|  |  |  |
| --- | --- | --- |
| **Unigrb** | UNIVERZITET U NIŠU  ELEKTRONSKI  FAKULTET | **logo_1960_4** |

SISTEMI ZA UPRAVLJANJE

BAZAMA PODATAKA

SEMINARSKI RAD

**INTERNA STRUKTURA I ORGANIZACIJA INDEKSA KOD APACHE CASSANDRA-E**

|  |  |
| --- | --- |
| Mentor:  Aleksandar Stanimirović | Student:  Dragana Korunović  1220 |

Niš, april 2021. godine



[1. Uvod 1](#_Toc70068211)

[2. Indeksiranje 1](#_Toc70068212)

[2.1. Primarni indeks 1](#_Toc70068213)

[2.2. Grupišući indeks 2](#_Toc70068214)

[2.3. Sekundarni indeks 3](#_Toc70068215)

[3. Apache Cassandra 4](#_Toc70068216)

[3.1. Prednosti korišćenja 4](#_Toc70068217)

[4. Interna struktura 5](#_Toc70068218)

[4.1. Keyspace 6](#_Toc70068219)

[4.2. Tabela (familija kolona) 7](#_Toc70068220)

[4.3. Kolona 8](#_Toc70068221)

[4.4. Ključevi 9](#_Toc70068222)

[**4.5.** **O tabelama** 10](#_Toc70068223)

[4.5.1. Statička tabela 10](#_Toc70068224)

[4.5.2. Dinamička tabela 11](#_Toc70068225)

[4.5.3. Kombinovanje statičkih i dinamičkih karakteristika 12](#_Toc70068226)

[4.5.4. Korišćenje kompozitnih kolona 12](#_Toc70068227)

[4.5.5. Korišćenje kolekcijskih kolona 13](#_Toc70068228)

[4.5.6. Nekompaktne tabele 15](#_Toc70068229)

[5. Organizacija indeksa 16](#_Toc70068230)

[5.1. Sekundarni indeksi 17](#_Toc70068231)

[5.2. Indeksiranje kolekcije 18](#_Toc70068232)

[5.3. Korišćenje većeg broja indeksa 20](#_Toc70068233)

[6. Zaključak 21](#_Toc70068234)

[7. Literatura 22](#_Toc70068235)

# Uvod

Nekada su se pod bazom podataka podrazumevale biblioteke, poslovna, medicinska i slična dokumenta. Danas je, sa razvojem računara, ovaj pojam dobio drugačije značenje, kao i sve veću važnost. Baza podataka predstavlja integrisanu kolekciju podataka, koja je obično tako obimna da se mora čuvati na sekundarnim uređajima za skladištenje, kao što su diskovi ili trake. Umesto čuvanja u vidu datoteka operativnog sistema, podaci se mogu čuvati u DBMS-u (database management system) tj. sistemu za upravljanje bazama podataka. Osim skladištenja, baze podataka imaju i mogućnost indeksiranja i obezbeđivanja pristupa podacima. Sa razvojem baza podataka, sve je jednostavnije obaviti svaku od pomenutih aktivnosti, kao i jeftinije i brže.

Da bi se olakšao pristup podacima u okviru baze podataka, uvedeno je indeksiranje. Ono predstavlja kreiranje pomoćne strukture za pretragu podataka po određenom kriterijumu. Indeks može biti primarni, sekundarni i grupišući.

Sa pojavom svakodnevne količine podataka na internetu koja se meri u terabajtima, koji nemaju strogo definisanu strukturu i brzo se generišu (*big data*) javila su se rešenja kao što je Apache Cassandra. To je nerelaciona baza podataka, tj. ona bez strogo definisane šeme podataka, koja koristi indeks kao pomoćnu strukturu.

U narednom poglavlju biće predstavljen pojam indeksiranja i opisane vrsrte indeksa, dok je treće posvećeno osnovnim podacima o bazi podataka Apache Cassandra i prednostima njenog korišćenja. Četvrto poglavlje posvećeno je internoj strukturi pomenute baze, tj. načinu skladištenja unetih podataka, a peto organizaciji njenog indeksa, načinu njegovog kreiranja i pretrage po istom. Oba poglavlja uključuju primere isprobane u bazi podataka Cassandra, najnovije verzije 3.11.10. U poslednjem poglavlju dat je zaključak rada.

# Indeksiranje

Indeks je pomoćna pristupna struktura, koja se koristi za pretragu podataka koji zadovoljavaju određene kriterijume. Ova struktura se kreira nad jednim ili više polja nekog dokumenta. Različiti tipovi indeksa koriste različite strukture podataka, radi ubrzanja pretrage. Najrasprostranjeniji su indeksi bazirani na uređenim dokumentima (jednonivovski), strukturi stabla (višenivovski indeks, B+ stabla), a mogu se zasnivati i na hash tabelama i drugim strukturama podataka.

U slučaju postojanja jednog polja po kome se vrši indeksiranje, indeks se sastoji od različitih vrednosti tog polja, od kojih svaka poseduje listu pokazivača na sve blokove na disku koji je sadrže. Ove vrednosti su sortirane (uređene), tako da se može izvršiti binarna pretraga indeksa. Binarna pretraga nije potrebna kod višenivovskih indeksa, pošto se kreiraju indeksi nad indeksom. Postoje tri vrste uređenih indeksa – primarni, grupišući i sekundarni.

## Primarni indeks

Primarni indeks je uređeni dokument koji se sastoji od parova polja istog tipa kao uređujuće polje (primarni ključ dokumenta) i polja pokazivača na blok na disku čiji prvi element sadrži dato polje. Primarni indeks nad uređenim dokumentom sa slike 1 dat je na slici 2. Polje *name* je primarni ključ, po kome je dokument sortiran.

|  |  |
| --- | --- |
|  |  |
| Slika 1 – blokovi uređenog (sekvencijalnog) dokumenta sa podacima o zaposlenima | Slika 2 – primarni indeks nad uređujućim poljem dokumenta sa slike |

Primeri elemenata indeksa su sledeći:

*<K(1) = (Aaron,Ed), P(1) = adresa bloka 1>*

*<K(2) = (Adams,John), P(2) = adresa bloka 2>*

Ovakav indeks poseduje manje elemenata nego što je blokova u dokumentu sa podacima. Takođe, više elemenata indeksa staje u jedan blok, pošto su manji od stavki dokumenta. Zbog toga binarno traženje nad indeksnim dokumentom zahteva manje pristupa blokovima nego nad dokumentom sa podacima. Ako indeksni dokument sadrži bi blokova, da bi se pronašao element sa određenom vrednošću ključa, potrebna je binarna pretraga indeksa i pristup bloku koji sadrži traženi element – log2bi + 1 pristupa. Ako primarni ključ elementa ima vrednost K, on se nalazi u bloku sa adresom P(i), gde je K(i) ≤ K < K(i+1). Svi elementi koji imaju vrednost ključa K nalaze se u i-tom bloku dokumenta sa podacima, pošto se sortiranje vrši po tom ključu.

## Grupišući indeks

Grupišuće polje je ono koje nije ključ i nema jedinstvenu vrednost za svaki element. Grupišući indeks se sastoji od parova koji se sastoje od grupišućeg polja jedinstvene vrednosti i pokazivača na prvi blok koji sadrži element sa tom vrednošću polja. ubrzava pretragu po ovom polju. Primer je dat na slici 3.

|  |  |
| --- | --- |
|  |  |
| Slika 3 – grupišući indeks nad poljem Deptnumber dokumenta Employee | Slika 4 – grupišući indeks sa posebnim skupom blokova za svaku grupu elemenata koji poseduju istu vrednost grupišućeg polja |

Da bi upis i brisanje podataka iz indeksa bio što efikasniji, često se ceo blok ili skup susednih blokova rezerviše za svaku od vrednosti grupišućeg polja, gde su svi elementi koji sadrže tu vrednost smešteni u blok ili skup blokova (Slika 4). Traženje po ovakvom indeksu koristi vrednost polja po kome se vrši pretraga.

