SVEUČILIŠTE U SPLITU

PRIRODOSLOVNO-MATEMATIČKI FAKULTET

SPLIT

**Gabriela Zdilar**

**STUPČASTA BAZA PODATAKA U CASSANDRI**

PROJEKT IZ RASPODIJELJENIH I NERELACIJSKIH BAZA PODATAKA

Split, 2023.

1. Sadržaj

[1. Sadržaj ii](#_Toc137048831)

[2. Uvod 1](#_Toc137048832)

[3. Razvojna tehnologija 2](#_Toc137048833)

[3.1. STUPČASTE BAZE PODATAKA 2](#_Toc137048834)

[3.2. Okvir(Cassandra i DataStax DevCenter) 7](#_Toc137048835)

[3.2.1. INSTALACIJA CASSANDRE 8](#_Toc137048836)

[3.2.2. DataStax DevCenter 24](#_Toc137048837)

[4. Izrada projekta 25](#_Toc137048838)

[4.1. Jednoslojna arhitektura 25](#_Toc137048839)

[4.2. Baza podataka 25](#_Toc137048840)

[4.3. Baza hotel i rezervacije 26](#_Toc137048841)

[4.4. Tehničke upute 33](#_Toc137048842)

[4.5. Upiti 33](#_Toc137048843)

[5. Zaključak 41](#_Toc137048844)

[6. Popis literature 42](#_Toc137048845)

1. Uvod

U ovom dokumentu bit će opisana izrada stupčaste baze podataka u Cassandri koristeći alat DataStax DevCenter. Baza koju sam za tu priliku izradila je baza koja služi za rezervaciju termina restorana u kojoj je moguće vidjeti slobodne termine, recenzije, pregled različitih podataka iz različitih uglova gledanja.

Međutim,prije nego se posvetimo samoj bazi, obradit ćemo detaljno stupčaste baze podataka,zašto se koriste,koja je razlika u odnosu na relacijske baze podataka te zašto smo ih baš koristili za ovaj problem.

Nadalje, upoznat ćemo se s Cassandrom,postupkom instalacije te konkretno spomenutim alatom u kojem smo implementirali bazu.

Nakon toga, detaljno će biti opisani koraci izrade naše baze te testiranje na konkretnim podacima.

Nadam se da ću uspjeti napraviti sve zacrtano ,da će rezultati biti zadovoljavajući te da će rad biti koristan svima koji će ga čitati.

1. Razvojna tehnologija

Budući da se u ovom projektu nećemo samo fokusirati na konkretnu izradu baze podataka, već želimo proučiti stupčaste baze podataka,njihovu strukturu,logiku i svrhu korištenja, u ovom poglavlju ćemo ih detaljno proučiti.

Zatim se posvećujemo stupčastim bazama podataka konkretno u Cassandri-sustavu baze podataka koji ćemo koristiti te posebno okruženju DatastaxDevCenter u kojem sam radila bazu podataka. Bit će detaljno opisani postupci instalacije obaju alata te njihovo međusobno povezivanje budući da i to ,samo po sebi, nije nimalo jednostavno te će olakšati nekom zainteresiranom čitatelju i sam postupak instalacije ako se na nju odvaži.

* 1. STUPČASTE BAZE PODATAKA

U ovom potpoglavlju opisat ćemo što su stupčaste baze podataka,kada se koriste te ih usporediti s relacijskim bazama podataka koje se poznatije velikoj većini ljudi.

Budući da smo većinom do sada koristili relacijske baze podataka,prirodno se zapitati zašto bismo odjednom koristili druge ako su nam relacijske baze podataka davale dosta dobre rezultate. . Naravno,sve ovisi o tipu problema s kojim se susrećemo:u slučaju da je bitna sigurnost podataka(banke,bolnički sustav…) ili da imamo malu količinu podataka, koriste se relacijske baze podataka.

Međutim,u sustavima u kojima imamo veliku količinu podataka(takozvani Big Data) ,kao što su društvene mreže, bi relacijske baze podataka bile izrazito spore. Također, tu se ne radi o podacima kod kojih nam je bitna velika sigurnost(kao što su OIB,zdravstveni kartoni,stanje na računu), dapače,na društvenim mrežama dijelimo informacije o sebi. Upravo kod sustava s takvim zahtjevima se koriste stupčaste baze podataka.

Sada ću ukratko opisati osnovne razlike između relacijskih i stupčastih baza podataka što se tiče logike izrade baze podataka i osnovnih značajki/svojstava:

* Temeljni elementi relacijske baze podataka su entiteti(predstavljali su stvarni objekt koji postoji,opisani jednom ili eventualno više tablica). Međutim,temeljni elementi stupčaste baze podataka su upiti. Upiti su nekakva temeljna pitanja ili zahtjevi koje mora zadovoljavati naša baza podataka, informacije koje možemo dobiti iz naše baze. Upravo zato su oni prvo što moramo napraviti kada krećemo u izradu stupčaste baze podataka i na osnovu njih radimo model te unosimo kasnije podatke.
* Svaka relacijska baza podataka je prilikom izvođenja transakcija zadovoljavala četiri svojstva dobro poznata pod kraticom ACID: ATOMARNOST (ili se izvrši cijela transakcija ili se ne izvrši ništa,to jest ukoliko dođe do pogreške prilikom izvođenja transakcije neće se ništa promijeniti),KONZISTENTNOST(transakcije se uvijek izvode pod određenim pravilima,podaci će uvijek biti ispravno uneseni u bazu ,poštujući njezinu strukturu),IZOLACIJA(moguće je istovremeno izvršavati radnje,rezultati će biti kao da su se izvodile jedna za drugom) te DURABILNOST(izvršene transakcije ostaju trajno,bez obzira na kvarove ili slične neprilike).

S druge strane, stupčaste baze podataka zadovoljavaju CAP teorem koji kaže da je moguće osigurati da baza zadovoljava samo dva od sljedeća tri svojstva:KONZISTENTNOST(svi klijenti vide sve podatke u isto vrijeme,bez obzira na koji se čvor spajaju),DOSTUPNOST-AVAILABILITY(svi čvorovi vraćaju valjan odgovor za bilo koji upit,bez iznimke),PARTITION TOLERANCE(cluster mora nastaviti raditi bez obzira na broj kvarova/slomova mreže). O ovome će biti još riječi kasnije jer nismo uveli pojam clustera još uvijek. Također,ističem da Cassandra,u kojoj radim bazu, zadovoljava dostupnost i partition tolerance.

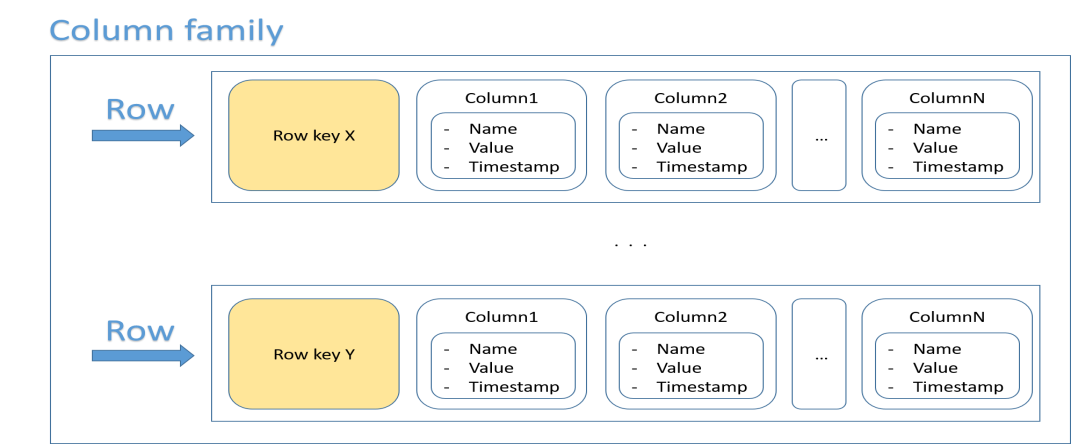
* Kod relacijskih baza podataka su primarni ključevi bili jedinstveni identifikatori retka,dok kod stupčastih baza podataka imaju više uloga:jedinstveni identifikatori retka,ali služe i za particiju i redoslijed podataka(detaljnije kad se budemo upoznavali s osnovnim pojmovima stupčastih baza podataka).
* Temeljna ideja relacijskih baza podatka je bila da se svaki podatak ponavlja jednom u bazi,koristili su se joinovi i povezivale tablice kako se podaci ne bi ponavljali te se taj proces nazivao normalizacija. Međutim, kod stupčastih baza podataka radimo suprotno – denormalizaciju ,odnosno podaci se ponavljaju kako bismo osigurali brzinu izvršavanja upita. Upravo zato su ovakve baze praktične za big dana podatke.

Dakle,jasno je da stupčaste baze podataka donose brzinu i lakši pregled podataka,ali su manje učinkovite kada treba unijeti novi podatak.

Zanimljivo je da se u ovim bazama podataka ne forsira referentna integrabilnost (kod relacijskih baza :ako se vrijednost jednog atributa referencira na vrijednost drugog,onda ta vrijednost mora postojati,strani ključevi),zahtjeva se čitanje prije pisanja.

Nakon što smo vidjeli osnovne razlike između ova dva tipa podataka,detaljnije ćemo se posvetiti stupčastim bazama podatka,odnosno njihovoj temeljnoj strukturi. Međutim,uvijek ćemo se vraćati i na relacijske baze podataka kako bismo usporedili stupčaste baze podataka s nečim korisnim i vidjeli čemu odgovara novi pojam kod dobro nam poznatih relacijskih baza.

Podaci se spremaju u stupce koristeći ključeve. Ti ključevi su povezani s vrijednostima,a vrijednosti su grupirane u OBITELJI STUPACA(column families). Obitelji stupaca se sastoje od parova ključ-vrijednost pri čemu se ključ preslikava u vrijednost koja je skup stupaca. Kod relacijskih baza bi obitelj stupaca bila tablica,a svaki par ključ-vrijednost bi bio redak.



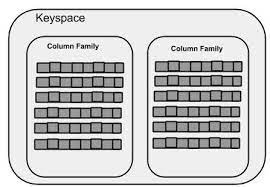
Slika :Obitelj stupaca

Svaki redak se sastoji od više obitelji stupaca i oznaka retka odgovara primarnom ključu.

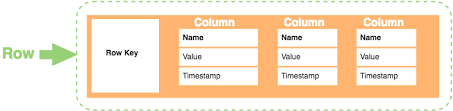
Svaki podatak je jedinstveno određen svojim retkom,nazivom stupca i vremenskom oznakom(stare vrijednosti se ne brišu).

Sada ćemo proučiti osnovne komponente stupčastih baza.

Krećemo strukture podataka najviše razine,a to je KEYSPACE. To je zapravo skup srodnih tablica. Kod relacijskih baza podataka bi odgovarao shemi.



Slika :Keyspace

Iduća važna komponenta je KLJUČ RETKA(row key) koji bi bio sličan primarnom ključu u relacijskim bazama podataka. On omogućava pristup elementima retka te se koristi i za redoslijed podataka(leksikografski redoslijed) i za PARTICIJU(baza podataka se logički dijeli na manje neovisne dijelove kako bi se smanjilo opterećenje baze podataka).

Slika :Ključ retka

Još jedna važna komponenta je STUPAC. To su zapravo nizovi bajtova koje se grupiraju u već spomenute obitelji stupaca. Svaki stupac sadrži naziv,vremensku oznaku i vrijednosti.

Struktura koja nije bila korištena u relacijskim bazama podataka je KLASTER(cluster).

To je zapravo skup poslužitelja koji zajedno rade.Klasteri omogućavaju skalabilnost,odnosno da se nesmetani rad nastavi i u slučaju povećanja broja podataka,primjerice da se neki server ne optereti dok su ostali slobodni.

Nadalje,COMMIT LOG je važna struktura koja sadrži popis svih transakcija,točnije sve naredbe su zapisane u log datoteku prije nego se izvrše kako bi bio moguć oporavak u slučaju nekih iznenadnih događaja.

BLOOMOV FILTER je prostorno učinkovita struktura podataka koji omogućava podacima dolazak do određenih stupaca te na taj način štedi vrijeme.Funkcionira tako da utvrdi da podaci definitivno nisu u datoteci ili da vjerojatno jesu(negativna pozitivna vjerojatnost).

BROJ REPLIKACIJA je također važan konfiguracijski parametar. Replikacija je zapravo proces stvaranja kopija baze i njihovo pohranjivanje na različitim lokacijama. Taj proces poboljšava dostupnost podataka.

Također je bitna i RAZINA KONZISTENCIJE koja je zapravo dinamička vrijednost koja kaže koliki je broj kopija(u klasteru) koje moraju potvrditi neku operaciju da bi ona bila uspješno izvršena.

To su ukratko osnovni pojmovi i objašnjenja vezani za stupčastu bazu podataka.

U idućem poglavlju ćemo ih sve vidjeti na konkretnom primjeru pa će biti jasniji.

* 1. Okvir(Cassandra i DataStax DevCenter)

Cassandra je distribuirana baza podataka koju je razvio Apache . To je NoSQL baza podatka kojoj su prednosti skalabilnost,mogućnost upravljanja velikim strukturama podataka koristeći mnogo poslužitelja te otpornost na pogreške i brzo dohvaćanje podataka. Upravo su to neki od razloga zašto ju koriste tvrtke kao što su Facebook,Netflix,Twitter…

Ona je ključ -vrijednost i stupac-orijentirana baza podataka.

