

Univerzitet u Nišu

Elektronski fakultet

Seminarski rad

**Obrada upita kod MongoDB baze podataka**

Sistemi za upravljanje bazama podataka, 2020/2021.

Student:

Vladimir Janjić, 1283

Mentor:

Aleksandar Stanimirović

**Sadržaj**

[Uvod 3](#_Toc69327434)

[Baza podataka GameReviews 4](#_Toc69327435)

[MongoDB upitni jezik 5](#_Toc69327436)

[Upotreba indeksa kod obrade upita 11](#_Toc69327437)

[Obrada upita kod rasparčavanja i replikacije 16](#_Toc69327438)

[Zaključak 20](#_Toc69327439)

[Literatura 21](#_Toc69327440)

# Uvod

Bilo da se radi o relacionim SQL ili NoSQL bazama podataka, pribavljanje podataka iz baze podataka izdvaja se kao jedna od osnovnih operacija. Različite baze podataka implementirale su različite načine pristupa podacima – direktni pristup, pristup preko određenog upita, pristup na osnovu određene reči ili fraze, odn. termina. Prva dva pristupa se mogu naći kod svih baza, dok se treći obično implementira samo kod NoSQL baza podatka i baza podataka koje služe kao indeksi za pretragu, poznatiji kao „pretraživači“ (eng. *Search Engines*). Najbitniji i najčešće korišćen pristup podacima jeste zadavanjem upita u određenom upitom jeziku. U najvećem broju slučajeva to je SQL ili upitni jezik slične sintakse nastao iz njega (kod relacionih baza podataka), dok se kod NoSQL baza podataka izdvajaju svojstveni jezici koji mogu strukturno da se oslanjaju na redosled zadavanja upita SQL-a, a mogu biti i potpuno nezavisni.

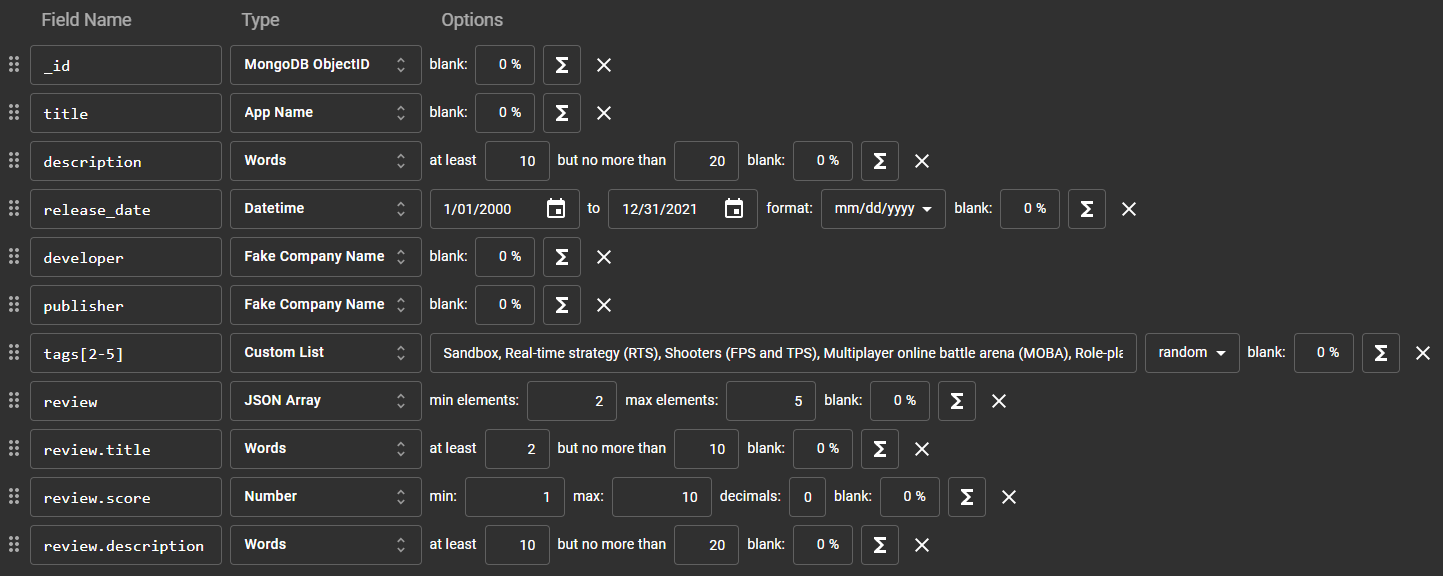
MongoDB predstavlja NoSQL bazu podataka koja se koristi prvenstveno za skladištenje polu-strukturiranih podataka u okviru dokumenta. Dokument predstavlja osnovnu jedinicu skladištenja i čuva se u obliku BSON formata, svojstvenog formata MongoDB baze podataka za binarnu reprezentaciju JSON formata dokumenata. Više dokumenata čini jednu kolekciju dokumenata nad kojom se, u najvećem broju slučajeva, zadaju i izvršavaju upiti. Za zadavanje upita koristi se specifičan MongoDB upitni jezik (eng. *MongoDB Query Language, MQL*), koji se svojom strukturom i operacijama oslanja na SQL upitni jezik. MQL sintaksa podseća na standardno pozivanje funkcija u programskim jezicima uz upotrebu JSON formata za prikaz objekata.

