NoSQL

FERIT

# MongoDB

## Indeksiranje i skalabilnost

Predavanje 8

# Sadržaj

- **Indeksiranje za Performanse:**
  - Zašto su indeksi ključni?
  - Tipovi indeksa u MongoDB-u:
    - Single Field (jednostruki)
    - Compound (složeni)
    - Multikey (za nizove)
    - Tekstualni, Geospatial, TTL... (pregled)
  - Kako odabrati pravi indeks? Analiza upita.
- **Skalabilnost i Visoka Dostupnost:**
  - Vertikalno vs. Horizontalno skaliranje
  - **Visoka Dostupnost: Replica Sets**
    - Arhitektura (Primary, Secondary, Arbiter)
    - Proces replikacije i automatski failover
  - **Horizontalno Skaliranje: Sharding**
    - Kada i zašto koristiti sharding?
    - Arhitektura (Shards, Mongos, Config Servers)
    - Shard Key (ključ za particioniranje)

# Dio 1:

- **Indeksiranje**

# Zašto su indeksi ključni?

- **Bez indeksa:**
  - MongoDB mora izvršiti **Collection Scan**.
  - Pregledava **svaki dokument** u kolekciji.
  - Performanse se linearno pogoršavaju s rastom broja dokumenata.
- **S indeksom:**
  - MongoDB pretražuje manju, sortiranu strukturu podataka (B-Tree).
  - Brzo pronalazi reference na tražene dokumente.
  - Performanse su logaritamske (O(log N)), što je puno bolje.
- **Analogija:** Traženje riječi u rječniku bez abecednog reda vs. s abecednim redom.

COLSCAN
VS
IXSCAN

# Tip Indeksa: Single Field (Jednostruki)

- **Opis:**
  - Najosnovniji tip indeksa.
  - Indeksira se vrijednost **jednog polja**.
  - Redoslijed sortiranja je bitan za sort() operacije, ali ne i za find() upite na jednom polju.
    - 1 za uzlazno
    - -1 za silazno
- **Kreiranje:**
  - db.collection.createIndex( { <polje>: 1 } )

# Tip Indeksa: Single Field (Jednostruki)

- **Primjer:** Imamo kolekciju `users` i često pretražujemo po emailu.

```
db.users.createIndex({ email: 1 })
```

- **Upiti koje ovaj indeks ubrzava:**
  - `db.users.find({ email: "test@example.com" })`
  - `db.users.find({ email: { $in: [...] } })`
  - `db.users.find().sort({ email: 1 })`
  - `db.users.find().sort({ email: -1 })`

# Tip Indeksa: Compound (Složeni)

- **Opis:**
  - Indeksira se **više polja** unutar jednog indeksa.
  - **Redoslijed polja** u definiciji indeksa je **IZUZETNO VAŽAN**.
- **Kreiranje:**
  - `db.collection.createIndex( { <polje1>: 1, <polje2>: -1, ... } )`

# Tip Indeksa: Compound (Složeni)

- **Primjer:** Često pretražujemo korisnike po `department` (odjelu), a zatim sortiramo po `lastName`.

```
db.users.createIndex({ department: 1, lastName: 1 })
```

- **Koje upite ovaj indeks podržava ("Index Prefixes")?**
  - ✓ `db.users.find({ department: "Sales" })`
  - ✓ `db.users.find({ department: "Sales", lastName: "Smith" })`
  - ✓ `db.users.find({ department: "Sales" }).sort({ lastName: 1 })`
  - ✗ `db.users.find({ lastName: "Smith" })` - Ne može koristiti ovaj indeks efikasno jer lastName nije prvo polje u indeksu!

# Tip Indeksa: Multikey (za Nizove)

- **Opis:**
  - Nije poseban tip indeksa, već **ponašanje** indeksa kada se kreira na polju koje sadrži **niz (array)**.
  - MongoDB automatski kreira **jedan indeks zapis za svaki element u nizu**.
- **Kreiranje:**
  - Isto kao i za Single Field indeks.
  - `db.products.createIndex({ tags: 1 })`

# Tip Indeksa: Multikey (za Nizove)

- **Primjer dokumenta:**

```
{
    "name": "Laptop Pro X1",
    "tags": ["electronics", "laptops", "powerful"]
}
```

- **Kako radi:**
  - Indeks će sadržavati tri zapisa za ovaj dokument: jedan za "electronics", jedan za "laptops" i jedan za "powerful". Svi pokazuju na isti dokument.
- **Upit koji ovaj indeks ubrzava:**
  - `db.products.find({ tags: "laptops" })`

# Ostali korisni tipovi indeksa

- **Text Indexes:**
  - Za pretraživanje teksta unutar string polja. Podržava pretragu po riječima, frazama, isključivanje riječi, i jezike (stemming).
  - `db.articles.createIndex({ content: "text" })`
  - Upit: `db.articles.find({ $text: { $search: "database performance" } })`
- **Geospatial Indexes (2dsphere):**
  - Za upite nad geoprostornim podacima (točke, linije, poligoni).
  - Podržava upite poput "pronađi sve u krugu", "pronađi što se siječe".
  - `db.places.createIndex({ location: "2dsphere" })`

