



UNIVERZITET U NIŠU  
ELEKTRONSKI FAKULTET  
Katedra za računarstvo



## **INTERNA STRUKTURA I ORGANIZACIJA INDEKSA KOD APACHE CASSANDRA BAZE PODATAKA**

Predmet: Sistemi za upravljanje bazama podataka  
-Seminarski rad-

Student:

Marijana Cvetković, br. ind. 1431

Mentor:

Doc. dr Aleksandar Stanimirović

Niš, april 2022. godina

# Sadržaj

1. Uvod .....	3
2. Apache Cassandra .....	3
2.1. Interna struktura .....	4
2.2. Koncept indeksiranja.....	6
2.2.1. Korišćenje sekundarnog indeksa .....	7
2.2.2. Korišćenje više indeksa.....	8
2.2.3. Indeksiranje kolekcija .....	9
2.2.4. Problemi sa korišćenjem indeksa .....	11
2.2.5. Primeri primene .....	12
3. Praktična primena .....	14
3.1. Pokretanje Cassandra baze podataka.....	14
3.2. Skup podataka.....	16
3.3. Unos i upotreba podataka .....	16
4. Zaključak .....	19
5. Literatura.....	20

## 1. Uvod

Apache Cassandra je NoSQL distribuirana baza podataka otvorenog koda. Predstavlja particionisani model skladištenja širokih kolona sa konačno doslednom semantikom. Po dizajnu NoSQL baze podataka su lagane, otvorenog koda, nerelacione i u velikoj meri distribuirane. Cassandra baze podataka lako se skaliraju kada je aplikacija pod velikim stresom, a distribucija takođe sprečava gubitak podataka usled kvara hardvera bilo kog data centra. Distribuirano znači da Cassandra može da radi na više mašina dok se korisnicima čini kao jedinstvena celina.

U ovom radu biće detaljno opisana interna struktura Cassandra baze podataka, kao i organizacija indeksa i dati primeri upotrebe.

U praktičnom delu ovog rada biće prikazana upotreba indeksa nad skupom podataka koji se odnose na zakazivanje koncerata, prikazan način kreiranja i dodavanja podataka...

Na kraju, u poslednjem poglavlju, nalazi se zaključak, izveden na osnovu teorijskog i praktičnog dela rada. Biće pomenute prednosti i mane korišćenja ove baze podataka i rada sa njom.

## 2. Apache Cassandra

Cassandra je open source distribuirani sistem za upravljanje bazom podataka. Ova baza podataka podržava upravljanje velikom količinom podataka. Razvijena je od strane Facebook-a, za potrebe inbox search-a, koristeći arhitekturu vođenu događajima za implementaciju kombinacije Amazanovih Dinamo tehnika distribuiranog skladištenja i replikacije Google-vog BigTable modela podataka i mehanizma za skladištenje podataka. I Dinamo i BigTable razvijeni su da zadovolje nove zahteve za skalabilnim, pouzdanim i visoko dostupnim sistemima za skladištenje podataka, ali svaki je imao oblasti koje bi se mogle poboljšati. [8]

Cassandra je dizajnirana kao najbolja kombinacija oba sistema u klasi kako bi se zadovoljile potrebe za skladištenjem podataka velikih razmera u nastajanju, kako u pogledu otisaka podataka, tako i u obimu upita. Sistemi kao što je Cassandra dizajnirani su i traže ciljeve kao što su: fleksibilna šema, skladištenje na robnom hardveru, globalna dostupnost uz malo kašnjenje, particionisani upiti orijentisani na ključ i ostalo. [8]

Cassandra je 2008. godine objavljena kao open source rešenje. Danas je deo Apache fondacije i jedno je od najpopularnijih NoSQL rešenja. Na slici 1 prikazan je logo Apache Cassandra baze podataka.



Slika 1: Cassandra logo

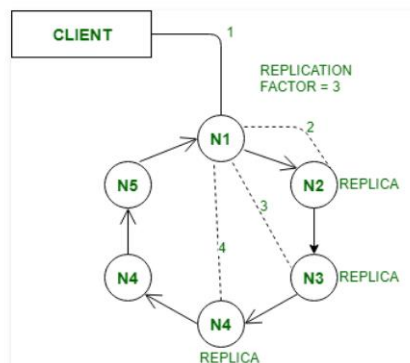
Cassandra baza podataka koristi svoj upitni jezik koji se naziva 'Cassandra Query Language (CQL)'. Cassandra nudi model sličan SQL-u, za kreiranje i ažuriranje šeme baze podataka i pristup podacima. Podaci se čuvaju u tabelama koje sadrže redove kolona.

U Apache Cassandri ne postoji arhitektura master-klijent. Ima peer to peer arhitektutu. Mogu se kreirati više kopija podataka u vreme kreiranja prostora ključeva, može se jednostavno definisati strategija replikacije i RF(faktor replikacije) za kreiranje više kopija podataka. Primer kreiranja se vidi na slici: [11]

```
CREATE KEYSPACE Example
WITH replication = {'class': 'NetworkTopologyStrategy',
                    'replication_factor': '3'};
```

Slika: Faktor replikacije

U ovom primeru je definisan faktor replikacije 3 što jednostavno znači da se kreiraju 3 kopije podataka preko više čvorova u smeru kazaljke na satu, što je prikazano na slici:



Slika: Faktor replikacije

## 2.1. Interna struktura

Interna struktura baze podataka predstavlja kako se podaci fizički skladište i organizuju (npr. indeksi, putanja pristupa,...). Odnosi se na organizaciju datoteka i njihov fizički zapis na disku. Interni (fizički) nivo obuhvata sadržaj cele baze podataka i definiše način fizičke organizacije na medijumima gde su podaci smešteni (formati zapisa u memoriji, organizacija i drugo).

