UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET

Replikacija podataka u MongoDB bazi podataka

Master akademske studije
Seminarski rad iz predmeta Sistemi za upravljanje bazama podataka

Mentor:
Prof. dr Aleksandar Stanimirović

Student:
Sanja Milenković 1549

Niš, septembar 2024. godine

Sadržaj

1. Uvod

Najranije baze podataka zasnivale su se na jedinstvenom, nezavisnom serveru. Ovakav pristup, iako predstavlja dobru polaznu osnovu, sa daljim razvojem i proširenjem sistema potencijalno može stvoriti probleme. Naime, šta će se desiti sa bazom podataka ako server prestane da funkcioniše ili postane nedostupan? Baza podataka će neminovno određeno vreme biti nedostupna. Ako, na primer, nastupi neki hardverski problem, možda će rešenje iziskivati premeštanje baze podataka na neki drugi server. U najgorem mogućem slučaju, koji je svakako verovatan, različiti problemi mogu podatke čuvane u bazi podataka uništiti i učiniti neupotrebljivim. Ovi rizici jasno ukazuju na potrebu za replikacijom, tehnikom koja omogućava čuvanje identičnih kopija podataka na više servera.

Replikacija omogućava da podaci budu bezbedni i da baza podataka neometano nastavi da funkcioniše, čak i ako dođe do greške na jednom od servera. Primera radi, ukoliko jedan server usled greške ili iz nekog drugog razloga postane nedostupan, drugi serveri mogu preuzeti njegovu ulogu, omogućavajući neometano funkcionisanje sistema. Takođe, replikacija pomaže u zaštiti podataka od gubitaka usled hardverskih grešaka, jer se podaci čuvaju na više lokacija.

Pored povećanja dostupnosti i sigurnosti podataka, replikacija može poboljšati performanse baze podataka. Distribucijom zahteva za čitanje na više servera, moguće je smanjiti opterećenje na pojedinačne servere i ubrzati pristup podacima. Ovaj pristup je posebno koristan u velikim sistemima gde je brzina pristupa podacima kritična.

U kontekstu modernih baza podataka, replikacija igra ključnu ulogu u obezbeđivanju visoke dostupnosti, otpornosti na greške i skalabilnosti. Upravo zbog svih navedenih razloga, razumevanje i implementacija replikacije postali su neophodni za stabilno i pouzdano funkcionisanje savremenih sistema za upravljanje bazama podataka.

2. Replikacija podataka

Replikacija podataka ostvaruje se čuvanjem višestrukog broja kopija istih podataka na većem broju servera koji su međusobno umreženi. Postoji više razloga zašto je potrebno obaviti replikaciju podataka:

- omogućava smanjenu latenciju pristupa podacima, budući da omogućava čuvanje podataka na lokaciji koja je geografski bliža samom korisniku
- povećava dostupnost sistema, budući da omogućava sistemu da neometano nastavi da funkcioniše, čak i ako je došlo do otkaza nekih od delova sistema
- omogućava veći protok čitanja, jer se replikacijom zapravo vrši horizontalno skaliranje servera koji mogu opslužiti sve primljene upite

Replikacija se može smatrati jednostvanim konceptom ako se podaci nad kojima se vrši replikacija ne menjaju – u tom kontekstu, potrebno je samo jednom kreirati kopiju podataka i replikacija se može smatrati obavljenom i završenom. Međutim, sva kompleksnost replikacije zapravo potiče od potrebe da se promene podataka ispropagiraju kroz sve kopije. U nastavku rada biće kratko dat osvrt na 3 najpopularnija algoritma za replikaciju podataka: **single-leader**, **multi-leader** i **leaderless replikacija**.

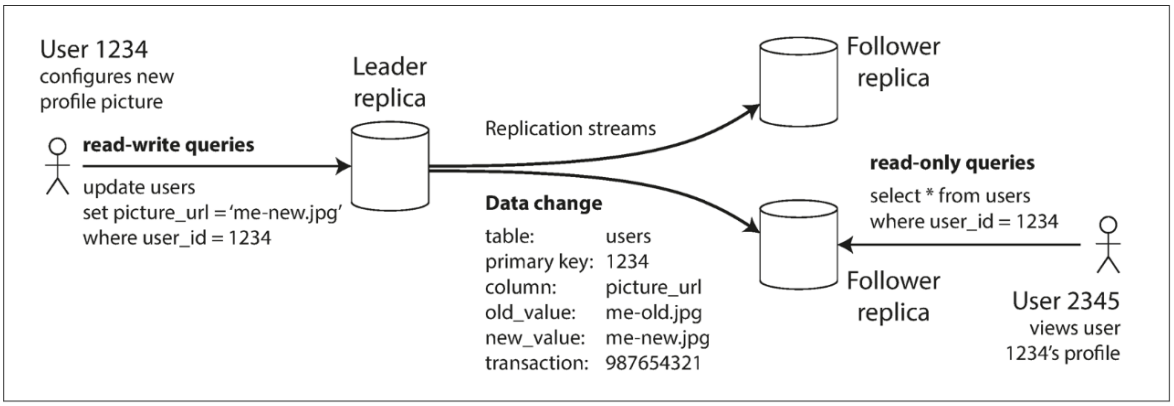
Postoji mnoštvo kompromisa koje je potrebno razmotriti u kontekstu same replikacije. Primera radi, da li je bolje upotrebiti sinhronu ili asinhronu replikaciju i kako rukovati replikama u kojima je nastala greška. Ovo zapravo jesu konfiguracije na nivou same baze podataka i, iako pojedini detalji mogu varirati između različitih baza podataka, generalni principi jesu slični.

2.1 Single-leader replikacija

Svaki čvor koji čuva kopiju podataka naziva se **replikom**. U sistemima koji se sastoje od više replika neminovno se nameće pitanje kako ostvariti konzistentnost podataka kroz sve replike?

