**Univerzitet u Nišu**

**Elektronski fakultet**

**Obrada transakcija, planovi izvršavanja transakcija, izolacija i zaključavanje**

**MongoDB**

**Student:** Jovana Nikolić 1147

**Profesor:** Doc. dr Aleksandar Stanimirović

**Predmet:** Sistemi za upravljanje bazama podataka

Sadržaj

[Uvod 3](#_Toc72727253)

[Transakcije 4](#_Toc72727254)

[Razlog uvođenja višedokumentnih ACID transakcija 5](#_Toc72727255)

[Višedokumentne ACID transakcije u MongoDB-u 7](#_Toc72727256)

[Čitanja van transakcije za vreme commit-a 8](#_Toc72727257)

[Transakcije i read preference 10](#_Toc72727258)

[Transakcije i read concern 11](#_Toc72727259)

[Transakcije i write concern 12](#_Toc72727260)

[Snapshot izolacija i konflikti pri upisu 13](#_Toc72727261)

[Ponovni pokušaj transakcija 13](#_Toc72727262)

[Kada koristiti višedokumentne transakcije 13](#_Toc72727263)

[Saveti prilikom korišćenja transakcija 14](#_Toc72727264)

[Izolacija 14](#_Toc72727265)

[Fantomska čitanja 15](#_Toc72727266)

[Non repeatable reads – Čitanja koja se ne ponavljaju 15](#_Toc72727267)

[Dirty reads – Prljava čitanja 16](#_Toc72727268)

[Izolacija kod MongoDB-a 16](#_Toc72727269)

[Read uncommitted i atomičnost pojedinačnog dokumenta 16](#_Toc72727270)

[Read uncommited i upis u više dokumenata 17](#_Toc72727271)

[Zaključavanje 18](#_Toc72727272)

[Koliko su granularna zaključavanja u MongoDB bazi podataka 19](#_Toc72727273)

[Izbegavanje zaključavanja 19](#_Toc72727274)

[Kako pratiti zaključavanje? 20](#_Toc72727275)

[Zaključak 22](#_Toc72727276)

[Literatura 23](#_Toc72727277)

# Uvod

MongoDB baza podataka je najpopularnija i najkorišćenija NoSQL baza podataka, spada u dokumentu orjentisane NoSQL baze podatala. Koristi JSON ili BSON dokumente za smeštanje podataka. MongoDB baza podataka je postalo open-source 2009. godine, pri čemu je jedna od njenih glavnih karakteristika performanse. MongoDB je nerelaciona baza koja je pravljena kao jako bliska relacionim bazama podataka, sa mogućnošću kreiranja sopstvene šeme, koja je jako fleksibilna [1]. Umesto korišćenja tabela za smeštanje podataka, podaci su smešteni u kolekcijama, dok umesto jasno definisanih i uređenih redova kod relacionih baza podataka, kod MongoDB baze podataka, podaci se smeštaju u dokumentima, koja se sastoje od key-value parova, ime key-value para predstavlja polje, koje predstavlja pandan kolonama kod relacionih baza podataka [2]. MongoDB baza podataka je od svih NoSQL baza podataka, baza koja najviše liči na relacione baze podataka, i koja kao cilj ima posedovanje mnogih svojstava relacionih baza podataka. Jedno od tih svojstava definitivno jesu transakcije. Smatra se da je razlog uvođenja transakcija kod MongoDB baze podataka dobijanje konkurentnosti na tržištu, i otvaranje mogućnosti saradnje sa novim kompanijama sa kojima do postojanja transakcija nisu mogli poslovati, jer one zatevaju postojanje ACID transakcija za regularni rad. Izolacija se definiše na malo drugačiji način, i definiše se jako, nezahvalno koriteći tri parametra, dok je zaključavanje jako slično metodama zaključavanja kod relacionih baza podataka kao što je Oracle.

Za izvršavanje višedokumentnih transakcija, neophodno je imati replika set, te je korišćen MongoDB Atlas sajt za kreiranje klastera, koji se nalazi na [3], treba napomenuti da je moguće konfigurisati set replika pomoću IP adresa i mongo shell-a, ali da zahteva postojanje više računara koji će se upravo konfigurisati pomoću IP adresa.

# Transakcije

Transakcija predstavlja osnovnu jedinicu rada u okviru jednog DBMS-a. Transakcija predstavlja bilo koju promenu u okviru baze podataka [4]. Sistemi baza podataka treba da garantuju konzistenciju i integritet podataka, naročito u situacijama kada su se radi sa osetljivim podacima.[5] Transakcije se kreiraju radi:

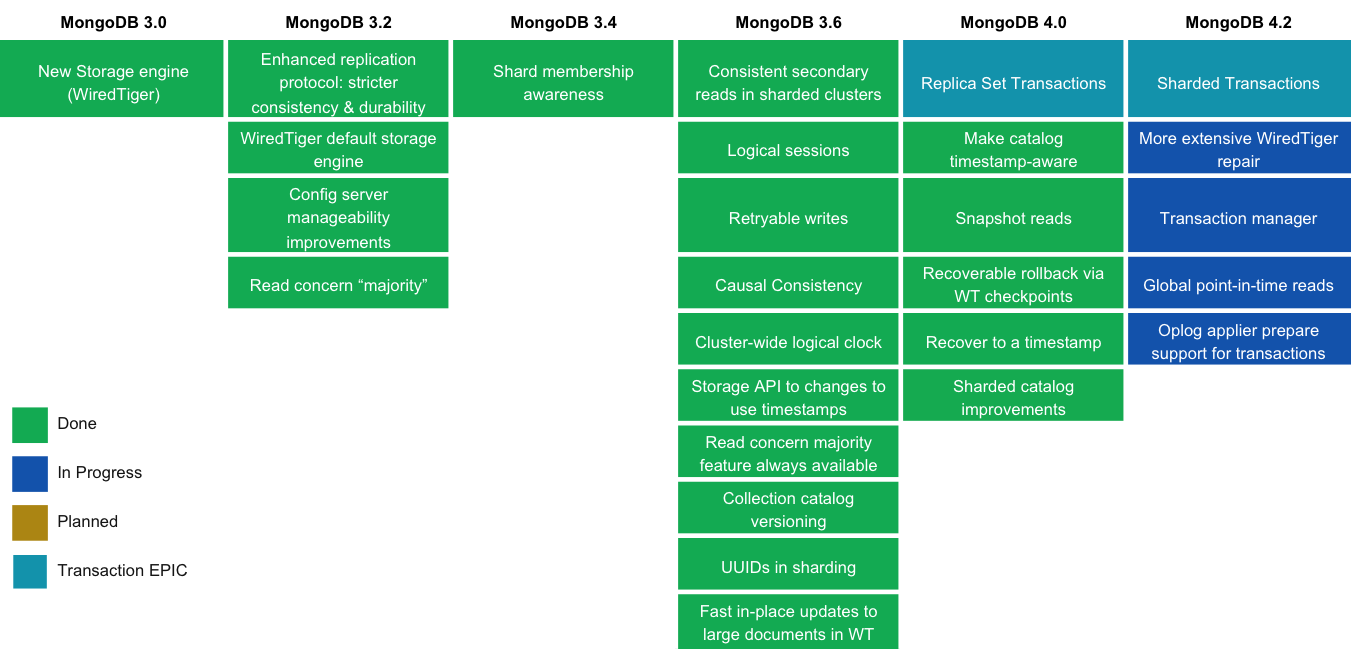
* Obezbeđivanja sigurnosti, validnosti i konzistencije podataka ukoliko sistem prestane da radi ili naiđe na grešku, jer u slučaju greške veliki broj operacija nad bazom podataka biva nezavršen, te je njihov status izvršenja nepoznat.
* Obezbeđivanje izolacije između programa koji pristupaju bazi podataka konkurentno. Bez postojanja izolacije, rezultati, i samo stanje baze podataka može biti pogrešno.

