Elektronski fakultet, Univerzitet u Nišu

Katedra za Računarstvo i informatiku

SEMINARSKI RAD

Distribuirane baze podataka - Apache Cassandra

Predmet: Sistemi za upravljanje bazama podataka

Student: Aleksandar Mitić

Broj indeksa: 1204

Sadržaj

[Uvod 3](#_Toc75472619)

[Struktura podataka 5](#_Toc75472620)

[Klaster topologija 7](#_Toc75472621)

[Komunikacija između čvorova 8](#_Toc75472622)

[Podesiva konzistentnost 10](#_Toc75472623)

[Putanja za čitanje i upis 11](#_Toc75472624)

[Zaključak 13](#_Toc75472625)

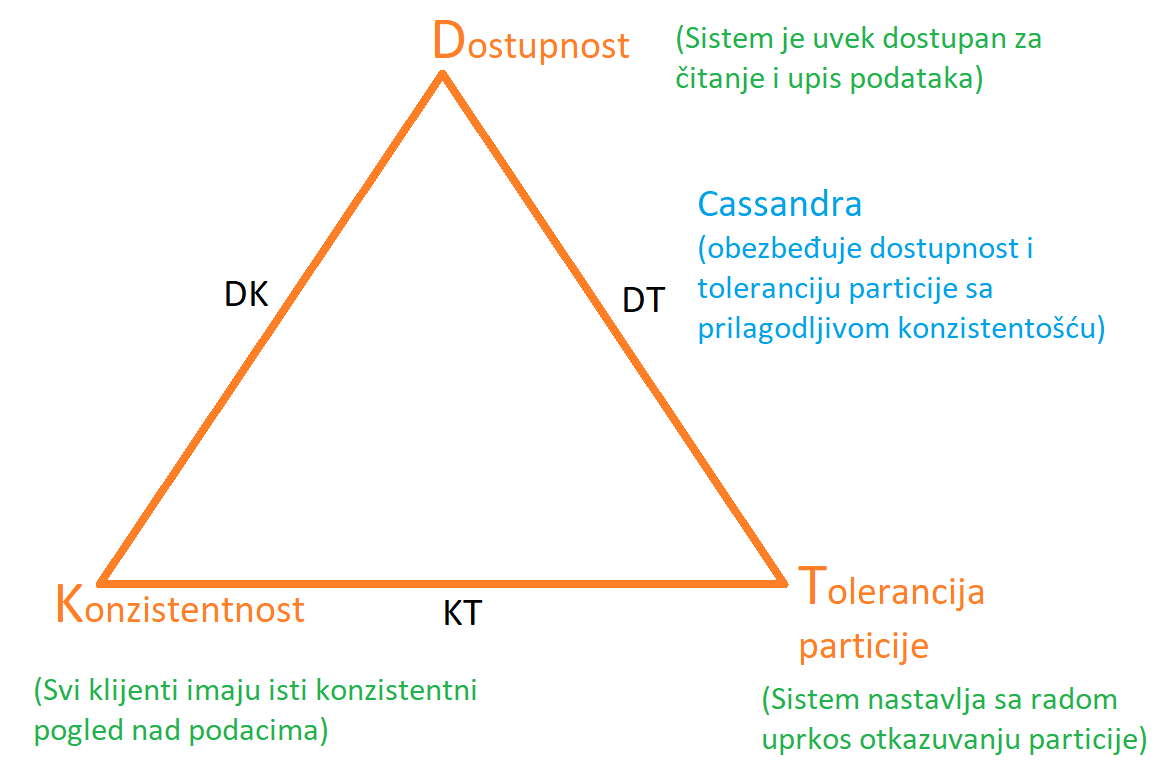
[Literatura 14](#_Toc75472626)

# Uvod

Pre izvesnog perioda svet softvera za upravljanje bazama podataka se promenio, uglavnom za visokotehnološke kompanije koje rukuju ogromnim količinama distribuiranih podataka preko klastera robnih računarskih mašina i koje se moraju suočiti sa uobičajenim problemima dostupnosti velikom broju korisnika koji žele da im pristupe u istom trenutku. Iz tog razloga, Facebook inženjeri odlučili su da naprave novo rešenje za probleme pretrage prijemnog sandučeta (eng. „inbox“) korisnika i naprave novi distribuirani sistem za skladištenje koristeći najbolje karakteristike dva druga postojeća softvera iz Amazona (Dynamo) i Google-a (Big Table). Ideja je bila da se napravi novi sistem za upravljanje strukturiranim podacima koji je dizajniran za skaliranje mnogih servera bez ijedne tačke neuspeha da bi se prevazišli uobičajeni „prekidi“ usluga i izbegao negativan uticaj na krajnje korisnike. To rešenje su nazvali Apache Cassandra.