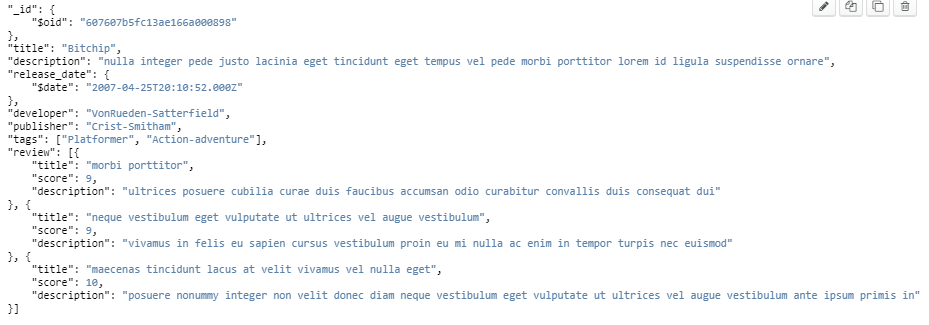
Obrada upita kod MongoDB baze podataka je složeni proces koji odpočinje zadavanjem upita prema bazi podataka, a okončava se prihvatanjem rezultata koji baza podataka vraća kao odgovor na upit. Redosled izvršavanja operacija prati određeni algoritam, međutim, pojedine faze se izvršavaju drugačije u zavisnosti od samog upita, strukture baze podataka, strukture dokumenata u okviru kolekcija, postojanja indeksa kao i različitih planova optimizacije koje baza podataka interno evaluira. Upiti se mogu keširati za određeni plan izvršenja, tako da se ponovnim zadavanjem upita skraćuje vreme dobijanja rezultata. Rezultat izvršenja upita predstavlja dokument, grupu dokumenata ili posebnu strukturu nazvanu „pokazivač“ (eng. *Cursor*), koja predstavlja privremenu kolekciju vraćenih dokumenata sa dodatnim opcijama (npr. iteracija primljenih dokumenata).

U nastavku rada će biti prikazani specifični slučajevi obrade upita. Biće posvećena paznja samom upitom jeziku i brojnim upitima od kojih zavisi način izvršenja upita. Takođe, biće prikazan uticaj indeksa na pretragu, u zavisnosti od njihovog tipa. Rasparčavanje i replikacija takođe mogu uticati na obradu upita. Na kraju, biće reči i o „mongod“ instanci servera kao i „mongos“ serveru za rutiranje upita.

# Baza podataka GameReviews

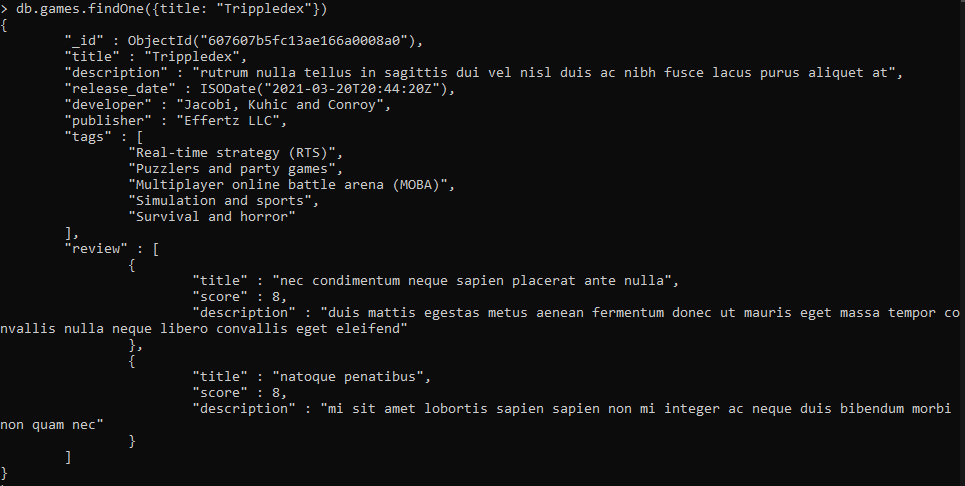
Kao baza podataka za demonstraciju, korišćena je baza koja se bavi recenzijama video igara, gde korisnici mogu da pregledaju igre i pišu recenzije. Sadrži kolekciju **games** koja sadrži opis video igara (naslov, opis, datum objavljivanja, ime razvojnog tima, ime izdavača, dva do pet tagova koji predstavljaju žanrove, kao i listu poddokumenata **review** koji predstavljaju recenzije sa svojim naslovom, ocenom i opisom). Za kreiranje „šeme“, korišćen je sajt Mockaroo kojim se ujedno i generišu podaci koji će biti skladišteni u bazi podataka.

Slika 1. Izgled „šeme“ kolekcije *games*

**Slika 2. Primer izgleda dokumenta u kolekciji

# MongoDB upitni jezik

Prvi korak u obradi upita jeste samo definisanje upita. Za to se koristi MongoDB upitni jezik MQL koji se svojom strukturom zasniva na SQL-u.