Kod DBMS-a transakcija predstavlja jedinicu rada, koja se ponekada sastoji od više operacija. Tranasakcija kod DBMS-a mora zadovoljavati ACID svojstva [4]. ACID svojstva su sledeća:

* *Atomičnost –* Operacija se ili mora izvršiti u celosti ili nemati nikakav uticaj (kao da se nije ni izvršavala) [6] .
* *Konzistencija –* Transakcija može prevesti bazu podataka samo iz jednog važećeg stanja u drugo važeće stanje, održavajući invarijante baze podataka [5] . Odnosno da sve promene u bayi podataka budu u skladu sa svim prethodno definisanim pravilima [7].
* *Izolacija –* Istovremeno izvršenje transakcija ostavlja bazu podataka u istom stanju koje bi se dobilo ukoliko bi transakcije bile izvršene sekvencijalno [5].
* *Trajnost –* Transakcija mora biti upisana u trajnu memoriju, te ukoliko sistem padne transakcija će biti osigurana [6].

Transakcije omogućavaju očuvanje integriteta podataka u okviru baze podataka, nakon izvršenja svake transakcije baza podataka ostati u konzistentnom stanju. Kao primer problema zbog kojih su i kreirane transakcije, može se uzeti transfer novca sa jednog računa na drugi. Ukoliko se novac sa izvornog računa uslovno rečeno skine, a nikada se ne prebaci na račun kome je bio namenjen, dolazi se do ozbiljnog problema. Jednako veliki problem bi bio ukoliko se novac prenese na željeni račun, a ta ista suma ne ukloni sa računa uplatioca. Oba upisa se ili moraju izvršiti ili ne izvršiti, da bi sistem ostao u ispravnom stanju. Ono što treba imati u vidu jeste da korišćenje transakcija ima svoje posledice, te DBMS mora da uradi dosta stvari u pozadini, da bi imao uvida o toku transakcije, i mora zaključati resurse koji su priključeni transakciji, što znači da drugi klijenti za to vreme ne bi imali pristup pomenutim resursima. Nedostupnost resursa klijentima bi za posledicu imala veliku latenciju, i ugrozila bi iskustvo korisnika [8]. Prva generacija NoSQL baza podataka je kreirana, zanemarujući značaj ACID karakteristika, smatrajući da su ACID svojstva potrebna samo starodmodnim aplikacijama koje svoje podatke čuvaju u jednom data centru. Premisa je bila da se moderne distribuirane aplikacije trebaju fokusirati na linearnu skalabilnost propraćenu sa niskom latencijom, najčešće tačnim operacijama nad jednim ključem, u deljivom prostoru. Takva pretpostavka se pokazala pogrešnom jer je veliki broj kompanija u oblasti finansija, prodaje, gejminga, želela korisničke aplikacije koje imaju integritet transakcija, nisku latenciju i lineranu skalabilnost kroz više cloud regiona. Trenutno je mali broj NoSQL baza podataka izašao sa pravim rešenjima za manjak podrške za transakcije u svojim rešenjima, MongoDB je jedna od retkih baza podataka koja je izašla sa verzijama koje podržavaju postojanje transakcija [9].

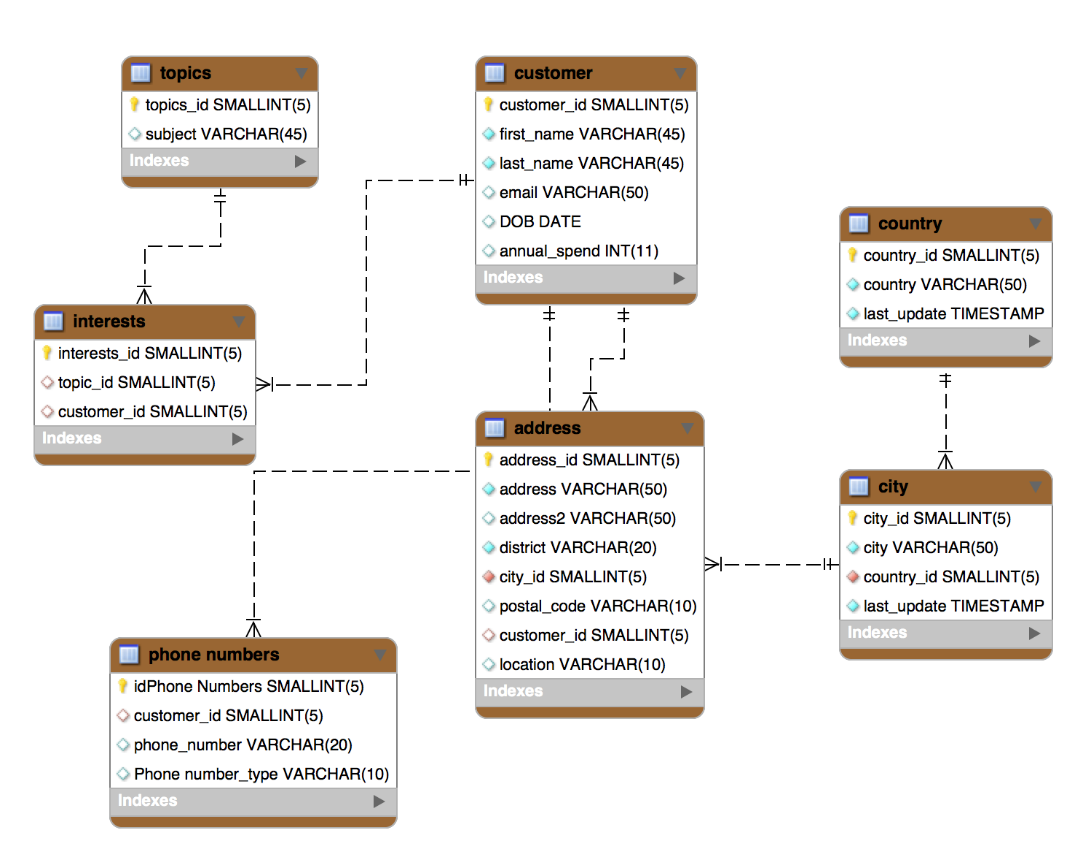
MongoDB baza podataka ima jako drugačiji model podataka u odnosu na relacione baze podataka. Dokument model za cilj ima smeštanje međusobno bliskih podataka u isti dokument, te je MongoDB uvek podržavao svojstva ACID transakcija nad istim dokumentom, samim tim što većini aplikacija nisu neophocne garancije ACID svojstava nad više dokumenata. Zarad zadovoljanja svih mogućih use-case scenarija od verzije 4.0 MongoDB podržava transakcije nad istom replikom, dok je od verzije 4.2 podržava transakcije nad razdeljenim podacima. Na slici 1 se može videti napredak i put ka kreiranju višedokumentnih transakcija kod MongoDB baze podataka.