Apache Cassandra je NoSQL distribuirana baza podataka. Po dizajnu, NoSQL baze podataka su lagane, otvorenog koda, nerelacione i uglavnom distribuirane. Među njihovim snagama ubrajaju se horizontalna skalabilnost, distribuirane arhitekture i fleksibilan pristup definiciji šeme. NoSQL baze omogućavaju brzu, ad-hoc organizaciju i analizu izuzetno velikih količina podataka kao i različitih tipova podataka. To je postalo važnije poslednjih godina, sa pojavom Big Data i potrebom da se baze podataka brzo skaliraju u oblaku (eng. „cloud“). Cassandra se nalazi među NoSQL bazama podataka koje su se bavile ograničenjima prethodnih tehnologija upravljanja podacima, poput SQL baza podataka.

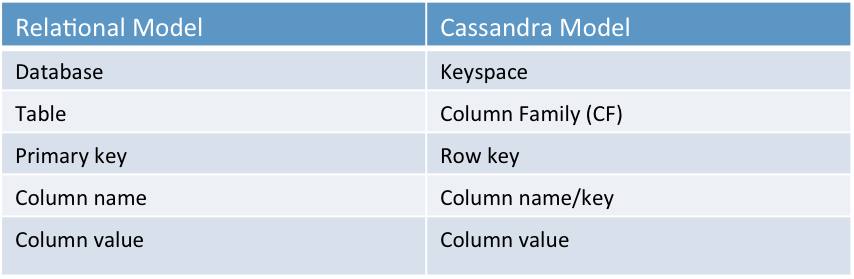
Apache Cassandra je dizajnirana da padne u presek „DT“ CAP teoreme [1] koja je prikazana na slici 1 i koja kaže da bilo koji distribuirani sistem može samo garantovati dve od sledećih mogućnosti istovremeno: konzistentnost, dostupnost i tolerancija particije. Na ovaj način Cassandra najbolje odgovara rešenju koje traži distribuiranu bazu podataka koja donosi visoku dostupnost sistema i takođe je vrlo tolerantna za podelu podataka kada je neki čvor u klasteru van mreže, što je uobičajeno u distribuiranim sistemima.

****

**Slika 1**: Grafički prikaz CAP teoreme

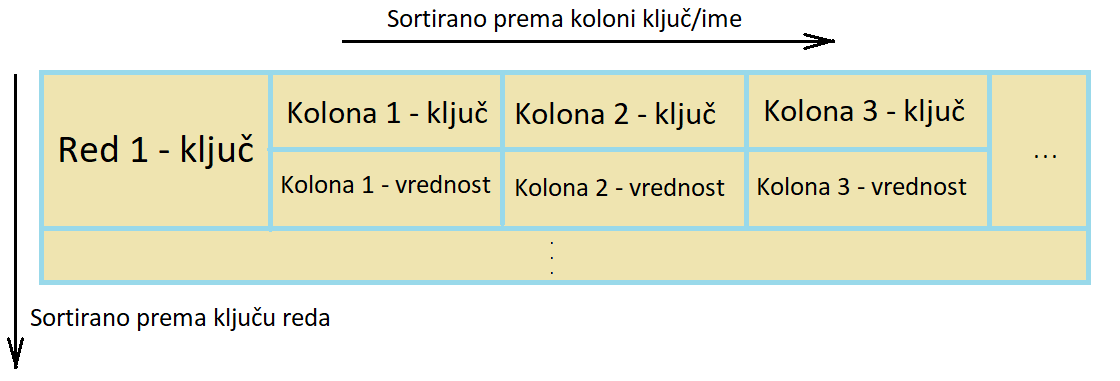
# Struktura podataka

Cassandra je klasifikovana kao baza podataka zasnovana na koloni (eng. „column-based“), što znači da se njena osnovna struktura za čuvanje podataka zasniva na skupu kolona koji se sastoji od para ključa i vrednosti kolone. Svaki red se identifikuje jedinstvenim ključem, stringom bez ograničenja u dužini, koji se naziva particijski ključ (eng. „partition key“). Svaki skup kolona naziva se porodicom kolona (eng. „column families“), slično tabeli relacione baze podataka. Analogija relacionog modela priakzana na slici 2, često se koristi za upoznavanje sa Cassandre sa novim korisnicima:



**Slika 2**: Analogija relacionog modela

Ova analogija pomaže u prelazu iz relacionog u nerelacioni svet. Međutim, nije poželjno da se koristiti ova analogija prilikom dizajniranja porodice kolona u Cassandri. Umesto ove analogije, porodicu kolona u Cassandri potrebno je zamisliti kao mapu mape: gde spoljna mapa predstavlja ključ reda a unutrašnja mapa ključ kolone. Obe mape su sortirane i prikazane su na slici 3:

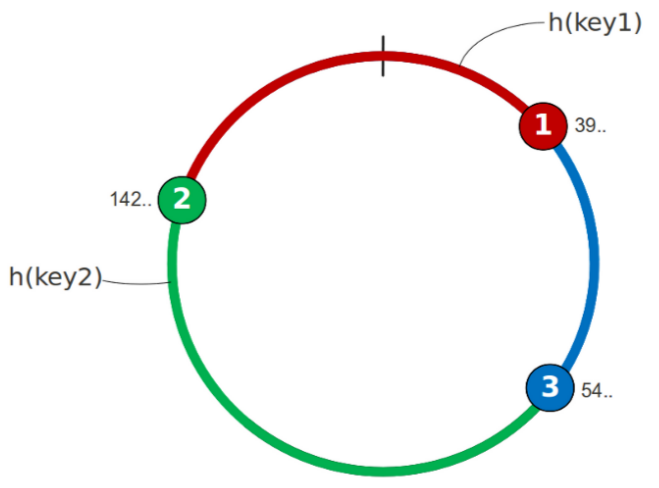


**Slika 3**: Mapiranje porodice kolona kod Cassandre

Prilikom kreiranja porodice kolona u Cassandri potrebno je posmatrati kolone na ovaj način, kao ugnježdenu sortiranu mapu, jer je to tačnija analogija koja u velikoj meri može pomoći u donošenju odluka o samom modelu podataka. Mapa obezbeđuje efikasno traženje ključeva, a njena sortirana priroda obezbeđuje i efikasno pretraživanje. Kod Cassandre, za efikasno traženje i skeniranje opsega vrednosti mogu se iskoristiti ključevi redova i ključevi kolona. Broj ključeva kolona je neograničen. Drugim rečima, kod Cassandre moguće je imati široke redove. Ključ može sam da sadrži vrednost, tako da je moguće imati i kolonu bez vrednosti.

# Klaster topologija

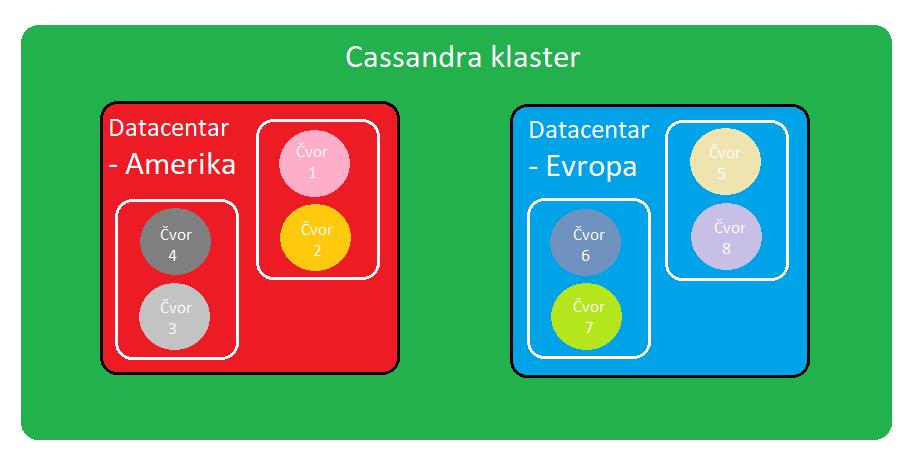
Instanca Cassandre skladišti jednu ili više tabela u skladu sa definicijom korisnika. Uobičajena topologija za instalaciju Cassandre je skup instanci instaliranih na različite čvorove servera, koji formiraju klaster čvorova koji se takođe kod Cassandre nazivaju i kao prsten (eng. „ring“). Svaki čvor u ringu je odgovoran za skladištenje kopije porodice kolona koji su definisani ključem particije i konfigurisani faktorom replikacije.



