

Univerzitet u Nišu

Elektronski fakultet

Seminarski rad

**Replikacija kod MongoDB baze podataka**

Sistemi za upravljanje bazama podataka, 2020/2021.

Mentor:

doc. dr Aleksandar Stanimirović

Student:

Vladimir Janjić, 1283

**Sadržaj**

[Uvod 3](#_Toc72695659)

[Replikacija – značaj, upotreba, ograničenja 4](#_Toc72695660)

[Replikacioni setovi 5](#_Toc72695661)

[Heartbeat 5](#_Toc72695662)

[Oplog (operacioni izveštaj) 6](#_Toc72695663)

[Izbori 8](#_Toc72695664)

[Rollback (premotavanje unazad) 9](#_Toc72695665)

[Write concern (potvrđivanje upisa) 10](#_Toc72695666)

[Skaliranje čitanja 10](#_Toc72695667)

[Primena replikacije 12](#_Toc72695668)

[Upravljanje članovima replikacionog seta 20](#_Toc72695669)

[Zaključak 21](#_Toc72695670)

[Literatura 22](#_Toc72695671)

# Uvod

Baze podataka predstavljaju veoma efikasan način skladištenja podataka za korišćenje od strane aplikacija, bilo da se radi o malim aplikacijama pokrenutim na jednom računaru, ili onim koje imaju servere na stotinama lokacija, često geografski razuđenih tako da pokrivaju veći deo površine planete Zemlje. Rad globalnih aplikacija neretko opslužuje višemilionsko tržiste korisnika, čiji se broj meri i stotinama miliona. Zbog toga je od velike važnosti da servisi koje aplikacija pruža budu dostupni bez prekida, uz to održavajući određeni stepen upotrebljivosti za optimalno zadovoljstvo korisnika.

Koliko god servera postojalo i koliko god oni bili udaljeni jedan od drugog, od velike je važnosti da korisnik u bilo kom trenutku, bilo gde u svetu, ima potpuno isti jedinstveni pogled na podatke aplikacije. Upotreba jedinstvenog skladišta sa kog se preuzimaju podaci i šalju serverima predstavljala bi veliko usko grlo, ne samo zbog ograničenosti brzine signala (koja mora biti manja od brzine svetlosti), već i zbog ogromnog broja zahteva u sekundi koji pristiže od milionskog auditorijuma. Takođe, pad sistema ili gubitak podataka bi u tom slučaju predstavljao velike gubitke za kompaniju, koji se ogleda ne samo u izgubljenom novcu, već i izgubljenom poverenju korisnika. Zbog toga bi distribuiranu prirodu aplikacije trebalo da prati i distribuirano skladište podataka, koje rešava najveće probleme centralnog skladišta, ali takođe ima i svoju cenu upotrebe. Dva glavna mehanizma preko kojih se ovo postiže su particionisanje i replikacija baze podataka. U okviru ovog rada, glavni fokus biće upravo na replikaciji.

Replikacija predstavlja pravljenje trenutnih kopija podataka, u celosti ili delovima, tako da se istovetni podaci mogu naći na više lokacija, na najčešće fizički udaljenim serverima. Ovim se postiže da podaci budu bliži korisnicima, čime se smanjuje kašnjenje odgovora. Omogućeno je slanje i primanje više zahteva na većem broju servera, pad sistema je ograničen na područije usluga jednog servera i podaci se često mogu nadomestiti sa obližnjeg servera, a takođe se i zahtevi privremeno mogu preusmeriti tako da korisnik ne bude svestan da problem uopšte i postoji (uz eventualnu malo veću latenciju). Kao loša strana ističe se potreba za većom konzistencijom podataka koja mora postojati između servera.

Jedna od baza podataka koja se može koristiti u ove svrhe je i MongoDB. Upravljanje replikacijom u okviru MongoDB-a je jednostavno, a mehanizmi replikacije su jasni i korisnicima bez velikog tehničkog poznavanja oblasti. U okviru ovog rada biće izložene pojedinosti upravljanja radom replikacije u okviru baze podataka MongoDB. Na početku će biti reči o tipu replikacije koji se koristi, na koji način se postiže rad replikacije, kakve sve mogućnosti pruža i koji su to problemi koje rešava. Zatim će biti reči o upravljanju replikacijom od kreiranja, preko provere statusa, do odklanjanja problema koji mogu da nastanu, kao i detalja oko održavanja konzistencije i poboljšanja dostupnosti. Sam rad replikacije biće prikazan na konkretnoj bazi podataka.

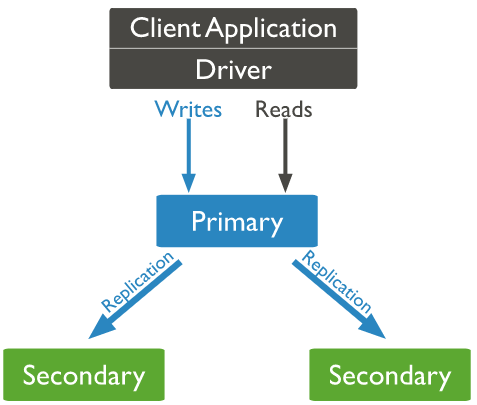
# Replikacija – značaj, upotreba, ograničenja

Do verzije 3.6, replikacija MongoDB baze podataka oslanjala se na dobro poznati koncept „master-slave“. Jedan od servera je imao glavnu ulogu (eng. *Master*) dok su ostali bili podređeni (eng. *Slave*), tj. zavisni od njega. Međutim, taj model je imao svoje nedostatke tako da se u novijim verzijama prešlo na korišćenje drugačijeg koncepta koji se naziva „replikacioni setovi“.