*Slika 1 – Uvođenje višedokumentnih transakcija*

## Razlog uvođenja višedokumentnih ACID transakcija

MongoDB baza podataka je kreirana 2009. godine sa idejom o novom pristupu dizajnu šema baza podataka, oslobađajući ih time striktnih ograničenja relacionih baza podataka. Sam dizajn baziran na fleksibilnim, proizvoljno velikim dokumentima, je omogućio 3-5 puta brži razvoj aplikacija. MongoDB baza podataka takođe omogućava distribuiranu arhitekturu koja pruža mogućnost smeštanja veće količine podataka, smeštanje na lokacijama koje su pogodne korisniku, kao i konstantnu raspoloživost. Zbog svojih osobina MongoDB je peta najkorišćenija baza podataka, kao i najkorišćenija od svih NoSQL rešenja, pri čemu svoju primenu nalazi u svim granama industrije [11]. U nastavku će biti prikazan razlog zašto jako dugo nije došlo do uvođenja višedokumentnih transakcija, na primerima relacionog modela podataka i njegovog prenosa na dokument model podataka.



*Slika 2 – Model podataka korisnika u relacionoj bazi podataka*

U slučaju bilo koje promene korisnika, na primer ukoliko dođe do promene posla, onda je veliki broj tabela koje će morati da se menjaju kao posledica *update* operacije, u jednoj transakciji, koja će uticati na više tabela, i desiće se promena ili u svim tabelama ili ni u jednoj.

Kod dokument modela situacija je drugačija. Umesto podele podataka i njihovog deljenja na više različitih roditelj-dete tabela, dokumenti mogu da smeste sve povezane dokumente u hijerarhijskoj strukturi koja uključuje poddokumente i nizove koja je prikazana na slici 2.



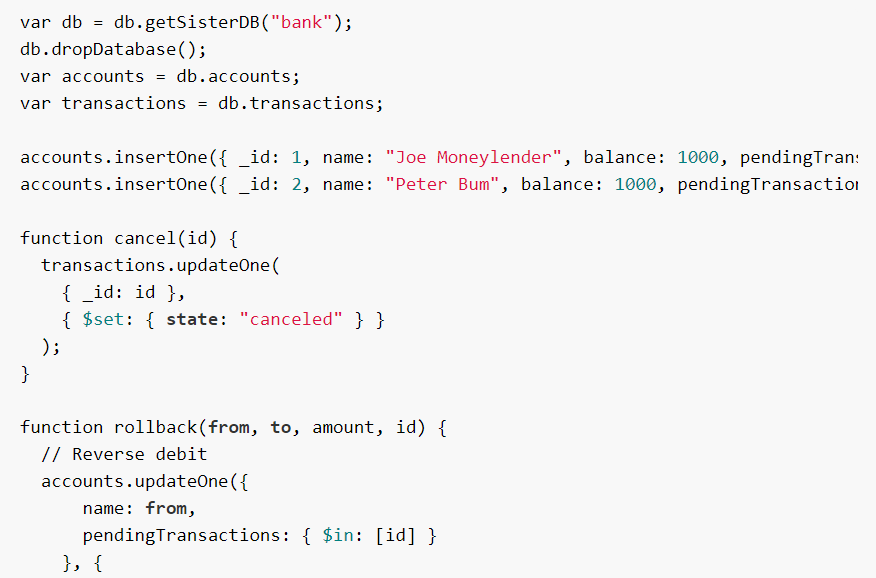
*Slika 3 – Dokument model podataka korisnika*

MongoDB obezbeđuje sve osobine transakcija, na nivou dokumenata, te izmena jednog ili više polja, ili poddokumenata je omogućena. Garancije koje obezbeđuje MongoDB, osiguravaju apsolutnu izolaciju za vreme modifikacije dokumenta, bilo koja greška za vreme izvršenja operacije izaziva roll-back operacije, omogućavajući klijentima konzitentan pogled na dokument.

## Višedokumentne ACID transakcije u MongoDB-u

U MongoDB bazi podataka, operacije nad jednim dokumentom su atomične. One su atomične jer se komd MongoDB-a koristi ugnježdavanje dokumenata kao i nizovi, za prestavljanje veza između dokumenata, umesto normalizacije podataka kroz više dokumenata i kolekcija. Samo modelovanje podataka smanjuje potrebu za postojanjem višedokumentnih transakcija za mnoge primene.

Pre postojanja transakcija, jedini način za implementaciju nečega sličnog transakcijama bilo je korišćenje commit procesa u 2 faze, čiji je jedan primer prikazan na slici 3 [11].



*Slika 4 – Primer commit procesa u dve faze*

Počev od verzije 4.0 MongoDB omogućava višedokuementne transakcije nad setovima replika. Počev od verzije 4.2 MongoDB uvodi distribuirane transakcije, što obezbeđuje podršku za višedokumentne transakcije nad shard-ovanim(izdeljenim) klasterima. Jako je bitno update-ovati MongoDB drajvere koje korisnici koriste na verziju 4.2 ili višu ukoliko se koriste višedokumentne transakcije.

Višedokumentne transakcije su atomične:

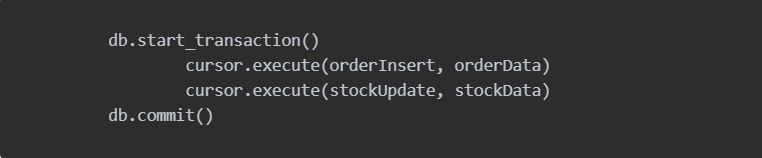
* Kada transakcija commit-uje svoje rezultata, sve promene napravljene u okviru transakcije su sačuvane i vidljive su van transakcije. Te, transakcija nikada neće sačuvati neke izmene, a roll-back-ovati druge. Dok transakcija ne commit-uje, promene nisu vidljive van transakcije. Ali, kada transakcija upisuje na više shard-ova, ne moraju sve operacije čitanja van transakcije da sačekaju da svi podaci budu vidljivi na svim shar-ovima. Na primer, ukoliko je transakcija commit-ovana i upis 1 je vidljiv na shard-u 1, ali upis 2 nije vidljiv još uvek na shard-u B, read operacija van transakcije, koja ima read concern local, može da pročita rezultate upisa 1, bez viđanja upisa 2.
* Ukoliko transakcija prestane, sve promene koje su obavljene se odbacuju, bez toga da ikada postanu vidljive.

## Čitanja van transakcije za vreme commit-a

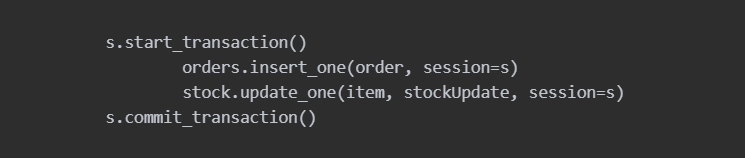
Za vreme commit-a transakcije, druge operacije možda pokušaju da pročitaju isti dokument,koji treba biti modifikovan od strane transakcije. Ukoliko transakcija vrši upis na više shard-ova:

* Spoljašnja čitanja koja koriste read concern snapshot ili linerizable, čekaju da svi upisi transakcije budu vidljivi
* Spoljašnja čitanja koja koriste druge read concern opcije, ne čekaju da svi upisi transakcije budu vidljivi, već čitaju verziju dokumenata pre same transakcije [12].

Transakcije u MongoDB bazi podataka jako liče na transakcije koje su korišćene kod relacionih baza podataka. One sadrže više naredbi, sa jako sličnom sintaksom, što ih čini jako poznatim bilo kom programeru koji ima ranije iskustvo u radu sa transakcijama. Naredbe koriste sintaksu MongoDB *query laguage-*a. Sledeće slike prikazuju sličnosti između MongoDB sintakse i sintakse korišćene od strane MySQL DBMS-a, što prikazuje celokupnu tendenciju MongoDB developera da bazu učini što sličnijom tradicionalnim, relacionim bazama podataka, kako bi olakšali prelazak sa relacionih baza podataka na MongoDB bazu podataka.