**Slika 4**: Cassandra prsten sa 3 čvora

Na slici 4, prikazan je prsten kod Cassandre koji sadrži tri čvora u kojima se čuvaju različiti ključevi koji se dobijaju izračunavanjem pomoću „hash“ funkcije i na osnovu njih se odlučuje lokacija podataka u sistemu kao i lokacija replika. Podrazumevana konfiguracija za faktor replikacije kod Cassandre je 3, što znači da će svi podaci, koji su uskladišteni na prvom čvoru, biti iskopirani i na preostala dva čvora. Boje u prstenu predstavljaju skup ključeva uskladištenih u svakom čvoru prema broju opsega koje vraća heš funkcija. U ovom primeru svi ključevi od 1 do 38 biće postavljeni na čvor 1, pa će ključ 1 biti sačuvan na čvoru 1, a jedna njegova kopija na čvorovima 2 i 3 s obzirom na konfiguraciju faktora replikacije od tri. U tom slučaju, ako neka aplikacija treba da pristupi ključu 1, a da pri tom čvor 1 nije u funkciji, Cassandra će pokušati da dobije njegovu kopiju iz čvorova 2 ili 3.

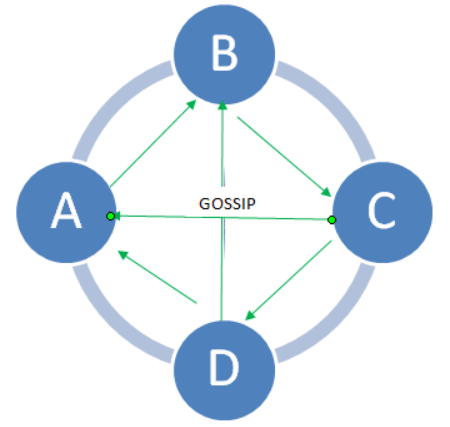
Kod Cassandre ne postoji glavni čvor u klasteru, već svaki čvor može da deluje kao koordinacioni čvor (eng. „master node“). Kada se korisnik poveže sa bilo kojim čvorom, taj čvor se proglašava koordinacionim čvorom i on postaje postaje odgovoran za svako čitanje i upis podataka u sistemu. Takođe, Cassandra podržava specifičnu konfiguraciju za postavljanje centara podataka (eng. „data center“), tako što se može definisati pozicije svakog čvora u centru podataka pa čak i pozicije svake grupe (eng. „rack“) čvorova u prstenu. Ova konfiguracija pomaže da se poveća nivo visoke dostupnosti, a takođe i da se smanji kašnjenje prilikom čitanja, tako što će korisnici čitati podatke iz najbližeg čvora. Na slici 5 je dat primer Cassandra klastera koji je konfigurisan sa 8 čvorova koji su pozicionirani u različitim centrima podataka:



**Slika 5**: Cassandra klaster konfigurisan sa 8 čvorova u različitim centrima

# Komunikacija između čvorova

Svi čvorovi kod Cassandre međusobno komuniciraju putem peer-to-peer [2] komunikacijskog protokola koji se još naziva i „Gossip Protocol“. Ovaj protokol emituje informacije o podacima i stanju čvorova u sistemu. Jedna zanimljivost ovog protokola je da on zapravo tračari! Jedan čvor ne treba da „razgovara“ sa svim ostalim čvorovima da bi saznao nešto o njima. Da bi se izbegao komunikacijski haos kada jedan čvor komunicira sa drugim čvorom kod Cassandre, on ne samo da pruža informacije o svom statusu, već pruža i najnovije informacije o čvorovima sa kojima je ranije komunicirao. Kroz ovaj proces dolazi do smanjenja mrežnog zagušenja, više informacija se čuva i povećava se efikasnost prikupljanja informacija.



