UNIVERZITET U NIŠU
ELEKTRONSKI FAKULTET

Replikacija baza podataka kao mehanizam oporavka baza podataka kod MongoDB

Seminarski rad
Studijski program: Računarstvo i informatika
Modul: Softversko inženjerstvo

Student:

Đorđe Petković, br. ind. 1614

Mentor:

prof. Aleksandar Stanimirović

Niš, Maj 2024. godine

`

Sadržaj

`

Uvod

Replikacija baza podataka je proces kopiranja i održavanja podataka iz jedne baze podataka na više servera ili lokacija. Ovaj mehanizam omogućava da podaci budu dostupni na više mesta istovremeno, što može pomoći u slučaju otkazivanja nekog od članova na kome se nalazi baza podataka.

Bitan faktor kod replikacije baza podataka je taj da je replikacija baza podataka stalan proces. Ukoliko korisnik pristupi ili izmeni podatke u izvornoj bazi podataka, te promene se sinhronizuju sa replikovanim bazama podataka. Ovo osigurava da korisnici uvek pristupaju najnovijim i najtačnijim podacima.

Jedna od podela replikacije je na dva tipa replikacije baza podataka:

Aktivna replikacija: Svaki sistem može obrađivati promene i baze podataka se sinhronizuju u svim pravcima. Obično je potrebna dobro planiranje i arhitektura baza podataka da bi se primenjivala aktivna replikacija kako bi se izbegli konflikti, a ova postavka se uglavnom koristi za balansiranje opterećenja ili za pružanje visoke dostupnosti. Ukoliko sistem raste, potrebno je distribuirati bazu podataka, radi bržeg pristupa i odziva.

Replikacija samo za čitanje (Read-only): Replikovane baze podataka primaju promene samo od primarne baze podataka. Korisnici mogu čitati podatke iz replikovanih baza podataka, ali ih ne mogu menjati. Baze podataka samo za čitanje se koriste za visoku dostupnost podataka, omogućavajući korisnicima pristup podacima bez mogućnosti da ih menjaju.

Neke od prednosti koje se dobijaju primenom replikacije baze podataka su:

Povećana dostupnost	Replikacija osigurava da baza podataka ostane dostupna čak i ako jedan ili više servera zakažu. U slučaju kvara primarnog servera, sekundarni serveri mogu preuzeti odgovornost za obradu zahteva, čime se minimiziraju prekidi u radu.
Poboljšane performanse	Distribucijom opterećenja na više servera, replikacija može značajno poboljšati performanse sistema. Korisnici mogu pristupati najbližem serveru, čime se smanjuje latentnost i ubrzava pristup podacima.
Oporavak podataka	U slučaju gubitka podataka ili korupcije na jednom serveru, replikacija omogućava obnavljanje podataka sa sekundarnih

`

	servera. Ovo čini sistem otpornijim na greške i gubitke, osiguravajući tačnost naših podataka.
Skalabilnost	Replikacija omogućava horizontalno skaliranje, gde se dodavanjem novih servera može lako povećati kapacitet sistema za obradu većeg broja zahteva.

Replikacija je naročito važna u modernim sistemima koji zahtevaju visoku dostupnost i pouzdanost, kao što su platforme za onlajn kupovinu, društvene mreže i razni sistemi za upravljanje poslovanjem. Kroz replikaciju, ovi sistemi mogu obezbediti neprekidan rad i visok nivo usluge svojim korisnicima, čak i u slučaju neočekivanih kvarova ili otkaza.

`

Replikacija baza podataka

Jedna od definicija replikacije baza podataka:

Replikacija baze podataka je proces kreiranja više kopija jedne baze podataka na različitim lokacijama ili članovima. Podaci iz jedne baze podataka (primarne baze podataka) se kopiraju i skladište u jednoj ili više drugih baza podataka (replikama baza podataka). Ova postavka rezultira višestrukim kopijama identičnih podataka u različitim bazama podataka. Ako jedna replika zakaže, dostupne su druge replike kako bi operacije mogle nesmetano da se nastave.

Neki od ključnih pojmova koje moramo znati, da bi uspešno radili sa replikacijom baza podataka su:

Primarna baza	Ovo je glavna baza podataka iz koje se podaci kopiraju. U primarnoj bazi se vrše sve promene i transakcije koje se zatim propagiraju na sekundarne baze. Kod objašnjenog **Read-only** pristupa replikaciji, nad ovom bazom se vrše svi write upiti, koji se kasnije propagiraju na ostale (sekundarne) baze podataka.
Sekundarne baze	Ove baze podataka primaju kopije podataka iz primarne baze. Sekundarne baze služe kao rezervne kopije podataka i mogu biti korišćene za čitanje podataka kako bi se rasteretila primarna baza. U slučaju **Aktivne replikacije**, nad ovim bazama takođe mogu biti primenjivani ne samo Read upiti.
Sinhronizacija	Ovo je proces kojim se osigurava da su podaci u sekundarnim bazama ažurni i identični podacima u primarnoj bazi. Sinhronizacija može biti sinhrona ili asinhrona, zavisno od zahteva sistema za konzistentnošću i performansama. Ovim putem osiguravamo konzistentnost podataka u svim našim bazama podataka.

`

Podela replikacije baza podataka

U ovom poglavlju biće navedeni neki od tipova replikacije podataka.

Podela replikacije baza podataka na osnovu načina identifikacije promena

Prva podela je na osnovu načina identifikacije promena [3]. Ovo su 3 glavna tipa replikacije baza podataka:

Full-table replikacija

Full-table replikacija kopira svaki podatak unutar tabele iz baze podataka na odredište (uglavnom na cloud-u). Ovo uključuje nove, postojeće i ažurirane podatke.

Prednosti	S obzirom na to da replicira celu tabelu, uvek će biti tačan skup podataka nakon svake sinhronizacije i možemo biti sigurni da su svi inserti, ažuriranja i brisanja zabeleženi.
Nedostaci	Ovo je najmanje efikasan tip replikacije baze podataka i prilično je resursno intenzivan jer se kopira svaki podatak unutar tabele bez obzira da li je promenjen ili ne. Takođe, ovo može dovesti do naglog opterećenja primarne baze podataka u zavisnosti od veličine i količine podataka unutar tabela.

Replikacija po ključu

Replikacija po ključu je metoda replikacije baze podataka koja koristi ključ za replikaciju kako bi identifikovala nove i ažurirane zapise na osnovu vremenskog podatka (timestamp-a) ili ključa u obliku celog broja.

