NoSQL baze podataka – projekt u Cassandri

Sabolić, David

Sveučilište u Rijeci – Fakultet informatike i digitalnih tehnologija

Poslovna informatika

David Sabolić

NoSQL baze podataka – projekt u Cassandri

Diplomski rad

Mentor: doc. dr. sc. Danijela Jakšić

Rijeka, kolovoz 2023.

Sveučilište u Rijeci
FIDIT Fakultet informatike i digitalnih tehnologija
www.inf.uniri.hr

Rijeka, 12. lipanj 2023.

Zadatak za diplomski rad

Pristupnik: David Sabolić

Naziv diplomskog rada: NoSQL baze podataka - projekt u Cassandri

Naziv diplomskog rada na eng. jeziku: NoSQL data stores - Cassandra project

Sadržaj zadatka: Cilj je diplomskog rada istražiti NoSQL baze podataka s fokusom na Cassandru. NoSQL baze podataka dobivaju sve veću pažnju posljednjih godina zbog mogućnosti obrade veće količine nestrukturiranih podataka. U prvom djelu ovog rada biti će dan pregled NoSQL baza podataka, objasniti će se osnovni koncepti, koje sve vrste NoSQL baza postoje te prednosti i nedostaci u usporedbi s relacijskim bazama podataka. Drugi dio u fokusu bi imao samu Apache Cassandru te upoznavanje s njenim ključnim konceptima, arhitekturom, sintaksom i funkcionalnostima, nad vlastitim primjerima i skupovima podataka.

Mentor:

Doc. dr. sc. Danijela Jakšić

Komentor:

Voditeljica za diplomske radove:

Prof. dr. sc. Ana Meštrović

Zadatak preuzet: 12. lipanj 2023.

(potpis pristupnika)

David Sabolić

Adresa: Radmile Matejčić 2
51000 Rijeka, Hrvatska
Tel: +385(0)51 584 700
E-mail: ured@inf.uniri.hr
OIB: 64218323816
IBAN: HR1524020061400006366

Sadržaj

Sažetak

Ovaj diplomski rad istražuje nastanak NoSQL baza razmatrajući razloge koji su doveli do njihova stvaranja i ključne čimbenike koji su utjecali na njihovu široku primjenu.

Fokus prvoga poglavlja je na upoznavanju s NoSQL bazama podataka. Istražiti će se bitne karakteristike koje NoSQL pruža u usporedbi s tradicionalnim SQL sustavima. Razvoj NoSQL tehnologije i njezina primjena čine drugi dio tog poglavlja, dok završni dio poglavlja stavlja u fokus neke razlike koje SQL baze podataka imaju s NoSQL bazama.

U drugom poglavlju, analizirati će se jedna od najpoznatijih NoSQL baza podataka - Apache Cassandru. Razmatrati će se ključni koncepti tehnologije ove baze te pružati uvid u arhitekturu i objekte koje koristi za pohranu i upravljanje podacima.

Treće poglavlje sadrži početak praktičnog rada u bazi podataka. Proučiti će se koraci instalacije, konfiguracije te unos podataka, prikazati kako stvoriti bazu i koristiti njezine mogućnosti kroz upoznavanje s CQL (Cassandra Query Language) .

U četvrtom poglavlju fokusiramo se na analitičke aspekte. Odabrani izvor podataka će se unijeti u izrađenu bazu podataka te će im se pristupiti iz analitičke perspektive, kako bi se doznale informacije vezane za razne točke interesa iz podatkovnog izvora. Prikazati će se i dodatni alati koji se mogu koristiti uz Cassandru za bolju analizu.

Ključne riječi: NoSQL, SQL, baza podataka, Cassandra, CQL, analiza podataka

1. Uvod

Tradicionalni sustavi upravljanja relacijskim bazama podataka (RDBMS) koji su nekoć vladali borili su se da održe korak s rastućim zahtjevima modernih aplikacija i poslovnih izazova, posebno onih koji uključuju ogromne količine nestrukturiranih ili polustrukturiranih podataka. Kako je količina podataka rasla, mogućnosti relacijskih baza podataka su postajale ograničenije, a rastuće digitalne poslovne potrebe dovele su stručnjake do potrage za novom vrstom baza podataka a njihovo rješenje je urodilo NoSQL bazama podataka.

Za razliku od SQL baza podataka, NoSQL baze podataka nisu se oslanjale na relacijske strukture, već su razine atomičnosti i dosljednosti žrtvovali kako bi prioritet u bazama bio na skalabilnosti i dostupnosti podataka.

Relacijske baze su dan danas najrasprostranjenije i najkorištenije, no NoSQL baze podataka pronašle su svoje mjesto na tržištu, te je za mnoga poslovanja bez njih jednostavno nezamislivo u današnjem dobu koristiti pun potencijal koji podaci posjeduju u Big data svijetu.

Cassandra je zanimljiva vrsta NoSQL baze podataka jer za razliku od drugih NoSQL baza podataka pruža donekle strukturiranu shemu koja omogućava fleksibilnost u radu s podacima velikih razmjera, ali istovremeno održava određenu razinu organizacije i strukture. Ova kombinacija čini Cassandru privlačnom opcijom za razne primjene, osobito u scenarijima gdje se radi o velikim količinama podataka. Ova kombinacija strukture i fleksibilnosti čini Cassandru privlačnom opcijom za sve koji se suočavaju s izazovima rasta i skaliranja svojih aplikacija.

2. Upoznavanje s NoSQL-om

Kako bi se stvorio temelj ovoga rada te lakše razumjeli koncepti, prvo se treba objasniti sam pojam baze podataka. Baza podataka je logički organizirana kolekcija podataka koji su međusobno povezani i prate određena definirana pravila. Podaci u bazi podataka organizirani su prema određenom modelu baze podataka[1]. Neke od tradicionalnih modela baza podataka su Hijerarhijska baza podataka, Mrežna baza podataka, Relacijska baza podataka te objektno orijentirana baza podataka. Iako se pretežito još uvijek koriste ovi modeli, kreću se rađati i neki novi modeli kako bi se prešlo preko razni prepreka. Primjer jednog takvog modela je ne-relacijska baza podataka koja se razvila iz modela relacijske baze podataka [2].

2.1. Što je NoSQL i zašto je bitan?

NoSQL je termin koji se koristi kako bi se referiralo na spremište podataka koje ne slijedi tradicionalni RDBMS model, a to specifično znači da podaci nisu relacijski uređeni te ne koriste SQL kao jezik upita. Jedna od definirajućih karakteristika ovakvih baza podataka je da fokus stavljaju na rješavanje problema skalabilnosti i dostupnosti, koje relacijski model žrtvuje u svrhu održavanja atomičnosti i dosljednosti [3].

Eric Evans je opisao pokret NoSQL-a ovako: “cijela poanta traženja alternative je da ti treba riješiti problem za koji relacijske baze podataka ne odgovaraju” [4].

2.1.1. Razlozi za stvaranje NoSQL-a

Jedna od karakteristika relacijskih baza podataka jest stroga struktura i konzistentnost podataka, što je u većini slučajeva dobra praksa. Međutim, postoje slučajevi kada nije nužno potrebno da podaci prolaze sve provjere konzistentnosti, već je dovoljno da ih se samo pohrani u memoriji [4].

Kada se razvijao relacijski model, nije bilo moguće predvidjeti kako će se jednoga dana raditi s podacima koji su u danas dostupni u enormnim količinama. Skalabilnost je tada bila usmjerena vertikalno, što je značilo da su postojali centralizirani serveri na koje su se svi podaci spremali. Ako je bila potrebna skalacija, nabavljao bi se novi server i podaci bi se prebacivali na novi hardver. Ovakav pristup spremanja podataka je dugo funkcionirao te se koristi i dan danas, ali vertikalno skaliranje više nije praktično zbog količine podataka s kojom se danas radi [5].

Ovaj nedostatak prevladan je uvođenjem NoSQL-ove horizontalne skalabilnosti, kod koje se skalira povećanjem broja jedinica za izvršavanje. Ovom metodom zadaci se ne izvršavaju samo na jednom stroju, već se mogu izvršavati paralelno putem većeg broja resursa, gdje svaki čvor obavlja dio posla te se čvorovi dodaju ovisno o količini posla koji se treba odraditi. Važno je napomenuti da ovo funkcionira samo ako je zadatak moguće podijeliti na više nezavisnih pod-zadataka koji se mogu riješiti u izolaciji od drugih pod-zadataka [6].

Propusnost je još jedna od karakteristika u kojoj NoSQL donosi novitet, budući da je u stanju kroz svoju fleksibilnu shemu procesuirati veći broja podataka u mnogo kraćem vremenu nego što je moguće u strogo strukturiranom relacijskom modelu [4].

Iako je prednost NoSQL-a mogućnost implementacije kao distribuiranog sustava, moguće je implementirati baze za rad na samo jednom serveru. Međutim, tim pristupom se miče naglasak sa dostupnosti i skalabilnosti. Zbog ovakvih odluka, kod dizajna baze podataka važno je točno definirati ravnotežu između karakteristika kao što su skalabilnosti, dostupnosti,

dosljednosti te tolerancije particija. Razni alati naglašavaju razne karakteristike, što se primjećuje u Cassandra bazama podataka koje fokus stavljaju na dostupnost i toleranciju na particije, dok MongoDB preferira dosljednost i toleranciju na particije [7].

2.1.2. Tipovi NoSQL baza podataka

Među NoSQL bazama podataka ističu se 4 tipa:

• Baze podataka parova ključ-vrijednost: Ovo je najjednostavnija vrsta NoSQL baze podataka, koja se sastoji od dvije komponente: ključeva i vrijednosti. Ključ predstavlja jedinstveni identifikator za određeni unos podataka, te se, ukoliko se koristi jedinstveni ključ, ne treba ga ponavljati. Vrijednost je vrsta podataka na koju se referira ključem. Karakteristike su mu jednostavnost, mogućnost rukovanja velikim količinama podataka, skaliranje na velike količine podataka te replikacija podatka korištenjem baze podataka formirane u obliku prstena [8].

Ključ-vrijednost baze podataka nemaju vlastiti jezik upita, no dopušteno je dodavati i uklanjati parove ključ-vrijednost. Vrijednosti se u upitima ne mogu koristiti za pretraživanje, već se kod postavljanja upita, koristi isključivo ključ. Ključ-vrijednost baze podataka koriste se kod upravljanja s velikom količinom podataka koji kontinuirano pristižu u manjim količinama za čitanje i pisanje. Također se koriste kada se spremaju samo ključni detalji o podacima koji se mogu upariti s jednim jedinstvenim ključem, te kad su u pitanju aplikacije usmjerene na rijetke ažuriranja i jednostavne upite [9].

Primjer ovakvih baza podataka su Redis, Memase, i Voldemort [3].

• Baze podataka dokumenata: Ova baza podataka sprema vrijednosti kao dokumente. Dokumenti su polustrukturirani entiteti, najčešće formatirani u JSON ili XML formatu. Svaka baza podataka pomoću pokazivača i haširanja pokazuje polja u dokumentu. Baze podataka dokumenata nemaju fiksiranu shemu, a karakteristike su da se dokument adresira ključem u bazi podataka, te se podaci organiziraju pomoću kolekcija, oznaka, nevidljivih metapodataka te hijerarhija direktorija do kojih se može doći pomoću ključ-vrijednost pretraživanja [8]. Ovakve baze podataka su potrebne jer ključ-vrijednost baze podataka dohvaćaju podatke pomoću ključa, ali imaju restrikcije za dohvaćanje podataka po vrijednosti atributa. Te restrikcije ne postoje u bazama podataka dokumenata koje nude napredne mogućnosti u postavljanju upita [7].

Baze podataka dokumenata koriste se u aplikacijama koje rade s raznovrsnim vrstama podataka koje se svojom strukturom ne mogu uklopiti u strogu tabličnu strukturu tradicionalnih baza podataka. Fleksibilna shema također je idealna za izvođenje CRUD operacija za aplikacije s dinamičkim modelima podacima kojima ne odgovara prilagođavanje podataka tradicionalnoj shemi [10].

Primjer ovakvih baza podataka su MongoDB, CouchD, RavenDB i Terrastore [3].

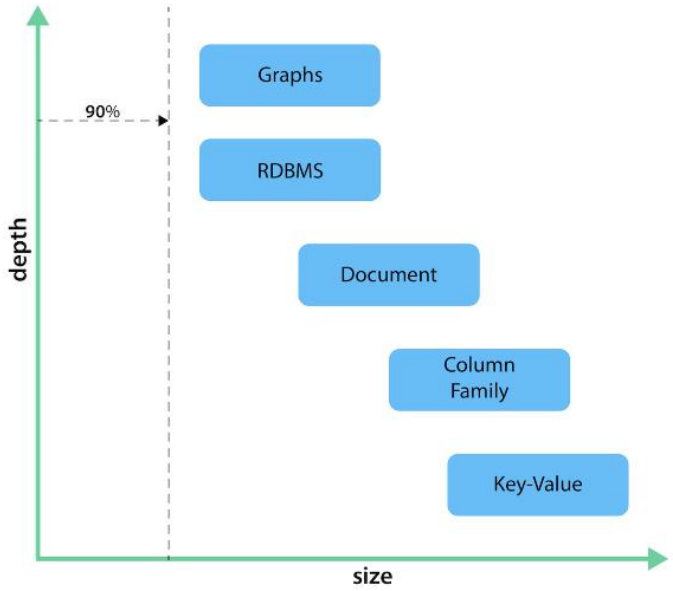
• Obitelj stupaca baze podataka: U ovakvim bazama podataka temeljna jedinica spremišta je stupac. Iako stupci postoje i u relacijskim bazama podataka, u ovom kontekstu se ne radi o tablicama. Baze podataka obitelji stupaca obično su denormalizirane ili strukturirane na način da su svi relevantni podaci o objektu u jednom vrlo širokom retku [7]. Karakteristike ovakvih baza podataka su brže pretrage kroz podatke prilikom postavljanja upita u odnosu na baze podataka temeljenih na redovima, primjena dodjeljivanja mjesta spremanja na svaki stupac te čitanje samo stupaca koji su naglašeni kako bi se ubrzao proces čitanja [8]. Struktura ovih baza podataka sastoji se od keyspace-a u kojem se nalaze tablice. Naziv *obitelj stupaca* je dodijeljen zbog toga što se u tablici nalaze grupirani podaci koji su međusobno relevantni te

im se često pristupa zajedno. Baze podataka tipa obitelj stupaca idealne su za aplikacije skladištenja podataka i poslovne inteligencije koje trebaju provoditi analizu i agregaciju velikih količina podataka [11].

Primjer ovakvih baza podataka su BigTable, HBase, Cassandra i SimpleDB [3].

• Grafičke baze podataka: Umjesto modeliranja podataka korištenjem stupaca i redova, grafička baza podataka koristi čvorove i veze, koji se još nazivaju vrhovi i bridovi. Čvor je objekt koji sadrži identifikator i skup atributa, dok je veza poveznica između čvorova [7].

Drugi modeli baza podataka fokusirani su na obuhvaćanje što veće količine podataka, dok grafički orijentirana baza podataka svoj fokus usmjerava na otkrivanje dubine odnosa među podacima, što je osobina relacijskih baza podataka. Kao što je vidljivo na niže prikazanom grafom, fokus na dubinu veza među podacima je još izraženiji od onog u relacijskim bazama podataka [12].



Slika 1 Dubinska analiza podataka pomoću grafova [12]

2.2. Razvoj NoSQL-a

Od 1970-ih godina, relacijske baze podataka su bile dominantna tehnologija za spremanje strukturiranih podataka. Iako su se rađali novi načini spremanja kao XML spremišta i objektne baze podataka, oni nisu zaista dosegli tržište kao relacijske baze podataka. Umjesto toga su, ili razvili nišu, primjerice u području OLAP-a, ili su postali dio relacijskog modela. Unatoč dominaciji relacijskog modela, 2000-tih godina dolazi do propitkivanja koncepta "one size fits all" kada su pitanju baze podataka. Kako su se pojavljivali zahtjevi koje ni jedno računalo samostalno nije moglo ispunjavati, javila se potreba za prijelazom s pojedinačne

baze podataka u klaster baze podataka. Ovaj pokret sažet je pod terminom NoSQL, koji opisuje sve češću uporabu ne-relacijskih baza podataka među developerima [4].

Carlo Strozzi je 1998. godine upotrijebio izraz NoSQL kako bi opisao relacijsku bazu podataka otvorenog koda kojoj je nedostajalo uobičajeno SQL sučelje. Ime je imalo za cilj opisati pojavu sve većeg broja ne-relacijskih, distribuiranih pohrana podataka koje nisu uvijek nastojale zadovoljiti jamstva ACID svojstava (Atomicity, Consistency, Isolation, Durability), što su važne karakteristike tradicionalnih sustava relacijskih baza podataka [13].

Sam NoSQL na stol donosi svoje karakteristike koje mu dopuštaju upravljanje podacima velikih razmjera, a to su: skalabilnost, trošak, fleksibilnost i dostupnost [7]. U početku razvoja ne-relacijskog modela baza podataka javlja se i pojava novog teorema kojeg donosi Brewer, a taj teorem opisuje distribuirane baze podataka kao baze podataka koje mogu imati same dvije od sljedećih tri zajamčene karakteristike: dosljednost, dostupnost i tolerancija particije, ili skraćeno CAP (Consistency, Availability, Partition Tolerance). Ovo je bilo važno jer NoSQL može bazama podataka pružiti jamstva CP (dosljednost, tolerancija particije) ili AP (dostupnost, tolerancija particije), no ne može jamčiti CA (dosljednost, dostupnost) kombinaciju u praktičnom primjeru ne-relacijske baze podataka [14]. Nakon nekog vremena, NoSQL baze podataka su se više počele orijentirati na dostupnosti i toleranciji particija te se razvio BASE (Basically Available, Soft-state, Eventually consistent) sustav [15].

NoSQL sustavi su počeli biti sve popularniji jer ne samo da mogu upravljati strukturiranim i nestrukturiranim podacima, već to rade i velikom brzinom. Kao rezultat toga, tvrtke poput Facebooka, Twittera, LinkedIna i Googlea počele su koristiti NoSQL sustave. Ove tvrtke rade s golemim količinama nestrukturiranih podataka kako bi otkrile obrasce i stekle poslovne uvide. Big Data je definiran kao službeni izraz 2005. godine. U kasnim 2000-tima, SQL je još uvijek zadržao svoje mjesto kao najkorišteniji model baze podataka, ali se NoSQL pokazao kao poželjna opcija za one kojima je skaliranje bilo prioritet. SQL ni dan danas nije uspio riješiti problem skaliranja, no ni sam NoSQL nije u mogućnosti osigurati razinu sigurnosti transakcija koje nude relacijske baze podataka [16].

2.3. Domene korištenja NoSQL baza podataka

Danas je činjenica da SQL baze podataka same nisu jedinstveno rješenje za sve potrebe korisnika, pogotovo onih koji su svoje poslovanje preselili u oblak. To potiče takve tvrtke na okretanje prema NoSQL bazama podataka kako bi ostvarile željene ciljeve. Situacije u kojima je skalabilnost visoki prioritet, kada se radi s velikim količinama podataka, kada je potrebna konstantna dostupnost i kada je potrebno analizirati u stvarnom vremenu, primjeri su u kojima je NoSQL poželjnija varijanta. U nastavku će biti opisani neki od slučajeva korištenja ne-relacijskih baza podataka koji dokazuju da NoSQL stvarno funkcionira u praksi.