**Slika 6**: Tračarenje čvorova peer-to-peer komunikacijom kod Cassandre

Na slici 6 je prikazan primer tračarenja čvorova D i B i čvorova A i C. Ukoliko čvor A ostvari komunikaciju sa čvorom B, zatim čvor B ostvari komunikaciju sa čvorom C, tada će čvor C od čvora B dobiti informaciju da je čvor A aktivan i dobiće sve informacije o njemu jer je čvor B imao komunikaciju sa čvorom A. Isto će se destiti, kada čvor C uspostavi vezu sa čvorom D, tada će čvor D dobiti informaciju o čvoru B jer je čvor C imao komunikaciju sa čvorom B.

Gossip protokol se takođe koristi i za otkrivanje čvorova koji su u grešci tj. za otkrivanje neaktivnih čvorova. On se ponaša vrlo slično TCP protokolu koji pokušava da dobije potvrdni odgovor (ACK poruku) pre nego što uzme u obzir da je čvor aktivan ili neaktivan. U osnovi, kada dva čvora međusobno komuniciraju; na primer, kada čvor 1 pošalje SYN [3] poruku (slično TCP protokolu) čvoru 2, on očekuje da primi nazad ACK poruku i zatim ponovo pošalje ACK poruku čvoru 2, potvrđujući trosmerno rukovanje (eng. „TCP 3-Way Handshake”). Ako čvor 2 ne odgovori na SYB poruku, označen je kao da je u grešci odnosno da je neaktivan. Čak i kada čvorovi odu u grešku, ostali čvorovi će mu povremeno slati poruke kako bi detektovali da li je i dalje neaktivan. Ovaj protokol je takođe važan za pružanje informacija o klijentovom upravljačkom programu (eng. „driver”) o klasteru, kako bi mu se omogućilo da odabere bolji raspoloživi čvor za povezivanje, sa ciljem da se napravi balans u sistemu i kako bi se pronašao najbliži i najbrži put za čitanje potrebnih podataka.

# Podesiva konzistentnost

U uvodu je već rečeno da je Cassandra glavni akcenat stavila na visoku propusnost i toleranciju particija ali to ne znači da je Cassandre konzistentnost podataka potpuno izostavljena. Zapravo, nivo konzistentosti kod Cassandre korisnik može sam prilagoditi svojim potrebama. Korisnik može izabrati konzistentnost od niskog pa sve do visokog nivoa. Uobičajeni termini koji se koriste za upis i čitanje podataka su jedan (eng. „one“), kvorum (eng. „QUORUM“) i sve (eng. „ALL“). Kvorum se dobija pomoću formule prikazane na slici 7, gde *replic* označava broj replika:



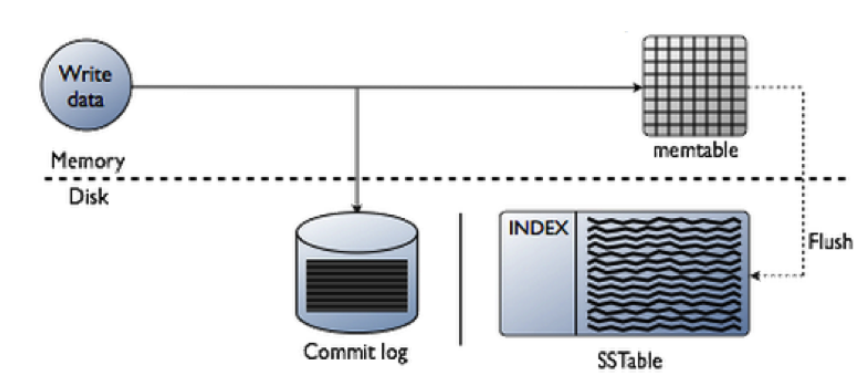
**Slika 7**: Izračunavanje kvoruma

To znači da će s obzirom na zadati faktor replikacije od tri (3) definisan za tabele prostora ključeva i nivo konzistentnosti „SVE“, jedna operacija upisa kod Cassandre će morati da sačeka da podaci budu upisani i potvrđeni od sva 3 čvora pre nego što se klijentu odgovori. Konzistentnost „KVORUMA“ znači većina čvorova (N / 2 + 1) treba da odgovori koordinacionom čvoru pre nego što koordinacioni čvor odgovori klijentu. U perspektivi klastera kada se klijent poveže sa čvorom da bi upisao neke podatke, prvo proverava kojem čvoru pripada particijski ključ tih podataka, a zatim koordincaioni čvor na koji je klijent povezan šalje te podatke čvoru koji treba da uskladišti taj ključ, u zavisnosti od nivoa konzistentnosti koji definiše korisnik (nivo konzistentnosti „SVI“) koordinator čeka da svi čvorovi odgovore na zahtev pre nego što odgovori klijentu. Dakle, na korisniku je da definiše koji je nivo konzistentnosti pogodan za svaki deo rešenja. Pored navedenih postoje i sledeći nivoi konzistentnosti kod Cassandre:

* ONE - samo jedna replika sme da vrati odgovor
* TWO - dve replike moraju da odgovore
* THREE - tri replike moraju da odgovore
* QUORUM - glavna replika od (n / 2 + 1) replike mora da odgovori
* ALL - sve replike moraju da odgovore
* LOCAL\_QUORUM - glavna replika u lokalnom centru podataka mora da odgovori
* EACH\_QUORUM - glavna replika u svakom centru podataka mora da odgovori
* LOCAL\_ONE - samo jedna replika mora da odgovori. U klasteru koji sadrži različite centre podataka ovo takođe garantuje da se zahtevi za čitanje ne šalju udaljenom centru podataka
* ANY - jedna replika može da vrati odgovor ili koordinator može da sačuva „nagoveštaj“ (eng. „hint“). Ako se nagoveštaj sačuva koordinator će kasnije pokušati da reprodukuje nagoveštaj i da dostavi mutaciju na replike. Ovaj nivo konzistentnosti prihvaćen je samo od strane operacija za upis
* SERIAL - ovaj nivo konzistentnosti se koristi samo za upotrebu nad lakim transakcijama. Ekvivalentno je QUORUM-u.
* LOCAL\_SERIAL - isto kao SERIAL ali se koristi za lokalno održavanje doslednosti u okviru jednog centra podataka. Ekvivalentno LOCAL\_QUORUM-u.

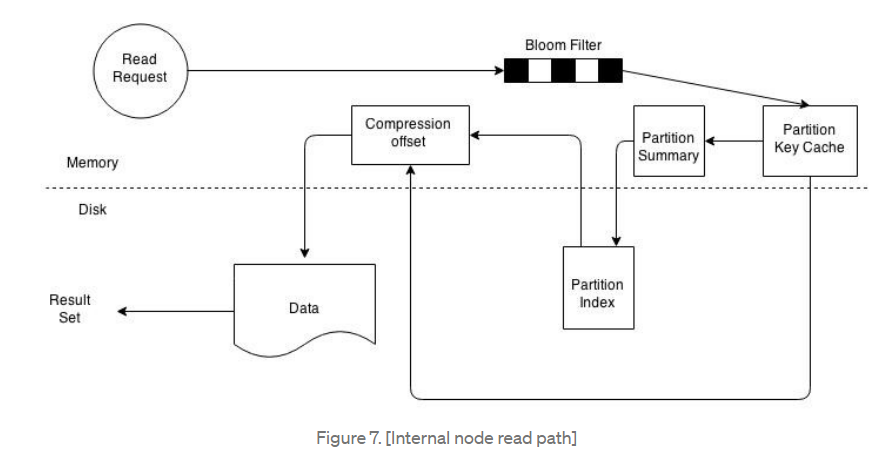
# Putanja za čitanje i upis

U perspektivi jednog čvora kada klijent zahteva da upiše podatke u Cassandrin čvor, zahtev se trajno upisuje u datoteci za evidencije (eng. „commit log file“) na disku, a zatim se podaci zapisuju u memorijsku tabelu koja se naziva „memtable“. Kada je memtable puna, nakon dostizanja unapred podešenog praga, vrši se upis na disk u nepromenljivoj strukturi koja se naziva SSTable. Svaka tabela kod Cassandre ima odgovarajuće memtable i SSTable. Na slici 8 prikazana je putanja za upis podataka kod Cassandre:



**Slika 8**: Putanja za upis na Cassandrinom čvoru [4]

Svaki Cassandrin čvor interno obrađuje podatke između memorije i diska koristeći mehanizme kako bi se izbegao što manje operacija pristupa disku, zbog toga koristi skup keš memorija i indeksa u memoriji da bi brže pronašao podatke na pravom mestu. Jedan od mehanizama je Bloom filter koji se nalazi u strukturi memorije i koji proverava da li postoje podaci redova u memtable-u pre pristupanja SSTables-u na disku. Drugi mehanizam je Particioni indeks koji čuva listu particijskih ključeva i početni položaj svakog redova u datoteci podataka koja je zapisana na disku. Ovaj proces predstavljen je na slici 9.



