**Univerzitet u Nišu**

**Elektronski fakultet**

**Katedra za računarstvo**

**Seminarski rad**

Interna stuktura indeksa i organizacija indeksa kod Mongo baze podataka

**Mentor: Student:**  
**Doc. dr. Aleksandar Stanimirović Emilija Bićanin 1474**

**Niš, 2023**

Sadržaj

[1. Uvod 2](#_Toc1420877849)

[2. Indeksi-teorijski pregled 3](#_Toc434317361)

[2.1 Pojam indeksa u bazi podataka 3](#_Toc1245738153)

[2.2 Klasifikacija indeksa 4](#_Toc1397075959)

[2.3 Pregled različitih implementacija indeksa 6](#_Toc1666274752)

[2.3.1 Heš indeks 7](#_Toc310899818)

[2.3.2 B-stablo 7](#_Toc1550542763)

[2.3.3 B+ stablo 8](#_Toc1444411487)

[2.3.4 Bitmap indeks 9](#_Toc418892496)

[2.3.5 Invertovani indeks 10](#_Toc1875349085)

[2.3.6 R-stablo 11](#_Toc708306384)

[3. Indeksi u MongoDB bazi podataka 12](#_Toc708385756)

[3.1 MongoDB 12](#_Toc1797453924)

[3.2 Indeksi u MongoDB-u 13](#_Toc1147798791)

[3.3 Vrste indeksa u MongoDB-u 14](#_Toc185396781)

[3.3.1 Indeks nad jednim poljem (engl. Single Field Index) 14](#_Toc1110033064)

[3.3.2 Složeni indeks (engl. Compound Index) 15](#_Toc1745421541)

[3.3.3 Multikey indeks 16](#_Toc1801571587)

[3.3.4 Geoprostorni indeks (engl. Geospatial Index) 16](#_Toc701642045)

[3.3.5 Indeks za pretragu teksta 16](#_Toc694047988)

[3.3.6 Heširani indeks 17](#_Toc1114411550)

[3.3.7 Wildcard indeks 17](#_Toc216301518)

[3.4 Karakteristike indeksa u MongoDB-u 18](#_Toc629875515)

[3.5 Strategije indeksiranja 19](#_Toc90004619)

[4. Praktična demonstracija indeksa u MongoDB-u 22](#_Toc80587159)

[4.1 Kreiranje single key, multikey i compound indeksa, brisanje indeksa 22](#_Toc1282371339)

[4.2 Tekstualni indeks 24](#_Toc1942826556)

[4.3 Geoprostorni 2dsphere indeks 25](#_Toc1191431942)

[5. Zaključak 27](#_Toc1055026746)

[6. Literatura 28](#_Toc448612589)

# Uvod

Indeks predstavlja strukturu podataka koja omogućava brži pristup podacima u bazi podataka. Indeksi mogu značajno poboljšati performanse upita baze podataka tako što smanjuju količinu vremena potrebnog za traženje i preuzimanje podataka. Međutim, oni takođe mogu imati neke nedostatke, kao što su povećani zahtevi za skladištenje i sporije performanse operacija upisa, pošto svaka operacija umetanja, ažuriranja ili brisanja takođe mora da ažurira odgovarajući indeks. Zbog toga je bitno pažljivo kreirati indekse i posedovati dobro poznavanje podataka nad kojima se kreira indeks i upita koji će biti izvršeni nad tim podacima.

U prvom delu rada biće detaljno opisani pojam indeksa u bazama podataka, koje sve različite vrste indeks postoje i različite strukture podataka pomoću kojih se indeksi implementiraju.

Drugi deo rada se bavi indeksima u konkretnoj bazi podataka: MongoDB. Ova baza je izabrana jer je veoma popularna i ima dobru podršku za rad sa indeksima. U MongoDB-ju postoji veliki broj različitih vrsta indeksa, pogodnih za pretragu različitih vrsta podataka kao što su tekstualni, prostorni, ugnježdeni dokumenti, itd. Međutim, treba biti pažljiv sa kreiranjem indeksa i treba pažljivo izabrati koje indekse je neophodno kreirati. Zato su ovom delu rada, pored mogućnosti koje MongoDB pruža sa indeksima, opisane i najbolje strategije kojima se treba voditi prilikom kreiranja indeksa.

U poslednjem delu seminarskog rada prikazani su primeri prakticne primene indeksa u MongoDB bazi. Prikazane su naredbe za kreiranje različitih vrsta indeksa, brisanje indeksa, kao i upiti koji koriste kreirane indekse.

# Indeksi-teorijski pregled

## 2.1 Pojam indeksa u bazi podataka

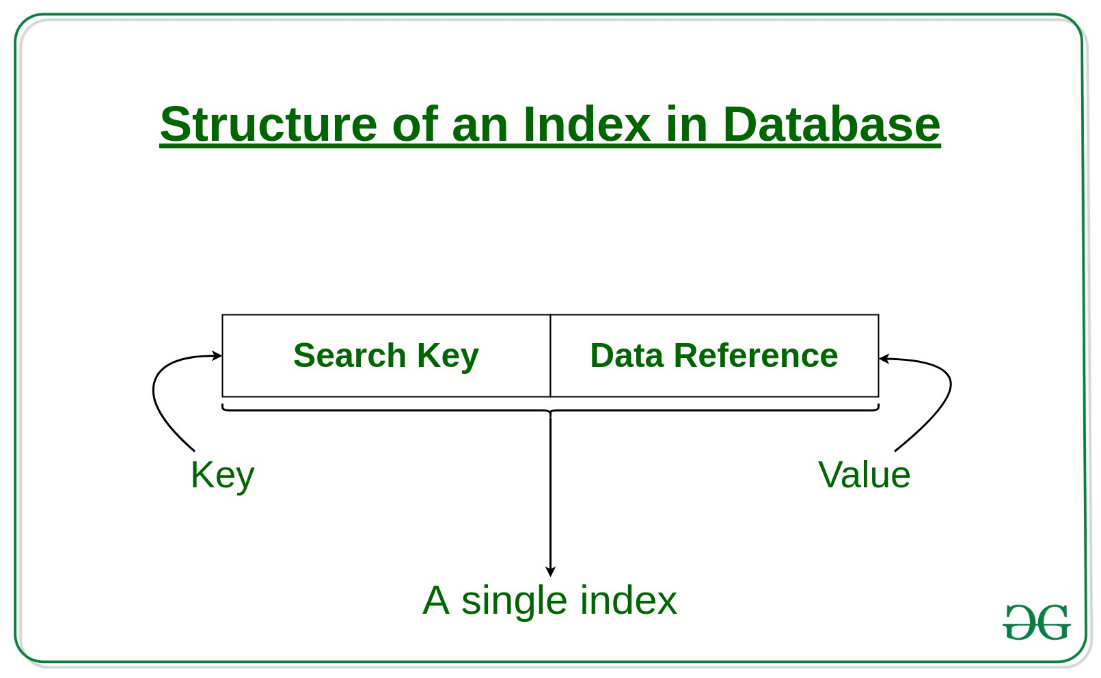
Indeks u bazi podataka predstavlja pomoćnu strukturu za ubrzan pristup podacima. On minimizira broj pristupa disku prilikom obrađivanja upita. Da bi se pokazala prednost koju indeksi pružaju, možemo koristiti analogiju sa indeksima u udžbenicima. Ukoliko u udžbeniku želimo da pronađemo određenu reč ili frazu, možemo krenuti od prve strane i čitati redom knjigu sve dok ne pronađemo željenu reč. Međutim, daleko brže ćemo naći ono što tražimo ukoliko pogledamo indeks na kraju knjige i odmah pročitamo na kojim se stranama nalazi željena reč. Istu ulogu imaju i indeksi kod baza podataka, omogućavaju nam da direktno u što manjem broju koraka dođemo do željenih podataka, umesto da tražimo sekvencijalno. Zbog toga je kreiranje odgovarajućih indeksa u ogromnim skupovima podataka možda najvažniji korak u optimizaciji baza podataka.

Međutim, iako su indeksi jedna od najvažnijih struktura za poboljšanje performansi baza podataka, neophodno je napomenuti i troškove koje oni sa sobom donose, jer kreiranje prevelikog broja indeksa može imati posledice. Indeksi poboljšavaju operacije pretraga podataka ali po ceni sporijih operacija upisa, ažuriranja i brisanja podataka. Takođe, svaki indeks zauzima određeni prostor na disku.

Indeksi se kreiraju korišćenjem nekoliko podataka iz baze podataka. Jedan element indeksa sastoji se iz 2 osnovna dela:

* Prvi deo predstavlja **ključ za pretragu** (engl. *search key*) koji ustvari predstavlja vrednosti kolone iz baze podataka po kojoj želimo da vršimo pretragu.
* Drugi deo predstavlja **referencu na podatak/podatke**, odnosno adrese onih podataka koji odgovaraju određenom ključu.

Na slici 1 prikazan je generalan izgled jednog elementa u strukturi indeksa.