## Sekundarni indeks

Sekundarni indeks omogućava dodatni način pristupa dokumentu, kada već postoji primarni indeks. Polje nad kojim sekundarni indeks može biti je kandidat za ključ sa jedinstvenom vrednošću za svaki element ili polje koje nije ključ sa vrednostima koje nisu jedinstvene. Element sekundarnog indeksa se sastoji od indeksnog polja, koje je iste vrste kao neko nesortirajuće polje dokumenta sa podacima, i pokazivača na blok ili element dokumenta. Jedan dokument može da ima više sekundarnih indeksa.

Najpre posmatramo slučaj kada je sekundarni indeks nad poljem ključem, koje ima jedinstvenu vrednost za svaki element. Svakom elementu dokumenta sa podacima odgovara jedan element indeksa, koji sadrži vrednost sekundarnog ključa i pokazivač na blok u kome se nalazi element dokumenta ili sam element. Neka je svaki element indeksa predstavljen kao par <K(i), P(i)>. Moguće je koristiti binarno traženje, pošto su elementi sortirani po vrednosti K(i). Takođe se koriste pokazivači na svaki element dokumenta, pošto elementi dokumenta sa podacima nisu fizički sortirani po vrednostima sekundarnog ključa. Slika 5 ilustruje ovakav indeks.

|  |  |
| --- | --- |
|  |  |
| Slika 5 – sekundarni indeks nad neuređujućim poljem ključem dokumenta | Slika 6 – sekundarni indeks nad poljem koje nije ključ, implementiran korišćenjem indirekcije u jednom nivou |

Iako sekundarni indeks zahteva više prostora na disku i duže vreme pretrage, nasumičan pristup elementu dokumenta je brži, jer bi bez sekundarnog indeksa moralo da se vrši linearno traženje nad dokumentom.

Drugi slučaj predstavlja sekundarni indeks nad poljem koje nije ključ dokumenta. Tada više elemenata dokumenta sa podacima može da ima istu vrednost indeksnog polja. Postoji više načina implementacije ovog indeksa:

1. Uključiti više elemenata indeksa sa istom vrednošću ključa – K(i), jedan za svaki element dokumenta.
2. Elementi promenljive dužine sa poljem pokazivača. Lista pokazivača <P(i,l), ... , P(i,k)> na blokove se nalazi u elementu indeksa za K(i). To su blokovi koji sadrže element čije je indeksirajuće polje po vrednosti jednako K(i).
3. Za svaku vrednost indeksnog polja kreirati jedan element indeksa, dok se pokazivačima barata na poseban način. U paru <K(i), P(i)>, pokazivač P(i) pokazuje na blok pokazivača na elemente dokumenta koji poseduju vrednost K(i) kao indeksirajuće polje. Ako broj pokazivača premašuje veličinu bloka, koristi se skup ili lančana lista blokova (Slika 6). Sada pristup zahteva dodatni pristup bloku zbog uvedenog nivoa indirekcije, ali se pristupi pod kompleksnim uslovima za selekciju vrše na osnovu pokazivača na elemente, bez njihovog nepotrebnog preuzimanja.

# Apache Cassandra

Baza podataka Cassandra je distribuirana, open-source nerelaciona baza podataka. Ova baza pruža visoke performanse, stalnu dostupnost i skalabilnost, kao i hiljade istovremenih operacija nad podacima, koji se mogu meriti u petabajtima. Replikacija se odvija na nivou hibridnih i multicloud okruženja. Cassandra objedinjuje model distribucije baze Dynamo i komponentu za skladištenje log podataka, preuzetu od Google BigTable-a. Kreirana je 2008. godine pod okriljem Facebook-a, dok je 2009. godine njen razvoj preuzela kompanija Apache. Sledeće godine, Cassandra je postala top-level projekat ove kompanije. Može se besplatno preuzeti pod licencom Apache License 2.0, a najnovija verzija, 3.0, takođe je besplatna i razijena je od strane firme DataStax.

## Prednosti korišćenja

Cassandra obezbeđuje mnoge usluge, kao što su obrada log podataka servera, email-ova, pdf dokumenata ili postova na društvenim mrežama i dr. Neke od prednosti korišćenja su sledeće:

* *Open-source* – zbog pristupačnosti i proširljivosti, Cassandra je jedan od poznatih open-source softvera. Ova karakteristika garantuje brži razvoj inovacija i prihvatanje od strane inženjera.
* *Prilagodljiv i uobičajen interfejs –* upitni jezik koji Cassandra koristi je CQL (Cassandra Query Language), sličan SQL-u, što mnogim programerima omogućava lako učenje i korišćenje.
* *Visoke performanse –* svaki čvor[[1]](#footnote-1) u arhitekturi Cassandra-e ima mogućnost da obavlja sve operacije upisa i čitanja. Zbog toga se performanse poboljšavaju, kao i oporavak od grešaka.
* *Konstantno aktivna –* podaci u okviru baze Cassandra se brzo repliciraju u okviru hibridnih cloud okruženja i geografskih lokacija. Razlog tome je sposobnost svakog čvora da vrši operacije čitanja i pisanja. U slučaju da neki čvor otkaže, korisnici se automatski preusmeravaju na najlbliži zdravi čvor. Zbog toga su podaci i aplikacije koje ih koriste uvek dostupni. Takođe, Cassandra poseduje i ugrađene servise koji rešavaju probleme u roku od sat vremena od njihovog nastanka.
* *Skalabilnost –* Cassandra omogućava horizontalnu skalabilnost, koja ne zahteva skupe mašine, pošto se svodi na dodavanje novih čvorova u klaster[[2]](#footnote-2). Tako se operacije odvijaju brže.
* *Neprekidna* *replikacija –* sve više kompanija preusmerava se na multi-data centre, hibridne cloud i multi-cloud sisteme. Razlog tome je mogućnost korišćenja prednosti različitih sistema, koji nisu vezani za isključivo jednog proizvođača. Da bi se najbolje iskoristila multi-cloud okruženja, potrebno je posedovati cloud bazu podataka koja omogućava skalabilnost, sigurnost, performanse i dostupnost.

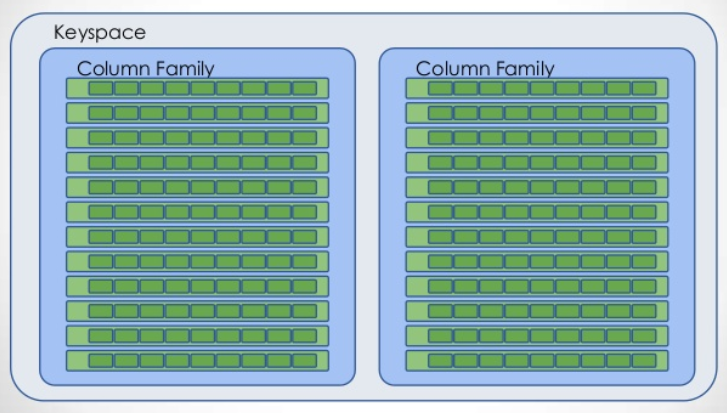
# Interna struktura

Interna struktura Cassandra-e definisana je uz pomoć njenog modela podataka (Slika 7). Ovaj model podataka poseduje keyspace, familije kolona, ključeve i kolone. Paralela koja se može povući između ovog i relacionog modela prikazana je u tabeli 1.

|  |  |
| --- | --- |
| Relacioni model | Cassandra-in model |
| Baza podataka | Keyspace |
| Tabela | Tabela (Familija kolona) |
| Primarni ključ | Ključ vrste |
| Ime kolone | Ime/ključ kolone |
| Vrednost kolone | Vrednost kolone |

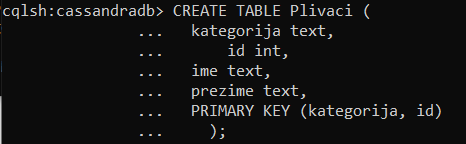
Tabela 1 – analogija između relacionog i Cassandra-inog modela podataka

Cassandra-in model podataka koristi pristup usmeren ka upitima, što znači da se podaci organizuju u skladu sa njima. Najbolji način pisanja upita je da pristupaju jednoj tabeli, tako da se entiteti koji su povezani nalaze u istoj. Sa druge strane, u relacionim bazama podataka se koriste strani ključevi za pristup povezanim entitetima. Cassandra vrši denormalzaciju podataka, što znači da se isti podaci nalaze u više tabela. Samo u slučaju da tabela ne može pokriti kompleksnost veza između entiteta se vrši join operacija u samoj klijentskoj aplikaciji. Zbog modela podataka i načina pisanja upita u Cassandra-i, oni se mogu brže izvršavati.