**Slika 9**: Putanja za čitanje internog čvora [5]

# Modeliranje podataka

Jedan od najtežih delova u radu sa Kasandrom, uglavnom zato što je to promena paradigme iz dobro poznatog sveta relacionih baza podataka. Prvo što se kao novi korisnik Cassandre primeti kada se započne sa radom jeste nedostatak „PRIDRUŽIVANJA“ (eng. „Joins“) između tabela. Tada se počinje sa razmišljanjem kako bi se mogli modelirati entiteti domena nekog poslovnog problema da bi se došlo do željenog rešenja? Odgovor je da se modeliranje ne realizuje na osnovu ključnih entiteta i njegovih veza sa ciljem da se normalizuju podaci, već treba da se modelira na osnovu upita na koji aplikacija treba da ispuni zahteve korisničkog interfejsa, stvarajući na taj način denormalizovani model. Dakle, ako je potrebno da se krera novi pogled istih podataka, preporučena praksa je stvaranje nove tabele (porodice kolona) za nju. Ova tehnika se naziva modeliranje zasnovano na upitima. Prvo se razmišlja o upitima koji trebaju da izvrše, a zatim da se na osnovu njih modeliraju tabele. Naravno, završiće se sa dupliciranim podacima, ali razlog za to je taj što se žrtvuje prostor na disku za bolje performanse čitanja, jer je prostor na disku jeftiniji.

Za potrebe izučavanja Apache Cassandra distribuirane baze, kreirane su nekoliko tabele, demonstrirani su upiti nad tim tabelama i kreiran je jedan mini projekat. Svi ovi primeri su dostupni na GitHub repozitorijumu [https://github.com/AleksandarPici/DBMS/](https://github.com/AleksandarPici/DBMS/tree/main/Seminarski%203/Kod).

# Zaključak

Jedna od stvari zbog kojih me impresionira Cassandra je nivo dostupnih opcija konfiguracije za prilagođavanje njenog ponašanja u skladu sa korisnikovim zahtevima, pa je kao distribuirana baza podataka spremna da donese visok nivo dostupnosti sistema bez ijedne tačke otkaza. Možda je neko čuo za Apache Cassandra i možda se ona čini zanimljivom za korišćenje u nekom projektu, preporuka je da se prvo proceni svaki poslovni zahtev i da se proveri da li taj projekat zahteva upotrebu ove vrste sistema za upravljanje bazama podataka, jer se u suprotnom vrlo lako može suočiti sa mnogim poteškoćama u primeni a što bi se moglo rešiti korišćenjem tradicionalnih relacionih baza podataka. Traba imati u vidu da je Cassandra stvorena za rešavanje specifičnih problema dostupnosti i brzine upisa i čitanja kod velikih količina podataka. U finansijskoj industriji postoje kompanije koje koriste Kasandru kao deo sistema za otkrivanje prevara. Dakle, predlog za one koji razmišljaju o korišćenju Cassandre, da započnu čitanje resursa za učenje, koji dostupnih na veb-u i od kompanija koje podržavaju razvoj Cassandre. Svakako, korišćenjem Cassandre kao distribuirano skladište, sigurno će pomoći u donošenju jasne i asertivne odluke za izgranju nekog dobrog rešenja.

# Literatura

1. CAP teorema <https://en.wikipedia.org/wiki/CAP_theorem>
2. Peer-to-peer komunikacija <https://www.computerhope.com/jargon/p/ptpcom.htm>
3. TCP komunikacija <https://www.guru99.com/tcp-3-way-handshake.html>
4. Putanja za upisi na Cassandrinom čvoru [https://medium.com](https://medium.com/@brunocrt/the-distributed-architecture-behind-cassandra-database-fba8b5cc4785)
5. Putanja za čitanje na Cassandrinom čvoru [https://medium.com/read](https://medium.com/@brunocrt/the-distributed-architecture-behind-cassandra-database-fba8b5cc4785)
6. Apache Cassandra <https://cassandra.apache.org/>
7. Learn How to Succeed with Apache Cassandra <https://www.datastax.com/dev>