Tehničke osobine koje su popularizirale Cassandru su:

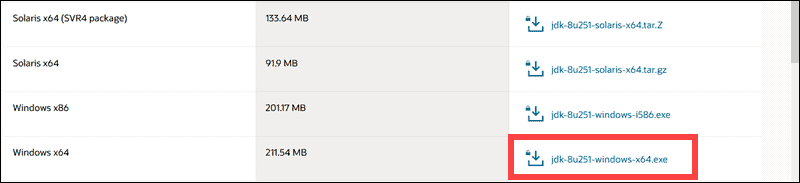
* PODRŽAVANJE TRANSAKCIJA
* SKALABILNOST-dodavanje hardvera za bolje zadovoljavanje potreba korisnika
* ELASTIČNO SKLADIŠTENJE PODATAKA-dinamički se prilagođava promjenama strukture podataka radi korisnikovih potreba
* BRZO PISANJE-može pohraniti veliku količinu podataka bez smanjenja učinkovitosti čitanja podataka
* BRZE PERFORMANSE-brzo vrijeme odziva

Budući da bazu radimo u Cassandri,istaknut ćemo osnovne tipove podataka:

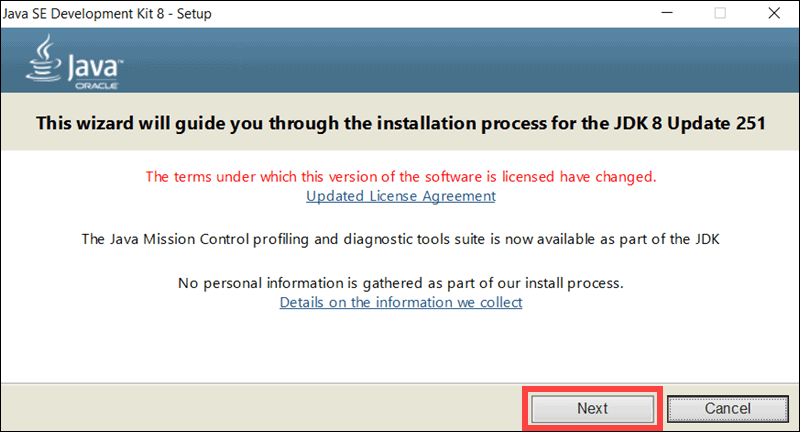
* BOOLEAN-dva stanja:istina ili laž
* COUNTER-brojač stupaca
* INT-cijeli broj(32 bita)
* DOUBLE-decimalni broj(dvostruka preciznost,64 bita)
* FLOAT-decimalni broj(32 bita)
* INET-IP adresa
* TIMESTAMP-vremenska oznaka
* UUID-jedinstveni identifikator
* VARCHAR-kodiran niz znakova
* LISTA-kolekcija jednog ili više poredanih elemenata
* MAPA-kolekcija parova ključ-vrijednost
* SET-kolekcija jednog ili više elemenata
  + 1. INSTALACIJA CASSANDRE

Kao što sam već spomenula,postupak instalacije Cassamdre nije nimalo lagan, treba paziti na dosta detalje pa ću u ovom dijelu rada detaljno opisati korake pri instalaciji kako bih olakšala nekim čitateljima koji će možda pokušati samo to napraviti.

Budući da koristim Windowse, bit će opisan postupak instalacije Cassandre na Windowse,ali ću na kraju,u Literature staviti link i za Linuxe.

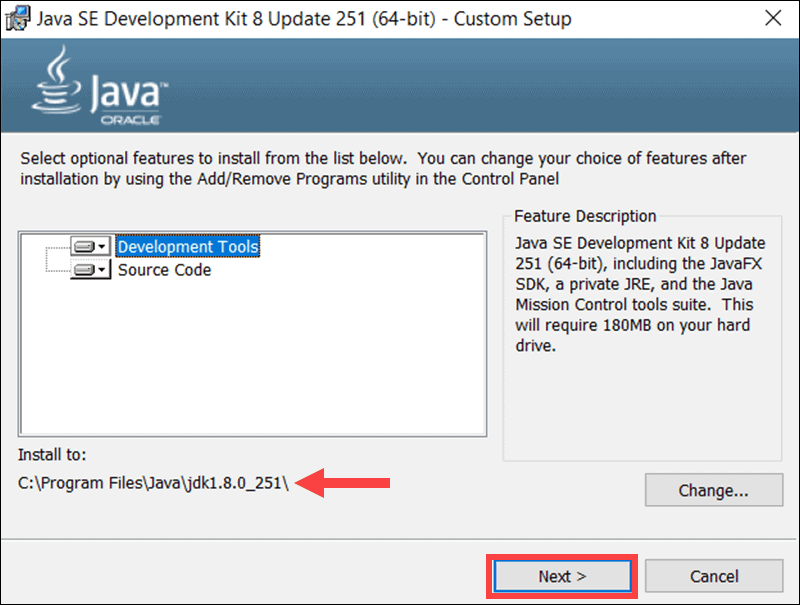
Prvi korak je instalirati Javu 8 na računalo jer ona sadrži sve potrebne alate za pokretanje aplikacije napisane u Javi. Dakle,potrebno je instalirati Oracle JDK 8 paket sa stranice <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>. Potrebno se registrirati na tu stranicu te preuzeti Java SE Development Kit 8u251 za Windows x64, kao što je prikazano na slici.

Slika :Java Development Kit

Nakon uspješnog preuzimanja, pokrenite preuzetu datoteku te pritisnite Next kao na slici:

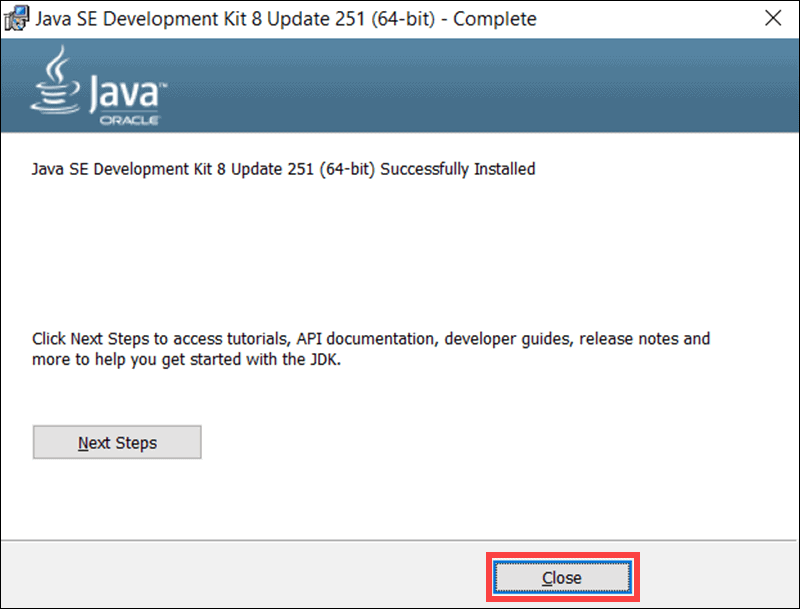
Slika :Next

Zatim,imate mogućnost birati performanse i opcije po želji te definirati putanju. Kad ste napravili željene promjene, ponovno kliknite Next:



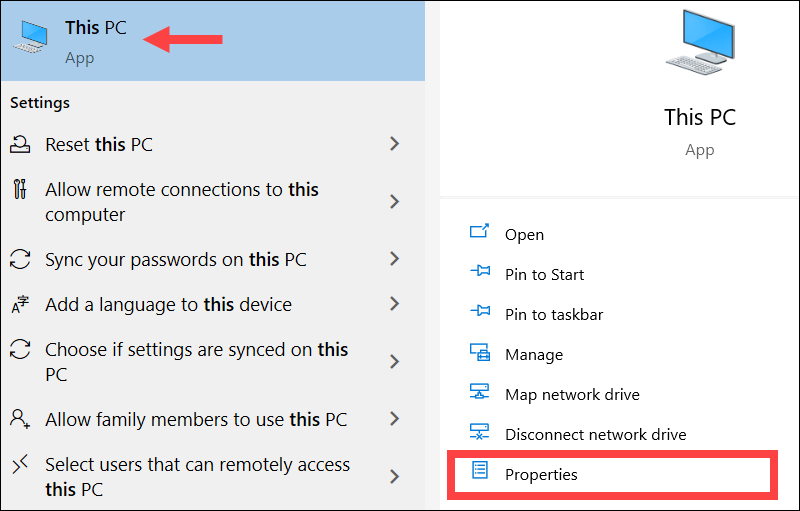
Slika :Putanja

Nakon ovoga slijedi proces instalacije(obično traje nekoliko minuta)..Pritisnite Close kad je proces završen.

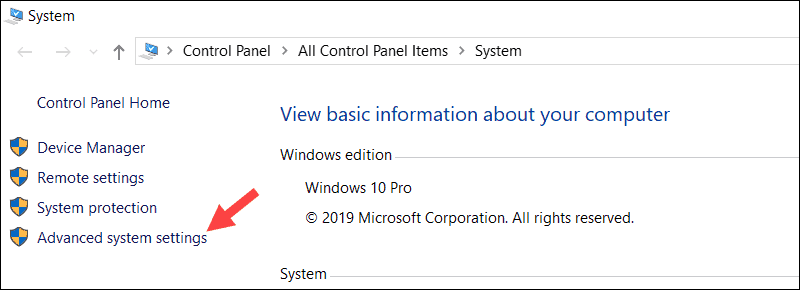


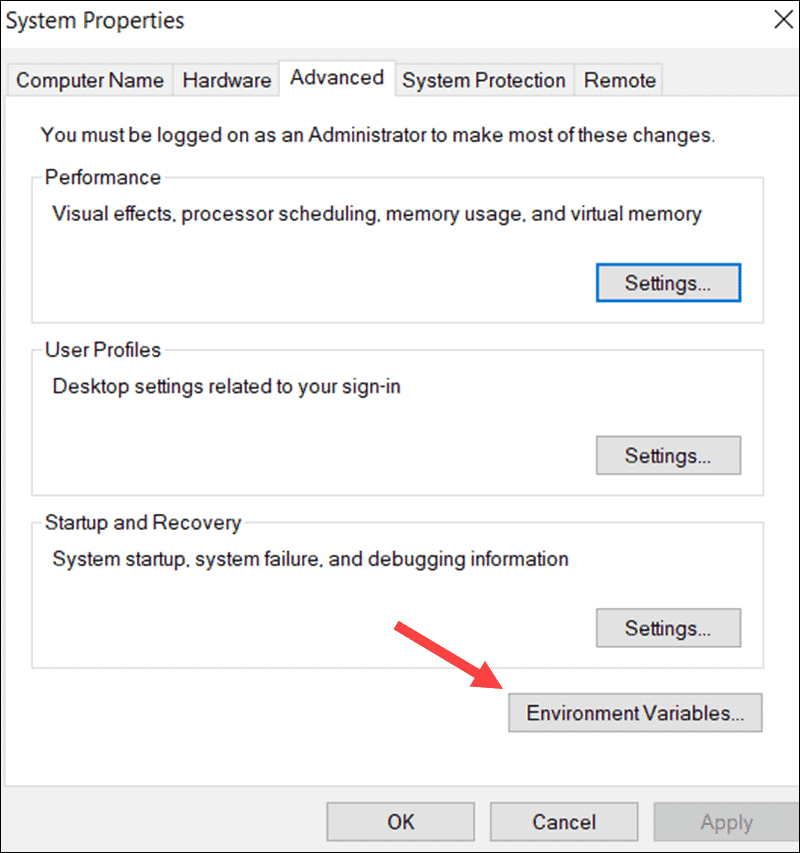
Slika :Instalacija

Idući važan korak je dodati varijable okruženja za instaliranu Javi 8.

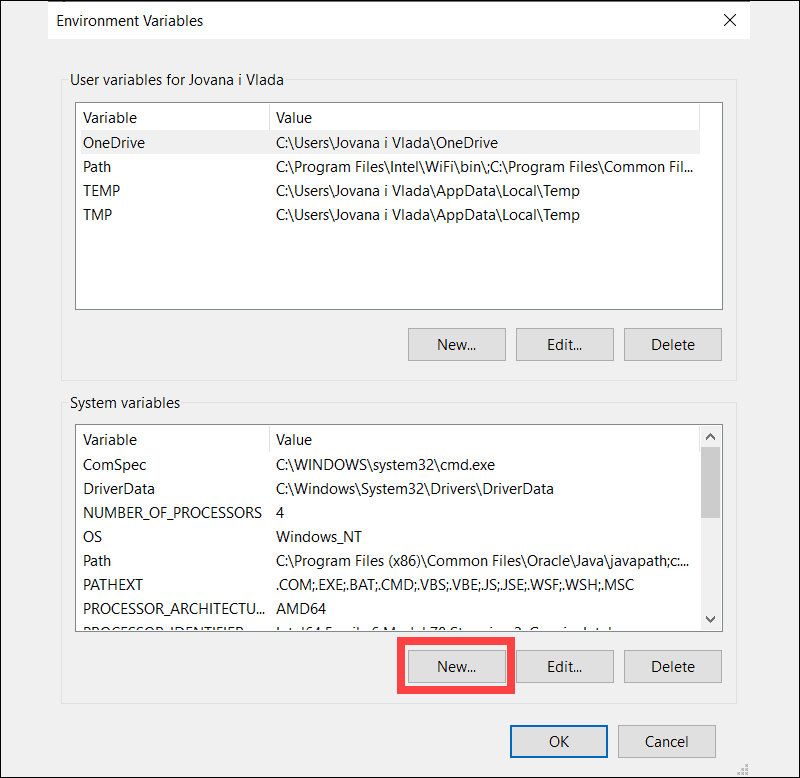


Slika :Svojstva



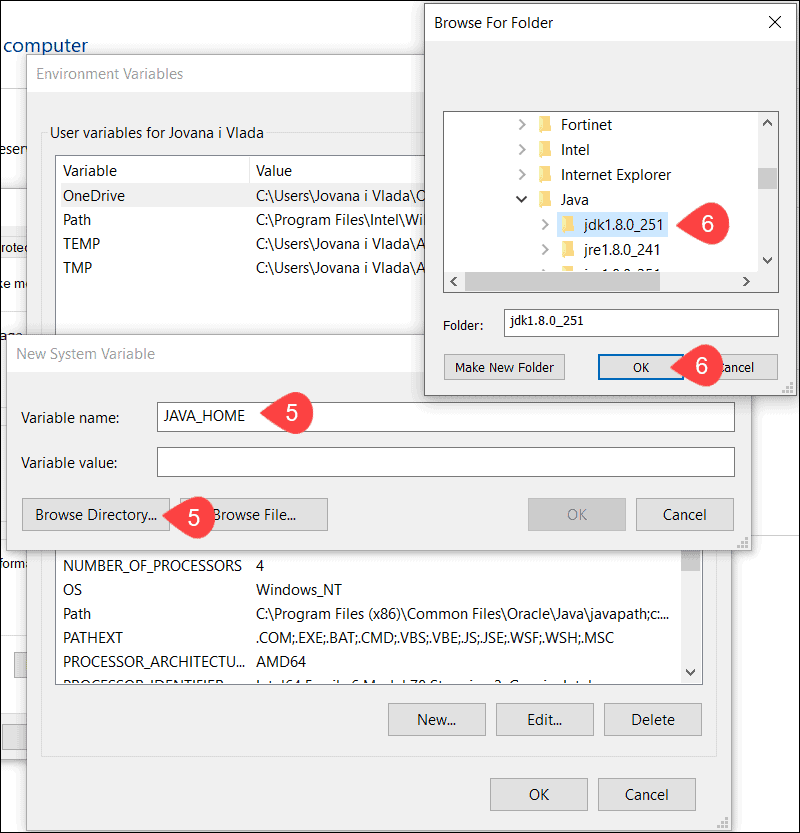
Slika :Napredne postavke sustava

Slika :Varijable okruženja

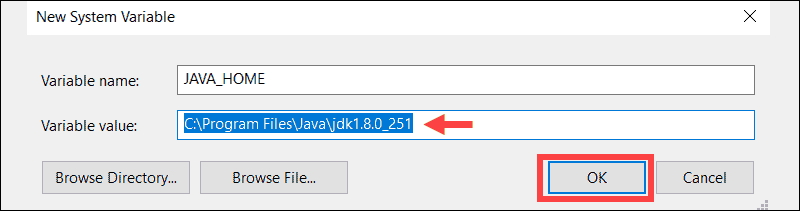
Slika :Varijable okruženja

Slika :dodavanje u putanju

Zatim dodajemo novo ime varijable JAVA\_HOME:

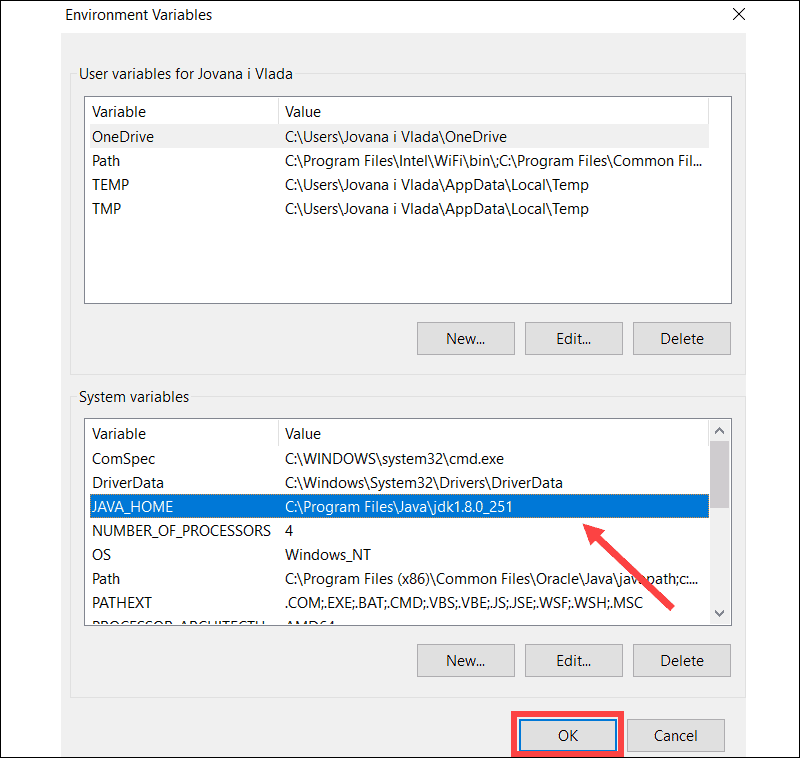


Slika :JAVA\_HOME

Zatim se „prebacimo“ do **This PC>Local Disk C:>Program Files> Java>jdk1.8.0\_251** i pritisnemo OK

Slika :JAVA\_HOME putanja

Pritisnemo OK za potvrđivanje procesa.

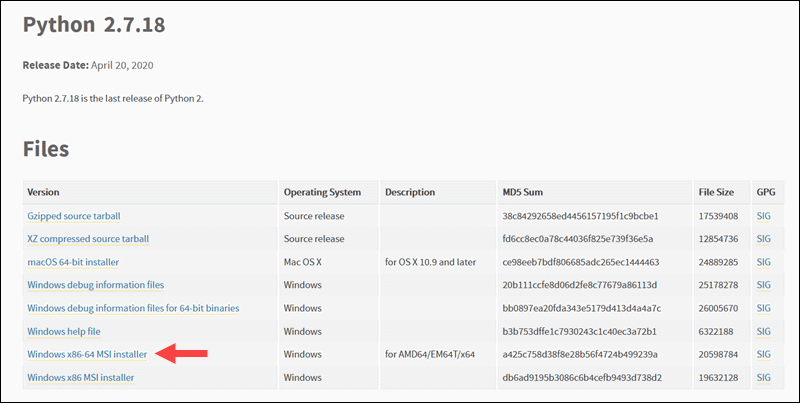


Slika :potvrda

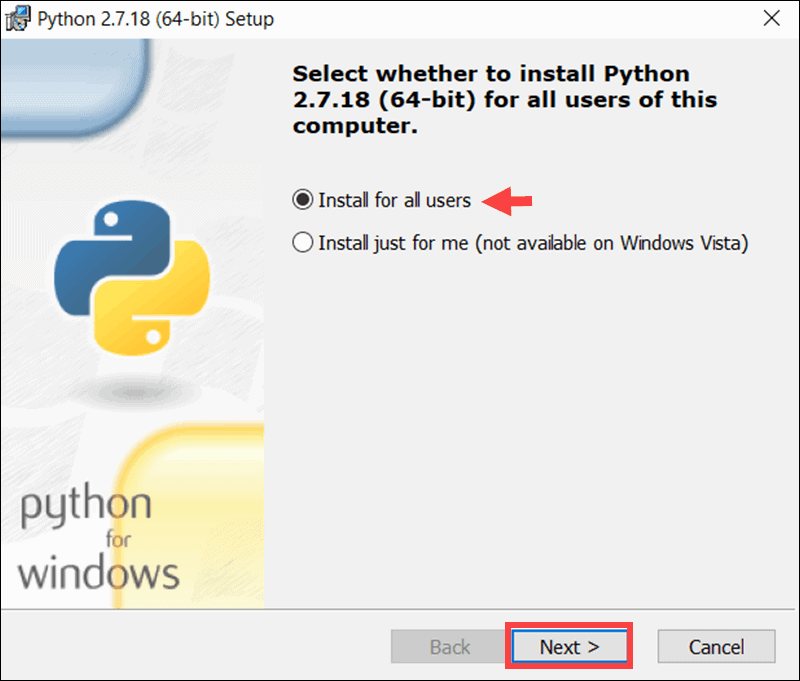
Ovim korakom je Java uspješno instalirana.

Idući korak je instalacija i konfiguracija Pythona 2.7,naime ta verzija Pythona podržava cqlsh koji omogućava komunikaciju Cassandre i korisnika.

Posjećujemo <https://www.python.org/downloads/release/python-2718/> te biramo verziju Windows x64:

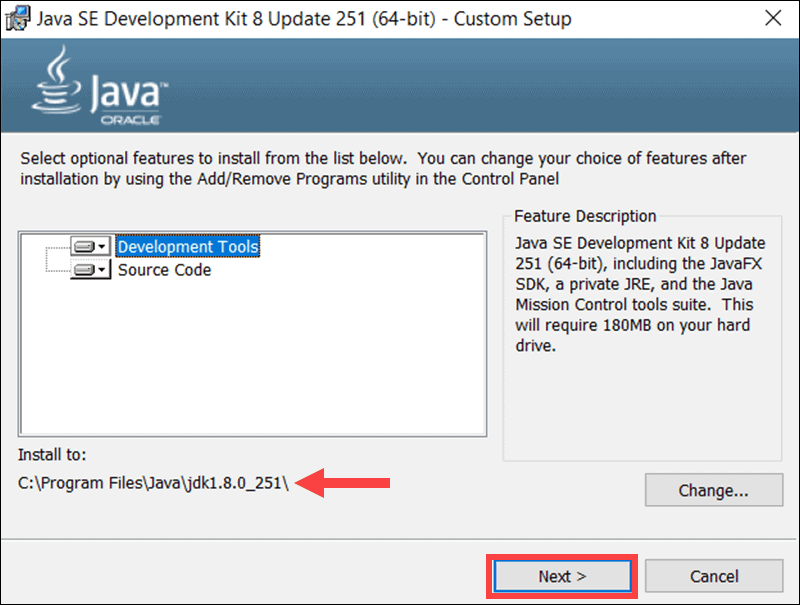


Slika :Instalacija Pythona 2.7

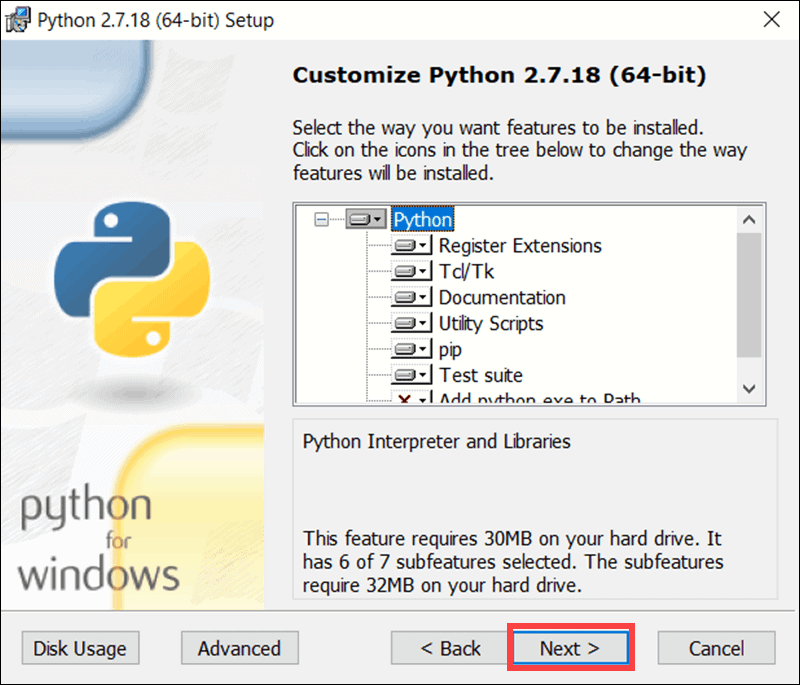


Slika :Next

.Nakon toga određujemo lokaciju na koju će se instalirati Python datoteka.

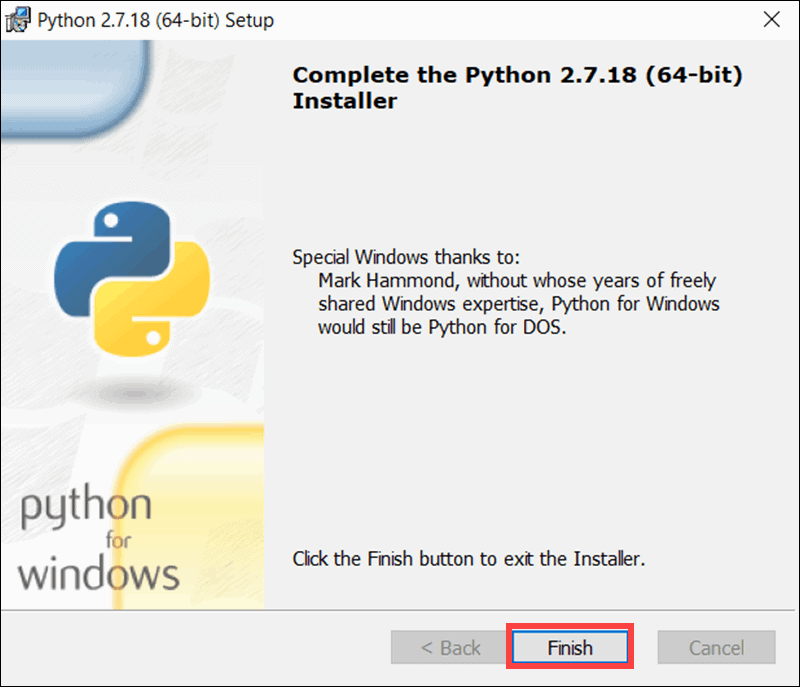


Slika :Python lokacija

Nastavljamo instalaciju s preporučenim postavkama.

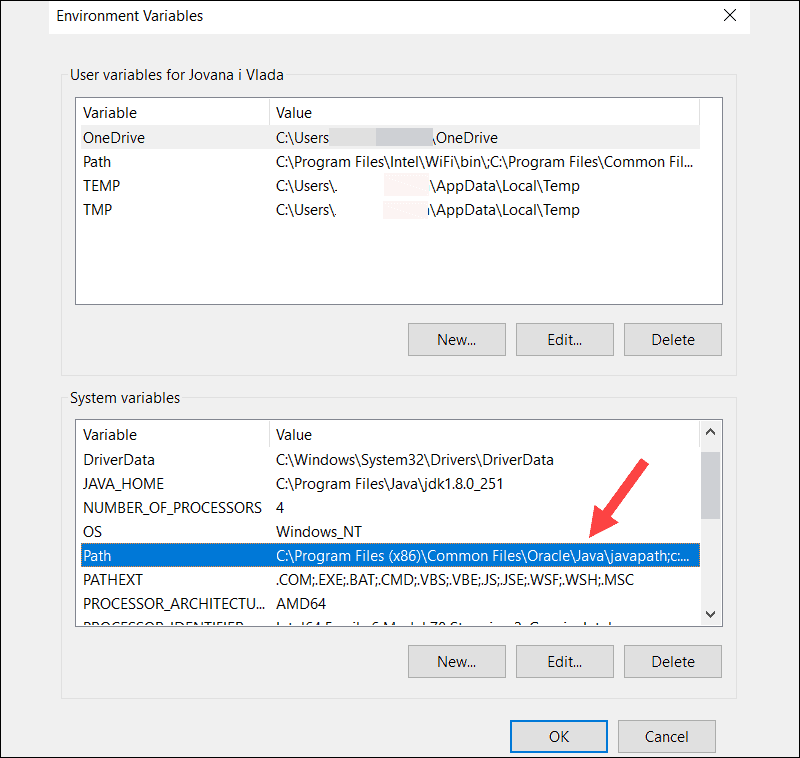
Slika :Postavke

Nakon toga instalacija traje nekoliko minuta.



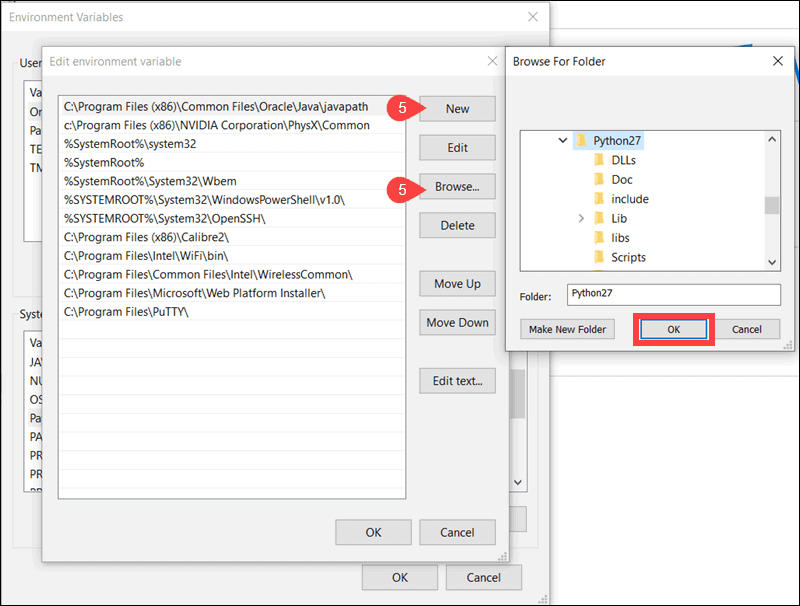
Slika :Finish

Ponovno,kao što smo radili i za Javu 8,dodajemo varijable okruženja te uđemo u postojeću putanju.