Svaki upis u bazu podataka potrebno je obraditi od strane svake replike. U suprotnom, konzistentnost same baze podataka bila bi narušena, budući da bi replike čuvale različite verzije podataka. Najčešće rešenje ovog problema jeste tzv. **lider-zasnovana replikacija** (eng. *leader-based replication,* u zavisnosti od literature, još se mogu sresti i nazivi **aktivna-pasivna** i **master-slave replikacija**). Naime, ovo rešenje podrazumeva najpre određivanje jedne replike koja će biti proglašena **liderom** (master replikom ili aktivnom replikom). Kada je potrebno obaviti upis podataka u bazu podataka, zahtevi se prosleđuju upravo replici lideru, koja najpre nove podatke čuva lokalno. Sve ostale replike u sistemu nazivaju se **replikama pratiocima** (još su u upotrebi termini slave replike, replike za čitanje, sekundarne replike itd.). Kada god su novi podaci sačuvani u okviru master replike, te promene podataka bivaju takođe prosleđene ka svim replikama pratiocima, nakon čega će se na svim replikama ažurirati lokalne kopije podataka, poštujući pritom redosled promena. Kada klijent zatraži podatke iz baze podataka, on se može ovim zahtevom obratiti i liderima i replikama pratiocima, međutim ukoliko se

zahtev odnosi na upis u bazu podataka, on može biti obrađen isključivo na replici lideru. Iz ugla klijenta, replike pratioci su isključivo **read-only**.



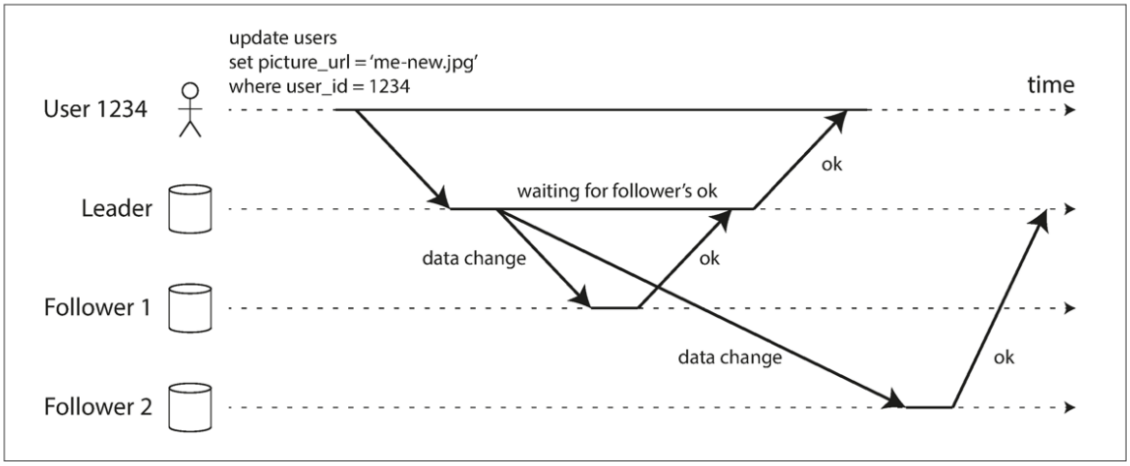
Slika 1. Lider zasnovana replikacija

Ovaj model replikacije koristi se u nekim relacionim bazama podataka, poput PostgreSQL i MySQL baze podataka. Takođe koristi se i u nekim nerelacionim bazama podataka, poput **MongoDB**, RethinkDB i drugih.

2.1.1 Sinhrona i asinhrona replikacija

Bitna stavka o kojoj treba pričati u kontekstu replikacije jeste način njenog odvijanja, tj. da li će se ona biti **sinhrona** ili **asinhrona**. U relacionim bazama, ovo je, u najvećem broju slučajeva, opcija koja se može konfigurisati, dok je u drugim sistemima najčešće unapred definisano jedno od dva moguća ponašanja.

Na slici 2 dat je prikaz komunikacije replike lidera sa sinhronim i asinhronim pratiocem. Naime, pratilac 1 je **sinhroni**, što znači da replika lider potvrdu o uspešno obavljenoj operaciji upisa korisniku prosleđuje tek nakon što je sinhroni pratilac potvrdio upis ovih podataka i tek nakon toga, izmene su vidljive i ostalim klijentima. Pratilac 2 je **asinhroni** – replika lider prosleđuje podatke, ali ne čeka na potvrdu od same replike.



Slika 2. Lider zasnovana replikacija sa sinhronim i asinhronim replikama pratiocima

U najvećem broju slučajeva, replikacija jeste brz proces budući da se izmene nad replikama pratiocima primenjuju u svega nekoliko sekundi. Međutim, sam proces potencijalno može

trajati i duže. Postoje okolnosti koje mogu uticati na kašnjenje replika pratioca za liderom i za po nekoliko minuta, ako se na primer dešava oporavak od otkaza ili ako je došlo do greške na mreži.

Glavna prednost sinhrone replikacije jeste garancija da će replike pratioci imati up-to-date kopiju podataka, koja je konzistentna sa podacima na samom lideru. U slučaju neočekivanog otkaza replike lidera, sinhrona replikacija nam garantuje da validna verzija podataka postoji na još makar jednoj replici. Negativna strana ovog tipa replikacije odnosi se na blokiranje operacije upisa, ako se desi da sinhrona replika iz nekog razloga ne odgovara. U ovom slučaju, lider mora da blokira sve dalje upisa i čeka sve dok sinhrona replika ne bude ponovo dostupna. Imajući sve prethodno navedeno u vidu, definisanje sistema tako da sve replike budu sinhrone potencijalno može da dovede do blokade celog sistema, samo usled otkaza u jednom čvoru. U praksi se definisanje sinhronih replika uglavnom odnosi na definisanje **jedne** takve replike, dok su sve ostale asinhrone. Ako se desi da odabrana sinhrona replika postane nedostupna ili izuzento spora, jedna od prethodno asinhronih replika preuzima njenu ulogu i prevodi se u sinhronu. Na ovaj način garantovano je postojanje up-to-date kopije podataka na makar dva čvora – replici lideru i sinhronom pratiocu. Ova konfiguracija se često naziva i **polu-sinhronom**. Često se međutim dešava da je lider-zasnovana replikacija apsolutno asinhrona. Ako je to slučaj i dođe do otkaza lidera, svi upisi koji još uvek nisu replikovani trajno su izgubljeni. S druge strane, ovakva konfiguracija ima i svoje prednosti koje se pre svega ogledaju u mogućnosti lidera da opslužuje sve zahteve za upisom u bazu podataka, bez obzira na to da li same replike pratioci kasne u obradi. Iako slabljenje održivosti podataka može zvučati kao loš kompromis, asinhrona replikacija je široko korišćeni mehanizam, posebno ako postoji veliki broj replika sledbenika koje su geografski udaljene.