Princip rada oba modela je sličan: glavni server (kod replikacionih setova nazvan primarni) sadrži podatke koje ostali serveri (kod replikacionih setova nazvani sekundarni) kopiraju i održavaju sinhronim sa trenutnim stanjem. Poboljšanje kod replikacionih setova se ogleda u automatskoj zameni primarnog servera – ukoliko se javi pad primarnog servera, njegovu ulogu može preuzeti neki od sekundarnih servera. Treba napomenuti da iako su replike kopije podataka, one nisu zamena za rezervne kopije (eng. *Backup*).

Glavni značaj replikacije ogleda se u redundantnosti. Serveri sa replikama su neretko geografski udaljeni, tako da replikacija predstavlja sigurnost od kvara. Kako se replikacija odvija asinhrono, postojanje replikacije ne utiče na performanse primara. Sekundare je moguće dodatno odložiti tako da kasne sa replikacijom podataka, kao sigurnost kod slučajnog brisanja kolekcije ili korupcije podataka.

Osim pomenutih pogodnosti, replikacija uprošćava održavanje. Skupe operacije je moguće izvršiti na nekom od sekundara, kao što su pravljenje rezervnih kopija ili građenje indeksa. Takođe, čitanje je moguće izvršiti i sa sekundara, ali se time potencijalno javljaju problemi. Prikaz jednostavnog replikacionog seta sa tri člana se može videti na slici 1.



Slika 1. Izgled replikacionog seta sa tri člana.

# Replikacioni setovi

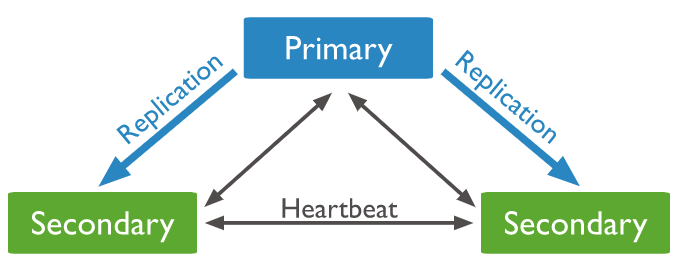
Rad replikacionog seta zasniva se na dva mehanizma: „heartbeat“ i „oplog“. Njima je omogućeno postojanje drugih mehanizama kao što su replikacija, izbori, glasanje, potvrđivanje upisa i premotavanje unazad. Utiču i na održavanje konzistencije kao na mogućnost skaliranja čitanja.

## Heartbeat

Heartbeat predstavlja malu poruku koju razmenjuju članovi replikacionog seta, obično na svake dve sekunde, kako bi proverili status ostalih članova. Igra glavnu ulogu u omogućavanju automatskog glasanja i „premošćavanja grešaka“ (eng. *Failover*). Premošćavanje grešaka se javlja u slučaju kada primar nije dostupan određeno vreme ograničeno trajanjem heartbeat-a i u tom slučaju javlja se mogućnost da neki od sekundara preuzime njegovu ulogu. Kod MongoDB-a, premošćavanje se vrši automatski, ukoliko postoji većina na izborima u okviru replikacionog seta. Drugi problem koji se može javiti jeste pad nekog od sekundara. Pri tome mogu nastati dva različita ponašanja u okviru seta:

* **postoji većina** – nakon pada sekundara i dalje ostaje dovoljan broj članova da bi se omogućila većina u glasanju; replikacioni set nastavlja normalno sa radom.
* **ne postoji većina** – nakon pada sekundara ne postoji dovoljan broj servera da bi se ostvarila većina, iako može postojati server koji bi mogao preuzeti ulogu primara; svi članovi postaju sekundari, primar degradira sebe u sekundar, replikacioni set ostaje bez primara.

Zbog kašnjenja na mreži, problem se može javiti i u samoj poruci, umesto na nekom određenom članu. Osim provere statusa članova, hearbeat obaveštava servere i o postojanju većine. Prema tome, greška u poruci može da izazove ponovne izbore, iako primar i dalje postoji u setu, ali određeni član nema informaciju o tome. Tada nastaje problem dva primara koji se mora rešiti kontaktiranjem administratora. Prikaz komunikacije heartbeat-om dat je na slici 2.



Slika 2. Provera stanja članova replikacionog seta pomoću heartbeat-a.

Pri proveri članova, svaki od njih se može naći u više stanja, od kojih stabilna stanja predstavljaju PRIMARY, SECONDARY i ARBITER, dok su ostala prelazna ili se javljaju u slučaju greške. Pregled ostalih stanja dat je u tabeli 1.

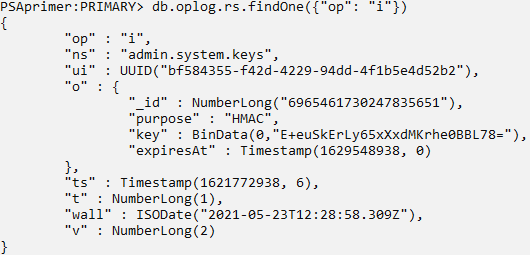
|  |  |
| --- | --- |
| Stanje | Opis stanja |
| STARTUP | Početno stanje, učitavanje MongoDB konfiguracije replikacionog seta. |
| STARTUP2 | Sledi nakon STARTUP, traje par sekundi, inicijalna sinhronizacija seta. |
| RECOVERING | Sledeće stanje, ispravan član, ali nije spreman za čitanje (na početku ili tokom duge operacije). Stanje kada server previše zaostane za replikacijom (nije greška, čeka na člana sa većim oplog-om da nastavi dalje). |
| PRIMARY | Stanje koje označava primar, postoji samo 1, jedini prima operacije upisa. |
| SECONDARY | Stanje koje označava sekundar, eventualno prima operacije čitanja, mogućnost da pređe u primar glasanjem. |
| ARBITER | Stanje koje označava arbiter, poseban server koji glasa, a ne čuva podatke. |
| DOWN | Član nedostupan. Može da radi i dalje, ali postoji greška na mreži. |
| UNKNOWN | Član nije komunicirano nikada sa ostalima, tako da ne zna svoje stanje. |
| REMOVED | Član koji je izbačen iz replikacionog seta. |
| FATAL | Član ne odgovara na poruke, konekcija i dalje ostvarena. Greška na mašini. |
| ROLLBACK | Stanje u koje ulazi član tokom premotavanja unazad. |

Tabela 1. Prikaz mogućih stanja članova replikacionog seta.