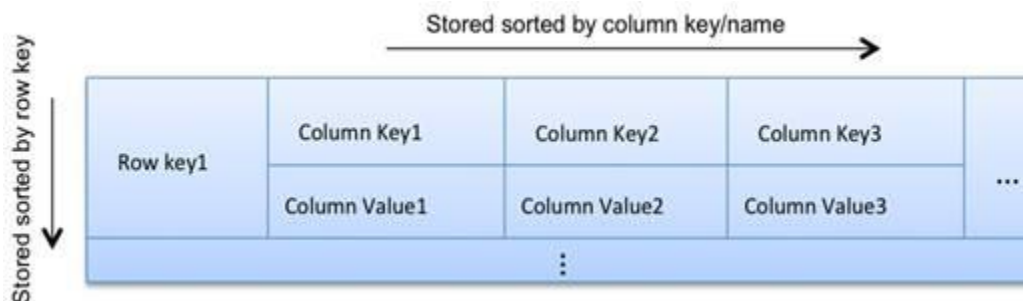
Cassandra model podataka je šema-opcionni model podataka orijentisan na kolone. To znači da, za razliku od relacione baze podataka, ne mora unapred da se modeluju sve kolone koje zahteva neka aplikacija, jer svaki red ne mora da ima isti skup podataka.

Cassandra model podataka sastoji se od prostora ključeva (keyspaces, analogno bazama podataka), porodica kolona (column families, analogno tabelama u relacionom modelu), ključeva (keys) i kolona (columns). [3] U tabeli 1 nalazi se prikaz pojmova koji se koriste u Cassandri u poređenju sa poznatim pojmovima kod relacionih baza podataka:

Tabela 1: Model podataka

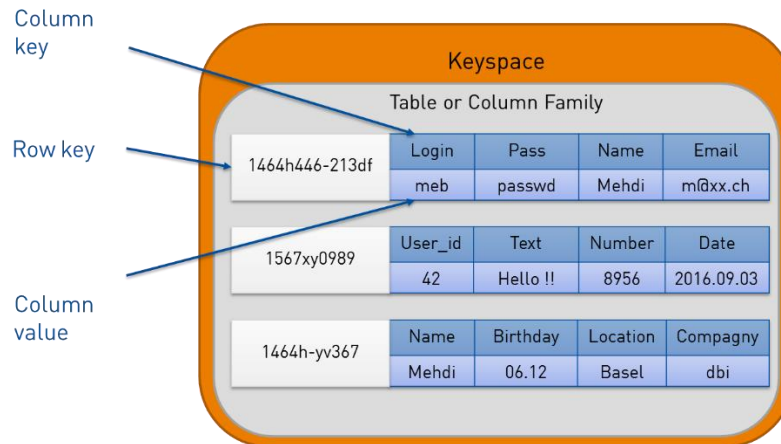
Relacioni model	Cassandra model
Database	Keyspace
Table	Column Family(CF)
Primary key	Row key
Column name	Column name/key
Column value	Column value

Svaka porodica kolona ne može se porediti sa relacionom tabelom. Nešto približnije poređenje je ugnježdjena struktura podataka mape. Interna struktura podataka u Cassandra bazi podataka prikazana je na slici:



Slika: Interna struktura podataka

Mapa omogućava efikasno traženje ključeva, a sortirana priroda daje efikasna skeniranja. U Cassandri broj ključeva kolona je neograničen, tj. mogu postojati dugački redovi. Ključ može da sadrži vrednost kao deo imena ključa, što znači da može postojati kolona bez vrednosti. Na slici je prikazan izgled DataModela i dat primer nekim sa podacima:



Slika: DataModel

## 2.2. Koncept indeksiranja

Podacima se može pristupiti korišćenjem atributa koji imaju ključ particije. Na primer Emp\_id ime kolone za tabelu Employee i ako je to particioni ključ te tabele, onda možemo da filtriramo ili pretražimo podatke uz pomoć particionog ključa. Klauzula where se koristi da se definiše uslov nad atributom kako bi se pretražili određeni podaci. Kod Cassandre, ukoliko postoji kolona koja nije deo particonog ključa, a mi želimo da pretražimo takvu tabelu i filtriramo, pretražimo i pristupimo podacima koristeći where klauzulu po toj koloni, takav upit neće biti uspešno izvršen i daće grešku. Kako bi moglo da se pristupa podacima koristeći attribute koji nisu deo particionog ključa za brzo i efikasno traženje podataka koji odgovaraju datom uslovu, onda mora da se definiše indeks. Indeks se može koristiti u različite svrhe, kao što su kolekcije, statičke kolone, kolone za prikupljanje i bilo koje druge kolone osim kolona brojača. Prednost je brzo i efikasno traženje podataka koji odgovaraju datom uslovu. U stvari, ukoliko nema indeksa na normalnoj koloni, čak nije dozvoljeno ni uslovno upiti po kolonama.

Indeks indeksira vrednosti kolona u zasebnoj, skrivenoj porodici kolona (tabeli) od one koja sadrži vresnosti koje se indeksiraju. Podaci indeksa su samo lokalni, što znači da neće replicirati vrednosti koje se indeksiraju. Ovo takođe znači da za upit podataka po indeksiranoj koloni, zahtevi moraju biti prosleđeni svim čvorovima, čekajući sve rakkije, a zatim se rezultati spajaju i vraćaju. Dakle, ukoliko postoje više čvorova, odgovor na upit se usporava kako se više mašina dodaje u klaster. [4]

Kada se koristi indeks:

- Ugrađeni indeksi su najbolja opcija u tabeli koja ima mnogo redova i koji redovi sadrže indeksiranu vrednost.
- U određenoj koloni koja kolona ima više jedinstvenih vrednosti u tom slučaju može se koristiti indeksiranje.
- Tabela koja ima više troškova zbog nekoliko razloga kao što je kolona koja ima više unosa nego u tom slučaju može se koristiti indeksiranje.
- Za ispitivanje i održavanje indeksa može se koristiti indeksiranje koje je u tom slučaju uvek dobra opcija. [2]

Indeks se ne koristi u situacijama:

- Na kolonama visokog kardinaliteta, jer se onda postavljaju upiti velikom broju zapisa za mali broj rezultata
- U tabelama koje koriste kolonu brojača
- U koloni koja se često ažurira i briše
- Da bi se pretražio red u velikoj particiji, osim ako nije usko upitan. [4]

### 2.2.1. Korišćenje sekundarnog indeksa

Koristeći CQL, može se kreirati indeks na kolni nakon definisanja tabele. Takođe može se indeksirati kolona kolekcije. Sekundarni indeksi se koriste za ispitivanje tabele pomoću kolone koja se inače ne može ispitivati.