2.3.1. Otkrivanje prijevara i autentifikacija identiteta

Zaštita osjetljivih osobnih podataka i osiguravanje pristupa aplikacijama isključivo klijentima, neki su od glavnih su prioriteti u današnjim aplikacijama. Potrebno je kontinuirano oslanjanje na podatke kako bi se spriječile prijevare, pri kojima neovlaštene osobe dobivaju pristup korisničkim računima klijenata, ili kako bi se potvrdio identitet klijenata. Važno je izgraditi sustav koji u stvarnom vremenu otkriva obrasce i abnormalnosti, čak i prije nego što se prijevara dogodi. Da bi se postigao taj cilj, potrebno je analizirati ogromne količine prošlih i sadašnjih podataka svih vrsta, kao što su korisnički profili, geografski podaci i biometrijski podaci.

U današnjem dobu društvenih mreža, tvrtke sve više moraju paziti na svoj brend. Stoga je važno korisnicima osigurati jednostavno korištenje aplikacije, istovremeno osiguravajući da korisnici ne moraju brinuti o zaštiti svojih korisničkih računa. Usklađivanje jednostavnosti i sigurnosti postaje sve teže zbog potrebe za analizom u stvarnom vremenu, rastućeg broja podataka i sve veće raznolikosti podataka. Upravo stoga se korištenje NoSQL tehnologije za otkrivanje prijevara i autentifikaciju nametnulo kao idealno rješenje.

Primjer jedne takve organizacije je ACI Worldwide, koja omogućuje klijentima plaćanje u stvarnom vremenu. Kao velika organizacija, suočavaju se s iznimno velikim brojem transakcija, zbog čega su analize podataka izuzetno kompleksne. Svi ovi zahtjevi doveli su do spoznaje da bi nastavak rada s SQL tehnologijom zahtijevao particioniranje. Međutim, takav pristup bio bi izrazito skup, vremenski prozor za otkrivanje prijevara morao bi biti kraći, a detekcija bi automatski postala manje precizna.

Podaci potrebni za autentifikaciju i otkrivanje prijevara uključuju demografske podatke, informacije iz CRM sustava, interakcije na web stranici, povijest kupovina i slično. Za takvu raznolikost podataka bilo bi nemoguće razviti shemu koja bi obuhvatila sve te informacije o klijentima. Stoga se pokazalo da je NoSQL fleksibilna shema, koja dopušta integraciju svih vrsta podataka, izvrsno rješenje [17].

2.3.2. Inventar i upravljanje katalogom

Karakteristike poput visoke dostupnosti, isplativosti i horizontalne skalabilnosti čine izvrsnu kombinaciju za e-commerce tvrtke čiji online katalozi neprestano rastu. Ono što je važno ovim tvrtkama jest fleksibilnost u ažuriranju svojih kataloga bez brige o ograničenjima volumena i straha od pada aplikacija tijekom ključnih trenutaka, poput Božića.

Prednost NoSQL-a za njihovo poslovanje prepoznale su mnoge tvrtke koje djeluju na ovom tržištu, a među njima je Macy's, koja se odlučila preći s relacijskih baza podataka na NoSQL baze podataka. Prije ovog prijelaza, Macy's je koristila normalizirane baze podataka koje su ih ograničavale u pogledu mogućnosti skaliranja svog kataloga i online inventara, no uvođenjem NoSQL-a u baze podataka, omogućene su im sljedeće mogućnosti: lakše rukovanje rastom prometa i ogromnim količinama podataka, isplativije skaliranje, brže osvježavanje kataloga, veći opseg online kataloga koji neprestano raste te analiza kataloga i inventara u stvarnom vremenu [17].

2.4. Usporedba NoSQL-a i SQL-a

Cilj ovog poglavlja jest napraviti kratak pregled SQL-a u usporedbi s NoSQL-om. Do sada smo razmotrili razlike kao što su skalabilnost i svojstva, dok ćemo kasnije kroz primjer Cassandre prikazati razliku u arhitekturi. U ovom poglavlju ćemo istražiti pristup analizi podataka koje ove baze podataka pružaju.

Jedna od glavnih prednosti samog SQL-a leži u sposobnosti analize podataka u bazi. NoSQL ima ograničenja u pogledu mogućnosti analiziranja podataka, što proizlazi iz velikih količina podataka koje alati u NoSQL bazama podataka teško mogu interpretirati. Zbog ovih ograničenja u analitičkom aspektu, mnogi analitičari transformiraju, izvlače, poravnavaju i normaliziraju NoSQL podatke u relacijskim bazama podataka kako bi dobili dublji uvid u podatke [18].

Ovakve prednosti proizlaze iz činjenice da se SQL koristi već više od 40 godina te je izuzetno dobro dokumentiran i raširen među korisnicima. SQL svoje korisnike ograničava na rad unutar definiranih struktura, zahtijevajući pažljivu izradu i razumijevanje podataka s kojima rade. S druge strane, NoSQL, zbog svoje fleksibilnosti, nudi korisnicima opuštenij

pristup, gdje greške u dizajnu strukture nisu toliko neopozive. Međutim, to znači da NoSQL baza podataka ne može pružiti istu razinu kvalitete u pogledu uvida u podatke i njihovih međuodnosa. U većini slučajeva, svaka NoSQL baza podataka ima vlastiti način manipulacije podacima, pa su upitni jezici znatno različiti. To rezultira manjkom dokumentacije u usporedbi s SQL bazama podataka koje dijele sličnu sintaksu [19].

Također, još uvijek nedostaje stručnjaka s dubokim razumijevanjem kompleksnog načina na koji NoSQL baze podataka funkcioniraju, pa je prijelaz korisnika tradicionalnih baza podataka na NoSQL izazovan i često spor.

3. Upoznavanje s Apache Cassandrom

Na službenoj web stranici, Apache Cassandra je definirana na sljedeći način: "Apache Cassandra je NoSQL distribuirana baza podataka otvorenog koda, kojoj tisuće tvrtki vjeruju zbog skalabilnosti i visoke dostupnosti, bez ugrožavanja performansi. Linearna skalabilnost i dokazana otpornost na pogreške na uobičajenom hardveru ili infrastrukturi oblaka čine je savršenom platformom za podatke kritične za misiju." [20]

3.1. Povijest Apache Cassandre

Cassandra je nastala iz sve veće potrebe za akomodacijom sve većih količina podataka, koje su se gomilale u web aplikacijama i društvenim medijima. Tvorci Amazonovog Dynama, Avinash Lakshman i Prashant Malik, razvili su početnu ideju Cassandra baze podataka 2009. godine, uzimajući za inspiraciju Dynamo i Bigtable principe.

Prvi važan zadatak koji je ova nova baza podataka imala bio je rješavanje problema inbox pretraživanja na Facebooku te se pokazala kao odlično rješenje postigavši visoku razinu propusnosti i skalabilnosti. Kasnije je zamijenjena HBase-om u zadacima vezanim za inbox pretraživanje na Facebooku, te se sada koristi u Instagramu.

Facebook je 2009. godine objavio dokument (whitepaper) koji detaljno opisuje koncepte i dizajn baze podataka Cassandre te je izvorni kod baze podataka Cassandre učinjen dostupnim javnosti. Cassandra se ubrzo nakon toga počela primjenjivati i u drugim tvrtkama poput Ciscoa, Digg-a i Twittera, a već 2012. godine postojalo je više od 1000 produkcijskih implementacija Cassandra baze podataka, i to u tvrtkama poput eBay-a, Disney-a i Netflix-a [21]. Danas je Cassandra zaslužila svoje mjesto kao dvanaesto najšire prihvaćeno rješenje baza podataka na tržištu [22].

3.2. Ključni koncepti Cassandre

Ranije u radu je spomenut pojam distribuirane baze podataka, a primjer jedne takve baze je sama Cassandra. Da je baza distribuirana, znači da ima sposobnost razdvajanja podataka i zadataka na obradu preko različitih računala uz pomoć čvorova koji odrađuju svaki svoj zadatak paralelno s drugima. Zbog visoke skalabilnosti, Cassandra je jedan od najboljih primjera distribuirane baze, jer kada se dodaju čvorovi, skalira se skoro u potpunosti linearno što se tiče izvedbe, tako da može rukovati s ogromnim količinama podataka bez zastoja ili prekida u radu aplikacija [23].

Visoka dostupnost i tolerancija particije su dva od tri CAP jamstva za baze podataka, na koje se Cassandra oslanja. Ova jamstva govore o tome da će sustav biti dostupan za neki zahtjev u bilo kojem trenutku, tako da ako dođe do greške u jednom dijelu sustava, neki drugi dio sustava preuzima zahtjev i odrađuje ga bez da korisnik primijeti da se greška dogodila [23].

Kod Cassandre postoji sigurnost da čak i ako dođe do pada cijelog podatkovnog centra, neće doći do gubitka podataka [20].

Cassandra također jamči izdržljivost podataka korištenjem replika, putem replikacijskog sustava koji je odgovoran za pohranu i ažuriranje replika. U Cassandri, svi čvorovi su pozicionirani u obliku prstena te nakon pozicioniranja prve replike, dodatne replike se pohranjuju na uzastopnim čvorovima u prstenu u smjeru kazaljke na satu [7]. Ako jedna

replika izgubi zbog nepopravljive greške na čvoru, podaci nisu u potpunosti izgubljeni zbog stvorenih replika [20].

U Cassandri postoji i garancija eventualne dosljednosti, što znači da, iako je fokus na ispunjenju performanci, skalabilnosti i dostupnosti, nakon nekog vremena dolazi i do uspostave jamstva dosljednosti u bazi [24].

3.3. Pogled na arhitekturu i objekte Cassandra baze podataka

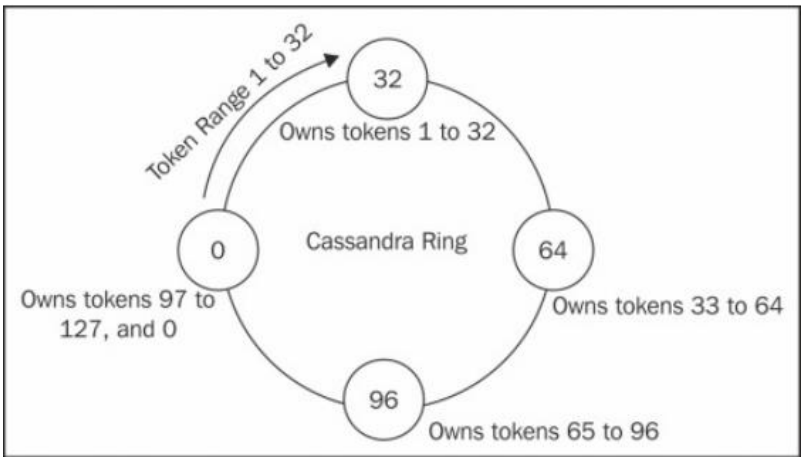
Cassandra je relativno novija opcija u svijetu distribuiranih baza podataka i stoga može koristiti dokazane mehanizme skladištenja podataka, kao što su Bigtable i Amazon Dynamo. Od Bigtablea je moguće primijetiti utjecaj na način prezentacije podataka u tabličnom obliku, iako ne strogo u obliku tablica. Više je slično strukturama poput rječnika, gdje svaki unos podataka sadrži još jedan sortirani rječnik ili kartu. Ovaj model je moćniji od klasičnog ključ-vrijednost spremišta. U Cassandri su preuzeta značajna obilježja od Dynama, kao što su eventualna dosljednost i decentralizacija [23].

Obitelj stupaca se može opisati kao velika tablica slična onoj u Excelu. No, za razliku od proračunske tablice, svaki red je identificiran pomoću ključa retka koji sadrži numeričku vrijednost nazvanu token. Također, svaka ćelija obitelji stupaca, za razliku od proračunske tablice, može imati jedinstveni naziv unutar retka. Zbog visoke tolerancije na particije, retci se distribuiraju kroz sva dostupna računala tako da se dijele u jednake grupe tokena. Obitelji stupaca se nalaze unutar logičkog spremnika koji se naziva keyspace. Keyspace ima sličnu ulogu kao baza podataka u RDBMS-u [23].

Često je očito da svi podaci nisu iste važnosti. Cassandra funkcionira tako da se podaci grupiraju u keyspaces koji služe kao spremnici za podatke aplikacije. U svakoj aplikaciji, klaster ima jedan keyspace [5].

Cassandrin klaster često nazivamo prstenom zbog prstenaste arhitekture. Razlog tome je što su njegovi čvorovi logički raspoređeni poput prstena, a podaci se automatski raspoređuju po svim čvorovima pomoću hash vrijednosti ključeva. Hash vrijednost je broj kojim se numerička vrijednost ključa mapira. Primjerice, string vrijednost 'ABC' je mapirana na broj 101, dok je decimalni broj 25.34 mapiran na broj 257. Ove vrijednosti se generiraju algoritmom na način da jedna vrijednost ključa ima istu hash vrijednost za sve instance u čvoru. Na primjer, sve instance 'ABC' će imati istu hash vrijednost i bit će pohranjene na istom čvoru [25].

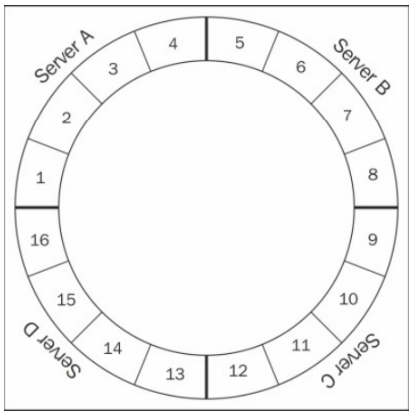
Ako postoji hash algoritam koji radi tako da generira tokene od 0 do 127, a postoje četiri Cassandra stroja za stvaranje klastera, algoritam će svakom od četiri čvora dodijeliti jednak broj tokena. Prvo računalo će biti odgovorno za tokene od jedan do 32, drugo za tokene od 33 do 64, treće za tokene od 65 do 96, a četvrto za tokene od 97 do 127 i 0 [23].



Slika 2 Tokeni u Cassandra prstenu [23]

Dodavanje i oduzimanje čvorova u ranijim verzijama Cassandre odvijalo se putem manualnog resetiranja vlasništva tokena za neki ili sve čvorove. Ovakvo resetiranje naziva se rebalansiranje. Ukoliko je bilo potrebno zamijeniti čvor novim čvorom, podatke je bilo potrebno kopirati na novi stroj replikacijom sa starog stroja. To je za veće baze podataka predstavljalo dugotrajan posao, s obzirom na to da su podaci dolazili s jednog stroja. Zbog ove situacije, u Cassandru su uvedeni virtualni čvorovi [23].

Virtualni čvorovi, poznati kao Vnodes, distribuiraju podatke među čvorovima s finijom granularnošću nego što je to moguće postići korištenjem izračunatih tokena. Vnodes pojednostavljuju mnoge zadatke u Cassandri tako što automatski kalkuliraju tokene i dodjeljuju ih čvorovima [26].



Slika 3 Prikaz 16 vnodes raspoređeni među 4 servera [23].

Svaki čvor odgovoran je za jedan kontinuirani raspon. Ako je replikacijski faktor 2 ili više, podaci se skladište na drugim strojevima. Replikacijski faktor predstavlja broj kopija tablice koja postoji u sustavu, što znači da ako je RF=2, postoje dvije kopije svakog zapisa za tablicu. Pogledajmo primjer u kojem postoji jedan čvor koji prestaje funkcionirati i postaje mrtvi čvor.

Ako postoji prsten s 30 čvorova i čvor s 256 virtualnih čvorova, te ako su čvorovi dobro raspoređeni po klasteru, svaki čvor će imati 8 ili 9 virtualnih čvorova u drugih 29 čvorova. Ti virtualni čvorovi su replike virtualnih čvorova koji su pripadali mrtvom čvoru. Ovakva redistribucija podataka pomaže održavati dosljednost podataka, dostupnost i performanse u klasteru [23].

3.3.1. Mehanizmi rada Cassandre

Proces pisanja

Cassandrin mehanizam pisanja dizajniran je na način koji omogućuje brzo pisanje. Koraci koji se koriste su:

-Prilikom inicijalizacije pisanja u Cassandri, podaci se upisuju u commitlog na disku [25]. Commitlog vrši posao dnevnika koja sadrži samo operacije pisanja redoslijedom kojim su primljene te čuva stalnu evidenciju aktivnosti pisanja i omogućuje toleranciju grešaka u slučaju kvara čvora te garantira izdržljivost podataka [27].

-Podaci se šalju odgovornim čvorovima ovisno o hash vrijednosti. Kao što je ranije objašnjeno, svaki zapis podataka ima primarni ključ koji prolazi kroz hashiranje kako bi se odredilo koji je čvor u klasteru odgovoran za taj podataka.

-Čvorovi pišu podatke u memorijsku tablicu pod nazivom memtable. U memtablelu se zadržavaju podaci do određenog definiranog ograničenja te se nakon što količina podataka pređe to ograničenje, podaci odlaze u SStable. Podaci se smatraju zapisanima kada su spremljeni u memtable i commitlog [5].

-Podaci se nakon toga pišu u SStable u memoriji, koja sadrži konsolidirane podatke svih ažuriranja tablice te se takvi podaci više ne mogu mijenjati[25].

-Iz SStable, podaci se ažuriraju u stvarnu tablicu.

Ažuriranje, brisanje i zbijanje podataka

Cassandra ne dozvoljava umetanje i ažuriranje zapisanih podataka, tako da ako postoji potreba za ažuriranjem podataka Cassandra samo upisuje nove podatke s novim vremenom zapisivanja u memtable, zatim u SStable. Kroz proces zbijanja (compaction), nakon nekog vremena, redundantni podaci nakupljeni u SStabelu se brišu. Proces brisanja odvija se tako da se markerom pod nazivom *tombstone* označuju podaci koje treba obrisati. Kada je sigurno da svi čvorovi imaju označene podatke, ti se podaci brišu [5].

Proces čitanja

Čitanje podataka se u Cassandri odvija u svim čvorovima paralelno, a ako neki od čvorova nije u funkciji, koriste se podaci iz njegove replike [25]. Cassandra kombinira rezultate iz memtablea i SStablesa. Cassandra obrađuje podatke kroz slijedeće korake duž putanje čitanja kako bi odredila gdje su pohranjeni, počevši s podacima u memtableu i završavajući s SSTables:

-Prvi korak je provjerava memtablea koji sadrži privremene podatke u memoriji. Ako su željeni podaci pronađeni vrača se rezultat [28].

-Nakon toga slijedi provjera predmemorije reda(row cache). Ako je provjera predmemorije reda omogućena vrača se cijeli red podataka, bez da koristi tvrdi disk. Ovaj proces obavlja brzo i potpuno te ako su podaci pronađeni vraća rezultat [23].

-Ako je provjera predmemorije reda onemogućena ili podaci nisu pronađeni, slijedi provjera Bloom filtera. Bloom filter je poznata metoda za određivanje pripadnosti nekog elementa određenom skupu [23].

-Ako je provjera uspješna, dalje se provjerava predmemorija ključa particije.

-Ako se particijski ključ otkrije, on se šalje do karte pomaka kompresije(Compression offset map), inače mora proći kroz sažetak particije nakon kojeg pristupa indeksu particije. Karta pomaka pohranjuje pokazivače na točnu lokaciju na disku na kojoj se mogu pronaći podaci o zatraženoj particiji [28].

-Pomoću karte pomaka kompresije se mogu locirati podaci na disku te se dohvaćaju podaci iz SStable [28].

Peer to peer arhitektura

U tradicionalnim bazama podataka, klijent šalje podatke na server, a njihova suradnja je izravna. Server postaje vođa (Leader) baze podataka u kojem se primaju i procesiraju svi podaci. Zbog potrebe za replikacijom podataka, povezan je sa sljedbenicima (Followers) koji primaju dnevnik (log) od vođe. Oni taj dnevnik pohranjuju i ažuriraju u vlastitoj lokalnoj kopiji podataka. Kada klijent zatraži čitanje iz baze podataka, može čitati podatke pohranjene kod vođe ili sljedbenika. No, proces pisanja se provodi na vođi serveru. Prilikom kvara vođe čvora, jedan od sljedbenika privremeno preuzima njegovu ulogu i djeluje kao vođa prema drugim sljedbenicima [29]. U takvim situacijama može doći do kratkotrajnih prekida u radu baze podataka, što nije idealno kad se radi s velikim količinama podataka.