Stanje svakog od članova je moguće proveriti i ručno upotrebom administratorskih komandi, a takođe i ručno promeniti status (npr. degradiranje primara u sekundar na određeno vreme).

## Oplog (operacioni izveštaj)

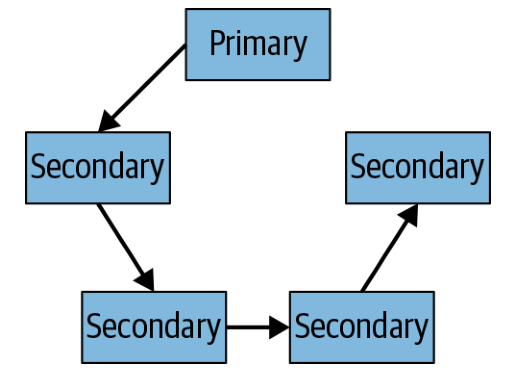
Oplog predstavlja ograničenu kolekciju koja se čuva u okviru „local“ baze podataka svakog od članova replikacionog seta. Ova baza podataka se, shodno imenu, ne replikuje, već je jedinstvena i menja je svaki član za sebe. Glavna uloga se ogleda u omogućavanju replikacije, koja se u okviru MongoDB baze podataka izvršava asinhrono. Jedna stavka oplog-a čuva se za svaku operaciju pisanja i sadrži dovoljno informacija da bi se operacija replikovala na sekundarima. Neki od podataka su vremenska markica, tip operacije (npr. insert), dokument na kojim se izvršava operacija. Nad oplog-om se može zadati upis kao i nad drugom bazom podataka. Primer izgleda jedne stavke oplog-a dat je na slici 3.



Slika 3. Izgled jedne stavke oplog-a.

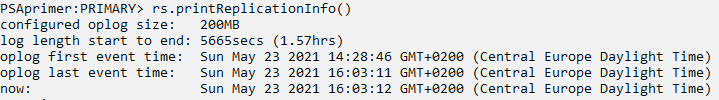
Na početku formiranja replikacionog seta, vrši se inicijalna sinhronizacija gde se obavlja potpuna kopija svih podataka sa primara (osim baze podataka „local“). Svaki novi član prolazi kroz istu proceduru (s tim što ne mora da kopira sa primara). Ukoliko su postojali podaci na sekundarima, oni će biti obrisani. Vrši se i kopiranje indeksa, ako je to omogućeno. Istovremeno, primar radi ispravno, tako da prihvata nove izmene koje se beleže i replikuju nakon kopiranja. Replikacija novih izmena može uticati na usporeno kopiranje jer se nove izmene prvo moraju uvesti u RAM (time se remeti kopiranje trenutnog dokumenta, ili više njih). Nakon završene inicijalne sinhronizacije, replikacioni set može da krene sa normalnim radom.

Proces replikacije kreće kada neki od sekundara pročita oplog primara ili nekog drugog sekundara sa kog se sinhronizuje (automatski izbor). Sekundar upoređuje svoju poslednju oplog stavku sa stavkama oplog-a člana sa kog se sinhronizuje i zahteva sve novonastale promene od jedinstvenog trenutka poklapanja. Prema tome, nije bitno ako sekundar padne na određeno vreme jer samo nastavlja gde je stao. Zatim se istovetne operacije izvršavaju nad dokumentima drugog sekundara i upisuju u njegov oplog. Time je moguće napraviti lanac sinhronizacije, ali se u tom slučaju javlja povećano kašnjenje (jer svaki sekundar mora da čeka da prethodni završi sinhronizaciju). Obično se sinhronizacija obavlja gotovo trenutno jer sekundari salju zahteve koji čekaju na promene, tako da primar može odmah da odgovori na iste. Na slici 4 prikazan je izgled lanca replikacije gde ne postoje dva člana koja se sinhronizuju od istog sekundara.



Slika 4. Primer izgleda replikacionog lanca.

Kako je oplog ograničen, treba voditi računa o veličini kolekcije jer će najstarije stavke biti prepisane nakon određenog vremena. Prema osnovnom podešavanju, oplog zauzima 5% slobodnog prostora na disku (odn. od 990MB do 50GB), što je u većini slučajeva dovoljno za potrebe razvoja. U okviru produkcijskog okruženja, međutim, često postoji potreba za ručnim podešavanjem veličine, što je moguće odraditi na početku formiranja replikacionog seta, ili kasnije u toku rada. Veličina koja će biti dovoljna zavisi od broja upisa u jedinici vremena. Ukoliko je poznat obim upisa, oplog bi trebalo podesiti tako da može da skadisti između jednog i nedelju dana stavki, što predstavlja broj dana pre nego što oplog bude prepisivan. Treba obratiti pažnju na to da se operacije nad više dokumenata razlažu, tako da se beleži po jedna stavka za svaki obrađeni dokumenat. Slika 5 prikazuje osnovne podatke o oplogu – veličinu, vreme između prve i poslednje stavke i trenutno vreme.



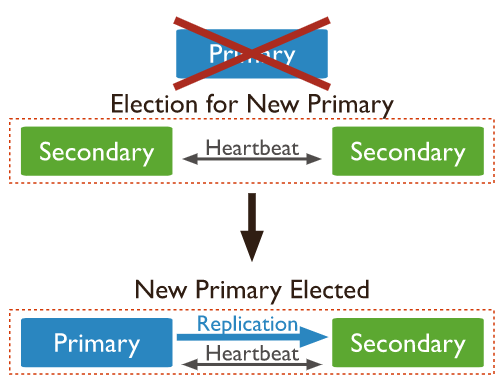
Slika 5. Informacije o oplog-u.