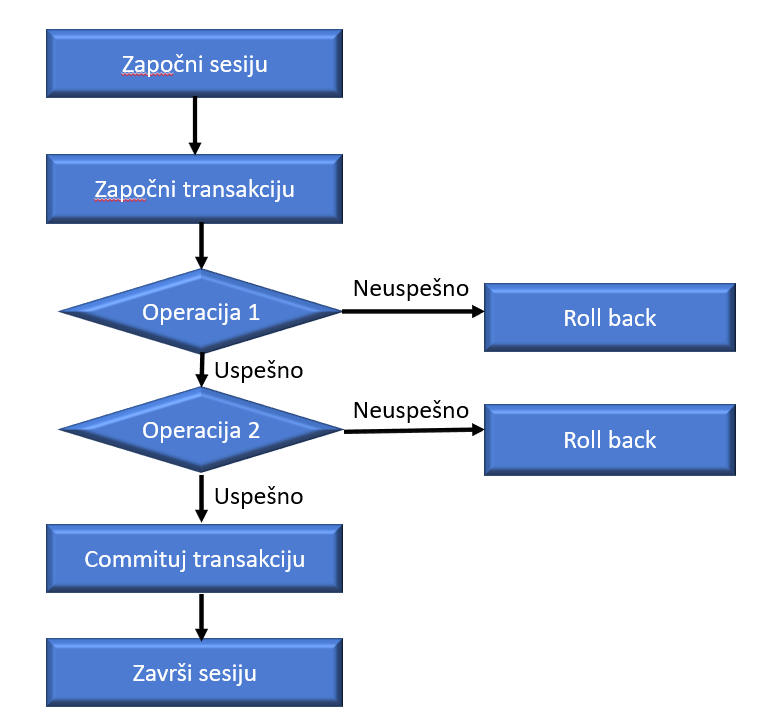


*Slika 5- MySQL primer transakcije*

**

*Slika 6 - MongoDB primer transakcije*

Na sledećoj slici je prikazana šema izvršavanja proizvoljne transakcije, sa dve operacije.



*Slika 7 -. Primer izvršenja transakcije*

Na primeru se može videti da je pre početka same transakcije neophodno započeti sesiju, što se obavlja sa naredbom *startSession(),* čije je korišćenje prikazano na slici ispod, u okviru Mongo *shell*-a.



*Slika 8 – Početak sesije*

Zatim je neophodno započeti izvršenje transakcije, to se obavlja izvršenjem naredbe *Session.startTransaction(),* koja započinje izvršenje višedokumentnih transakcija, za jednu sesiju najviše je moguće imati otvorenu jednu sesiju. Opciono se mogu definisati vrednosti za read concern i write concern [13]. Pri čemu read concern kontroliše konzistenciju i izolaciju podataka pročitanih sa replika ili delova replika, i to omogućava pet nivoa readConcern-a: local, avaliable, majority, linerizable, snapshot [14]. Dok writeConcern definiše broj potrvda od strane MongoDB baze podataka za operacije upisa [15]. Na slici ispod se nalazi primer započinjanja transakcije.



*Slika 9 – Početak transakcije*

Nakon početka izvršenja transakcije se mogu izvršavati razne operacije čitanja i upisa, od kojih je jedan primer dat na slici ispod. Ukoliko se desi do iznenadnog prestanka transakcije, nakon bilo koje operacije, izmenjeni podaci neće biti sačuvani.



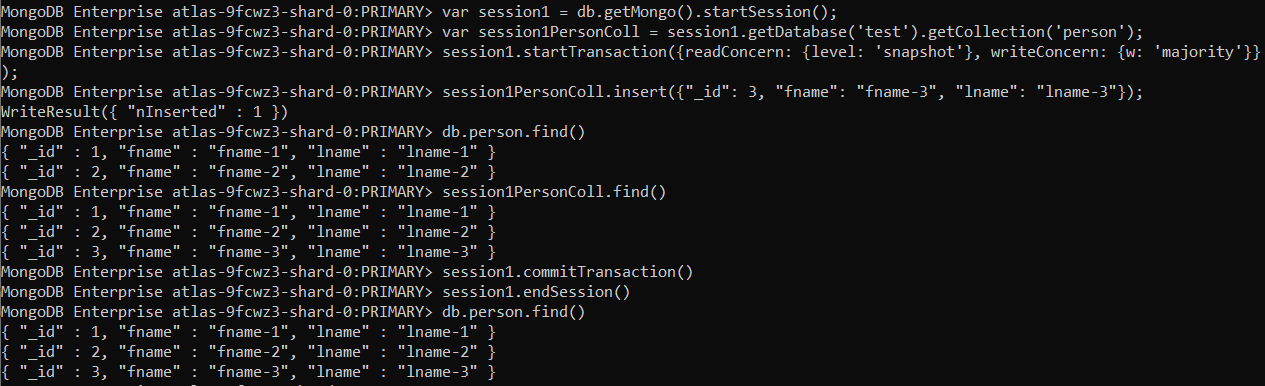
*Slika 10– Izvršenja operacije u okviru transakcije*

Ukoliko su sve operacije uspešno završene, neophodno je izvršiti perzistiranje izmena, odnosno izvršiti commit. Nakon toga je dobra praksa zatvoriti sesiju. Bitno je napomenuti da ukoliko se sesija završi sa otvorenom transakcijom, transakcija prestaje. Na slici ispod su prikazane komande za commit transakcije i završavanje sesije.



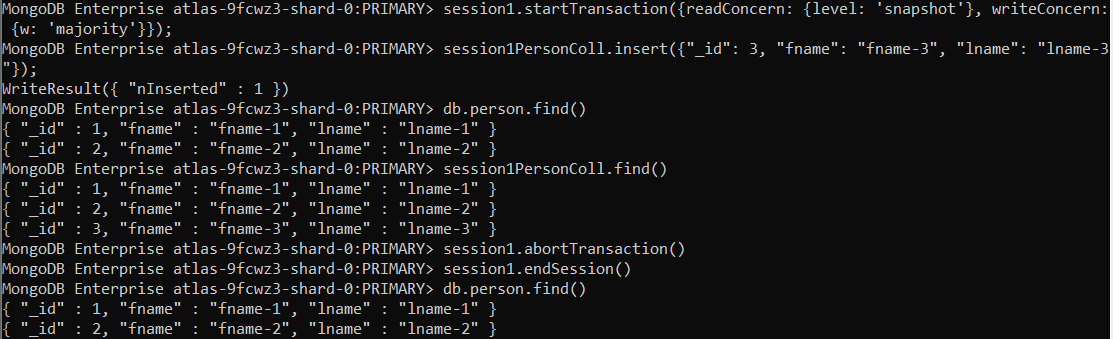
*Slika 11 – Commit transakcija i završavanje sesija*

Naredna slika predstavlja manipulaciju podacima u kolekciji osoba, koja sadrži određene podatke u sebi, poenta primera je prikaz, da se promene izvršene od strane operacija u transakciji vide isključivo u transakciji, sve dok transakcija nije commit-ovana.



*Slika 12 – Prikaz osobina transakcija*

Slika ispod prikazuje kako se nijedna promena kreirana u okviru transakcija ne vidi van transakcije ukoliko se prekine transakcija, isto bi važilo ukoliko bi došlo do neke greške.