Slika 1. Jedan element indeksa

## 2.2 Klasifikacija indeksa

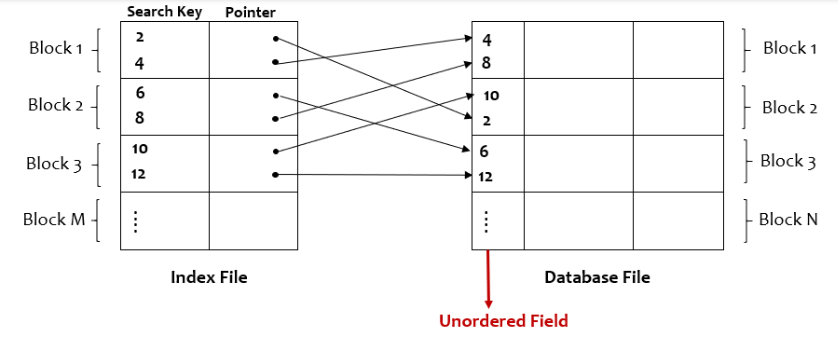
Postoji više kriterijuma na osnovu kojih možemo podeliti indekse. Ovde će biti opisane neke najčešće podele.

Prema tome koliko polja iz tabele se koristi za kreiranje ključa za pretragu, razlikujemo sledeće sve vrste indeksa:

1. **Indeks sa jednim ključem** (engl. *Single-key index*)-za pretragu se koristi samo jedno polje iz tabele.
2. **Indeks sa kompozitnim ključem** (engl. *Composite-key index*)-za pretragu se koriste konkatenirane vrednosti više polja iz tabele. Ukoliko se u upitima dva polja često pojavljuju zajedno, onda je pravljenje ovakvog indeksa verovatno dobro rešenje.

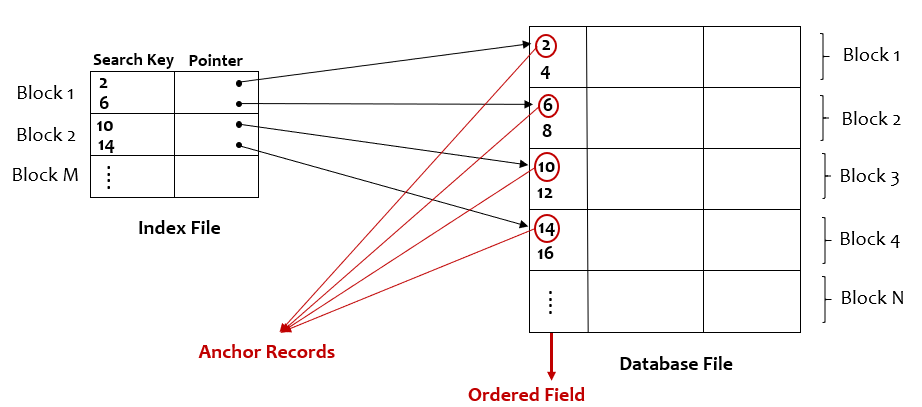
U odnosu na broj unosa/elemenata koji će indeks da sadrži, indeks može biti:

1. **Gust indeks** (engl. *Dense index*)-sadrži po jedan element za svaku vrednost ključa za pretragu. Podaci ne moraju biti sortirani po ključu za pretragu. Izgled gustog indeksa dat je na slici 2.



Slika 2. Gust indeks

1. **Redak indeks** (engl. *Sparse Index*)-ne sadrži elemente za sve vrednosti ključa za pretragu, već samo za neke. Kod ovog indeksa podaci moraju biti sortirani po vrednosti ključa za pretragu. Ukoliko je data referenca za neku vrednost ključa, podaci sa sukcesivnim vrednostima ključa mogu se naći na sukcesivnim adresama od date reference, pa nema potrebe čuvati reference za svaki. Na slici 3 je prikazan primer retkog indeksa.



Slika 3. Redak indeks

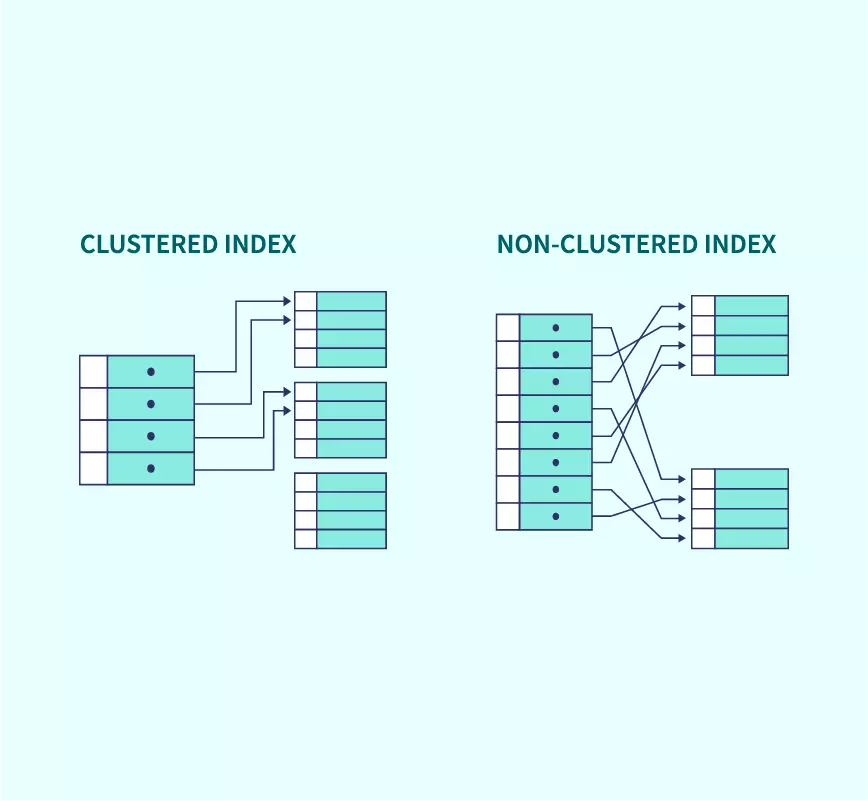
U zavisnosti od toga da li su dozvoljne duplirane vrednosti ključa za pretragu, indeks može biti:

1. **Jedinstven indeks** (engl. *Unique Index*)-Kod ovog indeksa jednoj vrednosti ključa za pretragu odgovara samo jedan unos/podatak iz baze podataka. Ukoliko pokušamo da unesemo podatak koji za vrednost ključa za pretragu ima neku vrednost koja već postoji, ovaj indeks to neće dozvoliti.
2. **Nejedinstven indeks** (engl. *Non-unique Index*)-Suprotno prethodnom, kod ovog indeksa je dozvoljeno da jednoj vrednosti ključa za pretragu odgovara više unosa/podataka iz baze.

U odnosu na to da li su podaci u bazi fizički sortirani po ključu za pretragu indeksa, indeks može biti:

1. **Klasterizovan indeks**-Podaci jesu sortirani po ključu za pretragu. Može postojati samo jedan ovakav indeks po tabeli i to je obično primarni ključ.
2. **Neklasterizovan indeks**-Podaci nisu sortirani po ključu za pretragu. Sukcesivne vrednosti ključa za pretragu ne odgovaraju sukcesivnim lokacijama u bazi.

Na slici 4 ilistrovani su klasterizovan i neklasterizovan indeks.



Slika 4. Klasterizovan i neklasterizovan indeks

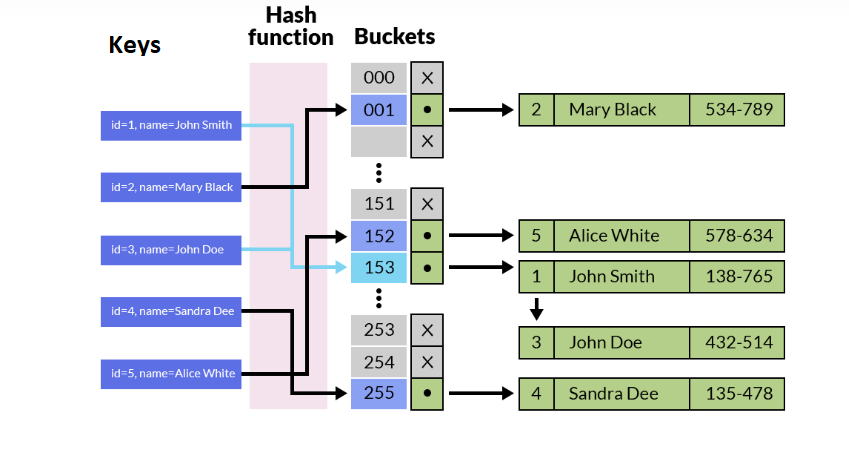
## 2.3 Pregled različitih implementacija indeksa

U ovom odeljku biće opisane različite strukture podataka koje se koriste za indeksiranje.

### 2.3.1 Heš indeks

Hešindeks predstavlja indeks koji je implementiran pomoću heš tablice. Za nalaženje podataka na osnovu ključa koristi se heš hunkcija koja za ulaz uzima ključ za pretragu, a kao izlaz vraća adresu tzv. *bucket-*a. *Bucket* čuva referencu na željeni podatak. Ukoliko se više vrednosti preslikava u isti *bucket*, onda se vrši ulančavanje: jedan bucket predstavlja lančanu listu podataka.

Heš indeks može biti izuzetno efikasan kod pretraga po jednakosti (npr. želimo da pronađemo korisnika sa određenom e-mail adresom). Međutim, ovaj tip indeksa nije efikasan kod upita kojima se traže vrednosti u određenom opsegu (npr. naći sve radnike sa platom većom od neke vrednosti). Princip rada heš indeksa prikazan je na slici 5.



Slika 5. Heš indeks

### 2.3.2 B-stablo

B-stabla se obično koriste za indeksiranje u bazama podataka jer su veoma efikasna u podršci upitima opsega i sortiranju, što su uobičajene operacije u sistemima baza podataka.