Slika 7 – model podataka Cassandra-e

Kao primer dat je upit koji pronalazi plivača koji se takmiči u određenoj kategoriji (Slika 8). Zbog efikasnosti, biciklisti se grupišu po kategoriji, pa se primarni ključ sastoji od kolone kategorija (particioni ključ) i od identifikatora u okviru nje, id-a (grupišući ključ). U nastavku će biti obrađen svaki od elemenata Cassandra-inog modela podataka.



Slika 8 – tabela Plivači

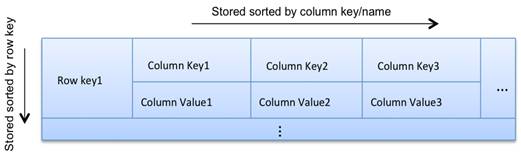
## Keyspace

Keyspace je ekvivalent šemi u relacionoj bazi podataka. Pre kreiranja tabela je neophodno kreirati keyspace, pri čemu svaka može pripadati samo jednom od njih. Pri kreiranju keyspace-a definiše se klasa, koja označava njegovu namenu i faktor replikacije. Faktor replikacije određuje koliko se kopija podataka kreira. U primeru (Slika 9) se pomoću CQL shell-a i jezika CQL3[[3]](#footnote-3) kreira keyspace *CassandraDB*, koji se upotrebljava za isprobavanje rada Cassandra-e, zbog čega je njegova klasa *SimpleStrategy*. Neće se kreirati kopije podataka keyspace-a, jer je faktor replikacije 1.

|  |  |
| --- | --- |
|  |  |
| Slika 9 – kreiranje keyspace-a | Slika 10 – kreiranje tabele |

## Tabela (familija kolona)

Svaka tabela može da se posmatra kao ugnježdena sortirana mapa (Slika 11). Zbog toga je moguća efikasna pretraga po ključevima, pri čemu je broj ključeva kolona neograničen. Tabela se sastoji od više vrsta i kolona (Slika 12), pri čemu je značajno definisati njen primarni ključ. On predstavlja kolonu koja ima jedinstvenu vrednost za svaku vrstu u tabeli. U primeru sa slike 10 možemo videti da je primarni ključ kolona id, zbog toga što ime i prezime nisu dovoljni da jedinstveno identifikuju korisnika. Na primeru kreiranja tabele (Slika 13) može se videti i njena interna struktura, tj. da je prvi deo definisanog primarnog kluča ključ vrste, drugi je ključ kolone i svaka od kolona sadrži svoj naziv (atribut koji nije primarni ključ) i vrednost. Vrste tabele mogu biti uske ili široke. Prve poseduju fiksan i relativno mali broj ključeva kolona, dok je kod drugih taj broj veliki (čak hiljade).



*Slika 11 – model tabele*

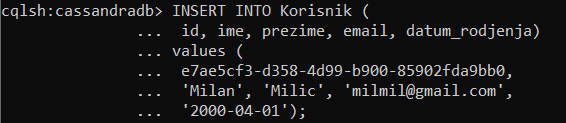
|  |  |
| --- | --- |
|  |  |
| Slika 12 – vrsta tabele | Slika 13 – izgled kreirane tabele |

## Kolona

Kolona predstavlja jedinicu podatka u Cassandra-i i poseduje polja ime, vrednost, timestamp i time-to-live (Slika 12). Ime kolone naziva se komparatorom, a vrednost validatorom. Preporučljivo je specificirati ime kolone, osim u slučaju da ja tabela statička (podaci nisu deo imena kolone) ili ako redosled uređenja nije bitan. Ime kolone nije moguće ažurirati, dok je moguće ažurirati njenu vrednost. Vrednost kolone je definisanog tipa, koji može biti celobrojna vrednost (integer, big integer), tekst (text), decimalna (float, double) i logička vrednost (Boolean), kao i skup (set), lista i mapa. Vrednost može biti deo naziva kolone (koji čine ključ kolone), pa polje vrednosti kolone može da izostane. Umesto vrednosti kolone može da stoji tombstone, što znači da je ta vrednost obrisana i sprečava da se obrisani podatak povrati tokom čitanja. Polje timestamp predstavlja vreme izraženo u mikrosekundama, kada je poslednji put upisana ili ažurirana vrednost kolone. Time-to-live određuje još koliko je vrednost kolone validna i predstavljeno je brojem sekundi.

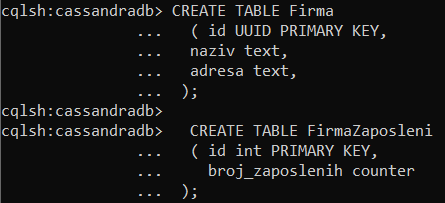
Kao specijalne vrste kolona izdvajaju se:

* UUID – 128-bitni celi broj (primer je vrednost kolone id sa slike 14),

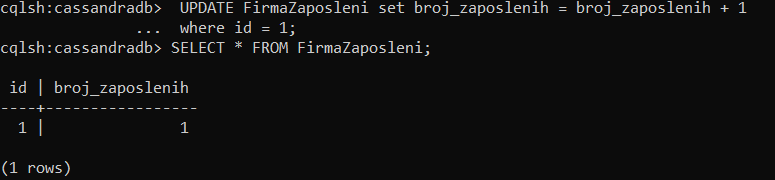
,

Slika 14 – ažuriranje tabele korisnik

* TimeUUID – jedinstvena vremenska oznaka dobijena u intervalu od 100 ns (na primer D2177dd0-eaa2-11de-a572-001b779c76e3),
* Counter (brojač) – specijalna kolona koja čuva broj koji se inkrementira. Inicijalna vrednost je 0, a vrednost se menja dodavanjem i oduzimanjem od trenutne vrednosti. Koristi se za brojanje pojava nekog događaja, procesa, entiteta. Na primer, kreirajući kolonu *broj\_zaposlenih* ovog tipa u tabeli *FirmaZaposleni* (Slika 15), inkrementiranjem podrazumevane vrednosti 0 dobija se vrednost 1, nakon izvršenja upita ažuriranja (Slika 16).



Slika 15 – kreiranje tabele koja služi za prebrojavanje zaposlenih u nekoj firmi



Slika 16 – inkrementiranje broja zaposlenih i prikaz ažurirane vrste

Zbog toga što Cassandra ne vrši operaciju *join* nad tabelama, javljaju se neprirodni načini skladištenja. Na primer, ako korisnik treba da ima mogućnost posedovanja više email adresa, u RDBMS-u bi se to modeliralo kreiranjem tabele *email\_adrese* sa vezom jedan na više ka tabeli *korisnici*. U Cassandra-i to se može postići kreiranjem više kolona – *email1, email2*... Iako iz ugla performansi nema razlike, ovaj pristup je neprirodan i zahteva čitanje pre dodavanja, da bi se znalo kako nazvati narednu kolonu. Zato su uvedene kolekcijske kolone. One služe da predstave skup podataka u jednoj koloni. Vrste kolekcijskih kolona su:

* *Set* – neuređen skup disjunktnih vrednosti, u vitičastim zagradama ( npr. { ‘Car’, ‘Truck,’Plane’ }). Elementi skupa se nakon upita vraćaju sortirani po vrednosti.
* *List* – uređen skup vrednosti (po redosledu unosa) koji mogu da se ponavljaju, poput niza, u okviru uglastih zagrada ( npr. [ ‘2011’, ‘2011’ ,‘2012’, ‘2013’ ]).
* *Map* – skup parova ključ-vrednost, u vitičastim zagradama, uređen po vrednostima ključeva ( npr. { ‘Audi’: ‘A6, ‘BMW: ‘i8 Roadster}).