S druge strane, Cassandra koristi Peer-to-peer arhitekturu umjesto Leader-Follower arhitekture. U Peer-to-peer arhitekturi svaki čvor u prstenu može ispunjavati zahtjeve za čitanje i pisanje. Čvorovi postižu to međusobnom komunikacijom putem protokola ogovaranja (Gossip protocol).

Protokol ogovaranja omogućuje svim čvorovima u prstenu da prime ažuriranja bez prisutnosti čvora poput vođe za koordinaciju [30]. Pomoću protokola Cassandri se otkriva gdje su čvorovi pohranjeni u klasteru i dobivaju se informacije o njihovom stanju. Proces ogovaranja je periodičan za svaki čvor. Taj čvor zatraži informacije o stanju drugih triju čvorova. Svaki od tih čvorova dalje traži informacije o stanju od tri druga čvora, što rezultira saznanjem o stanju novih 9 čvorova. Na taj način proces uključuje ukupno 13 čvorova. Ovaj proces se izvršava u nekoliko sekundi, bez obzira koliko čvorova klaster sadrži [25].

3.3.2. Objekti Cassandra baze podataka

Cassandra, kao NoSQL baza podataka, pripada obitelji stupaca. Kako je objašnjeno u prethodnom poglavlju, obitelji stupaca organiziraju podatke u analogne tablice unutar relacijske baze podataka. Redovi se prepoznaju po jedinstvenom ključu u svakoj obitelji stupaca, a osnovni objekti u Cassandri su [6]:

Keyspace: Keyspace je logički spremnik za grupiranje tablica, a na njega se može gledati kao na NoSQL verziju onoga što je baza podataka u relacijskom modelu. Odgovoran je za definiranje replikacije u podatkovnom setu koji vrijedi za sve tablice u definiranom keyspaceu [6].

Tablica: Kao i u relacijskim bazama podataka, tablice su struktura u koje se spremaju podaci, no razlika je u tome što Cassandrine tablice imaju mnogo veći broj specifikacija za stupce kako bi dopustile brže umetanje redaka i čitanje stupaca [31].

Stupac: Stupac je osnovna jedinica podataka u Cassandri te sadrži ime, vrijednost i timestamp. Stupci su organizirani u obitelji stupaca ili tablice. U Cassandri stupci nisu unaprijed definirani (za razliku od tablica), omogućavajući dodavanje stupaca u tablice u bilo kojem trenutku. Za razliku od tradicionalnih relacijskih baza podataka, Cassandra ne zahtijeva da su svi stupci prisutni u svakom pojedinačnom retku [32].

Redak: Redak u Cassandri sadrži kolekciju stupaca identificiranu pomoću jedinstvenog primarnog ključa koji se sastoji od particijskog ključa i opcionalnog ključa za klasteriranje [20].

4. Instalacija Cassandre i uvod u CQL

Iako je Cassandra primarno razvijena za korištenje na Linux operacijskom sustavu, moguće ju je pokrenuti i na Windows operacijskom sustavu. Za praktični dio ovoga rada koristit će se Windows operacijski sustav. Međutim, kako bi i to bilo moguće, bit će potrebno zadovoljiti neke od zahtjeva Cassandre.

Prvi zahtjev je instalacija Jave, budući da se Cassandra ne može pokrenuti bez Jave. Kako bih zadovoljio ovaj preduvjet, preuzeo sam Java Platform Standard Edition 8 — JDK 8u231. Putem uređivanja varijabli okruženja sustava, izradio sam novu varijablu nazvanu JAVA_HOME, a za vrijednost varijable postavio sam putanju preko diska C sve do jre1.8.0_231 verzije u Java direktoriju.

Osim Jave, potrebna je instalacija Python verzije 2.7 koja podržava cqlsh (Cassandra Query Language shell), dok druge verzije to ne podržavaju. Na kraju, potrebno je preuzeti i samu Apache Cassandru verzije 3.11.4. Nakon toga, pomoću programa za arhiviranje datoteka 7-zip, moguće je raspakirati zip datoteku u kojoj se nalazi odabrana verzija Cassandre.

Kako bi se pokrenula Cassandra na Windowsima, otvara se Naredbeni redak (Command prompt) te se upisuje naredba *cassandra*:

```
C:\Users\David>cd c:\apache-cassandra-3.11.4\bin

c:\apache-cassandra-3.11.4\bin>cassandra
```

Slika 4 Pokretanje Cassandre

U Naredbenom retku će se pojaviti i sljedeće upozorenje:

```
WARNING! Powershell script execution unavailable.
   Please use 'powershell Set-ExecutionPolicy Unrestricted'
   on this user-account to run cassandra with fully featured
   functionality on this platform.
```

Slika 5.Potreba za dodatnim instalacijama

Kako bi se Cassandra koristila sa svim svojim funkcionalnostima, otvara se PowerShell s administratorskim pravima, te se upisuje naredba *Set-ExecutionPolicy Unrestricted*. Zadnji korak prije korištenja Cql shell (cqlsh) je instalacija *pyreadline* putem Naredbenog retka kao administratora.

```
C:\WINDOWS\system32>pip install  pyreadline
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 w
n't be maintained after that date. A future version of pip will drop support for Python 2.7. More details about Python
 support in pip, can be found at https://pip.pypa.io/en/latest/development/release-process/#python-2-support
```

Slika 6. Instalacija pyreadline paketa

4.1. Stvaranje okruženja u Cassandri

Podaci se spremaju u tablice na različite načine kako bi se efikasnije koristila memorija. Na sljedećem primjeru pokazat ću na koji način se podaci u Cassandri mogu efikasno spremiti u tablice. Na sljedećoj slici su prikazane tablice kakve se očekuju prilikom korištenja relacijskog modela:

Zaposlenik

ID	Ime	Prezime	ID_auta	Plaća
1	Matej	Ujević	1	900,00 €
2	Marko	Horvat	2	950,00 €
3	Luka	Stipčević	3	1.000,00 €
4	Ante	Lončar	3	1.200,00 €
5	Josip	Brkić	4	800,00 €
6	Ana	Martinović	5	1.100,00 €
7	Matea	Šarić	1	1.000,00 €

Automobil

ID_auta	Marka	Model	Cijena
1	Ford	Focus	20.000,00 €
2	Toyota	Yaris	15.000,00 €
3	BMW	5-Series	50.000,00 €
4	BMW	3-Series	30.000,00 €
5	Ford	Fiesta	13.000,00 €
6	Fiat	Punto	22.000,00 €

Slika 7. Tablice u relacijskom stilu

Prva tablica prikazuje zaposlenike u nekoj organizaciji, dok druga tablica prikazuje automobile. U prvoj tablici nalazi se atribut *ID_auta,* koji je njen sekundarni ključ, te primarni ključ u tablici automobila. Ova struktura je idealna za relacijske baze podataka, jer se pomoću JOIN klauzule vrlo jednostavno mogu odgovoriti upiti pomoću SQL-a. Unutar Cassandre takvih upita nema, jer se za razliku od relacijskog modela koriste tablice i podaci s različitih računala. Time što Cassandra ne koristi JOIN klauzule, ona dobiva na brzini prilikom čitanja. Zato Cassandra koristi pristup upitu nazvan *Query first approach*, što znači da se tablice dizajniraju za specifičan upit. Zbog toga može doći do ponavljanja nekih podataka u različitim tablicama kako bi se zadovoljili različiti upiti. No, ovaj način je idealan za rad u Cassandri.

Upit pristupu tablice Zaposlenik i Automobil bi izgledale više kao na slici:

Zaposlenik po marki

Marka	ID_zaposlenika	Ime	Prezime	Plaća
BMW	3	Luka	Stipčević	1.000,00 €
BMW	4	Ante	Lončar	1.200,00 €
BMW	5	Josip	Brkić	800,00 €
Ford	1	Matej	Ujević	900,00 €
Ford	7	Matea	Šarić	1.000,00 €
Ford	6	Ana	Martinović	1.100,00 €
Toyota	2	Marko	Horvat	950,00 €

Slika 8. Primjer Tablice za određen upit

Ovakva tablica omogućava brzu pretragu svih marki automobila koje zaposlenici koriste putem jedne tablice.

Ova tablica je implementirana u Cassandri u sljedećem obliku:

Zaposlenik po marki				
BMW	{Id : 3}	{Ime : Luka}	{Prezime:Stipčevič}	{Plaća : 1.000.00 € }
BMW	{Id : 4}	{Ime : Ante}	{Prezime:Lončar}	{Plaća : 1.200.00 € }
BMW	{Id : 5}	{Ime : Josip}	{Prezime: Brkić}	{Plaća : 800.00 € }
Ford	{Id : 1}	{Ime : Matej}	{Prezime: Ujević}	{Plaća : 900.00 € }
Ford	{Id : 7}	{Ime : Matea}	{Prezime: Šarić}	{Plaća : 1.000.00 € }
Ford	{Id : 6}	{Ime : Ana}	{Prezime: Martinović}	{Plaća : 1.100.00 €}
Toyota	{Id : 2}	{Ime : Marko}	{Prezime: Horvat}	{Plaća : 950.00 € }

Slika 9. Tablica zapisana u Cassandri

Ovdje se može primijetiti obitelj stupaca modela o kojoj se ranije govorilo, pri čemu svaki stupac ima svoj ključ i vrijednost za svaki red. Particijski ključ ove tablice je marka automobila, a svi redovi tablice koji su marke BMW nalazit će se na istom čvoru u klasteru, čime će se do tih podataka pristupati velikom brzinom. Svaka instanca u kojoj je marka automobila BMW za zaposlenika će, putem hash funkcije, dobiti jednaku numeričku vrijednost zvanu token. U ovoj tablici evidentna je razlika između particijskog ključa Cassandre i tradicionalnog relacijskog primarnog ključa koji bi izgledao sličnije kao Id stupac ove tablice. Ovu tablicu moguće je prikazati na ekonomičniji način.

Zaposlenik po marki				
BMW*3	{Id : 3}	{Ime : Luka}	{Prezime:Stipčevič}	
	{Id : 4}		{Prezime:Lončar}	{Plaća : 1.200.00 € }
	{Id : 5}	{Ime : Josip}		{Plaća : 800.00 € }
Ford*3	{Id : 1}	{Ime : Matej}	{Prezime: Ujević}	{Plaća : 900.00 € }
	{Id : 7}		{Prezime: Šarić}	{Plaća : 1.000.00 € }
	{Id : 6}	{Ime : Ana}	{Prezime: Martinović}	{Plaća : 1.100.00 €}
Toyota	{Id : 2}	{Ime : Marko}	{Prezime: Horvat}	

Slika 10. Ekonomičnost Cassandra tablica

Ovakva tablica prikazuje mogućnost čitanja podataka, bez potrebe da se ponovno zapisuju u sljedeći redak kada dolazi do njihovog ponavljanja. Također, u tablici se može primijetiti da neki podaci nisu zapisani. To bi bilo nemoguće u relacijskim bazama podataka, no u Cassandri je ovakva tablica prihvatljiva.

4.2. CQL

Rad u Cassandri započinjem pokretanjem Cassandra shell-a pomoću naredbe *cqlsh* u naredbenom retku. Kao što je već ranije objašnjeno, prva stvar koju treba izraditi jest keyspace, što je analogno bazi podataka u relacijskim sustavima.

```
C:\Users\David>cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> DESCRIBE KEYSPACES

system_traces  system_schema  system_auth  system  system_distributed

cqlsh>  CREATE KEYSPACE keyspace_1 WITH replication ={'class': 'SimpleStrategy', 'replication_factor': '1'} AND durable_writes= 'true';
cqlsh> DESCRIBE KEYSPACES

keyspace_1     system_auth  system_distributed
system_schema  system       system_traces

cqlsh> USE keyspace_1;
cqlsh:keyspace_1>
```

Slika 11. Stvaranje Keyspacea

Naredba se inicijalizira pomoću dijela naredbe CREATE KEYSPACE keyspace_1, gdje *keyspace_1* predstavlja naziv keyspace-a. Drugi dio naredbe definira replikacijsku strategiju i replikacijski faktor za keyspace. U ovom primjeru, za klasu replikacije koristi se 'SimpleStrategy', što je uobičajeno za slučajeve kada se radi na jednom računalu. Replikacijski faktor postavljen je na '1', što znači da će se stvarati samo jedna kopija podataka u klasteru. Opcija "Durable writes" postavljena je na *'true'* kako bi podaci upisani u keyspace-u bili trajni.

U usporedbi s relacijskim bazama podataka, gdje se izrada baze podataka ostvaruje pokretanjem naredbe "CREATE DATABASE baza_podataka_1;", kod izrade keyspace-a koriste se drugačiji koraci.

Nakon što je keyspace izrađen, može se vidjeti na listi naredbom DESCRIBE KEYSPACES, a zatim se naredbom "USE keyspace_1;" prelazi na rad u stvorenom keyspaceu. Svrha keyspace-a je da sadrži jednu ili više tablica te definira replikaciju za podatke tih tablica.

```
cqlsh:keyspace_1> CREATE TABLE zaposlenik_po_id (id int PRIMARY KEY, ime text, prezime text, pozicija text);
cqlsh:keyspace_1> DESCRIBE TABLES

zaposlenik_po_id

cqlsh:keyspace_1>
```

Slika 12. Stvaranje tablice u CQL-u

Tablice se stvaraju unutar keyspace-a pomoću naredbe CREATE TABLE, koja se u ovom slučaju naziva *zaposlenik_id.* Unutar zagrada definiraju se nazivi atributa, primarni ključ i tipovi podataka. Razlika između CQL-a i SQL-a u izradi ovakve tablice jest to što bi u SQL-u bilo potrebno definirati vrstu podataka *text* kao *varchar* te odrediti maksimalnu duljinu, na primjer, "ime varchar(255)".

Ukoliko se želi izbrisati tablica, koristi se naredba DROP TABLE *zaposlenik_id*; te se provjerom DESCRIBE TABLES otvara lista svih tablica u keyspaceu. U sljedećem primjeru tablice jasno je definiran particijski ključ u atributima *marka* i *id*:

```
cqlsh:keyspace_1> CREATE TABLE zaposlenik_po_marki (marka text, id int, model text, PRIMARY KEY(marka, id));
cqlsh:keyspace_1> DESCRIBE TABLES
```

Slika 13. Tablica zaposlenik_po_marki

Upisivanjem naredbe “DESCRIBE TABLE zaposlenik_po_marki;“ dobiva se lista dodatnih informacija o tablici. Particijski ključ može se definirati i pomoću više stupaca:

```
cqlsh:keyspace_1> CREATE TABLE zaposlenik_po_marki_i_modelu(marka text, model text, id int, ime text, PRIMARY KEY((marka, model),id));
```

Slika 14. Više stupaca u particijskom ključu

U definiranju tablice, particijski ključ čine kombinacija stupaca marke i modela automobila. To znači da će svi podaci s istom kombinacijom marke i modela biti grupirani u čvorovima, ubrzavajući tako pretragu podataka. Što se tiče stupca *id*, on je dio primarnog ključa, a iako ne ulazi u sastav particijskog ključa, koristi se kao stupac za klasteriranje. U slučaju da je potrebno dodati stupac u već stvorenu tablicu, to se može izvesti na sljedeći način:

```
cqlsh:keyspace_1> ALTER TABLE zaposlenik_po_marki ADD prezime set<text>;
cqlsh:keyspace_1> SELECT * FROM zaposlenik_po_marki;

 marka | id | model    | prezime
-------+----+----------+---------
   BMW |  2 | Series-5 |    null
   BMW |  3 | Series-3 |    null
  Ford |  1 |   Fiesta |    null
```

Slika 15. Unos novog stupca u tablicu

Ovom naredbom stvoren je novi stupac, a pregledom podataka u tablici vidljivo je da su vrijednosti za ranije unesene retke u stupcu *prezime* jednake null.

Unos podataka u Cassandri odvija se na sličan način kao i u SQL-u:

```
cqlsh:keyspace_1> INSERT INTO zaposlenik_po_id (id, ime, pozicija) VALUES (1,'Marko', 'Programer');
cqlsh:keyspace_1> SELECT * FROM zaposlenik_po_id;

 id | ime   | pozicija  | prezime
----+-------+-----------+---------
  1 | Marko | Programer |    null
```

Slika 16. Unos podataka u CQL-u

Iako je *prezime* definirano kao stupac u tablici, moguće je unijeti "Marka" kao zaposlenika bez unesenog prezimena, što bi u SQL-u bilo nemoguće bez definiranja tog stupca kao stupca koji dopušta null vrijednosti pri izradi tablice. Nije dopušteno izostavljati stupce ako pripadaju

particijskom ključu definiranom tijekom izrade tablice. Ako kasnije želimo dodati prezime u tablicu *zaposlenik_id*, to možemo izvesti pomoću UPDATE naredbe koja glasi:

```
cqlsh:keyspace_1> UPDATE zaposlenik_id SET prezime='Ivanušević' WHERE id=1;
cqlsh:keyspace_1> SELECT * FROM zaposlenik_id;

 id | ime   | pozicija   | prezime
----+-------+------------+------------
  1 | Marko | Računovođa | Ivanušević
```

Slika 17. UPDATE tablice u CQL-u

Podatak iz tablice jednako tako možemo postaviti natrag na “null“ time što umjesto SET prezime="Ivanušević“ upišemo SET prezime={}.

Ako želimo pretražiti specifične podatke iz tablice koristimo WHERE klauzula:

```
cqlsh:keyspace_1> SELECT * FROM zaposlenik_id WHERE id=1;

 id | ime   | pozicija   | prezime
----+-------+------------+------------
  1 | Marko | Računovođa | Ivanušević
```

Slika 18. WHERE

Ova pretraga radi jer prikazuje zaposlenika s id=1, no ako se želi pretraživati po poziciji zaposlenika nastupa sljedeća greška:

```
cqlsh:keyspace_1> SELECT * FROM zaposlenik_id WHERE pozicija='Programer';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query
espite the performance unpredictability, use ALLOW FILTERING"
```

Slika 19. Use ALLOW FILTERING

Ovaj upit bio bi valjan u SQL-u, no zbog toga što stupac *pozicija* nije dio primarnog ključa, pretraga podataka ove tablice putem tog stupca nije moguća. U CQL-u je moguće filtrirati po stupcima koji nisu uključeni u primarni ključ, no takav pristup nije optimalan jer zahtijeva korištenje naredbe "ALLOW FILTERING" na kraju upita. Ovaj način pretrage nije efikasan jer zahtijeva pretraživanje cijele tablice, što značajno usporava pretragu i troši velike resurse, posebno ako se radi s velikim tablicama koje sadrže obilje podataka.