*Slika 13 – Prikaz efekta prestanka rada transakcije ili pozivanja abortTransaction*

## Transakcije i read preference

Read preference definiše način na koji MongoDB klijenti rutiraju operacije čitanja članovima skupa replika. Po difoltu aplikacija usmerava svoje operacije čitanja primarnom članu skupa replika, ali klijent može sam definisati read preference. Modovi read preference su:

* *primary* – Difoltni mod. Sve operacije čitaju sa primara. Višedokumentne transakcije moraju da koriste preferencu primary.
* *primaryPreffered* – U najvećem broju situacija čita se sa primara, ali ukoliko nije moguće, čitati sa sekundarnih članova.
* *secondary* – Sve operacije čitaju sa sekundarnih članova.
* *secondaryPreffered* – Sve operacije čitaju sa sekundarnih članova skupa replika.
* *nearest* –Operacije čitaju sa random slobodnog člana skupa replika, bazirano na vrednosti latencije [16].



*Slika 14 – Postavljanje read preference*

## Transakcije i read concern

Read concer opcija dozvoljava kontrolu konzistencije i izolacije podataka pročitanih iz setova replika ili shard-ova replika setova [14]. Operacije u okviru transakcija koriste read concern koji je definisan od strane transakcije. Odnosno bilo koji read concern koji je definisan na nivou kolekcije ili baze je ignorisan unutar transakcije.

Nivo read concern-a za transakciju se može definisati na samom početku transakcije.

* Ukoliko nivo nije definisan za transakciju, postavlja se nivo definisan sesijom.
* Ukoliko ni jedan, ni drugi nisu definisani, transakcioni nivo read concern-a je definisan klijentskim nivoom read concern-a. Po difoltu, klijentski nivo read concern-a je „local“, za sva čitanja primara.

Transakcije podržavaju sledeće nivoe read concern-a

* "local" – Vraća najsvežije moguće podatke, ali podaci mogu biti roll-back-ovani. Kod transakcija nad klasterom koji je shard-ovan, ovaj nivo ne garantuje da su podaci isti koji su na svim ostalim shard-ovima.
* "majority" – Vraća podatke koji su potvrđenih od strane većine članova skupa replika, ukoliko se transakcija commit-uje sa write concern-om "majority". U suprotnom "majority" read concern ne nudi nikakve garancije, da operacije čitaju većinu commit-ovanih podataka. Kod transakcija nad klasterom koji je shard-ovan, ovaj nivo ne garantuje da su podaci isti koji su na svim ostalim shard-ovima.
* "snapshot" – Vraća podatke iz snapshot-a većine commit-ovanih podataka, ukoliko je transakcija commit-ovana sa write concern-om "majority". Ukoliko transakcija ne koristi "majority" write concern, read concern ne nudi nikakve garancije, da operacije čitaju većinu commit-ovanih podataka. Kod transakcija nad klasterom koji je shard-ovan, ovaj nivo ne garantuje sinhronizovanost podataka na svim shard-ovima.

Osim navedenih postoje još dva nivoa read concern-a:

* "available" – Upit vraća podatke sa instance, bez garancija, da je podatak upisan na većinu replika (podaci mogu biti roll-back-ovani)
* "linearizable" – Upit vraća podatke koji prikazuje sve upise koji su potvrđeni od većine, a izvršeni pre početka starta operacije čitanja [15].

Na sledećoj slici je dat prikaz podešavanja read concern nivoa na "majority".



*Slika 15 – Podešavanje read concern-a*

Kroz snapshot izolaciju, transakcije omogućavaju jedinstveni pogled na podatke, i primenjuju sve ili ništa izvršenje, radi očuvanja integriteta podataka. Transakcije se mogu izvršavati za operacije nad više dokumenata u jednoj ili više kolekcija, i baza. Promene koje je MongoDB da bi omoućio uvođenje višedokumenih transakcija ne utiču na performanse upita kojima one nisu potrebne. Za vreme izvršenja transakcija može da očita svoje upise koji još uvek nisu commit-ovani, ali nijedan upis koji nije commit-ovan nije vidljiv od strane operacija van transakcije. Upisi koji nisu commit-ovani nisu replicirani na sekundarne čvorove, sve dok transakcija nije commit-ovana. Nakon commit-a transakcije, upisi se mogu replicirati na sve sekundarne replike.

Sama aplikacija specificira write concern, kao opcije u transakciji, koje određuju koliko čvorova moraju commit-ovati promene, pre nego što server vrati potvrdu o uspešno izvršenoj transakciji klijentu. Svi upisi koji nisu commit-ovani se nalaze isključivo na primarnoj kopiji.

Snapshot read concern omogućava da upiti i agregacije izvršene u okviru read-only transakcije se izvršavaju samo nad jednom, izolovanom snapshot-u na primarnoj replici, samim tim se korisniku vraća konzistentan pogled na podatke, bez obzira na operacije koje menjeju podatke.

## Transakcije i write concern

Write concern opisuje nivo potvrđivanja zahteva koji se traži od strane MongoDB-a, kod operacija upisa [15]. Operacije u okviru transakcija koriste write concern koji je definisan od strane transakcije. Odnosno bilo koji read concern koji je definisan na nivou kolekcije ili baze je ignorisan unutar transakcije.

Nivo write concern-a za transakciju se može definisati na samom početku transakcije.

* Ukoliko nivo nije definisan za transakciju, postavlja se nivo definisan sesijom.
* Ukoliko ni jedan, ni drugi nisu definisani, transakcioni nivo write concern-a je definisan klijentskim nivoom write concern-a. Po difoltu, klijentski nivo write concern-a je w:1, za sve upise.

Transakcije podržavaju sve moguće vrednosti , pri čemu :

* w:1 – Vraća potvrdu, nakon što je commit izvršen na primaru.
* w: "majority" – Vraća potvrdu nakon što se commit izvršio na većini primaru i M-1 sekundarnih replika.

Na sledećoj slici je prikazano postavljanje writeConcern-a na nivou transakcije [17].



*Slika 16 – Postavljanje write concern-a na nivou transakcije*

## Snapshot izolacija i konflikti pri upisu

Kada se dokument modifikuje, transakcija zaključava dokument, time zabranjujući dodatne promene dok se transakcija ne završi. Ukoliko transakcija ne uspe da zaključa dokument koji želi da izmeni, najčešći razlog za to je da druga transakcija već ima ključ nad dokumentom, te će nakom otprilike 5ms transakcija prestati i javiti konflikt pri upisu. Ukoliko tipičan upis želi da promeni dokument koji je trenutno zaključan od strane višedokumentne transakcije, taj upis će biti blokiran, sve dok se transakcija ne izvrši. Upis će biti proban ponovo neograničen broj puta, sve dok ne istekne vreme definisano sa *$maxTimeMS.* Čitanje ne zahteva isto zaključavanje kao modifikacija dokumenata. Dokumenta koja u sebi sadrže upise koji nisu commit-ovani, idalje mogu biti pročitana od strane read operacija, ali naravno read operacija vraća samo promene koje su commit-ovane. Jedino upis izaziva konflikt kod MongoDB baze podataka. Čitanje, kao što je predviđeno snapshot izolacijom, ne sprečava menjanje dokumenta od strane drugih operacija. Takođe no-op operacije kao što je set-ovanje polja na vrednost koju je već imalo, mogu biti optimizovane pre dolaska do strage engine-a, samim tim, ne izazivajući konflikt pri upisu.

## Ponovni pokušaj transakcija

Počev od verzije 4.0 MongoDB uvodi koncept labela grešaka. Labela transientna transakciona greška, obaveštava aplikaciju da je došlo do greške, čiji uzrok može biti u opsegu greška u mreži, do konflikta pri upisu, te da greška može biti privremena i da je sigurno ponovo izvršiti transakciju. Trajne greške, kao što je greška u parsovanju ne grenerišu tranzientna transakcionu grešku, jer ponovno izvršenje transakcije neće dovesti do uspešnog commit-a.

