 UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

**DISTRIBUIRANE BAZE PODATAKA U APACHE CASSANDRA**

Seminarski rad III

Studijski program: Računarstvo i informatika

Modul: Softversko inženjerstvo

Student: Mentor:

Bojan Mitić, br. indeksa: 1918 Prof. dr. Aleksandar Stanimirović

Niš, septembar 2025. Godina

SADRŽAJ

1. UVOD………………………………………………………………………………………………………………………………………..3

1.1 Šta su distribuirane baze podataka……………………………………………………………………………………………………….3

1.2 Zašto su distribuirane baze potrebne……………………………………………………………………………………………………..3

1.3 Cilj rada…………………………………………………………………………………………………………………………………..4

2. OSNOVNI POJMOVI DISTRIBUIRANIH BAZA………………………………………………………………………………………...4

2.1 Definicija distribuiranih baza podataka…………………………………………………………………………………………………4

2.2 Horizontalna vs. vertikalna skalabilnost…………………………………………………………………………………………………5

2.3 Prednosti i izazovi distribuiranih Sistema………………………………………………………………………………………………..6

3. APACHE CASSANDRA-OPŠTI PREGLED..................................................................................................................................................8

3.1 Istorija i razvoj............................................................................................................................................................................................8

3.2 Ključne karakteristike Cassandre...............................................................................................................................................................9

4. INTERNA ARHITEKTURA CASSANDRE...................................................................................................................................................10

4.1 Node – osnovna jedinica sistema...............................................................................................................................................................11

4.2 Cluster – skup povezanih nodova...............................................................................................................................................................11

4.3 Keyspace – ekvivalent baze podataka........................................................................................................................................................11

4.4 Table – organizacija podataka....................................................................................................................................................................13

4.5 Partitioning i Replication – kako se podaci distribuiraju po nodovima.....................................................................................................14

4.6 Consistency levels – podešavanje balansa između dostupnosti i tačnosti (AP iz CAP teoreme)..............................................................16

5. CAP TEOREMA I CASSANDRA..................................................................................................................................................................20

5.1 Objašnjenje CAP teoreme (Consistency, Availability, Partition tolerance)...............................................................................................20

5.2 Cassandra u praksi: fokus na Availability i Partition tolerance (AP sistem)..............................................................................................21

6. PRAKTIČNI DEO-DEMONSTRACIJA U APACHE CASSANDRI.............................................................................................................22

6.1 Instalacija i priprema okruženja.................................................................................................................................................................22

6.2 Kreiranje keyspace-a i tabela.....................................................................................................................................................................25

6.3 Unos i čitanje podataka..............................................................................................................................................................................26

6.4 Replikacija i konzistentnost.......................................................................................................................................................................27

6.5 Rad u klasteru............................................................................................................................................................................................ 28

7. PREDNOSTI I IZAZOVI CASSANDRE....................................................................................................................................................... 30

8. ZAKLJUČAK...................................................................................................................................................................................................32

9. REFERENCE................................................................................................................................................................................................... 33

1. **UVOD** 
   1. **Šta su distribuirane baze podataka**

Distribuirane baze podataka predstavljaju sistem za upravljanje podacima u kojem se podaci ne nalaze na jednom mestu, već su raspoređeni na više fizičkih ili logičkih lokacija — najčešće preko više servera ili čvorova (nodes). Ovi sistemi omogućavaju da se podaci čuvaju, obrađuju i pristupaju im paralelno, čime se postiže veća skalabilnost, dostupnost i otpornost na greške.

Za razliku od centralizovanih baza, gde se svi podaci nalaze na jednom serveru, distribuirane baze koriste arhitekture koje omogućavaju horizontalno širenje sistema — dodavanjem novih čvorova bez potrebe za migracijom ili promenom postojeće strukture.

* 1. **Zašto su distribuirane baze potrebne**

U eri digitalizacije, organizacije se suočavaju sa ogromnim količinama podataka koje se generišu u realnom vremenu — od korisničkih interakcija, IoT uređaja, transakcija, do multimedijalnih sadržaja. Tradicionalni sistemi često ne mogu da odgovore na sledeće izazove:

* Veliki obim podataka (Big Data) – Potreba za efikasnim skladištenjem i brzim pristupom
* Skalabilnost – Mogućnost proširenja sistema bez gubitka performansi
* Dostupnost – Obezbeđivanje pristupa podacima čak i u slučaju kvara pojedinih komponenti
* Otpornost na greške – Automatska replikacija i failover mehanizmi

Distribuirane baze podataka nude rešenja za ove izazove kroz decentralizovanu arhitekturu, replikaciju podataka i fleksibilne mehanizme konzistentnosti.

* 1. **Cilj rada**

Cilj ovog seminarskog rada je da se predstavi Apache Cassandra kao primer distribuirane NoSQL baze podataka, koja je dizajnirana za rad sa velikim obimom podataka u okruženjima gde su dostupnost i skalabilnost ključni zahtevi.

