati isti broj za različite ključeve tj. da moduo od broja 10 da isti ostatak.
To se zove **hash kolizija**.
U tom slučaju se za taj red kreira povezana lista koja sadrzi kljuc i vrednost
Problem je ukoliko se broj ključeva smanji ili poveća.
U tom slučaju mora da se odradi ponovo mapiranje svih elemenata po modulu za taj broj redova.
Rešenje je konzistetno heširanje

Konzistetno heširanje

Koncept **konzistentnog heširanja (Consistent Hashing)** se koristi u distribuiranim sistemima da bi se efikasno raspodelili podaci po serverima – **bez potrebe za ponovnim preslikavanjem svih ključeva** kada dođe do promene broja servera.



server1:5434

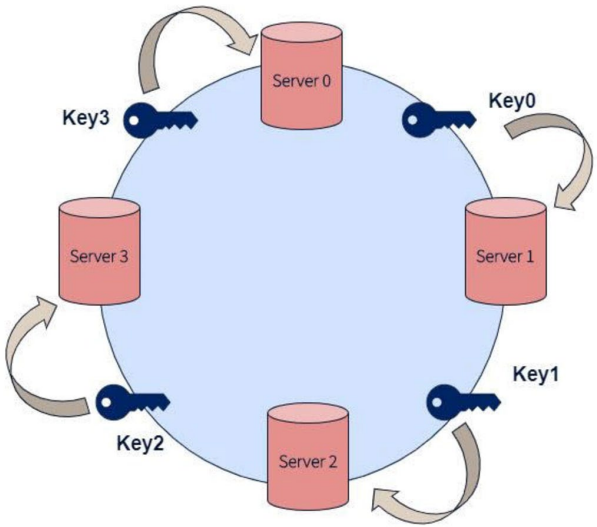
server2:5432

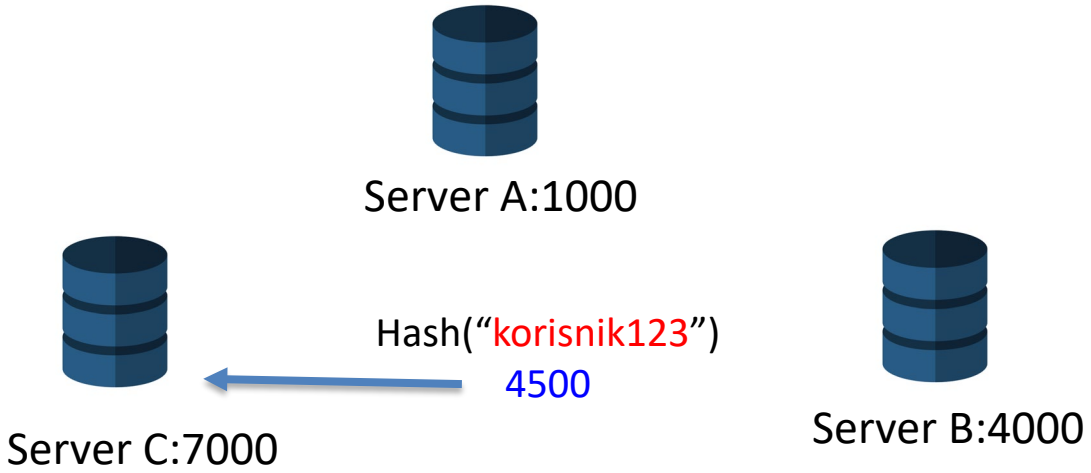
Hash(“Input1”)
5432
Hash(“Input2”)
5433
Hash(“Input3”)
5434

server3:5433

HEŠ PRSTEN (HASH RING)

Ideja je da se svi mogući rezultati hash funkcije poređani u krug (npr. od 0 do $2^{32} - 1$ kao brojčanik sata).
Svaki server se pomoću hash(server_naziv) smesti na određeno mesto na prstenu.
Svaki podatak (ključ) takođe se pomoću hash(key) smesti na određeno mesto.