Jedna od najvažnijih karakteristika MongoDB baze podataka jeste arhitektura koja obezbeđuje visoku dostupnost. Labele grešaka, samim tim omogućavaju aplikacijama da budu otporne na greške u mreži ili otkaze u čvorovima.

## Kada koristiti višedokumentne transakcije

Višedokumentne transakcije se koriste u situacijama kada su neophodna ACID svojstva za skup operacija nad više dokumenata. Aplikacije tipa *System of record* ili *Line od Business* su tipični predstavnici aplikacija gde su neophodne višedokumentne transakcije. Takvi primeri su:

* Obrada događaja u okviru aplikacije u slučajevima kada korisnik obavlja bitne akcije, kao što je promena stanja računa, istovremeno nad više računa
* Logovanje korisničkih aktivnosti
* N ka n relacije, gde je prirodan način smeštanja podataka, smeštanje podataka u objekte

Korišćenjem transakcija, korisnici MongoDB baze podataka skraćuju kod za u proseku oko 1000 linija. Takođe MongoDB komapnija je pronašla da je prilikom korišćenja višedokumentnih transakcija umesto in-house rešenja, latencija smanjena za 60% i propusnost povećana za 90% za transakcije koje su izvršavale dve operacije *update* nad dve kolekcije.

## Saveti prilikom korišćenja transakcija

Kreiranje dugačkih transakcija, ili pokušavanje izvršavanja predugačkih transakcija u okviru jedne ACID transakcije može izazvati veliki pritisak na WiredTriger keš. Razlog za to je to što keš mora da čuva stanje za svaki upis od poslednjeg snapshot-a, tokom rada, novi upisi se akumulišu u kešu za vreme izvršenja transakcije. Ovi upisi se ne mogu izbrisati sve dok transakcija ne commit-uje svoje rezultate ili abort-uje, jer u tom trenutku transakcije otpuštaju svoje lock-ove i WiredTriger može izbaciti podatke. Saveti prilikom korišćenja transakcija:

* Po difoltu, MongoDB baza podataka automatski zaustavlja sve višedokumentne transakcije koje su duže od 60 sekundi. Kako bi se izbegle posledice timeout-a, transakcije treba da budu razdeljene u manje delove koje dozvoljavaju izvršenje u konfigurisanom vremenskom periodu. Takođe dobro je imati optimizovane upite, sa dobro kreiranim indeksima, kako bi se obezbedio brz pristup podacima prilikom transakcije.
* Ne postoji limit u broju dokumenata koji mogu biti pročitani za vreme jedne transakcije. Kao dobra praksa, smatra se da ne treba modifikovati više od 1000 dokumenata tokom jedne transakcije. Za operacije koje menjaju više od 1000 dokumenata savetuje se deljenje transakcije u više delova koji će vršiti obradu dokumenata u batch-evima
* Počev od verzije 4.0, transakcija je predstavljena jednim *oplog* poljem, te dokument mora maksimalno zauzeti 16MB. Update operacije sadrže samo informacije o onome što se promenilo, insert operacije čuvaju ceo dokument. Ukoliko se pređe limit, transakcija će prestati sa izvršenjem i izvršiće se roll-back.
* Kada transakcije prestane, korisniku se vraća exception, i transakcija se roll-back-uje. Na programerima je ostavljeno kreiranje aplikacione logike, koja hvata exception i ponavlja izvršenje transakcije ukoliko je moguće [18].

# Izolacija

Izolacija se odnosi na vidljivost transakcionih operacija, drugim operacijama koje se paralelno izvršavaju.

U nastavku je primer koji objašnjava važnost nivoa izolacije:

* Transakcija A promeni korisniku 1 stanje na računu sa 50 na 100 dinara, ali ne commit-uje transakciju
* Transakcija B pročita da je korisniku 1, stanje na računu 100 dinara
* Transakcija A se roll-back-uje, pri čemu se stanje računa korisnika 1 ponovo vraća na 50
* Transakcija B misli da korisnik 1 ima 100 dinara, iako ustvari ima 50 dinara
* Transakcija B promeni korisniku 2 stanje na računu, dodavajući mu 100 dinara. Korisnik 2 dobija 100 dinara bez osnove, jer korisnik 1 ima samo 50 dinara

Izolacija načešće ima 4 nivoa, koja su navedena od najstriktnijih, do najmanje striktnih:

* *Serializable*
* *Repeatable read* (Ponovljeno čitanje)
* *Read commited* (Čitaj commit-ovano)
* *Read uncommited* (Čitaj ono što nije commit-ovano)

Problemi na koje može da se naiđe, u redosledu od najmanje ozbiljnih, do najozbiljnijih u zavisnosti od nivoa izolacije su:

* Fantomska čitanja
* Non repeatable reads (čitanja koja se ne mogu ponavljati)
* Dirty read (Prljava čitanja)
* Izgubljene izmene

Gubljenje podataka o operativnim promenama je najgora stvar koja se može desiti jednoj bazi podataka, jer čini bazu podataka beskorisnom, i smešta podatke za koje se ne može tvrditi da su tačni. Zbog toga, bez obzira na nivo izolacije, čak i izolacija read uncommited nivoa neće izgubiti podatke.

## Fantomska čitanja

Fantomska čitanja se dešavaju kada za vreme izvršenja transakcije, druga transakcija, izmeni svoj rezultat dodavajući ili brišući redove koji pripadaju samom rezulatu. Primer toga bi bio sledeći:

* Transakcija A izvršava upit koji vraća sve korisnike, 1000 korisnika se vraća, ali transakcija nije commit-ovana.
* Transakcija B dodaje novog korisnika, te sada ima 1001 korisnik u bazi.
* Transakcija A izvršava ponovo upit koji vraća sve korinsike, i sada je 1001 korisnik vraćen. Transakcija A sada jeste commit-ovana.

Pod striktom nivou izolacije *serializable*, transakcija B bi bila blokirana u dodavanju novog korisnika, sve dok transakcija A ne commit-uje svoju transakciju. Ovo može dovesti do degradacije performansi, jer svaka operacija upisa, zahteva da read operacije commit-uju svoju transakciju.

## Non repeatable reads – Čitanja koja se ne ponavljaju

Non repeatable read se dešava, kada za vreme izvršenja transakcije, isti red se vrati dva puta, dok su vrednosti polja reda različite u obe transakcije. Primer ovakvog događaja je dat:

* Transakcija B čita da je trenutno stanje na računu korisnika 1, 50 dinara.
* Transakcija A menja stanje na računu korisnika 1, sa 50 na 100 dinara, i commit-uje transakciju
* Transakcija B čita stanje na računu korisnika 1 ponovo i dobija novu vrednost, 100 dinara, zatim commit-uje transakciju.

Problem prestavlja to što je transakcija B dobila drugačiju vrednost za vreme trajanja transakcije, jer je vrednost menjala transakcija A. To predstavlja problem jer transakcija B dobija dve različite vrednosti u sopstvenoj transakciji. Upravo zato je read commited nivo izolacije, koji ne sprečava čitanja koja se ne ponavljaju, je najkorišćeniji nivo izolacije koji se koristi.

## Dirty reads – Prljava čitanja

Prethodno pomenuti primer, je primer prljavog čitanja, gde se novac kreirao ni iz čega, je tipičan primer prljavih čitanja. Read uncommitted nivo izolacije ne štiti od prljavih čitanja, i zbog toga se retko koristi.

Sledeća tabela prikazuje izolacione nivoe, kao i potencijalne probleme [19].