Problem koji može nastati kod premalog oplog-a naziva se „ustajalost podataka“ (eng. *Stale Data*). Mala veličina oplog-a znači da će najstarije stavke biti brzo prepisane. Ako je neki od sekundara bio nedostupan duže vreme, nastaje slučaj kada je nemoguće pronaći zajednički trenutak od kog će se nastaviti sinhronizacija. Javlja se greška „*replication data too stale, halting*“ i sekundar prestaje sa radom. Normalan rad je moguće nastaviti sinhronizacijom sa svežijom rezervnom kopijom, ili restartovanjem servera od početka. Često se ovaj problem javlja i kod inicijalne sinhronizacije, ako se radi o velikoj količini podataka koju je potrebno kopirati.

## Izbori

Izbori predstavljaju mehanizam koji se pokreće kada neki od članova replikacionog seta ne može da stupi u kontakt sa primarom preko heartbeat-a. Tada isti član šalje poruku svim ostalim da ih obavesti o glasanju. Ostali članovi glasaju za ili protiv toga da član koji je započeo glasanje postane primar. Problemi koji mogu sprečiti člana da postane primar jesu zaostalost ili postojanje drugog člana sa većim prioritetom koji takođe zahteva izbore. Glasanje je uspešno ako nema prepreka i član postaje novi primar, ali samo u slučaju ako je ostvarena većina na izborima. Ukoliko nije, može da inicira izbore kasnije. Novi primar će ostati u toj ulozi dok god ima većinu, radi ispravno, nije odstupio ili set nije rekonfigurisan. Obično izbori traju svega nekoliko milisekundi, nakon inicijalnih 2-10 sekundi, koliko je potrebno da heartbeat istekne. Da bi se sprečilo duplo glasanje, svaki od članova je obeležen svojim ID-jem. Broj članova koji glasaju je najviše sedam. Nekada je potrebno zabraniti izbore u potpunosti, što se postiže administratorskom komandom.

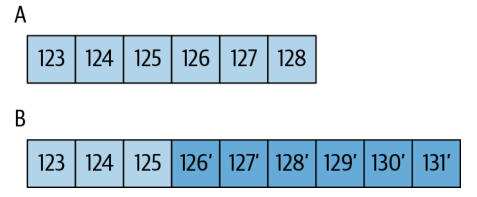
Posebnu ulogu u izborima imaju članovi arbiteri. Predstavljaju posebne male servere koji ne skladište podatke, već samo učestvuju u glasanju, obično kada je potrebno ostvariti većinu. Najčešće se koriste kod malih PSA arhitektura (Primary-Secondary-Arbiter). Jednom postavljen arbiter ostaje u ulozi arbitera u toku svog rada. Zbog svoje prirode, najbolje je koristiti samo jedan arbiter, ili još bolje ni jedan, ukoliko postoji mogućnost za pravljenje još jedne kopije podataka. Više arbitera ne ubrzava izbore već ih samo može komplikovati, zahtevajući više članova za većinu. Sa slike 6 se može videti ishod izbora kod tročlanog replikacionog seta kada primar padne.



Slika 6. Izbori kod tročlanog replikacionog seta.

## Rollback (premotavanje unazad)

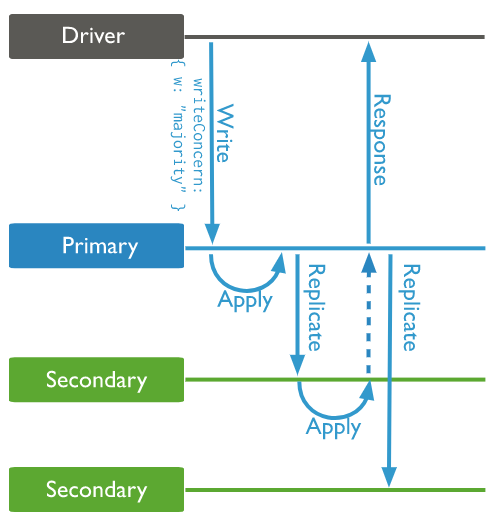
Replikacioni setovi, u produkcijskom okruženju, često podatke smatraju potvrđenim ukoliko su replikovani na većinskom broju članova (više od 50% ili n/2+1 servera). Greška se može javiti ukoliko primar postane nedostupan pre nego što sekundari uspeju da prekopiraju oplog. U skladu sa procedurom, nastaju izbori i bira se novi primar koji normalno nastavlja sa radom, primajući nove upise. Ako u međuvremenu stari primar postane dostupan, javlja se konflikt u njihovim oplog-ovima. Tada stari i novi primar pokreću proces „premotavanja unazad“ (eng. *Rollback*). Problem koji se javlja jeste nemogućnost pronalaženja jedinstvene stavke nakon koje bi se nastavila sinhronizacija (ni jedan od članova ne sadrži nereplikovane stavke). Upoređuju se oplog-ovi starog i novog primara i brišu sve operacije nastale nakon jedinstvene tačke sinhronizacije. Sve operacije nastale nad primarom koje nisu replikovane se čuvaju u novoj kolekciji sa datumom rollback-a. Na administratoru je da odluči šta će raditi sa tim podacima – obrisati ih, spojiti, prepisati. Na slici 7 dat je primer rollback-a. Oplog starog primara A sadrži stavke 126-128 koje se razlikuju od stavki oplog-a novog primara B, tako da ih je potrebno ukloniti.



Slika 7. Proces rollback-a kod starog i novog primara.