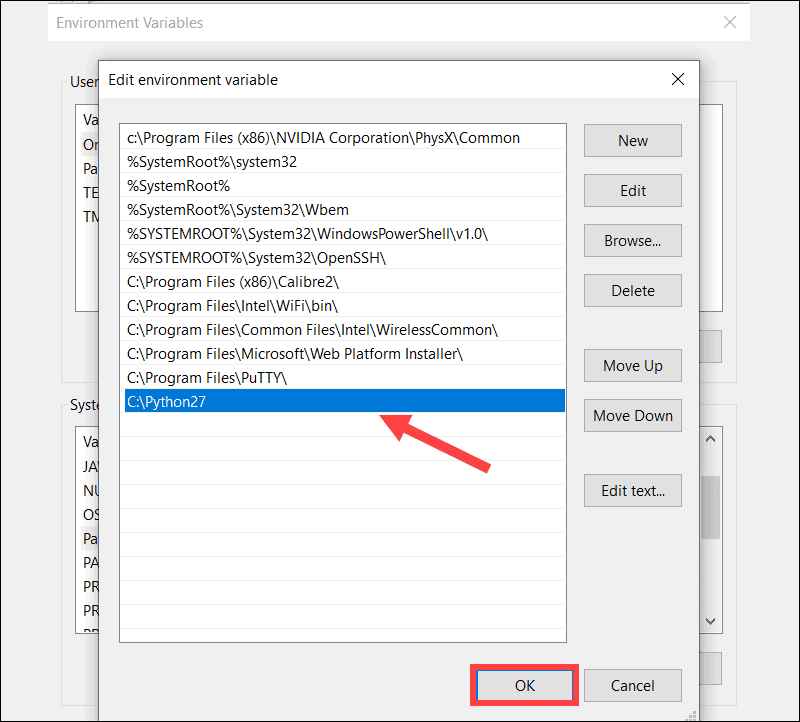
Slika :Postojeća putanja

Zatim dodajemo novu putanju,adresu Pythonove mape za instalaciju.



Slika :Dodaj Python putanju

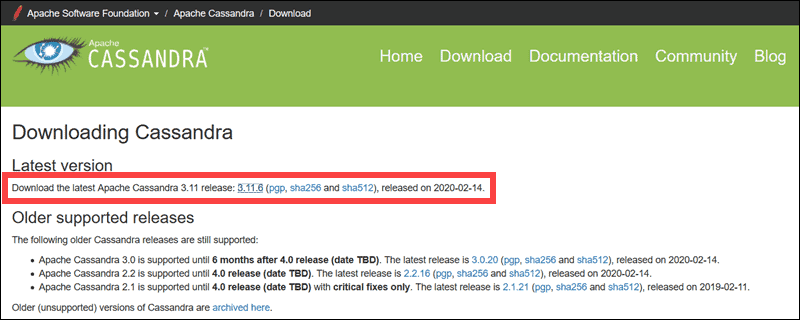
Pohranimo napravljene promjene.



Slika :Potvrdi promjene

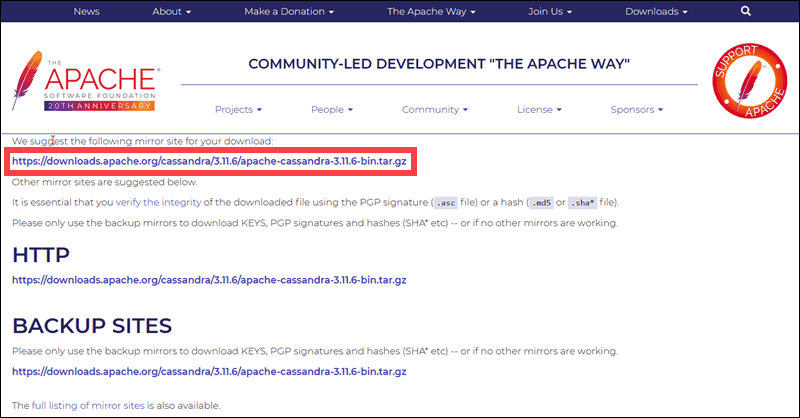
Sada je i Python uspješno instaliran.

Napokon su zadovoljene predispozicije za Cassandru te je idući korak njezina instalacija.

Prvo odaberemo željenu verziju za instalaciju na poveznici <https://cassandra.apache.org/download/>

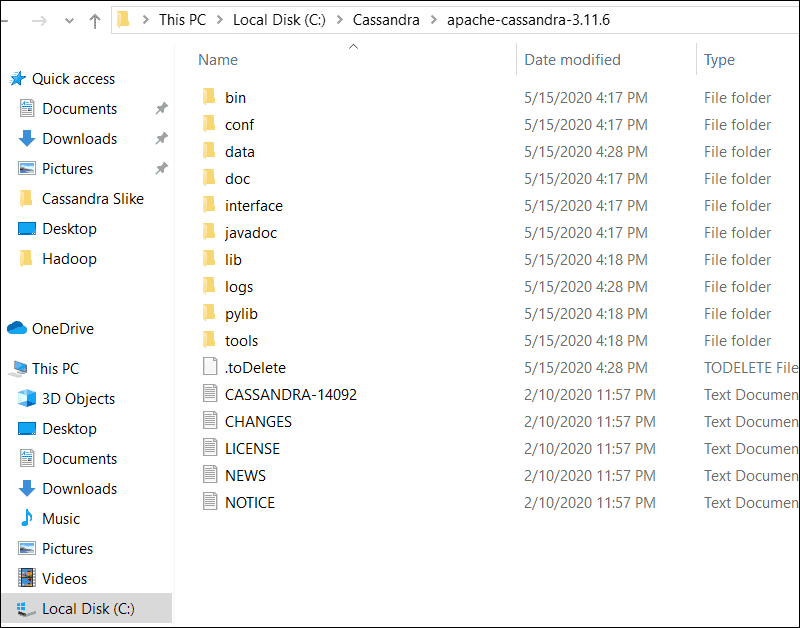
Slika :Preuzimanje Cassandre

Zatim započinjemo proces preuzimanja:

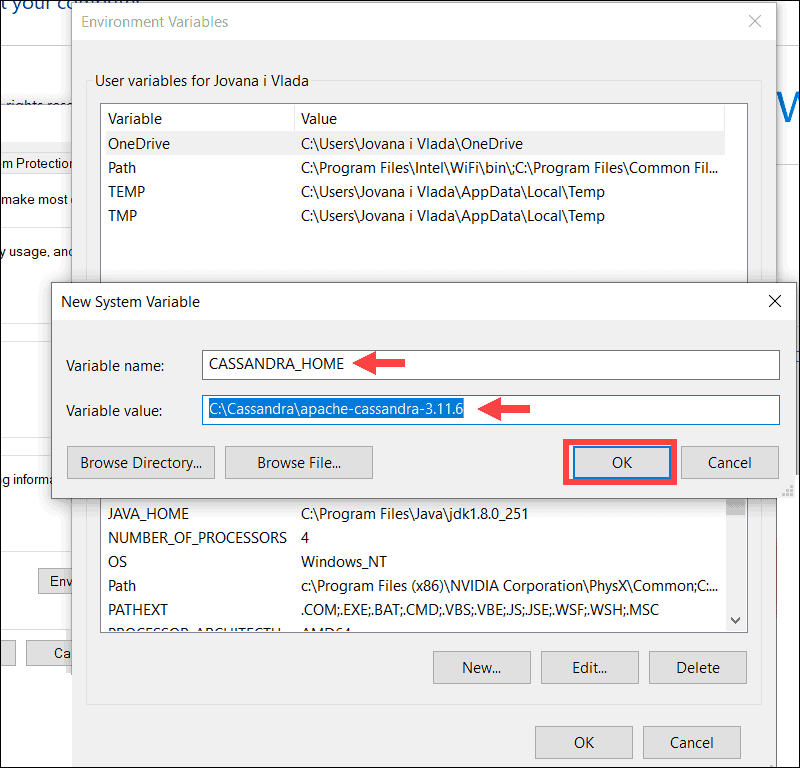


Slika :Preuzimanje Cassandre

Nakon toga otpakiramo tar.gz datoteku:

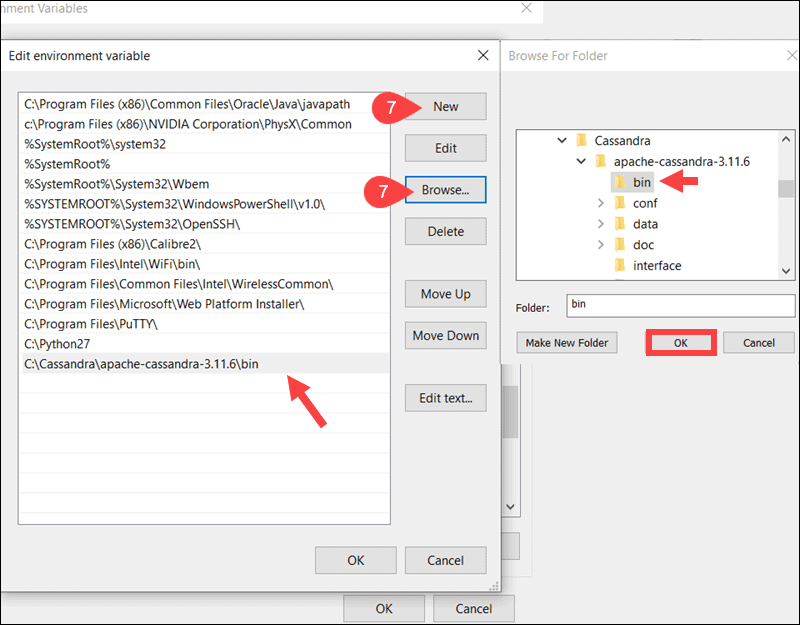


Slika :Otpakiranje

Ponovno je potrebno dodati Cassandru u varijable okruženja:

Slika :Casssandra-varijable okruženja

Sljedeći korak je dodavanje cijele putanje bin datoteke:

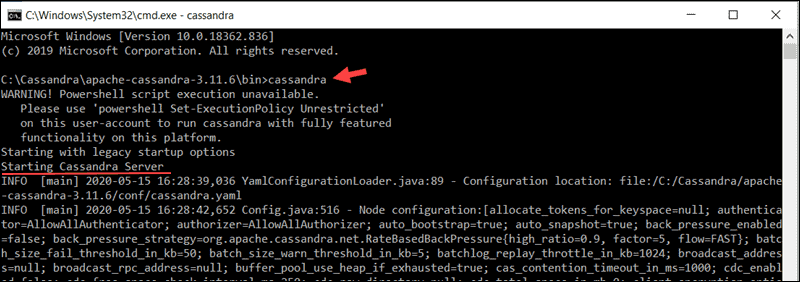


Slika :Putanja do bin datoteke

Pritiskom OK spremaju se promjene i Cassandra je uspješno instalirana.

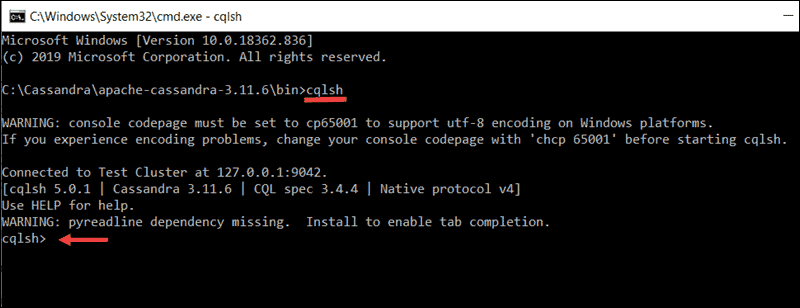
Sada je sve spremno za pokretanje Cassandre iz naredbenog retka.

Nakon toga upišemo naredbu *cassandra* u naredbenom retku:



Slika :cmd Cassandra

Zadnja stvar koju moramo napraviti je upisivanje naredbe *cqlsh* kojom dobivamo pristup do Cassandre.



Slika :cqlsh

Ovim korakom završena je instalacija Cassandre te se sada na računalu mogu izvršavati osnovne naredbe u Casasandri.

* + 1. DataStax DevCenter

DataStax DevCenter je alat koji sam odabrala za dizajn i implementaciju svog modela baze te za izgradnju upita.

Jako je praktičan alat,jednostavan za korištenje i rezultati se mogu lako vidjeti.

Alat automatski dopunjava naredbe te ističe sve pogreške u sintaksi napravljene za vrijeme pisanja upita i naredbi. Omogućava izvršavanje više CQL skripti i povezivanje s više klastera.

Također je moguće imati uvid u svoje upite zbog dobro razvijenog „traga upita“(eng. Query trace).

Alat je moguće besplatno preuzeti na sljedećoj poveznici <https://academy.datastax.com/downloads>

Važna napomena je da se mora ispravno spojiti na Cassandru ,a to radimo koristeći opciju **Connections** u izborniku.

1. Izrada projekta

Nakon što smo se upoznali sa stupčastim bazama podataka i instalirali sve potrebne alate u kojima ćemo raditi bazu, napokon dolazimo do najzanimljivijeg dijela:izrada baze.

U ovom poglavlju detaljno ćete biti upoznati s problemom,zahtjevima,modelom,upitima te načinom na koji smo to implementirali u instaliranim alatima.

* 1. Jednoslojna arhitektura

Prilikom izrade baze korištena je najjednostavnija jednoslojna arhitektura budući da je cilj bio upoznati stupčaste baze podataka i vidjeti na koji način se izrađuje takva baza podataka. Dakle,svi potrebni elementi se nalaze na jednom računalu te bilo koji zahtjev klijenta ne zahtjeva mrežnu vezu za izvođenje akcije na bazi podataka.

* 1. Baza podataka

U ovom poglavlju bit će detaljno opisana tema baze, proces izrade baze te krajnji rezultat.

Za početak će biti navedene osnovne razlike prilikom dizajniranja ovakvih baza i dobro nam poznatih relacijskih baza.

Prvo,kao što smo već spomenuli,nema joinova. Uobičajan način radi u Cassandri je denormalizacija,odnosno ponavljanje podataka po potrebi jer je ovdje cilj brzina budući da se u stvarnim sustavima radi s velikom količinom podataka.

Nadalje,ovdje nije uobičajena praksa stranih ključeva, već se često kombiniraju primarni ključevi potrebnih tablica tako da je taj podatak jedinstven sa svim komponentama(istaknut ću to na konkretnom primjeru u bazi).