Treba napomenuti da se kolekcija može preuzeti samo kao celina, pa zbog toga nije preporučljivo da bude previše velika. Trenutno ne postoji podrška za ugnježdene kolekcije, niti za formiranje sekundarnog indeksa nad kolekcijama. Kolekcijska kolona ne može biti deo primarnog ključa. U poglavlju 4.5.5. će biti reči o smeštanju kolona ovog tipa.

## Ključevi

Cassandra poseduje ključ pod nazivom primarni ključ, koji služi za jedinstveno identifikovanje elementa tabele i može se sastojati od jednog ili više polja. Ako se primarni ključ sastoji jednog polja, naziva se jednostavnim ključem i predstavlja ključ vrste, a ako se sastoji od više kolona, naziva se kompozitnim ključem. Prvi deo kompozitnog ključa koristi se za particionisanje podataka i naziva se particionim ključem ili ključem vrste. Podaci svake od particija su razmešteni u klasteru i poređani po vrednostima ostatka kompozitnog ključa (klučeva kolona), pod nazivom grupišući (clustering) ključ.

Ako je potrebno particionisati podatke po vrednostima više kolona, kolone treba da se grupišu kao prvi deo primarnog ključa, na primer:

*primary key((ključ1, ključ2), ključ3, ključ4) –* podaci su particionisani po ključevima 1 i 2 i grupisani po ključevima 3 i 4.

Pošto su vrste koje sadrže vrednosti kolone *ključ3* smeštene u isto fizičko skladište, izvršenje upita po vrednosti tog ključa je ubrzano (npr. za *ključ3 = 3*). Vrednost primarnog ključa se ne može menjati nakon unosa.

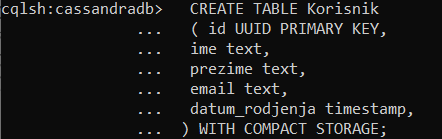
|  |  |
| --- | --- |
|  |  |
| Slika 17 – kreiranje tabele frizerski\_saloni | Slika 18 – ubacivanje podataka u tabelu frizerski\_saloni |

U primeru sa slike 17 kreirana je tabela frizerski\_saloni sa kompozitnim primarnim ključem. Nakon unosa podataka na slici 18, svaka particija poseduje particioni ključ tipa *drzava:okrug*, pa možemo posmatrati particiju *Srbija:Nisavski*. U okviru ove particije podaci su smešteni kao kolona *Nis:Hairstyle*, bez vrednosti, zatim kolona *Nis:Hairstyle:postanski\_broj*, sa vrednošću *18000* i kolona *Nis:Hairstyle:adresa*, uz vrednost *Vojvode Tankosica 5*.

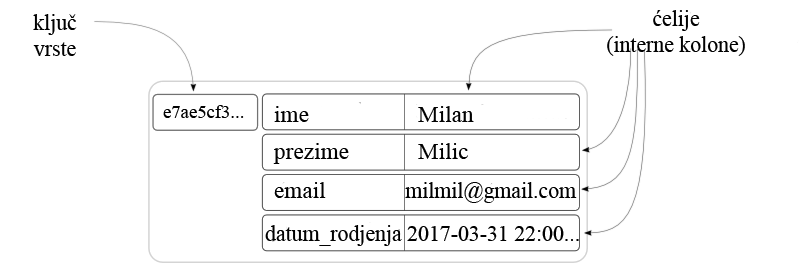
* 1. O tabe**lama**

### Statička tabela

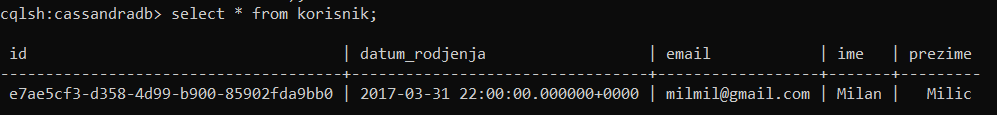
Statička tabela je ona u kojoj je isti, konačan skup ćelija prisutan u većini vrsta. To je najčešće slučaj kada se modeliraju profili korisnika neke aplikacije. Primer kreiranja statičke tabele *Korisnik* dat je na slici 19, a način smeštanja (interna reprezentacija) na slici 20. Svaka od internih kolona poseduje element tabele koji nije primarni ključ kao naziv i unetu vrednost te kolone u polju vrednosti. Rezultat SELECT upita dat je na slici 21.



*Slika 19 – kreiranje statičke tabele Korisnik*



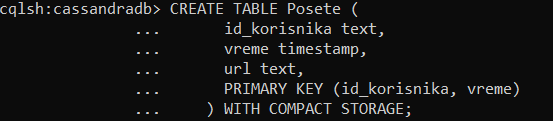
*Slika 20 – interna reprezentacija kreirane statičke tabele Korisnik*



*Slika 21 – preuzimanje tabele Korisnik*

### Dinamička tabela

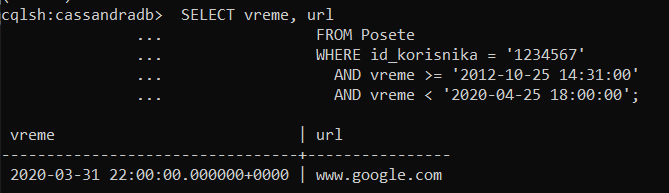
Dinamička tabela je ona u kojoj svaka vrsta može da sadrži potpuno različite skupove ćelija. Razlika u odnosu na statičku tabelu je kompozitni primarni ključ. U primeru sa slike 22, CQL3 mapira prvi deo primarnog ključa (*id\_korisnika*) na ključ interne vrste i drugi deo (*vreme*) na interno ime ćelije. Poslednji deo (*url*) se mapira na vrednost ćelije. Tako od jedne široke vrste nastaje više CQL3 vrsta, jedna po ćeliji te vrste (Slika 23). Moguće je preuzeti željene podatke u traženom opsegu (Slika 24).



*Slika 22 – kreiranje dinamičke tabele Posete*



*Slika 23 – interna reprezentacija kreirane dinamičke tabele Posete*



*Slika 24 – select iskaz nad tabelom Posete*

### Kombinovanje statičkih i dinamičkih karakteristika

U određenim slučajevima korisno je ubaciti dinamičke elemente u statičke tabele. Na primer, ako treba dodati oznake (tag-ove) profilu korisnika (sa slike 19), može da se kreira dinamička tabela, koja za svakog korisnika sadrži oznaku, ali i da se oznake dodaju direktno u tabelu *korisnik*. Druga opcija donosi prednost čitanja podataka o korisniku iz jedinstvene tabele, umesto dvaput, tj. iz dve tabele. Nakon ubacivanja imena ćelije *tag:admin* u tabelu, bez njene vrednosti, dobija se struktura sa slike 25.

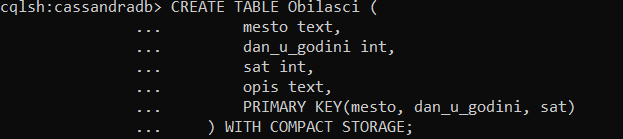


*Slika 25 – interna reprezentacija vrste tabele Korisnik*

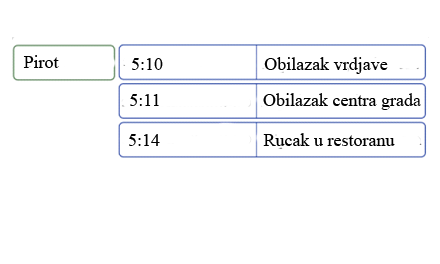
Takođe, skupovi su u CQL3-u implementirani kao i tag iz navedenog primera, tj. svaki element skupa je interno sačuvan kao posebna ćelija sa praznom vrednošću. Zbog toga je poželjno koristiti kolekcijske kolone pri kombinovanju statičkog i dinamičkog ponašanja (poglavlje 4.5.5).