Kao primjer, razmotrimo sljedeći upit gdje je marka automobila particijski ključ tablice. U tom primjeru možemo vidjeti uspješno izvršenu pretragu:

```
cqlsh:keyspace_1> SELECT * FROM zaposlenik_po_marki WHERE marka='BMW' ORDER BY id;

 marka | id | model
-------+----+----------
   BMW |  2 | Series-5
   BMW |  3 | Series-3
```

Slika 20. ORDER BY

ORDER BY je moguć samo za stupce koji su dio primarnog ključa. To znači da bi, ako zamijenimo stupac *id* sa stupcem *model*, rezultiralo pogreškom. S druge strane, ukoliko unesemo novog zaposlenika koristeći isti *id* koji već postoji za drugog zaposlenika, Cassandra će preko postojećeg zapisa unijeti novi *id*. Ovakva vrsta unosa podataka u tablicu nije moguća u SQL bazama podataka.

```
cqlsh:keyspace_1> SELECT * FROM zaposlenik_po_marki_i_modelu WHERE marka='BMW';
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this que
ou want to execute this query despite the performance unpredictability, use ALLOW FILTERING"
```

Slika 21. Dvostruki particijski ključ

Ako pogledamo tablicu *zaposlenik_po_marki_i_modelu*, u kojoj je particijski ključ definiran kao spoj stupaca *marka* i *model*, upiti u kojima se koristi filtriranje pomoću jednog od tih stupaca rezultiraju greškom. Primjećujemo da svaki upit koji uključuje filtriranje mora sadržavati sve dijelove particijskog ključa kako bi rezultat bio ispravan. Ako želimo pristupiti podacima iz tablice pomoću nečega što nije dio primarnog ključa, koristimo sekundarne indekse.

```
cqlsh:keyspace_1> CREATE INDEX ON zaposlenik_id (prezime);
cqlsh:keyspace_1> SELECT * FROM zaposlenik_id WHERE prezime='Petrović';

 id | ime   | pozicija  | prezime
----+-------+-----------+----------
  2 | Marin | Programer | Petrović
```

Slika 22. Sekundarni Indeksi

Iako su sekundarni indeksi bolja opcija od korištenja naredbe ALLOW FILTERING zbog brzine pretraživanja kroz podatke, oni stvaraju povećani zahtjev za pohranom podataka. Preporučuje se indeksiranje samo onih stupaca koji se često koriste za pretraživanje.

U relacijskim bazama podataka uobičajen je pristup stvaranju primarnog ključa koji je jedinstveni auto-inkrementalni integer, osiguravajući time jedinstvenost zapisa u tablici. U Cassandra bazama podataka postoji sličan koncept koji generira jedinstvenu identifikaciju za svaki redak u tablici, a naziva se UUID (Universally Unique Identifier). UUID je 128-bitna vrijednost prikladna za distribuirane baze podataka. Primjer jedne tablice u kojoj je primarni ključ zadan kao UUID vrijednost je sljedeći:

```
cqlsh:keyspace_1> INSERT INTO zivotinje_uuid(id, ime, vrsta, lokacija) VALUES(uuid(), 'Simba', 'Lav', 'Objekt 7');
cqlsh:keyspace_1> INSERT INTO zivotinje_uuid(id, ime, vrsta, lokacija) VALUES(uuid(), 'Kory', 'Kornjača', 'Objekt 4')
cqlsh:keyspace_1> INSERT INTO zivotinje_uuid(id, ime, vrsta, lokacija) VALUES(uuid(), 'Biggy', 'Krokodil', 'Objekt 5'
cqlsh:keyspace_1> SELECT * FROM zivotinje_uuid;

 id                                   | ime   | lokacija | vrsta
--------------------------------------+-------+----------+----------
 3b9e489c-96f1-4282-af2a-ed493e7d58d1 |  Kory | Objekt 4 | Kornjača
 1112553d-b171-4fcd-b931-7942b9021139 | Biggy | Objekt 5 | Krokodil
 72b5d407-572d-44c9-bafd-efdb5311c199 | Simba | Objekt 7 |      Lav
```

Slika 23. UUID

Sličnije UUID vrijednosti u tablici mogu se dobiti korištenjem primarnog ključa Time UUID, čija je 128-bitna vrijednost određena prema specifičnom vremenskom okviru u kojem je redak upisan u tablicu.

```
cqlsh:keyspace_1> CREATE TABLE biljke_tuuid (id timeuuid PRIMARY KEY, vrsta text, lokacija text);
cqlsh:keyspace_1> INSERT INTO biljke_tuuid(id, vrsta, lokacija) VALUES(now(), 'Pampas', 'Vrt 15');
cqlsh:keyspace_1> INSERT INTO biljke_tuuid(id, vrsta, lokacija) VALUES(now(), 'Palma', 'Vrt 41');
cqlsh:keyspace_1> SELECT * FROM biljke_tuuid;

 id                                   | lokacija | vrsta
--------------------------------------+----------+--------
 1fc26f00-32f0-11ee-b680-c11d06d6aa82 |   Vrt 15 | Pampas
 2d303d20-32f0-11ee-b680-c11d06d6aa82 |   Vrt 41 |  Palma
```

Slika 24. TUUID

5. Podatkovna analiza

5.1. Unos podataka iz CSV datoteke u Cassanddra tablicu

Kako bi se radilo s podacima u Cassandri, nije uvijek potrebno samostalno ručno ispuniti cijelu bazu podataka. Postoji mogućnost unosa podataka iz prethodno ispunjenog izvora podataka. Za potrebe ovog rada, odabrao sam CSV datoteku koju sam preuzeo s Kettle web stranice, te sam je uredio u Excelu kako bi sadržavala samo stupce relevantne za analizu podataka koja slijedi.Kako bi radio s čistom bazom podataka izradio sam novi keyspace pod nazivom *emp_keyspace*:

```
CREATE KEYSPACE  emp_keyspace
WITH replication = {
   'class': 'SimpleStrategy',
   'replication_factor': 3
};
```

Za uspješan prijenos podataka iz CSV datoteke u Cassandru, potrebno je stvoriti tablicu iste strukture kao i u CSV datoteci. Svaki stupac u tablici mora imati isti naziv kao stupac u datoteci, a stupci moraju biti poredani u istom redoslijedu kao u datoteci.

```
CREATE TABLE emp_keyspace.employee_id (
    employee_id int,
    department text,
    job_role text,
    age int,
    attrition text,
    business_travel text,
    education int,
    education_field text,
    gender text,
    job_involvement int,
    job_level int,
    job_satisfaction int,
    monthly_income int,
    num_companies_worked int,
    over_time text,
    performance_rating int,
    total_working_years int,
    years_at_company int,
    years_in_current_role int,
    PRIMARY KEY (employee_id, department, job_role)
```

Slika 25. Tablica employee_id

Podatkovni set prikazuje zaposlenike u organizaciji i sadrži podatke od interesa HR odjelu unutar iste. Stupci tablice su sljedeći:

1. employee_id: numerička vrijednost koja jedinstveno identificira zaposlenika.
2. department: tekstualna vrijednost koja svrstava zaposlenika u odjel.

3. job_role: tekstualna vrijednost koja opisuje poziciju na kojoj zaposlenik radi.

4. attrition: tekstualna vrijednost koja definira radni odnos zaposlenika.

5. business_travel: tekstualna vrijednost koja definira koliko često zaposlenik putuje.

6. education: numerička vrijednost koja prikazuje razinu obrazovanja.

7. education_field: tekstualna vrijednost koja opisuje područje obrazovanja zaposlenika.

8. gender: tekstualna vrijednost koja opisuje spol zaposlenika.

9. job_involvement: numerička vrijednost koja prikazuje koliko je zaposlenik uključen u poslovne aktivnosti.

10. job_level: numerička vrijednost koja opisuje poziciju zaposlenika unutar hijerarhije.

11. job_satisfaction: numerička vrijednost koja prikazuje zadovoljstvo poslovnim životom u organizaciji.

12. monthly_income: numerička vrijednost koja predstavlja mjesečnu plaću zaposlenika.

13. num_companies_worked: numerička vrijednost koja pokazuje koliko je organizacija zaposlenik prethodno radio.

14. over_time: tekstualna vrijednost koja označava radi li zaposlenik prekovremeno.

15. performance_rating: numerička vrijednost koja prikazuje ocjenu uspješnosti zaposlenika na poziciji.

16. total_working_years: numerička vrijednost koja pokazuje ukupan broj godina radnog iskustva.

17. years_at_company: numerička vrijednost koja prikazuje koliko godina zaposlenik radi u organizaciji.

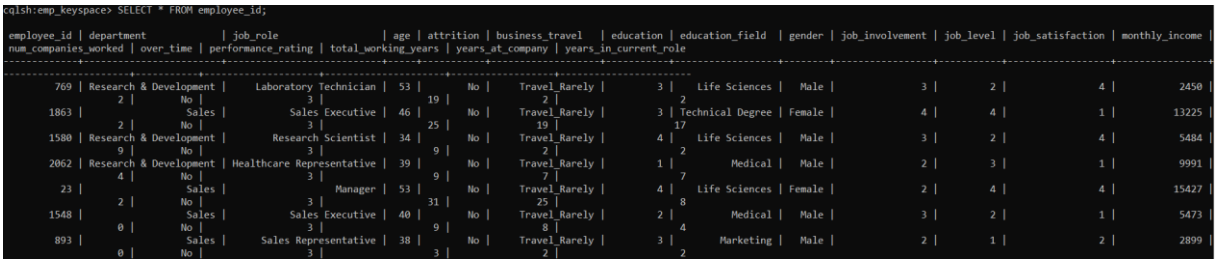
18. years_in_current_role: numerička vrijednost koja pokazuje koliko godina zaposlenik obnaša trenutnu poziciju.

19. PRIMARY KEY (employee_id, department, job_role): primarni ključ sastavljen od stupca employee_id kao particijskog ključa, te stupaca department i job_role kao klasterirajućih stupaca.

Nakon što je tablica stvorena sljedećom naredbom unose se podaci:

COPY employee_id FROM 'C:\Users\David\Desktop\employee_data.csv' WITH DELIMITER=';' AND HEADER=TRUE;

Potrebno je definirati tablicu u koju će se unijeti podaci. Nakon toga, potrebno je definirati putanju do CSV datoteke te odrediti način razdvajanja podataka u datoteci. Budući da se radi o CSV datoteci, koristi se točka zarez ';' kao separator. Na kraju, upotrebom naredbe HEADER=TRUE označava se prisutnost zaglavlja u CSV datoteci.



```
cqlsh:emp_keyspace> SELECT * FROM employee_id;

 employee_id | department            | job_role                | age | attrition | business_travel   | education | education_field  | gender | job_involvement | job_level | job_satisfaction | monthly_income |
 num_companies_worked | over_time | performance_rating | total_working_years | years_at_company | years_in_current_role
----------------------+-----------+--------------------+---------------------+------------------+-----------+------------------+--------+-----------------+-----------+------------------+----------------+
         769 | Research & Development |     Laboratory Technician |  53 |       No |     Travel_Rarely |         3 |    Life Sciences |   Male |               3 |         2 |                4 |           2450 |
                   2 |        No |                  3 |                  19 |                2 |                 2
        1863 |                  Sales |          Sales Executive |  46 |       No |     Travel_Rarely |         3 | Technical Degree | Female |               4 |         4 |                1 |          13225 |
                   2 |        No |                  3 |                  25 |               19 |                17
        1580 | Research & Development |       Research Scientist |  34 |       No |     Travel_Rarely |         4 |    Life Sciences |   Male |               3 |         2 |                4 |           5484 |
                   9 |        No |                  3 |                   9 |                2 |                 2
        2062 | Research & Development | Healthcare Representative |  39 |       No |     Travel_Rarely |         1 |          Medical |   Male |               2 |         3 |                1 |           9991 |
                   4 |        No |                  3 |                   9 |                7 |                 7
          23 |                  Sales |                  Manager |  53 |       No |     Travel_Rarely |         4 |    Life Sciences | Female |               2 |         4 |                4 |          15427 |
                   2 |        No |                  3 |                  31 |               25 |                 8
        1548 |                  Sales |          Sales Executive |  40 |       No |     Travel_Rarely |         2 |          Medical |   Male |               3 |         2 |                1 |           5473 |
                   0 |        No |                  3 |                   9 |                8 |                 4
         893 |                  Sales |     Sales Representative |  38 |       No |     Travel_Rarely |         3 |        Marketing |   Male |               2 |         1 |                2 |           2899 |
                   0 |        No |                  3 |                   3 |                2 |                 2
```

Slika 26. Ispis tablice

Podaci su sada u tablici, no zbog količine stupaca, ispis je neuredan. Kako bi se lakše pregledali podaci, mogu se stvoriti materijalizirani pogledi koji mogu pomoći organizirati podatke za određene uzorke upita.

Materijalizirani pogled je pogled stvoren iz glavne tablice i sadrži novi primarni ključ. Također, moguće je odabrati koje će stupce iz glavne tablice sadržavati. Kod Cassandre, tablice se stvaraju s ciljem prilagodbe za određeni upit. Ako tablica u primarnom ključu ne sadrži stupac koji je potreban za taj upit, stvara se nova tablica.

Kada su u pitanju podaci niske kardinalnosti, prikladno je koristiti sekundarne indekse. No, kada se suočavamo s visokom kardinalnošću podataka, tada se koriste materijalizirani pogledi. Za stvaranje materijaliziranog pogleda, prilikom njegova definiranja, potrebno je uključiti sve stupce koji se nalaze u primarnom ključu glavne tablice. Važno je napomenuti da je u primarni ključ materijaliziranog pogleda moguće dodati samo jedan novi stupac [33].

5.2. Analiza podatkovnog seta

U ovom djelu rada pogledat ćemo podatkovni set s više različitih kutova kako bi došli do nekih zaključaka vezano za zaposlenike u ovoj organizaciji. To ću učiniti stvaranjem materijaliziranih pogleda za razne točke interesa u podatkovnom setu.

5.2.1. Prva tablica – Analiza odjela i pozicija u organizaciji

Prvi pogled

Prvi takav pogled je *emp_keyspace.employee_by_department_role,* koji prikazuje stupce: *department, job_role, job_satisfaction, employee_id* i *age.* Kao što znamo, *employee_id, department* i *job_role* tvore primarni ključ glavne tablice, a njima je u ovom pogledu dodan *job_satisfaction*. U ovom djelu analize, zanima nas informacije koje možemo izvući iz podataka vezanih za odjele i pozicije u organizaciji, što je vidljivo iz samog izbora primarnog ključa s kojim je vidljiva potreba za filtriranjem kroz relevantne stupce u tablici.

```
cqlsh:emp_keyspace> CREATE MATERIALIZED VIEW emp_keyspace.employee_by_department_role AS
                ... SELECT employee_id, department, job_role, age, job_satisfaction
                ... FROM emp_keyspace.employee_id
                ... WHERE department IS NOT NULL  AND job_role IS NOT NULL AND job_satisfaction IS NOT NULL  AND employee_id IS NOT NULL
                ... PRIMARY KEY (department, job_role, job_satisfaction, employee_id);
```

Slika 27. Materialized view employee_by_department_role

Podaci u ovom pogledu izgledaju kao na sljedećoj slici:

```
cqlsh:emp_keyspace> SELECT * FROM emp_keyspace.employee_by_department_role;

 department      | job_role        | job_satisfaction | employee_id | age
-----------------+-----------------+------------------+-------------+-----
 Human Resources | Human Resources |                1 |         133 |  37
 Human Resources | Human Resources |                1 |         424 |  31
 Human Resources | Human Resources |                1 |         590 |  34
 Human Resources | Human Resources |                1 |         878 |  36
 Human Resources | Human Resources |                1 |        1098 |  44
 Human Resources | Human Resources |                1 |        1231 |  42
```

Slika 28. Prikaz podataka u employee_by_department_role pogledu

Sada kada imamo podatke prikazane na pregledniji način i fokusiran na specifične stupce koji su nam od interesa, možemo proći kroz neke upite:

Prosječna starost zaposlenika u odjelu za istraživanje i razvoj, po poziciji, predstavlja zanimljivu analitičku točku. Ukoliko želimo saznati koliko godina prosječno ima zaposlenik u tom odjelu, možemo postaviti sljedeći upit:

```
SELECT department, job_role, AVG(age) AS avg_age FROM
emp_keyspace.employee_by_department_role WHERE department = 'Research &
Development' GROUP BY department, job_role;
```

Kako je stupac *department* prvi u nizu primarnog ključa, potrebno je po njemu započeti filtriranje, te nije moguće filtrirati po *job_role* stupcu dok nije definiran uvijet za *department* stupac. Ovo je jedna od razlika od SQL upita gdje, nema ograničenja što se tiče redoslijeda postavljanja uvjeta u upitu. Također, ako se filtrira pomoću više stupaca, oni se razdvajaju pomoću ključne riječi AND.

```
 department              | job_role                    | avg_age
-------------------------+-----------------------------+---------
 Research & Development | Healthcare Representative |      39
 Research & Development |     Laboratory Technician |      34
 Research & Development |                   Manager |      45
 Research & Development |    Manufacturing Director |      38
 Research & Development |         Research Director |      44
 Research & Development |        Research Scientist |      34
```

Slika 29. Research & Development

Zanima nas zadovoljstvo svih menadžera unutar odjela *Sales*. Osim toga, fokus je stavljen samo na nezadovoljne menadžere. Nezadovoljstvo je u stupcu prikazano vrijednostima 1 ili 2, dok vrijednosti 3 i 4 prikazuju zadovoljne zaposlenike. Prikaz svih zaposlenika čije je nezadovoljstvo 1 ili 2:

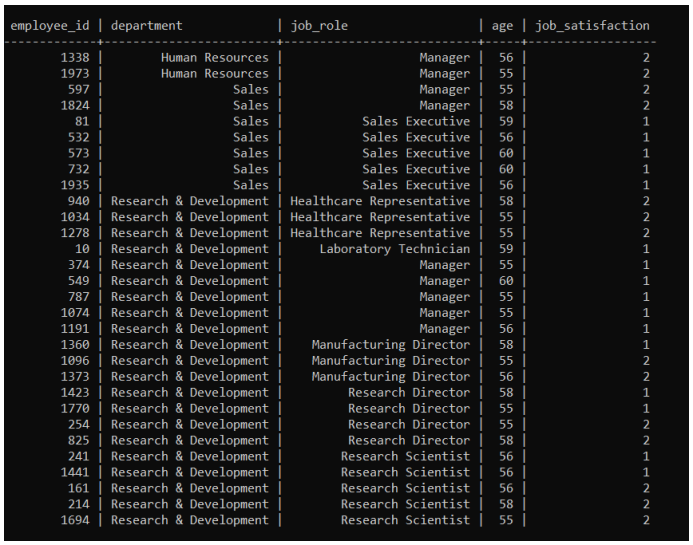
```
SELECT employee_id, age, job_satisfaction FROM
emp_keyspace.employee_by_department_role WHERE department = 'Sales' AND
job_role='Manager' AND job_satisfaction < 3;
```

employee_id	age	job_satisfaction
38	46	1
625	41	1
776	52	1
1204	46	1
1602	46	1
363	46	2
410	41	2
597	55	2
1267	44	2
1277	45	2
1280	46	2
1282	51	2
1527	46	2
1676	47	2
1740	40	2
1824	58	2

Slika 30. Nezadovoljni menadžeri u odjelu prodaje.