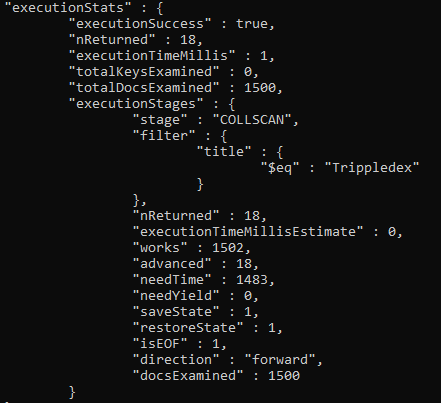
Osnovni upit predstavljaju funkcije **find** i **findOne**. Analogne su SELECT naredbi u SQL-u. Razlikuju se prema rezultatu koji vraćaju. Funkcija „find“ vraća pokazivač objekat koji sadrži više dokumenata i može da se iterira. Za razliku od nje, funkcija „findOne“ vraća samo jedan dokument, ako je pronađen. Ukoliko se upitom poklopi više dokumenata, vraća se samo prvi poklopljeni, koji je najčešće prvi po redosledu dodavanja u kolekciju. Definisanje termina upita se vrši navođenjem parova ključ-vrednost u okviru parametara funkcija. Ukoliko se navedu više parova, podrazumeva se korišćenje logičkog „I“ operatora, tj. svi parovi moraju biti sadržani u okviru istog dokumenta. Svi termini za pretragu su osetljivi na veličinu slova. Podržana je i upotreba posebne vrednosti „null“ koja predstavlja prazno polje.

Slika 3. Pronalaženje jednog dokumenta

Za pronalazenje jednog dokumenta, vrši se skeniranje dokumenata redom dok se ne dođe do prvog dokumenta koji zadovoljava upit. Ne mogu se koristiti neke od opcija podešavanja rezultata. Takođe, ne postoji pristup planu izvršenja.



Pokretanjem funkcije „find“ uz objašnjenje izvršenja, vidi se način obrade upita.



Slika 4. Uvid u izvršenje „find“ upita

Na slici 4. se može videti kako je obrađen upit. U ovom slučaju, iskorišćeno je skeniranje čitave kolekcije (faza označena sa „COLLSCAN“). Pri tome, ispitani su svi mogući dokumenti, njih 1500 (polje „docsExamined“), dok su samo 18 vraćena (polje „nReturned“).

Za upravljanje rezultatima u okviru pokazivač objekta, koriste se funkcije **skip**, **limit** i **sort**. One u kombinaciji omogućavaju straničenje rezultata. Korišćenje funkcija se vrši nadovezivanjem na funkciju „find“ (eng. *Method chaining*). Funkcijom „skip“ preskaču se prvih N rezultata navedenih kao parametar funkcije, funkcija „limit“ ograničava broj rezultata koji se vraćaju u pokazivač objektu, dok funkcija „sort“ uređuje rezultate po rastućem ili opadajućem redosledu na osnovu vrednosti nekog polja u okviru dokumenta. Redosled navođenja funkcija nije bitan. Izvršenje ovih funkcija je sporo i preporučuje se korišćenje indeksa.

Izvršeno je pokretanje istog upita uz korišćenje svih funkcija. Sortiranje se obavlja po naslovu, a takođe se vrši i straničenje gde se preskaču prvih 5 rezultata i prikazuju naredna 5.



Slika 5. Obrada upita korišćenjem sortiranja i straničenja

Na slici 5. se može videti redosled izvršenja operacija kod upita. Prvo se vrši skeniranje kolekcije i prikupljanje svih dokumenata koji zadovoljavaju upit. Zatim se od prvobitnih 18 pribavljenih dokumenata dalje obrađuju samo njih 10, kao zbir parametara za „skip“ i „limit“. Vrši se sortiranje u memoriji po navedenom ključu. Nakon toga sledi preskakanje 5 dokumenata tako da na kraju budu vraćena samo 5.

Iz ovog primera se primećuje zbog čega je preskakanje dokumenata loše. Da bi se izvršilo preskakanje dokumenata, potrebno ih je pribaviti, iako se oni na kraju ne prikazuju korisniku. Upit koji prikazuje 5 dokumenata nakon 10000 preskočenih znači pribavljanje 10005 dokumenata. Ako se uzme u obzir straničenje narednih 10 strana, to je više od 100000 bespotrebno pribavljenih dokumenata.

U okviru upitnog jezika, podržano je korišćenje i regularnih izraza. Oni omogućavaju poklapanje dokumenata ukoliko je samo deo termina, koji se pretražuje, poznat. Još jedna opcija za pretraživanje je navođenjem opsega vrednosti polja. Podržane su tekstualne verzije najčešće korišćenih operatora: **$lt** (less than, manje od), **$gt** (greater than, veće od), **$lte** (manje ili jednako) i **$gte** (veće ili jednako). Navodi se ime polja i vredonosti opsega, pri tom se vodi računa da vrednosti koje se upoređuju budu istog tipa.

Primer izvršenja funkcije koja pronalazi datum između dveju navedenih vrednosti.

Slika 6. Obrada upita sa opsegom

Pre nego što odpočne sa obradom upita, potrebna je interna konverzija datuma prema standardu koji koristi MongoDB. Nakon toga je izvršenje isto kao kod osnovnog upita.