### Korišćenje kompozitnih kolona

Jedan od slučajeva korišćenja dinamičkih tabela je onaj kada je kolona kompozitnog tipa. U primeru sa slike 26 kreirana je tabela o obilasku mesta *Obilasci*, koja poseduje kompozitni ključ. Način na koji su podaci smešteni nakon unosa je prikazan na slici 27, gde je primer kompozitnog imena ćelije *5:14* (*dan\_u\_godini*, *sat*). Kompozitni tip se kod CQL3-a mapira na jednu kolonu za svaku komponentu imena kompozitne ćelije, dok se vrednost nalazi u polju vrednosti.



Slika 26 – kreiranje kompaktne tabele Obilasci

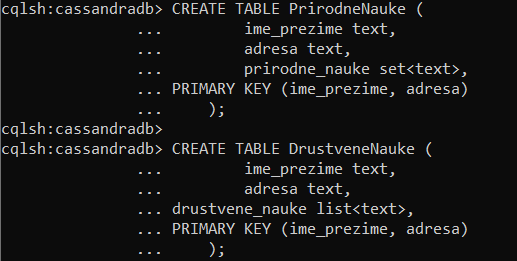


Slika 27 – interna struktura tabele Obilasci

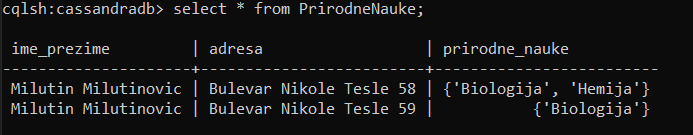
### Korišćenje kolekcijskih kolona

Kolekcijske kolone predstavljaju skup podataka u jednoj koloni koji je ograničenog kapaciteta. U nastavku su dati primeri tabela koje ih koriste, uz njihovu internu strukturu nakon unetih podataka.

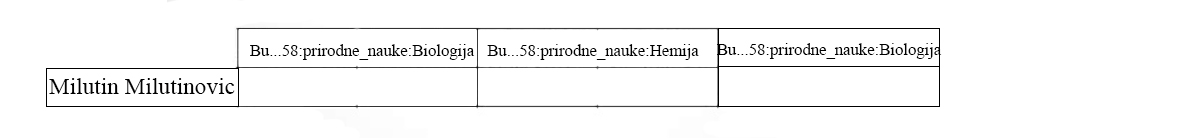
Nakon kreiranja tabele *Prirodnenauke* sa kolonom *prirodne\_nauke* tipa set (Slika 28) i unosa podataka, tabela koja se dobija je prikazana na slici 29, a način smeštanja je takav da se prvi deo primarnog ključa mapira na ključ vrste, a svako ime kolone se sastoji od vrednosti grupišućeg ključa *adresa*, naziva kolekcijske kolone – *prirodne\_nauke* i odgovarajuće vrednosti te kolone (Slika 30).



Slika 28 – kreiranje tabele Prirodnenauke

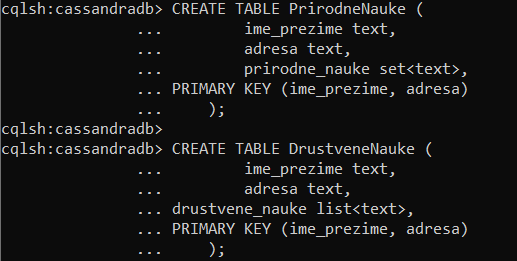


Slika 29 – podaci tabele Prirodnenauke

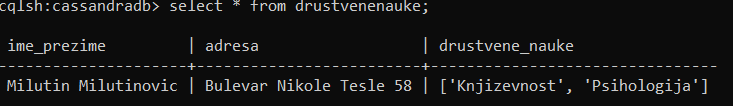


Slika 30 – interna struktura tabele Prirodnenauke

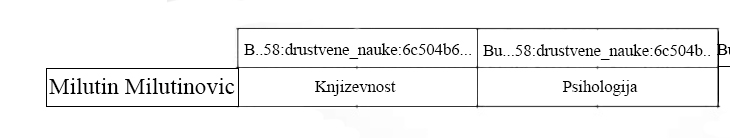
Sledeći primer uključuje kolonu tipa list, *drustvene\_nauke*, u tabeli *Drustvenenauke* (Slika 31). Nakon ubacivanja podataka, izgled tabele prikazan je na slici 32, a interna struktura na slici 33. Može se videti da se ime svake od kolona sastoji od vrednosti grupišućeg ključa *adresa*, imena kolone *drustvene\_nauke* i jedinstvene vrednosti UUID, dodeljene da bi se podaci u listi mogli sortirati. Vrednost svake kolone je uparena sa njenim nazivom i predstavlja element liste.



Slika 31 – kreiranje tabele Drustvenenauke

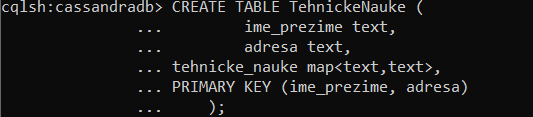


Slika 32 – podaci tabele Drustvenenauke

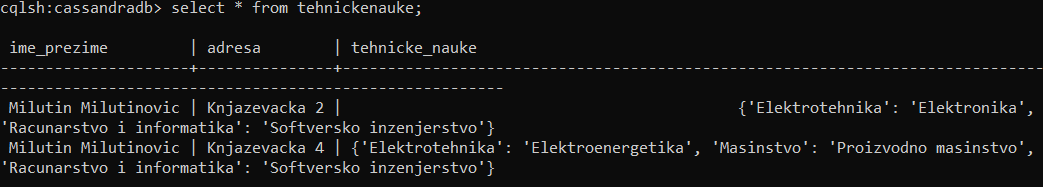


Slika 33 – interna struktura tabele Drustvenenauke

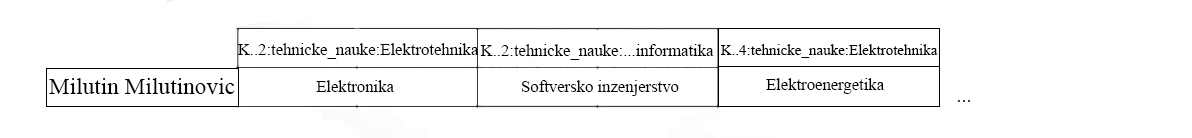
Primer tabele sa slike 34 uključuje kolonu *tehnicke\_nauke*, koja je tipa map. Pošto su ubačeni probni podaci, na slici 35 može se videti kreirana tabela sa svojim sadržajem, a interna struktura je takva da se naziv kolone sastoji od vrednosti grupišućeg ključa *adresa*, imena kolekcijske kolone *tehnicke\_nauke* i ključa elementa mape. Vrednost svake od kolona predstavlja vrednost u mapi uparenu sa odgovarajućim ključem (Slika 36).



Slika 34 – kreiranje tabele Tehnickenauke



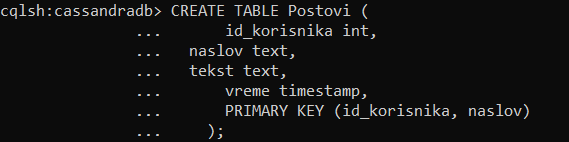
Slika 35 – podaci tabele Tehnickenauke



Slika 36 – interna struktura tabele Tehnickenauke

### Nekompaktne tabele

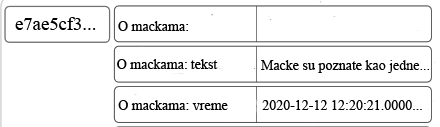
Nekompaktne tabele ne poseduju ograničenje da sme da postoji jedan ključ vrste i jedan kolone (kompozitan), kao i jedna vrednost kolone sa podacima. Takođe, dozvoljavaju izmene i brisanje kolona. Na primeru sa slike 26 kreirana je nekompaktna tabela *Postovi*.