Sekundarni indeksi su teški za korišćenje i mogu značajno uticati na performanse. Tabela indeksa se čuva na svakom čvoru u klasteru, tako da upit koji uključuje sekundarni indeks može brzo postati noćna mora performansi ako se pristupi više čvorova. Opšte pravilo je da se indeksira kolona sa malom kardinalnošću od nekoliko vrednosti. Pre kreiranje indeksa, treba biti svestan kada i ne treba kreirati indeks, što je opisano u poglavlju iznad. [7]

CQL podržava kreiranje sekundarnih indeksa na tabelama, omogućavajući upitima u tabeli da koriste te indekse. Sekundarni indeks je identifikovani imenom definisanim sa:

```
index_name ::= re('[a-zA-Z_0-9]+')
```

Kreiranje sekundarnog indeksa na tabeli koristi CREATE INDEX naredbu:

```
create_index_statement ::= CREATE [ CUSTOM ] INDEX [ IF NOT EXISTS ] [ index_name ]  
    ON table_name '(' index_identifier ')'  
    [ USING string [ WITH OPTIONS = map_literal ] ]  
index_identifier ::= column_name  
    | ( KEYS | VALUES | ENTRIES | FULL ) '(' column_name ')'
```

Na primer:

```
CREATE INDEX userIndex ON NerdMovies (user);  
CREATE INDEX ON Mutants (abilityId);  
CREATE INDEX ON users (keys(favs));  
CREATE CUSTOM INDEX ON users (email)  
    USING 'path.to.the.IndexClass';  
CREATE CUSTOM INDEX ON users (email)  
    USING 'path.to.the.IndexClass'  
    WITH OPTIONS = { 'storage': '/mnt/ssd/indexes/' };
```

Naredba CREATE INDEX koristi se za kreiranje novog (automatskog) sekundarnog indeksa za datu (postojeću) kolonu u tabeli. Ime za sam indeks može se navesti ispred ON ključne reči, ako se želi. Ako podaci postoje već za kolonu, oni će biti asinhrono indeksirani. Nakon kreiranja indeksa, novi podaci za kolonu se automatski indeksiraju u vreme umetanja.

Pokušaj kreiranja već postojećeg indeksa će vratiti grešku osim ako se IF NOT EXISTS opcija ne

koristi. Ako se koristi, izjava će biti no-op ako indeks već postoji.

Kada se kreira indeks na maps<maps>, mogu se indeksirati ili ključevi ili vrednosti. Ako je identifikator kolone postavljen unutar keys() funkcije, indeks će biti na ključevima mape, što omogućava da se koriste CONTAINS KEY u WHERE klauzulama. U suprotnom, indeks će biti na vrednostima karte.

Brisanje sekundarnog indeksa se vrši uz pomoć DROP INDEX:

```
drop_index_statement::= DROP INDEX [ IF EXISTS ] index_name
```

DROP INDEX se koristi za brisanje postojećeg indeksa. Argument naredbe je ime indeksa, koje može opcionalno da specificira prostor ključeva indeksa. Ako indeks ne postoji, naredba će vratiti grešku, osim ako IF EXISTS se ne koristi u kom slučaju je opcija bez opcije. [5]

### 2.2.2. Korišćenje više indeksa

Indeksi se mogu kreirati na više kolona i koristiti u upitima. Opšte pravilo o kardinalnosti važi za sve indeksirane kolone. U stvarnoj situaciji, određene kolone možda nisu dobar izbor, u zavisnosti od njihove kardinalnosti.[10]

Primeri:

- Tabela cycling.cyclist\_alt\_stats može dati statistiku o biciklistima:

```
cqlsh> CREATE TABLE cycling.cyclist_alt_stats ( id UUID PRIMARY KEY, lastname text, birthday timestamp, nationality text, weight text, height text );
```

- Kreiranje indeksa na kolonama rođendan i nacionalnost:

```
cqlsh> CREATE INDEX birthday_idx ON cycling.cyclist_alt_stats ( birthday );  
  
CREATE INDEX nationality_idx ON cycling.cyclist_alt_stats ( nationality );
```

- Upit za sve bicikliste sa određenim rođendanom iz određene zemlje:

```
cqlsh> SELECT * FROM cycling.cyclist_alt_stats WHERE birthday = '1982-01-29' AND nationality = 'Russia';
```

```
cqlsh:cycling> SELECT * FROM cycling.cyclist_alt_stats WHERE birthday = '1982-01-29' AND nationality = 'Russia';  
InvalidRequest: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"
```

Slika: Rezultat greška

- Indeksi su kreirani na odgovarajućim kolonama niske kardinalnosti, ali upit i dalje ne uspeva. Odgovor je u ključu particije, koji nije definisan. Kada se pokuša sa potencijalno skupim upitom, kao što je pretraživanje niza redova, Cassandra zahteva direktivu ALLOW FILTERING. Greška nije zbog više indeksa, već zbog nedostatka definicije ključa particije u upitu. [10]

```
cqlsh> SELECT * FROM cycling.cyclist_alt_stats WHERE birthday = '1990-05-27' AND nationality = 'Portugal'  
ALLOW FILTERING
```

Sada se dobija rezultat koji je očekivan (slika):



id	birthday	height	lastname	nationality	weight
1ba0417d-62da-4103-b710-de6fb227db6f	1990-05-27 00:00:00-0700	null	PAULINHO	Portugal	null

Slika: Rezultat

### 2.2.3. Indeksiranje kolekcija

Kolekcije se mogu indeksirati i ispitivati da bi se pronašla kolekcija koja sadrži određenu vrednost. Skupovi i liste su indeksirani malo drugačije od mapa, s obzirom na prirodu ključ/vrednost mapa.

Skupovi i liste mogu indeksirati sve vrednosti pronađene indeksiranjem kolone kolekcije. Mape mogu indeksirati ključ mape, vrednost karte ili unos karte koristeći metode prikazane u nastavku. Više indeksa se može kreirati na istoj koloni mape u tabeli, tako da se mogu ispitivati ključevi mapre, vrednosti ili unosi. Pored toga, zamrznute kolekcije mogu se indeksirati pomoću indeksiranje punog sadržaja zamrznute kolekcije. [9]

Sva upozorenja o korišćenju sekundarnih indeksa odnose se na kolekcije indeksiranja.