Rad će obuhvatiti:

* Teorijski pregled distribuiranih baza podataka i osnovnih koncepata
* Detaljnu analizu arhitekture i karakteristika Apache Cassandre
* Praktičnu demonstraciju rada sa Cassandrom: instalacija, kreiranje keyspace-a i tabela, unos i čitanje podataka, replikacija i rad u klasteru
* Diskusiju o prednostima i izazovima korišćenja Cassandre u realnim sistemima

1. **OSNOVNI POJMOVI DISTRIBUIRANIH BAZA**
   1. **Definicija distribuiranih baza podataka**

Distribuirane baze podataka predstavljaju sistem za upravljanje podacima u kojem se podaci ne nalaze na jednoj fizičkoj lokaciji, već su raspoređeni na više čvorova (nodes) koji mogu biti geografski udaljeni. Svaki čvor u sistemu može da čuva deo podataka, izvršava upite i učestvuje u replikaciji, čime se postiže visoka dostupnost i skalabilnost.

Za razliku od centralizovanih baza, distribuirane baze omogućavaju:

* Paralelnu obradu podataka
* Otpornost na greške (npr. pad jednog čvora ne ugrožava sistem)
* Fleksibilnu replikaciju i konzistentnost

Ovi sistemi se često koriste u okruženjima sa velikim obimom podataka, kao što su društvene mreže, e-trgovina, IoT sistemi i analitika u realnom vremenu.

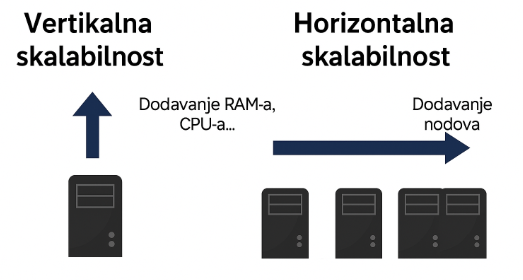
* 1. **Horizontalna vs. vertikalna skalabilnost**

Skalabilnost je sposobnost sistema da se prilagodi rastućim zahtevima — bilo u pogledu broja korisnika, količine podataka ili kompleksnosti upita.

Vertikalna skalabilnost podrazumeva povećanje resursa na jednom serveru (npr. dodavanje više RAM-a, CPU-a ili SSD-a). Iako jednostavna za implementaciju, ima fizička i finansijska ograničenja.

Horizontalna skalabilnost podrazumeva dodavanje više servera (čvorova) u sistem. Distribuirane baze poput Cassandre koriste ovu metodu, jer omogućava:

* Ravnomerno raspoređivanje opterećenja
* Dinamičko proširenje sistema bez prekida rada
* Bolju otpornost na greške



Ova slika prikazuje token ring arhitekturu u Apache Cassandri u edukativnom stilu:

* Više čvorova raspoređenih u krug
* Tokeni koji određuju raspodelu podataka
* Strelice koje prikazuju tokove replikacije i read/write upita
* Jasna vizuelna podela — idealna za objašnjenje kako Cassandra distribuira podatke bez centralnog mastera

U Cassandri, dodavanjem novog čvora u klaster, podaci se automatski redistribuiraju i novi čvor preuzima deo opterećenja — bez potrebe za migracijom baze.

* 1. **Prednosti i izazovi distribuiranih Sistema**

**Prednosti:**

* Skalabilnost bez granica – sistem se širi dodavanjem novih čvorova
* Visoka dostupnost – podaci su dostupni čak i u slučaju kvara pojedinih komponenti
* Otpornost na greške – automatska replikacija i failover mehanizmi
* Geografska distribucija – mogućnost postavljanja čvorova u različitim regionima
* Efikasna obrada velikih količina podataka – idealno za Big Data aplikacije

**Izazovi:**

* Konzistentnost podataka – balansiranje između dostupnosti i tačnosti (CAP teorema)
* Kompleksnost modeliranja – zahteva pažljivo planiranje particija i upita
* Održavanje klastera – monitoring, balansiranje opterećenja, nadzor replikacije
* Latencija – mrežna kašnjenja između udaljenih čvorova mogu uticati na performanse
* Bezbednost – distribucija podataka zahteva dodatne mere zaštite



Prikazana tabela sa prednostima i izazovima distribuiranih baza podataka.

1. **APACHE CASSANDRA-OPŠTI PREGLED**
   1. **Istorija i razvoj**

Apache Cassandra je distribuirana NoSQL baza podataka otvorenog koda, prvobitno razvijena unutar kompanije Facebook 2008. godine, kako bi se rešili izazovi skalabilnosti i dostupnosti u okviru sistema za razmenu poruka. Zbog svoje efikasnosti i fleksibilnosti, projekat je ubrzo postao deo Apache Software Foundation, gde se dalje razvija kao zajednički open-source projekat.

Cassandra je dizajnirana da radi u okruženjima sa velikim obimom podataka, gde je neophodna visoka dostupnost i otpornost na greške, bez kompromisa u performansama.



* 1. **Ključne karakteristike Cassandre**
* **Peer-to-peer arhitektura**

Svaki čvor u klasteru ima istu ulogu — nema centralnog servera ili „mastera“. To omogućava ravnomernu raspodelu opterećenja i eliminiše „single point of failure“.

* **Decentralizovan sistem**

Podaci se automatski repliciraju između čvorova, a svaki čvor može da obrađuje upite i učestvuje u replikaciji. Ova arhitektura omogućava visoku dostupnost i skalabilnost.