Prednosti	Replikacija po ključu je efikasan tip replikacije baze podataka jer replicira samo ažurirane i umetnute redove, koristeći tako manje resursa.
Nedostaci	Bilo koji podaci koji su trajno obrisani iz primarne baze podataka neće biti automatski replicirani u sekundarnim bez dodatnog vremena i truda uloženog u procese koji mogu

`

	identifikovati brisanja. To može da dovede do neželjenih podataka na nekoj od sekundarnih baza podataka.

Replikacija zasnovana na logovima

Replikacija zasnovana na logovima kopira promene na osnovu binarnih log datoteka baze podataka. Ona se obično naziva **oplog**, i to je datoteka koja beleži šablone, aktivnosti i operacije unutar baze podataka.

Prednosti	Ovaj tip replikacije baze podataka je najefikasniji jer čita direktno iz binarnih log datoteka i ne takmiči se sa drugim upitima baze podataka.
Nedostaci	Replikacija zasnovana na logovima je dostupna samo za određene baze podataka. U nekim slučajevima, pristup logovima baze podataka nije moguć, na primer ukoliko je hostovana od strane third-party host-a. Takođe, postavljanje replikacije zasnovane na logovima može biti veoma vremenski intenzivno, komplikovano i sklono greškama. Ukoliko nije pokriven neki slučaj, može doći do netačne/neuspešne replikacije.

Podela replikacije baza podataka na osnovu vremena izvršenja

Druga podela tehnika za replikaciju baza podataka na osnovu vremena posle kog se dešava sinhronizacija, i načina na koji se ta sinhronizacija izvršava.

Sinhrona replikacija

Sinhrona replikacija je tip replikacije baze podataka gde dve ili više baza podataka uvek imaju iste informacije. Pri promeni na glavnoj (primarnoj) bazi podataka, promene se odmah odražavaju i na sekundarne (replike). Time se postiže konzistentnost i tačnost podataka u svakom trenutku.

Sinhrona replikacija je korisna za aplikacije koje trebaju brz pristup podacima, koji su u svakom trenutku ažurirani i tačni. Neke od aplikacija koje bi trebalo da koriste sinhronu replikaciju su sistemi za onlajn bankarstvo, društvene mreže, online prodavnice..

`

Asinhrona replikacija

Asinhrona replikacija je tip replikacije baze podataka gde se kopije istih podataka čuvaju na različitim lokacijama, koje možda neće biti potpuno identične u svakom trenutku. To je posledica činjenice da, ukoliko se nešto promeni na jednoj bazi podataka (pr. write upis), može proći određeno vreme dok se te promene ne propagiraju, i vide na svim ostalim replikama.

Asinhrona replikacija je dobra za aplikacije koje ne zavise u potpunosti od podataka koje se nalaze u bazi, kao što su analitičke i aplikacije koje se koriste za kreiranje ili pružanje izveštaja. Takođe omogućava veću fleksibilnost jer se podaci mogu ažurirati na jednoj lokaciji bez potrebe da se promene odmah propagiraju na sve replike.

Snapshot replikacija

Snapshot replikacija podrazumeva pravljenje „slike“ (Snapshot-a) podataka u određenom vremenskom trenutku i repliciranje tih podataka u ostale replike.

Snapshot replikacija je korisna za baze podataka gde su promene podataka retke, i prihvatljivo je da podaci ne budu ažurirani kontinualno. Prednosti snapshot replikacije su što je brza i laka za postavljanje. Može se koristiti za aplikacije kao što su analitika ili izveštavanje, ili za druge slučajeve gde su periodična ažuriranja dovoljna. Ukoliko neka prodavnica, koja čuva svoj katalog u bazi podataka, želi da replicira svoje proizvode, Snapshot replikacija može biti veoma dobra. Ukoliko se katalog proizvoda ažurira jednom nedeljno, ili mesečno, može se iskoristiti Snapshot replikacija, zbog svoje jednostavnosti implementacije i održavanja.

Merge replikacija

Merge replikacija je metod replikacije baze podataka koji omogućava da dve ili više baza podataka nezavisno sakupljaju promene i zatim ih spajaju. U ovom tipu replikacije, nad svakom bazom podataka se mogu nezavisno vršiti izmene, a zatim, u procesu sinhronizacije, te promene se spajaju i kombinuju.

`

Korisna je u okruženjima sa više korisnika gde korisnici trebaju da rade sa svojim lokalnim kopijama baze podataka i zatim sinhronizuju promene sa centralnim serverom. Primeri su kada postoji više timova u okviru jednog projekta ili jedne kompanije, koji ažuriraju svoje lokalne baze podataka dok su van kancelarije i zatim sinhronizuju podatke kada se vrate.

Replikacija baze podataka u realnom vremenu

Replikacija baze podataka u realnom vremenu (Real-time Database Replication) odnosi se na proces kontinuiranog kopiranja podataka iz jedne baze podataka u drugu skoro u realnom vremenu. Promene napravljene na izvornoj bazi podataka brzo se propagiraju do replike, ali proces ne zahteva nužno da replike budu savršeno sinhronizovane u trenutku potvrde transakcije. Koristi se za aplikacije kojima je potreban brz pristup podacima, kao što su sistemi za online bankarstvo i društvene mreže.

Replikacija baze podataka u realnom vremenu se često koristi za svrhe oporavka od katastrofe. Ako primarna baza podataka zakaže ili postane nedostupna, replika baza podataka može preuzeti sa minimalnim zastojem, osiguravajući visoku dostupnost.

Za razliku od sinhrone replikacije, replikacija u realnom vremenu ne čeka potvrde svih replika da je promena uspešno izvršena na njima, što čini replikaciju u realnom vremenu bržom nego sinhrona replikacija. Takođe, replikacija u realnom vremenu ima mnogo lakšu i bolju primenu nad distribuiranim aplikacijama.

Sa druge strane, ukoliko dođe do otkaza, u vremenskom intervalu između propagiranja promena i izvršenja promena na svim replikama, može doći do nekonstantnosti u podacima između baza podataka. Ukoliko je izbor između sinhrone replikacije i replikacije u realnom vremenu, sinhrona se bira ukoliko je konzistentnost podataka ključna, dok se replikacija u realnom vremenu bira ukoliko i pri postojanju malih nekonzistentnosti u slučaju otkaza aplikacija može da nastavi da funkcioniše.