*Slika 37 – kreiranje nekompaktne tabele Postovi*

Kreirana tabela sadrži kompozitni primarni ključ i dve kolone koje nisu deo tog ključa. Interno, ova tabela koristi *CompositeType* komparator, gde se njegova prva komponenta mapira na kolonu *naslov*, a druga na ime kolone čijeg je tipa ćelija (*tekst, vreme*). Ovo je prikazano na slici 27.

Svaka CQL3 vrsta je mapirana na statički deo internih ćelija, gde je prva prazna ćelija rezervisana za detalje implementacije. Ovo rešenje olakšava korišćenje kolona tipa integer. Takođe je moguće ažurirati i brisati svaku kolonu posebno, pošto su podeljene na individualne ćelije.



*Slika 38 – interna struktura tabele Postovi*

Nekompaktne tabele su pogodne za nove korisnike jer omogućavaju izmene i naknadno dodavanje kolekcijskih kolona. Međutim, u slučaju skladištenja velikih količina podataka može doći do zauzeća i do 30 puta više dodatnog memorijskog prostora, pa treba biti oprezan pri njihovom korišćenju.

# Organizacija indeksa

Cassandra poseduje ugrađeni indeks, pri čemu on sme da sadrži samo jednu kolonu. Najbolje je kreirati indeks nad poljem koje nema puno distinktnih vrednosti, kao što je ime autora u bazi pesama. Takođe, Cassandra uvodi sekundarni indeks, da bi bio moguć pristup poljima koja nisu primarni ili particioni ključ. Na ovaj način je pristup podacima koji zadovoljavaju određeni kriterijum brz i efikasan.

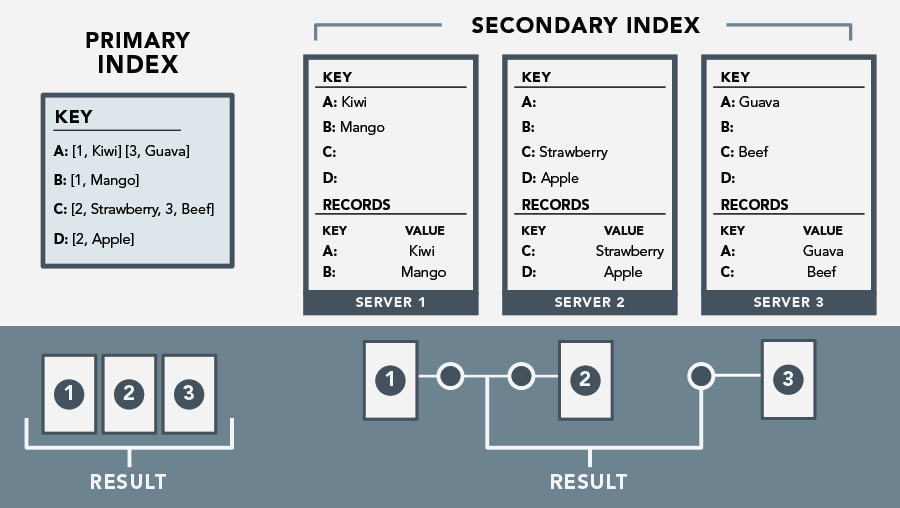
Indeks se kreira tako što se vrednosti kolona smeštaju u familiju kolona, koja je odvojena od izvorne i sakrivena. Selekcija vrednosti u određenom opsegu nije podržana jer su ključevi u skrivenoj familiji kolona neuređeni. Postoje situacije u kojima nije preporučljivo koristiti indeks:

* Nad kolonama sa mnogo distinktinih vrednosti, jer se radi malog broja rezultata pretražuje velika količina podataka, veće je kašnjenje pri izvršenju upita i održavanju indeksa.
* Nad kolonama sa malo distinktnih vrednosti, kao što je kolona tipa *boolean*, pošto bi se dobile dve potencijalno ogromne vrste (ako ima mnogo elemenata tabele) koje čine indeks, za vrednosti *true* i *false*.
* Nad tabelama koje koriste brojačke kolone (*counter*).
* Nad kolonom čije se vrednosti često menjaju ili brišu, jer na mestu obrisane vrednosti ćelije u indeksu stoji specijalna vrednost *tombstone*, pri čemu nakon dostizanja 100 hiljada takvih ćelija upit koji ih koristi neće moći da se izvrši.
* Pri pronalaženju vrste u velikoj particiji, osim pri suženoj oblasti pretrage. Ako je klaster veliki, a oblast pretrage nije sužena, odgovor na upit je usporen. To se dešava iz razloga što se odgovori više particija podataka pribavljaju, pa stapaju u jedan.

U nastavku će biti reči o sekundarnim indeksima.

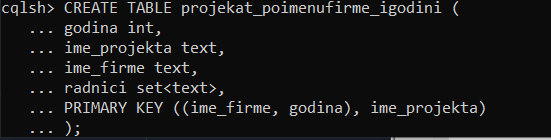
## Sekundarni indeksi

Sekundarni indeksi su oni koji su formirani nad vrednostima kolona. To su kolone koje ne pripadaju primarnom indeksu, a potrebna je pretraga po njihovoj vrednosti. Moguće je indeksirati i kolekcijske kolone. Ovi indeksi su jednostavni za kreiranje i efikasni nad skupovima podataka sa malom kardinalnošću. Međutim, korišćenje sekundarnih indeksa može negativno da utiče na performanse. Kod sekundarnog indeksa, za razliku od primarnog, svaki čvor u klasteru poseduje indeksnu tabelu, pa se upit mora izvršiti nad svakom od njih (Slika 39). Zbog toga je preporučljivo da se ineksira kolona sa nekoliko vrednosti male kardinalnosti.

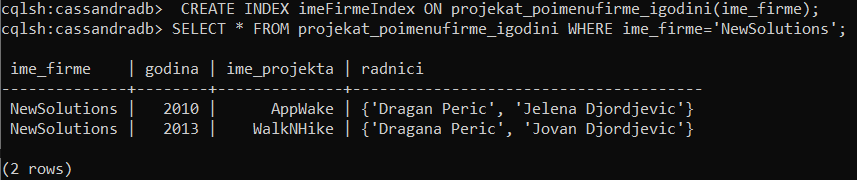


Slika 39 – primarni, sekundarni indeks i formiranje rezultata upita

Tabela sa slike 40 se može pretraživati po vrednosti particionog ključa, pri čemu se ne može navesti samo prvi deo – *ime\_firme*. Zbog toga se kreira sekundarni indeks nad tom kolonom, čije ime je opciono i treba da bude jedinstveno u okviru keyspace-a. Nakon toga je moguće izvršiti pretragu po vrednosti ove kolone (Slika 41).

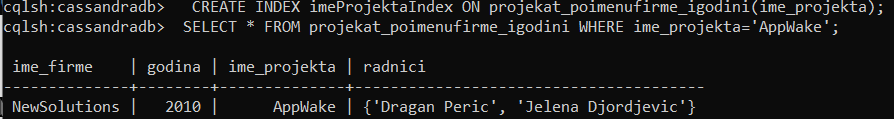


Slika 40 – kreiranje tabele projekat\_poimenufirme\_igodini



Slika 41 – kreiranje indeksa nad kolonom ime\_firme i pretraga po njemu

Moguće je kreirati indeks i nad grupišućom kolonom, što je u datom primeru kolona *ime\_projekta* (Slika 42).