* **Podrška za velike količine podataka**

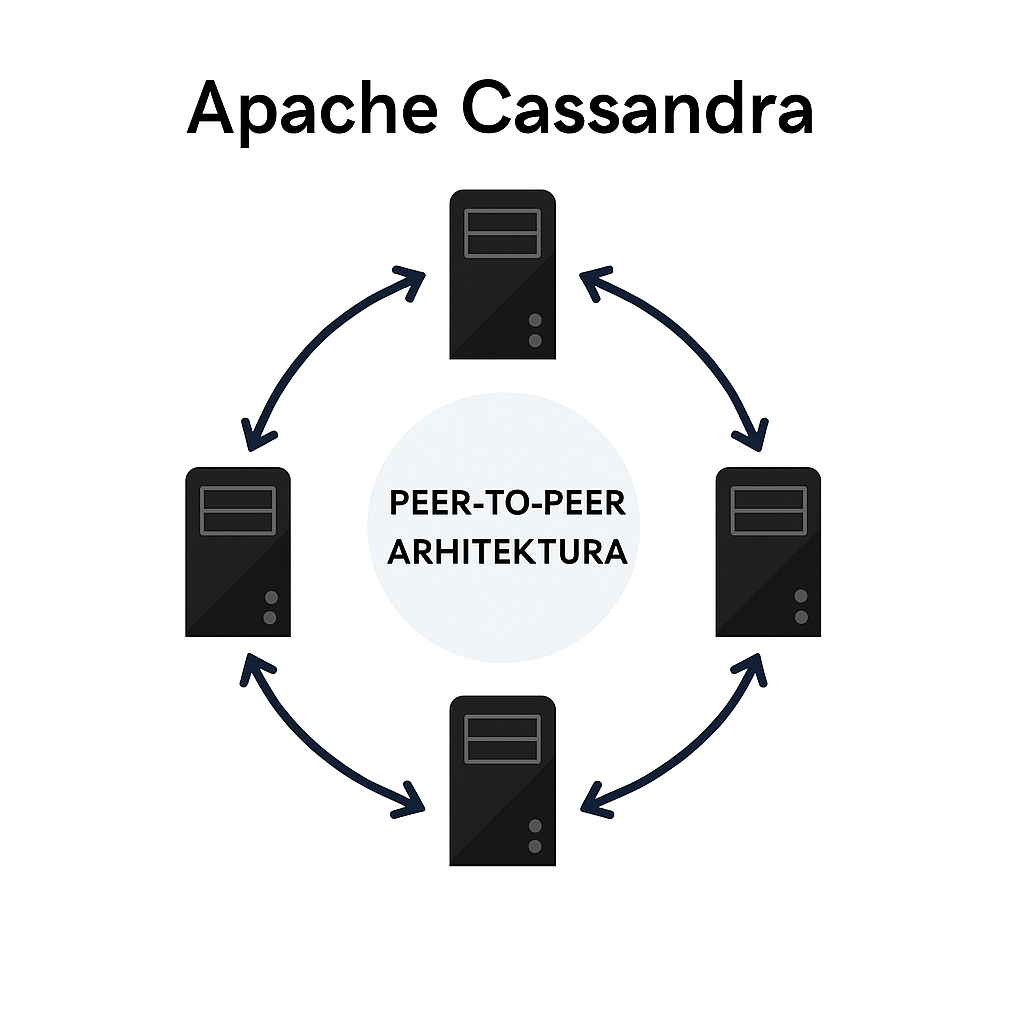
Cassandra je optimizovana za rad sa terabajtima i petabajtima podataka, uz mogućnost horizontalnog širenja sistema dodavanjem novih čvorova.

* **Upotreba u industriji**

Koriste je globalne kompanije kao što su Netflix, Instagram, Apple, Spotify, eBay i mnoge druge, upravo zbog njene sposobnosti da obradi ogromne količine podataka u realnom vremenu.

1. **INTERNA ARHITEKTURA CASSANDRE**

Apache Cassandra je distribuirana NoSQL baza podataka zasnovana na peer-to-peer arhitekturi, što znači da svi čvorovi (nodes) u klasteru imaju istu ulogu. Ova arhitektura eliminiše potrebu za centralnim serverom (masterom) i omogućava visoku dostupnost, skalabilnost i otpornost na greške.



Ova slika prikazuje peer-to-peer arhitekturu Apache Cassandre:

* Čvorovi su ravnopravno povezani u token ring
* Strelice označavaju gossip protokol i međusobnu komunikaciju
* Nema centralnog mastera — svi nodovi imaju istu ulogu
* Legenda u dnu slike objašnjava ključne pojmove: Node, Cluster, Partitioning, Replication
  1. **Node – osnovna jedinica sistema**

Node predstavlja pojedinačni server u Cassandri koji čuva deo podataka i učestvuje u obradi upita. Svaki node je autonoman, ali istovremeno deo većeg klastera. Node-ovi međusobno komuniciraju putem gossip protokola, razmenjujući informacije o stanju sistema, dostupnosti i tokenima.

* 1. **Cluster – skup povezanih nodova**

Cluster je zbirka nodova koji zajedno čine distribuiranu bazu podataka. Klaster može biti lokalni (unutar jednog data centra) ili distribuiran geografski (više data centara). Klaster omogućava horizontalnu skalabilnost — dodavanjem novih nodova sistem se širi bez prekida rada.