# Ostali korisni tipovi indeksa

- **TTL (Time-To-Live) Indexes:**
  - Automatski briše dokumente nakon određenog vremena.
  - Indeksira se polje s datumom. Dokument se briše *expireAfterSeconds* nakon tog datuma.
  - `db.sessions.createIndex({ createdAt: 1 }, { expireAfterSeconds: 3600 })`
- **Unique Indexes:**
  - Osigurava da su sve vrijednosti za indeksirano polje jedinstvene.
  - `db.users.createIndex({ email: 1 }, { unique: true })`

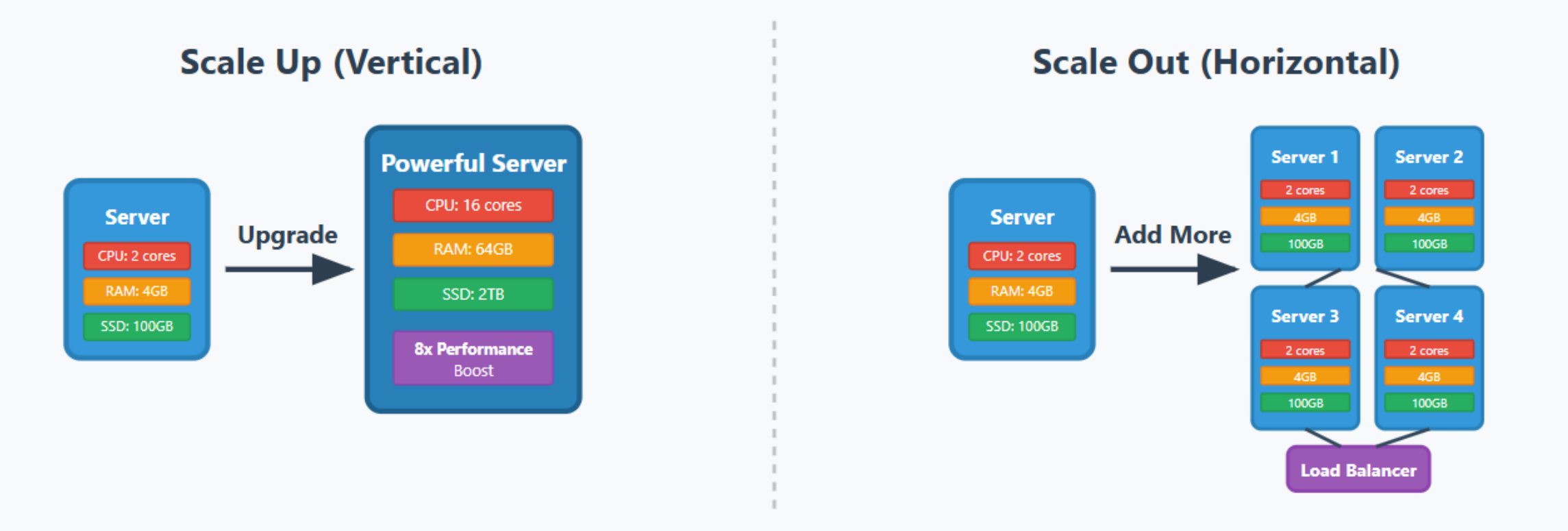
# Kako Odabrati Pravi Indeks? (ESR Pravilo)

- **Pristup temeljen na analizi upita:**
  - **E**quality (Jednakost): Prvo polje u indeksu treba biti ono po kojem pretražujete s točnom vrijednošću.
  - **S**ort (Sortiranje): Sljedeće polje treba biti ono po kojem sortirate.
  - **R**ange (Raspon): Zadnje polje u indeksu treba biti ono po kojem pretražujete po rasponu ($gt, $lt).
- **Primjer upita:**
  - Pronađi sve aktivne korisnike (status: "Active") iz odjela "Sales" (department: "Sales"), sortirane po datumu registracije (registeredAt).
  - `db.users.find({ status: "Active", department: "Sales" }).sort({ registeredAt: -1 })`
- **Dobar kandidat za indeks (po ESR pravilu):**
  - `db.users.createIndex({ status: 1, department: 1, registeredAt: -1 })`