`

MongoDB u kontekstu replikacije

MongoDB je program za upravljanje bazama podataka otvorenog koda koji se koristi za NoSQL (ne samo SQL) pristup, kao alternativa tradicionalnim relacionim bazama podataka. NoSQL baze podataka su veoma korisne za rad sa velikim skupovima distribuiranih podataka. MongoDB je alat koji može upravljati informacijama koje se skladište kao dokumenti, čuvati ili pribaviti potrebne informacije [1].

MongoDB je dobar kandidat za ilustraciju i primenu replikacije baza podataka, zbog svoje šireoke rasporstranjenosti. MongoDB je čest izbor pri kreiranju raznih aplikacije, koje mogu biti veoma velike i robusne. Takođe, zbog svoje fleksibilnosti i pouzdanosti, omogućava aplikacijama visoku dostupnost i adekvatnu replikaciju, što ga čini idealnim za situacije gde je potrebna visoka tačnost i dostupnost podataka.

Takođe, zbog svoje popularnosti, MongoDB ima veliku i aktivnu zajednicu korisnika, kao i komercijalnu podršku kroz MongoDB Inc. Zbog toga postoji veliki broj resursa i alata, dobra dokumentacija, što čini proces implementacije i održavanja replikacija lakšim.

U kontekstu replikacije, MongoDB nudi dosta prednosti. Neke od njih su:

Fleksibilnost i skalabilnost	MongoDB je dizajniran kao NoSQL baza podataka koja može lako skalirati horizontalno. To znači da se mogu dodavati novi serveri (članovi) kako bi se povećao kapacitet i performanse sistema. Replikacija u MongoDB omogućava lako skaliranje i raspodelu podataka preko više članova, čime se obezbeđuje bolja dostupnost i otpornost.
Jednostavna konfiguracija replike seta	MongoDB koristi koncept replike seta, što je skup MongoDB instanci koje održavaju iste podatke. Postavljanje i održavanje replike seta je relativno jednostavno, što omogućava brzu implementaciju i lako upravljanje replikacijom.
Automatski failover	Jedna od ključnih prednosti MongoDB replike seta je automatski failover. U slučaju da primarni član (server) zakaže, jedan od sekundarnih članova automatski preuzima ulogu primarnog bez potrebe za manuelnom intervencijom. Ovo obezbeđuje kontinuitet rada i minimalizuje prekide u dostupnosti podataka.

`

Podrška za distribuirane sisteme	MongoDB je odličan za distribuirane aplikacije koje zahtevaju visoku dostupnost i otpornost na greške. Replikacija omogućava distribuciju podataka preko različitih geografskih lokacija, čime se smanjuje latencija i poboljšava iskustvo korisnika.

Ove prednosti čine MongoDB idealnim kandidatom za ilustraciju kako replikacija baza podataka može efikasno funkcionisati i kako doprinosi visokoj dostupnosti, otpornosti i skalabilnosti sistema, što će u nastavku biti prikazano.

Replikacija kod MongoDB baze podataka

Skup replika u MongoDB-u je grupa **mongod** procesa koji održavaju isti skup podataka. Skupovi replika pružaju redundantnost i visoku dostupnost, i osnova su za sve korisničke (produkcijske) implementacije. U ovom poglavlju biće objašnjena replikacija u MongoDB-u, kao i komponente i arhitektura skupova replika.

Redundantnost i Dostupnost Podataka

Replikacija pruža redundantnost i povećava dostupnost podataka. Sa više kopija podataka na različitim serverskim bazama podataka, replikacija pruža nivo tolerancije na otkaz u slučaju gubitka jednog serverskog baziranog sistema.

U nekim slučajevima, replikacija može pružiti povećani kapacitet čitanja, pošto klijenti mogu slati čitanje operacija na različite servere. Održavanje kopija podataka u različitim centrima može povećati lokalnost i dostupnost podataka za distribuirane aplikacije. Takođe mogu se održavati dodatne kopije za posebne svrhe, kao što su oporavak od katastrofe, izveštavanje ili rezervna kopija.

Skup replika sadrži nekoliko članova koji sadrže podatke i opciono jedan član arbitera (sudije). Od članova koji sadrže podatke, jedan član se smatra primarnim članom, dok se ostali članovi smatraju sekundarnim članovima.

`

Članovi seta replika

Članovi seta replika mogu biti:

- Primarni
- Sekundarni
- Arbitreri

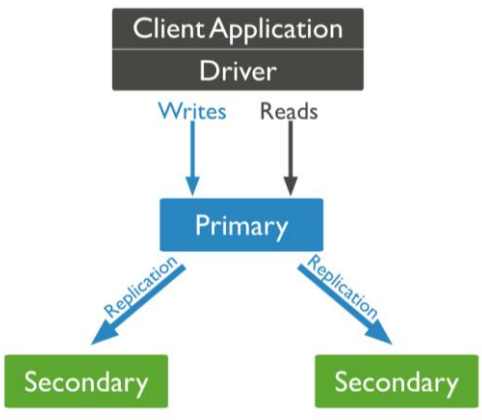
Ukoliko se implementira replikacija, minimalna konfiguracija za skup replika je skup sa tri člana koji sadrže podatke, odnosno jedan primarni i dva sekundarna člana. U nekim slučajevima, ukoliko dodavanje drugog sekundarnog člana nije moguće, može se dodatiž arbiter. Arbiter učestvuje u izborima (glasanju), ali ne čuva podatke (tj. ne pruža redundanciju podataka). Ograničenje MongoDB-a je 50 člana, ali maksimalno 7 mogu da učestvuju u glasanju.

Set replika može da sadrži jedan primarni član. Ukoliko dođe do otkaza tog člana, glasa se za postavljanje novog primarnog člana, o čemu će biti reči u nastavku rada.

Primarni član skupa replika

Primarni član je jedini član u skupu replika koji prima operacije upisa (write operacije) [4]. MongoDB primenjuje operacije upisa nad primarnim članom, a zatim beleži operacije u **oplog** (operacione logove) primarnog člana. Sekundarni članovi repliciraju ovaj **oplog** i primenjuju operacije nad svojim skupovima podataka.

U skupu replika sa tri člana, primarni član prihvata sve operacije upisa. Zatim se oplog replikuje na sekundarne članove kako bi se primenio na njihove skupove podataka.