## Write concern (potvrđivanje upisa)

Javljaju se slučajevi kada je replikovanje upisa na većinskom broju članova, pre nastavka sa radom, sporo. Osnovno podešavanje zahteva potvrdu upisa samo na primaru, što je dovoljno za razvoj, ali je zbog sigurnosti bolje upotrebiti većinu kada se radi o produkcijskom okruženju. Ono na šta treba obratiti pažnju jeste kašnjenje. Garantovanjem upisa na većem broju servera znači da će replikacioni set biti nedostupan dok se ne izvrši sinhronizacija. Uz to postoji i mogućnost pojave grešaka na mreži. Da bi se to izbeglo, ostavljena je mogućnost ručnog podešavanja broja servera za potvrdu, kao i vremena pre isteka zahteva. Podešavanje se vrši preko promenljivih „w“ i „wtimeout“, odn. navodi se broj servera i broj milisekundi pre nego što se vrati greška, respektivno. Postojanje ograničenja trajanja potvrde je važno jer bi u suprotnom replikacioni set mogao da ostane nedostupan beskonačno dugo. Primar može da nastavi sa radom kada je „w“ zadovoljeno, o čemu ima informaciju. Na slici 8 dat je prikaz ponašanja replikacionog seta sa potvrđivanjem upisa.



Slika 8. Odziv aplikacije kada je potrebno replikovati upis na bar jedan sekundar (w=2).

Osim definisanja broja članova kao vrednost za „w“, postoji i opcija definisanja korisničkih pravila za potvrđivanje upisa. Vrši se na osnovu oznaka predstavljenih u obliku ključ-vrednost. Time se definišu pravila koja uključuju samo određene servere kada ih je nemoguće podeliti samo na osnovu broja. Jedna od primena se ogleda u garantovanju replikacije na jednom serveru po datacentru, ako postoji više lokacija.

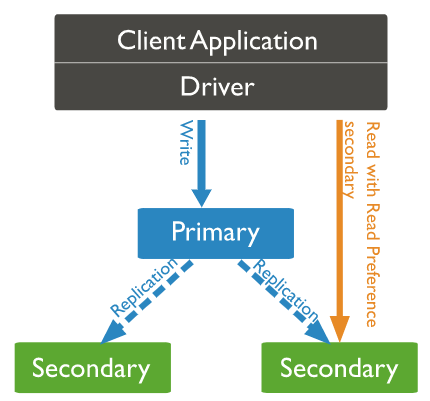
## Skaliranje čitanja

Iako replikacioni set sadrži više članova, u osnovnom podešavanju, MongoDB baza podataka omogućava čitanje samo sa primara. Za poboljšanje u brzini odziva, moguće je ostvariti čitanje sa nekog od sekundara. Za određivanje ponašanja, koristi se jedan od modova:

* **primary** – osnovno podešavanje, čitanje dozvoljeno samo sa primara i rezultati čitanja su uvek konzistentni. Ako se javi kvar, javlja grešku.
* **primaryPreferred** – čitanje se obavlja na primaru kad god je moguće. Ukoliko dođe do greške, čitanje se obavlja na nekom od sekundara i nije garantovana konzistentnost.
* **secondary** – čitanje samo sa sekundara, potencijalno nekonzistentni podaci. Greška ukoliko se javi kvar na sekundarima.
* **secondaryPreferred** – čitanje kad god je moguće sa sekundara. Čitanje sa primara ukoliko se javi greška na sekundarima.
* **nearest** – čitanje se obavlja sa najbližeg člana replikacionog seta po merilima kašnjenja na mreži. Ne garantuje se konzistentnost podataka.

Čitanje sa sekundara može biti korisno iz više razloga. Primar se može koristiti samo za upis, primar i sekundar imaju različite indekse (da bi se sprečila sinhronizacija), potreban brz odziv aplikacije. Međutim, čitanje sa sekundara unosi i niz problema, od kojih je najvažniji problem konzistencije, tj. validnosti podataka. Klijent koji se povezuje sa replikacionim setom nema informaciju o starosti podataka, tako da neki skorašnji podaci mogu ostati nereplikovani na vreme pre nego što im se pristupa.

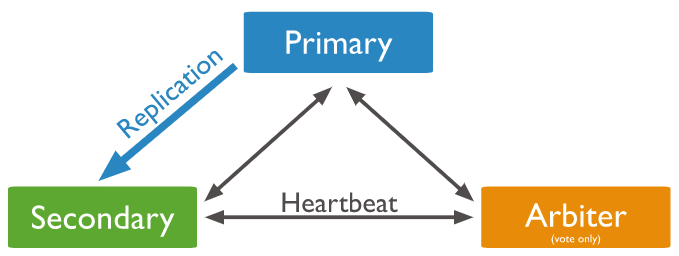
Ako je omogućeno, čitanje se obavlja sa najbližeg sekundara po merilima kašnjenja na mreži, čak iako mod „nearest“ nije uključen. Jedina razlika je u tome što „nearest“ uključuje i primar. Pre samog čitanja, obično se vrši odbacivanje servera sa velikim kašnjenjem i bira jedan od preostalih. Osnovno ograničenje kod skaliranja čitanja se javlja u okviru ograničenja na maksimalno 50 članova replikacionog seta. Takođe treba uzeti u obzir i resurse sekundara koji su u najvećem broju slučajeva manji nego kod primara. Slučaj kada je nemoguće koristiti skaliranje čitanja jeste kada aplikacija eksplicitno zahteva konzistentne podatke. Tada ostaje mogućnost podele podataka na kritične (obavezno konzistentne) i nekritične (možda konzistentne) i odgovarajuća raspodela čitanja. Napredne opcije rutiranja čitanja je moguće ostvariti preko oznaka, po istom principu kao i kod garantovanja upisa. Na slici 9 je dat prikaz čitanja kada je omogućeno čitanje sa sekundara.



Slika 9. Primer čitanja sa sekundara.

