NoSQL baze podataka - Oracle NoSQL kroz praktične primjere

Markanović, Mia

Sveučilište u Rijeci – Odjel za informatiku

Preddiplomski jednopredmetni studij informatike

Mia Markanović

NoSQL baze podataka - Oracle NoSQL kroz praktične primjere

Završni rad

Mentor: doc. dr. sc. Danijela Jakšić

Rijeka, 15.7.2021.

ODJEL ZA INFORMATIKU
SVEUČILIŠTE U RIJECI

Rijeka, 8.6.2021.

Zadatak za završni rad

Pristupnik: Mia Markanović

Naziv završnog rada: NoSQL baze podataka - Oracle NoSQL kroz praktične primjere

Naziv završnog rada na eng. jeziku: NoSQL databases - Oracle NoSQL Database with practical examples

Sadržaj zadatka:

NoSQL ili nerelacijska baza podataka termin je koji se koristi za baze podataka koje su najbolji izbor za velike, web aplikacije gdje je potrebna veća skalabilnost i brzina. Jedna od takvih baza podataka koja će se prikazati u ovom završnom radu je Oracle NoSQL Database. Oracle nudi svoju besplatnu uslugu Oracle NoSQL Database Cloud Service kako bi korisnici mogli spojiti svoje aplikacije na Oracle Cloud gdje mogu izraditi i upravljati bazom. Oracle nudi dokumentne, stupčane i ključ-vrijednost nerelcijske baze podataka. U ovom završnom radu na vlastitom primjeru tvrtke za iznajmljivanje vozila bit će prikazano kako raditi s Oracle NoSQL bazom podataka.

Mentor

doc.dr.sc. Danijela Jakšić

Voditelj za završne radove

doc. dr. sc. Miran Pobar

Zadatak preuzet: 8.6.2021.

(potpis pristupnika)

Sadržaj

Sažetak

NoSQL ili nerelacijska baza podataka termin je koji se koristi za baze podataka koje su najbolji izbor za velike, web aplikacije gdje je potrebna veća skalabilnost i brzina. Zbog napretka razvoja i korištenja web aplikacija potrebno je pružiti usluge milijardama korisnika i uređaja kao što su pametni mobiteli, tableti, internet TV-i te ostali pametni uređaji. Za razliku od relacijskih baza podataka koje pohranjuju strukturirane podatke u tablice gdje postoje stupci i reci te se među njima postavljaju veze, nerelacijske baze rješavaju problem konzistentnosti. Ne baziraju se na tablicama i vezama među njima tj. na pravilima RDMS-a, što je kratica za sustav za upravljanje relacijskim bazama podataka. Web aplikacije bile bi užasno spore ukoliko bi pratile samo ACID transakcije koje im taj sustav nalaže. Stoga je bitno da svaki zahtjev garantira odgovorom, bio on uspješan ili ne, da se stanje sustava može mijenjati s vremenom bez ikakvog unosa oslanjajući se na posljednje dostupne podatke te da je baza eventualno konzistentna tj. da će sustav biti konzistentan s vremenom [3].

Postoje četiri vrste nerelacijskih baza podataka, a to su dokumentne, stupčane, ključ-vrijednost i graf NoSQL baze podataka. S obzirom na više vrsta postoji više baza podataka koje se mogu koristit poput MongoDB, Oracle NoSQL Database, Google Firebase, CouchDB, Cassandra itd.