# Dio 2:

- **Skalabilnost i visoka dostupnost**

# Vertikalno vs. Horizontalno skaliranje



# Vertikalno vs. Horizontalno skaliranje

- **Vertikalno Skaliranje (Scale-Up):**
  - Dodavanje više resursa (CPU, RAM, Disk) postojećem serveru.
  - **Prednosti:** Jednostavno.
  - **Mane:** Postaje izuzetno skupo. Postoji fizička gornja granica. Stvara jednu točku kvara (*Single Point of Failure*).
- **Horizontalno Skaliranje (Scale-Out):**
  - Distribucija opterećenja i podataka na više servera (klaster).
  - **Prednosti:** Može skalirati gotovo neograničeno. Koristi jeftiniji hardver (commodity hardware). Može pružiti visoku dostupnost.
  - **Mane:** Kompleksnije za postaviti i upravljati.
- **MongoDB je dizajniran za horizontalno skaliranje.**

# Visoka dostupnost (*Replica Set)*

- **Svrha:** Osigurati da je baza dostupna čak i ako jedan server padne. **Rješava problem visoke dostupnosti, ne skalabilnosti pisanja.**
- **Arhitektura:**
  - Klaster od (obično) 3 ili više mongod instanci.
  - **1 Primary:** Jedini čvor koji prima operacije pisanja. Glavni izvor podataka.
  - **N Secondaries:** Kopiraju podatke s Primary-a (asinkrono). Mogu služiti za operacije čitanja.
  - **Heartbeats:** Članovi seta stalno komuniciraju da provjere jesu li "živi".

# Automatski *Failover* (*Replica Set*)

- **Proces:**
  - Primary server postane nedostupan (pad, mrežni problem).
  - Preostali Secondary čvorovi to primijete jer ne dobivaju heartbeat signale.
  - Održavaju se **izbori (election)** da se izabere novi Primary.
  - Jedan od Secondary-a s najsvježijim podacima biva izabran i promoviran u novog Primary-a.
  - Aplikacija (preko drivera) automatski prepoznaje novog Primary-a i preusmjerava operacije pisanja na njega.
- **Cijeli proces je automatski i traje nekoliko sekundi.**
- **Arbiter (sudac):** Lagani mongod proces koji ne drži podatke, ali može glasati u izborima. Koristi se da bi se osigurao neparan broj članova i spriječio "split-brain" problem.

# Horizontalno Skaliranje: *Sharding*

- **Svrha:** Distribucija podataka i opterećenja pisanja preko više servera (ili Replica Set-ova). **Rješava problem skalabilnosti pisanja i velikih skupova podataka.**
- **Kada koristiti Sharding?**
  - Kada skup podataka premašuje kapacitet RAM-a jednog servera.
  - Kada volumen operacija pisanja premašuje kapacitet jednog Primary-a.
  - Kada želite pohraniti arhivske podatke na jeftiniji hardver.
- **Sharding je kompleksan i treba ga uvesti samo kada je nužno.**

# *Sharding*: Arhitektura

- **Komponente:**
  - **Shards (particije):** Svaki shard je **Replica Set** koji čuva dio ukupnih podataka. Npr. Shard A čuva korisnike od A-M, Shard B čuva korisnike od N-Z.
  - **Mongos (Query Router):** Lagani proces koji djeluje kao ruter. Aplikacija se spaja na mongos, a ne direktno na shardove. mongos zna na kojem shardu se nalaze traženi podaci i prosljeđuje upit tamo. Može ih biti više radi dostupnosti.
  - **Config Servers:** Replica Set koji čuva **metapodatke** klastera – mapu koja govori mongos-u koji podaci se nalaze na kojem shardu. Izuzetno važna komponenta.

# *Sharding*: *Shard Key*
## (Ključ za Particioniranje)

- **Definicija:**
  - Jedno ili više polja u dokumentu koja MongoDB koristi da bi odlučio na koji shard će smjestiti dokument.
  - **Odabir dobrog shard ključa je NAJVAŽNIJA odluka kod shardinga!**
  - Nakon što je kolekcija particionirana, shard ključ se ne može promijeniti.
- **Karakteristike dobrog Shard Ključa:**
  - **Visoki Kardinalitet (High Cardinality):** Puno jedinstvenih vrijednosti. (Loš: spol, status. Dobar: _id, email, username).
  - **Niska Frekvencija (Low Frequency):** Nema jedne vrijednosti koja se pojavljuje puno češće od drugih.
  - **Ne-monotono rastući (Non-Monotonic):** Ne bi trebao biti samo rastući (npr. timestamp), jer bi to sve nove upise slalo na samo jedan, zadnji shard ("hotspotting"). Često se koristi heširani shard ključ da bi se ovo izbjeglo.
- **Cilj:** Ravnomjerna distribucija podataka i opterećenja preko svih shardova.

# Sažetak današnjeg predavanja

- Indeksi su neophodni za performanse upita. Postoje različiti tipovi (Single, Compound, Multikey...) za različite potrebe. ESR pravilo pomaže u dizajnu.
- Visoka dostupnost se postiže pomoću **Replica Set-ova**, koji omogućavaju automatski failover.
- Horizontalno skaliranje se postiže pomoću **Shardinga**, koji distribuira podatke preko više shardova.
- Odabir dobrog **Shard Ključa** je kritičan za uspjeh sharded klastera.