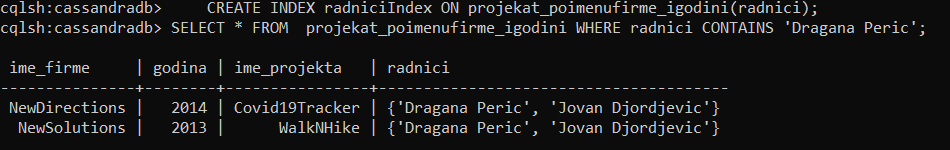


Slika 42 – kreiranje indeksa nad kolonom ime\_projekta i pretraga po njemu

## Indeksiranje kolekcije

Kolekcije (kolekcijske kolone) mogu da se indeksiraju i pretražuju da bi se našla ona koja sadrži određenu vrednost. Skupovi i liste mogu da indeksiraju sve vrednosti koje se nalaze u kolekcijskoj koloni. Mape mogu da indeksiraju ključ mape, vrednost mape ili element mape. Može se kreirati više indeksa nad istom kolonom mape u tabeli, tako da se može pretraživati po indeksiranim poljima. Takođe, kolekcije koje se nazivaju *frozen* (one koje se tretiraju kao celina pri manipulisanju njima) mogu da se indeksiraju korišćenjem ključne reči *FULL*, što znači da se indeksira celokupan sadržaj ovakve kolekcije.

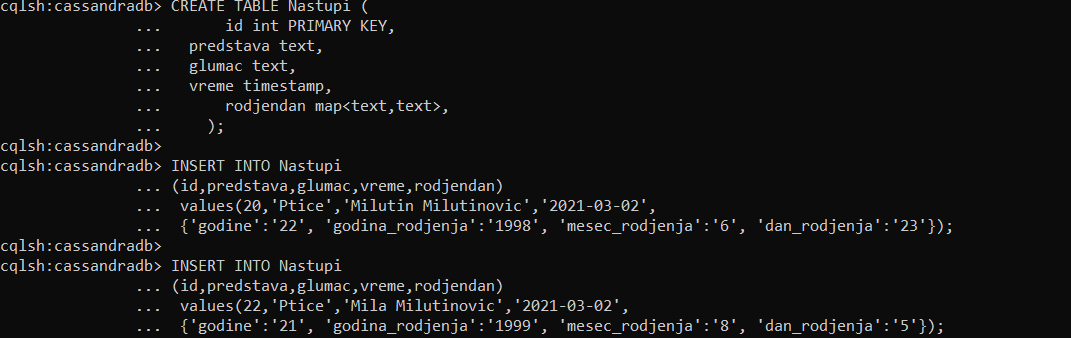
Prvi primer se odnosi na skupove i liste. Najpre se kreira indeks nad kolekcijskom kolonom radnici, koja sadrži skup imena radnika, a zatim se vrši upit koji vraća sve vrste koje sadrže vrednost Dragana Peric unutar te kolone (Slika 43).



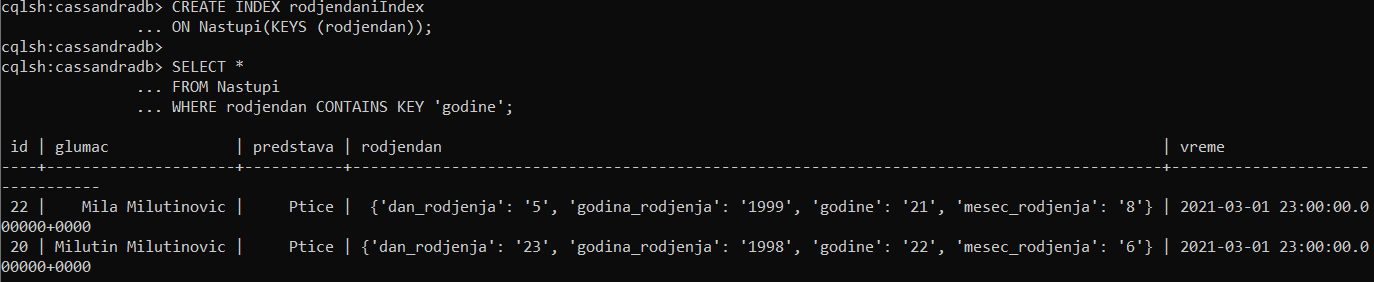
Slika 43 – kreiranje indeksa nad kolekcijskom kolonom radnici i pretraga po vrednosti koju treba da sadrži

Sledeći primeri odnose se na map kolekciju. Postoje tri mogućnosti za standardnu i jedna za mapu tipa frozen:

1. Kreira se indeks nad ključem mape *rodjendan* da se pronađu podaci o rođenju glumca koji se nalaze u svakom od unesenih nastupa. Oba unesena nastupa sadrže iste podatke o rodjenju glumca, ali to ne mora biti slučaj (Slika 44). U datom primeru pronalaze se nastupi gde je navedena starost glumca u okviru mape *rodjendan* (Slika 45).

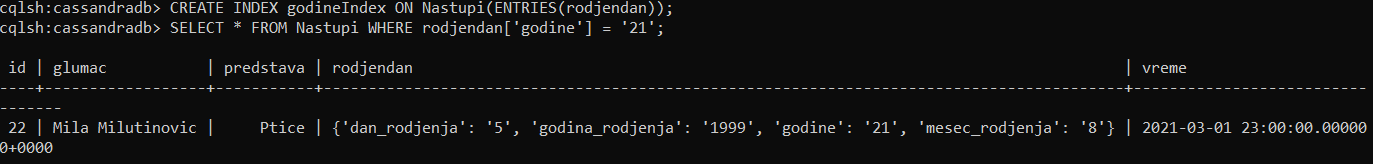


Slika 44 – kreiranje i unos podataka u tabelu Nastupi

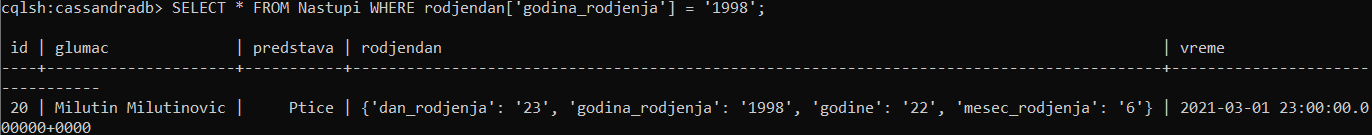


Slika 45 – pretraga po vrednosti ključa mape rodjendan

1. Indeks se kreira nad unosima mape *rodjendan* (entries), da bi se pronašli nastupi glumaca po vrednosti određenog unosa u mapu. Prvi upit pronalazi nastupe glumaca starih 21 godinu (Slika 46). Koristeći isti indeks, mogu se pronaći svi nastupi glumaca čija je godina rođenja 1998. (Slika 47).

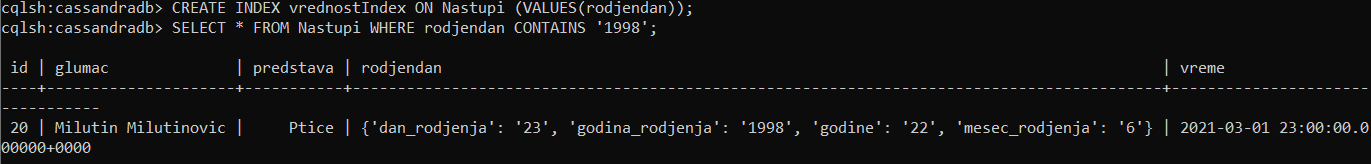


Slika 46 – kreiranje indeksa nad unosima mape rodjendan i pronalaženje nastupa glumca po njegovoj starosti



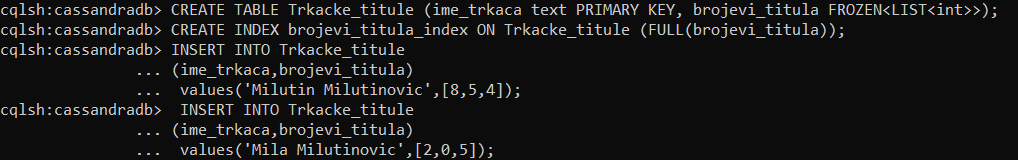
Slika 47 – pronalaženje nastupa glumca po njegovoj godini rođenja

1. Kreira se indeks nad vrednostima mape *rodjendan*, a zatim se pronalazi nastup koji u okviru te mape sadrži vrednost 1998. (Slika 48).