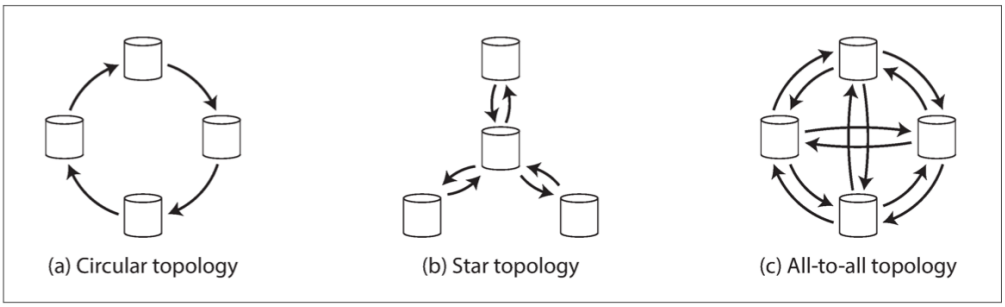
2.2 Multi-leader replikacija

Replikacija zasnovana na jedinstvenom lideru, koja je bila objašnjena u prethodnom poglavlju, ima jedan glavni nedostatak – svi upisi u bazu podataka moraju biti obavljeni preko jednog čvora – čvora lidera. Ukoliko je, iz bilo kog razloga, nemoguće uspostaviti konekciju sa liderom, takođe će biti nemoguće izvršiti bilo kakav upis u bazu podataka. Prirodno proširenje ovog modela replikacije podrazumeva omogućavanje višestrukom broju replika da prihvati operacije upisa. Sam proces replikacije i u takvoj organizaciji obavljaće se na isti način – svaki čvor koji obrađuje upise u bazu podataka mora te podatke proslediti svim ostalim replikama. Ova konfiguracija naziva se **multi-leader replikacija**, a često se sreću i nazivi master-master ili active-active replikacije.

2.2.1 Topologije multi-leader replikacije

Topologija replikacije opisuje putanje komunikacije putem kojih se upisi u bazu podataka propagiraju kroz sve čvorove u sistemu. Ukoliko govorimo o sistemu sa 2 lidera, postoji samo jedna topologija – lider 1 mora sve upise da prosledi do lidera 2 i obratno. Ako pak govorimo

o sistemima sa više od dva lidera, višestruke topologije su moguće, a samo neki od primera dati su na slici 3.



Slika 3. Primeri topologija multi-lider replikacije

Najopštija topologija je all-to-all, prikazana na slici 3-c. Ovakva organizacija podrazumeva da će svaki lider operacije upisa prosleđivati svim ostalim liderima u sistemu. Ipak, nešto restriktivnije topologije su takođe široko korišćene, primera radi, MySQL podrazumeva samo cirkularnu topologiju, koja je prikazana na slici 3-a. U ovoj topologiji svaki čvor prima upise od samo jednog, susednog čvora i te izmene prosleđuje, zajedno sa svojim izmenama, ako ih ima, sledećem čvoru u sistemu. Na slici 3-b prikazana je topologija zvezde, gde jedan koreni čvor prosleđuje sve operacije upisa svim ostalim čvorovima u sistemu. Cirkularna topologija i topologija zvezde podrazumevaju da upis mora proći kroz nekoliko čvorova pre nego stigne do svih replika, pri čemu su čvorovi zaduženi da ostalim replikama proslede izmenjene podatke koje su primili. Kako bi se prevenirale beskonačne petlje replikacije, svaki čvor u sistemu jedinstveno je identifikovan i svaki upis taguje se identifikatorima čvorova ka kojima su ove promene već propagirane. Ako neka replika primi podatke koji su već tagovani njegovim sopstvenim indikatorom, izmene podataka se ignorišu, jer to znači da je konkretna replika već obradila ove izmene.

2.3 Leaderless replikacija

Obe prethodno opisane konfiguracije replikacije zasnivale su se na istoj ideji – klijent zahteve za upisom u bazu podataka šalje samo jednom čvoru – čvoru lideru, a sam sistem baze podataka zadužen je za kopiranje ovih izmena i njihovo propagiranje do ostalih replika u sistemu. Lider je taj koji definiše kojim će se redosledom upisi izvršavati, a replike sledbenici primenjuju izmene nad podacima u istom redosledu. Neki sistemi baziraju se na nešto drugačijem pristupu, napušta se koncept lidera i dozvoljava se svim replikama da direktno prihvate zahteve za upisom od klijenta. Neki od najranijih sistema koji su podržavali replikaciju bili su leaderless, ali je ovaj koncept postao manje popularan u eri dominacije relacionih baza podataka. Ipak, ova konfiguracija ponovo je postala popularna nakon što je Amazon upotrebio u svom rešenju, DynamoDB-u. Mnoge open-source baze podataka su potom, inspirisane ovim primerom, implementirale isti tip replikacije, pa se ova konfiguracija često naziva i Dynamo-style replikacijom.

U nekim verzijama implementacije ove replikacije, klijent direktno šalje zahteve za izmenama podataka ka nekoliko replika, dok u nekim drugim verzijama, postoji čvor koordinator koji se time bavi. Ipak, za razliku od sistema koji imaju definisanog lidera,

koordinator ne forsira redosled kojim će se upisi izvršavati, što značajno utiče na način korišćenja ovih baza podataka.

3. Replikacija u MongoDB bazi podataka

MongoDB je dokument-orijentisana baza podataka, karakterišu je visoka fleksibilnost i skalabilnost. Veoma grubom analogijom, jedan dokument u MongoDB bazi podataka može se smatrati jednim redom u relacionoj bazi podataka, ali sa daleko fleksibilnijim modelom. Naime, ugnježdavanje dokumenata omogućava da se u bazi podataka predstave kompleksne hijerarhijske veze. Nema predefinisanih shema podataka, te ključevi i vrednosti u samim dokumentima nemaju fiksirane tipove ili veličine.