Jedna od baza podataka koje je korištena u ovom završnom radu je Oracle NoSQL Database. Oracle nudi svoju besplatnu uslugu Oracle NoSQL Database Cloud Service kako bi korisnici mogli spojiti svoje aplikacije na Oracle Cloud gdje mogu izraditi i upravljati bazom. Oracle nudi ključ-vrijednost nerelacijske baze podataka [5]. Nakon registracije na Oracle službenim stranicama korisnik je u mogućnosti vrlo jednostavno izrađivati bazu podataka koja mu je potrebna. Oracle NoSQL Database primjenjuje se u raznim poljima poput svijeta video igara, IoT-a, mobilnih aplikacija, otkrivanja prijetnji u stvarnom vremenu itd. Neke od prednosti su što developeri nemaju potrebu za administriranjem poslužiteljskih podataka, nudi jednostavne CRUD upite, podržava modeliranje na JSON shemi i bez nje, Oracle izvodi upravljanje bazom podataka i pohranom podataka itd. [5] U ovom završnom radu na vlastitom primjeru tvrtke za iznajmljivanje vozila biti će prikazano kako upravljati Oracle bazom. Koristit će se jednostavni upiti za kreiranje, brisanje, ažuriranje i dohvaćanje podataka iz tablica. Također, prikazat će se i nekoliko složenijih upita.

Ključne riječi

NoSQL, Oracle NoSQL Database Cloud Service.

1. Uvod

Godinama su se za pohranu podataka koristile relacijske baze podataka. No, u zadnjih nekoliko godina zbog razvoja velikih web aplikacija pojavila se potreba za boljom skalabilnosti i brzinom, što nerelacijske baze mogu omogućiti. To ne znači da je upotreba relacijskih baza podataka nestala. Potrebno je dobro poznavati oba dvije skupine kako bi se znalo koju kada, gdje i kako primijeniti. U relacijskim bazama podataka strukturirani podaci pohranjuju se u grupe koje nazivamo tablice. Zbog veza kroz stupce tablica potrebno je horizontalno skaliranje koje je gotovo nemoguće. Stoga nerelacijske baze podataka rješavaju taj problem. Početkom 21. stoljeća kreće veliki fokus na skalabilne web aplikacije odnosno aplikacije koje trebaju pružiti usluge velikom broju korisnika i pametnih uređaja, ne oslanjajući se na poznati sustav za upravljanjem bazama podataka i ACID transakcije. Ono čega se treba pridržavati je da svaki zahtjev ima i svoj odgovor, da se stanje sustava može mijenjati s vremenom pomoću posljednje dostupnih podataka te da će sustav biti konzistentan [3].

Pitanje rješenja problema povezanog s podacima velikog volumena koji su polustrkturirani dovelo je do potrebe za zamjenjivanjem tradicionalnog načina upravljanja bazom nekim novim. Došlo je do pojave novih vrsta baza podataka kao što su dokumentne, stupčane, ključ-vrijednost i graf baze podataka te su sve zajedno nazvane NoSQL [16].

U ovom završnom radu biti će objašnjena NoSQL baza podataka kao i njene karakteristike, vrste te će se na vlastitim praktičnim primjerima tvrtke za iznajmljivanje vozila pokazati upiti nad bazom. Za primjere će se koristiti Oracle-ova baza podataka putem besplatnog Oracle NoSQL Database Cloud Service-a.

Rad je organiziran na dva glavna poglavlja. U prvom poglavlju biti će opisan sam pojam NoSQL-a, te će se ukratko spomenuti povijest nerelacijskih baza podataka odnosno kada se razvila potreba za njima. U tom poglavlju spomenut će se karakteristike nerelacijskih baza zajedno s njezinim vrstama i opisom svake od njih. Na kraju poglavlja će prikazati prednosti u nedostaci upotrebe NoSQL-a. U drugom poglavlju govori se o alatu koji će se koristiti za praktične primjere. Podijeljen je na nekoliko podpoglavlja gdje se prvo opisuje usluga Oracle NoSQL Database Cloud Service, njezine karakteristike i značajke, a zatim se kroz razne praktične primjere prikazuje njezina primjena.