Još jedna velika razlika je da je primarna uloga upita(query), naime kod relacijskih baza započinjemo s modelom na osnovu kojeg sve drugo radimo dok ovdje prvo pišemo upite. Oko upita su organizirani svi podaci,oni su nekakav kostur. Zapravo,želimo da naša baza daje odgovore na zahtjeve postavljene upitima.

Nadalje, tablice u Cassandri su dijelovi većih struktura-partitionsa koji su pohranjeni na različitim dijelovima diska pa je važno povezati sve ovisne tablice. Međutim,bitno je da njihov broj bude što je moguće manji.

Također,sortiranje je bilo jako bitno kod relacijskih baza dok je ovdje stvar dizajna i nema veliku važnost.

* 1. Baza hotel i rezervacije

Napokon dolazimo da izrade baze podataka. Moram napomenuti da sam model pronašla u jednom radu koji mi se svidio(ostavit ću link u Literaturi) jer na njemu mogu objasniti osnovne koncepte za izradu baze i opisati cjelokupan proces djelovanja.

Dakle,izradila sama bazu hotela u kojoj se nalaze hoteli, područja interesa koja su im blizu,sustav rezervacije po datumima,gostima,informacije o gostima…Iako nema dovoljno velik broj podataka koliki bi trebao biti za ovaj tip baza, ovo su najprikladnije baze za brzo pohranjivanje velikog broja podataka uz ne toliko bitnu sigurnost(u usporedbi s bolničkim sustavom na primjer).

Baza se temelji na sljedećim upitima:

PRVI UPIT:Pronaći hotele blizu područja interesa(kazališta,crkve,restorani,turističke atrakcije…).

DRUGI UPIT:Informacije o hotelu(ime,lokacija…)

TREĆI UPIT:Pronaći područja interesa u blizini odabranog hotela.

ČETVRTI UPIT:Pronaći dostupne sobe za određeni datum.

PETI UPIT:Pronaći stopu ugodnosti određene sobe,pogodnosti koje nudi.

ŠESTI UPIT:Pogledati rezervaciju koristeći broj koji korisnik dobije nakon rezervacije.

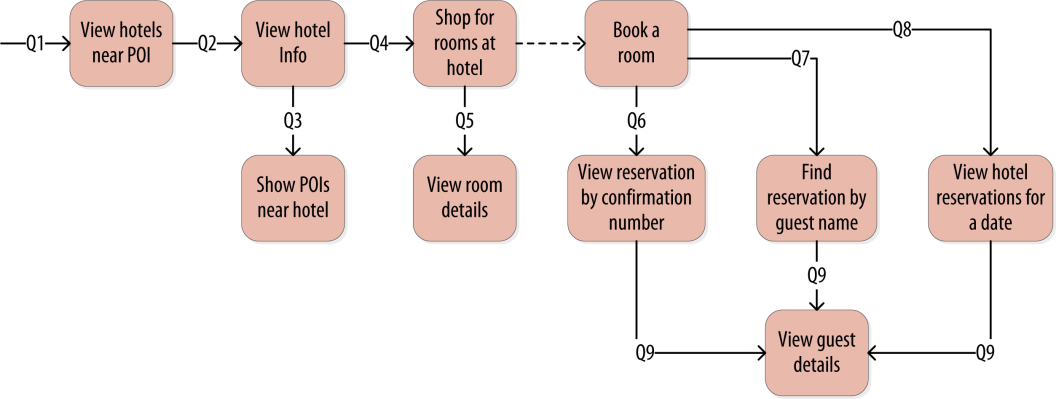
SEDMI UPIT:Pogledati rezervaciju po hotelu,datumu i imenu gosta.

OSMI UPIT:Pogledati rezervacije po imenu gosta.

DEVETI UPIT:Pogledati detalje gosta.

Iz samih upita nam je jasno da će se podaci ponavljati,treći i prvi upit su „obrnuti“,jednostavno bitno je željenu informaciju dobiti u što kraćem roku.

U spomenutom radu to je prikazano na sljedeći način:



Slika :Shema

Također napomenimo da ćemo stvoriti poseban tip podataka,adresu zbog praktičnih razloga.

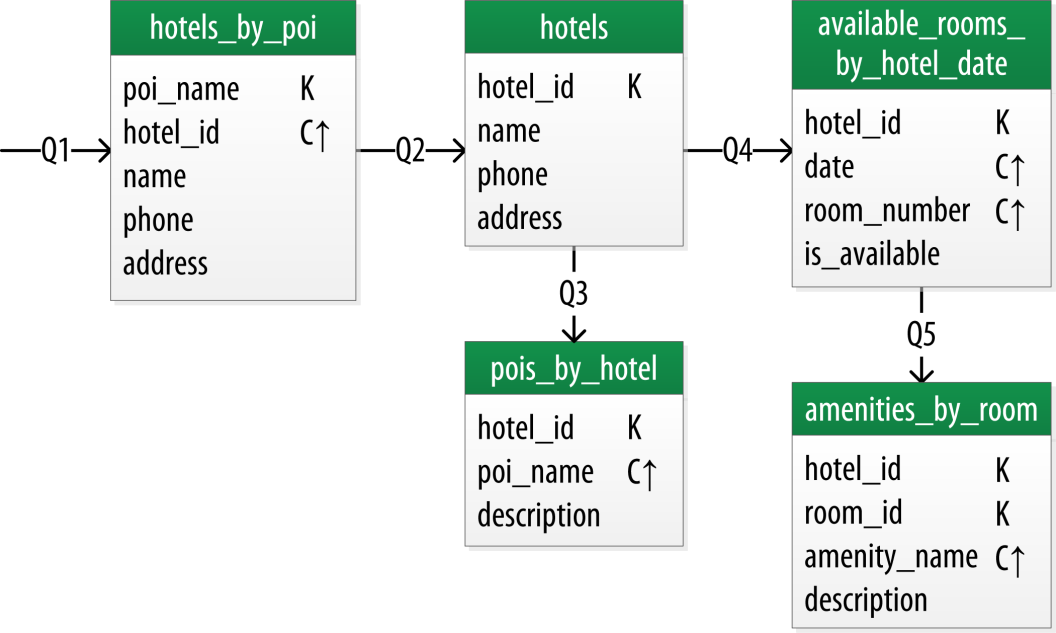
Nakon stvaranja upita,počinjemo dizajnirati tablice sa svim entitetima i vezama. Nakon toga određujemo primarne ključeve i partition ključeve kako bi se osigurala jedinstvenost. Zatim dodajemo dodatne atribute koji nam se nameću upitom, u slučaju da su neki od atributa jednaki za svaku instancu,označimo stupac sa STATIC.

U radu se koristi uobičajena notacija da se primarni ključ sastoji od dijelova označenih s K(stupci partition keya) i C za „clustering columns“.

Već sam spomenula da koristimo dva partitiona:to će biti hotel i rezervacija jer su to dvije prirodno odvojive cjeline.

S jedne strane,u hotelu će biti općenite,nepromjenjive informacije o hotelu,dok su u rezervaciji promjenjive(primjerice jedna soba danas može biti rezervirana,a sutra ne).

Partition hotel izgleda ovako:



Slika :Hotel

Sada će biti objašnjeno zašto to upravo ovako izgleda.

U prvom upitu tražimo hotele oko područja interesa pa se nameće da područje interesa bude primarni ključ. Ne koristimo nekakav ID područja interesa jer će ga korisnici pretraživati po imenu. Koristi se hotel\_id kao clustering column jer je moguće imati više hotela oko područja interesa.

Drugi upit zahtjeva informacije o hotelu pa se koristi isti id kao u prvoj tablici.

Treću upit je samo obratan od prvog,korisnici u različitim prilikama trebaju različite informacije.

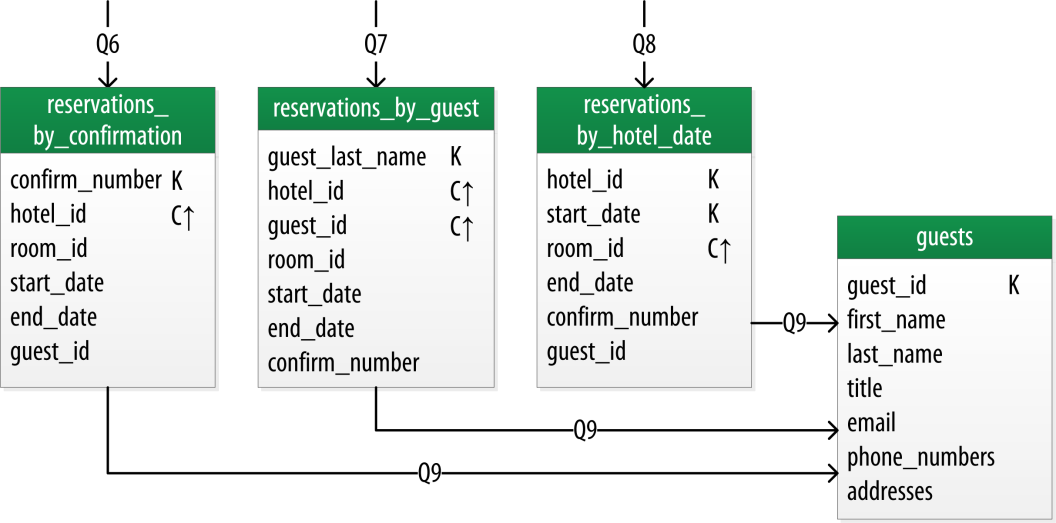
U četvrtom upitu tražimo dostupne sobe u određenom vremenskom periodu pa će primarni ključ ponovno biti hotel\_id(zbog brzine) i datum kao clusternig key jer se radi o većemo broju podataka.

Ovdje smo koristili „wide partition pattern“ jer smo grupirali povezane retke kako bismo omogućili brzo izvođenje podataka.

U petom upitu nema ništa novo,koristili smo logiku kao i u prethodnima.

Ovime je završen partition hotel.

Sad krećemo na partition reservation. Njegova shema izgleda ovako:



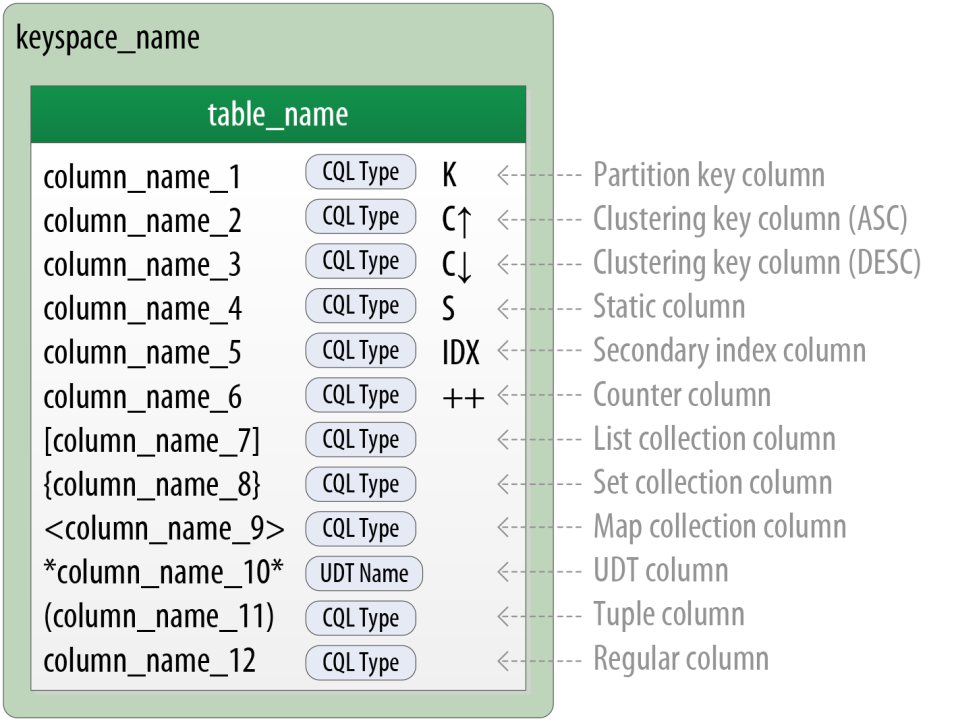
Slika :Reservation

Šesti upit nam omogućava da pogledamo rezervaciju po imenu gosta.

U sedmom upitu na neki način omogućavamo gostima da vide rezervacije pa se zbog praktičnosti koristi prezime,no zbog njegove nejedinstvenosti,dodaje se i njegov id.

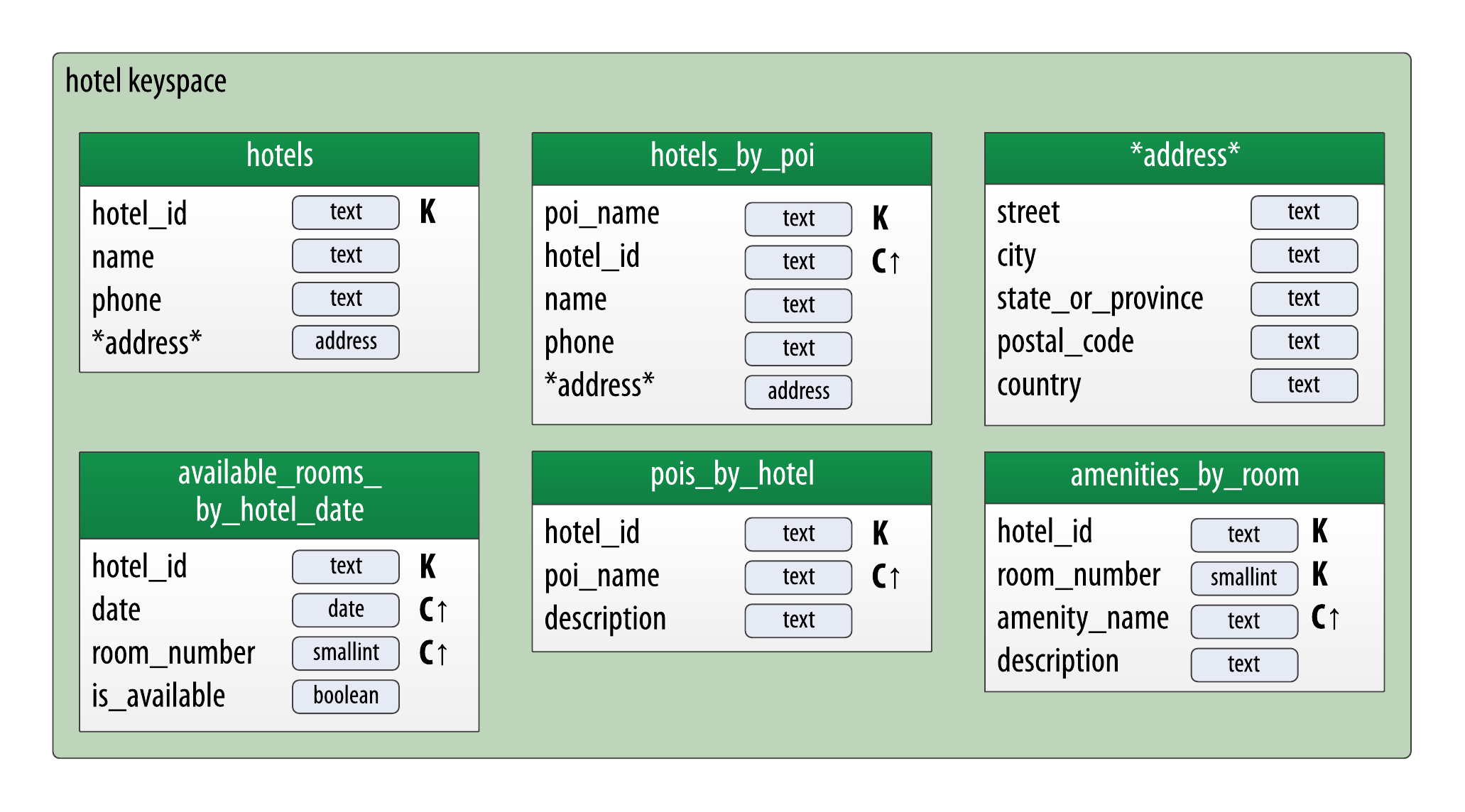
Osmi i deveti upit su prikladni možda za osoblje hotela da vide njima potrebne informacije.