# Primena replikacije

Najjednostavniji slučaj upotrebe replikacije se sastoji u formiranju replikacionog seta koji se sastoji od samo tri člana, obično primara, sekundara i arbitara (PSA arhitektura). Izgled arhitekture dat je na slici 10. Svaki od čvorova je potrebno pokrenuti kao zasebnu mongod instancu, koje imaju svoje direktorijume za skladištenje podataka. Na slici 11 dat je primer pokretanja čvorova sa potrebnim podešavanjima.



Slika 10. Izgled jednostavne PSA arhitekture.

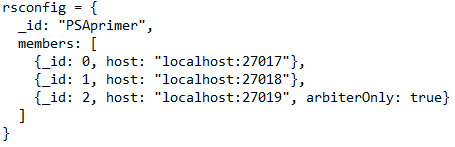


Slika 11. Komande upotrebljene za pokretanje replikacionog seta.

* **--repl\_set:** argument za definisanje imena replikacionog seta kome se čvorovi pridružuju.
* **--dbpath:** argument za definisanje putanja do direktorijuma čvora.
* **--port:** argument za vezivanje instance za port.
* **--oplogSize:** opcioni argument za definisanje veličine oplog-a.

Ukoliko nije navedeno, vrši se vezivanje za lokalni IP, tj. localhost. Ukoliko je potrebno navesti specifičnu IP adresu, koristi se argument „--bind\_ip“ nakon kojeg se navodi adresa.

Pokretanjem pojedinačnih čvorova i dalje se ne ostvaruje replikacija jer čvorovi nemaju informacije od ostalim članovima, tako da je u sledećem koraku potrebno povezivanje. Svi čvorovi jednog replikacionog seta dele zajedničku konfiguraciju. Najjednostavnija konfiguracija sastoji se od imena replikacionog seta i liste članova sa identifikatorom i adresom. Slika 12 prikazuje izgled korišćene konfiguracije. Set se inicijalizuje povezivanjem na jednom od članova i pokretanjem komande za inicijalizaciju uz prosleđivanje konfiguracije: **rs.initiate(rsconfig)**, nakon čega se vraća poruka o uspešnosti operacije. Kako su svi istog prioriteta, primar će biti odabran nasumično između dva „normalna“ servera.



Slika 12. Konfiguracija tročlanog PSA seta.

Osim osnovnih, konfiguracioni fajl može sadržati i druge opcije:

* **arbiterOnly:** uzima vrednost „true“ ili „false“ u zavisnosti da li je član u ulozi arbitera. Ne može se naknadno promeniti.
* **priority:** prioritet člana izražen brojem od 0 do 100. Veći broj označava veću važnost člana, odn. veći prioritet da postane primar. Poseban prioritet 0 označava pasivnog člana seta (ne može da postane primar).
* **votes:** broj glasova svakog člana. Osnovno podešavanje je 1 i u najvećem broju slučajeva nije potrebno menjati (loše podešavanje može da izazove rollback). Članu se može zabraniti glasanje. Ako postoji više od 7 članova replikacionog seta sa po jednim glasom, svaki naredni član mora da ima 0 glasova.
* **hidden:** označava da li član skriven, tj. da li je vidljiv u okviru seta od strane klijenta. Uvek ima prioritet 0, ne služi za čitanje, može da se koristi za rezervnu kopiju.
* **buildIndexes:** uzima vrednost „true“ ili „false“ koja određuje da li član replikuje indekse. Negativna vrednost se može iskoristiti samo kod pasivnih članova i služi za postojanje indeksa koji su različiti od onih koji se replikuju. Ne može se naknadno promeniti.
* **slaveDelay:** broj izražen u sekundama koliko neki od članova kasni za primarom. Korisno za obezbeđivanje dodatnog prostora za odklanjanje kritičnih grešaka, kao što su brisanje kolekcije ili korupcija podataka.
* **tags:** parovi ključ-vrednost koji obeležavaju člana (ili grupu članova). Služe za definisanje proizvoljnih pravila za upravljanjem potvrđivanja upisa i kontrole čitanja.

Od osnovnih podešavanja dozvoljeno je menjati IP adresu, ali tada sve adrese članova moraju biti lokalne ili udaljene, bez mešanja.

Proverom statusa preko komande **rs.status()** dobijaju se dodatne informacije o replikacionom setu vezane za ime seta, vreme, informacije o heartbeat-u, oplog-u, informacije o članovima kao što su identifikator, adresa, status, vreme rada, oplog kao i informacije o izborima. Na slikama 13 i 14, prikazan je izgled poziva komande za konkretni set.



Slika 13. Deo izveštaja koji prikazuje podatke o samom replikacionom setu.



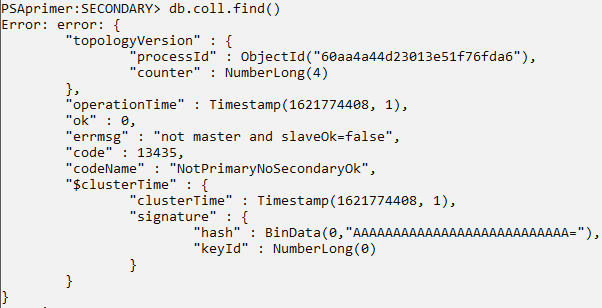
Slika 14. Deo izveštaja o članovima replikacionog seta.

Nakon što je replikacioni set uspostavljen i izabran primar, upravljanje setom se vrši izvršavanjem komandi nad primarom. Dobijanje informacija o setu može se obaviti i pozivom komande **db.isMaster()** čiji izlaz zavisi od člana seta koji je poziva. Najčešće se koristi za dobijanje informacija o primaru i sažetom izveštaju o setu. Slika 15 prikazuje izveštaj komande nakon pokretanja na primaru.