2. Nerelacijske baze podataka (NoSQL)

U mnogim literaturama nerelacijske baze podataka interpretiraju se kao „not only SQL“ misleći se na to da ne koriste SQL za upravljanje podacima. NoSQL se ne pridržava pravila sustava za upravljanje bazama podataka (RDBMS) te se ne bazira na tablicama pri izgradnji. Poznate ACID transakcije zamijenjene su sa BASE transakcijama (slika 1.). ACID je skraćenica za atomičnost, konzistentnost, izolaciju i izdržljivost te se smatra zlatnim pravilom RDBMS-a, a BASE označava osnovnu dostupnost, soft state i eventualnu konzistentnost (tablica 1.). Eroc Brexer osmislio je BASE transakcije jer je smatrao da je nemoguće za distribuirana računala provoditi konzistentnost, dostupnost i dijeljenje tolerancije istovremeno, pa se taj teorem naziva CAP teorem (eng. consistency, availability, partition tolerance). Podaci bi trebali biti konzistentni i nakon izvođenja operacije što bi značilo da kad su jednom zapisani uvijek trebaju biti dio zahtjeva za čitanje. Svi podaci trebali bi biti uvijek dostupni te sustav bi trebao nastaviti funkcionirati i kada komunikacija između servera nije stabilna. Tako će ostali dijelovi baze podataka biti dostupni ukoliko jedan nije. Po CAP teoremu nemoguće je izvođenje više od dvaju od navedenih svojstava zajedno [3]. Pojavljuje se puno izazova za RDBMS sustav koji se odnosi na dobro definiranu strukturu kod podataka. Podaci su gusti i uglavnom slični. Gradi se na preduvjetu da postavke podataka mogu biti definirani unaprijed i da je njihov međusobni odnos dobro uspostavljen. Indeksi mogu biti dosljedno definirani nad skupovima podataka i da takvi indeksi mogu biti ravnomjerno iskorišteni za brže postavljanje upita. Nažalost, pred RDBMS se pojavljuju problemi. RDBMS se može nositi s nekim nepravilnostima i nedostatkom strukture u kontekstu masivnih skupova podataka sa slabo definiranom strukturom. RDBMS se zapravo čini prisiljen prilagoditi. S velikim skupovima podataka tipični mehanizmi pohrane i metode pristupa također su rastegnuti. Potrebna je denormalizacija tablica, popuštanje ograničenja i transakcijskog jamstva kako bi pomoglo RDBMS-u, ali nakon takvih promjena RDBM počinje ličiti NoSQL-u. No, fleksibilnost ima svoju cijenu. NoSQL ublažava probleme koje RDBMS nameće i olakšava rad s velikim raspršenim podacima. Zapravo, jedno od svojstava koje najviše nedostaje NoSQL-u je sam SQL na čemu se radi kako bi se u budućnosti premostio taj jaz [16].