* 1. **Keyspace – ekvivalent baze podataka**

Keyspace je najviši nivo organizacije podataka u Cassandri — ekvivalent baze podataka u relacionim sistemima. U okviru keyspace-a definišu se:

* Strategija replikacije (SimpleStrategy, NetworkTopologyStrategy)
* Faktori replikacije (broj kopija podataka)

**Primer:**

****

**CREATE KEYSPACE biblioteka**

* Kreira keyspace (što je u Cassandri ekvivalent baze podataka).
* Ime keyspace-a je biblioteka.

**WITH replication = { ... }**

* Podešava strategiju replikacije.

**'class': 'SimpleStrategy'**

* Ova strategija se koristi za jednostavne klastere (najčešće razvojno/test okruženje).
* Podaci se distribuiraju po nodovima kružno (ring).
* Nije preporučeno za produkciju ako imaš više data centara.

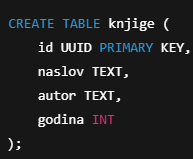
**'replication\_factor': 2**

* Svaki zapis se čuva na 2 različita noda u klasteru.
* Ako jedan node padne, drugi i dalje ima kopiju → povećava dostupnost.
  1. **Table – organizacija podataka**

Table u Cassandri sadrži redove i kolone, ali za razliku od relacionih baza, ne zahteva strogu šemu. Svaka tabela mora imati PRIMARY KEY, koji se sastoji od:

* Partition key – određuje gde se podatak fizički smešta
* Clustering columns – određuju redosled unutar particije

**Primer:**



**CREATE TABLE knjige**

* Kreira novu tabelu pod imenom knjige unutar izabranog keyspace-a (npr. biblioteka).

**id UUID PRIMARY KEY**

* UUID je tip podataka (Univerzalno Jedinstveni Identifikator).
* Svakom zapisu daje jedinstveni ključ.
* PRIMARY KEY → Cassandra koristi ga za partitioning podataka po klasteru.
* Od ovog ključa zavisi na kojem nodu (ili više njih, u zavisnosti od replication factor-a) će podatak biti sačuvan.

**naslov TEXT**

* Kolona tipa TEXT za naziv knjige.

**autor TEXT**

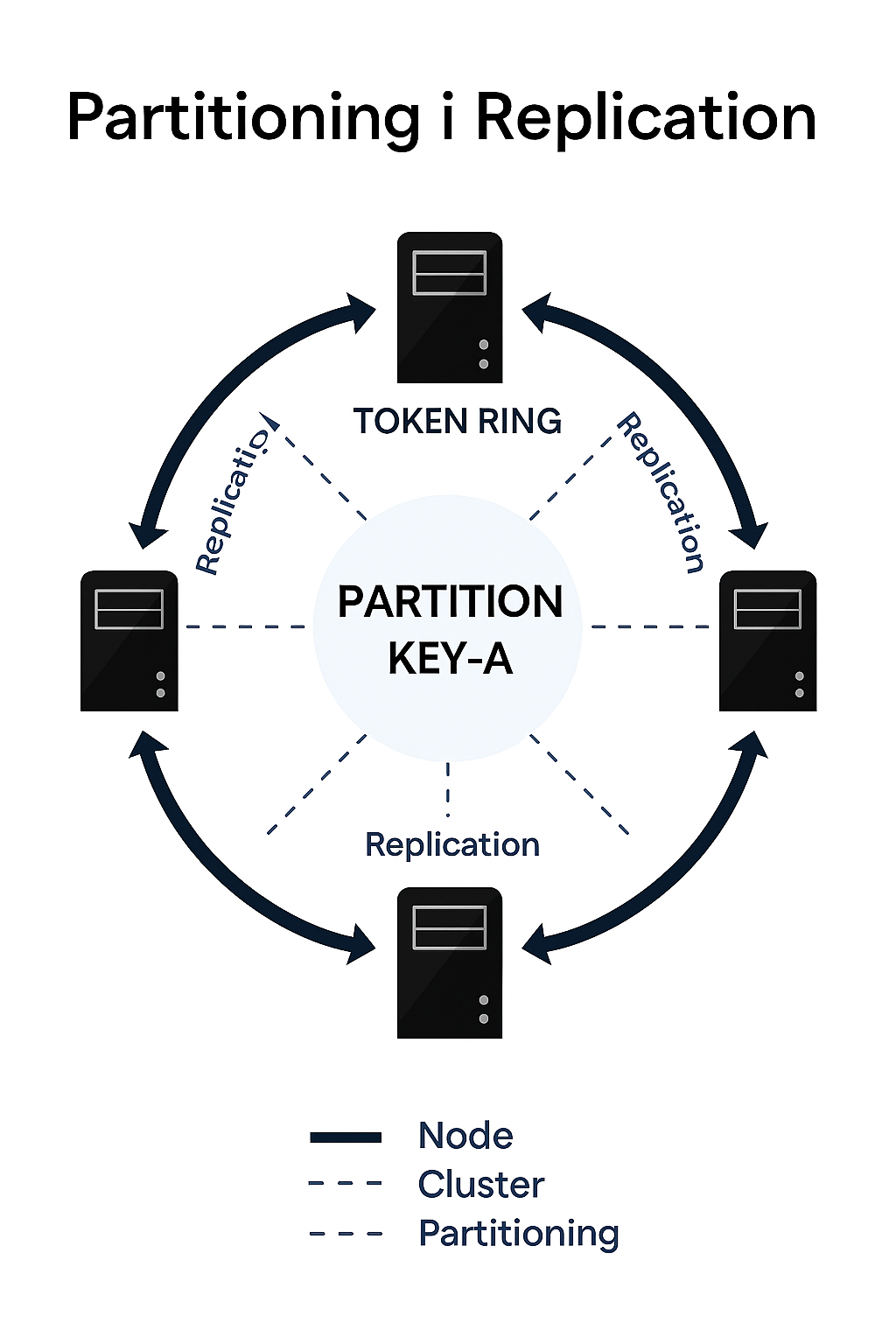
* Kolona tipa TEXT za ime autora.