Dokumenti se dalje mogu organizovati u kolekcije, koje su ekvivalent tabelama u relacionim bazama podataka. Kolekcije odlikuju dinamičke sheme, što znači da dokumenti u okviru jedne kolekcije mogu imate apsolutnu različitu strukturu, iako se to smatra lošom praksom.

3.1 Set replika u MongoDB bazi podataka

Generalna priča o replikaciji, koja je bila opisana u prvom poglavlju ovog seminarskog rada, ne menja se mnogo ni kada govorimo konkretno o MongoDB bazi podataka. Replikacija podrazumeva način kojim se identične kopije podataka čuvaju na višestrukom broju servera i preporučeni je način upotrebe MongoDB baze podataka u produkciji. Replikacija omogućava da baza podataka nastavi neometano da funkcioniše, čak i ako nastupi greška na nekom od servera.

Replikacija se u MongoDB bazi podataka podešava kreiranjem **seta replika**. Set replika podrazumeva postojanje grupe servera, od kojih neki čuvaju podatke i među kojima, opciono, može postojati jedan **čvor arbitar**. Među čvorovima koji čuvaju podatke isključivo jedan je proglašen tzv. **primarnim serverom** koji prima klijentske zahteve dok se ostali smatraju **sekundarnim serverima** i oni čuvaju kopiju podataka primarnog servera. Za slučaj da dođe do otkaza primarnog servera, sekundarni serveri vrše izbor novog primarnog servera između sebe. Bitno je naglasiti da svaka replika može pripadati isključivo jednom setu replika.

U konktekstu MongoDB-ja, set replika je grupa **mongod** procesa, koji održavaju isti set podataka. Kroz postajenje setova replika ostvarena je redundantnost i visoka dostupnost. U pojedinim slučajevima, replikacija može da poveća i kapacitet čitanja iz baze podataka, budući da klijenti zahteve za čitanjem mogu da pošalju različitim serverima.

3.1.1 Kako dizajnirati set replika?

Kako bi se kreirao set replika koji je održiv, postoji nekoliko koncepata koje treba ispoštovati, među kojima se najbitniji odnose na **većine**. Naime, potrebna je većina svih replika kako bi se izglasao novi primarni server, primarni server ostaje primarni sve dok može da uspostavi vezu sa većinom replika u setu i upis u bazu podataka smatra se uspešnim i sigurnim samo ako je replikovan na većini replika u setu. Većina se definiše kao **više od polovine od ukupnog broja članova seta**, pri čemu se iz kalkulacije ne isključuju replike koje su trenutno nedostupne, već se većina definiše nad samom konfiguracijom baze podataka.

3.1.2 Sinhronizacija replika

MongoDB replikaciju postiže kroz čuvanje logova operacija, **oplog**-a (eng. *operations log*), koji sadrži informacije o svim upisima u bazu podataka putem primarnog servera. Radi se, dakle, o ograničenoj kolekciji koja se čuva u lokalnoj bazi podataka primarnog servera. Sekundarni serveri, potom, zahtevaju pristup ovoj kolekciji kako bi dobili informaciju o operacijama koje treba da replikuju, kreiraju lokalnu kopiju i potom primenjuju te operacije nad lokalnim podacima, kako bi se poistovetili sa podacima primarnog servera. Svaki sekundarni server održava sopstveni oplog, gde vodi evidenciju o svim operacijama koje je replikovao sa primarnog servera čime je omogućeno da bilo koji član postane izvor sinhronizacije za sve ostale članove u setu.

3.2 Tipovi članova seta replika

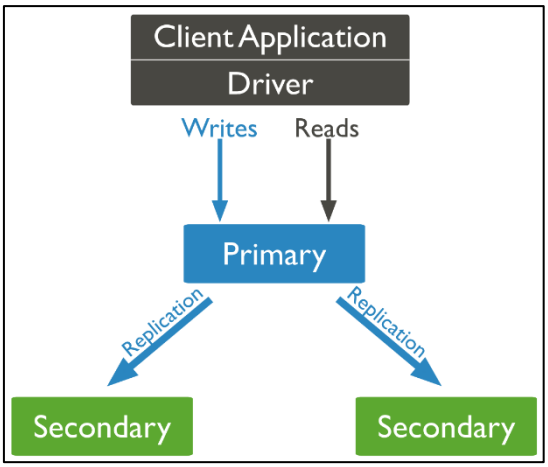
Članovi seta replika mogu biti:

- primarni
- sekundarni
- arbitri.

Minimalna preporučena konfiguracija seta replika podrazumeva da se u samom setu nalaze minimum 3 člana i to jedan primarni i dva sekundarna servera. Ukoliko dođe do situacije da je dodavanje drugog sekundarnog servera skupo, moguće je umesto njega uključiti arbitra. Naime, čvor arbitar učestvuje u glasanju, ali ne čuva kopiju podataka, te ne doprinosi redundantnosti podataka. MongoDB setovi ograničeni su na 50 replika, a maksimalno 7 replika može učestvovati u glasanju.

3.2.1 Primarni članovi

Kao što je ranije naglašeno, primarni server jeste jedini čvor u sistemu koji može da **prima operacije upisa**. MongoDB sve operacije upisa najpre primenjuje na primarnom čvoru, a redosled izvršenih operacija čuva u **oplog**-u primarnog servera. Sekundarni serveri, potom, replikuju ovaj oplog i primenjuju operacije nad svojim setom podataka, kao što je prikazano na slici 4. S druge strane, sve replike mogu da prihvataju operacije čitanja, međutim, podrazumevano ponašanje je da se čak i operacije čitanja prosleđuju primarnom čvoru.

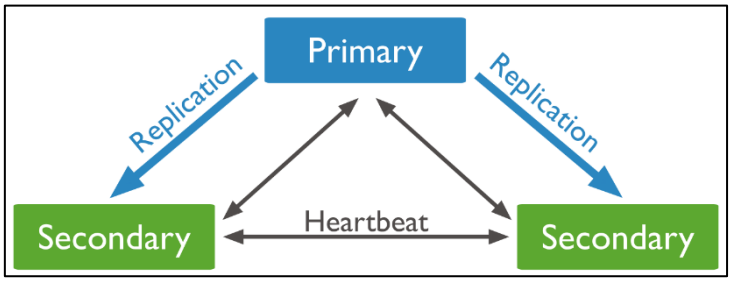


Slika 4. Set replika od 3 člana

3.2.2 Sekundarni članovi

Sekundarni serveri održavaju kopiju podataka sa primarnog servera. Kako bi replikovali podatke, sekundarni serveri primenjuju operacije onim redosledom kojim su zapisani u oplog-u primarnog servera na asinhroni način. Kao što je već naglašeno, u jednom setu replika moguće je imati jedan ili više sekundarnih čvorova.