- Za kolekcije skupova i lista, treba napraviti indeks na imenu kolone. Napraviti indeks na skupu da bi se pronašli svi biciklisti koji su bili u određenom timu. Primer:

```
CREATE INDEX team_idx ON cycling.cyclist_career_teams ( teams );
```

```
SELECT * FROM cycling.cyclist_career_teams WHERE teams CONTAINS 'Nederland bloeit'; [9]
```

Gde bi rezultat bio recimo kao na slici:

id	lastname	teams
5b6962dd-3f90-4c93-8f61-eabfa4a803e2	VOS	{'Nederland bloeit', 'Rabobank Women Team', 'Rabobank-Liv Giant', 'Rabobank-Liv Woman Cycling Team'}

Slika: Rezultat

- Za kolekcije mapa, treba napraviti indeks na ključu mape, vrednosti karte ili unosu karte. Napraviti indeks na ključu mape da bi se pronašle sve kombinacije biciklista/tim za određenu godinu. Primer:

```
CREATE INDEX team_year_idx ON cycling.cyclist_teams ( KEYS (teams) );
```

```
SELECT * From cycling.cyclist_teams WHERE teams CONTAINS KEY 2015;
```

Gde bi rezultat bio kao na slici:

id	firstname	lastname	teams
cb07baad-eac8-4f65-b28a-bddc06a0de23	Elizabeth	ARMITSTEAD	{2011: 'Team Garmin - Cervelo', 2012: 'AA Drink - Leontien.nl', 2013: 'Boels-Dolmans Cycling Team', 2014: 'Boels-Dolmans Cycling Team', 2015: 'Boels-Dolmans Cycling Team'}
5b6962dd-3f90-4c93-8f61-eabfa4a803e2	Marianne	VOS	{2011: 'Nederland bloeit', 2012: 'Rabobank Women Team', 2013: 'Rabobank-Liv Giant', 2014: 'Rabobank-Liv Woman Cycling Team', 2015: 'Rabobank-Liv Woman Cycling Team'}
e7cd5752-bc0d-4157-a80f-7523add8dbcd	Anna	VAN DER BREGGEN	{2009: 'Team Flexpoint', 2012: 'Sengers Ladies Cycling Team', 2013: 'Sengers Ladies Cycling Team', 2014: 'Rabobank-Liv Woman Cycling Team', 2015: 'Rabobank-Liv Woman Cycling Team'}

Slika: Rezultat2

- Pravljenje indeksa na usnosima na mapi i nalaženje biciklista koji su istih godina. Korišćenje indeksa ENTRIES važi samo za mape:

```
CREATE TABLE cycling.birthday_list (cyclist_name text PRIMARY KEY, blist map<text,text>);
```

```
CREATE INDEX blist_idx ON cycling.birthday_list (ENTRIES(blist));
```

```
SELECT * FROM cycling.birthday_list WHERE blist['age'] = '23';
```

Rezultat:

cyclist_name	blist
Claudio HEINEN	{'age': '23', 'bday': '27/07/1992', 'nation': 'GERMANY'}
Laurence BOURQUE	{'age': '23', 'bday': '27/07/1992', 'nation': 'CANADA'}

Slika: Rezultat3

- Korišćenjem indeksa, pronalaženje biciklista iz iste zemlje:

```
SELECT * FROM cyclist.birthday_list WHERE blist['nation'] = 'NETHERLANDS';
```

cyclist_name	blist
Luc HAGENAARS	{'age': '28', 'bday': '27/07/1987', 'nation': 'NETHERLANDS'}
Toine POELS	{'age': '52', 'bday': '27/07/1963', 'nation': 'NETHERLANDS'}

Slika: Rezultat

- Pravljenje indeksa na vrednostima karte i nalaženje biciklista koji imaju određenu vrednost koja se nalazi na navedenoj mapi. Korišćenje indeksa VALUE važi samo za mape. Primer:

```
CREATE TABLE cycling.birthday_list (cyclist_name text PRIMARY KEY, blist
```

```
map<text,text>;
```

```
);CREATE INDEX blist_idx ON cycling.birthday_list (VALUES(blist));
```

```
SELECT * FROM cycling.birthday_list CONTAINS 'NETHERLANDS';
```

```
lorina@cqlsh:cycling> SELECT * FROM cycling.birthday_list WHERE blist CONTAINS 'NETHERLANDS';
```

cyclist_name	blist
Luc HAGENAARS	{'age': '28', 'bday': '27/07/1987', 'nation': 'NETHERLANDS'}
Toine POELS	{'age': '52', 'bday': '27/07/1963', 'nation': 'NETHERLANDS'}

Slika: Rezultat

- Pravljenje indeksa na punom sadržaju FROZEN mape. Tabela u ovom primeru čuva broj Pro pobeda, Grand Tour traka i Klasičnih trka u kojima se biciklista takmičio. Naredba SELECT pronalazi svakog biciklistu koji ima 39 pobeda u Pro trkama, 7 startova Grand Toura i 14 klasičnih startova:

```
CREATE TABLE cycling.race_starts (cyclist_name text PRIMARY KEY, rnumbers FROZEN<LIST<int>>);
CREATE INDEX rnumbers_idx ON cycling.race_starts (FULL(rnumbers));
SELECT * FROM cycling.race_starts WHERE rnumbers = [39,7,14];
```

cyclist_name	rnumbers
John DEGENKOLB	[39, 7, 14]

Slika: Rezultat

#### 2.2.4. Problemi sa korišćenjem indeksa

- Problemi sa korišćenjem indeksa kolone visoke kardinalnosti

Ukoliko se kreira indeks na koloni visoke kardinalnosti, koji ima mnogo različitih vrednosti, upit između polja će izazvati mnogo traženja za vrlo malo rezultata. U tabeli sa milijardu pesama, traženje pesama po piscu (vrednost koja je tipično jedinstvena za svaku pesmu) umesto po izvođaču verovatno će biti veoma neefikasno. Verovatno bi bilo efikasnije ručno održavati tabelu kao oblik indeksa umesto da se koristi ugrđeni indeks Cassandra. Za kolone koje sadrže jedinstvene podatke, ponekad je dobro u pogledu performansi koristiti indeks radi pogodnosti, sve dok je obim upita za tabelu koja ima indeksiranu kolonu umeren i nije pod konstantnim opterećenjem.