Izvršenje ostalih upita u nastavku ne utiče previše na izveštaj izvršenja „executionStats“, tako da će biti samo izložene ostale mogućnosti jezika, uz navođenje pojedinosti koje mogu da utiču na izvršenje upita.

Posebnu vrstu operatora predstavljaju „set“ operatori koji obuhvataju: **$in**, **$all** i **$nin**. Služe za navođenje termina, koji se pretražuju, u obliku listi. Navode se u okviru jednog polja. Operator „in“ služi kao logičko „ILI“ – vraća dokumente koji se poklapaju sa bilo kojim od termina, operator „nin“ je suprotan njemu – vraća sve dokumente koji se ne poklapaju ni sa jednom od termina, dok operator „all“ vraća dokumente koji se poklapaju sa svim terminima. Treba istaći da operatori „in“ i „all“ mogu da koriste indekse za pretragu, dok „nin“ ne može, već zahteva skeniranje čitave kolekcije. Zbog toga se preportučuje kombinacija sa drugim operatorima koji koriste indeks, ili predefinisanje upita.

Za definisanje kompleksnijih upita, dostupni su logički operatori:

* **$ne** – poklapanje ako argumenat nije jednak sa terminom
* **$not** – invertuje rezultate pretrage
* **$or** – poklapanje ako je bilo koji par ključ-vrednost istinit
* **$nor** – poklapanje ako ni jedan od parova nije istinit
* **$and** – poklapanje ako su svi parovi istiniti
* **$exists** – poklapanje ako termin postoji u dokumentu

Operator „ne“, slično operatoru „nin“, ne može da se koristi sa indeksima. Razlika je u tome što se „ne“ koristi za jednu vrednost. Operator „not“ je koristan, iako većina operatora ima negativnu formu, zbog toga što uključuje dokumente koji nisu evaluirani upitom (npr. ako ne postoji polje u okviru dokumenta koje je navedeno kao ključ iz para). Operator „or“ se razlikuje od operatora „in“ zbog toga što se primenjuje nad više parova ključ-vrednost. Korišćenje operatora „and“ u najvećem broju slučajeva nije potrebno jer je podrazumevani operator ukoliko se navede više parova ključ-vrednost. Najviše eksplicitnih primena ima u kombinaciji sa drugim logičkim i set operatorima.

Zbog toga što MongoDB nema fiksnu šemu, operator „exists“ se koristi da se proveri postojanje nekog polja (ključa) u okviru dokumenta. Zadavanjem upita može da se proveri postojanje, odn. nepostojanje ključa, navođenjem „true“ ili „false“ kao vrednosti operatora. Treba napomenuti da se vrši provera postojanja samog ključa, a ne da li je vrednost ključa postojana ili je „null“.

Dokumenti MongoDB baze podataka često u sebi mogu sadržati ugnježdene dokumente. Ti dokumenti mogu da imaju svoje parove ključ-vrednost, pa i same ugnježdene dokumente. Za pristup ugnježdenim ključevima, koristi se operator tačka „.“ za svaki nivo ugnježdenosti. Na taj način je omogućeno poklapanje poddokumenata na osnovu vrednosti ključa. Međutim, moguće je i poklapanje čitavih poddokumenata navođenjem kompozitnog ključa sa samim ključevima poddokumenta. Pri tom treba voditi računa da se vrednosti ključeva navode u istom redosledu kao pri kreiranju dokumenata zbog toga što se vrši stiktna pretraga bajt-po-bajt. Ukoliko nije poznato da li se čuva redosled ključeva, bolje je predpostaviti pri pisanju upita da redosled neće biti očuvan.

Ključevi u okviru dokumenata mogu da imaju nizove različitih tipova kao vrednost. Za pretragu nizova koriste se operatori **$elemMatch** i **$size**. Operator „elemMatch“ vrši poklapanje ako su svi navedeni termini u okviru istog poddokumenta, dok „size“ vrši poklapanje ako je dužina poddokumenta iste veličine kao navedena vrednost. Nalaženje jednog termina iz niza se svodi na definisanje upita preko para ključ-vrednost. Ukoliko se zahteva postojanje termina na određenoj poziciji, moguće je korišćenje tačka operatora uz navođenje pozicije. Poboljšanje performansi upita se dobija kreiranjem indeksa za ugnježdena polja. Izuzetak je operator „size“ koji ne može da koristi indekse, niti navoženje raspona vrednosti. Ukoliko je ova opcija potreba, moguće je definisanje posebnog ključa koji će pratiti veličinu niza.

Ako upiti ne mogu biti predstavljeni prethodno navedenim operatorima, postoji mogućnost korišćenja JavaScript koda operatorom **$where** koji kao vrednost ima JavaScript kod. Kad god je to moguće, trebalo bi se držati ugrađenih operatora zbog toga što JavaScript kod nema mogućnost korišćenja indeksa i izvršava se znatno sporije od istovetnih operatora. Poboljšanje se ogleda u kombinaciji sa upitom koji koristi indeks, tako da se smanjuje broj dokumenta koji se moraju skenirati. Upotreba JavaScript koda je nepogodna i zbog malicioznih napada jer je moguće izvršiti bilo koji kod u okviru upita.