Na slici 5, dat je primer seta replika koji ima 3 člana, od kojih su dva sekundarni članovi. Svaki sekundarni server može postati primarni. Naime, za slučaj da trenutni primarni čvor postane nedostupan, održavaju se **izbori** (eng. *elections*) na nivou seta replika, kojima se određuje koji će tačno sekundarni čvor postati novi primarni. Provera da li je neki čvor još uvek aktivan vrši se razmenom heartbeat-ova. Heartbeat se šalje na svake 2 sekunde. Svaki čvor koji primi heartbeat mora da se na isti odazove. Ako se dogodi da se jedan čvor ne odaziva na 5 uzastopnih heartbeat-ova, ostale replike u setu smatraju da je on nedostupan.



Slika 5. Komunikacija između replika u setu

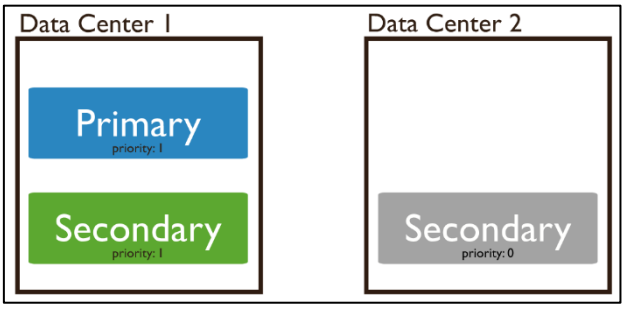
Zavisno od konfiguracije, MongoDB razlikuje tri podtipa sekundarnih čvorova:

- Članovi sa prioritetom 0 (eng. *priority 0 members*)
- Skriveni članovi (eng. *hidden members*)
- Odloženi članovi (eng. *delayed members*)

Članovi sa prioritetom 0

Član sa prioritetom 0 jeste replika seta koja ne može da postane primarni server niti može pokrenuti izbore. Ovi članovi mogu da potvrde operacije upisa čiji je **write concern** konfigurisan određenim brojem replika od kojih se očekuje potvrda. Write concern definiše stepen potvrde koju MongoDB zahteva kako bi operacije upisa smatrao uspešnim, pri čemu se može zahtevati da upis bude propagiran do određenog broja replika, do instanci koje imaju neki specifični tag ili jednostavno do većine (eng. *majority*). Ako je write concern postavljen na **majority**, član sa prioritetom 0 mora takođe biti i član sa pravom glasa, tj. **members[n].votes** mora biti veće od 0. Kao što je već napomenuto, ukupan broj replika u setu može maksimalno iznositi 50, a kako je maksimalni broj onih koji mogu glasati 7, putem ovog polja moguće je prilikom same konfiguracije replike dodeliti joj ukupno 1 glas ili joj ih u potpunosti zabraniti.

Osim gorenavedenih ograničenja, sekundarni serveri čiji je prioritet 0 funkcionišu kao i svi ostali sekundari, čuvaju jednu kopiju podataka, opslužuju operacije čitanja i glasaju na izborima. Server treba konfigurisati kao člana sa prioritetom 0 u slučajevima kada se konkretni član nalazi u data centru koji je udaljen u odnosu na glavni deployment centar, pa stoga potencijalno može imate veću latenciju. Na ovaj način, on može obavljati lokalne zahteve čitanja, ali ne može biti idealni kandidat za obavljanje dužnosti primarnog servera, upravo zbog latencije.

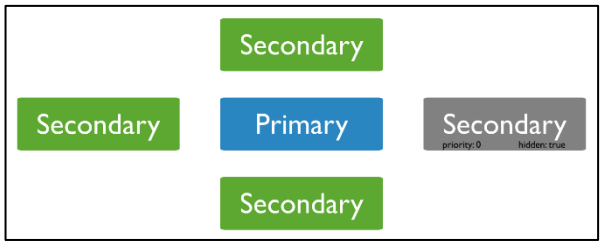


Slika 6. Ilustracija člana sa prioriretom 0

Na slici 6, dat je prikaz 2 data centra - data centar 1, u kome se nalaze primarni i sekundarni server i data centar 2, u kome se nalazi sekundarni server, koji je konfigurisan tako da ima piroritet 0, čime je onemogućeno njegovo proglašenje za primarni server. U slučaju otkaza aktuelnog primarnog čvora, usled kreirane konfiguracije, samo članovi iz data centra 1 mogu da postanu primarni prilikom izbora.

Sakrivene replike

Sakrivene replike čuvaju kopiju podataka koja se nalazi na primarnom serveru, ali su skrivene za klijentske aplikacije. Sakrivene replike moraju uvek biti članovi sa prioritetom 0, jer nikako ne bi smeli postati primarni čvorovi u setu.



Slika 7. Sakrivena replika

Ponašanje ovih replika razlikuje se od klasičnih sekundarnih čvorova, budući da klijentski zahtevi za čitanjem iz baze podataka neće biti prosleđivani sakrivenim replikama. Kao rezultat toga, ovi čvorovi mimo bazične replikacije, ne učestvuju u drugom saobraćaju. Stoga, skrivene replike treba koristiti za određene taskove poput izveštavanja i backup-ovanja. Sa druge strane, sakrivene replike mogu glasati u izborima na nivou setova replika. Ponašanje ovih čvorova prilikom upisa tj. potvrde upisa je slično ponašanju koje imaju čvorovi sa prioritetom 0. Ove replike mogu potvrditi upis ako je write concern definisan samo brojem replika čija je potvrda potrebna. Ako je write concern postavljen na **majority**, skrivena replika mora takođe biti i replika sa pravom glasa, tj. **members[n].votes** mora biti veće od 0.