`

Svi članovi skupa replika mogu prihvatiti operacije čitanja. Međutim, po default-u, aplikacija usmerava svoje operacije čitanja na primarni član (read preference mode = **primary**) [4]. To ponašanje se može promeniti ukoliko se odabere neka od drugih prefernci:

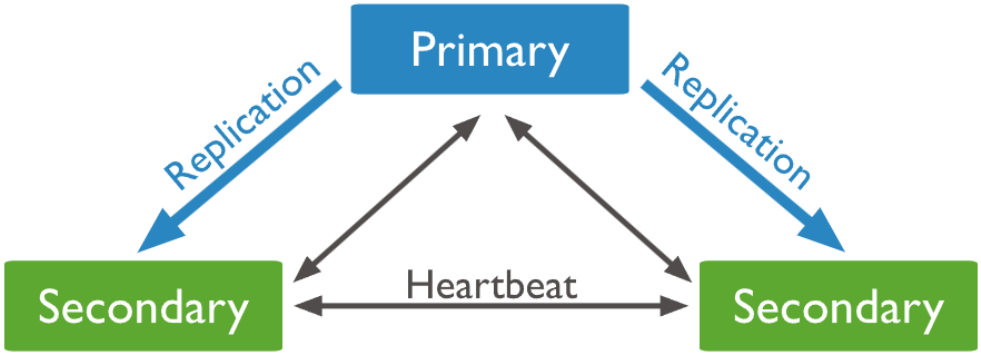
primary	Podrazumevani mod. Sve read operacije se šalju primarnom članu. Distribuirane transakcije koje sadrže read operacije moraju da koriste ovaj mod. Sve operacije u transakcijama moraju da se usmere na isti član.
primaryPreferred	Ukoliko je primarni član dostupan, read operacija će se izvršiti nad njim. Ukoliko nije, izvršiće se nad sekundarnim.
secondary	Sve read operacije primenjuju se nad sekundarnim članovima (replikama).
secondaryPreffered	Ukoliko je sekundarni član dostupan, read operacija će se izvršiti nad njim. Ukoliko nije, izvršiće se nad primarnim. Ukoliko postoji samo primarni član, bez sekundarnih, izvršiće se nad njim.
nearest	Operacija read se izvršava nad slučajnim članom, nevezano da li je primarni ili sekundarni, već se on bira na osnovnu praga latencije, koji se računa na osnovu prosleđenih opcija (vreme latencije za odabir instance – localThresholdMS, maksimalni replikacioni zaostatak – maxStalnessSeconds ..)

Sekundarni članovi

Sekundarni član održava kopiju primarnog skupa podataka. Za replikaciju podataka, sekundarni član primenjuje operacije iz **oploga** primarnog člana na svoj skup podataka u asinhronom procesu (MongoDB koristi asinhroni tip replikacije) [5]. Skup replika može imati jedan ili više sekundarnih članova.

Na ovom primeru, sa 2 sekundarna i jednim primarnim članom, vidimo da sekundarni članovi repliciraju podatke sa primarnog. Heartbeat (ping) se izvršava na svake 2 sekunde, i on govori članovima seta da li je neki član i dalje aktivan. Ukoliko se ne odazove na 5 heartbeat-a, članovi smatraju taj član kao nedostupan.

`



Iako se podaci ne mogu pisati direktno na sekundarne članove, mogu se čitati podatci iz sekundarnih članova, u zavisnosti od preferenci objašnjenim u prethodnom poglavlju.

Sekundarni član može postati primarni. Ako trenutni primarni član postane nedostupan, skup replika održava izbore (elections) kako bi odabrao koji od sekundarnih članova postaje novi primarni.

U zavisnosti od konfiguracije, možemo imati tri tipa sekundarnih članova [5]. To su:

- Članovi sa prioritetom 0
- Skriveni članovi
- Odloženi (delayed) članovi

Članovi sa prioritetom 0

Član sa prioritetom 0 je član skupa replika koji ne može postati primarni i ne može pokrenuti izbore (election). Članovi sa prioritetom 0 mogu potvrditi operacije upisa izdane sa postavkom **write concern** od **w: <broj>.** Za **"majority"** write concern, član sa prioritetom 0 mora biti glasački član kako bi mogao potvrditi upis (members[n].votes je veći od 0).

Osim gore pomenutih ograničenja, sekundarni članovi sa prioritetom 0 funkcionišu kao normalni sekundarni članovi. Oni održavaju kopiju skupa podataka, prihvataju operacije čitanja i glasaju na izborima.

Konfigurisanje člana skupa replika sa prioritetom 0 može biti poželjno ako je taj član implementiran u data centru koji je udaljen od glavne implementacije i stoga ima veću latenciju. Takav član može dobro služiti za lokalne zahteve za čitanje, ali nije idealan kandidat za obavljanje dužnosti primarnog člana zbog svoje latencije. Ukoliko bi takav član bio izabran za primarni čvor, rad samog sistema bi bio dosta sporiji.

`

Skriveni članovi

Skriveni članovi održavaju kopiju skupa podataka primarnog čvora, ali su nevidljivi za klijentske aplikacije. Klijenti neće distribuirati operacije čitanja sa preferencijom čitanja iz sekundarnih članova ka skrivenim članovima. Kao rezultat toga, ovi članovi ne primaju nikakve zahteve niti operacije, osim osnovne replikacije. Skriveni članovi koriste se za specijalizovane zadatke kao što su izveštavanje i pravljenje rezervnih kopija. Skriveni članovi moraju uvek biti članovi sa prioritetom 0 i stoga ne mogu postati primarni čvor. Skriveni članovi, takođe, mogu glasati na izborima.

Skriveni članovi skupa replika mogu potvrditi operacije upisa izdane sa postavkom **w: <broj>**. Međutim, za operacije upisa izdane sa write concern w: "majority", skriveni članovi moraju takođe glasati kako bi mogli potvrditi upis sa "majority" write concern.

Odloženi (delayed) članovi

Odloženi članovi sadrže kopije skupa podataka skupa replika. Međutim, skup podataka odloženog člana odražava ranije, ili odloženo, stanje skupa. Na primer, ukoliko član ima kašnjenje od 2 sata, na njemu neće biti primenjena nijedna operacija izvršena u proteklih 2 sata.

Zbog toga što su odloženi članovi „rezerva“ ili tekući „istorijski“ snapshot skupa podataka, oni mogu pomoći u oporavku od različitih vrsta grešaka. Na primer, odloženi član može omogućiti oporavak od neuspešnih migracija i grešaka operatera, uključujući obrisane baze podataka i kolekcije.

Odloženi članovi takođe mogu da učestvuju u glasanju, i u obavezi su da učestvuju ukoliko je prosleđen write concern „majority“, a njihov members[n].votes je veći od 0. Česta je praksa postavljanja tog property-ja na 0, da bi rad odloženih članova bio efikasniji.