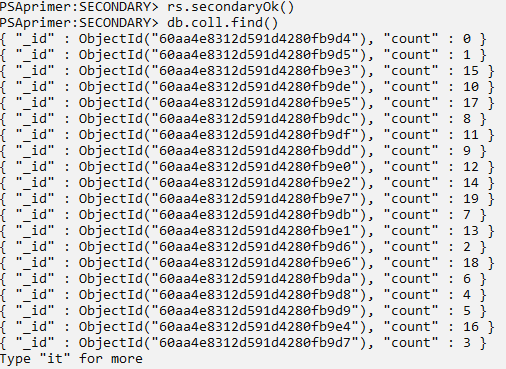


Slika 15. Izveštaj komande za proveru primara.

Osnovno podešavanje moda za čitanje jeste „primary“. To znači da je dozvoljeno čitanje samo sa primara. Pokušaj čitanja sa sekundara vraća poruku o grešci, kako se ne bi vratili potencijalno nekonzistentni podaci. Ukoliko je dozvoljeno postojanje nekonzistentnih podataka, čitanje sa određenog sekundara se omogućava pozivom komande **rs.secondaryOk()**. Na slikama 16 i 17 prikazano je čitanje sa sekundara pre i nakon izvršenja komande.

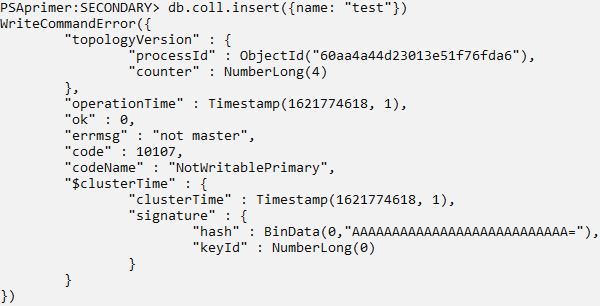


Slika 16. Greška nakon pokušaja čitanja sa sekundara.



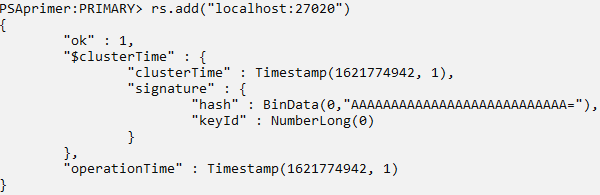
Slika 17. Uspešno čitanje sa sekundara nakon izvršenja komande.

Pokušaj pisanja na sekundaru takođe vraća grešku. Međutim, ne postoji podešavanje koje će dozvoliti ovakvo ponašanje. Sva pisanja se moraju obaviti samo na primaru. Zaobilazak ovog mehanizma je moguć samo forsiranjem failover-a gašenjem primara. Na slici 18 prikazana je greška koju sekundar vraća.

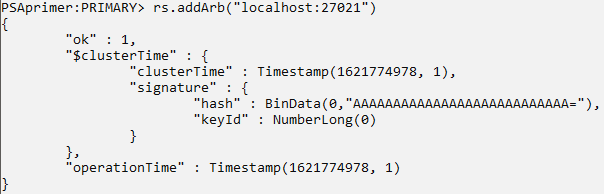


Slika 18. Greška kod pokušaja pisanja na sekundaru.

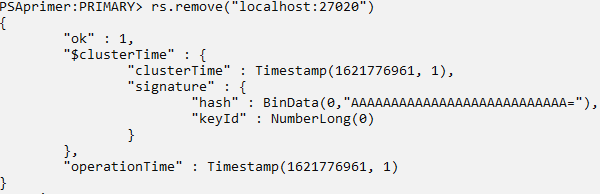
Replikacioni set od samo tri člana mali je za konkretne upotrebe. Proširienje seta obavlja se pokretanjem novih servera i dodavanjem istih u postojeći set. Pozivom komande **rs.add(<address>)** dodaje se novi server kao „normalni“ član navođenjem adrese servera. Iako je moguće navesti podešavanje u okviru konfiguracionog fajla, ostavljena je mogućnost za lako dodavanje arbitera pozivom komande **rs.addArb(<address>)**, takođe navođenjem adrese servera. Suprotno tome, izbacivanje servera iz seta obavlja se komandom **rs.remove(<address>)** navođenjem adrese željenog servera. Upotreba ovih komandi se ne preporučuje kada su potrebna napredna podešavanja servera. Prikaz rada ovih komandi dat se na slikama 19, 20 i 21, respektivno.



Slika 19. Dodavanje servera u replikacioni set.

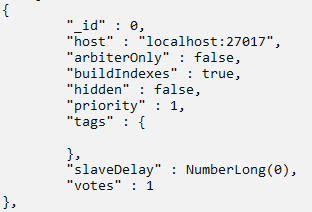


Slika 20. Dodavanje arbitera u replikacioni set.

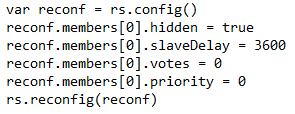


Slika 21. Izbacivanje servera iz replikacionog seta.

Dodavanjem novih servera u replikacioni set, dodaju se i odgovarajući elementi u okviru konfiguracionog fajla. Međutim, bez eksplicitnog navođenja, svi novi serveri će raditi u osnovnom podešavanju. Izmena novih i postojećih servera moguća je upotrebom rekonfiguracije. Slika 22 prikazuje izgled stavke jednog člana seta, dok slika 23 prikazuje izvršene izmene. Treba napomenuti da nove konfiguracije moraju biti ispravne da bi bile prihvaćene. U suprotnom se vraća poruka sa informacijama o grešci.



Slika 22. Izgled konfiguracije jednog člana seta.



Slika 23. Primer navođenja rekonfiguracije za neki od članova seta.

Izvršenjem komande **rs.reconfig(reconf)** sa navedenim podešavanjima, server se skriva i odlaže za jedan sat (replikacija kasni jedan sat za primarom). Za korektno ispisanu konfiguraciju, potrebno je navesti sve zavisne vrednosti eksplicitno. U slučaju skrivanja servera, vrednosti glasova i prioriteta moraju biti prisutne i postavljene na 0.