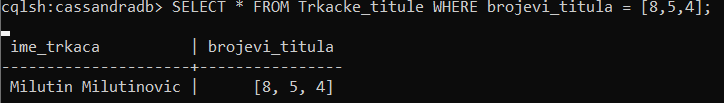


Slika 48 – pronalaženje nastupa glumca po vrednosti sadržanoj u mapi rođendan

1. Indeks se kreira nad frozen kolonom *brojevi\_titula* (lista) tabele *Trkacke\_titule* (Slika 49), pa se može pretraživati po vrednostima ove liste (Slika 50).



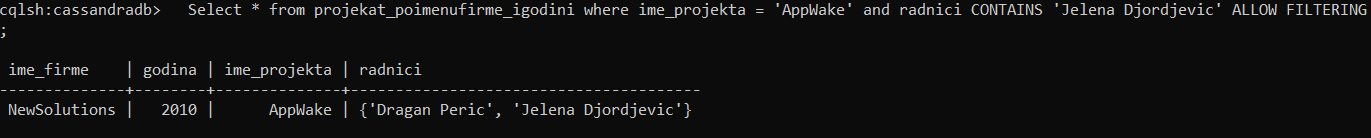
Slika 49 – kreiranje tabele Trkacke\_titule, indeksa nad celom frozen kolonom brojevi\_titula i ubacivanje podataka



Slika 50 – pretraživanje tabele Trkacke\_titule po vrednosti frozen kolone brojevi\_titula

## Korišćenje većeg broja indeksa

Moguće je vršiti upite po vrednostima više definisanih indeksa. U primeru tabele *projekat\_poimenufirme\_igodini* sa slike 40 kreirani su indeksi nad poljuma *ime\_projekta* i *radnici* (slike 42 i 43). Zbog toga se vrši upit koji pronalazi sve projekte čije je ime *AppWake*, a sadrže radnika po imenu *Jelena* *Đorđević*. Direktiva *ALLOW FILTERING* se koristi jer upit može biti zahtevniji, pošto nije naveden particioni ključ u upitu (Slika 51).



Slika 51 – pronalaženje projekta po njegovom imenu i imenu radnika

# Zaključak

Zbog potrebe za zajedničkim skladištem sličnih podataka nastale su baze podataka, čiji je razvoj pratio razvoj interneta. Tako su nastale nerelacione baze podataka, kao što je Cassandra, koje ne zahtevaju da podaci koji se skladište budu strogo definisane strukture. Takođe, zbog potrebe brze i efikasne pretrage podataka, nastale su pomoćne strukture pod nazivom indeksi. Njihov tip i način upotrebe prilagođeni su svakom od rešenja koje ih koristi.

Osim reči o indeksnim strukturama, pomenute su prednosti korišćenja DBMS-a Cassandra. Cassandra je jednostavnog dizajna, sa mogućnošću horizontalnog skaliranja i kontrolisanja dostupnosti podataka. Ova baza podataka je otporna na otkaze delova sistema i omogućava brz upis i čitanje. Zbog mogućnosti baratanja velikom količinom podataka, koriste je kompanije kao što su Apple, Facebook, Netflix i Cisco.

Najveći deo rada posvećen je internoj strukturi ove baze i njenom indeksu. Opisan je model podataka koji Cassandra koristi i svi njegovi elementi. Prikazane su razlike između izgleda tabele koja je prikazana korisniku i načina na koji je ona interno smeštena. Kroz primere je približen svaki od tipova tabela i kolona. Zatim su opisane vrste indeksa koje se koriste u Cassandra-i i njihovi slučajevi korišćenja su dati i kroz primere. Navedeno je i kada je nepoželjno koristiti indeksiranje.

Iako rad ne pokriva svaki od aspekata Cassandra-e, predstavljene su njene osnovne prednosti i jednostavnost korišćenja, kao i primeri njene primene. Razvoj ovog DBMS-a se nastavlja i očekuje se porast njegovog korišćenja, vođen primerom jednih od najvećih kompanija današnjice.

# Literatura

1. *What is Cassandra?*, dostupno na: <https://www.datastax.com/cassandra> (pristupljeno 25. 3. 2021)
2. Ramakrishnan, R., Gehrke J. (2002): *Database Management Systems*, 3rd edition, dostupno na: <http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/solutions/ans3ed-oddonly.pdf> (pristupljeno 28. 3. 2021)
3. Berg, K., Seymor, T. (2012): *History Of Databases*, dostupno na: <https://www.researchgate.net/publication/298332910_History_Of_Databases> (pristupljeno 28. 3. 2021)
4. Emasri R., Navathe S. (2010) *Fundamentals Of Database Systems*, dostupno na: <https://seu1.org/files/level6/IT344/Fundamentals_of_Database_Systems,_6th_Edition.pdf> (pristupljeno 28. 3. 2021)
5. *DataStax Documentation*, dostupno na: https://docs.datastax.com/en/cql-oss/3.3 (pristupljeno 8. 4. 2021)
6. *A Thrift to CQL3 Upgrade Guide,* dostupno na: <https://www.datastax.com/blog/thrift-cql3-upgrade-guide> (pristupljeno 8. 4. 2021)
7. *Learn Cassandra*, dostupno na: <https://teddyma.gitbooks.io/learncassandra/content/> (pristupljeno 8. 4. 2021)
8. Apache Cassandra Certification Training, dostupno na: <https://www.simplilearn.com/cassandra-data-model-tutorial-video> (pristupljeno 9. 4. 2021)
9. *Coming in 1.2: Collections support in CQL3*, dostupno na: <https://www.datastax.com/blog/coming-12-collections-support-cql3> (pristupljeno 9. 4. 2021)
10. *Cassandra at Scale: The Problem with Secondary Indexes,* dostupno na: https://pantheon.io/blog/cassandra-scale-problem-secondary-indexes (pristupljeno 10. 4. 2021)
11. *Understanding How CQL3 Maps to Cassandra's Internal Data Structure*, dostupno na: https://www.slideshare.net/DataStax/understanding-how-cql3-maps-to-cassandras-internal-data-structure (pristupljeno 10. 4. 2021)
12. *Cassandra Data Modeling Best Practices, Part 2*, dostupno na: <https://tech.ebayinc.com/engineering/cassandra-data-modeling-best-practices-part-2/> (pristupljeno 11. 4. 2021)
13. *Understanding How CQL3 Maps to Cassandra’s Internal Data Structure: Sets, Lists, and Maps*, dostupno na: https://opensourceconnections.com/blog/2013/07/24/understanding-how-cql3-maps-to-cassandras-internal-data-structure-sets-lists-and-maps/ (pristupljeno 12. 4. 2021)

1. Mašina na kojoj je Cassandra pokrenuta. [↑](#footnote-ref-1)
2. Kolekcija čvorova koji sadrže slične podatke. [↑](#footnote-ref-2)
3. The Cassandra Query Language (CQL) je osnovni jezik za komunikaciju sa Apache Cassandra bazom podataka, koja se najčešće odvija preko CQL shell-a, koji omogućava kreiranje keyspace-ova, tabela, vršenja upita nad njima i dr. Trenutna verzija CQL-a je CQL3, a podverzija 3.3. On pruža novi API za rad sa Cassandra-om, koji nudi sloj apstrakcije nad internom strukturom baze podataka. CQL3, za razliku od CQL-a, pruža model koji poseduje vrste i kolone sa istim značenjem kao u SQL-u. Interna vrsta sadrži ćelije, dok CQL3 vrsta sadrži kolone. [↑](#footnote-ref-3)