Arbiteri

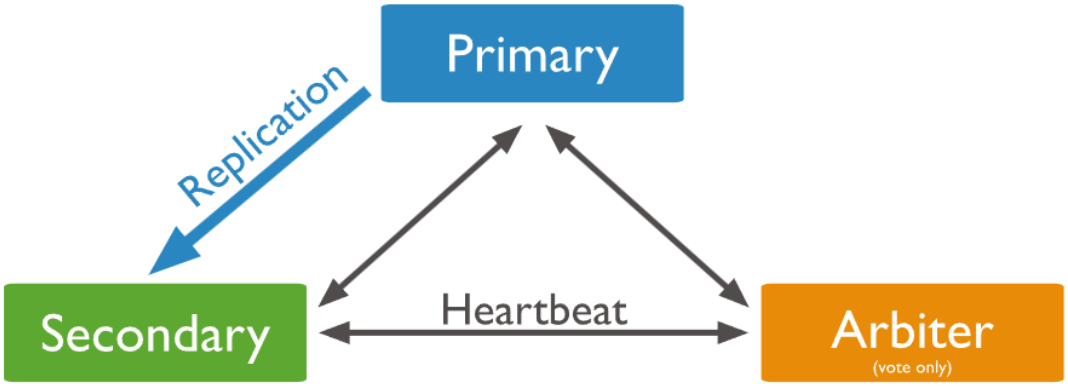
U nekim slučajevima, ukoliko je moguće postojanje samo jednog primarnog i sekundarnog člana, može se dodati arbiter u skup replika. Arbiter učestvuje u izborima za primarni čvor, ali nema kopiju skupa podataka i ne može postati primarni čvor [6].

Arbiter ima tačno 1 glas u izborima. Po defaultu, arbiter ima prioritet 0.

Uglavnom je preporučeno korišćenje jednog arbitera kako bi se izbegli problemi sa konzistentnošću podataka. Više arbitera sprečava pouzdano korišćenje "majority" write concern.

`

Da bi se osiguralo da će upis opstati nakon pada primarnog čvora, "majority" write concern zahteva da većina čvorova potvrdi operaciju upisa. Arbitri ne čuvaju nikakve podatke, ali doprinose ukupnom broju čvorova u skupu replika. Kada skup replika ima više arbitera, manja je verovatnoća da će većina čvorova koji nose podatke biti dostupna nakon pada čvora.



Oplog

Log operacija (oplog) je specijalna ograničena kolekcija koja sadrži zapise svih operacija koje menjaju podatke koje se čuvaju u bazi podataka. Ukoliko write operacije ne utiču na podatke u bazi podataka, one se ne čuvaju u oplog-u.

Za razliku od ostalih kolekcija ograničenih veličinom, oplog može da naraste iznad specificiranog limita veličine, da bi se izbeglo brisanje nekih nereplikovanih operacija. MongoDB primenjuje operacije nad primarnim članom, i upisuje ih u oplog primarnog člana. Sekundarni članovi iz oploga čitaju operacije, i primenjuju ih u procesu asinhrone replikacije. Svi sekundarni članovi moraju imati kopiju oplog-a, u **local.oplog.rs** kolekciji, koji im omogućava sinhronizaciju sa primarnim članom [7].

Svi članovi šalju heartbeat-ove (pingove) međusobno. Bilo koji sekundarni član može preuzeti oplog od bilo kog drugog člana.

Svaka operacija u oplog-u je idempotentna. Oplog operacije daju iste rezultate, ma koliko puta se primenjuju na bazu podataka.

`

Veličina oplog-a

Veličina oploga pri kreiranju seta replika zavisi od konfiguracije. Ukoliko nije prosleđena veličina, MongoDB kreira oplog default-ne veličine.

Za Unix i Windows sisteme, veličina oplog-a je po default-u 5% od slobodne memorije na disku ili fizičke memorije. Takođe, minimalna veličina oplog-a je 990MB. Ukoliko je na disku ili u fizičkoj memoriji, biće napravljen oplog veličine 990MB. Ukoliko je 5% slobodne memorije veće od 50GB, biće napravljen oplog veličine 50GB. Kod MacOS uređaja, minimalna veličina oploga je 192MB.

Uglavnom, default veličina oploga je dovoljna za normalno funkcionisanje sistema. Ukoliko je za popunjavanje oplog-a potrebno 2 dana, to znači da ukoliko neki sekundarni član otkaže ili prestane da replicira operacije iz oploga na manje od 2 dana, moći će da se oporavi pri povratku u sistem. Ukoliko neki član otkaže na vremenski interval duži od vremena za koje se oplog popuni, taj član se neće moći oporaviti samostalno [7].

Željena veličina oplog-a se može proslediti pomoću opcije **oplogSizeMB** pri kreiranju seta replika. Ukoliko je potrebno resize-ovati oplog, može se koristiti opcija **replSetResizeOplog** administrativna komanda, koja će dinamički resize-ovati oplog bez restartovanja **mongod** procesa.

Minimalni period očuvanja oplog-a

Može se specifirati minimalni broj sati tokom kojih će sve operacije u oplog-u biti sačuvane. U tom slučaju, operacije će biti izbrisane iz oplog-a samo ukoliko je oplog popunjen do maksimalne konfigurisane veličine, i ukoliko je ta operacija starija od minimalnog broja sati očuvanja oplog-a.

MongoDB po default-u ne postavlja ovu opciju, već automatski briše operacije iz oplog-a počevši od najstarijeg, tako da održi veličinu oplog-a ispod konfigurisane maksimalne veličine.

Da bi se postavila minimalni period očuvanja pri startovanju mongod, može se:

- Dodati storage.**oplogMiniRetentionHours** podešavanje u mongod konfiguarcioni fajl

`

- Dodati –**oplogMinRetentionHours** opcija komandne linije

Ukoliko je potrebno promeniti minimalni period očuvanja oplog-a, može se iskoristiti opcija **replSetResizeOplog** komanda. Ukoliko se u njoj specifira minimalni period očuvanja oplog-a, override-ovaće se sve vrednosti, koje su postavljenje pri pokretanju mongod procesa.

Replikacija i sinhronizacija skupa replika

Da bi se održale ažurirane kopije deljenog skupa podataka, sekundarni članovi skupa replika sinhronizuju ili repliciraju podatke od drugih članova. MongoDB koristi dva oblika sinhronizacije podataka: početnu sinhronizaciju za popunjavanje novih članova kompletnim skupom podataka i replikaciju za primenu tekućih promena na ceo skup podataka [8].