Nakon definirane logičke strukture i prije nego vidimo fizički model,na sljedećoj slici će biti vidljive neke uobičajene oznake svojstvene Cassandri.



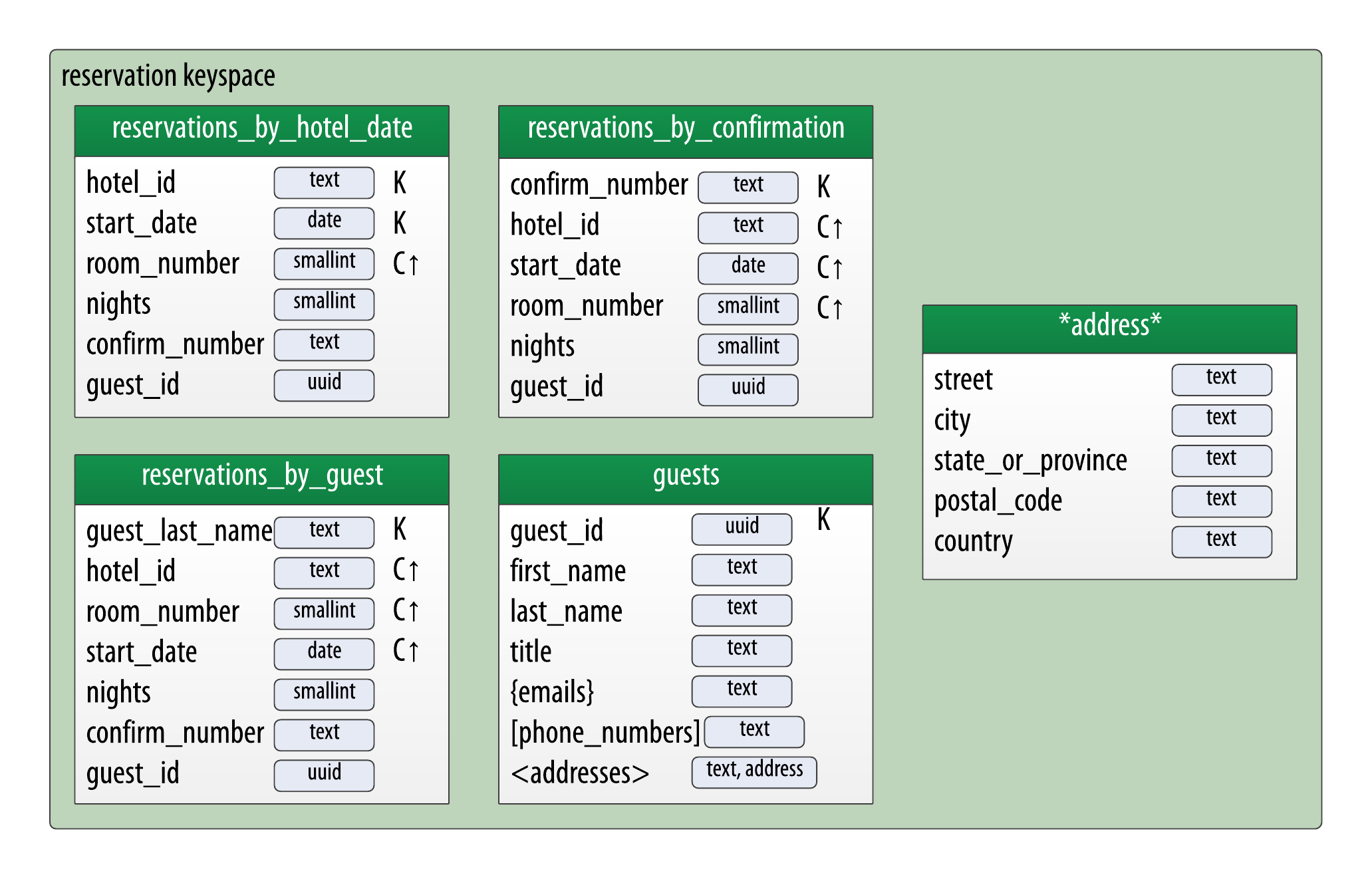
Slika :Oznake

Sada ćemo vidjeti psihičku strukturu modela za hotel



Slika :Hotel fizička shema

te za reservation:



Slika :Reservation fizička shema

Sada će biti napisana sama baza u Cassandri,nakon upita što smo i zašto radili:

CREATE KEYSPACE **hotel** WITH replication =

{'class': 'SimpleStrategy', 'replication\_factor' : 3};

CREATE TYPE **hotel**.**address** (

**street** text,

**city** text,

**state\_or\_province** text,

**postal\_code** text,

**country** text );

CREATE TABLE **hotel**.**hotels\_by\_poi** (

**poi\_name** text,

**hotel\_id** text,

**name** text,

**phone** text,

**address** frozen<**address**>,

PRIMARY KEY ((**poi\_name**), **hotel\_id**) )

WITH comment = 'Q1. Find hotels near given poi'

AND CLUSTERING ORDER BY (**hotel\_id** ASC) ;

CREATE TABLE **hotel**.**hotels** (

**hotel\_id** text PRIMARY KEY,

**name** text,

**phone** text,

**address** frozen<**address**>,

**pois** text )

WITH comment = 'Q2. Find information about a hotel';

CREATE TABLE **hotel**.**pois\_by\_hotel** (

**poi\_name** text,

**hotel\_id** text,

**description** text,

PRIMARY KEY ((**hotel\_id**), **poi\_name**) )

WITH comment = 'Q3. Find pois near a hotel';

CREATE TABLE **hotel**.**available\_rooms\_by\_hotel\_date** (

**hotel\_id** text,

**date** date,

**room\_number** smallint,

**is\_available** boolean,

PRIMARY KEY ((**hotel\_id**), **date**, **room\_number**) )

WITH comment = 'Q4. Find available rooms by hotel date';

CREATE TABLE **hotel**.**amenities\_by\_room** (

**hotel\_id** text,

**room\_number** smallint,

**amenity\_name** text,

**description** text,

PRIMARY KEY ((**hotel\_id**, **room\_number**), **amenity\_name**) )

WITH comment = 'Q5. Find amenities for a room';

CREATE KEYSPACE **reservation** WITH replication =

{'class':'SimpleStrategy', 'replication\_factor' : 3};

CREATE TYPE **reservation**.**address** (

**street** text,

**city** text,

**state\_or\_province** text,

**postal\_code** text,

**country** text );

CREATE TABLE **reservation**.**reservations\_by\_confirmation** (

**confirm\_number** text,

**hotel\_id** text,

**start\_date** date,

**end\_date** date,

**room\_number** smallint,

**guest\_id** uuid,

PRIMARY KEY (**confirm\_number**) )

WITH comment = 'Q6. Find reservations by confirmation number';

CREATE TABLE **reservation**.**reservations\_by\_hotel\_date** (

**hotel\_id** text,

**start\_date** date,

**end\_date** date,

**room\_number** smallint,

**confirm\_number** text,

**guest\_id** uuid,

PRIMARY KEY ((**hotel\_id**, **start\_date**), **room\_number**) )

WITH comment = 'Q7. Find reservations by hotel and date';

CREATE TABLE **reservation**.**reservations\_by\_guest** (

**guest\_last\_name** text,

**hotel\_id** text,

**start\_date** date,

**end\_date** date,

**room\_number** smallint,

**confirm\_number** text,

**guest\_id** uuid,

PRIMARY KEY ((**guest\_last\_name**), **hotel\_id**) )

WITH comment = 'Q8. Find reservations by guest name';

CREATE TABLE **reservation**.**guests** (

**guest\_id** uuid PRIMARY KEY,

**first\_name** text,

**last\_name** text,

**title** text,

**emails** text,

**phone\_number** text,

**address** frozen<**address**>,

**confirm\_number** text )

WITH comment = 'Q9. Find guest by ID';

* 1. Tehničke upute

Kao što je već spomenuto, radila sam u alatu Datastax DevCenter koji omogućuje jednostavno izvršavanje cql naredbi.

Baza i svi potrebni upiti su predani te se na jednostavan način mogu otvoriti i izvršiti u spomenutom alatu(koristimo naredbu Open te učitavamo predane dokumente u mapi upiti\_za\_bazu).Jedina bitna napomena da se prvo učita i pokrene *baza\_stvaranje.cql* kojom se stvaraju keyspaceovi i tablice,a ostali upiti služe za unos podataka te se omogućuje pregled nekih osnovnih upita na bazu.

Potrebno je samo voditi računa da se spojimo na Cassandru,kako je gore opisano te izaberemo ispravan connection.

Iako je alat jako praktičan,ima i neke nedostatke kao što su nemogućnost učitavanja csv podataka(ali zato je moguće podatke izvesti u csv formatu što sam i napravila).

Upravo zbog toga sam podatke ručno unosila u json formatu te nisam unijela jako velik broj podataka(nekoliko tisuća podataka) jer je trebalo paziti da svi podatci budu usklađeni kako bi se tablice povezale na ispravan način.

Upiti kojima sam ih unijela su dostupni među predanim dokumentima.

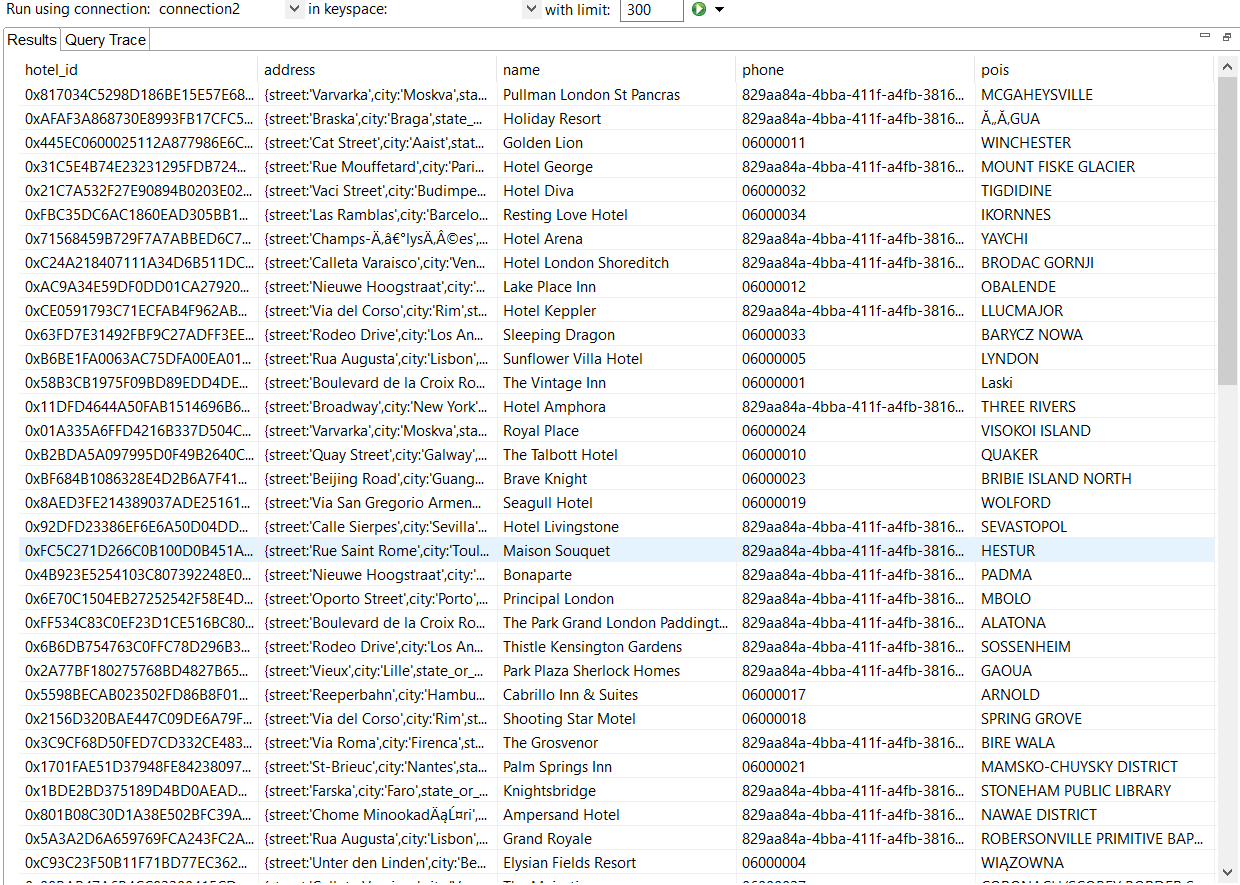
* 1. Upiti

Nakon napravljene baze,sad ćemo konačno vidjeti neke rezultate,odnosno upitima dobiti podatke.

U ovom odlomku će biti dostupan upit te rezultat u bazi kojeg daje.