Ako nas pak zanima koji od naših starijih zaposlenika ne osjećaju zadovoljstvo na radnom mjestu te želimo ih izdvojiti kako bismo mogli putem razgovora identificirati uzrok nezadovoljstva, cilj nam je pronaći zaposlenike u dobi od 55 do 65 godina koji su svoje zadovoljstvo na poslu ocijenili ocjenama 2 i 1. Koristi se sljedeći upit:

```
SELECT employee_id, department, job_role, age, job_satisfaction FROM
emp_keyspace.employee_by_department_role WHERE age >= 55 AND age <= 65 AND
job_satisfaction <3 ALLOW FILTERING;
```



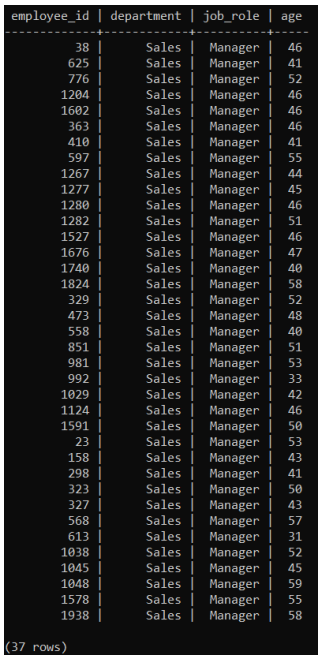
employee_id	department	job_role	age	job_satisfaction
1338	Human Resources	Manager	56	2
1973	Human Resources	Manager	55	2
597	Sales	Manager	55	2
1824	Sales	Manager	58	2
81	Sales	Sales Executive	59	1
532	Sales	Sales Executive	56	1
573	Sales	Sales Executive	60	1
732	Sales	Sales Executive	60	1
1935	Sales	Sales Executive	56	1
940	Research & Development	Healthcare Representative	58	2
1034	Research & Development	Healthcare Representative	55	2
1278	Research & Development	Healthcare Representative	55	2
10	Research & Development	Laboratory Technician	59	1
374	Research & Development	Manager	55	1
549	Research & Development	Manager	60	1
787	Research & Development	Manager	55	1
1074	Research & Development	Manager	55	1
1191	Research & Development	Manager	56	1
1360	Research & Development	Manufacturing Director	58	1
1096	Research & Development	Manufacturing Director	55	2
1373	Research & Development	Manufacturing Director	56	2
1423	Research & Development	Research Director	58	1
1770	Research & Development	Research Director	55	1
254	Research & Development	Research Director	55	2
825	Research & Development	Research Director	58	2
241	Research & Development	Research Scientist	56	1
1441	Research & Development	Research Scientist	56	1
161	Research & Development	Research Scientist	56	2
214	Research & Development	Research Scientist	58	2
1694	Research & Development	Research Scientist	55	2

Slika 31. Stariji nezadovoljni zaposlenici

U ovom upitu vidimo potrebu za korištenjem naredbe ALLOW FILTERING zbog stupca *age*, koji nije dio primarnog ključa, te nedostaju *department* i *job_role* u filtriranju. Ranije je spomenuto kako ovo nije optimalna metoda, jer je izuzetno zahtjevna i spora kada se radi s velikim skupovima podataka. Utjecaj na performanse može rezultirati velikom upotrebom CPU-a i memorije. Za ovaj dataset koji ne sadrži tako veliku količinu podataka, ova metoda nije previše štetna, ali nije preporučljivo njeno korištenje.

Ako nas zanima koji zaposlenici imaju određenu poziciju u određenom odjelu, to možemo doznati sljedećim upitom:

```
SELECT employee_id, department, job_role, age FROM
emp_keyspace.employee_by_department_role WHERE department = 'Sales' AND job_role =
'Manager';
```



employee_id	department	job_role	age
38	Sales	Manager	46
625	Sales	Manager	41
776	Sales	Manager	52
1204	Sales	Manager	46
1602	Sales	Manager	46
363	Sales	Manager	46
410	Sales	Manager	41
597	Sales	Manager	55
1267	Sales	Manager	44
1277	Sales	Manager	45
1280	Sales	Manager	46
1282	Sales	Manager	51
1527	Sales	Manager	46
1676	Sales	Manager	47
1740	Sales	Manager	40
1824	Sales	Manager	58
329	Sales	Manager	52
473	Sales	Manager	48
558	Sales	Manager	40
851	Sales	Manager	51
981	Sales	Manager	53
992	Sales	Manager	33
1029	Sales	Manager	42
1124	Sales	Manager	46
1591	Sales	Manager	50
23	Sales	Manager	53
158	Sales	Manager	43
298	Sales	Manager	41
323	Sales	Manager	50
327	Sales	Manager	43
568	Sales	Manager	57
613	Sales	Manager	31
1038	Sales	Manager	52
1045	Sales	Manager	45
1048	Sales	Manager	59
1578	Sales	Manager	55
1938	Sales	Manager	58

(37 rows)

Slika 32. Svi menadžeri u odjelu prodaje

Slijedećim upitom dobivena je lista svih menadžera u odjelu prodaje, zajedno s njihovim godinama. Ono što je od interesa u ovom pogledu jest utvrditi koja je pozicija u odjelu prodaje najzadovoljnija svojim poslom:

```
cqlsh:emp_keyspace> SELECT department, job_role, AVG(job_satisfaction) AS avg_satisfaction
                ... FROM emp_keyspace.employee_by_department_role
                ... WHERE department='Sales'
                ... GROUP BY department, job_role;

 department | job_role            | avg_satisfaction
------------+---------------------+------------------
      Sales |             Manager |                2
      Sales |     Sales Executive |                2
      Sales | Sales Representative |                2
```

Slika 33. Prosječno zadovoljstvo u odjelu prodaje

Koristi se agregacija AVG ili prosječna vrijednost koja računa prosjek zadovoljstva zaposlenika za svaku poziciju unutar Sales odjela. Za svaku od pozicija u odjelu Prodaje, prosječno zadovoljstvo iznosi 2.

Drugi pogled

CREATE MATERIALIZED VIEW emp_keyspace.employee_by_attrition_level AS

SELECT employee_id, attrition, job_level, years_at_company, job_role, department

FROM emp_keyspace.employee_id

WHERE attrition IS NOT NULL AND department IS NOT NULL AND job_role IS NOT NULL AND employee_id IS NOT NULL

PRIMARY KEY (attrition, department, job_role, employee_id);

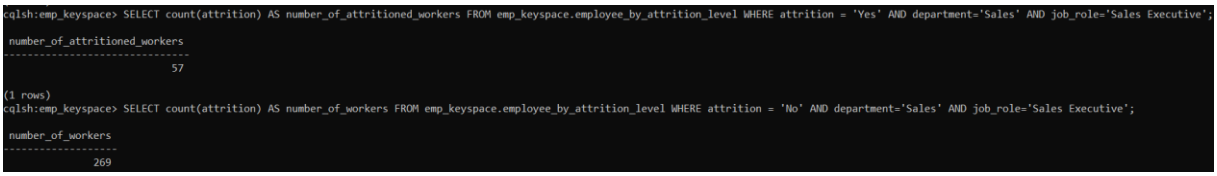
U ovom pogledu riječ je o zaposlenicima u tablici koji su napustili organizaciju i oni koji još uvijek rade u njoj. Još uvijek su nam od interesa *department* i *job_role,* no sada je primarnom ključu u pogledu dodan stupac *attrition*.

Prvi upit odnosi se na broj radnika na svakoj poziciji u odjelu prodaje, s obzirom na to jesu li još uvijek zaposleni u organizaciji ili ne.

U rezultatima prvoga upita vidimo da je organizaciju napustilo 33 radnika na odjelu *Sales*, s ulogom *Sales Representative* dok ih je još uvijek zaposleno 50.

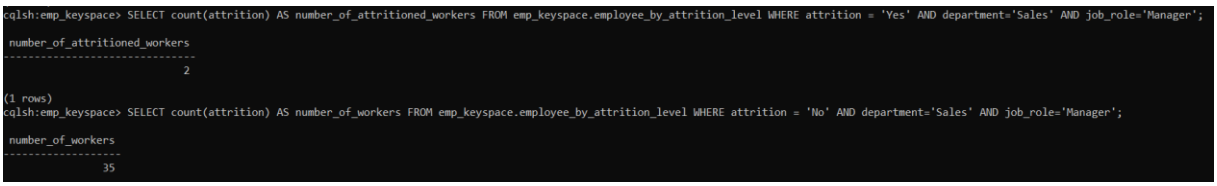
Za brojanje zaposlenika koristi se agregacija COUNT, koja prolazi kroz podatke i broji svakog zaposlenika koji pripada određenom odjelu i poziciji.

Drugi upit vezan je za *Sales Executive* poziciju, na kojoj vidimo da je organizaciju napustilo 57 radnika, dok ih je još uvijek zaposleno 269.



Slika 34. Sales Executive atricija

Treći upit je usmjeren na *Manager* poziciju unutar odjela *Sales,* gdje primjećujemo da za organizaciju više ne rade 2 radnika u bazi, dok ih 35 još uvijek ima mjesto unutar organizacije.



Slika 35. Manager atricija

Može se izvući zaključak da pozicija *Sales Representative* možda ima neki unutarnji problem, jer primjećujemo da na toj poziciji dolazi do najvišeg napuštanja organizacije. S druge strane, *Manager* pozicija ima najmanji broj zaposlenika koji su napustili organizaciju.

Treći pogled

CREATE MATERIALIZED VIEW emp_keyspace.employee_by_monthly_income AS

SELECT employee_id, monthly_income, job_level, job_role, department, years_at_company

FROM emp_keyspace.employee_id

WHERE monthly_income IS NOT NULL AND department IS NOT NULL AND job_role IS NOT NULL AND employee_id IS NOT NULL

PRIMARY KEY (department, job_role, monthly_income, employee_id);

U zadnjem pogledu koji se bavi točkom interesa vezanog uz radna mjesta dodajemo nove stupce u pogled koji nisu bili uključeni u prijašnjem pogledu.

Zanima nas kakvi su odnosi plaća unutar odjela. Uzimamo stupce: *department, job_role, monthly_income, job_level* i *years_at_company* kako bismo dokučili što sve utječe na visinu mjesečne plaće zaposlenika. Upit je sljedeći:

```
SELECT department, job_role,AVG(monthly_income) AS average_income, AVG(job_level) AS average_job_level, AVG(years_at_company) AS average_years_at_company FROM emp_keyspace.employee_by_monthly_income GROUP BY department, job_role;
```

department	job_role	average_income	average_job_level	average_years_at_company
Human Resources	Human Resources	4235	1	5
Human Resources	Manager	18088	4	16
Sales	Manager	16986	4	15
Sales	Sales Executive	6924	2	7
Sales	Sales Representative	2626	1	2
Research & Development	Healthcare Representative	7528	2	8
Research & Development	Laboratory Technician	3237	1	5
Research & Development	Manager	17130	4	13
Research & Development	Manufacturing Director	7295	2	7
Research & Development	Research Director	16033	3	10
Research & Development	Research Scientist	3239	1	5

Slika 36. Razlike u plaćama među pozicijama

Iz dobivenih podataka je vidljiva velika varijacija kada je riječ o visinama mjesečnih plaća u organizaciji, a također se uočavaju i pokazatelji koji objašnjavaju zašto je to tako. Naime, većina visoko plaćenih pozicija u prosjeku ima višu vrijednost u stupcu *job_level*, te su zaposlenici na tim pozicijama u prosjeku zaposleni već dulje vrijeme. Također se primjećuje da su menadžerske pozicije unutar tvrtke znatno bolje plaćene od ostalih, te da zaposlenici na tim pozicijama imaju značajno viši prosjek godina rada u organizaciji, što ukazuje na njihovo iskustvo u menadžerskim ulogama unutar organizacije.

Ako se samo usporede međusobno, primjećuje se da su ukupne prosječne plaće jednake. No, i dalje se uočava da zbog niže vrijednosti stupca *job_level* i manjeg prosjeka godina u organizaciji, odjel Research & Development ima nižu prosječnu plaću od ostalih odjela.

department	average_income	average_job_level	average_years_at_company
Human Resources	6654	2	7
Sales	6959	2	7
Research & Development	6281	1	6

Slika 37. Razlike u plaćama među odjelima

Ranije u radu se spominje praksa u Cassandri u kojoj se tablice definiraju ovisno o vrsti upita koje želimo postavljati, zato se u dosadašnjim upitima fokusiralo na odjele i pozicije unutar tih odjela za što svjedoči i sama struktura primarnog ključa tablice *employee_id*.

Zbog ovakvog načina postavljanja upita u Cassandri je filtriranje upita ograničeno na filtriranje samo stupcima u primarnom ključu osim ako se koristi naredba ALLOW

FILTERING kao u nekim prethodnim upitima. Ranije je spomenuto da je korištenje naredbe ALLOW FILTERING loša praksa zbog zahtjevnijeg čitanja tablice što za masivne tablice koje se u praksi koriste u Cassandri nisu optimalne. Kako bi se nastavila analizu podataka bez ovakvih opterećenja na Cassandri, može se napraviti još jedna tablica s drugačijom strukturom primarnog ključa.

5.2.2. Druga tablica – Analiza edukacije zaposlenika

Edukacija zaposlenika

```
CREATE TABLE emp_keyspace.employee_id_education (
    employee_id int,
    education_field text,
    job_role text,
    gender text,
    age int,
    attrition text,
    business_travel text,
    department text,
    education int,
    job_involvement int,
    job_level int,
    job_satisfaction int,
    monthly_income int,
    num_companies_worked int,
    over_time text,
    performance_rating int,
    total_working_years int,
    years_at_company int,
    years_in_current_role int,
    PRIMARY KEY (employee_id, education_field, job_role, gender)
```

Slika 38. Tablica employee_id_education

COPY employee_id_education FROM 'C:\Users\David\Desktop\employee_data.csv' WITH DELIMITER=';' AND HEADER=TRUE;

U primarnom ključu vidi se da su od interesa, u nadolazećim upitima, edukacija zaposlenika, njihova pozicija u organizaciji te spol zaposlenika.

Prvi Materijalizirani pogled:

CREATE MATERIALIZED VIEW emp_keyspace.employee_by_gender_income AS

SELECT employee_id, education_field, job_role, gender, monthly_income, years_at_company, education FROM emp_keyspace.employee_id_education

WHERE employee_id IS NOT NULL AND education_field IS NOT NULL AND job_role IS NOT NULL AND gender IS NOT NULL

PRIMARY KEY (education_field, job_role, gender, employee_id);

Prvim upitom želimo dohvatiti informacije ovisno o prosječnoj mjesečnoj plaći zaposlenika te godinama rada u tvrtki, uzimajući u obzir karakteristike kao što su edukacija, pozicija i spol. Ova vrsta upita mogla bi otkriti kako plaća i trajanje zaposlenja fluktuiraju ovisno o demografiji i odgovornostima unutar tvrtke.

SELECT education_field, job_role, gender, AVG(monthly_income) AS average_monthly_income, education, AVG(years_at_company) FROM emp_keyspace.employee_by_gender_income GROUP BY education_field, job_role, gender

education_field	job_role	gender	average_monthly_income
Technical Degree	Healthcare Representative	Female	8550
Technical Degree	Healthcare Representative	Male	8115
Technical Degree	Human Resources	Female	2742
Technical Degree	Human Resources	Male	3194
Technical Degree	Laboratory Technician	Female	4275
Technical Degree	Laboratory Technician	Male	3174
Technical Degree	Manager	Female	16643
Technical Degree	Manager	Male	15568
Technical Degree	Manufacturing Director	Female	9505
Technical Degree	Manufacturing Director	Male	6862
Technical Degree	Research Director	Female	15845
Technical Degree	Research Director	Male	12808
Technical Degree	Research Scientist	Female	3100
Technical Degree	Research Scientist	Male	3040
Technical Degree	Sales Executive	Female	7051
Technical Degree	Sales Executive	Male	7831
Technical Degree	Sales Representative	Female	2487
Technical Degree	Sales Representative	Male	2908
Medical	Healthcare Representative	Female	7582
Medical	Healthcare Representative	Male	8163
Medical	Human Resources	Female	2128
Medical	Human Resources	Male	5203
Medical	Laboratory Technician	Female	3216
Medical	Laboratory Technician	Male	3186
Medical	Manager	Female	16680
Medical	Manager	Male	16924
Medical	Manufacturing Director	Female	7023
Medical	Manufacturing Director	Male	6614
Medical	Research Director	Female	15388
Medical	Research Director	Male	16554
Medical	Research Scientist	Female	3395
Medical	Research Scientist	Male	3086
Medical	Sales Executive	Female	6662
Medical	Sales Executive	Male	6743
Medical	Sales Representative	Female	2495
Medical	Sales Representative	Male	2108

Slika 39. Prosječne plaće po edukacijskom smijeru i spolu

Prema podacima vidimo da plaće među spolovima variraju: žene više zarađuju na nekim pozicijama, dok muškarci više zarađuju na drugima. Budući da je moguće da postoje još pokazatelja, nastavljamo dodavati nove stupce u upitu.

education_field	job_role	gender	average_monthly_income	average_education	average_years_at_company
Technical Degree	Healthcare Representative	Female	8550	3	12
Technical Degree	Healthcare Representative	Male	8115	3	7
Technical Degree	Human Resources	Female	2742	4	2
Technical Degree	Human Resources	Male	3194	2	3
Technical Degree	Laboratory Technician	Female	4275	3	7
Technical Degree	Laboratory Technician	Male	3174	2	4
Technical Degree	Manager	Female	16643	3	4
Technical Degree	Manager	Male	15568	2	14
Technical Degree	Manufacturing Director	Female	9505	2	11
Technical Degree	Manufacturing Director	Male	6862	2	7
Technical Degree	Research Director	Female	15845	3	15
Technical Degree	Research Director	Male	12808	1	9
Technical Degree	Research Scientist	Female	3100	2	4
Technical Degree	Research Scientist	Male	3040	3	5
Technical Degree	Sales Executive	Female	7051	2	8
Technical Degree	Sales Executive	Male	7831	3	6
Technical Degree	Sales Representative	Female	2487	1	3
Technical Degree	Sales Representative	Male	2908	2	3

Slika 40. Dodavanje edukacijske razine i prosjeka godina rada u organizaciji

Nakon dodavanja novih stupaca *education* i *years_at_company* vidljiva je korelacija s mjesečnom plaćom zaposlenika. Ne postoji instanca za poziciju u kojoj jedan spol zaposlenika ima veću prosječnu plaću od drugoga, a istovremeno ima nižu prosječnu razinu obrazovanja te u prosjeku manji broj godina rada za organizaciju. Postoje određene pozicije u kojima jedan spol ima jednak prosječan nivo obrazovanja i duži staž u organizaciji, ali nižu prosječnu plaću. Ti primjeri ponekad idu u korist ženskih, a ponekad u korist muških zaposlenika.

Može se zaključiti da, na temelju analiziranih stupaca, nema vidljivih razlika u mjesečnim plaćama između spolova.

U drugom upitu želimo doznati koji broj zaposlenika u svakom polju edukacije za svaki spol radi u organizaciji.

SELECT education_field, gender, job_role, COUNT(*) AS employee_count

FROM emp_keyspace.employee_by_gender_income GROUP BY education_field, job_role, gender;

education_field	gender	job_role	employee_count
Technical Degree	Female	Healthcare Representative	8
Technical Degree	Male	Healthcare Representative	6
Technical Degree	Female	Human Resources	1
Technical Degree	Male	Human Resources	3
Technical Degree	Female	Laboratory Technician	4
Technical Degree	Male	Laboratory Technician	15
Technical Degree	Female	Manager	3
Technical Degree	Male	Manager	2
Technical Degree	Female	Manufacturing Director	6
Technical Degree	Male	Manufacturing Director	8
Technical Degree	Female	Research Director	4
Technical Degree	Male	Research Director	1
Technical Degree	Female	Research Scientist	12
Technical Degree	Male	Research Scientist	27
Technical Degree	Female	Sales Executive	9
Technical Degree	Male	Sales Executive	10
Technical Degree	Female	Sales Representative	5
Technical Degree	Male	Sales Representative	8
Medical	Female	Healthcare Representative	22
Medical	Male	Healthcare Representative	26

Slika 41. Broj zaposlenika s određenim poljem edukacije

U edukacijskom polju tehničkog usmjerenja, zaposlen je veći broj muškaraca (80) nego žena (52), dok su žene u prosjeku unutar organizacije provele 7 godina, a muškarci 5. Iako tehničko usmjerenje ima više muškaraca, žene ipak u prosjeku dulje rade unutar organizacije.

U medicinskom obrazovanju, broj žena iznosi 109 nasuprot 274 muškaraca. Opet, žene u ovom području prosječno dulje rade za organizaciju od muškaraca (žene 7 godina, muškarci 5). Trend se nastavlja, iako je ovoga puta razlika mnogo primjetljivija, s više muškaraca nego žena. Žene i dalje ostaju dulje u organizaciji.