Početna sinhronizacija

Početna sinhronizacija kopira sve podatke sa jednog člana skupa replika na drugog člana. Mogu se navesti preferirani izvor početne sinhronizacije pomoću parametra **initialSyncSourceReadPreference**. Ovaj parametar se može navesti samo prilikom pokretanja **mongod**.

Proces Logičke Početne Sinhronizacije

Kada se obavlja logička početna sinhronizaciju, MongoDB:

- Klonira sve baze podataka osim lokalne baze podataka. Za kloniranje, mongod skenira svaku kolekciju u svakoj izvornoj bazi podataka i unosi sve podatke u svoje kopije tih kolekcija.
- Gradi sve indekse kolekcije dok se dokumenti kopiraju za svaku kolekciju.
- Preuzima nove zapise oploga tokom kopiranja podataka. Potrebno je da svi članovi imaju dovoljno prostora na disku u lokalnoj bazi podataka za privremeno skladištenje ovih oplog zapisa tokom trajanja ove faze kopiranja podataka.
- Primenjuje sve promene na skup podataka. Koristeći oplog sa izvora, mongod ažurira svoj skup podataka da odražava trenutno stanje skupa replika.

Kada se početna sinhronizacija završi, član prelazi iz stanja STARTUP2 (stanje koje dobija pri pokretanju mongod procesa) u SECONDARY.

`

Otpornost na Greške

Ako sekundarni član koji obavlja početnu sinhronizaciju naiđe na trajnu mrežnu grešku tokom procesa sinhronizacije, sekundarni član ponovo započinje proces početne sinhronizacije od početka.

Sekundarni član koji obavlja početnu sinhronizaciju može pokušati da nastavi proces sinhronizacije ako ga prekine privremena mrežna greška, brisanje kolekcije ili preimenovanje kolekcije.

Podrazumevano, sekundarni član pokušava da nastavi početnu sinhronizaciju 24 sata. Može se koristiti parametar **initialSyncTransientErrorRetryPeriodSeconds** za kontrolu vremena tokom kojeg sekundarni član pokušava da nastavi početnu sinhronizaciju. Ako sekundarni član ne može uspešno nastaviti proces početne sinhronizacije tokom podešenog vremenskog perioda, on bira novi zdravi izvor iz skupa replika i ponovo započinje proces početne sinhronizacije od početka.

Sekundarni član pokušava da ponovo započne početnu sinhronizaciju do 10 puta pre nego što vrati fatalnu grešku.

Izbor Izvora za Početnu Sinhronizaciju

Izbor izvora za početnu sinhronizaciju zavisi od vrednosti parametra **initialSyncSourceReadPreference** prilikom pokretanja mongod:

Za **initialSyncSourceReadPreference** postavljen na primary (default), bira se primarni čvor kao izvor sinhronizacije. Ako primarni čvor nije dostupan ili nedostižan, beleži se greška i periodično se proverava dostupnost primarnog čvora.

Za **initialSyncSourceReadPreference** postavljen na primaryPreferred pokušava se izbor primarnog čvora kao izvora sinhronizacije. Ako primarni čvor nije dostupan ili nedostižan, vrši se izbor izvora sinhronizacije iz preostalih članova skupa replika.

Sekundarna replikacija

Sekundarni članovi neprekidno repliciraju podatke nakon početne sinhronizacije. Sekundarni članovi kopiraju oplog iz svog izvora sinhronizacije i primenjuju ove operacije u asinhronom procesu.

`

Sekundarni članovi mogu automatski promeniti svoj izvor sinhronizacije po potrebi na osnovu promena u vremenu latencije i stanju replikacije drugih članova.

Streaming (Strujna) replikacija

Izvori sinhronizacije šalju kontinuirani tok oplog unosa svojim sekundarnim članovima. Streaming replikacija umanjuje latentnost za čitanje iz sekundarnih članova u uslovima visokog opterećenja i visoke latencije mreže. Pored toga, smanjuje zastarelost za čitanja iz sekundarnih članova, rizik od gubitka operacija pisanja, smanjuje latenciju operacija pisanja sa **w: "majority" i w: >1** (tj. bilo koji nivo write concern za pisanje koji zahteva čekanje na replikaciju).

`

Mehanizmi oporavka uz pomoć replikacije

MongoDB koristi nekoliko mehanizama oporavka i replikacije kako bi osigurao visoku dostupnost i otpornost na greške.Ovi mehanizmi obezbeđuju da baze podatka, i pri otkazu nekod od čvorova, nastavi da radi i pruža odgovore na zahteve i operacije.

Na osnovu konfiguracije i postavki replika setova, objašnjenih u radu, MongoDB obezbeđuje nekoliko mehanizama oporavka. Neki od njih su:

- Automatski failover (izbor novog primarnog čvora)
- Rollback tokom failovera

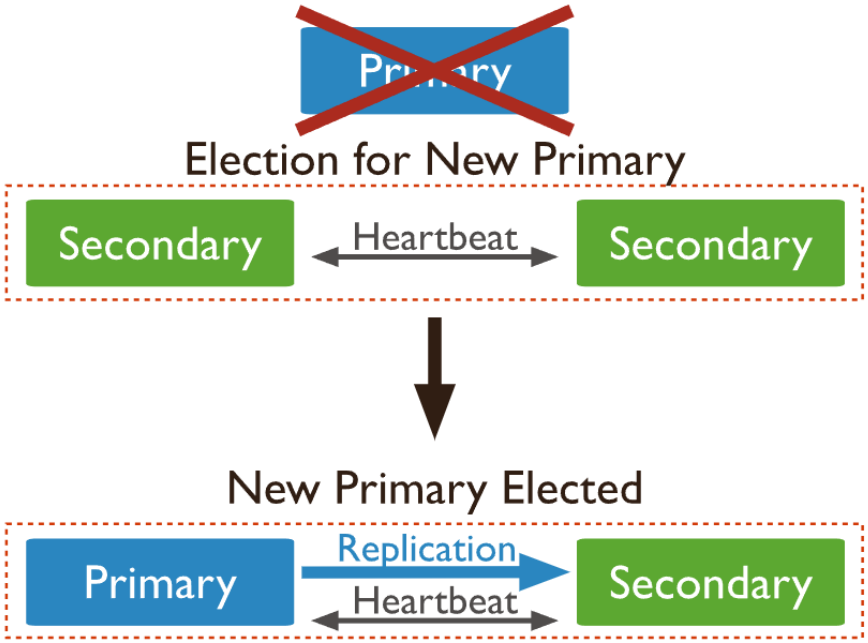
Automatki failover

Replika setovi koriste izbore (elections) kako bi odredili koji član skupa će postati primarni. U ovim izborima učestvuju svi članovi koji imaju pravo glasa (members[n].votes > 0). Replika skupovi pokreću izbore kao odgovor na razne događaje, kao što su:

- Dodavanje novog čvora u skup replika
- inicijalizacija replika skupa
- izvođenje održavanja (mainentance) skupa replika korišćenjem metoda kao što su **rs.stepDown()** ili **rs.reconfig()**
- sekundarni članovi koji gube konekciju sa primarnim članom na više od konfigurisanog vremenskog ograničenja (podrazumevano 10 sekundi – odnosno 5 hearthbeat-ova)

Ukoliko je primarni čvor je bio nedostupan duže od konfigurisanog vremenskog ograničenja, pokreće se proces automatskog failover. Jedan od preostalih sekundarnih članova poziva na izbore kako bi se izabrao novi primarni član i automatski nastavilo normalno funkcionisanje.

`



Sve do završetka izbora, set replika ne može da procesira write operacije. Može da izvršava read operacije, ukoliko su one konfigurisane da se čitaju sa sekundarnih čvorova (secondary ili secondaryPreferred postavljeno na read preference mode).

Dužina failovera zavisi od raznih faktora, ali pri korišćenju default-nih postavki, ne bi trebalo da traje duže od 12 sekundi (10 sekundi je potrebno za identifikaciju otkazanog člana, a 2 za održavanje izbora). Sa većim brojem članova, i većom mrežnom latencijom, ovaj proces može trajati i duže, dok se ne odabere novi primarmi član.

U narednom delu biće navedeni neki faktori koji utiču na dužinu failover-a

Verzija replikacionog protokola

Verzija replikacionog protokola može da utiče na vreme potrebno za oporavak od otkaza primarnog člana. Ukoliko se postavi **protocolVersion: 1**, to može smanjiti vreme potrebno za oporavak

Heartbeat-ovi

Po podrazumevanim postavkama, članovi seta replika šalju jedni drugima heartbeat-ove na svake dve sekunde. Ukoliko se odgovor ne vrati u roku od pet heartbeat-ova, član koji nije uzvratio odgovorom se smatra nedostupnim. Interval između heartbeat-ova se može

`

konfigurisati podesavanjem **heartbeatIntervalMillis** i **heartbeatTimeoutSecs**. Takođe, tajmaut između izbora se može podesiti pomoću **electionTimeoutMillis**.

```
"settings" : {
   "chainingAllowed" : true,
   "heartbeatIntervalMillis" : 2000,
   "heartbeatTimeoutSecs" : 10,
   "electionTimeoutMillis" : 10000,
   "catchUpTimeoutMillis" : 2000,
```

Prioritet članova

Nakon što skup replika ima stabilnog primarnog člana, algoritam za izbore će težiti da sekundarni član sa najvišim prioritetom pozove na izbore. Prioritet člana utiče i na vreme i na ishod izbora. Sekundarni članovi sa višim prioritetom pozivaju na izbore relativno brže od sekundarnih članova sa nižim prioritetom, i takođe imaju veću verovatnoću da pobede. Međutim, instanca sa nižim prioritetom može biti izabrana za primarnog na kratke periode, čak i ako je sekundarni član sa višim prioritetom dostupan. Članovi replika skupa nastavljaju da pozivaju na izbore dok član sa najvišim dostupnim prioritetom ne postane primarni.

Članovi sa prioritetom 0 (pr **hidden** ili **delayed**) ne mogu biti izabrani kao primarni čvor. Takođe, oni ne mogu pozvati izbore, iako na njima mogu da glasaju.

Particija mreže

Mrežna particija može odvojiti primarni čvor u particiju sa manjinom čvorova. U tom slučaju, primarni čvor je relativno izolovan, i može komunicirati samo sa manjim delom članova koji su deo skupa replika. Kada primarni čvor detektuje da može videti samo manjinu čvorova koji imaju prava glasa u skupu replika, on se samostalno povlači i postaje sekundarni. Nezavisno od toga, član u particiji koji može komunicirati sa većinom glasačkih čvorova (uključujući i sebe) organizuje izbore da postane novi primarni čvor.

`

Rollback tokom failovera

Povratak (rollback) poništava operacije upisa na bivšem primarnom čvoru kada se član ponovo pridruži svom skupu replika nakon što je drugi član preuzeo ulogu primarnog člana. Vraćanje je neophodno samo ako je primarni čvor prihvatio operacije upisa koje sekundarni članovi nisu uspešno replicirali pre nego što se primarni član povukao. Kada se primarni član ponovo pridruži skupu kao sekundarni, on poništava svoje operacije upisa, kako bi održao doslednost baze podataka sa ostalim članovima.

MongoDB pokušava da izbegne vraćanja, koja bi trebala biti retka. Kada se rollback dogodi, često je rezultat mrežne particije. Ukoliko je bivši primarni član bio u nekom delu mreže, sekundarni članovi koji nemaju direktnu komunikaciju sa njim. Sekundarni čvorovi koji ne mogu da prate protok operacija na bivšem primarnom čvoru povećavaju veličinu i uticaj rollback-a.

Rollback se ne dešava ako se operacije upisa repliciraju na drugog člana replika skupa pre nego što se primarni čvor povuče i ako taj član ostane dostupan i pristupačan većini replika skupa.

Za replika skupove, write concern { w: 1 } samo pruža potvrdu operacija upisa na primarnom čvoru. Podaci mogu biti vraćeni (rollback) ako se primarni čvor povuče pre nego što su operacije upisa replicirane na bilo koji od sekundarnih čvorova. Ovo uključuje podatke zapisane u transakcijama sa više dokumenata koje se potvrđuju koristeći write concern { w: 1 }.

Da bi se sprečilo vraćanje podataka koji su potvrđeni klijentu, svi članovi koji imaju pravo glasa trebaju biti pokrenuti sa omogućenim journaling-om. Takođe, korišćenje write concern { w: "majority" } (default nakon verzije 5.0 MongoDB-a) garantuje da se operacije upisa propagiraju na većinu čvorova replika skupa pre nego što vrate potvrdu klijentu.

Kada je writeConcernMajorityJournalDefault postavljen na false, MongoDB ne čeka da zapisi { w: "majority" } budu zapisani u journal na disku pre nego što potvrdi upise. Kao takve, "majority" operacije upisa bi mogle biti vraćene u slučaju otkaza i ponovnog pokretanja većine čvorova u nekom skupu replika.