Rekonfiguracija je dozvoljena samo sa aktivnih članova, i to najčešće sa primara. Ako se rekonfiguracija forsira sa nekog od sekundara, potrebno je omogućiti ponašanje preko promenljive „force“ postavljene na vrednost „true“. Nova konfiguracija mora da bude ispravna i može da bude primenjena samo sa člana koji je bio uključen u prethodnu važeću konfiguraciju. Verzija konfiguracije ostalih članova se uvećava za veliki broj, kako bi se izbegli mogući konflikti.

## Upravljanje članovima replikacionog seta

Upravljanje članovima replikacionog seta nije moguće u toku njihovog rada. Nekada je potrebno izvršiti određene izmene na nekom serveru, ili mu pristupiti u svrhe održavanja. Da bi se omogucio rad sa ovakvim serverima, potrebno ih je pokrenuti kao samostalne servere koji ne pripadaju ni jednom setu. Ovo se postiže pokretanjem servera u režimu bez replikacije. Time se postiže pisanje i čitanje bez ograničenja i bez uticaja na ostale članove seta. Nakon što su obavljene sve potrebne korekcije, server se može ponovo priključiti setu i nastavlja sa replikacijom od poslednje stavke u oplog-u.

Od ostalih komandi nad serverom treba napomenuti i mogućnost provere sa kog se člana svaki od sekundara sinhronizuje. Time se ima uvid u potencijalno kašnjenje sekundara, a moguć je i vizuelni prikaza replikacionog seta (graf replikacije). Svaki od sekundara takođe poseduje i znanje o tome koliko kasni sa replikacijom u odnosu na primar.

Pažnju treba posvetiti i građenju indeksa u okviru replikacionog seta. Poznato je da je građenje indeksa skupa operacija koja se dodatno usložnjava prisustvom drugih servera koji takođe moraju da imaju kopiju istog. U najgorem slučaju, odmah nakon primara, svi sekundari kreću sa replikovanjem indeksa u isto vreme. Time se ostvaruje veliko opterećenje na mreži, što uzrokuje velikom latencijom u odgovoru. Rešenje problema se ogleda u pravljenju indeksa na svakom od sekundara koji su pokrenuti kao samostalni serveri, pre vraćanja u set. Za pravljenje indeksa na primaru moguća su dva scenarija. Prvi scenarijo jeste pravljenje indeksa kada je opterećenje na serveru malo. Replikovanje na sekundarima je trenutno jer već poseduju indekse. Drugi scenarijo jeste pravljenje indeksa po istom principu kao kod sekundara, uz automatski failover. Treba napomenuti da su indeksi na svim članovima seta istovetni zbog sinhronizacije. Za postojanje različitih indeksa, potrebno je prebaciti određeni server u pasivan mod.

# Zaključak

Replikacija kod MongoDB baze podataka predstavlja jednostavan, ali veoma moćan koncept kojim se ostvaruje velika izdržljivost i dostupnost. Organizovanjem replikacije oko koncepta replikacionih setova, pruža se mogućnost za veliko automatizovanje čitavog procesa formiranja i održavanja istog. Postojanje automatskog failover-a pruža sigurnost pri padu jednog, ili čak više servera. Podaci ostaju dostupni sve dok postoji većina servera, koji će mehanizmom izbora i glasanja odabrati najpogodniji server da preuzme ulogu primara. Održavanje dostupnosti servera obavlja se većinom automatskim putem upotrebom heartbeat-a, male poruke koju članovi seta razmenjuju za proveru informacija o okruženju. Kopiranje između servera u svrhe održavanja konzistentnosti i sinhronizacije svodi se na kopiranje stavki iz oplog-a, lokalne baze podataka koja čuva sve izmene nad primarom. Operacije se zatim izvršavaju istovetno na svakom sekundaru, tako da nema dodatnog opterećenja na sam set. Još jedan mehanizam za održavanje rada seta ogleda se u postojanju automatskog rollback-a, operacije koja briše nereplikovane upise, ukoliko primar padne pre nego što dovoljan broj sekundara pročita iz oplog-a.

Osim održavanja dostupnosti, upotreba replikacije ogleda se u skaliranju čitanja. Omogućavanjem čitanja sa sekundara, postiže se brz odgovor, uz eventualne probleme konzistentnosti podataka. Funkcija upisa i dalje ostaje samo na primaru, dok se kombinovanjem sa particionisanjem i ovde ostvaruje skaliranje.

Upravljanje na nivou replikacionog seta ili samih servera sastoji se na poziv komandi jasno definisanog interfejsa, sa čitljivim izveštajima o greškama, ukoliko dođe do njih. Nadgledanje se postiže korišćenjem komandi koje pružaju uvid u različite delove rada replikacionog seta, kao što su podaci o samom setu, serverima koji ga čine ili o nekim specifičnostima mehanizama koji pokreću replikaciju. MongoDB baza podataka pruža dovoljno fleksibilnosti u korišćenju, tako da je pogodna i za najmanje aplikacije, kao i za one koje se prostiru širom sveta.

# Literatura

1. Knjiga - MongoDB 4 Quick Start Guide Learn the skills you need to work with the world’s most popular NoSQL database by Doug Bierer
2. Knjiga - MongoDB in Action Covers MongoDB version 3.0 by Kyle Banker, Peter Bakkum, Shaun Verch, Doug Garrett, Tim Hawkins
3. Knjiga - Mastering MongoDB 4.x expert techniques to run high-volume and fault-tolerant database solutions using MongoDB 4.x by Alex Giamas
4. Knjiga - MongoDB The Definitive Guide by Kristina Chodorow, Michael Dirolf
5. Knjiga - MongoDB The Definitive Guide Powerful and Scalable Data Storage 3rd Edition by Kristina Chodorow
6. Slike 1, 2, 6, 8, 9, 10 preuzete sa MongoDB zvanične dokumentacije o replikaciji: <https://docs.mongodb.com/manual/replication/>