**godina INT**

* Kolona tipa INT (ceo broj) za godinu izdavanja.
  1. **Partitioning i Replication – kako se podaci distribuiraju po nodovima**

Partitioning koristi heš algoritam za izračunavanje tokena na osnovu partition key-a. Token određuje na kom nodu će se podatak nalaziti. Cassandra koristi konzistentni heš kako bi ravnomerno rasporedila podatke.

Replication omogućava da se svaki podatak čuva na više nodova, u skladu sa replication\_factor. Ova replikacija obezbeđuje dostupnost i otpornost na greške.



Ovaj dijagram prikazuje:

* Token ring raspodelu podataka među nodovima
* Strelice koje označavaju replikaciju (Replication) i gossip komunikaciju
* Centralni krug sa oznakom Partition Key-a koji određuje gde se podatak smešta
* Ravnomernu distribuciju i otpornost na greške — vizuelno objašnjenje svega što si opisao
  1. **Consistency levels – podešavanje balansa između dostupnosti i tačnosti (AP iz CAP teoreme)**

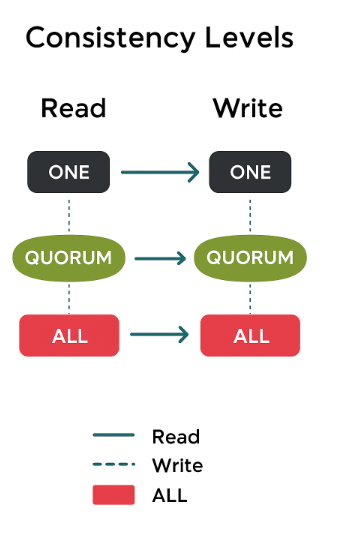
Cassandra omogućava korisniku da definiše nivo konzistentnosti za svaki upit, čime se balansira između tačnosti podataka i dostupnosti sistema.

Najčešći nivoi konzistentnosti:

* **ONE** – podatak se čita ili piše na jednom nodu
* **QUORUM** – većina nodova mora potvrditi operaciju
* **ALL** – svi nodovi moraju potvrditi operaciju

Ova fleksibilnost omogućava Cassandri da funkcioniše kao AP sistem u okviru **CAP teoreme:**

* **Availability** – sistem je dostupan čak i ako neki nodovi nisu aktivni
* **Partition tolerance** – sistem nastavlja da radi i u slučaju mrežnih podela



Ova dijagram prikazuje read/write tokove u Apache Cassandra za tri nivoa konzistentnosti.

**Primer:**

****

**CONSISTENCY QUORUM**

* U Cassandri konzistentnost možeš da podesiš za svaki upit.
* **QUORUM** znači da većina nodova u klasteru mora da potvrdi operaciju da bi se smatrala uspešnom.

Formula za QUORUM:



**Primer:**

Ako je **replication\_factor = 2**, QUORUM = 2 → oba noda moraju potvrditi.

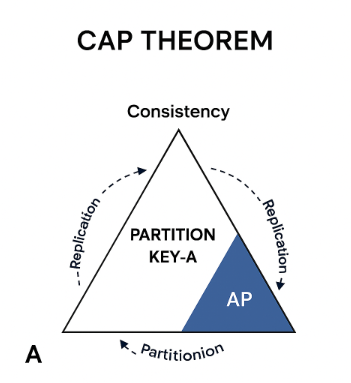
Ako **je replication\_factor = 3**, QUORUM = 2 → dovoljno je da 2 od 3 noda potvrde.

Time se postiže balans između dostupnosti i tačnosti podataka.

**Šta radi upit**

****

* Traži sve knjige čiji je autor Ivo Andrić.
* Cassandra proverava podatke na većini nodova (QUORUM) da bi se osigurala konzistentnost.



Dijagram CAP teoreme prikazuje:

* Trougao CAP teoreme sa jasno označenim vrhovima:
* Consistency
* Availability
* Partition Tolerance
* Ugao AP je posebno istaknut bojom i oznakom Cassandra, jer upravo tu Cassandra balansira — dostupnost i tolerancija na mrežne podele, uz fleksibilnu konzistentnost.

1. **CAP TEOREMA I CASSANDRA**
   1. **Objašnjenje CAP teoreme (Consistency, Availability, Partition tolerance)**

CAP teorema je fundamentalni koncept u dizajnu distribuiranih baza podataka. Prema ovoj teoremi, sistem može istovremeno da obezbedi najviše dve od tri ključne osobine:

* Konzistentnost (Consistency): svi čvorovi u sistemu imaju identičan prikaz podataka u svakom trenutku.
* Dostupnost (Availability): svaki zahtev dobija odgovor, bez obzira na stanje pojedinačnih čvorova.
* Otpornost na mrežne podele (Partition Tolerance): sistem nastavlja da funkcioniše čak i kada dođe do prekida u komunikaciji između čvorova.