Regularni izrazi predstavljaju moćnu mogućnost pri definisanju termina upita. MongoDB koristi PCRE (Perl Compatible Regular Expressions), koji podžava različite oblike regularnih izraza. Korišćenje regularnih izraza se vrši preko operatora **$regex** ili korišćenjem posebne sintakse. Osim prefiksne pretrage, regularne izraze nije moguće koristiti sa indeksima, tako da ih ne treba previše koristiti. Ne treba zaboraviti da su pretrage osetljive na veličinu slova. Upotrebom flega „i“, omogućuje se pretraga nezavisno od veličine slova, ali time i onemogućava korišćenje indeksa. Regularni izrazi su dosta spori jer je potrebno evaluirati sve moguće kombinacije simbola, pa je zbog toga poželjno što preciznije definisati upit.

Svaki od upita zahteva termin koji se pretražuje. Ako je ostavljen prazan, podrazumeva se prazan termin „{ }” koji vrši poklapanje svih dokumenata. Za svaki od upita, postoje dodatne opcije koje modifikuju odgovor.

Projekcije predstavljaju mogućnost vraćanja samo dela objekta navođenjem ključeva koje je potrebno vratiti, odn. ključeva koji nisu potrebni. Ukoliko nije eksplicitno navedeno, identifikator dokumenta se uvek vraća. Dodatno, moguće je vratiti deo niza vrednosti korišćenjem operatora **$slice** od početka niza, sa kraja ili iz sredine. Za pojedinosti funkcije „sort“ treba istaći mogućnost sortiranja prema više ključeva, gde se uzima u obzir redosled navođenja.

# Upotreba indeksa kod obrade upita

Indeks predstavlja strukturu koja služi za pamćenje dodatnih informacija o dokumentima da bi se omogućilo bolje i brže izvršavanje upita. Formiranje indeksa se vrši na nivou kolekcije za jedan od ključeva dokumenta, tako da indeks sadrži sve vrednosti koje ključ može da ima u paru sa dokumentom u kojem se određena vrednost nalazi. U najvećem broju slučajeva se kod obrade upita koristi samo jedan indeks, iako indeks može postojati za više ključeva. Postoji mogućnost preseka indeksa – vrši se pretraga po jednom indeksu, pa po drugom, a zatim se vrši presek dobijenih dokumenata. Ponašanje zavisi od konkretnog upita, ali optimizator upita cilja na upotrebu indeksa koji će najviše smanjiti vreme pretrage.

Podržano je više tipova indeksa. U najvećem broju slučajeva radi se o indeksu sa jednim ključem (eng. *Single-key index*). Jedna stavka predstavlja jednu vrednost ključa iz svakog indeksiranog dokumenta, čak i ako je vrednost prazna. Primer je indeks koji se uvek kreira za polje identifikatora dokumenta „\_id“. Ako je potrebna česta pretraga po više indeksa, postoji opcija kreiranja složenog indeksa (eng. *Compound-key index*). Svaka stavka indeksa predstavlja kombinaciju vrednosti dva ili više ključeva i dokumenta u kojem se nalaze. Efikasnost složenih ključeva zavisi od redosleda i uređenosti. U slučaju da je potrebno pronaći opseg dokumenata, kod dobro uređenog indeksa je dovoljno pronaći prvu stavku i očitati ostale, dok je kod loše uređenog potrebno pronaći svaku posebno.

Pokretanjem istog upita sa početka, na slici 7 može se videti bitna razlika u izvršenju. Kreiran je indeks sa jednim ključem na polju „title“. Kao plan izvršenja, izabrano je čitanje indeksa (faza „IXSCAN“). Može se primetiti da su iz indeksa očitane samo 18 stavke, koje odgovaraju broju pronađenih dokumenata koji su poklopljeni upitom. Kako je indeks uređen, dovoljno je bilo pronaći samo prvu stavku, a zatim očitati ostale sekvencijalnim redosledom (unapred, označeno sa „direction: forward“). Sledeća faza je faza pribavljanja dokumenata (faza „FETCH“). Dovoljno je skenirati samo 18 dokumenata koji su pronađeni u indeksu i time se vreme izvršenja upita smanjuje oko pedeset puta u konkretnom slučaju. Treba napomenuti da se kao granice indeksa navodi ista stavka („Trippledex“). To je slučaj kod poklapanja po jednakosti, dok će kod poklapanja po rasponu, granice biti zamenjene minimalnom i maksimalnom vrednošću. Kod regularnih izraza, granice predstavljaju prvu i poslednju kombinaciju koja se pretražuje.

Slika 7. Korišćenje indeksa u upitu.

Postojanje indeksa je važno zbog toga što je u suprotnom potrebno skenirati čitavu kolekciju dok uslov upita ne bude ispunjen. Kao što je napomenuto, MongoDB će koristiti samo jedan ključ pri upitu, osim ako se radi o korišćenju „or“ operatora. Tada je moguće upotrebiti po jedan indeks za svaki od termina koji se pretražuju u okviru operatora. Ukoliko je potrebno koristiti kombinaciju ključeva, moguće je definisati složeni indeks. Pri tom treba voditi računa o redosledu ključeva. Postojanje posebnih indeksa za A i B, kao i složenog A-B, predstavlja slučaj kada je indeks A redundantan, dok B nije.