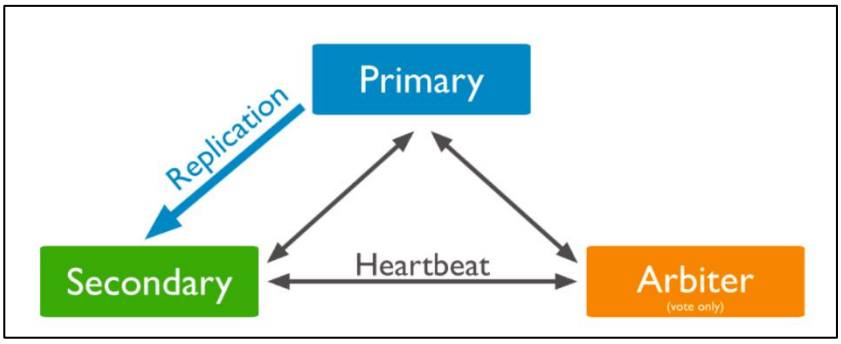
Odloženi članovi

Odloženi članovi, takođe, čuvaju kopiju podataka. Međutim, kopija koju oni poseduju nije aktuelna trenutno, već se radi o kopiji koja je bila aktuelna ranije, ili će biti aktuelna kasnije. Na primer, ako je trenutno vreme 9:52, a odloženi član je podešen da ima sat vremena kašnjenja, to bi značilo da je poslednja operacija koja je izvršena nad podacima odloženog člana obavljena najkasnije u 8:52. Ove se replike smatraju „rolling backup“ replikama ili istorijskim snimkom podataka i mogu poslužiti u svrhu oporavka usled otkaza nastalih ljudskog greškom. Ove članove seta replika prate i određena ograničenja. Naime, prioritet ovih čvorova mora biti 0, kako bi se onemogućilo njihovo proglašenje za primarne čvorove. Ovi čvorovi moraju, takođe, biti sakriveni, kako bi se preveniralo slanje zahteva za podacima čvorovima koji nemaju ažurnu kopiju podataka. U zavisnosti od konfiguracije, tj. broja glasova koji su im dodeljeni kroz polje **members[n].votes**, mogu učestvovati u glasanju, ali mogu i ostati bez prava glasa i to se generalno smatra boljom praksom.

3.2.3 Članovi arbitri

U pojedinim slučajevima, kada imamo set primarnih i sekundarnih replika, gde bi dodavanje još jednog sekundarnog čvora bilo jako skupo, moguće je umesto njega dodati čvor arbitar. Ovi čvorovi učestvuju u izborima novog primarnog čvora, ali ne poseduju kopiju podataka i sami ne mogu biti izglasani za primarni čvor. Arbitar mora imati pravo glasa, a podrazumevano ima prioritet 0.

Kako bi se izbegli potencijalni problemi sa konzistentnošću podataka, bolje je kreirati sisteme sa jedinstvenim arbitrom. Postojanje većeg broja čvorova arbitara u sistemu može sprečiti pouzdanu upotrebu većine za potvrdu upisa.



Slika 8. Skup replika sa čvorom arbitrom

3.3 Oplog u MongoDB bazi podataka

Kao što je ranije pomenutp, oplog (eng. *operations log*, log operacija) je ključna komponenta MongoDB replikacije. Radi se o specijalnoj, ograničenoj kolekciji koja beleži sve promene podataka u bazi podataka, odnosno sve operacije upisa koje modifikuju stanje podataka. Primarni član replikacionog seta primenjuje operacije i beleži ih u svoj oplog, dok sekundarni članovi asinhrono repliciraju te promene tako što ih čitaju iz oplog-a i primenjuju u svojim instancama baze onim redom kojim su i zapisane. Svaki sekundarni član seta replika ima **svoju kopiju** oploga u kolekciji **local.oplog.rs**, koja omogućava sinhronizaciju sa primarnim članom. Sekundarni članovi mogu preuzimati oplog čak i drugih članova seta, ne

samo od primarnog, što dodatno doprinosi otpornosti sistema. Oplog operacije su **idempotentne**, što znači da njihovo višestruko izvršavanje ne utiče na stanje podataka u bazi podataka.

Veličina oploga zavisi od resursa sistema, ali uglavnom bude dovoljna za normalno funkcionisanje sistema. Ako je veličina oploga dovoljna da pokrije dvodnevni period, to znači da će svaki sekundarni član koji ne replicira promene duže od dva dana biti u mogućnosti da se sinhronizuje kada ponovo postane aktivan. Međutim, ako vreme za koje se oplog popuni bude kraće od perioda neaktivnosti sekundarnog člana, on neće moći da se samostalno oporavi.

Veličina oploga se može ručno podesiti korišćenjem opcije **oplogSizeMB** pri samoj konfiguraciji seta replika, a moguće je i dinamički promeniti veličinu oploga bez restarta korišćenjem administrativne komande **replSetResizeOplog**.

4. Sinhronizacija replika u setu

Kako bismo bili sigurni da su sve kopije deljenog seta podataka ažurirane i sadrže up-to-date podatke, sekundarne replike se sinhronizuju tj. replikuju podatke sa ostalih članova seta. MongoDB baza podataka koristi dva načina sinhronizacije replika: **inicijalna sinhronizacija**, kojom se populišu novi članovi podacima i **replikacija**, kojom se primenjuju novonastale promene nad celim setom podataka.

4.1 Inicijalna sinhronizacija

Inicijalna sinhronizacija kopira sve podatke sa jednog člana replika seta na drugi. Postoji mogućnost izbora izvora sinhronizacije koristeći parametar **initialSyncSourceReadPreference** prilikom pokretanja **mongod**. Počev od verzije MongoDB 5.2, inicijalna sinhronizacija može se podeliti na **logičku** ili može biti bazirana na **kopiranju fajlova**.

4.1.1 Logička inicijalna sinhronizacija

Tokom logičke inicijalne sinhronizacije, MongoDB baza podataka:

- Klonira sve baze podataka, osim lokalne baze podataka (naime, svaka mongod instanca ima svoju lokalnu bazu podataka, koja je za samu replikaciju nevidljiva – kolekcije iz ove baze podataka se ne replikuju). Kako bi obavio kloniranje, mongod skenira svaku kolekciju u svakoj bazi podataka i sve podatke unosi u svoje kopije ovih baza podataka
- Kreira sve indekse za kolekcije tokom kopiranja dokumenata
- Povlači nove zapise iz *oplog*-a dok kopiranje podataka traje
- Primenjuje sve promene nad setom podataka, koristeći zapise iz *oplog*-a kako bi ažurirao svoje podatke u skladu sa trenutnim stanjem replika seta.