Nasuprot tome, kreiranje indeksa na koloni izuzetno niske kardinalnosti, kao što je logička kolona, nema smisla. Svaka vrednost u indeksu postaje jedan red u indeksu, što rezultira ogromnim redom za sve lažne vrednosti, na primer. Indeksiranje mnoštva indeksiranih kolona koje imaju `foo==true` i `foo==false` nije korisno.

- Problemi sa korišćenjem indeksa u koloni koja se često ažurira ili briše

Cassandra čuva nadgrobne spomenike u indeksu sve dok granica nadgrobničkih spomenika ne dostigne 100.000 ćelija. Nakon prekoračenja ograničenja nadgrobničkih spomenika, upit koji koristi

indeksiranu vrednost neće uspeti.

- **Problemu sa korišćenjem indeksa za traženje reda u velikoj particiji osim ako se ne pitaju usko**

Upit za indeksiranu kolonu u velikom klasteru obično zahteva sređivanje odgovora sa više particija podataka. Odgovor na upit se usporava kako se više mašina dodaje u klaster. Može se izbeći smanjenje performansi kada se traži red u velikoj particiji sužavanjem pretrage. [6]

### 2.2.5. Primeri primene

Postoji tabela za unos utakmica u kriket sa milion unosa za igrače u stotinama mečeva i potrebno je pretražiti rang igrača prema broju odigranih utakmica. Mnogi rangovi igrača će deliti istu vrednost kolone za godinu utakmice. Kolona match\_year je dobra opcija za indeks:

Ispod na slici je data sintaksa za kreiranje indeksa:

```
CREATE INDEX [ IF NOT EXISTS ] index_name
ON [keyspace_name.]table_name
([ ( KEYS | FULL ) ] column_name)
(ENTRIES column_name);
```

Slika: Sintaksa za kreiranje indeksa

Za kreiranje tabele koristi se keyspace1 kao prostor ključeva i Task kao ime tabele. Na slici 2 je dat prikaz kreiranja tabele sa odgovarajućim atributima i tipovima podataka:

```
CREATE TABLE keyspace1.Task
(
    Task_id text,
    Task_name text,
    Task_time timestamp,
    T_location text,
    PRIMARY KEY (Task_id, Task_name)
);
```

Slika: Kreiranje tabele

Pošto je Cassandra distribuirana i decentralizovana baza podataka sa podacima organizovanim po ključu particije, u opštem slučaju, upiti klauzule WHERE moraju uključiti particioni ključ. Na slici 3 dat je primer jedne SELECT naredbe sa određenim where uslovom:

```
SELECT *
FROM Task
WHERE Task_id = 'T210' AND Task_name; 'set alarm';
```

Slika: Primer select naredbe

Kao što se vidi sa slike broj 5 Task\_id i Task\_name su delovi primarnog ključa i upit sa prethodne slike bi dobro funkcionisao. Ukoliko bi se na primer napisao upit kao na slici, takav ne bi dobro funkcionisao, jer kao što se i vidi Task\_time nije deo primarnog ključa.

```
SELECT * FROM Task WHERE Task_time= '2019-09-30 15:02:56';
```

Slika: Primer upita

Da bi se rešila ovakva vrsta grešaka, mogu se kreirati indeksi na koloni za grupisanje. Potrebno je definisati tabelu koja ima kompozitni particioni ključ, a zatim da se kreira indeks na koloni za grupisanje. Na slici se vidi primer kreiranja indeksa i novog keyspace-a.

```
CREATE TABLE keyspace1.Task (  
    Task_id text,  
    Task_name text,  
    Task_time timestamp,  
    T_location text,  
    PRIMARY KEY ((Task_id, Task_name), Task_time)  
);  
  
CREATE INDEX ON keyspace1.Task(Task_time);
```

Slika: Kreiranje tabele i indeksa

Sada bi prethodni upit, sa slike \*\*\* bio uspešno izvršen i vraćao bi dobre rezultate.

Potrebno je znati, da kreiranje sekundarnih indeksa ne znači da će biti povećana brzina upita u Cassandri. Jedna od važnih prednosti sekundarnih indeksa pomaže pristupu podacima koji jednostavno mogu učiniti tako da se klauzula where koje upućuju na vrednosti u koloni izvan primarnih i kolona za grupisanje mogu pokrenuti.

Povećanje brzine upita u Cassandri može se postići kreiranjem tabele posebno za upit. Primer je dat na slici:

```
CREATE TABLE Student_record  
(  
    Stu_state text,  
    Stu_zip text,  
    Stu_address text,  
    PRIMARY KEY(Stu_state, Stu_zip)  
);
```

Slika: Kreiranje

U ovoj tabeli Stu\_state i Stu\_zip mogu biti isti, tako da, da bi se definisao jedinstveni zapis u tabeli, može se dodati Stu\_id kao primarni ključ kako bi jedinstveno definisao zapis. To se može postići vršenjem izmena u postojećoj tabeli korišćenjem komande ALTER u CQL-u na sledeći način (slika):

```
ALTER TABLE Student_record ADD Stu_id int PRIMARY KEY;
```

Slika: Naredba alter

Nakon izvršenja ove naredbe, izgled tabele biće kao na slici:

column 1	column 2	column 3	column 4
Stu_state	Stu_zip	Stu_address	Stu_id

Slika: Evidencija učenika

Kako bi se proverila ova kreirana tabela u Cassandra, može se napisati sledeći upit(slika):

```
SELECT *  
FROM Student_record  
WHERE Stu_id = '107';
```

Slika:

A primer izlaza se vidi na slici:

	column 1	column 2	column 3	column 4
ROW 1	Stu_state	Stu_zip	Stu_address	Stu_id
	UP	12345	noida	107

Slika: Izlaz

### 3. Praktična primena

U ovom poglavlju redom biće opisan način pokretanja Cassandra baze podataka i alata koji se koristi za lakše upravljanje baze. Nakon toga biće pomenut skup podataka koji se koristi, dati upiti za kreiranje, indeksi koji su kreirani i mogu se koristiti.