Bez obzira na write concern operacije upisa, klijenti koji koriste "local" ili "available" read concern mogu videti rezultat operacije upisa pre nego što je operacija upisa potvrđena

`

klijentu koji ju je izdao. Klijenti koji koriste "local" ili "available" read concern mogu čitati podatke koji kasnije mogu biti rollback-ovani tokom preuzimanja uloge primarnog člana (failover) u skupu replika.

Za operacije u transakciji sa više dokumenata, kada se transakcija potvrdi, sve promene podataka napravljene u transakciji su sačuvane i vidljive van transakcije. To znači da transakcija neće potvrditi neke od svojih promena dok rollback-uje druge. Dok se transakcija ne potvrdi, promene podataka napravljene u transakciji nisu vidljive van transakcije.

Vreme izvršenja rollback-a po default-u je 24 časa, ali može biti promenjeno postavljanjem parametra **rollbackTimeLimitSecs**.

MongoDB podržava sledeće algoritme vraćanja (rollback), koji imaju različita ograničenja veličine:

Vraćanje na vremenski pečat (Recover to a Timestamp)

Ovaj algoritam omogućava bivšem primarnom čvoru da se vrati na konzistentnu tačku u vremenu i primeni operacije dok ne dostigne sinhronizaciju sa ostalim članovima. Ovo je default-ni algoritam za vraćanje. Kada se koristi ovaj algoritam, MongoDB ne ograničava količinu podataka koja se može vratiti.

Vraćanje putem ponovnog preuzimanja (Rollback via Refetch)

Ovaj algoritam omogućava bivšem primarnom čvoru da pronađe zajedničku tačku između svog oplog-a i oplog-a izvora sinhronizacije. Zatim član pregledava i rollback-uje sve operacije u svom oplog-u dok ne dostigne ovu zajedničku tačku. Vraćanje putem ponovnog preuzimanja se događa samo kada je **enableMajorityReadConcern** postavka u konfiguracionom fajlu postavljena na false.

`

Primena na replika skupu

Za potrebe praktičnog dela, biće napravljen replika skup od 3 člana. Pri kreiranju, jedan će biti primarni član, dok će ostala dva biti sekundarna.

```
mongod --replSet replikacija --logpath "1.log" --dbpath rs1 --port 27017 &
mongod --replSet replikacija --logpath "2.log" --dbpath rs2 --port 27018 &
mongod --replSet replikacija --logpath "3.log" --dbpath rs3 --port 27019
```

Ovom komandom, kreiraće se replika set, pod nazivom replikacija, sa tri člana, na portovima 27017, 27018, 27019.

Da bi se inicijalizovao replika set, koristi se komanda rs.initiate(). Njoj možemo da prosledimo konfiguraciju, da bi definisali kako će nas skup replika da izgleda. U ovom slučaju, prosleđujemo sledeću konfiguraciju.

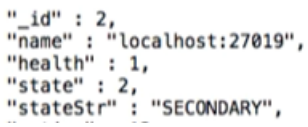
```
config = {_id: "replikacija", members: [
    { _id : 0, host : "localhost:27017"},
    { _id : 1, host : "localhost:27018"},
    { _id : 2, host : "localhost:27019"}
];

rs.initiate(config)
```

Pokretanjem komande rs.status() vidimo u kom su statusu naši članovi skupa replika.

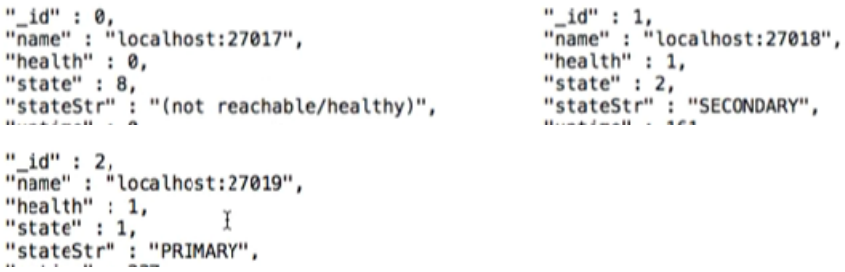
```
"members" : [
        {
                "_id" : 0,
                "name" : "localhost:27017",
                "health" : 1,
                "state" : 1,
                "stateStr" : "PRIMARY",
```

```
"_id" : 1,
"name" : "localhost:27018",
"health" : 1,
"state" : 2,
"stateStr" : "SECONDARY",
```

`



Ukoliko želimo da izazovemo failover, to možemo učiniti tako što ćemo ubiti člana koji je trenutno primarni. Ukoliko ubijemo primarnog člana, a nakon toga ponovo pokrenemo komandu rs.status(), dobijamo sledeće rezultate.



Vidimo da je pri otkazu primarnog člana, član sa id-jem 2 odabran kao novi primarni član, dok je bivši primarni član nedostupan.

`

Zaključak

U vremenu kada se svako od nas oslanja na internet za svakodnevnu komunikaciju i čuvanje svojih podataka, potrebno je da baze koje opsluzuju naše zahteve budu efikasne, stabilne i visoko dostupne. Da bi se postiglo normalno funkcionisanje i pružanje usluga, koristno je implementirati skup replika.

U slučaju otkaza, ili nekih drugih problema, klijent i dalje može biti opslužen, i njegov zahtev izvršen. To može dovesti do boljeg korisničkog iskustva, kao i poboljšati bezbednost i konzistentnost podataka koji se čuvaju u bazama podataka.

Kroz ovaj rad predstavljen je i teoretski objašnjen proces kreiranja replika, njihov izgled i konfiguraciju, kao i način rada. Predstavljen je način na koji se MongoDB baza podataka oporavlja kada se dese neke greške, i kako uspeva da pruži visoku dostupnost i konzistentne podatke klijentima, čak i ukoliko dođe do nepredviđenih otkaza.

`

Reference

[1] https://www.mongodb.com/company/what-is-mongodb

[2] https://rivery.io/data-learning-center/complete-guide-to-data-replication/

[3] https://www.fivetran.com/learn/database-replication#:~:text=Download%20report-,What%20is%20database%20replication%3F,-to-date)%20data.

[4] https://www.mongodb.com/docs/manual/core/replica-set-primary

[5] https://www.mongodb.com/docs/manual/core/replica-set-secondary

[6] https://www.mongodb.com/docs/manual/core/replica-set-arbiter

[7] https://www.mongodb.com/docs/manual/core/replica-set-oplog/

[8] https://www.mongodb.com/docs/manual/core/replica-set-sync/