4.1.2 Sinhronizacija bazirana na kopiranju fajlova

Ovaj metod je dostupan samo u MongoDB Enterprise verziji, a sama sinhronizacija se odvija tako što se fajlovi direktno kopiraju i premeštaju na fajl sistemu, što može značajno ubrzati proces u poređenju sa logičkom inicijalnom sinhronizacijom. Da bi se ovakva sinhronizacija omogućila, potrebno je da se parametar **initialSyncMethod** postavi na vrednost **fileCopyBased**, na odredišnom članu seta replika koji će obaviti inicijalnu sinhronizaciju. Sinhronizacija se postiže tako što se lokalna baza podataka odredišnog člana zameni lokalnom bazom podataka izvornog člana.

Međutim, iako može dovesti do potencijalno boljih rešenja, ova sinhronizacija ima i određena ograničenja:

- Nije moguće kreirati backup na članu koji se sinhronizuje, niti na članu sa kog se vrši sinhronizacija, sve dok je inicijalna sinhronizacija u toku
- Tokom sinhronizacije nije moguće pisati u lokalnu bazu podataka koji se sinhronizuje.
- Inicijalna sinhronizacija se može vršiti samo od strane jednog člana replika seta u datom trenutku.

5. Visoka dostupnost seta replika

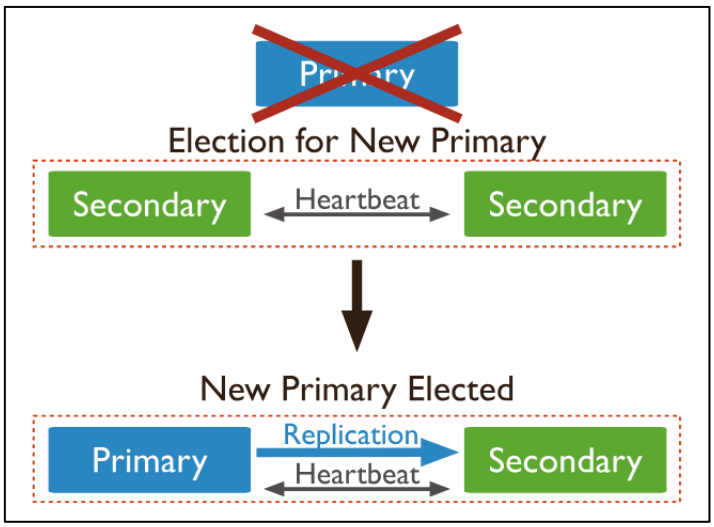
Setovi replika u MongoDB-ju koriste **sistem izbora** (eng. *elections*) kako bi postigli visoku dostupnost. Visoka dostupnost ukazuje na to da je sistem dizajniran da bude izdržljiv, redundantan i da ima automatski oporavak od grešaka. Aplikacije koje su podržane ovakvim sistemom mogu da nastave da funkcionišu, bez perioda downtime-a jako dug vremenski period.

5.1 Izbori u setovima replika

Setovi replika koriste izbore kako bi ustanovile koji član seta će postati novi primarni čvor. Izbori mogu biti izazvani i pokrenuti kao rezultat velikog broja događaja, poput:

- dodavanje novog čvora u replika setu
- inicijalizacija seta replika
- održavanje nad setom replika pomoću metoda **rs.stepDown()** ili **rs.reconfig()**
- nakon što sekundarni članovi izgubi vezu sa primarnim serverom i prekid traje duže od definisanog timeout perioda, koji podrazumevano traje 10 sekundi.

Na slici 9, dat je prikaz sistema replika u kome je primarni čvor bio nedostupan period duži od definisanog timeout perioda, te je pokrenut proces **automatskog oporavka** (eng. *automatic failover process*). Naime, u okviru ovog procesa, jedan od preostalih aktivnih sekundara proglašava početak izbora novog primarnog čvora i automatski nastavlja sa normalnim tokom operacija.



Slika 9. Ilustracija izbora novog primarnog čvora

Međutim, set replika ne može da nastavi sa obradom operaicja upisa sve dok proces izbora nije uspešno završen, ali može da nastavi da opslužuje operacije čitanja, ukoliko su upiti ovog tipa omogućeni da rade na sekundarnim serverima.

Prosečno vreme izbora novog primarnog člana unutar klastera tipično ne bi trebalo da potraje više od 12 sekundi, uzimajući kao primer podrazumevanu konfiguraciju replika. U ovo vreme uračunato je i vreme potrebno da se trenutni primarni čvor označi kao nedostupan, da se

otvore i uspešno završe izbori. Svakako, dužina trajanja celog ovog procesa može se i manuelno podesiti kroz konfiguraciju polja **electionTimeoutMillis**. Drugi faktori, poput mrežne latencije takođe mogu uticati i produžiti vreme potrebno da se izbori na nivou seta replika obave, čime se direktno utiče na vreme koje kluster može da funkcioniše bez primarnog čvora.

5.2 Rollback-ovi prilikom replika set failover-a

Rollback podrazumevano vraća na staro stanje operacije upisa koje su obavljene na prethodnom primarnom čvoru, kada se ovaj čvor ponovo pridruži setu replika nakon oporavka (eng. *failover*). Rollback je neophodan ukoliko dođe do situacije da je primarni čvor prihvatim operacije upisa koje sekundarni čvorovi još uvek nisu replicirali pre nego je došlo do otkaza primarnog čvora. Kada se stari primarni čvor ponovo aktivira u setu replika, on postaje sekundarni čvor i briše (eng. *rolls-back*) sve operacije upisa, kako bi se održala konzistentnost podataka kroz čitavu bazu podataka.

MongoDB pokušava da izbegne rollback-ove, koji svakako i treba da se dešavaju retko. Takođe, rollback se neće desiti ukoliko su operacije upisa koje su se obavile na primarnom čvoru replikovale na bilo kom drugom čvoru u setu replika pre otkaza primarnog i ukoliko je taj čvor ostao dostupan većini u setu replika.