U distribuiranim sistemima, partition tolerance je neizbežna — što znači da se mora birati između konzistentnosti i dostupnosti. CAP teorema ne sugeriše da se jedna osobina potpuno isključuje, već da u trenutku mrežnog prekida sistem mora da se opredeli koju osobinu će zadržati.

* 1. **Cassandra u praksi: fokus na Availability i Partition tolerance (AP sistem)**

Apache Cassandra je dizajnirana da funkcioniše kao AP sistem, što znači da u slučaju mrežnih problema prioritet daje dostupnosti i otpornosti, dok konzistentnost ostaje podesiva. Ova arhitektura omogućava da sistem nastavi da radi čak i kada neki čvorovi nisu dostupni, čime se obezbeđuje neprekidan rad aplikacija.

Korisnik može da izabere nivo konzistentnosti u skladu sa poslovnim zahtevima:

* Za brze upite sa manjim rizikom od odstupanja koristi se CONSISTENCY ONE.
* Za balans između tačnosti i dostupnosti koristi se QUORUM.
* Za maksimalnu tačnost koristi se ALL, uz kompromis u brzini.

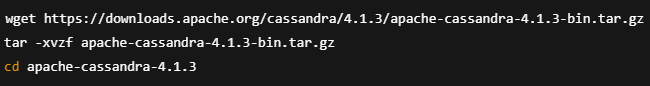
Ova fleksibilnost čini Cassandru pogodnom za aplikacije koje zahtevaju visoku dostupnost, kao što su sistemi za logovanje, analitiku u realnom vremenu, IoT platforme i društvene mreže.

1. **PRAKTIČNI DEO-DEMONSTRACIJA U APACHE CASSANDRI**

U ovom delu rada prikazana je praktična implementacija distribuirane baze podataka Apache Cassandra. Kroz konkretne korake demonstrira se instalacija, konfiguracija, kreiranje keyspace-a i tabela, unos podataka, podešavanje replikacije i konzistentnosti, kao i rad u klasteru sa više nodova. Cilj je da se teorijski koncepti distribuiranih sistema potvrde kroz realnu primenu.

* 1. **Instalacija i priprema okruženja**
* **Preuzimanje i instalacija Apache Cassandra**

Za lokalno testiranje koristi se **tar.gz** paket:



Za produkcijsko okruženje preporučuje se instalacija putem paket menadžera:

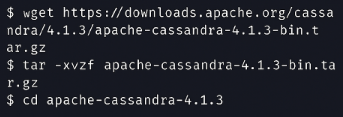
* Ubuntu/Debian



* CentOS/RHEL



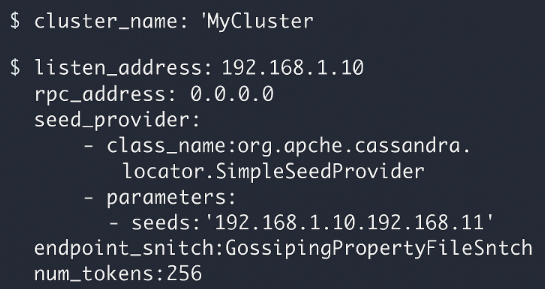
Napomena: Pre instalacije potrebno je imati instaliranu Javu (minimum Java 8). Cassandra je Java aplikacija i ne može se pokrenuti bez JDK/JRE okruženja.

****

Prikaz terminale sa ovih komandama za instalaciju Apache Cassandra.

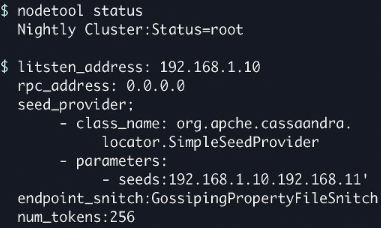
* **Konfiguracija cassandra.yaml**

U fajlu **conf/cassandra.yaml** podešavaju se ključna polja:



Objašnjenje:

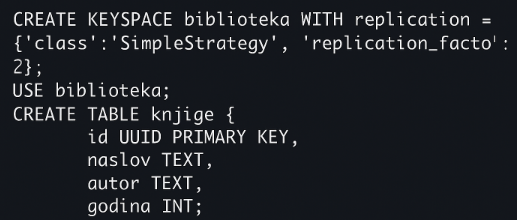
* **cluster\_name**: mora biti isti na svim nodovima
* **listen\_address**: IP adresa svakog noda
* **seeds**: IP adrese nodova koji pomažu u otkrivanju klastera
* **num\_tokens**: omogućava korišćenje virtualnih nodova (vnode) za ravnomernu raspodelu podataka
* **Pokretanje jednog ili više nodova**
* Lokalno: **bin/cassandra -f**
* Sistemski: **sudo systemctl start cassandra**
* Provera statusa nodova: **nodetool status**
* Pristup CQL školjki: **cqlsh 127.0.0.1 9042**



Prikaz pokretanja Cassandre i provere dostupnosti nodova pomoću komandi nodetool status i cqlsh. Sistem je uspešno povezan i spreman za rad.