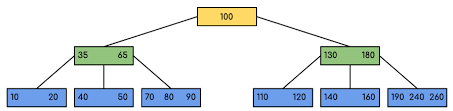
Za razliku od tradicionalnih binarnih stabala pretrage, B-stabla karakteriše veliki broj ključeva koje mogu da pohrane u jednom čvoru, zbog čega su poznata i kao stabla „velikih ključeva“. Svaki čvor u B-stablu može da sadrži više ključeva, što omogućava stablu da ima veći faktor grananja, a samim tim i manju visinu. Ova plitka visina dovodi do manjeg broja pristupa disku, što rezultira bržim operacijama pretraživanja i umetanja. B-stabla su posebno pogodna za sisteme za skladištenje podataka koji imaju spor, glomazan pristup podacima kao što su čvrsti diskovi, fleš memorija i CD-ROM-ovi.

B-stabla održavaju ravnotežu tako što osiguravaju da svaki čvor ima određen minimalan broj ključeva, tako da je stablo uvek uravnoteženo. Ova ravnoteža garantuje da je vremenska složenost za operacije kao što su umetanje, brisanje i pretraživanje uvek O(log n), bez obzira na početni oblik stabla.

B-stabla imaju nekoliko ključnih osobina:

* Svi listovi su na istom nivou.
* Svaki čvor može da sadrži više ključeva i njima pridruženih vrednosti.
* Ključevi u čvoru se čuvaju u sortiranom redosledu.
* Koren stabla mora imati najmanje 2 dece, a svaki drugi čvor može imati najmanje *m*/2 dece i najviše *m* dece, gde je m korisnički definisani parametar koji se naziva „red“ B-stabla.

Na slici 6 prikazan je primer jednog B-stabla. Čvor koji nije list u B-stablu pored ključeva za pretragu čuva pokazivače na svoju decu: jedan pokazivač čvora ukazuje na podstablo sa ključevima većim od prethodne vrednosti ključa čvora, a manjim od narednog ključa čvora. Takođe, svaki čvor kod B-stabla sadrži i reference na podatke koji odgovaraju ključevima pohranjenim u njemu.



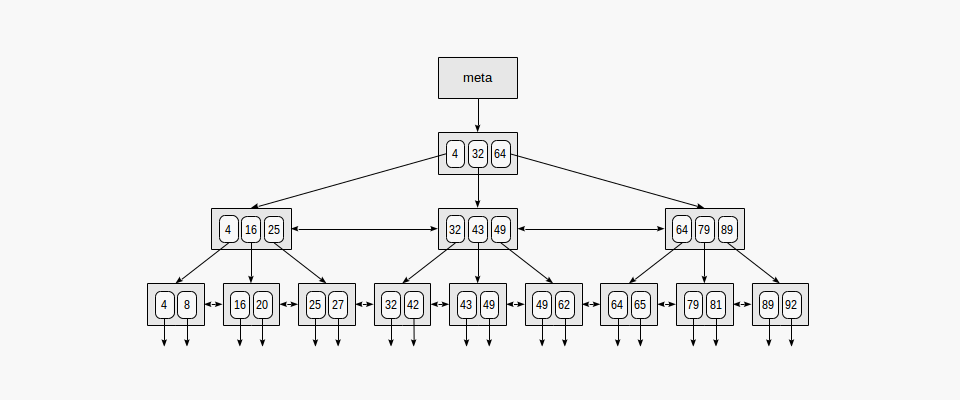
Slika 6. Primer B stabla

### 2.3.3 B+ stablo

B+ stabla su vrsta samouravnoteženih struktura podataka stabla koja se obično koriste za efikasno indeksiranje i pronalaženje velikih količina podataka. Oni su slični B-stablima, ali sa nekim ključnim razlikama u njihovoj strukturi i upotrebi.

Jedna od glavnih razlika između B+ stabala i B-stabala je u tome što B+ stabla čuvaju sve podatke u listovima. Nasuprot tome, B-stabla mogu da čuvaju podatke i u unutrašnjim i u lisnim čvorovima. To znači da se svi ključevi i pridružene vrednosti čuvaju samo u lisnim čvorovima B+ stabla, što može dovesti do efikasnijeg korišćenja memorije i bržih upita opsega. Pošto su svi lisni čvorovi povezani zajedno u lančanu listu, B+ stabla mogu brzo da pređu odgovarajuće lisne čvorove da bi izvršili upite opsega.

Još jedna važna karakteristika B+ drveća je njihov *fanout* faktor, koji je obično veći od B-stabala. Faktor *fanout*-a je maksimalan broj ključeva koji se mogu uskladištiti u svakom čvoru stabla. Pošto B+ stabla čuvaju ključeve samo u internim čvorovima, a svi podaci se čuvaju u lisnim čvorovima, faktor *fanout-*a može biti mnogo veći u B+ stablima, što može dovesti do efikasnijeg korišćenja memorije i bržeg obilaska stabla. Na slici 7 prikazana je struktura B+ stabla.



Slika 7. B+ stablo

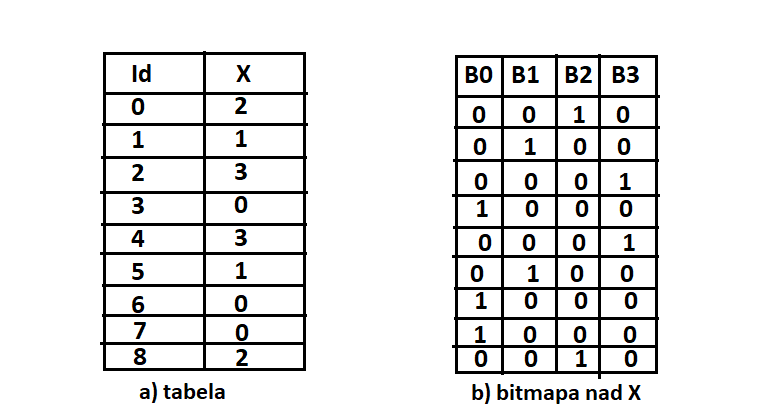
Sve u svemu, B+ stabla su moćna i široko korišćena struktura podataka za efikasno indeksiranje i preuzimanje velikih količina podataka. Čuvanje podataka samo u listovima, veći fanout faktor i efikasni upiti opsega čine ih posebno pogodnim za upotrebu u sistemima baza podataka.

### 2.3.4 *Bitmap* indeks

Bitmap index je poseban tip indeksa gde se podaci čuvaju u vidu polja bitova i upiti nad podacima se izvršavaju primenom logičkih *bitwise* operacija nad ovim poljima. Ovakva vrsta indeksa korisna je kada želimo indeks nad kolonom/atributom čija je kardinalnost jako mala, odnosno broj različitih vrednosti koje atribut može da uzme je jako mali. Indeksi poput B i B+ indeksa su najefikasniji kada imamo veliki broj različitih vrednosti ključa za pretragu, dok je u slučajevima male kardinalnosti bolje kreirati bitmap indeks.

Struktura bitmap indeksa prikazana je na slici 8. Na slici 8a prikazana je tabela koja se sastoji iz kolona *Id* i *X*, pri čemu je *X* kolona koju želimo da indeksiramo. Indeks (prikazan na slici 8b) se formira tako što se svakoj različitoj vrednosti kolone *X* pridruži niz bitova koji ima onoliko elemenata koliko redova ima tabela. Vrednost 0 na poziciji *i* u takvom nizu označava da red *i* ima neku drugu vrednost od one kojoj je pridruženo to polje bitova, a vrednost 1 znači da red *i* ima istu vrednost kao i vrednost pridružena polju.

Da bi izvršili upit nad tabelom prikazanom na slici koji na primer traži sve redove za koje važi X>1, potrebno je da se nad poljima B2 i B3 izvrši logička OR operacija. Pošto su logičke *bitwise* operacije odlično podržane od strane hardvera, ovakvi upiti su izuzetno brzi.

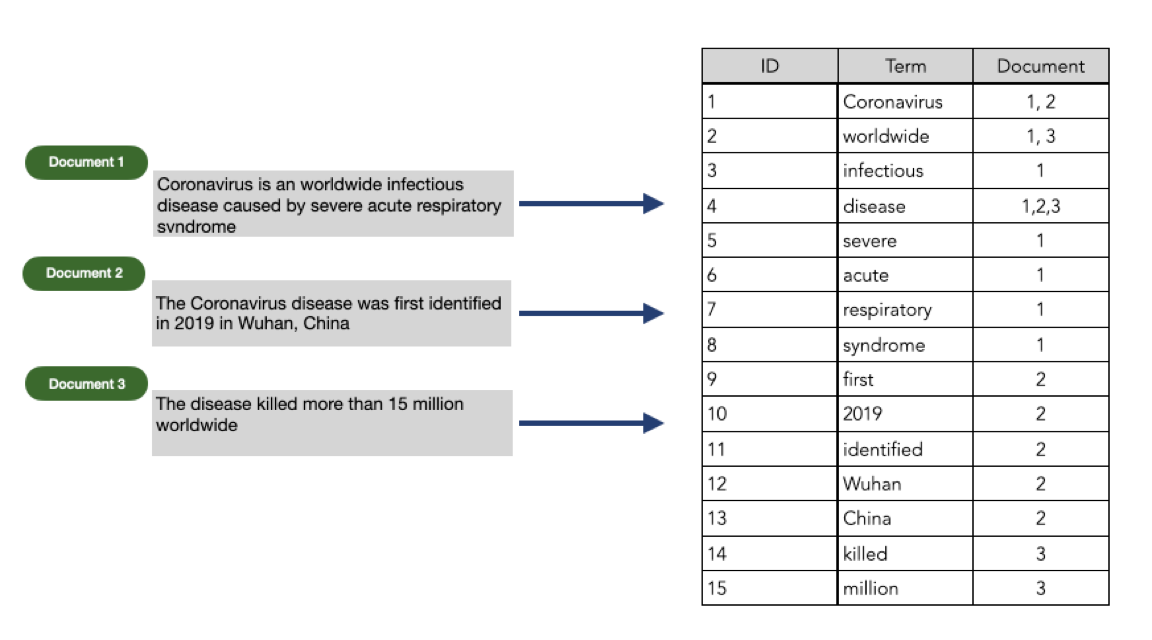


Slika 8. Bitmap indeks

### 2.3.5 Invertovani indeks

Invertovani indeks je struktura podataka koja se koristi za efikasno pretraživanje tekstualnih podataka na osnovu specifičnih termina ili ključnih reči.