6. Praktična implementacija MongoDB replikacije

U ovom poglavlju biće prikazan postupak kreiranja seta replika u MongoDB bazi podataka. Pre drugih konfiguracija, najpre je kreiran određen folder koji će funkcionisati kao centralizovani repozitorijum za MongoDB konfiguracione fajlove, fajlove podataka baze podataka i drugih neophotnih resursa. Set replika je konfigurisan tako da jedan čvor tj. jedna replika bude primarna, dok su ostale sekundarne.

Pre početka konfigurisanja seta replika, potrebno je postarati se da je MongoDB pravilno instaliran. Potrebno je instalirati MongoDB Community Server. Nakon što je instalacija završena, može se komandom **mongo --version** proveriti koja je trenutno aktivna MongoDB verzija.

```
C:\Users\Korisnik>mongod --version
db version v7.0.14
Build Info: {
    "version": "7.0.14",
    "gitVersion": "ce59cfc6a3c5e5c067dca0d30697edd68d4f5188",
    "modules": [],
    "allocator": "tcmalloc",
    "environment": {
        "distmod": "windows",
        "distarch": "x86_64",
        "target_arch": "x86_64"
    }
}
```

Slika 10. Aktivna MongoDB verzija

Najpre, kreiran je pomenuti folder u kome su kreirani podfolderi rs0-0, rs0-1 i rs0-2, kao i zaseban **logs** folder u kome će biti smešteni logovi. Potom, kao što je prikazano na slici 11, izvršena je mongod komanda za svakog člana seta replika, gde je naznačeno na kom portu će se koji izvršavati, povezani su sa localhostom i definisane su putanje do baze podataka i direktorijuma koji će čuvati logove.

```
C:\Users\Korisnik>mongod --replSet rs0 --port 27017 --dbpath C:\replication\rs0-0 --logpath C:\replication\logs\rs0-0.log --logappend

C:\Users\Korisnik>mongod --replSet rs0 --port 27018 --dbpath C:\replication\rs0-1 --logpath C:\replication\logs\rs0-1.log --logappend

C:\Users\Korisnik>mongod --replSet rs0 --port 27019 --dbpath C:\replication\rs0-2 --logpath C:\replication\logs\rs0-2.log --logappend
```

Slika 11. Komande za kreiranje replika u setu

Ovim je kreiran set replika pod nazivom rs0 koje se izvršavaju na portovima 27017, 27018 i 27019 i čiji se podaci baze podataka i odgovarajući log fajlovi čuvaju na adekvatnim lokacijama. Potom je u mongosh (MongoDB shell - interaktivni javascript interfejs za MongoDB) kreirana konfiguracija kao na slici 12.

```
> rsconf = {_id: "rs0", members:[{_id: 0, host: "localhost:27017"}, {_id: 1, host: "localhost:27018"}, {_id: 2, host: "localhost:27019"}]}
< {
    _id: 'rs0',
    members: [
      { _id: 0, host: 'localhost:27017' },
      { _id: 1, host: 'localhost:27018' },
      { _id: 2, host: 'localhost:27019' }
    ]
  }
```

Slika 12. Kreirana konfiguracija seta replika

Nakon poziva komande **initiate**, kojoj je kao parametar prosleđen prethodno kreiran konfiguracioni objekat, dobijamo potvrdu da je replika set uspešno kreiran, a komandom **status** možemo dobiti informacije o samom setu, ali i svakoj replici pojedinačno kroz **members** niz.

Slika 13. Potvrda konfiguracije seta replika

7. Zaključak

MongoDB je trenutno najpopularnija dokument-orijentisana baza podataka. Kao relativno novo rešenje koje se pojavilo na polju baza podataka, MongoDB je stekao popularnost prvenstveno među NoSQL bazama. Njegova otvorena struktura čini ga pogodnim za izgradnju visokoperformantnih skladišta podataka. Dodatno, jednostavna integracija i široka podrška omogućavaju brzo konfigurisanje baze podataka i njeno testiranje, što je čini privlačnom opcijom.

Koncept replikacije u MongoDB bazi podataka donosi brojne prednosti. Pre svega, omogućava visoku dostupnost podataka i povećava otpornost sistema na otkaze čvorova. Uz replikaciju, podaci se čuvaju na više replika unutar klastera, što obezbeđuje kontinuitet rada čak i u slučaju neplaniranih prekida. Takođe, replikacija doprinosi boljem balansiranju opterećenja i omogućava geografski distribuirane sisteme, što je ključno za globalno prisutne aplikacije.

Ipak, implementacija replikacije nosi sa sobom i određene izazove. Jedan od ključnih izazova je održavanje konzistentnosti podataka u realnom vremenu. Takođe, prilikom skaliranja replikacije na veliki broj čvorova, moguće je suočiti se sa povećanom latencijom i složenošću u upravljanju klasterom. Uprkos ovim izazovima, pravilna konfiguracija i optimizacija replikacije mogu značajno unaprediti performanse sistema i obezbediti stabilan rad u zahtevnim okruženjima.

Replikacija u MongoDB bazi podataka svakako predstavlja ključni mehanizam za obezbeđivanje otpornosti i performansi u modernim aplikacijama, uz adekvatno upravljanje mogućim izazovima.

8. Literatura

[1] “Designing Data Intensive Applications”, Martin Kleppmann, O’Reilly, 2017.

[2] “MongoDB – The Definitive Guide”, Kristina Chodorow, O’Reilly, 2013.

[3] https://www.mongodb.com/docs/manual/replication/

[4] https://www.mongodb.com/resources/products/capabilities/replication

[5] “Implementing replica set: Strategy to Improve the Performance of NoSQL Database Cluster in MongoDB”, Wisnu Uriawan, Ridwan Ahmad Fauzan, Reski Firmansyah, 2024.

[5] “Fault-Tolerant Replication with Pull-Based Consensus in MongoDB”, Siyuan Zhou, Shuai Mu, April 2021.

[6] “A review on various aspects of MongoDB databases”, Anjali Chauhan, 2019.

[7] „Survey on MongoDB: An open-source document database“, Smita Agrawal, Jai Prakash Verma, Brijesh Mahidhariya, Nimesh Patel, 2015.

[8] “Practical MongoDB - Architecting, Developing and Administering MongoDB”, Shakuntala Gupta Edward, Navin Sabharwal

[9] https://www.fivetran.com/learn/mongodb-database-replication