#### 3.1. Pokretanje Cassandra baze podataka

Da bi Cassandra bila pokrenuta na računaru, neophodno je njeno preuzimanje sa: [https://cassandra.apache.org/\\_/download.html](https://cassandra.apache.org/_/download.html). Trenutno dostupna najnovija verzija je 4.3.0. U ovom radu je korišćena verzija 3.11.9. Nakon uspešno preuzimanja i raspakivanja, neophodno je podesiti promenjive okruženja na svom računaru. To su (slika):

Korisničke promenljive za Zeljko

Promenljiva	Vrednost
CASSANDRA_HOME	D:\apache-cassandra-3.11.9-bin\apache-cassandra-3.11.9
COMPOSE_CONVERT_WIN...	true
HADOOP_HOME	C:\Hadoop
JAVA_HOME	C:\Program Files\Java\jre1.8.0_241
MOZ_PLUGIN_PATH	C:\Program Files (x86)\Foxit Software\Foxit Reader\plugins\

Slika: Promenljive okruženja

Nakon završenog podešavanja, može se Cassandra se može pokrenuti klikom na .bat fajl koji se nalazi u folderu preuzete instalacije na putanji ..\apache-cassandra-3.11.9-bin\apache-cassandra-3.11.9\bin. Kao rezultat uspešnog pokretanja biće prikaz kao na slici:

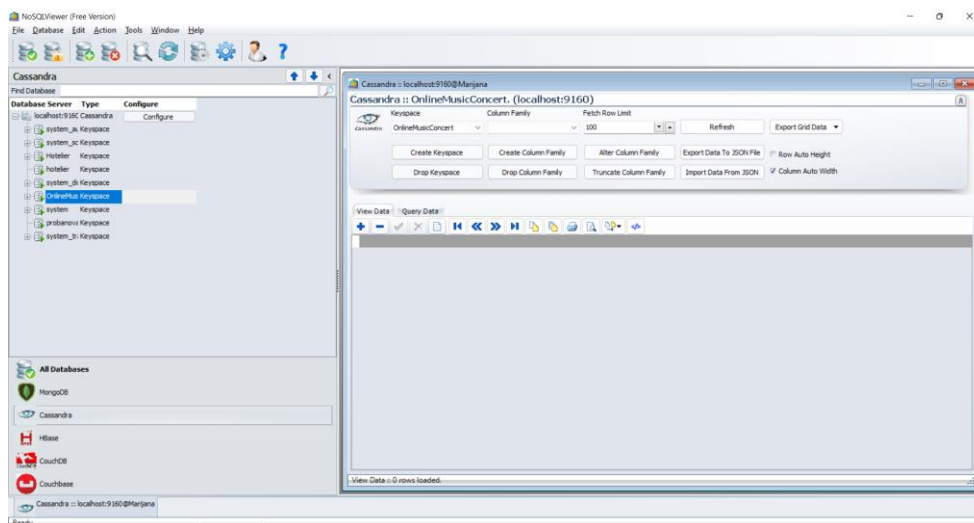
```

C:\WINDOWS\system32\cmd.exe
99, 7888487396927513643, 7965925950549430491, 8025818275962762557, 8067710503774624198, 8124041454415920466, 81449635081
85302583, 815717315030859714, 8182843434403444200, 8239004193574534295, 8316306243721240479, 8342438404139754948, 83484
80808081142800, 8396374670050426555, 8425505929639859808, 8476707151695724697, 8533178804501162704, 858975026250471288,
8659183605472482189, 8660118116150380981, 8665472500241802397, 870809247954255013, 8805673477016170063, 894971582153845
7306, 998789745154608498, 989798095148003435]
INFO [main] 2022-04-06 20:40:09,818 StorageService.java:1492 - JOINING: Finish joining ring
INFO [main] 2022-04-06 20:40:09,836 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='Online
MusicConcert', ColumnFamily='Koncert')
INFO [main] 2022-04-06 20:40:09,837 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='Online
MusicConcert', ColumnFamily='preporuka')
INFO [main] 2022-04-06 20:40:09,837 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='Online
MusicConcert', ColumnFamily='Izvodjac')
INFO [main] 2022-04-06 20:40:09,838 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='Online
MusicConcert', ColumnFamily='Numeri')
INFO [main] 2022-04-06 20:40:09,839 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='Online
MusicConcert', ColumnFamily='Bend')
INFO [main] 2022-04-06 20:40:09,839 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='Online
MusicConcert', ColumnFamily='Zakazivanje')
INFO [main] 2022-04-06 20:40:09,841 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='Hotell
er', ColumnFamily='Room')
INFO [main] 2022-04-06 20:40:09,841 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='Hotell
er', ColumnFamily='Hotel')
INFO [main] 2022-04-06 20:40:09,841 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='Hotell
er', ColumnFamily='Reservation')
INFO [main] 2022-04-06 20:40:09,842 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='Hotell
er', ColumnFamily='Guest')
INFO [main] 2022-04-06 20:40:09,842 SecondaryIndexManager.java:512 - Executing pre-join tasks for: CFS(Keyspace='Hotell
er', ColumnFamily='Room1')
INFO [main] 2022-04-06 20:40:09,869 StorageService.java:2408 - Node localhost/127.0.0.1 state jump to NORMAL

```

Slika: Cassandra

Postoje alati za upravljanje NoSQL bazom podataka, koji pomažu da se poboljša produktivnost. Svaki od alata ima korisnički interfejs, koji pruža korisničko iskustvo u razvoju. Neki od njih su: Compass, NoSQL Manager, NoSQL Booster, Robo Mongo, QueryAssist,... Za potrebe izrade ovog rada korišćen je NoSQL Viewer. To je besplatan inovativan i moćan softverski proizvod za popularne NoSQL baze podataka, kao što su Apache Cassandra, MongoDB, Hbase i druge. Izgled ovog okruženja pri pokretanju se vidi na slici:



Slika: NoSQL Viewer

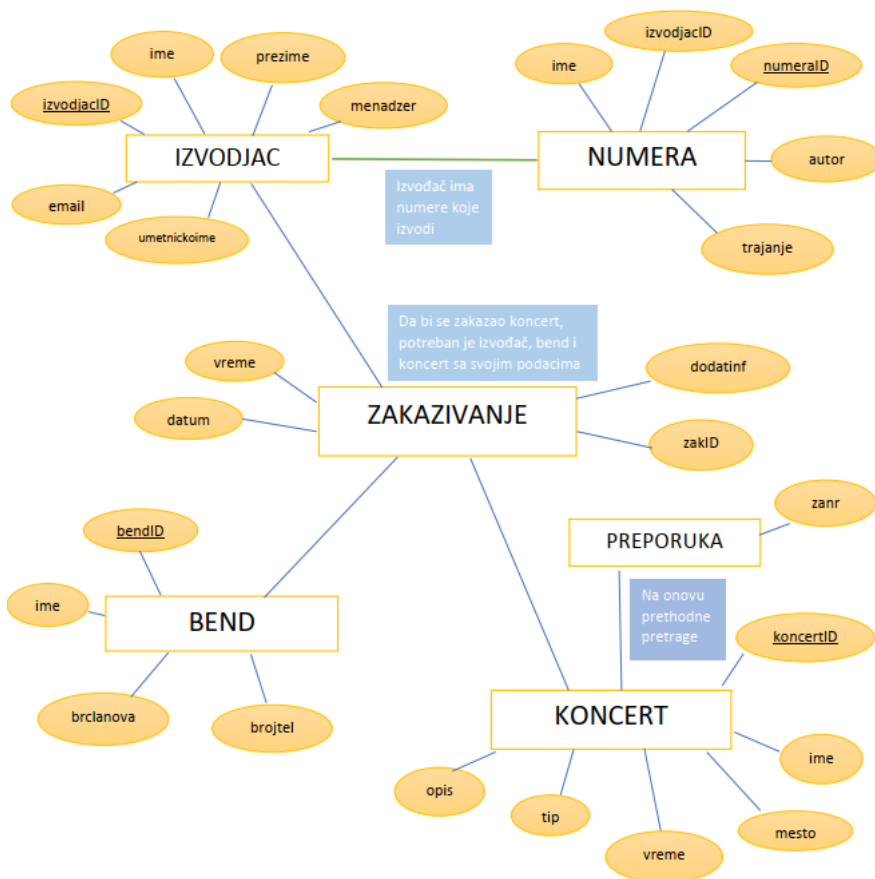
Sa leve strane prozora prikazana je baza podataka na koju smo povezani, u ovom slučaju je to



Cassandra, zatim svi keyspace-ovi koji su već kreirani sa svojim column family-jama i njihovim nazivima. U manjem prozoru, sa desne strane, vidi se prostor u kome je moguće napisati neki od upita i izvršiti ga ukoliko je ispravan.

### 3.2. Skup podataka

Skup podataka odnosi se na zakazivanje online koncerata, gde se nalaze podaci o koncertima, izvođačima, bendovima, numerama koje izvođači izvode i zakazivanju. Na slici je prikazan dijagram koji prikazuje sve entitete sa atributima, gde su naznačeni atributi koji predstavljaju primarne ključeve.



Slika: Dijagram

### 3.3. Unos i upotreba podataka

- Prvi korak je kreiranje keyspace. To se čini na sledeći način:

```
create keyspace OnlineMusicConcert;
```

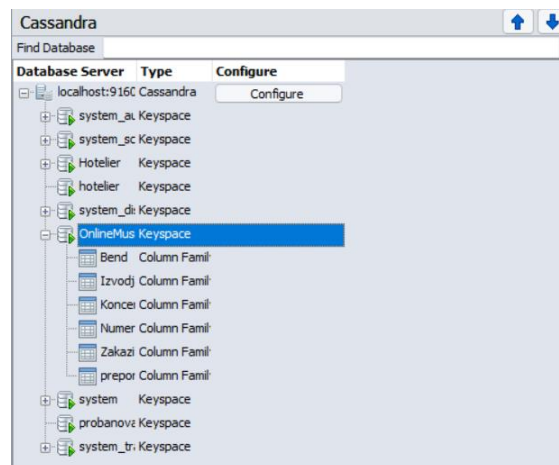
- Kada je keyspace uspešno kreiran, mogu se kreirati column family unutar njega. Neophodno je kreirati Koncert, Izvodjac, Bend, Numera, Zakazi, preporuka. I to na sledeći način:



```
CREATE TABLE "Koncert" (
  "koncertID" text,
  ime text,
  opis text,
  organizator text,
  sponzor text,
  tip text,
  PRIMARY KEY ("koncertID")
)
```

```
CREATE TABLE "Zakazivanje" (
  "zakID" text,
  "izvodjacID" text,
  "bendID" text,
  "koncertID" text,
  vreme text,
  datum text,
  dodatinf text,
  PRIMARY KEY("zakID")
)
```

Na isti način kreiraju se i ostale tabele. Kada su sve uspešno kreiranje u okviru keyspace-a OnlineMusicConcert u NoSQL Viewer-u može se sa leve strane videti struktura kao na slici:



Slika: OnlineMusicConcert keyspace

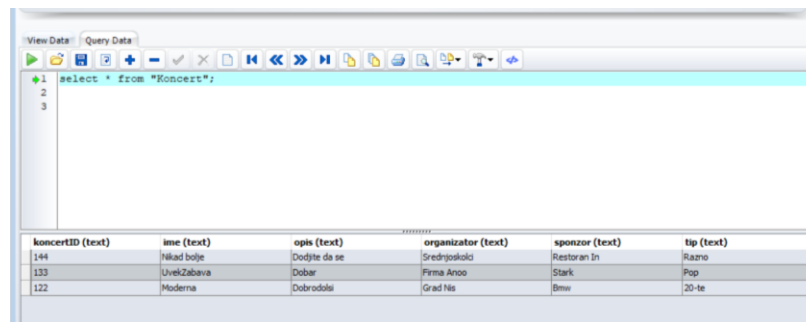
- Kada su uspešno kreirane tabele, mogu se dodati podaci. Ispod se nalaze primeri nekih od njih:

```
insert into "Koncert" ("koncertID",ime,opis,organizator,sponzor,tip) values ('122','Moderna','Dobrodolsi','Grad Nis','Bmw','20-te');
```

```
insert into "Koncert" ("koncertID",ime,opis,organizator,sponzor,tip) values ('133','UvekZabava','Dobar provod.','Firma Anoo','Stark','Pop');
```