U invertovanom indeksu, svaki termin ili ključna reč u skupu dokumenata je povezana sa listom pokazivača na dokumente koji sadrže taj termin. Na primer, ako imamo skup dokumenata koji sadrže termine „mačka“, „pas“ i „hrčak“, obrnuti indeks će sadržati tri liste, od kojih svaka sadrži pokazivače na dokumente koji sadrže odgovarajući termin. Na slici 9 dat je primer jednog invertovanog indeksa.



Slika 9. Invertovani indeks

Postoje dve vrste invertovanih indeksa: *record-level* indeks koji za svaki termin sadrži samo listu dokumenata koji ga sadrže (indeks sa prethodne slike je *record-level*) i *word-level* indeks koji za svaki termin pored dokumenata u kome se on nalazi, sadrži i poziciju na kojoj se termin nalazi u svakom dokumentu.

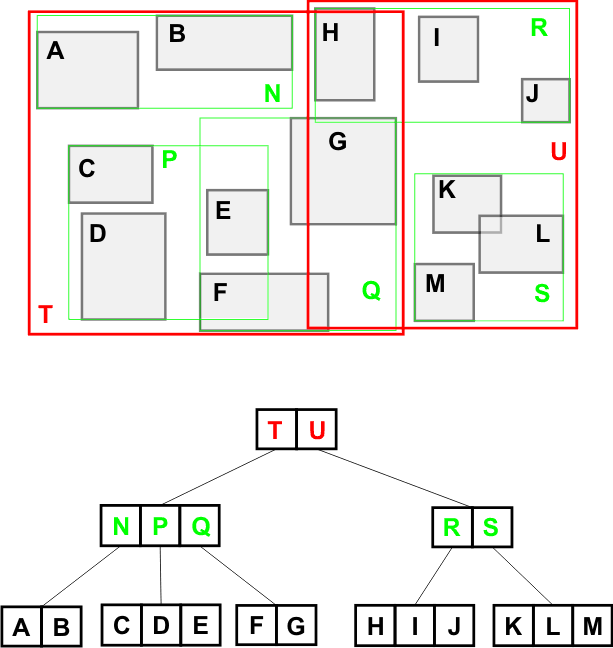
Invertovani indeks je izuzetno popularan u bazama podataka i internet pretraživačima za pretragu teksta. Ima jednostavnu strukturu i lak je za kreiranje. Međutim, zahteva određen memorijski prostor i veće troškove za operacije ažuriranja, brisanja i dodavanja dokumenata.

### 2.3.6 R-stablo

R-stabla su strukture podataka koje se koriste za implementaciju prostornog traženja, tj. za indeksiranje višedimenzionalnih informacija kao što su geografske koordinate, pravougaonici ili poligoni. R-stablo je predložio Antonin Guttman 1984. godine. R-stabla su veoma korisna za skladištenje prostornih objekata kao što su lokacije restorana ili poligoni od kojih se sastoje mape: ulice, zgrade, obrisi jezera, obale itd. i brzo pronalaženje odgovora na upite kao što je „Pronađi sve muzeje u krugu od 2 km od moje trenutne lokacije“, „preuzmi sve segmente puta u krugu od 2 km od moje lokacije“ ili „pronađi najbližu benzinsku pumpu“ .

Ključna ideja ove strukture podataka je grupisanje obližnjih objekata i njihovo predstavljanje sa njihovim minimalnim graničnim pravougaonikom u sledećem višem nivou stabla; "R" u R-stablu označava pravougaonik (engl. *Rectangle*). Pošto svi objekti leže unutar ovog graničnog pravougaonika, upit koji ne seče granični pravougaonik takođe ne može preseći nijedan od sadržanih objekata. Na nivou lista, svaki pravougaonik opisuje jedan objekat; na višim nivoima agregacija uključuje sve veći broj objekata. Ovo se takođe može posmatrati kao sve grublja aproksimacija skupa podataka. Na slici 10 prikazan je primer R stabla za indeksiranje 2D podataka.

Slično B-stablu, R-stablo je takođe uravnoteženo stablo pretrage (tako da su svi čvorovi lista na istoj dubini). R-stabla se mogu posmatrati kao B-stabla za višedimenzionalne podatke. Kao i kod većine stabala, algoritmi pretraživanja su prilično jednostavni. Ključna ideja je korišćenje graničnih okvira za odlučivanje da li će se pretraživati unutar podstabla ili ne. Na ovaj način, većina čvorova u stablu se nikada ne obiđe tokom pretrage. Kao i B-stabla, R-stabla su pogodna za velike skupove podataka i baze podataka, gde se čvorovi mogu preneti u memoriju kada je to potrebno, pa se celo stablo ne mora čuvati u glavnoj memoriji.



Slika 10. Dvodimenzionlano R-stablo

# Indeksi u MongoDB bazi podataka

## 3.1 MongoDB

MongoDB predstavlja NoSQL, dokument-orjentisanu bazu podataka. Ovde se baza podataka sastoji iz **kolekcija,** koje su analogne tabelama kod relacionih baza podataka. Svaka kolekcija sastoji se iz skupa **dokumenata,** analogno redovima kod relacionih baza. Međutim, za razliku od relacionih baza, ovde ne postoji šema nad podacima, već dokumenti unutar kolekcije ne moraju imati istu strukturu.

Jedan dokument se sastoji iz **ključ-vrednost** parova. Ključ predstavlja naziv polja, dok vrednost polja predstavlja podatak koji može biti: *objectID*, string, broj, niz, datum, drugi dokument, binarni podatak, regularni izraz, *null* vrednost...

Na slici 11 je prikazan primer jednog dokumenta u Mongo bazi.



Slika 11. Primer MongoDB dokumenta

MongoDB čuva podatke u binarnom JSON formatu koji se zove BSON (engl. *Binary JSON*). BSON je binarno kodirana serijalizacija dokumenata sličnih JSON-u, koja omogućava efikasno i kompaktno skladištenje podataka u MongoDB. BSON proširuje JSON model podataka dodatnim tipovima podataka i optimizacijama, kao što je podrška za binarne podatke i kompresiju podataka.

Jedna od prednosti MongoDB-a je njegova fleksibilnost u rukovanju strukturama dokumenata. MongoDB dokumenti ne moraju da imaju istu šemu, što omogućava laku modifikaciju i evoluciju modela podataka tokom vremena. Pored toga, MongoDB pruža fleksibilan jezik upita koji omogućava da se složeni upiti i agregacije izvrše na podacima uskladištenim u kolekcijama.

MongoDB poseduje niz funkcija za podršku trajnosti i pouzdanosti podataka, uključujući replikaciju, deljenje i rezervne kopije. Replikacija omogućava održavanje više kopija podataka na različitim serverima radi visoke dostupnosti, dok deljenje omogućava horizontalno skaliranje baze podataka na više servera. Mogu se napraviti rezervne kopije kako bi se osiguralo da se podaci mogu vratiti u slučaju gubitka ili oštećenja podataka.

MongoDB takođe podržava indekse, koji omogućavaju brz pristup podacima zasnovanim na određenim poljima ili kombinacijama polja. Indeksi se mogu kreirati nad bilo kojem polju u kolekciji i mogu se konfigurisati da podržavaju različite obrasce upita.

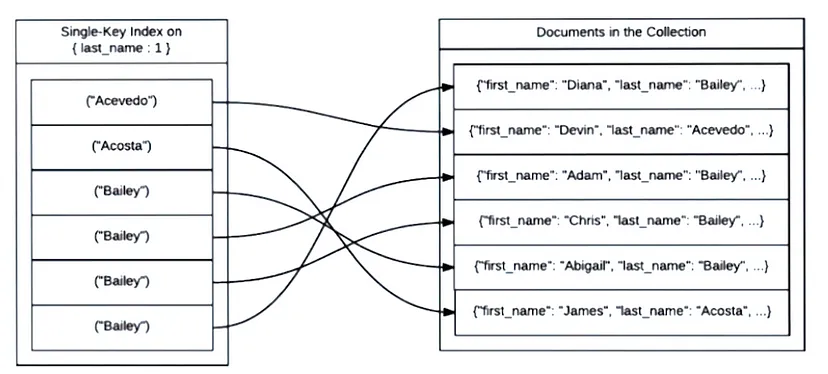
## 3.2 Indeksi u MongoDB-u

Indeksi podržavaju efikasno izvršavanje upita u MongoDB-u. Bez indeksa, MongoDB mora da izvrši skeniranje čitave kolekcije, tj. da skenira svaki dokument u kolekciji, da bi izabrao one dokumente koji odgovaraju naredbi upita. Ako postoji odgovarajući indeks za upit, MongoDB može koristiti indeks da ograniči broj dokumenata koje mora da pregleda. Na slici 12 prikazan je primer indeksa u MongoDB-ju. Indeksi su u MongoDB-u implementirani pomoću **B-stabla**.