U području životnih znanosti, tablica prikazuje 366 muškaraca u usporedbi s 240 žena. Međutim, muškarci u prosjeku ostaju u organizaciji 7 godina, dok žene ostaju 6 godina. U ovom području trend se blago mijenja, ali muškarci i dalje dulje ostaju u organizaciji.

Trend se nastavlja i u marketingu, gdje je zabilježeno 90 muškaraca s tom diplomom, nasupram 69 žena. Ponovno, žene u prosjeku rade dulje od muškaraca.

U području ljudskih resursa, evidentirano je 19 muškaraca i 8 žena. Prosječno, muškarci ostaju unutar organizacije 7 godina, dok žene ostaju 5.

Vidljivo je da iz svakog obrazovnog područja postoji više zaposlenih muškaraca.

5.2.3. Treća tablica – Pozicije u organizaciji po spolovima

Pozicije po spolovima

Izradom nove tablice od interesa je razlika između muškaraca i žena na višim pozicijama unutar organizacije.

CREATE TABLE emp_keyspace.employee_gender_position(attrition text, job_role text, gender text, job_level int, employee_id int, age int, department text, business_travel text, education_field text, education int, job_involvement int, job_satisfaction int, monthly_income

int, num_companies_worked int, over_time text, performance_rating int, total_working_years int, years_at_company int, years_in_current_role int, PRIMARY KEY (job_role, gender, job_level, attrition, employee_id));

```
CREATE MATERIALIZED VIEW emp_keyspace.employee_gender_job_role AS
SELECT job_role, gender, monthly_income, years_at_company, job_level, attrition, employee_id FROM emp_keyspace.employee_gender_position
WHERE job_role IS NOT NULL AND gender IS NOT NULL  AND job_level IS NOT NULL AND attrition IS NOT NULL  AND employee_id IS NOT NULL
PRIMARY KEY ( job_role, gender, job_level, attrition, employee_id);
```

Slika 42. Materialized view employee_gender_job_role

Upitima ćemo pokušati doći do spoznaja vezanih uz razlike u plaćama, među zaposlenicima na višim pozicijama s obzirom na spol zaposlenika:

```
cqlsh:emp_keyspace> SELECT count(employee_id) AS number_of_female_managers, job_role, avg(monthly_income) FROM emp_keyspace.employee_gender_job_role WHERE job_role='Manager' AND gender='Female';

 number_of_female_managers | job_role | system.avg(monthly_income)
---------------------------+----------+----------------------------
                        47 |  Manager |                      16915

(1 rows)
cqlsh:emp_keyspace> SELECT count(employee_id) AS number_of_male_managers, job_role, avg(monthly_income) FROM emp_keyspace.employee_gender_job_role WHERE job_role='Manager' AND gender='Male';

 number_of_male_managers | job_role | system.avg(monthly_income)
-------------------------+----------+----------------------------
                      55 |  Manager |                      17409
```

Slika 43. Razlike u menadžerskim pozicijama

U tablici organizacije vidimo upisane 47 menadžerica i 55 menadžera, te je prosječna mjesečna plaća za žene na menadžerskoj poziciji iznosila 16.915 dolara mjesečno, dok je za muškarce iznosila 17.409 dolara.

```
cqlsh:emp_keyspace> SELECT count(employee_id) AS female_managment,
r='Female' AND job_level=4 AND attrition='No';
 female_managment | job_role | avegare_monthly_income | attrition
------------------+----------+------------------------+-----------
               23 |  Manager |                  16657 |        No

(1 rows)
cqlsh:emp_keyspace> SELECT count(employee_id) AS female_managment,
r='Male' AND job_level=4 AND attrition='No';

 female_managment | job_role | avegare_monthly_income | attrition
------------------+----------+------------------------+-----------
               24 |  Manager |                  16571 |        No
```

Slika 44. Trenutni menadžeri

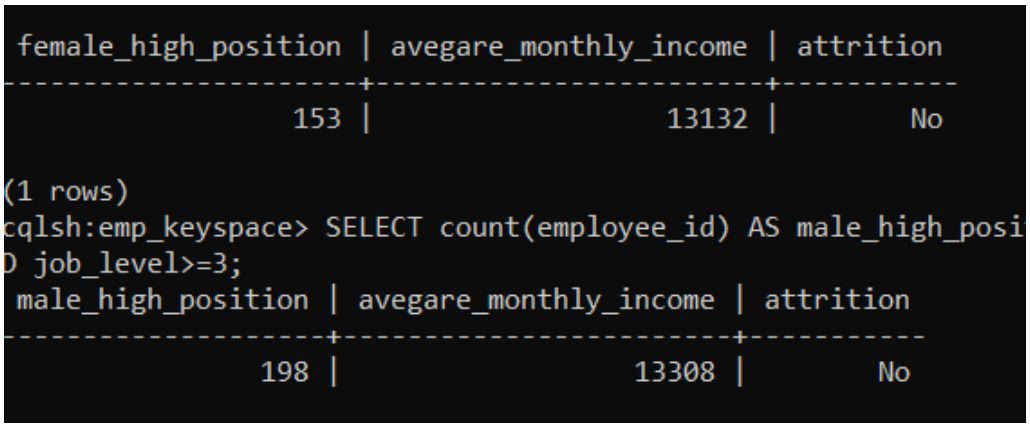
Trenutno se na poziciji menadžera u organizaciji nalazi 23 menadžerice i 24 menadžera.

Kako bih postigao sljedeće rezultate, mijenjam redoslijed stupaca koji čine primarni ključ u novoj *tablici employee_gender_job_role2*:

PRIMARY KEY (gender, attrition, job_level, job_role, employee_id).

Osim samih menadžerskih pozicija, sve više nas zanimaju pozicije u organizaciji gdje je *job_level* >= 3, te želimo istražiti kakva je razlika među spolovima u tom kontekstu. Koristit ćemo upit:

```
SELECT count(employee_id) AS female_high_position, avg(monthly_income) AS
avegare_monthly_income, attrition FROM emp_keyspace.employee_gender_job_role2
WHERE gender='Female' AND attrition='No' AND job_level>=3;
```



Slika 45. Zaposlenici na visokom položaju

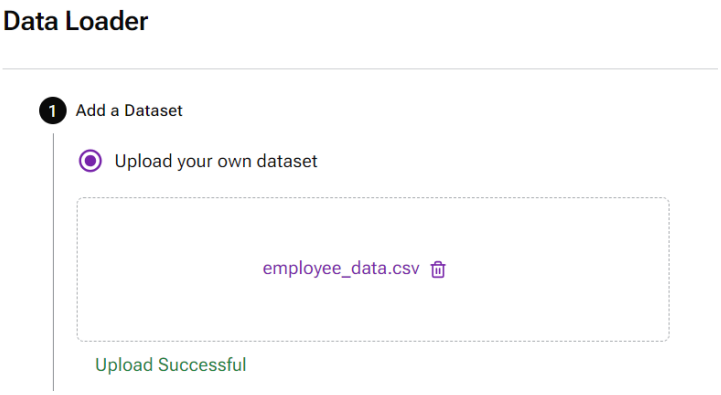
Vidimo da je trenutno u organizaciji 153 žene na visokim pozicijama u odnosu na 198 muškaraca te saznajemo da žene na tim pozicijama imaju sličnu plaću kao i muškarci.

U zaključku za ovaj dio analize možemo primijetiti da žene imaju plaće proporcionalne onima muškaraca te da visina plaće ovisi o duljini radnog perioda u organizaciji i stupnju obrazovanja. Organizacija zapošljava veći broj muškaraca, no kod poslova više razine (job_level >= 3) žene su vrlo blizu zastupljenosti muškaraca te čine 43,59% menadžmentskog osoblja. Na nižim pozicijama (job_level <= 2) žena je 348, a muškaraca 534, pa žene čine 39,45%.

5.3. Dodatni alati za analizu

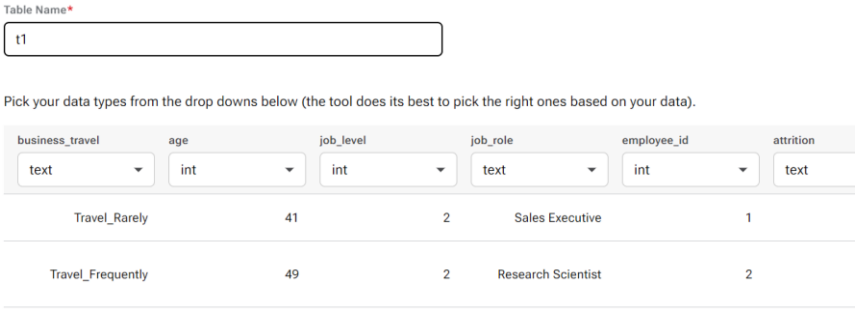
5.3.1. Datastax Astra

Zadnju tablicu ću izraditi pomoću *DataStax Astra* CQL terminala. Za razliku od cmd terminala, tablica se u *Datastax Astri* izrađuje ručno. Izrađuje se baza podataka u kojoj može biti više keyspaceova, te se putem *Data Loader-a* u bazu podataka uvoze željeni podatkovni izvori za analizu.

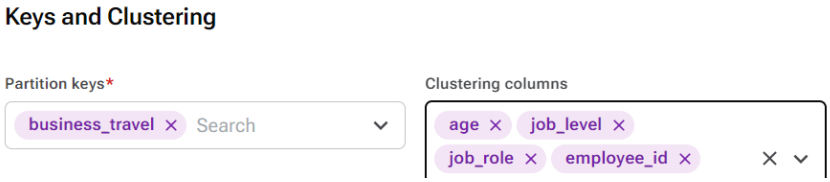


Slika 46. Data Loader

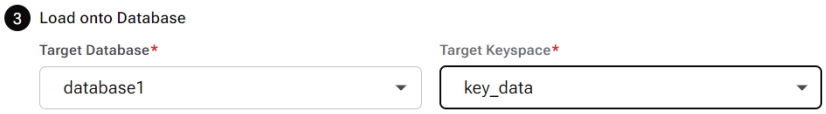
Nakon što se odabere CSV datoteka, sučelje nudi moguće tipove podataka za svaki stupac. Kada se provjeri točnost tipova, biraju se particijski ključevi i stupci za kastiranje. Na kraju se bira baza podataka i keyspace kojem će tablica pripadati.



Slika 47. Unos podataka u tablicu



Slika 48. Odabir primarnog ključa



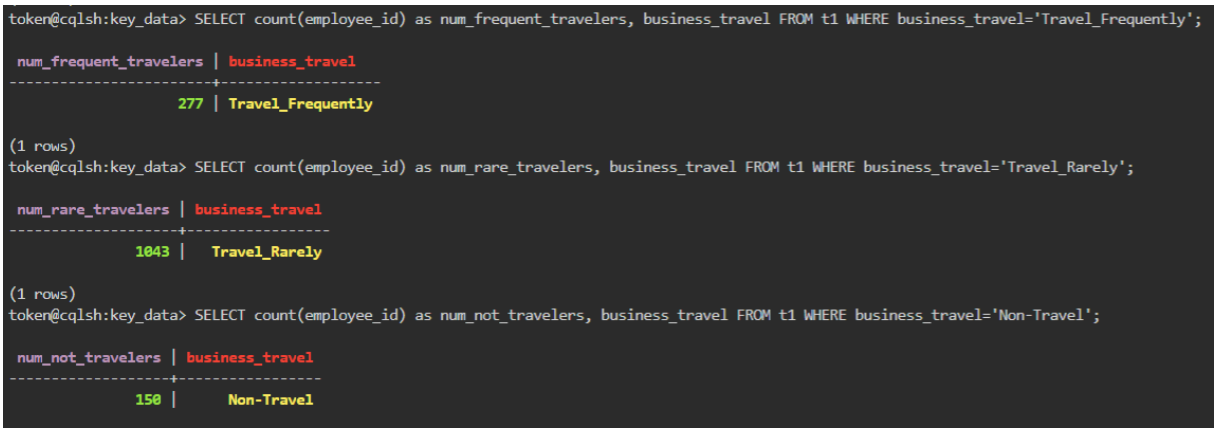
Slika 49. Odabir keyspacea u koji se upisuje tablica

Tablica koju sam odlučio napraviti istražit će podatke vezane za poslovna putovanja unutar organizacije, a tablica je u sljedećem obliku :

Slika 50. Tablica t1

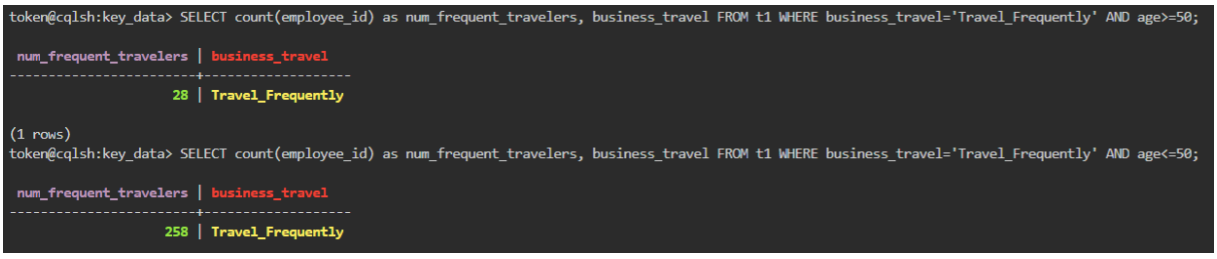
U slučaju tablica napravljenih u *Datastax Astri* nije moguća izrada materijaliziranih pogleda tako da sam upite usmjerio direktno na tablicu.

SELECT count(employee_id) as num_frequent_travelers, business_travel FROM t1 WHERE business_travel='Travel_Frequently';



Slika 51. Broj putnika za svaku kategoriju frekvencije putovanja

Po rezultatima prethodnog upita, vidimo da su u organizaciji najzastupljeniji zaposlenici koji rijetko putuju (70,95%), dok 18,84% često putuje, a 10,20% čine zaposlenici koji uopće ne putuju. Prvo zapažanje je da najviše zaposlenika rijetko poslovno putuje (1043), dok njih 277 često putuje, a najmanje je onih koji ne putuju (150).



Slika 52. Broj putnika po generacijama

Nakon analize putovanja po godinama došao sam do sljedećih zaključaka:

Ako se radi o grupi zaposlenika starijih od 50 godina, njih 28 putuje često, 132 rijetko putuje, te ih 13 ne putuje. Ukupno ih je 173, te 16 % putuje često, 76 % putuje rijetko, a 7.6 % ne putuje.

Ako se radi o grupi zaposlenika mlađih od 50 godina, a starijih od 45, njih 25 putuje često, 95 rijetko putuje, te ih 10 ne putuje. Ukupno ih je 130, te 19 % putuje često, 73 % putuje rijetko, a 7.7 % ne putuje.

Ako se radi o grupi zaposlenika mlađih od 45 godina, a starijih od 40, njih 37 putuje često, 130 rijetko putuje, te ih 25 ne putuje. Ukupno ih je 192, te 19.2 % putuje često, 67.7 % putuje rijetko, a 13 % ne putuje.

Ako se radi o grupi zaposlenika starijih od 40 godina, a starijih od 35, njih 51 putuje često, 193 rijetko putuje, te ih 33 ne putuje. Ukupno ih je 277, te 18 % putuje često, 69 % putuje rijetko, a 12 % ne putuje.

Ako se radi o grupi zaposlenika mlađih od 35 godina, a starijih od 30, njih 76 putuje često, 231 rijetko putuje, te ih 36 ne putuje. Ukupno ih je 343, te 22 % putuje često, 67 % putuje rijetko, a 10.5 % ne putuje.

Ako se radi o grupi zaposlenika mlađih od 30 godina, a starijih od 25, njih 52 putuje često, 189 rijetko putuje, te ih 22 ne putuje. Ukupno ih je 263, te 19.7 % putuje često, 71 % putuje rijetko, a 8.36 % ne putuje.

Ako se radi o grupi zaposlenika mlađih od 25 godina, a starijih od 20, njih 14 putuje često, 84 rijetko putuje, te ih 8 ne putuje. Ukupno ih je 106, te 13 % putuje često, 79 % putuje rijetko, a 7.5 % ne putuje.

Najviše zaposlenika koji često putuju nalaze se u dobnoj skupini od 30 do 35 godina (76), a u toj dobnoj skupini je i najveći postotak čestih putnika, što iznosi 22%. To sugerira da će mlađi zaposlenici vjerojatnije sudjelovati u poslovnim putovanjima. Unutar dobne skupine od 50 godina i više, nalazi se najveći postotak zaposlenika koji rijetko putuju (79%), dok ih je najviše u skupini od 30 do 35 godina (231). Najveći postotak zaposlenika koji ne putuju je u grupi od 40 do 45 godina starosti, gdje njih 13% ne putuje. Zanimljivo je kako zaposlenici u grupi iznad 50 godina i grupi između 25 i 30 godina imaju najmanji postotak ne putnika u organizaciji. Jedan od zaključaka bi bio da se mladim zaposlenicima rano pruža prilika za poslovna putovanja te da su mladi zaposlenici na početku karijere izrazito ambiciozni i željni dokazivanja.

Ako se u obzir uzimaju samo visoke pozicije u organizaciji podaci su sljedeći:

-Ako se radi o grupi zaposlenika starijih od 50 godina, njih 19 putuje na visokim pozicijama, 83 rijetko putuje na visokim pozicijama, 8 ne putuje na visokim pozicijama.

-Ako se radi o grupi zaposlenika mlađih od 50 godina, a starijih od 45, njih 14 putuje na visokim pozicijama, 60 rijetko putuje na visokim pozicijama, 4 ne putuje na visokim pozicijama.

-Ako se radi o grupi zaposlenika mlađih od 45 godina, a starijih od 40, njih 12 putuje na visokim pozicijama, 54 rijetko putuje na visokim pozicijama, 6 ne putuje na visokim pozicijama.

-Ako se radi o grupi zaposlenika starijih od 40 godina, a starijih od 35, njih 13 putuje na visokim pozicijama, 50 rijetko putuje na visokim pozicijama, 11 ne putuje na visokim pozicijama.

-Ako se radi o grupi zaposlenika mlađih od 35 godina, a starijih od 30, njih 13 putuje na visokim pozicijama, 40 rijetko putuje na visokim pozicijama, 5 ne putuje na visokim pozicijama.

-Ako se radi o grupi zaposlenika mlađih od 30 godina, a starijih od 25, njih 4 putuje na visokim pozicijama, 16 rijetko putuje na visokim pozicijama, 3 ne putuje na visokim pozicijama.

-Ako se radi o grupi zaposlenika mlađih od 25 godina, a starijih od 20, njih 0 putuje na visokim pozicijama, 0 rijetko putuje na visokim pozicijama, 0 ne putuje na visokim pozicijama.

Kada sam uključio visinu plaće u upite saznao sam sljedeće:

-Ako se radi o grupi zaposlenika starijih od 40 godina 42 putuje na visokim pozicijama, 197 rijetko putuje na visokim pozicijama, 19 ne putuje na visokim pozicijama. Ukupno je 258 zaposlenika na visokim pozicijama koji su stariji od 40 godina, od čega 16,27% često putuje često te u prosjeku zarađuju 13.607 dolara mjesečno, 76,35% rijetko putuje, te zarađuju prosječno 14.494 dolara mjesečno, 7.36% ne putuje te zarađuje prosječno 15.029 dolara mjesečno.

-Ako se radi o grupi zaposlenika mlađih od 40 godina, njih 30 putuje na visokim pozicijama, 106 rijetko putuje na visokim pozicijama, 19 ne putuje na visokim pozicijama. Ukupno je na visokim pozicijama mlađe od 40, 155 zaposlenika, od kojih 19.35% često putuju te u prosjeku zarađuju 11.071 dolara mjesečno, 68.39% rijetko putuje te u prosjeku zarađuju 10.637 dolara mjesečno, 12.26% ne putuju te zarađuju u prosjeku 11.146 dolara mjesečno.