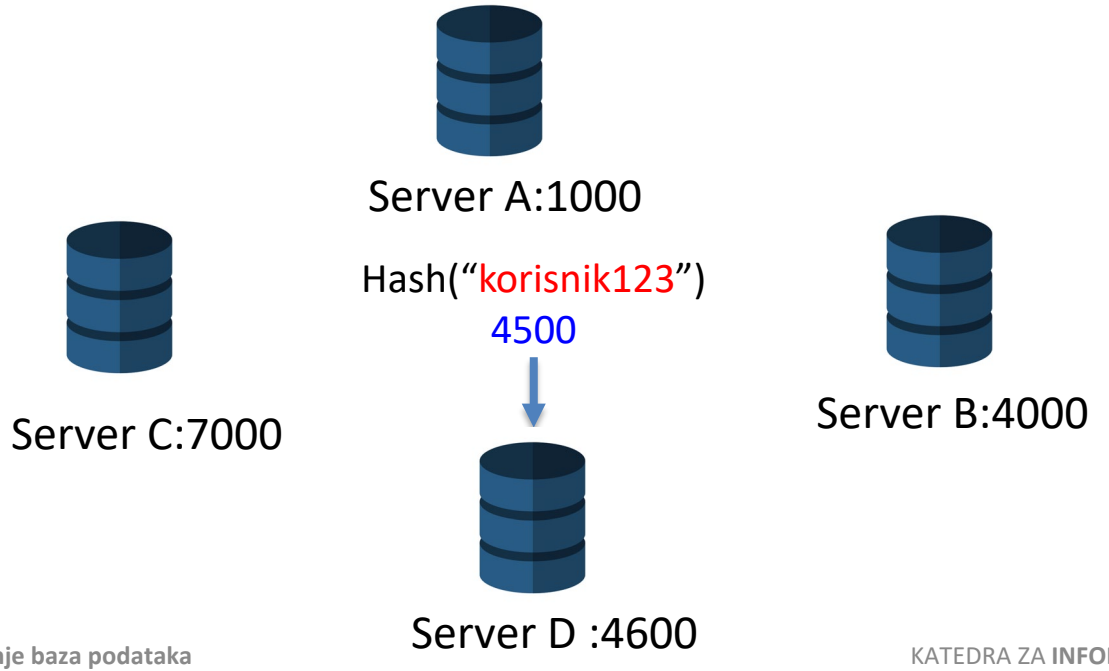


Dodavanje novog servera

Server D → hash("Server D") = 4600 → između 4500 i 7000

"Korisnik123" hash = 4500 → najbliži server = **Server D**

Samo "Korisnik123" se pomera, **svi drugi podaci ostaju gde jesu** ➞ Ovo je glavna **prednost** konzistentnog heširanja.



Horizontalno particionisanje

Opis

Tabela se deli na više manjih tabela u okviru iste baze podataka (na istom serveru).

Karakteristike

Deljenje po logici, npr. prema regionu ili godini (customers_Serbia, customers_USA, customers_2024)

Lokacija

Sve se nalazi unutar iste DBMS instance i često koristi istu šemu

Cilj

Lakše indeksiranje i brža lokalna pretraga ali ne nužno skalabilnost

SHARDING

Distribucija

Tabela se deli na više manjih delova (shardova), ali se ti delovi nalaze na različitim serverima (fizički odvojenim bazama).

Šema

Ime tabele i šema obično ostaju isti.

Usmeravanje

Klijentska aplikacija, proxy sloj ili middleware odlučuje na koji shard će pristupiti.

Svrha

Namijenjen je za horizontalno skaliranje i obradu velikog broja zahtjeva.

Horizontalno particionisanje vs Sharding

Karakteristika	Horizontalno particionisanje	Sharding
Lokacija	Jedna baza (jedan server)	Više baza (više servera)
Cilj	Organizacija i optimizacija	Skalabilnost i raspodela opterećenja
Promena šeme/tabele	Da (npr. orders_2022, orders_2023)	Ne (tabela se zove isto)
Skaliranje	Ograničeno	Horizontalno skaliranje
Primer upotrebe	Arhiviranje po godinama	Globalna web aplikacija sa velikim brojem korisnika
Oporavak u slučaju greške	Lakše	Složeniji zbog distribuiranosti

Prednosti primene Sharding tehnike

- ✓ Skalabilnost
  - ✓ Deljenje resursa na više servera
    - ✓ Podaci (podela opterećenja)
    - ✓ Memorija
- ✓ Bezbednost jer nam omogućava da imamo određenu kontrolu pristupa zahteva koji dolaze jer ih usmeravamo na određeni shard
- ✓ Optimalni i manji indeksi dovode do boljih performasi baze podataka

Nedostaci Sharding tehnike

- ✓ Transakcije kroz shard su problem
- ✓ Rollbacks
- ✓ Promena šeme je problem i Join upiti