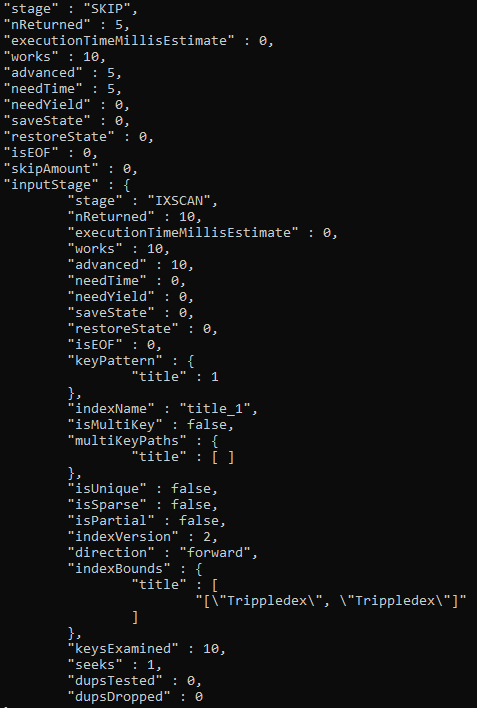
Na slici 8 prikazano je uporedo korišćenje više indeksa operatorom „or“. Prethodno je kreiran indeks nad ključem „publisher“. Vrši se čitanje iz oba indeksa i rezultati šalju dalje na upoređivanje. U okviru „OR“ faze, odbacuju se duplikati, odn. dokumenti koji su pronađeni u oba indeksa. Pokretanjem upita, pronađeno je 18 dokumenata iz indeksa „title“ i dva iz indeksa „publisher“, od kojih je jedan duplikat, tako da je odbačen. Preostalih 19 dokumenata se skenira i vraća kao odgovor.

Slika 8. Upotreba više indeksa u istom upitu korišćenjem „or“ operatora

Interno, MongoDB čuva indekse kao binarna stabla. Ovakva stuktura je pogodna za različite vrste upita: potpuna poklapanja, raspon vrednosti, sortiranje, poklapanje prefiksa kao i upite koji koriste samo indeks. Svaki od čvorova može da sadrži više stavki, od kojih svaka sadrži pokazivač prema dokumentu i prema čvoru deteta. Dodatno, svaki čvor pokazuje prema drugom čvoru koji ima vrednosti manje nego najmanja stavka.

Drugi značajan tip indeksa za pretragu predstavljaju heš indeksi. Svaka vrednost pre upisa u indeks prolazi kroz heš funkciju. Ovim je izgubljen redosled dokumenata, tako da identično poklapanje i dalje radi, ali nije moguće poklapanje po opsegu. Značajno kod rasparčavanja.

Određivanje koji će indeks biti upotrebljen za pretragu se vrši na nivou planera upita. U ovom koraku se vrši uporedo izvršavanje više planova obrade upita i bira onaj koji skenira najmanje dokumenata. Ukoliko su indeksi loše definisani, kao jedan od planova predstavlja potpuno skeniranje kolekcije. Nakon odabira optimalnog plana, vrši se keširanje, tako da se u sledećem ponovnom pokretanju upita izvršava unapred poznat plan. Keš ističe pri dodavanju i uklanjanju indeksa.

Kod složenih upita, moguće je izvršenje upita koji u sebi sadrži pretragu termina koji se nalaze u indeksu, i nekog uslova koji uključuje polje koje nije indeksirano. Tada se korišćenje indeksa upotrebljava kako bi se što više smanji opseg pretrage, dok je za ostatak uslova potrebno skeniranje dokumenata.

Slike 9 i 10. Korišćenje indeksa kod „sort“, „limit“ i „skip“ funkcija.

Na slikama 9 i 10 se može primetiti razlika u izvršenju prvobitnog upita kada se koristi indeks kod funkcija za uređivanje i straničenje. Korišćenjem indeksa se smanjuje broj skeniranih dokumenata i rešava problem sa preskakanjem. Preskakanje dokumenata se vrši odmah nakon prolaska kroz indeks, tako da se pribavljanje i skeniranje dokumenata svodi na broj koji će biti prikazan na kraju (očitano 10 stavki iz indeksa, skenirano samo 5 dokumenata).

Slika 11. Pribavljanje dokumenata pokrivenih indeksom

Poseban slučaj korišćenja indeksa sastoji se iz pribavljanja samo onih ključeva koji su pokrivene indeksom, što se može videti na slici 11. To se postiže projekcijom navođenjem onih ključeva koji su indeksirani, uz isključivanje podrazumevanog ključa „\_id“. Time se u potpunosti preskače potreba za skeniranjem dokumenata, već se čitav sadržaj, koji se prikazuje, pribavlja direktno iz indeksa (faza „PROJECTION\_COVERED“).