U prosjeku, zaposlenici mlađi od 40 godina koji često putuju zarađuju manje od svojih kolega koji rijetko putuju ili ne putuju. Među obje dobne skupine, zaposlenici koji ne putuju, a zauzimaju visoke položaje obično imaju najveće prosječne prihode. To bi moglo značiti da bi ti zaposlenici mogli imati važne pozicije koje ne zahtijevaju česta putovanja, ali su dobro plaćene. Unutar svake dobne skupine postoji jasan trend gdje najveći prosječni prihod ostvaruju zaposlenici koji ne putuju, zatim slijede oni koji rijetko putuju, a potom oni koji često putuju.

5.3.2. Stargate Document API

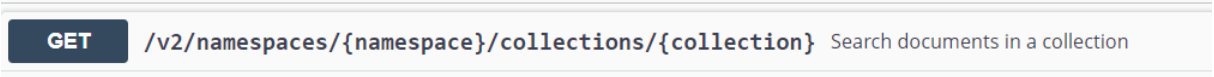
Analizirali smo podatke koji su bili strukturalno slični onima koji se nalaze u SQL-u. Podaci u CSV datotekama su prilično dobro strukturirani, pa se ova datoteka češće analizira pomoću SQL-a. U sljedećem dijelu želim prikazati na koji način je moguća analiza podataka kada su podaci pohranjeni u JSON formatu. JSON format je prilično neprikladan za direktnu integraciju u samu Cassandru, stoga je jedno od rješenja ručni unos JSON podataka direktno u bazu podataka.

Za analizu ću koristiti *Datastax Astra* platformu, koja omogućava izradu baza podataka u oblaku i nudi olakšani te jednostavniji način implementacije, upravljanja i korištenja baza podataka putem CQL-a. Također ću koristiti alat *Stargate* kako bih pripremio podatke za obradu nestrukturiranih podataka i učinio ih razumljivima Cassandri, jer Cassandra sama po sebi ne razumije JSON format.

Prilikom odabira baze podataka, koristi se *Connection* preko kojeg se generira token, a potom se preusmjerava na *Swagger UI*. Na Swaggeru je dostupan *Stargate Document API* putem kojeg se stvaraju i manipuliraju podaci pohranjeni kao nestrukturirani JSON dokumenti unutar kolekcija. Za stvaranje nove kolekcije, koristi se akcija *Create a collection*.

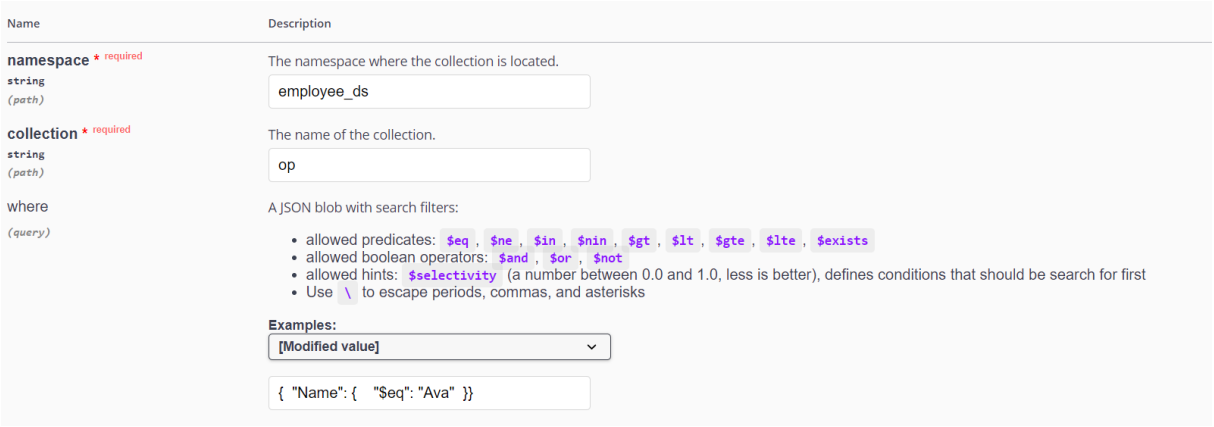
Slika 53. Stvaranje kolekcije

Upisuje se tolken za trenutnu sesiju te upisuje naziv keyspacea. Nakon toga se u JSON polju imenuje zbirka. Nakon imenovanja zbirke preko *Create a document* unosimo dokumente u stvorenu zbirku. Kada je dokument stvoren možemo ga preko *Search documents in a collection* analizirati.



Slika 54. Pretraga dokumenata

Slika 55. JSON dokument

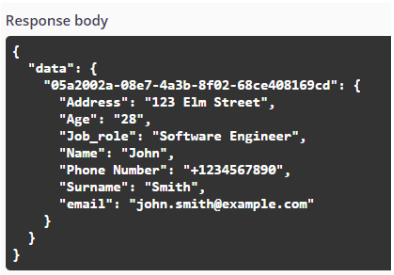


Slika 56. Upiti nad JSON dokumentom

Kolekcija se naziva *op,* te se u njoj preko ovog sučelja mogu provoditi upiti pomoću filtera pretraživanja, poput $eq, $ne, $in, $nin, $gt, $lt i $exist, koji su tipični za dokumentno orijentirane baze podataka, poput MongoDB-a:

Upit koji se koristi ako nas zanimaju svi zaposlenici s 28 godina:

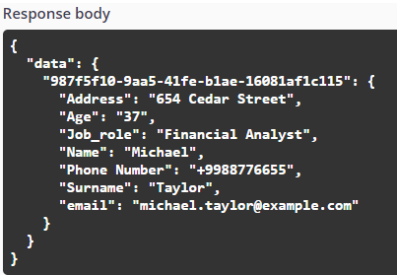
```
{ "Age": { "$eq": "28" }
```



Slika 57. Prikaz osoba s 28 godina

Upit u kojem tražimo zaposlenika pod prezimenom Taylor koji je na poziciji *Financial Analyst*.

```
{ "Surname": { "$eq": "Taylor" }, "Job_role": { "$eq": "Financial Analyst" }}
```



Slika 58. Prikaz osoba s prezimenom Taylor, koji rade na poziciji Financial Analyst

Upit u kojem nas zanimaju svi zaposlenici mlađi od 29 godina:

{ "Age": { "$lt": "29" }}

Slika 59. Prikaz osoba mlađih od 29 godina

Upit u kojem nas zanimaju samo zaposlenici koji nisu na poziciji *Financial Analyst*:

{ "Job_role": { "$ne": "Financial Analyst" }}

Upit koji vraća sve zaposlenike koji u dokumentu imaju zabilježen broj telefona:

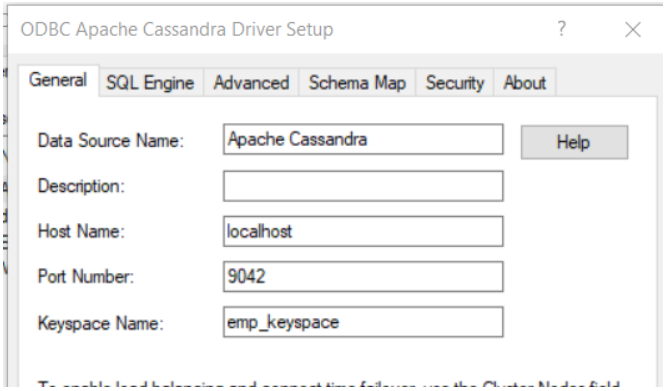
{ "Phone Number": { "$exists": true } }

Upit kojim tražimo zaposlenike koji su ili *Project Manager* ili *Operations Manager*

{ "$or": [{ "Job_role": { "$eq": "Project Manager" } }, { "Job_role": { "$eq": "Operations Manager" } }]}

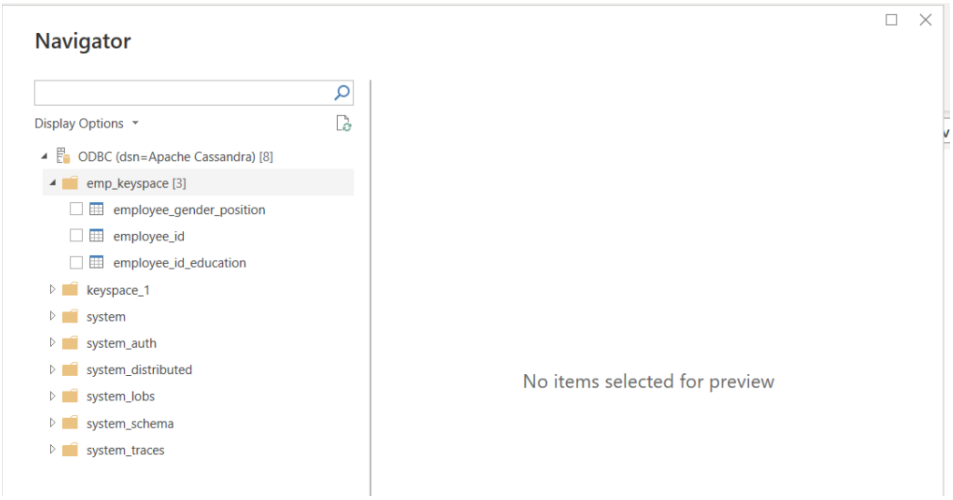
5.3.3. Izvještaji u Microsoft Power BI programu

Kako bih napravio dodatnu analizu i izvještaj o podacima iz baze podataka, odabrao sam alat Microsoft Power BI. Za povezivanje s Apache Cassandra bazom podataka koristio sam ODBC alat te stvorio ODBC izvor podataka koji sadrži informacije potrebne drugim alatima za uspješno povezivanje s bazom podataka.



Slika 60. ODBC

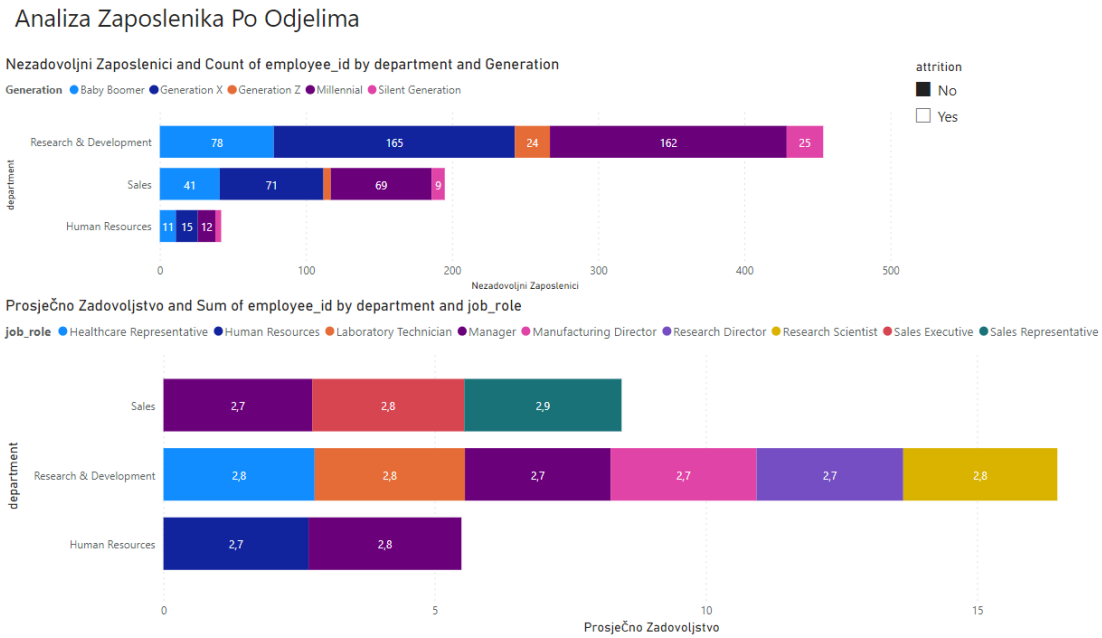
Nakon što je ODBC postavljen i nakon što je konekcija uspješno postavljena, u Power BI se preko *Get Data* izbornika odabire se ODBC kao izvor podataka koje se želi analizirati.



Slika 61. Povezivanje na ODBC Apache Cassandra

Nakon što se sve tablice unesu u Power BI može se započeti s analizom.

Izvještaj o odjelima, pozicijama, zadovoljstvu i generacijama:

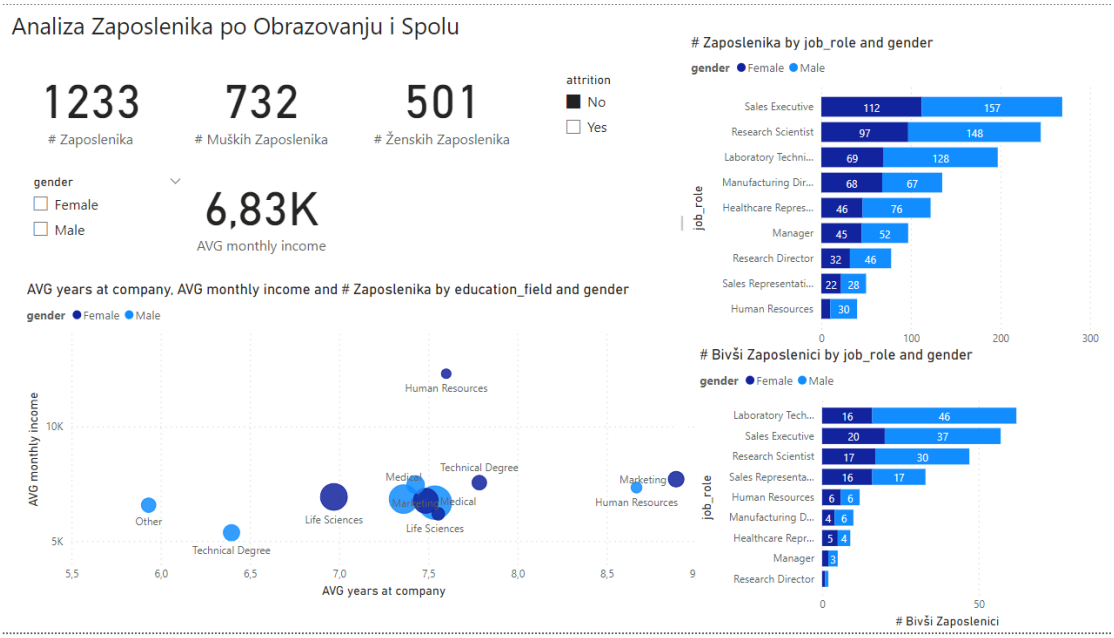


Slika 62. Izvještaj o odjelima, pozicijama, zadovoljstvu i generacijama

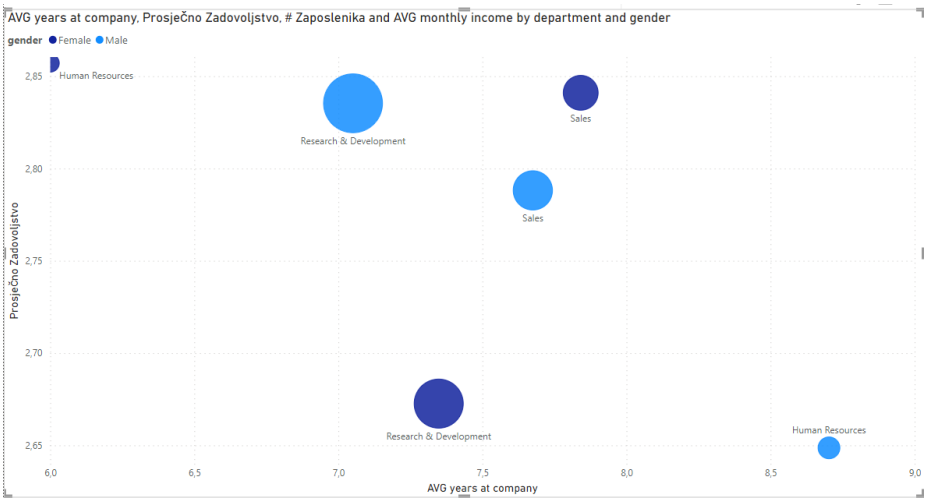
Kako bih dobio bolji uvid u podatke, svrstao sam zaposlenike po generacijama. U ovom izvještaju pružen je dublji uvid u prosječna zadovoljstva, pri čemu je jasno vidljivo da su najzadovoljniji *Sales Representatives*, dok su najnezadovoljniji na pozicijama unutar *Human*

Resources. Također, primjećuje se da je najveći broj nezadovoljnih zaposlenika u *Generation X*, gdje je od ukupno 313 zaposlenika njih 165 izrazilo nezadovoljstvo. Što se tiče *Human Resources*, u *Silent Generation* svi četiri zaposlenika iskazuju nezadovoljstvo.

Izvještaj o obrazovanju i spolovima.



Slika 63. Izvještaj o obrazovanju i spolovima

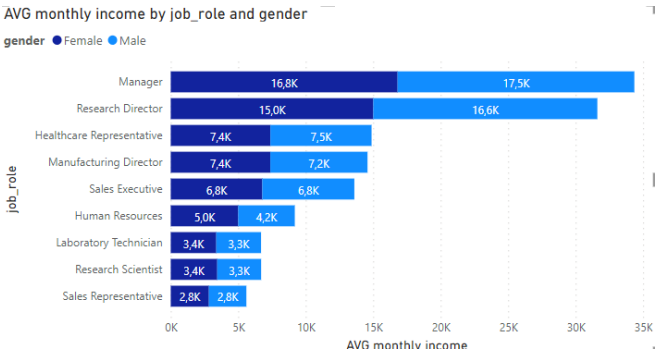


Slika 64. Zadovoljstvo unutar odjela ovisno o spolovima

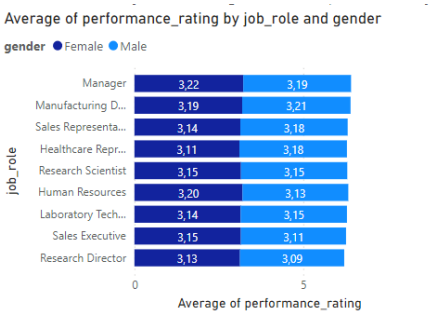
Jedna od zanimljivosti vidljivih u ovom izvještaju jest da žene imaju višu prosječnu plaću od muškaraca. Zanimljivo je da, iako muškarci koji posjeduju edukaciju u području *Human Resources* i u prosjeku dulje rade u organizaciji od žena s istom edukacijom, zarađuju u prosjeku čak 5 tisuća dolara manje. Razlika u parametru *performance_rating* iznosi prosječnu

ocjenu 3.02, pri čemu su muškarci s ovom edukacijom najlošije ocijenjeni u cijeloj organizaciji, dok su žene s ovom edukacijom najbolje ocijenjene s rezultatom od 3.20. Unatoč tome što muškarci u odjelu *Human Resources* rade u prosjeku najdulje u odnosu na sve ostale odjele, predstavljaju najmanje zadovoljnu skupinu u organizaciji, dok su žene najzadovoljnija skupina s najkraćim prosječnim brojem godina u organizaciji. U odjelu *Research & Development* primjećuje se manja razlika u broju godina radnog staža, ali veća u prosječnoj ocjeni zadovoljstva. Međutim, uzrok ovome vjerojatno nije plaća, s obzirom da nema primjetljive razlike između muškaraca i žena.