1.UPIT

select\*from **hotel**.**hotels**;



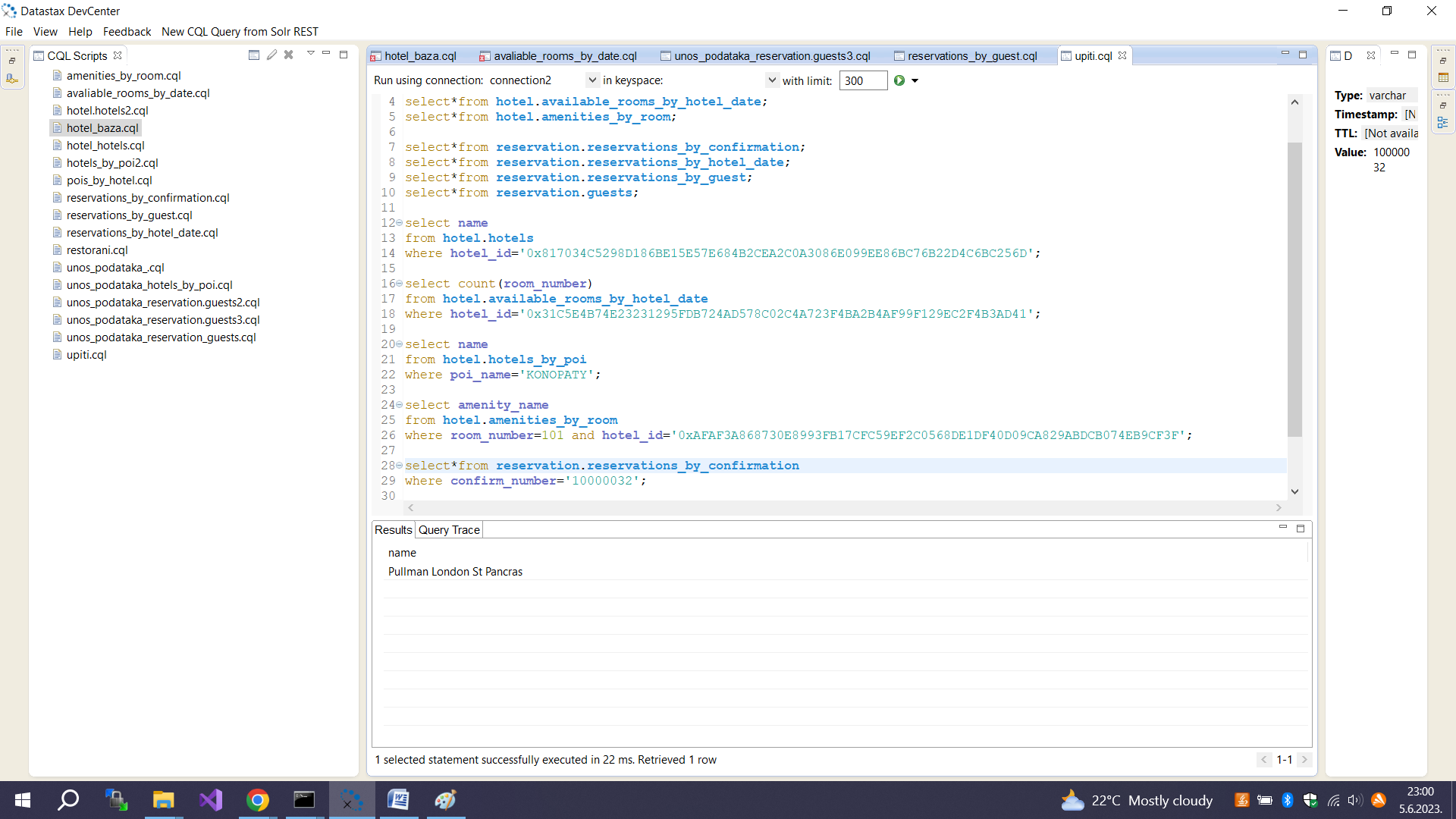
Slika :PRVI UPIT

2.UPIT

select **name**

from **hotel**.**hotels**

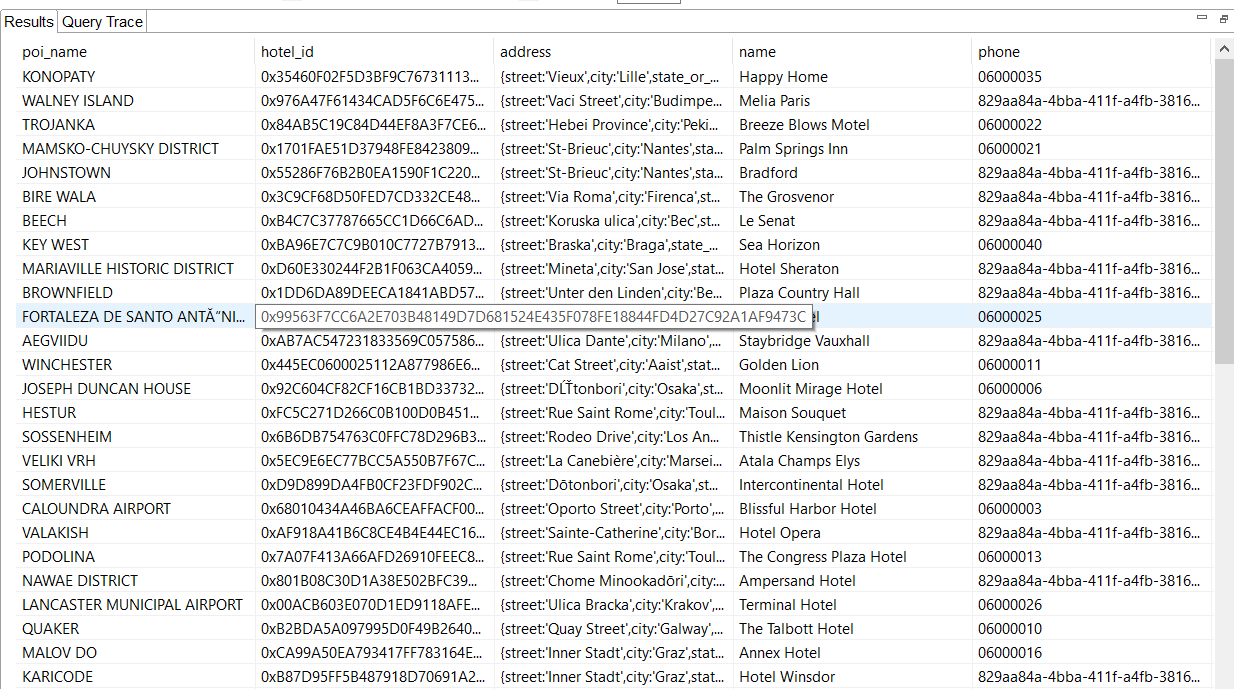
where **hotel\_id**='0x817034C5298D186BE15E57E684B2CEA2C0A3086E099EE86BC76B22D4C6BC256D'

****

Slika \_DRUGI UPIT

3.UPIT:

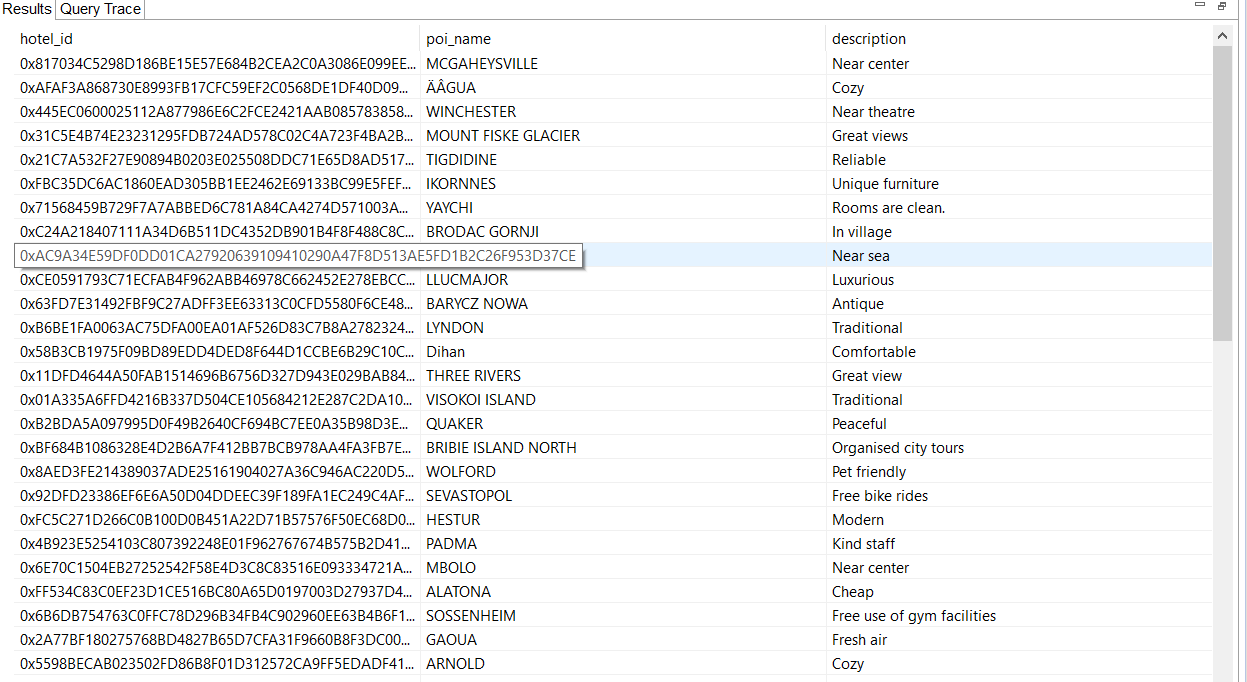
select\*from **hotel**.**hotels\_by\_poi**;



Slika :TREĆI UPIT

4.UPIT:

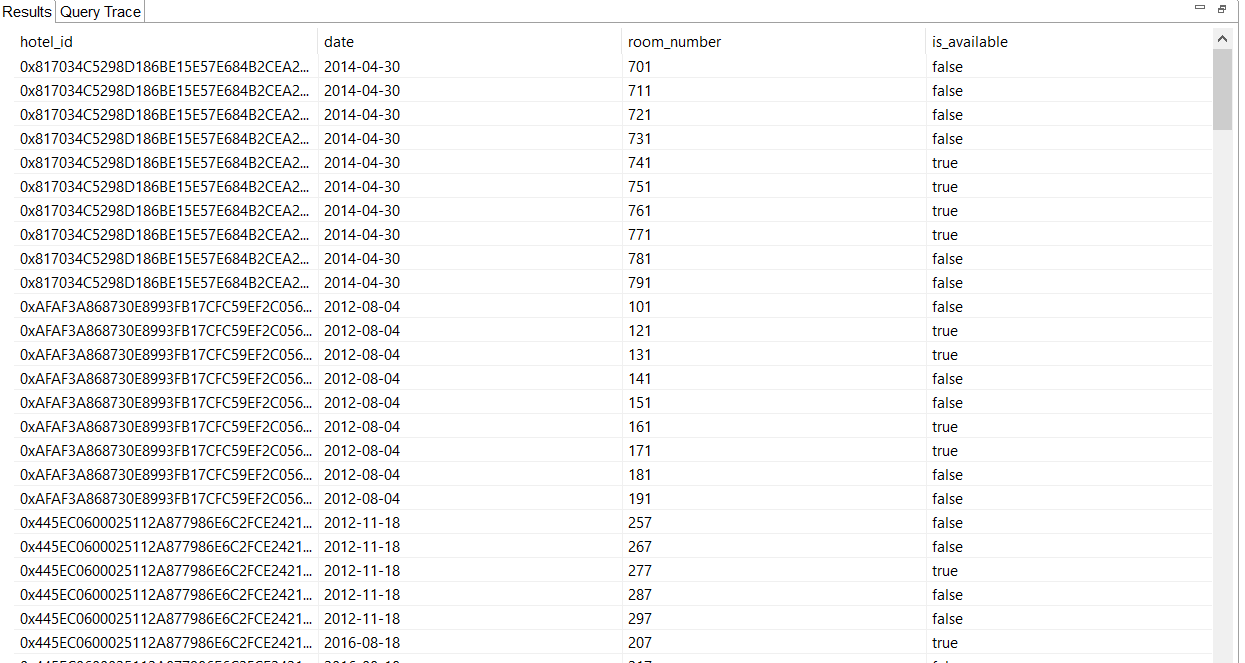
select\*from **hotel**.**pois\_by\_hotel**;



Slika :ČETVRTI UPIT

5.UPIT:

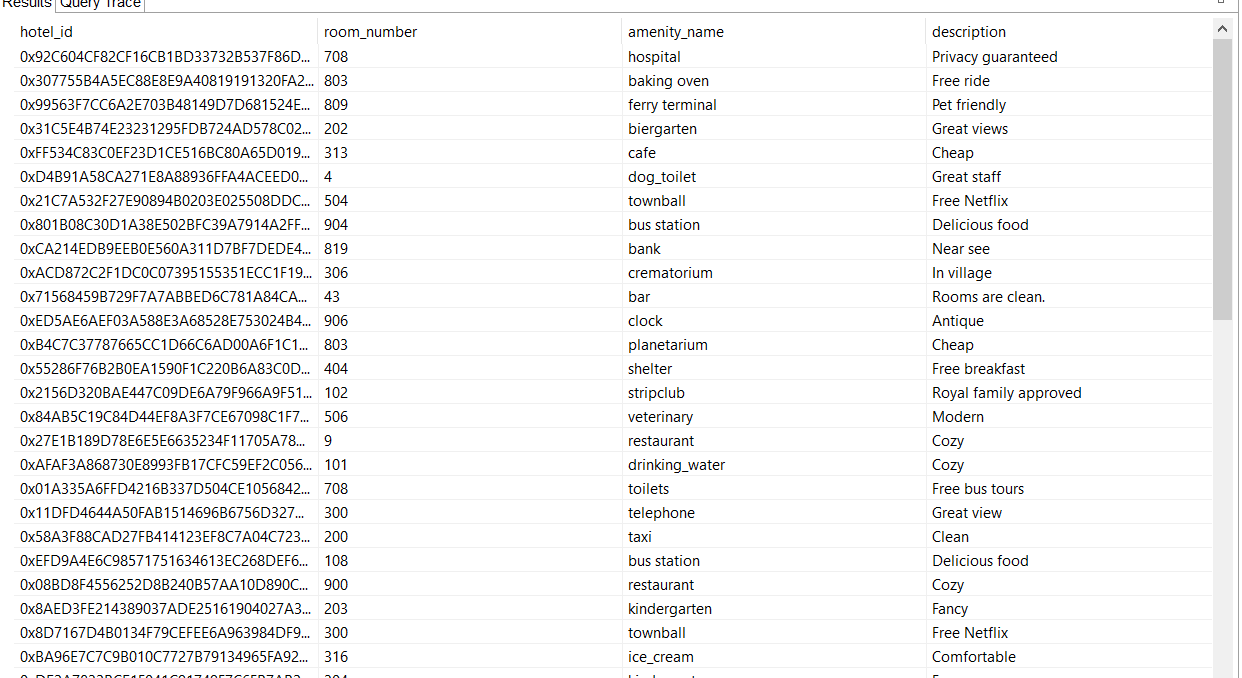
select\*from **hotel**.**available\_rooms\_by\_hotel\_date**;