Kada MongoDB uveze podatke u kolekciju, kreiraće indeks nad primarnim ključem. Ali ne može da pretpostavi druge indekse koji su potrebni jer ne postoji način da predvidi vrstu pretraživanja, sortiranja i agregiranja koju će korisnik želeti da izvrši nad podacima. Zato je na korisniku da definiše indekse koji su mu potrebni, u skladu sa upitima koje planira da obavlja nad podacima.

Indeks se u MongoDB-ju može kreirati nad bilo kojim poljem dokumenta. Postoje i posebne vrste indeksa kao što su složeni indeksi, indeksi za pretragu tekstualnih podataka, indeksi za pretragu prostornih informacija, itd. O vrstama indeksa biće više reči kasnije.

Svaki upit ili ažuriranje će generalno koristiti samo jedan indeks ako postoji odgovarajući. Indeks obično može pomoći u performansama bilo koje operacije sa podacima, ali to nije uvek slučaj. Korisnici mogu biti u iskušenju da isprobaju pristup „scattergun“ – kreiranje mnogo različitih indeksa, kako bi osigurali da će postojati jedan koji će verovatno biti prikladan – ali loša strana je u tome što svaki indeks koristi resurse i sistem mora da ga održava kad god se podaci menjaju. Ako se preteruje sa indeksima, oni će dominirati memorijskim stranicama i dovesti do prekomernog I/O diska. Zato je mali broj visoko efikasnih indeksa najbolji.



Slika 12. Primer indeksa u MongoDB-u

## 3.3 Vrste indeksa u MongoDB-u

### 3.3.1 Indeks nad jednim poljem (engl. Single Field Index)

*Single field* indeks predstavlja indeks koji se kreira nad samo jednim poljem dokumenta. Ovaj tip indeksa se može kreirati nad bilo kojim poljem dokumenta. Na slici 13 je dat primer jednog dokumenta. Nad poljem dokumenta \_id, koje predstavlja jedinstveni identifikator dokumenta, indeks je kreiran po difoltu pri kreiranju dokumenta. *Single field* indeks se može kreirati nad poljem **score**, ali se može kreirati i nad poljem **location** čiji tip predstavlja dokument. Indeks se može kreirati i nad poljima **location.state** i l**ocation.city**, koja su ugnježdena. Ne postoje nikakava ograničenja za kreiranje single field indeksa.



Slika 13. Primer dokumenta

Sledećom naredbom kreira se indeks nad poljem score:



Vrednost koja se navodi prilikom kreiranja indeksa označava redosled kojim će indeks urediti stavke. Vrednost 1 označava rastući, dok vrednost –1 označava opadajući redosled. Međutim, kod *single field* indeksa, nije bitno u kom redosledu indeks uređuje stavke. Ovaj indeks će podržati operacije sortiranja u bilo kom smeru. O redosledu indeksa je potrebno voditi računa kada je indeks složen.

### 3.3.2 Složeni indeks (engl. *Compound Index*)

MongoDB podržava složene indekse, gde jedna struktura indeksa sadrži reference na više polja unutar dokumenata kolekcije. Složeni indeksi podržavaju upite u kojima je specificirano više polja od jednog. Složeni indeks se kreira sledećom naredbom:

*db.collection.createIndex( { <field1>: <type>, <field2>: <type2>, ... } ),*

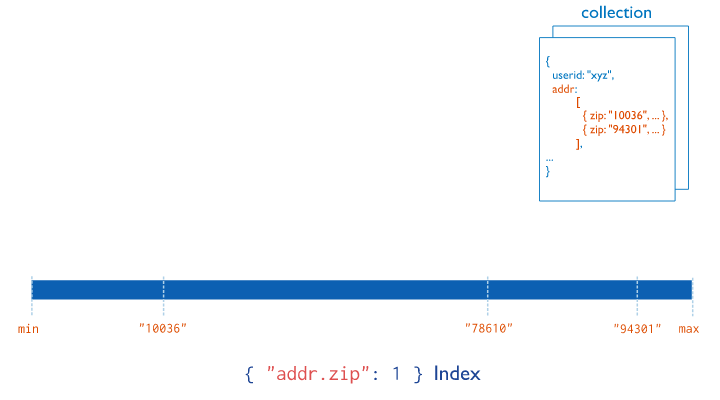
Pri čemu su sa filed1, field2,.. označeni nazivi polja nad kojima se kreira indeks, a sa type1, type2, .. su označene vrednosti kojima se redosled kojima indeks sortira stavke (1 za rastući i –1 za opadajući redosled).

Da bi upit koristio složeni indeks prilikom pretrage podataka, upit mora sadržati **prefiks** ključeva nad kojim je indeks definisan. Na primer, ukoliko je indeks definisan nad poljima {a,b,c,d}, prefiksi ovog indeksa bili bi: {a},{a,b},{a,b,c} i {a,b,c,d}. Prema tome, indeks će se koristiti najefikasnije za one upite koji sadrže polja koja sadrži neki od prefiksa. Recimo pretraga po poljima {a,d} može koristiti indeks, jer a predstavlja jedan od prefiksa, ali ne na toliko efikasan način kao kada bi sva polja činila neki prefiks. Pretraga po poljima {b,c} neće uopšte koristiti indeks, jer ne sadrži nijedan prefiks.

Da bi upit koristio složeni indeks za sortiranje, navedeni pravac sortiranja za sve ključeve u upitu mora da odgovara uzorku ključa indeksa ili da se podudara sa inverznim uzorkom ključa indeksa. Na primer, šablon ključa indeksa { a: 1, b: -1 } može da podrži sortiranje na { a: 1, b: -1 } i { a: -1, b: 1 }, ali ne i na { a: - 1, b: -1 } ili {a: 1, b: 1}. Složeni indeks se može koristiti i za sortiranje nad prefiksom polja nad kojim je definisan: na primer indeks definisan sa {a:1,b:1,c:1} se može koristiti za sortiranje nad {a:1} i {a:1,b:1}, ali ne i nad {b:1,a:1} , {a:1,c:1} ili {b:1,c:1}.

### 3.3.3 Multikey indeks

Ova vrsta indeksa se koristi za indeksiranje polja čija vrednost predstavlja niz. Ovakav indeks omogućava efikasne pretrage po poljima tipa niz. Multikey indeks se može kreirati nad nizom koji sadrži skalarne vrednosti, ali i nad nizom čiji su elementi ugnježdeni dokumenti. Na slici 14 prikazan je primer multikey indeksa.



Slika 14. *Multikey* indeks

Multikey indeks prate i neke vrste ograničenja. Na primer, pri definisanju složenog multikey indeksa samo jedno polje može biti niz.

### 3.3.4 Geoprostorni indeks (engl. *Geospatial Index*)

Da bi podržao efikasne upite geoprostornih koordinatnih podataka, MongoDB obezbeđuje dva posebna indeksa: 2d indekse koji koriste planarnu geometriju kada vraćaju rezultate i 2dsphere indekse koji koriste sferičnu geometriju za vraćanje rezultata.

### 3.3.5 Indeks za pretragu teksta

MongoDB obezbeđuje tekstualne indekse koji podržavaju upite za pretragu teksta na sadržaju stringova. Kolekcija može imati samo jedan indeks za pretragu teksta, ali taj indeks može pokriti više polja.

Prilikom kreiranja tekstualnog indeksa, treba navesti željeno polje koje mora biti tipa string ili niz koji sadrži elemente tipa string. Primer naredbe kojom se kreira tekstualni indeks bi bio:

db.reviews.createIndex( { comments: "text" } ).

Tekstualni indeks se može kreirati i nad više polja, ukoliko sva ta polja predstavljaju string podatke. Takođe, prilikom kreiranja tekstualnog indeksa nad više polja može se navesti težina svakog polja, tako da relevantnija polja više utiču na rezultat pretrage.

Za kreiranje tekstualnog indeksa nad više polja, može se koristiti i *wildcard* znak ($\*\*). Ovim se kreira tekstualni indeks nad svim poljima u dokumentima kolekcije koja sadrže string podatke. Ovakav indeks je posebno koristan kada imamo nestruktuirisane podatke kod kojih ne znamo koja će tačno polja sadržati tekst. *Wildcard* tekstualni indeks kreira se sledećom naredbom:

db.collection.createIndex( { "$\*\*": "text" } )

### 3.3.6 Heširani indeks

Indeksi u kolekciji obično zahtevaju dodatni prostor za skladištenje. Heširani indeksi se mogu koristiti za smanjenje prostora koji indeksi zauzimaju. Heširani indeks čuva heširane vrednosti indeksiranog polja umesto samih vrednosti. Heširani indeksi ne podržavaju indekse sa više ključeva. Heširani indeksi se mogu kreirati isključivo nad jednim poljem, ali vrednost tog polja može biti i ugnježdeni dokument. Heširani indeks se može koristiti i kao **sharding key**, odnosno kao ključ po kojem se podaci distribuiraju na više mašina.