`insert into "Koncert" ("koncertID",ime,opis,organizator,sponzor,tip) values ('144','Nikad bolje','Dodjite da se zabavimo','Srednjoskolci','Restoran In','Razno');`

- Naredbom `select * from "Koncert";` želimo da prikazemo sve podatke koji su uneti. Na slici se vidi rezultat ovog upita:



koncertID (text)	ime (text)	opis (text)	organizator (text)	sponzor (text)	tip (text)
144	Nikad bolje	Dodjite da se	Srednjoskolci	Restoran In	Razno
133	UvekZabava	Dobar	Firma Anoo	Stark	Pop
122	Moderna	Dobrodoli	Grad Nis	Bmw	20-te

Slika: Rezultat upita

Na potpuno isti način vrši se dodavanje podataka u ostalim tabelama. Primeri takvih upita nalaze se u pratećem dokumentu, koji se dostavlja uz ovaj rad, pod nazivom 'DodavanjePodataka.txt'.

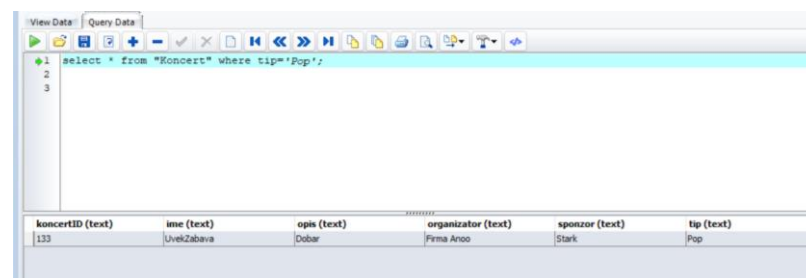
- Kada su dodate sve tabele i podaci za svaku od njih, kako bi se podaci pretraživali i po kolonama koje nisu deo primarnog ključa, kreiraćemo indekse. Primer jednog od njih je sledeći:

`CREATE INDEX ON "Koncert"(tip);`

To znači da je kreiran indeks u tabeli 'Koncert' nad kolonom tip. Kada je uspešno kreiran, moguće je pretraživati neophodne podatke nad tom kolonom.

`select * from "Koncert" where tip='Pop';`

Rezultat ovog upita je sledeći (slika ):



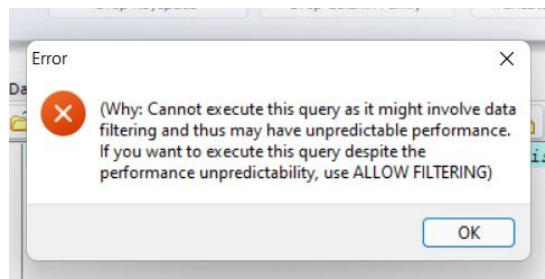
koncertID (text)	ime (text)	opis (text)	organizator (text)	sponzor (text)	tip (text)
133	UvekZabava	Dobar	Firma Anoo	Stark	Pop

Slika: Rezultat upita

Ukoliko upit izgleda ovako:

`select * from "Koncert" where sponzor='Grad Nis';`

Nastaće problem. Nije moguće pretražiti po koloni 'sponzor' i rezultat ovog upita biće greška (slika):



Slika: Greška

## 4. Zaključak

U radu je detaljno opisana Apache Cassandra baza podataka. U prvom delu su date opšte karakteristike ove NoSQL baze podataka. Zatim su detaljno opisane interna struktura i koncept indeksiranja, kao i dati neki primeri primene. U trećem poglavlju opisan je način pokretanja Cassandra baze podataka na računaru, kao i alata za lakše korišćenje. Nakon toga je opisan skup podataka koji se koristi....

## 5. Literatura

- [1] <https://www.geeksforgeeks.org/concept-of-indexing-in-apache-cassandra/>
- [2] Concept of indexing in Apache Cassandra, dostupno na: [https://docs.datastax.com/en/cql-oss/3.3/cql/cql\\_using/usePrimaryIndex.html](https://docs.datastax.com/en/cql-oss/3.3/cql/cql_using/usePrimaryIndex.html)
- [3] Internal Data Structure, dostupno na: [https://teddyma.gitbooks.io/learncassandra/content/model/internal\\_data\\_structure.html](https://teddyma.gitbooks.io/learncassandra/content/model/internal_data_structure.html)
- [4] Indexing, dostupno na: <https://teddyma.gitbooks.io/learncassandra/content/model/indexing.html>
- [5] Secondary Indexes, dostupno na: <https://cassandra.apache.org/doc/latest/cassandra/cql/indexes.html>
- [6] When to use an index, dostupno na: [https://docs.datastax.com/en/cql-oss/3.3/cql/cql\\_using/useWhenIndex.html](https://docs.datastax.com/en/cql-oss/3.3/cql/cql_using/useWhenIndex.html)
- [7] Using a secondary index, dostupno na: [https://docs.datastax.com/en/cql-oss/3.3/cql/cql\\_using/useSecondaryIndex.html](https://docs.datastax.com/en/cql-oss/3.3/cql/cql_using/useSecondaryIndex.html)
- [8] Apache Cassandra, dostupno na: <https://cassandra.apache.org/ /index.html>
- [9] Indexing a collection. Dostupno na: [https://docs.datastax.com/en/cql-oss/3.3/cql/cql\\_using/useIndexColl.html](https://docs.datastax.com/en/cql-oss/3.3/cql/cql_using/useIndexColl.html)
- [10] Using multiple indexes, dostupno na: [https://docs.datastax.com/en/cql-oss/3.3/cql/cql\\_using/useMultIndexes.html](https://docs.datastax.com/en/cql-oss/3.3/cql/cql_using/useMultIndexes.html)
- [11] Apache Cassandra (NoSQL database) <https://www.geeksforgeeks.org/apache-cassandra-nosql-database/?ref=rp>