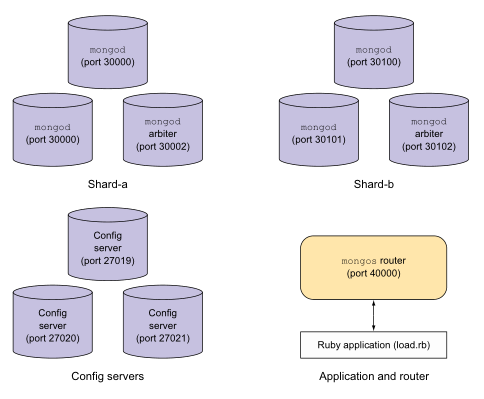
# Obrada upita kod rasparčavanja i replikacije

Rasparčavanje (eng. *Sharding*) predstavlja proces podele baze podataka na manje delove gde svaki deo predstavlja jedan odlomak (eng. *Shard*). Jedan od razloga za uvođenje podele jeste raspored kolekcije na više uređaja koji su ograničeni svojim resursima, tj. horizontalno skaliranje. Kao još jedan razlog za uvođenje može biti balansiranje upita koji se izvršavaju drugačije kod prisustva rasparčavanja.

Replikacija je proces kopiranja kolekcije, odn. odlomka prvenstveno u cilju postojanja redundantnosti među podacima. U kontekstu upita, takođe se u određenim slučajevima može koristiti za raspored opterećenja. Primarni odlomak i njegove replikacije zajedno čine jedinicu replikacionog seta. Postoji više modova za upravljanjem čitanja nad replikacionim setom:

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

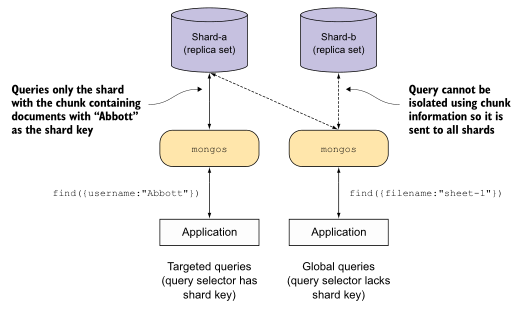
Kod MongoDB baze podataka, rasparčavanje se najčešće vrši na nivou kolekcije. Kako se upiti izvršavaju na nivou kolekcija, javlja se izmenjeno ponašanje rada baze podataka. Sa stanovišta aplikacije, slanje upita prema čitavoj kolekciji ili prema jednom odlomku je istovetno, međutim, javlja se razlika u internoj obradi upita.

Slika 12. Izgled arhitekture rasparčanog klastera

Kod jednostavne arhitekture MongoDB sistema, bitne elemente predstavljaju instanca baze, odn. odlomka „mongod“ i server za rutiranje „mongos“. Instanca servera ima ulogu u skladištenju podataka i učestvuje u čitanju i pisanju dokumenata. Server za rutiranje odlučuje koje će instance učestvovati u obradi upita na osnovu strukture i parametara upita. Služi kao jedinstvena tačka pristupa klasteru i upravlja operacijama čitanja, pisanja i upravljanja. Omogućuje jedinstven pogled prema klasteru, nezavisno od postojanja odlomaka. Nije potrebno pokretanje ako se ne koristi rasparčavanje i replikacija.

Postoje dva tipa upita sa stanovišta rutiranja. Prvi od njih je ciljani upit (eng. *Targeted query*). Svaki od odlomaka je definisan svojim ključem odlomka (eng. *Shard key*) koji određuje koji deo kolekcije se čuva na njemu. Server za rutiranje poseduje informacije o tim ključevima. Ako se u okviru upita navede ključ odlomka, „mongos“ zna na koji se deo kolekcije upit odnosi i obavlja prosleđivanje samo prema jednom odlomku čiji je ključ naveden naveden u upitu.

Ukoliko se ne navede ključ odlomka, prosleđivanje upita se vrši prema svim odlomcima da bi se upit obslužio. Ovakav tip upita se naziva globalni ili „rasej/sakupi“ (eng. *Scatter/gather*). Ovakav tip upita je loš po performanse klastera i treba ih izbegavati.

Slika 13. Ciljni i globalni upiti kod rasparčavanja

Promena u ponašanju ističe se i kod „sort“, „limit“ i „skip“ funkcija. Ako se koristi sortiranje i uključuje ključ odlomka, server za rutiranje može odrediti kojim redosledom da izvršava upit nad odlomcima. Ako upit ne sadrži kriterijum za uređivanje, „mongos“ server prikuplja podatke sa odlomaka po principu „round robin“. Ograničavanje rezultata se vrši u dva stepena, prvo lokalno na svakom od odlomaka, zatim još jednom na nivou „mongos“ servera, zbog toga što se mogu javiti rezultati sa više odlomaka. Preskakanje rezultata se može obaviti samo na „mongos“ serveru nakon što se prikupe svi rezultati.

Još jedna od pogodnosti u obradi upita jeste i postojanje indeksa za svaki odlomak zasebno. Na taj način pristup jednom odlomku sa indeksom znači pristup samo delu indeksa koji obuhvata taj deo kolekcije, čime se smanjuje vreme odziva. Da bi se upotrebio indeks, potrebno je da upit sadrži ključ odlomka. Moguće je proveriti koliko i koji od odlomaka učestvuju u obradi upita i da li se koristi indeks.