### 3.3.7 Wildcard indeks

MongoDB podržava kreiranje indeksa na polju ili skupu polja za podršku upitima. Pošto MongoDB podržava dinamičke šeme, aplikacije mogu da postavljaju upite prema poljima čija imena ne mogu biti poznata unapred ili su proizvoljna. MongoDB 4.2 uvodi wildcard indekse za podršku upita kada struktura podataka nije dobro poznata i podaci sadrže proizvoljna polja čiji se nazivi ne znaju unapred. Sledećom naredbom dat je primer kreiranja wildcard indeksa:

db.wildcardIndex.createIndex({"data.$\*\*":1});

Ovim je kreiran indeks nad svakim atributom koji polje *data* može da ima. Ukoliko imamo dokument sa sledećim sadržajem:

{  
 "\_id" : 1,  
 "data" : {  
 "a" : 1728,  
 "b" : 6740,  
 "c" : 6481,  
 "d" : 2066,  
 "e" : 3173,  
 "f" : 1796,  
 "g" : 8112  
 }  
},

Indeks će biti kreiran nad svakim od polja ugnježdenog dokumenta data: a,b,c,d,e,f i g. Naredba za kreiranje indeksa nad svim poljima u dokumentima bila bi sledeća:

db.collection.createIndex( { "$\*\*" : 1 } ).

## 3.4 Karakteristike indeksa u MongoDB-u

MongoDB Indeksi mogu imati neke specijalne osobine koje je potrebno specificirati prilikom kreiranja indeksa. Neke od njih su:

* **Jedinstvenost**-*Unique* indeks osigurava da se vrednosti polja nad kojim je indeks kreiran ne dupliraju. Po difoltu, MongoDB kreira unique indeks nad \_id poljem dokumenta prilikom kreiranja kolekcije. Svi ostali indeksi podrazumevano nisu jedinstveni, odnosno dozvoljavaju dupliranje vrednosti. Da bi index bio jedinstven naredba kojom se kreira treba da izgleda ovako:

db.collection.createIndex( <key and index type specification>, { unique: true } ).

* **Parcijalnost-** Podrazumevano se indeks kreira nad celom kolekcijom dokumenata. Međutim, postoje delimični (parcijalni) indeksi koji indeksiraju samo dokumente u kolekciji koji ispunjavaju specificirani izraz filtera. Indeksiranjem podskupa dokumenata u kolekciji, delimični indeksi imaju niže zahteve za skladištenjem i smanjene troškove kreiranja i održavanja indeksa.

Da bi indeks bio parcijalni, prilikom kreiranja indeksa treba specificirati filter u kome se definišu uslovi koje dokumenti treba da zadovolje da bi bili uključeni u kreirani indeks. Uslovi koji se navode u filteru mogu biti: uslovi koji se definišu operatorima =,>,<,>=,<= (npr. Da se kreiraju indeks samo nad dokumentima čije određeno polje ima određenu vrednost ili vrednost veću/manju od specificirane), uslovi koji se definišu *and, or* i *in* operatorima, uslov koji se definiše *type* operatorom. Primer naredbe kojom se definiše parcijalni indeks izgleda ovako:

db.restaurants.createIndex(

{ cuisine: 1, name: 1 },

{ partialFilterExpression: { rating: { $gt: 5 } } }},

Pri čemu je *partialFilterExpression* opcija u kojoj se specificira filter prilikom kreiranja indeksa. Ovom naredbom kreiran je složen indeks nad poljima cuisine i name, a indeks je kreiran samo nad dokumentima koji imaju polje rating sa vrednošću većom od 5.

* **Životni vek** -Moguće je specificirati životni vek indeksa kojim se ustvari definiše nakon kog vremena će dokumenti nad kojima je kreiran indeks biti obrisani iz kolekcije. TTL (engl. *Time To Live*) indeksi su specijalni indeksi sa jednim poljem koje MongoDB može da koristi za automatsko uklanjanje dokumenata iz kolekcije nakon određenog vremena ili u određeno vreme. Istek podataka je koristan za određene tipove informacija kao što su podaci o događajima generisani od strane mašina, evidencije i informacije o sesiji koje treba da ostanu u bazi podataka samo ograničeno vreme.

Sledeći primer ilistruje kreiranje TTL indeksa nad  *lastModifiedDate* poljem dokumenta:

db.eventlog.createIndex( { "lastModifiedDate": 1 }, { expireAfterSeconds: 3600 } ),

Ovom naredbom je definisano da svaki dokument bude izbrisan nakon 3600 sekundi.

* **Skrivenost**- Skriveni indeksi nisu vidljivi planeru upita i ne mogu se koristiti za podršku upita. Sakrivanjem indeksa od planera, korisnici mogu da procene potencijalni uticaj brisanja indeksa bez stvarnog brisanje indeksa. Ukoliko se pokaže da su perfomanse bolje ukoliko se indeks koristi, korisnik može da otkrije indeks umesto da mora ponovo da pravi obrisani indeks.
* **Retkost**- Retki indeksi sadrže samo unose za dokumente koji imaju indeksirano polje, čak i ako indeksno polje sadrži *null* vrednost. Retki indeks preskače svaki dokument kome nedostaje indeksirano polje. Indeks je redak jer ne obuhvata sve dokumente kolekcije. Nasuprot tome, neretki indeksi sadrže sve dokumente u kolekciji, ubacujući nulte vrednosti za one dokumente koji ne sadrže indeksirano polje. Redak indeks može kreirati sledećom naredbom:

db.collectionName.createIndex( { "field": 1 }, { sparse: true } ).

Geoprostorni, tekstualni i wildcard indeksi su podrazumevano retki, dok za ostale indekse to ne važi.

* **Osetljivost na mala i velika slova**-Za indekse koji obavljaju pretragu nad string podacima, može se specificirati da budu neosetljivi na mala i velika slova.

## 3.5 Strategije indeksiranja

Da bi se kreirali najbolji indeksi moraju se uzeti u obzir brojni faktori, uključujući vrste upita koje se očekuju, odnos čitanja i pisanja podataka i količinu slobodne memorije na sistemu.

Prilikom razvijanja strategije indeksiranja, bitno je posedovati duboko razumevanje upita aplikacije. Pre kreiranja indeksa, treba analizirati polja koja se koriste pri upitima i razmotriti ta polja kao potencijalne ključeve indeksa. Indeksi dolaze sa troškovima performansi, ali su više nego vredni troškova za česte upite u velikim skupovima podataka. Treba razmotriti relativnu učestalost svakog upita u aplikaciji i da li upit opravdava indeks.

Najbolja opšta strategija za kreiranje indeksa je profilisanje različitih konfiguracija indeksa sa skupovima podataka sličnim onima koje će se koristiti u proizvodnji da bi se uvidelo koje konfiguracije imaju najbolje rezultate. Treba pregledati trenutne indekse kreirane za kolekcije da bi se osigurali da oni podržavaju trenutne i planirane upite. Ako se indeks više ne koristi, treba ga obrisati.

Generalno, MongoDB koristi samo jedan indeks da ispuni većinu upita. Međutim, svaka klauzula upita sa operacijom $or može da koristi drugačiji indeks, a pored toga, MongoDB može da koristi presek više indeksa (ali samo u određenim uslovima).

Prilikom kreiranja indeksa u MongoDB bazi, preporučuju se sledeće strategije:

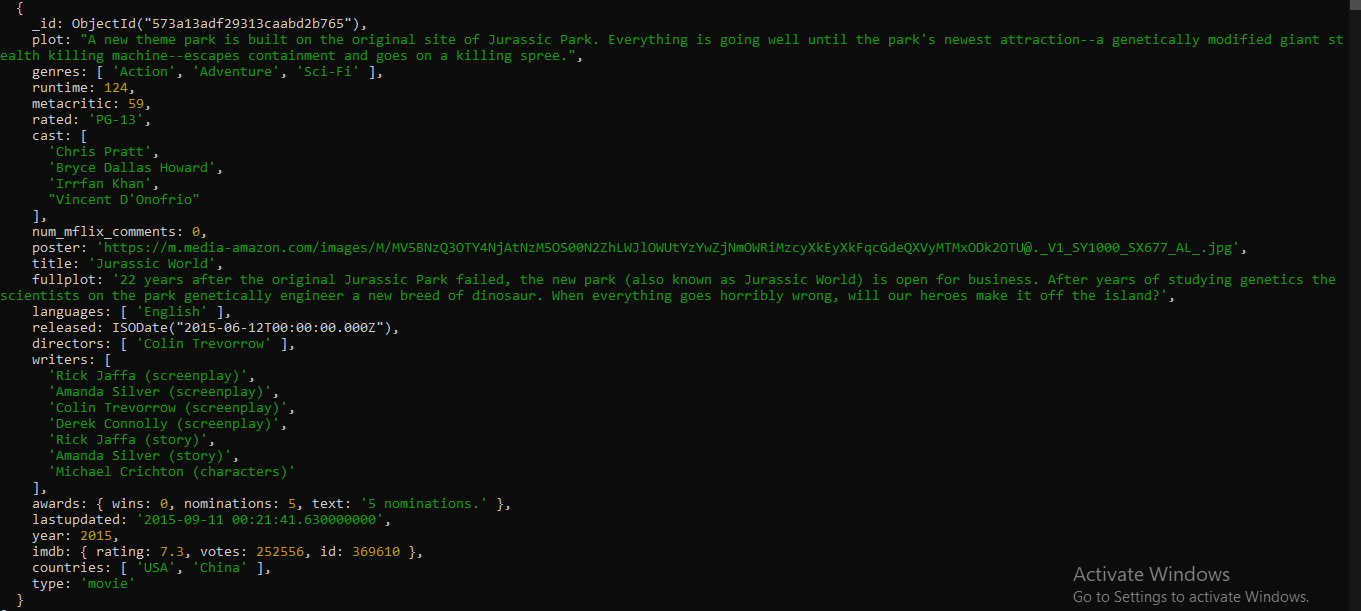
1. **ESR (Equality, Sort, Range) pravilo**-Prilikom kreiranja složenog indeksa, o kome je bilo reči, bitno je voditi računa o redosledu polja iz kojih se ovaj indeks sastoji. Prilikom navođenja redosleda polja od kojih se sastoji složen ključ, prvo treba navesti ona polja koja se u upitima koriste za pretragu po jednakosti. Zatim treba navesti ona polja koja se u upitima koriste za sortiranje. Polja za sortiranje se navode nakon polja za pretragu jednakosti, jer pretraga po jednakosti smanjuje broj dokumenata, pa je efikasnije iskoristiti indeks za sortiranja nakom što se obavi pretraga po jednakosti. Na kraju treba navesti ona polja kojim se upitima pretražuje neki opseg. To je zato što se sortiranje korišćenjem indeksa ne može obaviti nad podacima koji su rezultati range filtera.
2. **Kreiranje indeksa za podršku većeg broja upita**-Indeks podržava upit onda kada indeks sadrži sva polja koja se javljaju u upitu. Međutim, ukoliko postoji veliki broj različitih vrsta upita, kreiranja posebnog indeksa za svaki od njih bi mogao negativno da utiče na performanse. Nekad je moguće kreirati jedan indeks koji će podržati veći broj upita. Ukoliko na primer imamo upit koji pretražuje podatke po jednom polju i drugi upit koji pretražuje podatke po istom tom polju i još jednom drugom, složeni indeks za podršku drugom upitu može da se iskoristi i za prvi upit (zbog pravila o prefiksima, o kojima je bilo reči).
3. **Korišćenje indeksa za operacije sortiranja**-Ukoliko ne postoji odgovarajući indeks, MongoDB mora da učita sve dokumente kolekcije u RAM memoriju i da izvrši sortiranja. Ovakav način sortiranja je neefikasan i spor i MongoDB će prekinuti ovu operaciju i vratiti grešku ukoliko je za sortiranje potrebno više od 100 megabajta, što će biti slučaj kod velikih kolekcija. Međutim, ukoliko kreiramo indeks nad poljem ili poljima nad kojima želimo da sortiramo podatke, indeks će čuvati vrednosti u sortiranom redosledu, tako da će operacija sortiranja nakon kreiranja indeksa biti mnogo brža.
4. **Vođenje računa da indeks staje u RAM memoriju**-Da bi indeks bio najefikasniji, potrebno je da njegova veličina ne bude prevelika, tako da može da stane u memoriju. Na ovaj način, neće biti potrebno da se stranice indeksa koje ne mogu da stanu u memoriju čitaju sa diska.

# Praktična demonstracija indeksa u MongoDB-u

U ovom poglavlju biće prikazano kreiranje, održavanje i brisanje indeksa, kao i demonstracija upita koji koriste indekse. Podaci koji su se koristili za demonstraciju skinuti su sa MongoDB stranice: [link](https://www.mongodb.com/developer/products/atlas/atlas-sample-datasets/).

## 4.1 Kreiranje single key, multikey i compound indeksa, brisanje indeksa

Najpre će biti prikazano kreiranje indeksa nad kolekcijom filmova. Kolekcija sadrži preko 25 hiljada filmova. Na sledećoj slici prikazan je izgled jednog dokumenta iz ove kolekcije:

Slika 15. Primer dokumenta iz kolekcije movies

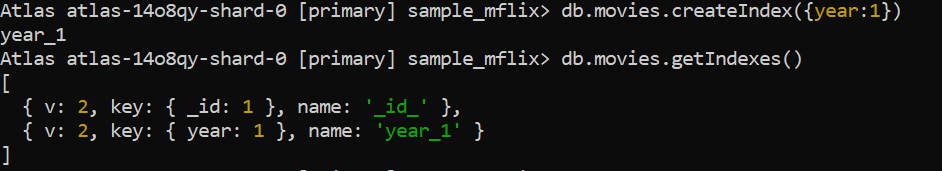
Kao što se vidi sa slike 15, jedan film sadrži veliki broj informacija: radnja filma, naslov filma, godina, podatke o režiserima, glumcima, imdb oceni, žanru, nagradama,..Ovaj veliki broj informacija nam omogućava pretragu filmova po raznim kriterijumima. Međutim, da bi te pretrage bile efikasne neophodno je da prvo napravimo indekse.

Prilikom kreiranja kolekcije kreira se samo jedan indeks: indeks nad \_id poljem dokumenata. Naredbom *db.collectionName.getIndexes()* možemo da proverimo koji sve indeksi postoje u kolekciji i uverimo se u prethodnu tvrdnju:



Za početak će biti kreiran indeks nad poljem “year” kako bi pretraga filmova po godinama bila olakšana. Pošto ovo polje ima skalarnu vrednost, tip kreiranog indeksa biće *Single Key Index*.

Kreiranje indeksa nad poljem year dato je na sledećoj slici. Na slici je takođe prikazan i rezultat poziva getIndexes() funkcije nakon kreiranja novog indeksa:



Slika 16. Kreiranje indeksa

Nakon što je indeks kreiran, možemo izvršiti pretragu filmova po godinama. Kao što se sa prethodne slike vidi, ime kreiranog indeksa je year\_1. Da bi proverili da li željeni upit zaista koristi kreirani indeks, potrebno je nad upitom pozvati explain() operaciju. Na sledećoj slici prikazan je rezultat operacije explain nad upitom koji vraća 5 filmova sa godinom izdavanja većom od 2010. U dokumentu koji vraća reultat, u polju winningPlan naveden je indeks year\_1, što znači da je upit izvršen korišćen indeksa.



Slika 16. Primer upita koji koristi indeks

Često prilikom pretraživanja imamo potrebu da specificiramo više atributa. Za takve slučajeve radi efikasnije pretrage neophodno je kreirati složen indeks koji će se sastojati od više polja. Na primer, ukoliko želimo da filmove pretražujemo po godini, žanru i imdb oceni (i baš tim redosledom), treba kreirati sledeći indeks:



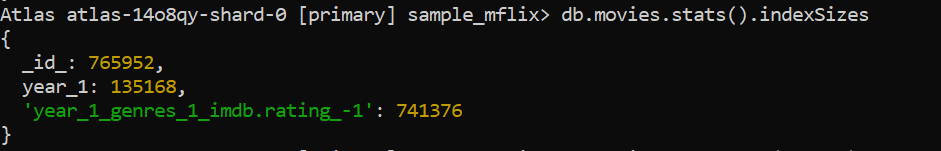
Ovakav indeks je složen *multikey* indeks, jer je kreiran nad više polja i polje *genres* predstavlja niz. Za polja year i genres je naznačeno da budu u rastućem redosledu, dok je za polje imdb.rating naveden opadajući redosled.

Na slici ispod naveden je primer upita koji koristi ovaj indeks. Ovakav upit vraća sve filmove snimljene nakon 2010 godine, čiji je žanr drama i ocena veća od 6.

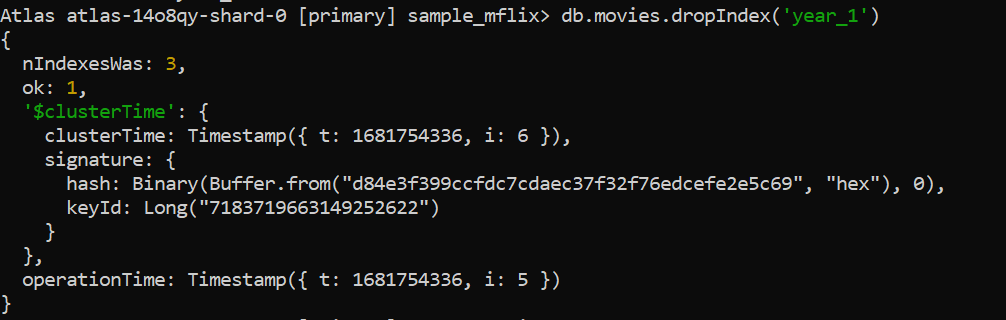


Slika 17. Upit nad složenim indeksom

Ukoliko želimo da vidimo koliku količinu memorije zauzima svaki indeks u bajtovima, to je moguće izvršenjem sledećeg upita:



Kako složen indeks koji je kreiran redom nad poljima “year”, “genres” i “imdb.rating” među prefiksima sadrži polje “year”, onda je indeks kreiran samo nad poljem year (year\_1) moguće izbrisati, jer složeni indeksi mogu podržati pretragu po svim svojim prefiksima. Kao što se sa prethodne slike vidi, brisanjem ovog indeksa oslobodićemo se memorije od 135168 bajtova. Brisanje indeksa year\_1 prikazano je na sledećoj slici:



Slika 18. Brisanje indeksa

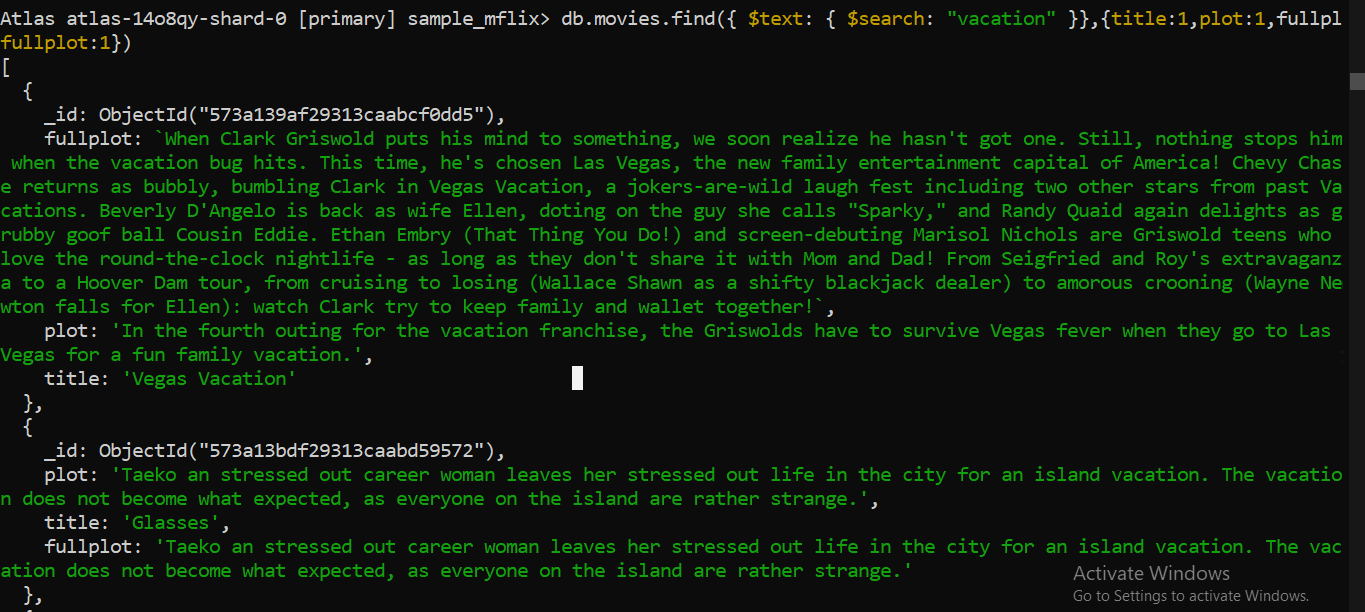
## 4.2 Tekstualni indeks

Prilikom pretraživanja filmova, upiti mogu često da se odnose na pretraživanje filmova po radnji. Za te potrebe zgodno je kreirati tekstualni indeks nad kolekcijom indeksa. Ovde je kreiran složen tekstualni indeks koji se sastoji od 2 polja: “plot” i “fullplot”. Polje “plot” ukratko opisuje film, dok “fullplot” daje detaljniji opis filma. Poljima u indeksu su dodate i težine, redom 10 i 5, kako bi se polju “plot” dala veća relevantnost od polja “fullplot”. Kreiranje ovakvog indeksa prikazano je naredbom na sledećoj slici:



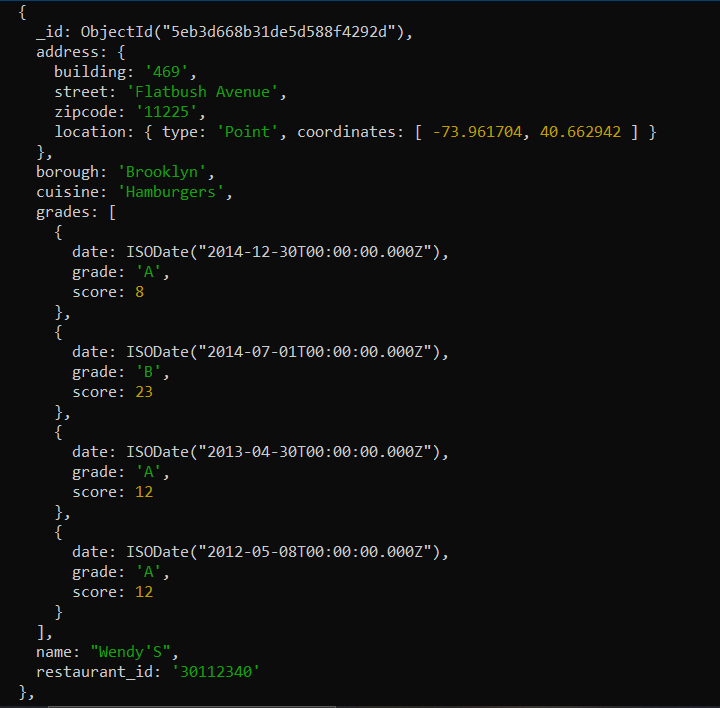
Slika 19. Kreiranje tekstualnog indeksa

Kako u kolekciji može postajati samo 1 tekstualni indeks, prilikom pretrage teksta nije potrebno specificirati polja u kojima se traži neki tekst, željeni tekst se traži po onim poljima koja su navedena prilikom kreiranja indeksa. U ovom slučaju to su polja “plot” i “fullplot.” Na sledećoj slici dat je upit kojim se vraćaju naslovi svih filmova koji u svom opisu sadrže reč “vacation” i prvih par rezultata koje vraća upit:

Slika 20. Rezultat upita po tekstualnom indeksu

## 4.3 Geoprostorni 2dsphere indeks

Sledeći indeks koji će biti demonstriran jeste **geoprostorni** indeks u MongoDB-ju. Za te potrebe iskorišćena je kolekcija *restaurants* čiji je jedan dokument prikazan na slici ispod. Kao što se vidi na slici, dokument poseduje ugnježdeno polje “adress.location” koje predstavlja dvodimenzionalne koordinate lokacije restorana.



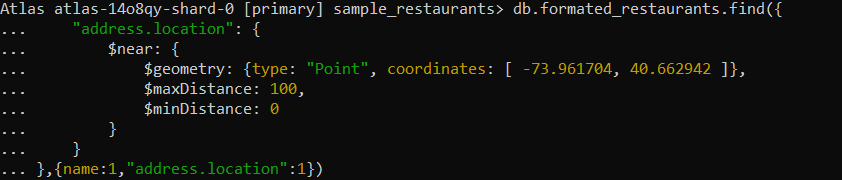
Slika 20. Primer dokumenta iz kolekcije *restaurants*

Geopstostorni *2d sphere* indeks kreiran je nad ovom kolekcijom na sledeći način:



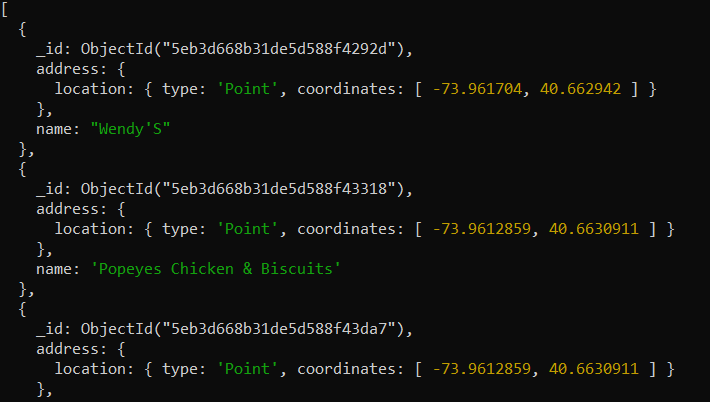
2dshere indeks jeste vrsta geoprostornog MongoDB indeksa kod koji se koristi za računanje geometrije na sferi poput Zemlje.

Ukoliko uzmemo proizvoljnu tačku, npr. tačku sa koordinatama [ -73.961704, 40.662942 ], možemo pomoću geoprostornog indeksa na efikasan način izračunati koji se sve restorani nalaze na rastojanju od na primer 100 metara od date tačke. Na sledećoj slici prikazan je upit za ovakvu pretragu.



Slika 21. Primer upita koji koristi geoprostorni indeks

Na sledećoj slici prikazani su nazivi i koordinate prvih nekoliko restorana koji odgovaraju ovakvoj pretrazi:



Slika 22. Rezultat upita

# Zaključak

Upotreba indeksa može značajno da utiče na performanse baze podataka. Ipak, treba ih kreirati pažljivo kako troškovi koji indeksi donose sa sobom ne bi zasenili prednosti njihove upotrebe. Zato treba dobro poznavati kako podatke i upite sa kojima se radi, tako i same indekse: načine na koji su implementirani i njihove osobine.

Cilj ovog rada bilo je bolje upoznavanje sa indeksima-vrstama indeksa, različitim načinima na koje mogu biti implementirani i rad sa indeksima demonstriran na praktičnim primerima. Praktični primeri demonstirani su u MongoDB bazi, koja podržava veliki broj različitih vrsta indeksa. MongoDB omogućava indeksiranje i višedimenzionalnih i tekstualnih podataka. Takođe, MongoDB omogućava da se za neki upite koristi i veći broj indeksa a ne samo jedan. Ipak, mora se paziti prilikom odabira koji će indeksi biti kreirani i na koji način, kako bi indeksi doneli više koristi nego štete.

# Literatura

* [Indexing in DBMS: What is, Types of Indexes with EXAMPLES](https://www.guru99.com/indexing-in-database.html)
* [R-Tree: algorithm for efficient indexing of spatial data](https://www.bartoszsypytkowski.com/r-tree/)
* [DBMS - Indexing](https://www.tutorialspoint.com/dbms/dbms_indexing.htm)
* [Understanding MongoDB Indexes - DEV Community](https://dev.to/jamiescript/understanding-mongodb-index-3mke)
* <https://www.mongodb.com/docs/manual/indexes/>