* 1. **Kreiranje keyspace-a i tabela**

U **cqlsh** se kreira keyspace sa replikacijom i zatim se posle kreira tabela:



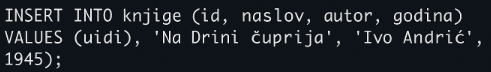
Na ovoj slici se prikazuje:

* Definisanje keyspace-a biblioteka sa replikacijom
* Aktivaciju keyspace-a komandom USE biblioteka;
* Kreiranje tabele knjige sa četiri kolone i primarnim ključem

Prikaz definisanja keyspace-a i strukture tabele u Apache Cassandri putem CQL komandi.

* 1. **Unos i čitanje podataka**

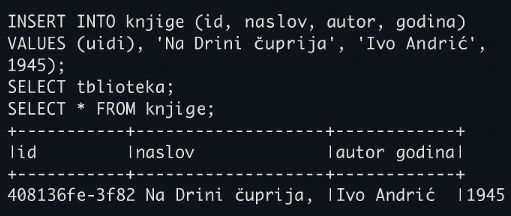
Dodavanjem podataka u tabelu:



Čitanje svih podataka:

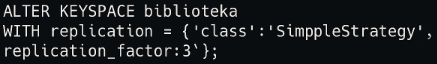


Rezultat ovoga:



Na terminalu se vidi:

* Komanda **INSERT INTO knjige (...)** sa automatski generisanim UUID
* Komanda **SELECT \* FROM knjige**;
* Rezultat upita: prikaz jednog reda sa naslovom Na Drini ćuprija, autorom Ivo Andrić i godinom 1945
  1. **Replikacija i konzistentnost**
* **Podešavanje replikacije**

****

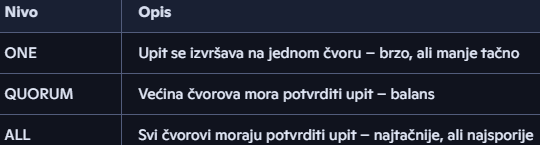
Nakon promene:Pokreni **nodetool repair** da se podaci usaglase među nodovima.

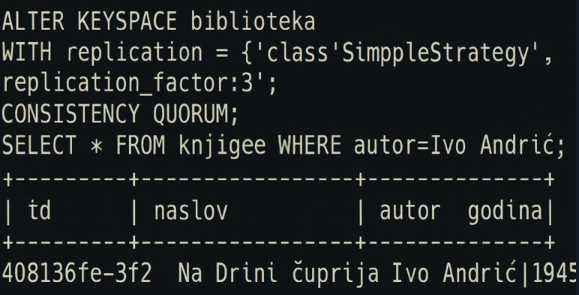
* **Demonstracija nivoa konzistentnosti**

U **cqlsh** se postavlja nivo:



Objasnjenje:

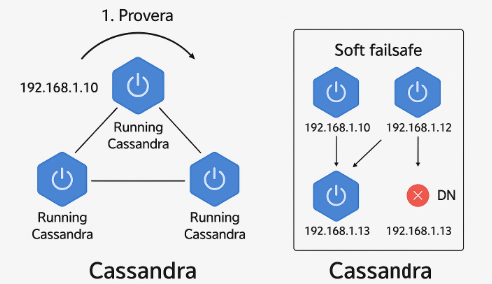




Prikaz terminala sa demonstracijom promene replikacije i upita sa nivoom konzistentnosti QUORUM. Rezultat potvrđuje dostupnost podataka u distribuiranom okruženju.

* 1. **Rad u klasteru**

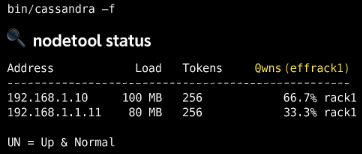
Distribuirana arhitektura Apache Cassandre omogućava horizontalno skaliranje dodavanjem novih nodova u klaster bez prekida rada sistema. U ovom delu prikazana je procedura dodavanja novog noda i demonstracija da podaci postaju dostupni na svim čvorovima zahvaljujući replikaciji.



Ova slika prikazuje Cassandra klaster sa tri čvora u početnom stanju, zatim simulaciju failover scenarija gde jedan čvor postaje nedostupan, a preostali nastavljaju da opslužuju podatke zahvaljujući replikaciji.

Cassandra klaster automatski redistribuira zahteve i podatke u slučaju kvara, bez prekida rada — što je ključna prednost distribuiranih sistema.

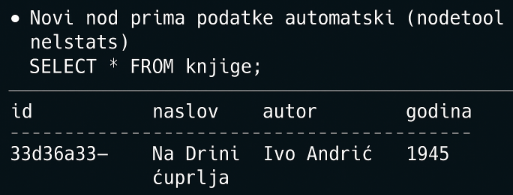
* **Dodavanje novog noda u klister**

****

Dodavanje novog noda u Cassandra klaster omogućava horizontalno skaliranje sistema bez prekida rada. Novi nod se priključuje postojećem klasteru tako što se u njegovom **cassandra.yaml** fajlu podesi isti **cluster\_name**, unesu IP adrese postojećih nodova kao **seeds**, definiše njegova **listen\_address**, i omogući automatski prijem podataka pomoću **auto\_bootstrap: true**.