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nivo izolacije** | **Izgubljene izmene** | **Prljava čitanja** | **Neponovljiva čitanja** | **Fantomska čitanja** |
| **Read uncommited** | Ne dešavaju se | Mogu se desiti | Mogu se desiti | Mogu se desiti |
| **Read commited** | Ne dešavaju se | Ne dešavaju se | Mogu se desiti | Mogu se desiti |
| **Repeatable read** | Ne dešavaju se | Ne dešavaju se | Ne dešavaju se | Mogu se desiti |
| **Serializable** | Ne dešavaju se | Ne dešavaju se | Ne dešavaju se | Ne dešavaju se |

*Tabela 1 – Izolacioni nivoi i potencijalni problemi*

## Izolacija kod MongoDB-a

Izolacija kod čitanja je određena postojanjem transakcija, read concern-om i write concern-om (u retkim slučajevima). U zavisnosti od read concern-a, klijenti mogu videti rezultate upisa pre samog perzistiranja upisa.

* Bez obzira na write concern, drugi klijenti koji koriste local ili avaliable read concern mogu da vide rezultat operacije upisa pre nego što je operacija upisa vratila potvrdu o upisu klijentu koji je izdao naredbu za upis.
* Klijenti koji koriste local ili avaliable read concern mogu da pročitaju podatke koji kasnije mogu biti roll-back-ovani.

Za operacije kod višedokumentnih transakcija, nakon commit-a transakcije, sve promene koje su posledice izvršenja transakcije, su sačuvane i vidljive van transakcije. Što znači da transakcija neće commit-ovati neke promene, a roll-back-ovati neke od svojih izmena. Dok se transakcija ne commit-uje, promene u podacima izazvane transakcijama, neće biti vidljive van transakcije. Ali kada transakcija vrši upis na više shard-ova, ne moraju sve operacije čitanja van same transakcije, da čekaju da se rezultat vidi kod svih shard-ova. Na primer, ukoliko je transakcija commit-ovana i upis 1 je vidljiv na shard-u A ali upis 2 nije vidljiv na shard-u B, čitanje sa readConcern-om lical može da pročita rezultate upisa 1 bez viđanja upisa 2. Read uncommited je difoltni nivo izolacije [20].

## Read uncommitted i atomičnost pojedinačnog dokumenta

Operacije upisa su atomične za pojedinačne dokumente, odnosno ukoliko upis menja više polja u dokumentu, operacija čitanja nikada neće videti samo neka izmenjena polja. Ali iako klijent neće videti delimično izmenjen dokument, read uncommitted znači da konkurentne operacije čitanja mogu videti izmenjeni dokument, pre nego što promene postanu trajne. Kod obične instance *mongod*, skup operacija čitanja i upisa u jedan dokument je serijalizovan (serializable). Sa skupom replika, skup operacija čitanja i upisa u jedan je serijalizovan samo u odsustvu rollback-a.

## Read uncommited i upis u više dokumenata

Kada jedna operacija upisa (primer updateMany) izmenjuje više dokumenata, modifikacija pojedinačnog dokumenta jeste atomična, ali cela operacija nije atomična. Prilikom izmene više dokumenata, bez obzira na to da li se izmena obavlja kroz jednu ili više operacija, druge operacije, se mogu umešati. Upravo u tim situacijama ukoliko je neophodna atomičnost operacija čitanja ili upisa, treba koristiti transakcije [20].

Bez izolacije višedokumentnih operacija upisa, MongoDB baza podataka se ponaša na sledeći način.

1. Operacije čitanja non-point-in-time. Ukoliko operacija čitanja započne u trenutku T1 i počne da čita dokument, operacija upisa zatim commit-uje upis u jedan od dokumenata u nekom kasnijem trenutku, T2. Osoba koja čita dokument može videti izmenjenu verziju dokumenta.
2. Operacije koje nije moguće serijalizovati. Ukoliko operacija čitanja počne sa čitanjem dokumenta D1, u trenutku T1, i operacija upisa izmeni dokument D1, u nekom kasnijem trenutku T3. Ovo izaziva zavisnost između čitanja i upisa, te ukoliko su operacije serijalizovane, operacija čitanja se mora izvršiti pre operacije upisa. Ali takođe ukoliko operacija izmeni dokument D2 u trenutku dT, te operacija čitanja, pročita dokument D2 u nekom trenutku 4, izaziva se write-read(upis-čitanje) zavisnost, koja zahteva da se čitanje izvrši nakon operacije upisa u serijalizovanom rasporedu. Postoji ciklust zavisnosti koji čini serijalizaciju nemogućom.
3. Operacije čitanja mogu propustiti neke dokumente koji odgovaraju traženim, i koji su izmenjeni za vreme izvršavanja operacije čitanja.

Read concern kontroliše konzistenciju i nivoe izolacije kod čitanja, definiše broj čvorova, kod setova replika koji su commit-ovali upis [20].

# Zaključavanje

MongoDB baza podataka koristi multigranularno zaključavanje, koje dozvoljava zaključavanje na globalnom nivou, na nivou baze ili kolekcije, i dozvoljava individualnim storage engine-ima da implementiraju svoju kontrolu konkurentnosti ispod nivoa kolekcije (kod WiredTriger-a, je na nivou dokumenata). MongoDB baza podataka koristi lock-ove kod upisa i čitanja koji dozvoljavaju konkurentnim čitanjima deljiv pristup resursu kao što je baza ili kolekcija. Osim shared (deljivog) moda za čitanje i ekskluzivnog moda zaključavanja za operacije upisa, intent shared (IS) i intent exclusive (IX) modovi, indiciraju ka nameri za korišćenje resursa na finijoj granularnosti lock-a. Na primer, ukoliko se zaključava kolekcija za upis (koriteći mod X), i odgovarajuća baza i globalni lock, moraju biti u IX mod-u. Baza može istovremeno da bude zaključana i u IS i u IX mod-u, ali X lock ne može da koegzistira ni sa jednim drugim modom, dok S lock može da koegzistira samo sa IS lock-om. Zaključavanja su fer, jer operacije čitanja i upisa bivaju smeštana u red. Kako bi se optimizovala propusnost, nakon što se jedan zahtev prihvati na obradu, svi kompatibilni zahtevi će biti prihvaćeni u isto vreme. Na primer ukoliko je X lock upravo otpušten i red za konflikte izgleda:

IS ->IS->X->X->S->IS

Pri striktnom FIFO uređenju, samo prva dva IS moda bi bila dopuštena. Umesto toga MongoDB će dozvoliti sve IS i S modove, i nakon njih prihvatiti X mod. N taj način nikada ne dolazi do izgladnjavanja. U okviru db.serverStatus() i db.currentOp() izlaza, modovi zaključavanja su predstavljeni na sledeći način:

* R – Shared (deljen) lock (S)
* W – Ekskluzivan lock (X)
* r – Intent shared lock (IS)
* w – Intent exclusive lock (IX)

Na sledećoj slici je prikazano koji mod zaključavanja primenjuju određene klijentske operacije kod storage engine-a koji primenjuju zaključavanje na nivou dokumenata.



*Slika 17 – Lista nekih operacija i tipova zaključavanja koje koriste*

## Koliko su granularna zaključavanja u MongoDB bazi podataka

Za većinu operacija upisa i čitanja, WiredTriger koristi optimističnu kontolu konkurencije. WiredTriger koristi samo *intent* zaključavanja na globalnom nivou, nivou baze i kolekcije. Kada storage engine uoči konflikt između dve operacije, jedna će dobiti konflikt pri upisu, što će izazvati MongoDB bazu podataka da ponovi operaciju. Postoje određene globalne operacije, najčešće kratke operacije koje uključuju više baza podataka koje zahtevaju učešće više baza podatka, koje zahtevaju globalno zaključavanje.