Slika 14. Izvršenje ciljnog upita

Slika 15. Izvršenje globalnog upita

Na slici 14 dat je prikaz iz knjige koji obrađuje ciljni upit. Faza „SHARDING\_FILTER“ govori o tome da se obrađuju dokumenti iz jednog odlomka korišćenjem indeksa, nakon čega se vrši pribavljanje poklopljenih dokumenata.

Na slici 15 dat je prikaz iz knjige koji obrađuje globani upit. Postoje dva odlomka nad kojima se izvršava upit. Pošto ne koriste ključ odlomka, vrši se skeniranje svih dokumenata koji pripadaju odlomcima („COLLSCAN“ u okviru „SHARDING\_FILTER“). Nakon toga se rezultati kombinuju u fazi „SHARD\_MERGE“.

# Zaključak

MongoDB baza podataka podržava veliki broj opcija za zadavanje upita. Stuktura baze podataka kao i struktura samih upita, utiču na to kako će teći obrada upita.

Postupak obrade upita kreće zadavanjem i slanjem upita iz aplikacije. Upit dospeva do „mongos“ servera za rutiranje (ako je omogućeno rasparčavanje i replikacija). Proverava se ispravnost upita – sintaksno i semantički – i vraća se greška ukoliko upit nije ispravno napisan. Ukoliko jeste, vrši se parsiranje i određivanje sadržaja upita, odn. evaluacija (npr. određivanje finalnog upita na osnovu složenog upita sa relacionim operatorima). Nakon parsiranja, određuje se prostor pretrage (eng. *Namespace*) koji sadrži ime baze i kolekciju nad kojom se izvršava upit.

Ukoliko plan izvršavanja nije keširan, odpočinje rad planera upita. Predlažu se planovi na osnovu postojanja indeksa za određene ključeve. Indeksi se mogu obilaziti unapred ili unazad. Kao jedan od planova uvek postoji i skeniranje kolekcije, koji se izvršava ako ne postoje indeksi, ili su performanse indeksa nezadovoljavajuće. Kada su određeni mogući planovi za izvršenje, planer pokreće sve planove istovremeno i ostavlja ih da rade probni period (eng. *Trial period*). Nakon završetka, vrši se izbor onog koji je najbrže vratio rezultate, ili deo rezultata. Vrši se keširanje plana koji će se u buduće koristiti za upite istog oblika. Sledi pokretanje najboljeg plana.

Ukoliko se radi o obradi upita bez indeksa, skenira se čitava kolekcija dok se ne prikupe svi podaci koji zadovoljavaju upit, što znači skeniranje svih odlomaka ako postoje. Ovo takođe podrazumeva da nije naveden ključ odlomka, tako da će se sortiranje podataka izvršiti lokalno pre nego što budu predati ruteru. Ako je izabran plan sa indeksom, vrši se prolazak kroz indeks u potrazi za relevantnim stavkama. To takođe znači da će se upit izvršiti samo na određenim odlomcima i to u određenom redosledu, ako je naveden. Nakon očitanog indeksa, opciono se preskaču dokumenti, a potom se vrši skeniranje pronađenih dokumenata kako bi se pribavio čitav sadržaj, osim ako se radi o upitima koji pokrivaju indekse (vraća se samo indeksirani ključ). Pre nego što podaci budu poslati ruteru, vrši se limitiranje rezultata, ako je navedeno u okviru upita.

Ruter zatim vrši prikupljanje podataka u zadatom redosledu (ako postoji u upitu) i još jednom vrši limitiranje podataka. Pre izvršenja limitiranja, dokumenti se modifikuju prema parametrima iz projekcije, tj. odstranjuju se ključevi koji nisu potrebni. Na kraju, vrši se slanje pronađenih dokumenata u obliku „pokazivač“ objekta kao privremene kolekcije.

# Literatura

1. Knjiga - MongoDB 4 Quick Start Guide Learn the skills you need to work with the world’s most popular NoSQL database by Doug Bierer
2. Knjiga - MongoDB in Action Covers MongoDB version 3.0 by Kyle Banker, Peter Bakkum, Shaun Verch, Doug Garrett, Tim Hawkins
3. Knjiga - Mastering MongoDB 4.x expert techniques to run high-volume and fault-tolerant database solutions using MongoDB 4.x by Alex Giamas
4. Knjiga - MongoDB The Definitive Guide by Kristina Chodorow, Michael Dirolf
5. Knjiga - MongoDB The Definitive Guide Powerful and Scalable Data Storage 3rd Edition by Kristina Chodorow
6. <https://blog.exploratory.io/an-introduction-to-mongodb-query-for-beginners-bd463319aa4c>
7. <https://www.oreilly.com/library/view/mongodb-the-definitive/9781449344795/ch04.html>
8. <https://docs.mongodb.com/manual/faq/fundamentals/>
9. <https://docs.mongodb.com/manual/reference/explain-results/>
10. <https://docs.mongodb.com/manual/core/query-plans/>
11. <https://docs.mongodb.com/manual/core/query-optimization/>
12. <https://docs.mongodb.com/manual/tutorial/analyze-query-plan/>
13. <https://docs.mongodb.com/manual/core/write-performance/>
14. <https://www.mockaroo.com/>