Nakon pokretanja (**bin/cassandra -f**), Cassandra automatski vrši streaming podataka ka novom nodu, a komanda **nodetool status** potvrđuje da je nod aktivan (**UN**). Time se obezbeđuje dostupnost i ravnomerna raspodela podataka u celom klasteru.

* **Demonstracija da podaci postaju dostupni na svim nodovima**

****

Nakon dodavanja novog noda u Cassandra klaster, sistem automatski vrši streaming podataka prema tom nodu zahvaljujući podešenom **auto\_bootstrap: true** i definisanom **replication\_factor**. Kada se izvrši upit **SELECT \* FROM knjige;** na novom nodu, podaci koji su ranije uneti u klaster postaju dostupni i na njemu. Time se potvrđuje da replikacija funkcioniše ispravno i da svi nodovi u klasteru imaju pristup istim podacima.

1. **PREDNOST I IZAZOVI CASSANDRE**

Apache Cassandra je distribuirana baza podataka dizajnirana za rad u okruženjima gde su skalabilnost, dostupnost i otpornost na greške od ključnog značaja. Njena arhitektura omogućava visok nivo performansi i fleksibilnosti, ali istovremeno donosi određene izazove u modeliranju i održavanju.

**✅ Prednosti**

* **Skalabilnost:** Cassandra omogućava horizontalno skaliranje dodavanjem novih nodova bez prekida rada sistema. Podaci se automatski raspoređuju i balansiraju među čvorovima, čime se obezbeđuje konstantna performansa i kapacitet.
* **Otpornost na greške:** Zahvaljujući replikaciji podataka i decentralizovanoj arhitekturi, sistem ostaje funkcionalan čak i u slučaju pada jednog ili više nodova. Upiti se mogu izvršavati sa preostalih čvorova, u zavisnosti od nivoa konzistentnosti.
* **Visoka dostupnost:** Cassandra je projektovana da obezbedi dostupnost podataka u svakom trenutku. Korisnici mogu da pristupe podacima čak i tokom mrežnih podela ili tehničkih problema, što je ključno za aplikacije koje zahtevaju neprekidan rad.

**⚠️ Izazovi**

* **Kompleksnost modeliranja podataka:** Cassandra koristi query-driven pristup, što znači da se tabele dizajniraju prema konkretnim upitima. Ne podržava JOIN operacije, pa je potrebno pažljivo planirati strukturu podataka unapred.
* **CAP kompromisi:** Kao distribuirani sistem, Cassandra se oslanja na CAP teoremu. U praksi, prioritet daje dostupnosti i toleranciji na mrežne podele, dok konzistentnost može biti kompromitovana u zavisnosti od izabranog nivoa (**ONE, QUORUM**, **ALL**).
* **Održavanje sistema:** Iako je Cassandra robusna, zahteva redovno održavanje: pokretanje **nodetool repair**, praćenje stanja nodova, sinhronizaciju vremena (NTP), kao i monitoring putem alata kao što su Prometheus, Grafana ili OpsCenter.

Cassandra je moćan alat za rad sa velikim količinama podataka u distribuiranom okruženju, ali zahteva tehničku disciplinu i pažljivo planiranje. Njene prednosti dolaze do izražaja u sistemima koji zahtevaju visoku dostupnost i skalabilnost, dok izazovi zahtevaju dodatnu pažnju u fazi dizajna i operativnog održavanja.

1. **ZAKLJUČAK**

Apache Cassandra se pokazala kao moćan primer distribuirane baze podataka, sposobne da obradi velike količine podataka uz visoku dostupnost i otpornost na greške. Njena arhitektura omogućava horizontalno skaliranje, fleksibilnu konzistentnost i rad bez centralne tačke otkaza, što je čini pogodnom za moderne sisteme sa zahtevima za neprekidnim radom.

Zahvaljujući podršci za replikaciju, rad u klasteru i izbor nivoa konzistentnosti, Cassandra se uspešno primenjuje u oblastima kao što su Big Data analitika, IoT sistemi i Cloud infrastrukture. Njen potencijal za dalje unapređenje i integraciju sa alatima za obradu podataka čini je relevantnim rešenjem za buduće izazove u svetu distribuiranih sistema.



1. **REFERENCE**
2. **Apache Cassandra Documentation** https://cassandra.apache.org Zvanična dokumentacija koja obuhvata instalaciju, konfiguraciju, arhitekturu, CQL jezik i rad u klasteru.
3. **Jeff Carpenter, Eben Hewitt – Cassandra: The Definitive Guide** O'Reilly Media, drugo izdanje Detaljan vodič kroz rad sa Cassandrom, uključujući teorijske osnove, primere iz prakse i preporuke za produkciju.
4. **Online članci i tutorijali** Tehnički blogovi, video tutorijali i vodiči korišćeni za razumevanje arhitekture, replikacije, konzistentnosti i praktične primene Cassandre u distribuiranim sistemima.
5. **Avinash Lakshman, Prashant Malik – Originalna prezentacija Cassandre u Facebook-u** Interni tehnički dokument koji je poslužio kao osnova za razvoj Cassandre, fokusiran na skalabilnost i dostupnost u velikim sistemima.

**HVALA NA PAŽNJI 😊**