## Izbegavanje zaključavanja

Kada dve operacije pristupaju istom resursu, zbog zaključavanja drugi procesi će morati da čekaju na resurs. Čekanje za dobijanje lock-a nad resusom se može redukovati, i to koristeći metodu koja je prikazana na sledećoj slici.



*Slika 18 – Podešavanje parametra maxTransactionLockRequestTimeoutMillis*

Ovaj parametar definiše maksimalno vreme u milisekundama, koje višedokumentne transakcije treba da sačekaju da dobiju lock-ove, neophodne za operacije u okviru transakcije. Po difoltu čekaće 5 ms, sa tim da će konfigurisanje optimalne vrednosti znatno poboljšati zaključavanje.

## Kako pratiti zaključavanje?

Kod globalnih lock-ova, korišćenjem funkcije *db.serverStatus().globalLock*. Pri čemu je na slici prikazan rezultat izvršenja ove naredbe.



*Slika 19– Prikaz izvršenja funkcije db.serverStatus().globalLock*

Pri čemu ukoliko je vrednost za *totalTime* polje veća nego vreme rada baze podataka, baza je previše dugo u zaključanom stanju. Polje currentQueue je uvećano kada veliki broj zahteva čeka na puštanje lock-a.

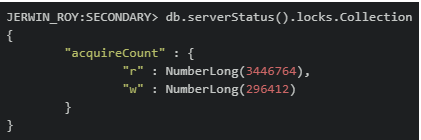
Ukoliko se prati zaključavanje na nivou baza podataka, praćenje se vrši korišćenjem naredbe *db.serverStatus().locks.Database.*



*Slika 20 - Prikaz izvršenja funkcije db.serverStatus().locks.Database*

Pri čemu *acquireCount* predstavlja broj puta koliko je zaključavanje bilo zatraženo, *acquireWaitCount* prikazuje koliko je puta zaključavanje izazvalo čekanje, zbog konfliktnih zaključavanja, a *timeAcquiringMicros* prikazuje ukupno vreme koje je provedeno u čekanju na dobijanje lock-a, u mikrosekundama.

Za prikaz informacija o zaključavanjima na nivou kolekcija koristi se naredba *db.serverStatus().locks.Collection,* čije je dejstvo prikazano na sledećoj slici.



*Slika 21 – Prikaz izvršenja funkcije db.serverStatus().locks.Collection*

# Zaključak

MongoDB baza podataka, sa dolaskom verzije 4.0, uvodi dugo očekivane transakcije, i jedna je od pionira u uvođenju transakcija u NoSQL baze podataka. Veliko je pitanje da li se to kosi sa osnovnim principima na kojima je MongoDB baza podataka izgrađena, kao i o sličnostima sa relacionim bazama podataka. Takođe podrška i dokumentacija za kreiranje i upravljanje transakcijama je jako mala, i jako često se naglašava manjak potrebe za korišćenjem transakcija u MongoDB bazi podataka, ali da podrška, kakva god da je postoji. Svakako ne sme se zanemariti veliki skok koji je načinila MongoDB baza podataka, ali stiče se utisak da ima dosta prostora za napredak na polju transakcija, svakako ne stiče se utisak da će u skorije vreme transakcione mogućnosti biti bilo gde približno nivoa relacionih baza podataka, što je i logično kada se u obzir uzme da je Oracle baza podataka uvela transakcione mogućnosti slične MongoDB bazi podataka, još davne 1988. godine [21]. Mehanizmi zaključavanja su jako slični onima koji se primenjuju u relacionim bazama podataka, dok se izolacija definiše na malo specifičniji način, koji je manje user-friendly u odnosu na neke druge relacione baze podataka. Ono što takođe treba pomenuti da su sva polja obrađena u ovom radu u konstantnom trenutno razvoju od strane MongoDB developera, te se update-i stalno moraju proveravati, i mora se proveravati stalno dokumentacija sa ciljem provere važenja dosadašnjih pravila.

# Literatura

1. Liberios Vokorokos, Matúš Uchnár, Ľubor Leščišin Performance optimization of applications based on non-relational databases - 2016 International Conference on Emerging eLearning Technologies and Applications (ICETA), ISBN: 978-1-5090-4701-7
2. What is MongoDB? Introduction, Architecture, Features & Example - <https://www.guru99.com/what-is-mongodb.html> (Pristup 10.05.2021)
3. MongoDB Atlas <https://cloud.mongodb.com/v2/60aa8aacfc078d52db124b4e#metrics/replicaSet/60aa8b92f0387d3bbc91b1c7/explorer/test/person/find>
4. Database transaction - <https://en.wikipedia.org/wiki/Database_transaction> (Pristup 10.05.2021)
5. An Overview of Multi-Document ACID Transactions in MongoDB and How to Use Them - <https://severalnines.com/database-blog/overview-multi-document-acid-transactions-mongodb-and-how-use-them> (Pristup 10.05.2021)
6. ACID - <https://sr.wikipedia.org/srec/ACID_(%D1%80%D0%B0%D1%87%D1%83%D0%BD%D0%B0%D1%80%D1%81%D1%82%D0%B2%D0%BE)> (Pristup 10.05.2021)
7. ACID - <https://en.wikipedia.org/wiki/ACID> (Pristup 10.05.2021)
8. ACID Transactions in MongoDB - <https://www.mongodb.com/basics/transactions>
9. 6 Signs You Might be Misunderstanding ACID Transactions in Distributed Databases - <https://medium.com/yugabyte/6-signs-you-misunderstand-acid-transactions-in-distributed-databases-43dcaba24485> (Pristup 10.05.2021)
10. The Most Popular Databases in 2020 - <https://learnsql.com/blog/most-popular-sql-databases-2020/> (Pristup 11.05.2021)
11. MongoDB Transactions vs Two-Phase commit -https://www.codementor.io/@christkv/mongodb-transactions-vs-two-phase-commit-u6blq7465 (Pristup 12.05.2021)
12. Production Considerations - <https://docs.mongodb.com/manual/core/transactions-production-consideration/#std-label-transactions-prod-consideration-outside-reads> (Pristup 12.05.2021)
13. Session.startTransaction()¶ - <https://docs.mongodb.com/manual/reference/method/Session.startTransaction/#mongodb-method-Session.startTransaction> (Pristup 14.05.2021)
14. Read Concern - <https://docs.mongodb.com/manual/reference/read-concern/#std-label-read-concern> (Pristup 14.05.2021)
15. Write Concern - <https://docs.mongodb.com/manual/reference/write-concern/#std-label-write-concern> (Pristup 14.05.2021)
16. Read Preference¶ - <https://docs.mongodb.com/manual/core/read-preference/#std-label-replica-set-read-preference> (Pristup 14.05.2021)
17. Transakcije - <https://docs.mongodb.com/manual/core/transactions/> (Pristup 14.05.2021)
18. MongoDB 4 Update: Multi-Document ACID Transactions - <https://www.mongodb.com/blog/post/mongodb-multi-document-acid-transactions-general-availability> (Pristup 17.05.2021)
19. Alex Giamas – Mastering MongoDB 4.x Second edition
20. Read Isolation, Consistency, and Recency - <https://docs.mongodb.com/manual/core/read-isolation-consistency-recency/#monotonic-writes> (Pristup 17.05.2021)
21. Limitations in MongoDB Transactions - <https://www.dbta.com/Columns/MongoDB-Matters/Limitations-in-MongoDB-Transactions-127057.aspx> (Pristup 20.05.2021)