Slika :PETI UPIT

6.UPIT:

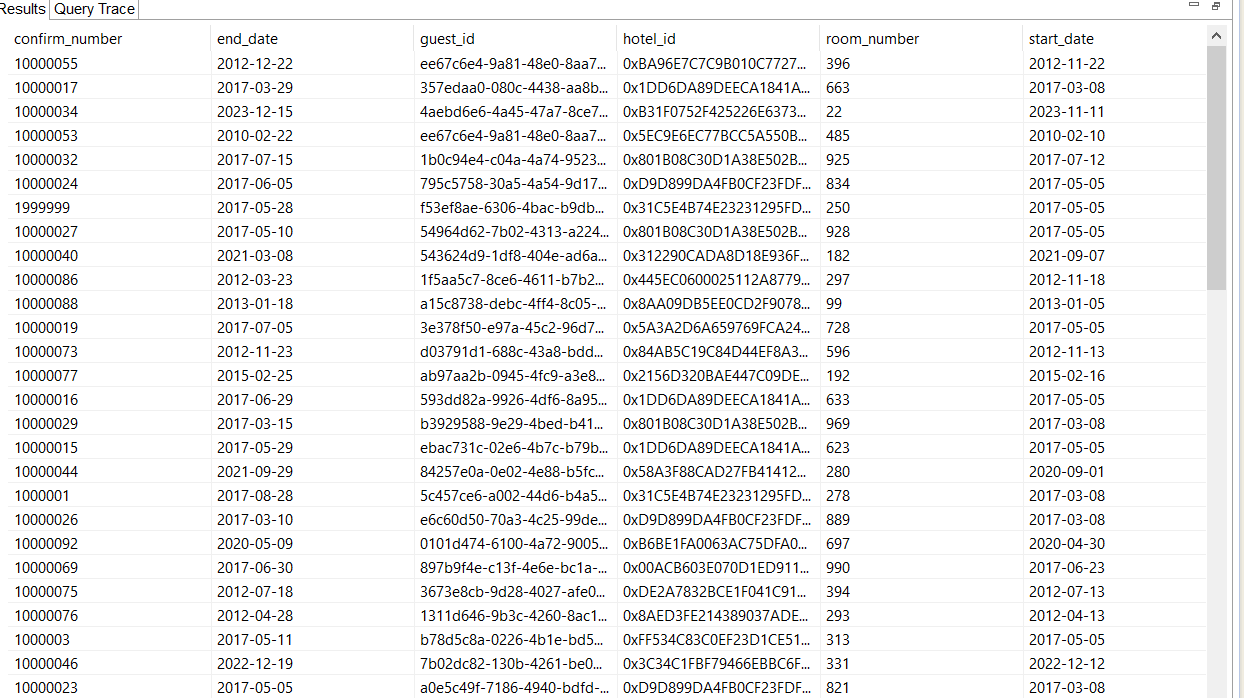
select\*from **hotel**.**amenities\_by\_room**;



Slika :ŠESTI UPIT

7.UPIT:

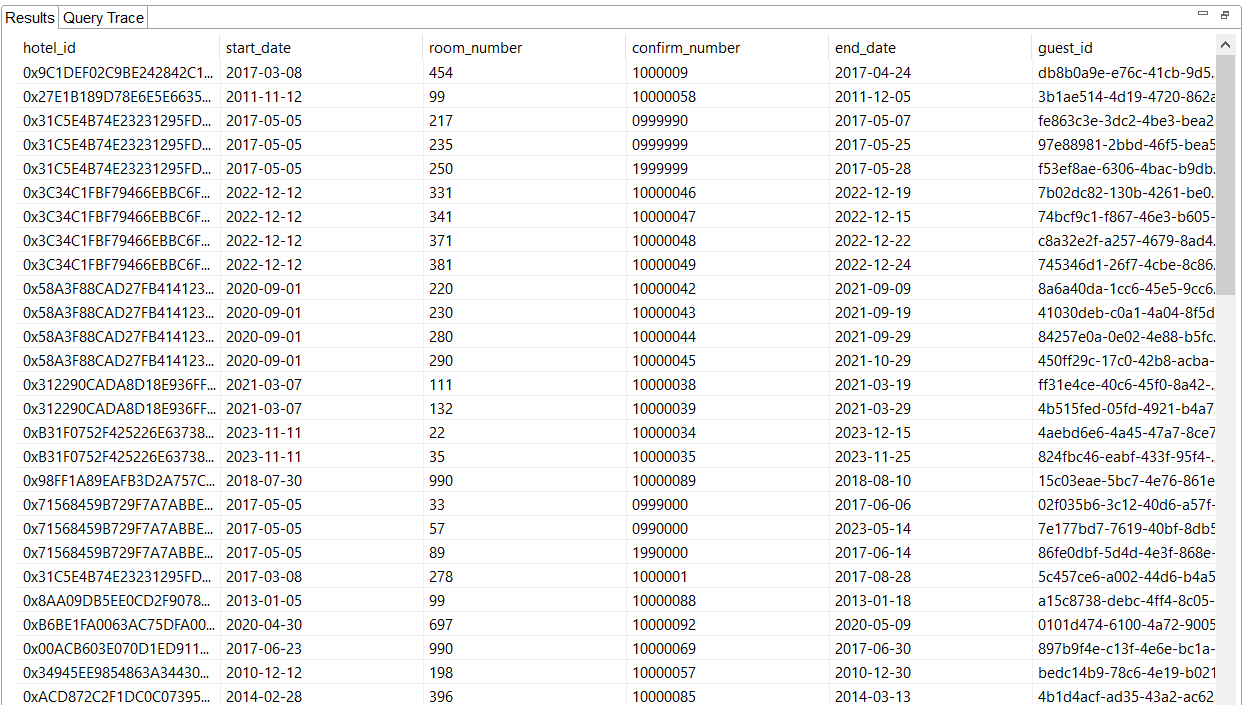
select\*from **reservation**.**reservations\_by\_confirmation**;



Slika :SEDMI UPIT

8.UPIT:

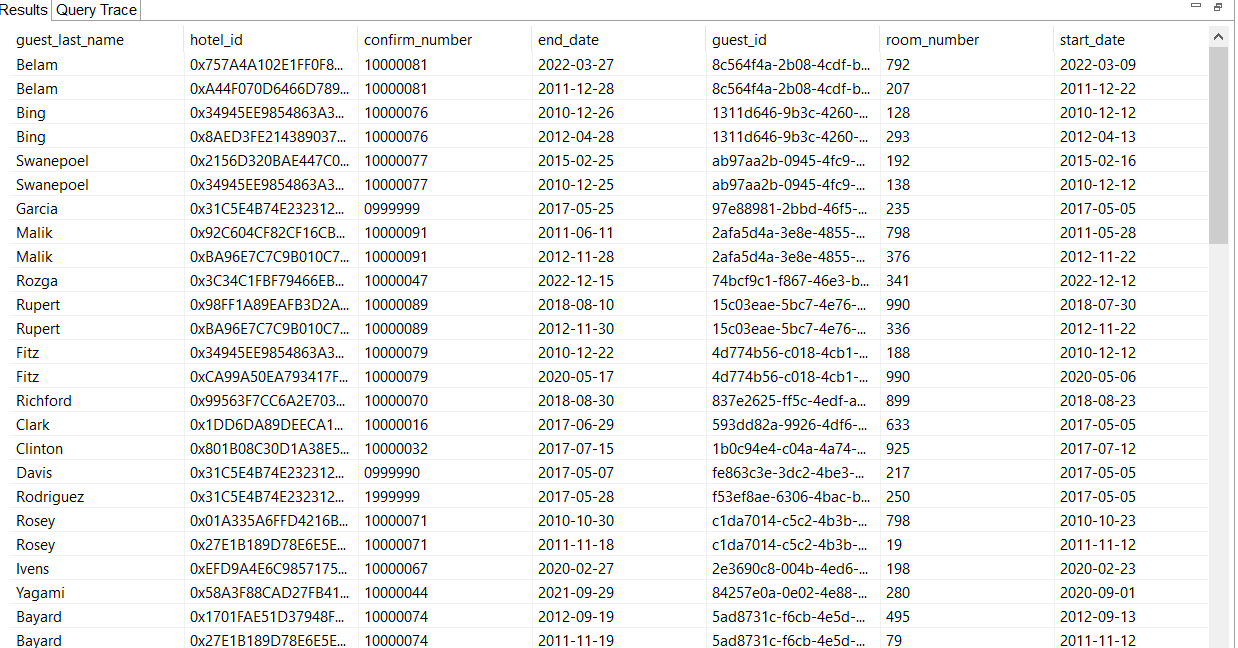
select\*from **reservation**.**reservations\_by\_hotel\_date**;



Slika :OSMI UPIT

9.UPIT:

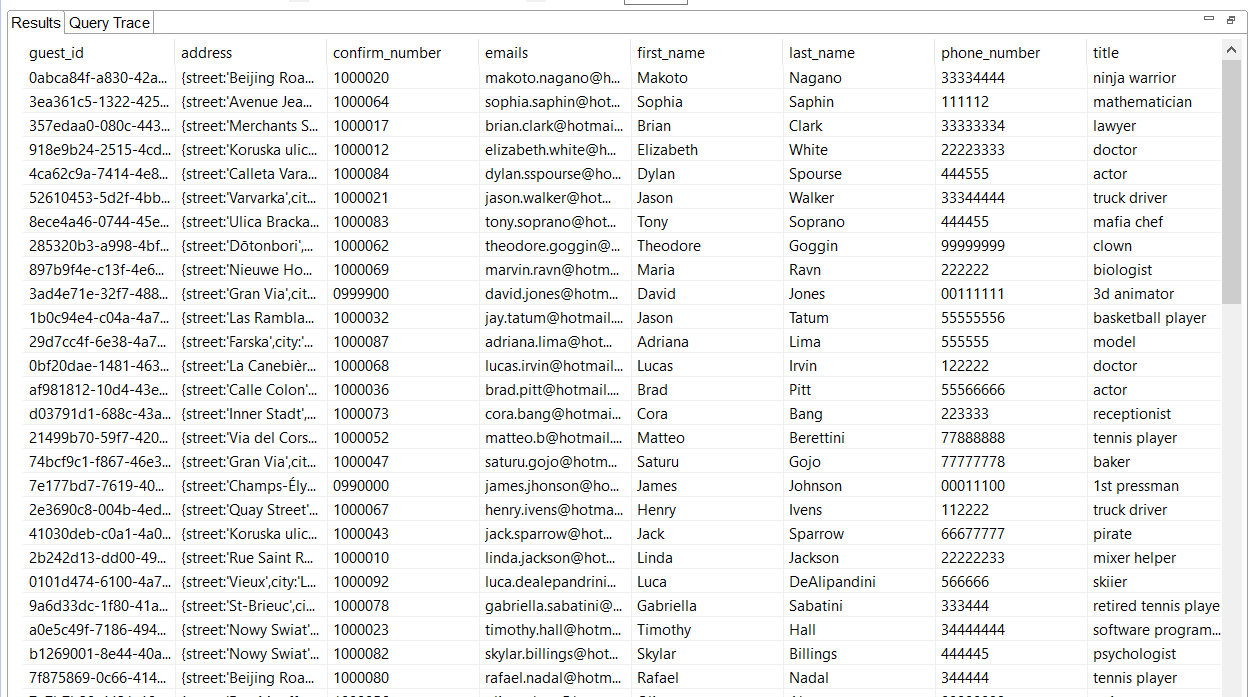
select\*from **reservation**.**reservations\_by\_guest**;



Slika :DEVETI UPIT

10.UPIT:

select\*from **reservation**.**guests**;



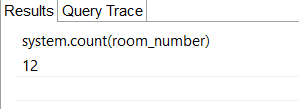
Slika :DESETI UPIT

11.UPIT:

select count(**room\_number**)

from **hotel**.**available\_rooms\_by\_hotel\_date**

where **hotel\_id**='0x31C5E4B74E23231295FDB724AD578C02C4A723F4BA2B4AF99F129EC2F4B3AD41';



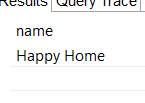
Slika ;JEDANAESTI UPIT

12.UPIT:

select **name**

from **hotel**.**hotels\_by\_poi**

where **poi\_name**='KONOPATY';



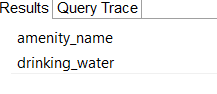
Slika :DVANAESTI UPIT

13.UPIT:

select **amenity\_name**

from **hotel**.**amenities\_by\_room**

where **room\_number**=101 and **hotel\_id**='0xAFAF3A868730E8993FB17CFC59EF2C0568DE1DF40D09CA829ABDCB074EB9CF3F';

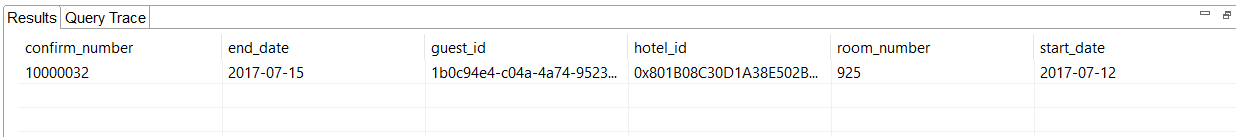


Slika :TRINAESTI UPIT

14.UPIT:

select\*from **reservation**.**reservations\_by\_confirmation**

where **confirm\_number**='10000032';



Slika :ČETRNAESTI UPIT

15.UPIT:

update **reservation**.**reservations\_by\_guest**

SET **confirm\_number**='10000073'

WHERE **hotel\_id**='0x27E1B189D78E6E5E6635234F11705A78A43F3FFBA994C0E5A82FCDFC7F86596D' and

**guest\_last\_name**='Bang' ;

Uspješno ažuriranje podataka.

1. Zaključak

Na kraju,uspjela sam napraviti sve ono što je bilo i planirano napraviti:instalirati Cassandru,spojiti ju s Datastax DevCenterom ,detaljno obratiti i shvatiti stupčaste baze podataka te uspjeti izraditi bazu podataka i na njoj demonstrirati osnovne pojmove vezane za Cassandru i stupčaste baze podataka.

Smatram da će rad biti koristan za početnike i nekoga tko želi naučiti osnove spomenutih tema na jednostavan i postepen način te da će potaknuti čitatelje na daljnje proučavanje ovih tema.

1. Popis literature

Jeff Carpenter, Eben Hewitt(2020). Cassandra, The Definitive Guide.

<https://www.youtube.com/watch?v=J-cSy5MeMOA&t=2993s&pp=ygUSY2Fzc2FuZHJhIHR1dG9yaWFs>