Što se tiče iskustva unutar poduzeća, najduži prosjek godina radnog staža zabilježen je u odjelu marketinga - 8.89 godina. Zanimljivo je da su osobe ženskog spola imale najviše prosječne godine staža i mjesečna primanja unutar tvrtke. Značajna razlika između broja zaposlenika i zaposlenica uočena je na radnom mjestu *Laboratory Technician*, gdje je broj muškaraca premašio broj žena za 59 zaposlenika. Veliki faktor odlaska zaposlenika iz organizacije bila je neravnoteža u plaćama unutar većine odjela.



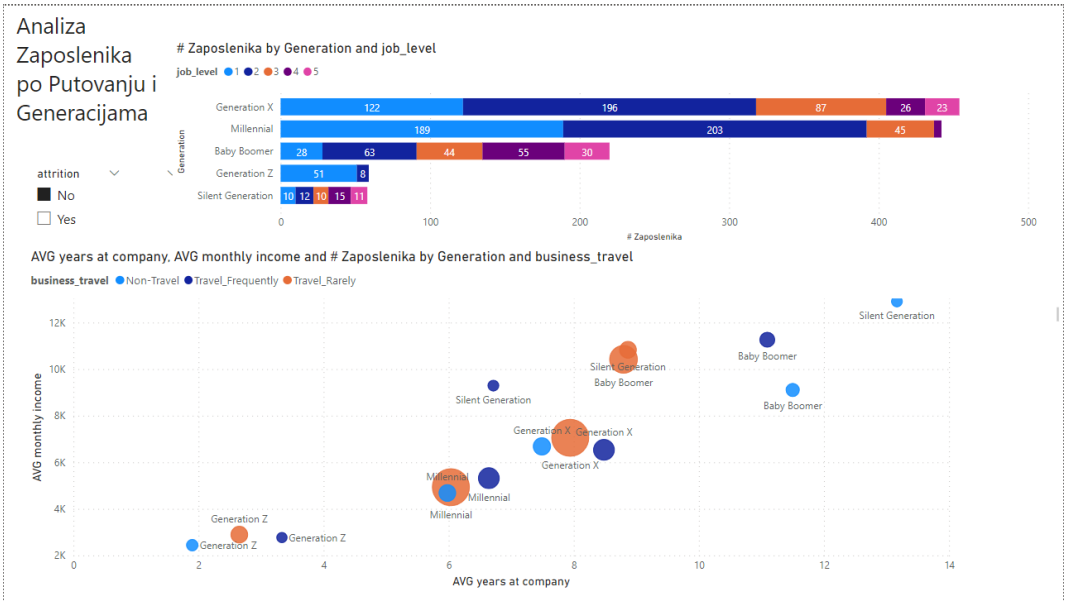
Slika 65. Prosječna plaća po poziciji i spolu



Slika 66. Prosječni rezultati izvedbe

Što se tiče prosječnih rezultata iz stupca *performance_rating* i mjesečnih plaća postoje neke male razlike no nema značajnih anomalija.

Izvještaj o putovanju ovisno o generacijama



Slika 67. Izvještaj o putovanju ovisno o generacijama

Među različitim generacijama, *Silent Generation* se istaknula najvećim prosječnim brojem godina provedenih u tvrtki od 13,17 godina, uz najveći prosječni mjesečni prihod od 12.905,67 dolara. Pri ispitivanju kategorija poslovnih putovanja, primijećeno je da kategorija *Non-Travel* ima najviši prosjek godina u tvrtki i najviši prosječan mjesečni prihod.

Za većinu generacija, prosječne godine provedene u tvrtki uglavnom su bile ispod 7,37 godina, dok je prosječni mjesečni prihod ostao ispod 7.142,03 dolara.

6. Pregled drugih NoSQL baza podataka

U završnom dijelu rada, analizirat će se NoSQL baze podataka, od kojih svaka predstavlja jedinstveni pristup upravljanju podacima. Prvo će se napraviti kratak pregled Redis-a, koji predstavlja bazu podataka parova ključ-vrijednost, istaknutu po brzom dohvaćanju podataka putem jednostavnog pretraživanja ključeva. MongoDB će biti predstavnik baza podataka orijentiranih na dokumente, koja pohranjuje podatke u fleksibilnim dokumentima nalik na JSON, namijenjena za dinamičke i skalabilne aplikacije. Budući da je Cassandra već opisana u ovom radu, u ovom poglavlju će Hadoop poslužiti kao primjer baze podataka usmjerene na stupce. Posljednja baza podataka koja će biti predstavljena je Neo4j, primjer baze podataka temeljene na grafovima.

Redis

Redis ili "Remote Dictionary Server" je NoSQL ključno-vrijednosna baza podataka koja se primarno koristi kao predmemorija za aplikacije ili kao baza podataka za brze odgovore. Ono što je čini izrazito brzom i pouzdanom jest to što podatke pohranjuje u memoriji, a ne na disku ili SSD-u. Budući da podatke pohranjuje u memoriju koja je bliža aplikaciji nego vanjskim izvorima podataka, smanjuje se promet i poboljšava performansa prilikom pisanja i čitanja podataka. Model ključ-vrijednost Redisa čini ga prikladnim za predmemoriju, upravljanje sesijama i analitiku u stvarnom vremenu.

Razlikuje se od drugih NoSQL baza podataka zbog svoje primarne svrhe - poboljšanja performansi aplikacije [34]. Za razliku od baza podataka poput Cassandre, nudi set jednostavnih naredbi za upite i manipulaciju podataka, stoga nije dizajnirana za kompleksne upite.

MongoDB

MongoDB je NoSQL baza podataka koja, za razliku od baza podataka koje koriste tablice i redove za spremanje i procesiranje podataka, koristi fleksibilne dokumente te nudi elastičan model za spremanje podataka u različitim oblicima. MongoDB pruža korisnicima visoku skalabilnost za višeplatformske aplikacije i usluge. Najosnovnija jedinica podataka korištena u MongoDB-u je dokument koji se nalazi u zbirci dokumenata (collection), a ti dokumenti su formatirani u JSON (JavaScript Object Notation) formatu koji je sposoban za spremanje različitih tipova podataka.

Shema koju MongoDB koristi je dinamična te korisnicima pruža fleksibilnost u stvaranju podatkovnih zapisa, izgradnji kompleksnih upita i analizi velikih količina podataka. Za razliku od Cassandre, koja zadržava tradicionalnu strukturu kroz tablice, redove i stupce, MongoDB je mnogo fleksibilniji što se tiče strukture baze podataka [35]. Također postoje značajne razlike u sintaksi između CQL-a i MQL-a, pri čemu je CQL blizak SQL-u, dok MongoDB koristi sintaksu koja je inspirirana JavaScript jezikom i kompatibilna s JSON dokumentima [36].

HBase

HBase je NoSQL baza podataka usmjerena na stupce, koja djeluje putem HDFS-a (Hadoop Distributed File System). HBase je posebno prikladan za upravljanje velikim količinama strukturiranih podataka s izvrsnim protokom čitanja i pisanja. To ga čini popularnim izborom za podatke vremenskih serija, zabilježavanje događaja i web aplikacije. HBase nema upitni jezik poput CQL-a. Podacima se obično pristupa putem API poziva niske razine ili drugih alata ugrađenih u HBase, kao što su Apache Avro, REST i Thrift [37].

Iako su Cassandra i HBase obje usmjerene na stupce, postoje neke razlike među njima. Za razliku od Cassandre, HBase koristi samo jedan stupac kao primarni ključ u tablicama te ne podržava stupce za klasteriranje. Također, HBase koristi Vođa-sluga arhitekturu umjesto Cassandrine peer-to-peer arhitekture. Dok čvorovi u Cassandri komuniciraju putem protokola ogovaranja, HBase koristi Zookeeper protokol, gdje je jedan čvor vođa, a ostali čvorovi primaju podatke od njega. Cassandra podržava i spremanje i upravljanje podacima, dok je HBase dizajniran samo za upravljanje podacima te se oslanja na HDFS za pohranu podataka [38].

Neo4j

Neo4j je grafički temeljena NoSQL baza podataka koja umjesto tablica i dokumenata sprema čvorove i veze među tim čvorovima unutar strukture grafikona. Ti grafikoni se mogu analizirati pomoću semantičkih upita kojima se identificiraju podatkovni uzorci među čvorovima. Osim čvorova, postoje i rubovi (edges) koji predstavljaju veze među čvorovima. Pomoću tih rubova, korisnicima je olakšano razumijevanje podataka, a svaki rub ima definirani početni i završni čvor, tip veze te smjer.

Neo4j se najčešće koristi kada je glavni zadatak otkrivanje veza među podacima koji se teško mogu otkriti pomoću relacijskih baza. Zbog toga Neo4j omogućuje korisnicima dublji uvid u podatke, posebno u slučajevima gdje postoji mnogo veza, interakcija, asocijacija ili ovisnosti između podataka, kao što su detekcija prijevara ili sustavi preporuka [39].

Sam Neo4j je izvorna grafički temeljena baza podataka, što znači da je potpuno osmišljena za rukovanje grafičkim podatkovnim strukturama. Ovo je u kontrastu s drugim grafički orijentiranim bazama podataka koje se oslanjaju na tehnologije drugih modela baza podataka i koje se suočavaju s problemima kod većih i povezanijih skupova podataka. Fokus na izvornoj grafičkoj bazi podataka i upotreba moćnog upitnog jezika pod nazivom Cypher čine Neo4j preferiranim izborom za mnoge primjene povezane s grafovima [40].

7. Zaključak

NoSQL baze podataka predstavljaju revolucionaran pristup upravljanju podacima u svijetu informacijske tehnologije. NoSQL omogućava skalabilnost, visoku dostupnost i fleksibilnost u obradi i pohrani podataka, što je postalo ključno u doba kada se generira ogromna količina podataka iz različitih izvora.

Tijekom rada smo upoznali različite vrste NoSQL baza podataka, kao što su ključ-vrijednost, dokumentne, obitelji stupaca i grafičke baze podataka, koje nude različite pristupe organizaciji podataka, što omogućava prilagodbu specifičnim potrebama aplikacija. Sve te vrste se do neke mjere razlikuju jedna od druge, a primjer u ovome radu bila je Cassandra, koja pokazuje kako NoSQL baze podataka mogu kombinirati fleksibilnost s donekle strukturiranom shemom, karakterističnom za tradicionalne baze podataka, pružajući efikasan način upravljanja ogromnim količinama podataka.

Unatoč prednostima, važno je prisjetiti se da smo u radu došli do zaključka da NoSQL nije univerzalno rješenje za sve scenarije. SQL baze podataka i dalje imaju svoje mjesto, posebno u situacijama gdje je potrebna duboka analiza podataka i kompleksni upiti. Također, prijelaz s tradicionalnih SQL baza podataka na NoSQL može biti izazovan i zahtijeva pažljivo planiranje.

Tijekom analize podataka sam isprobao razne načine na koje se može pristupiti podatkovnom skupu unutar Cassandra okruženja, te sam došao do zaključka da u analizi podataka SQL pruža mnogo intuitivniji pristup što se tiče dolaska do korisnih uvida u podatke. Podatke iz CSV-a sam morao konstantno kopirati u nove tablice zbog "Prvo upit" pristupa koji nije najjednostavniji za analizu. Zaključio sam da je SQL bazama podataka zahtjevan dio rada u samoj izgradnji strukture i planiranju baze podataka, dok je taj dio fleksibilan u CQL-u, ali je analiza mnogo zahtjevnija u CQL-u.

Vjerujem da je prednost CQL-a brzina unosa podataka i rad s podacima u stvarnom vremenu, koje je bitno konstantno prihvaćati bez straha od kvarova unutar sustava, no za podatke s kojima sam ja radio, SQL bi bila bolja opcija. Smatram da bi Cassandru potpunije istražio trebao bi raditi u okruženju u kojem s timom ljudi radim s konstantnim tokom podataka u stvarnom vremenu. Mnogo sam naučio o Cassandri i nadam se jednoga dana primijeniti svoje znanje na stvarnim projektima.

Budućnost Cassandre je vjerojatno u Cloud prostoru, te vjerujem da će se ići u smjeru unapređenja analitičkih mogućnosti baze podataka bilo kroz sam CQL ili kroz integriranje drugih alata s mogućnostima za analizom. Također, AI bi mogao imati veliku ulogu u analizi podataka jer pruža posebnu dimenziju uvida u podatke.

8. Literatura

[1] Hans-Petter Halvorsen, Introduction to Database Systems. Preuzeto 15.07.2023. sa https://www.halvorsen.blog/documents/tutorials/resources/Introduction%20to%20Database%20Systems.pdf

[2] Lucidchart, What is a Database Model. Preuzeto 15.07.2023. sa https://www.lucidchart.com/pages/database-diagram/database-models

[3] Gaurav Vaish, Getting Started with NoSQL. Preuzeto 15.07.2023. sa https://download.blackball.lv/data/library/Getting_Started_with_NoSQL_%282013%29.pdf

[4] Jesús Sánchez Cuadrado, A repository for scalable model management Preuzeto 15.07.2023. sa https://www.researchgate.net/profile/Jesus-Sanchez-Cuadrado/publication/257491810_A_repository_for_scalable_model_management/links/568baf0508ae051f9afc5857/A-repository-for-scalable-model-management.pdf

[5] Marko Švaljek, Cassandra Succinctly. Preuzeto 16.07.2023. sa https://www.scribd.com/document/385723591/Cassandra-Succinctly#

[6] Piotr Fulmański, NoSQL. Theory and examples. Preuzeto 16.07.2023. sa https://fulmanski.pl/tutorials/wp-content/data/doc/about/nosql_theory_and_examples.pdf

[7] Dan Sullivan, NoSQL for Mere Mortals. Preuzeto 20.07.2023. sa https://datubaze.files.wordpress.com/2021/03/nosql-for-mere-mortals.pdf

[8] Meenu Dave, SQL and NoSQL Databases. Preuzeto 20.07.2023. sa https://www.researchgate.net/profile/Meenu-Dave/publication/303856633_SQL_and_NoSQL_Databases/links/5758557f08ae9a9c954a7573/SQL-and-NoSQL-Databases.pdf

[9] Redis, Understanding Key-Value Databases. Preuzeto 21.07.2023. sa https://redis.com/nosql/key-value-databases/

[10] Redis, Document Databases. Preuzeto 22.07.2023. sa https://redis.com/nosql/document-databases/

[11] Databasetown, Wide Column Database (Use Cases, Example, Advantages & Disadvantages). Preuzeto 22.07.2023. sa https://databasetown.com/wide-column-database-use-cases/

[12] Neo4j, Concepts: NoSQL to Graph. Preuzeto 22.07.2023. sa https://neo4j.com/developer/graph-db-vs-nosql/

[13] Kristi L. Berg, Tom Seymour, Richa Goel, History Of Databases Preuzeto 25.07.2023. sa https://clutejournals.com/index.php/IJMIS/article/view/7587

[14] What is the CAP theorem? Preuzeto 25.07.2023. sa https://www.ibm.com/topics/cap-theorem

[15] Manoveg Saxena, NoSQL Databases- Analysis, Techniques, and Classification. Preuzeto 25.07.2023. sa https://www.researchgate.net/profile/V-Singh-10/publication/271513372_NoSQL_Databases-

_Analysis_Techniques_and_Classification/links/54e0ade70cf2953c22b597de/NoSQL-Databases-Analysis-Techniques-and-Classification.pdf

[16] Dan Kelly, A brief history of databases: From relational, to NoSQL, to distributed SQL

Preuzeto 25.07.2023. sa https://www.cockroachlabs.com/blog/history-of-databases-distributed-sql/

[17] Rich Edwards, NoSQL Use Cases: When to Use a Non-Relational Database. Preuzeto 25.07.2023. sa https://www.datastax.com/blog/sql-vs-nosql-whats-the-difference

[18] Avantika Monnappa, The Rise of NoSQL and Why it Should Matter to You. Preuzeto 26.07.2023. sa https://www.simplilearn.com/rise-of-nosql-and-why-it-should-matter-to-you-article

[19] SQL vs NoSQL: Differences, Databases, and Decisions. Preuzeto 26.07.2023. sa https://www.talend.com/resources/sql-vs-nosql/

[20] What is Apache Cassandra. Preuzeto 29.07.2023. sa https://cassandra.apache.org/_/index.html

[21] Scylla, History of the Cassandra Database. Preuzeto 29.07.2023. sa https://www.scylladb.com/learn/apache-cassandra/introduction-to-apache-cassandra/

[22] DB-Engines Ranking**.** Preuzeto 29.07.2023. sa https://db-engines.com/en/ranking

[23] Nishant Neeraj, Mastering Apache Cassandra. Preuzeto 30.07.2023. sa https://hoclaptrinhdanang.com/downloads/pdf/devops/Mastering-Apache-Cassandra-2nd-(2015).pdf

[24] Cassandra Documentation. Preuzeto 30.07.2023. sa https://cassandra.apache.org/doc/latest/cassandra/architecture/guarantees.html

[25] Simplilearn, Apache Cassandra Architecture From The Ground-Up. Preuzeto 01.08.2023. sa https://www.simplilearn.com/tutorials/big-data-tutorial/cassandra-architecture

[26] Datastax, Virtual nodes. Preuzeto 01.08.2023. sa https://docs.datastax.com/en/cassandra-oss/3.0/cassandra/architecture/archDataDistributeVnodesUsing.html

[27] Alex Sorokoumov. Learn How CommitLog Works in Apache Cassandra**.** Preuzeto 03.08.2023. sa https://cassandra.apache.org/_/blog/Learn-How-CommitLog-Works-in-Apache-Cassandra.html

[28] Datastax, How is data read? Preuzeto 04.08.2023. sa https://docs.datastax.com/en/cassandra-oss/3.x/cassandra/dml/dmlAboutReads.html

[29] Nader Medhat, Understand Database Replication. Preuzeto 04.08.2023. sa https://nadermedhatthoughts.medium.com/understand-database-replication-the-good-and-ugly-b26ea5f7c0cf

[30] Robbie Strickland, Cassandra 3.x High Availability. Preuzeto 06.08.2023. sa https://www.packtpub.com/product/cassandra-3x-high-availability-second-edition/9781786462107

[31] Aaron Boissonnault, Cassandra and Relational database schema comparison – Query vs relationship modeling. Preuzeto 06.08.2023. sa https://www.navisite.com/blog/cassandra-and-relational-database-schema-comparison-query-vs-relationship-modeling/

[32] Scylla, Cassandra Column Family. Preuzeto 06.08.2023. sa https://www.scylladb.com/glossary/cassandra-column-family/

[33] Datastax, Creating a materialized view. Preuzeto 06.08.2023. sa https://docs.datastax.com/en/cql-oss/3.x/cql/cql_using/useCreateMV.html

[34] IBM, What is Redis? Preuzeto 20.08.2023. sa https://www.ibm.com/topics/redis

[35] IBM, What is MongoDB. Preuzeto 20.08.2023. sa https://www.ibm.com/topics/mongodb?mhsrc=ibmsearch_a&mhq=mongodb

[36] NoSQL Databases. Preuzeto 20.08.2023. sa https://codeahoy.com/learn/dbctutorial/ch5/

[37] What is HBase? Preuzeto 20.08.2023. sa https://www.ibm.com/topics/hbase?mhq=hbase&mhsrc=ibmsearch_a

[38] Apeksha Mehta, HBase vs Cassandra: Which is Better of the Two NoSQL Databases? Preuzeto 21.08.2023. sa https://appinventiv.com/blog/hbase-vs-cassandra/#4

[39] Mrinal Singh Walia, A Comprehensive Guide on Neo4j. Preuzeto 21.08.2023. sa https://www.analyticsvidhya.com/blog/2022/01/a-comprehensive-guide-on-neo4j-graph-database/

[40] John Stegeman, Native vs. Non-Native Graph Database**.** Preuzeto 21.08.2023. sa https://neo4j.com/blog/native-vs-non-native-graph-technology/

9. Popis slika