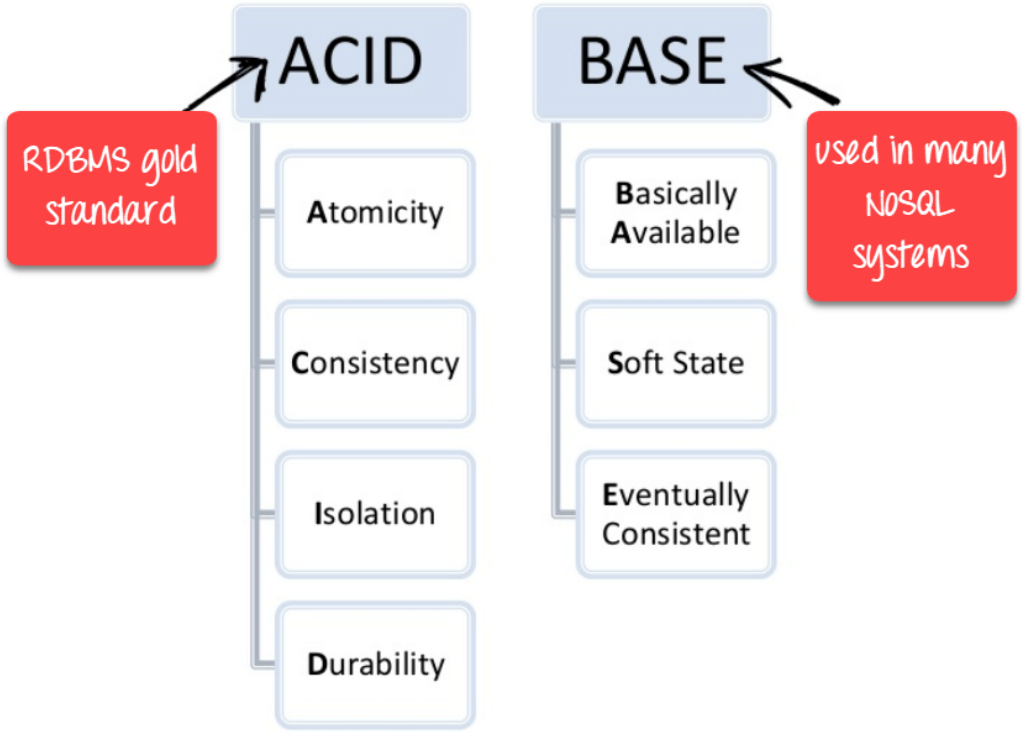
Rast sadržaja kojem upravlja korisnik utjecalo je na brz porast volumena i vrste podataka kojima se upravlja. Također su se pojavili i novi izvori podataka poput senzora, GPS-a (eng. Global Positioning Systems), uređaja za pronalaženje (eng. trackers), sustava za praćenje i ostalog. Ukratko takve podatke velikog volumena nazivamo big data[1] te su zadali nove izazove pohrani i analizi podataka. Podaci su postali polustrukturirano raščlanjeni, pa se tako bilo teško držati definirane sheme tablice. Pojavile su se nove vrste baze podataka koje spadaju pod NoSQL, a razlikuju se po svojim značajkama i vrijednostima koje mogu podnijeti. Stoga je ponekad teško izabrati koju bazu koristiti, pa takva odluka zahtjeva iskustvo developera [16].

Nerelacijske baze podataka nisu nešto novo. Prva takva pohrana pojavljuje se kada je izumljeno računalo. Postojale su godinama u specifičnim domenama poput hijerarhijskih direktorija za pohranu podataka o autentifikaciji i autorizaciji. No, pojam NoSQL javlja se pojavom skalabilnih Internet aplikacija. Na početku su se nerelacijske baze podataka koristile kod distribuiranih i paralelnih sustava. Prvi put kad je RDBMS pokazao problem s upravljanjem s masivnim podacima bilo je kod Google-ovog stroja Inktomi koji je ujedno i prvi pravi stroj pretraživač. Problem se odnosio na učinkovitu obradu, paralelizam, skalabilnost i cijenu. Google

[1] Big Data je pojam koji se odnosi na podatke velikog obujma velike brzine i složenosti te na promjenjive podatke koji zahtijevaju napredne tehnologije kako bi se ti podaci mogli prikupljati, pohranjivati, distribuirati, upravljati i analizirati. [11]

je s vremenom izgradio skalabilnu stabilnu infrastrukturu za takav stroj i ostale aplikacije poput Google Maps-a, Google Earth-a, Gmail-a, Google Apps-a itd. Pristup je bio takav da se riješio problem na svakom stogu. Cilj je bio stvoriti skalabilnu infrastrukturu za paralelnu obradu podataka velikog volumena. Kreirali su cijeli mehanizam koji uključuje distribuirani sustav datoteka, dokumentnu pohranu podataka, distribuirani koordinirani sustav, MapReduce[2] paralelni algoritam itd. Također objavili su dokument gdje objašnjavaju ključne značajke te infrastrukture. Iako se nekada terabajt osobnih podataka činio kao nešto ogromno i nemoguće za pohraniti, danas postoje lokalni diskovi za takvu količinu pohrane. Živimo u vremenu bujnog rasta podataka. Naše slike s kamera uređaja, blogovi, društvene mreže i njihova svakodnevna ažuriranja. elektronički dokumenti, glazbene i video datoteke rastu velikom brzinom. Koristimo i proizvodimo ogromne količine podataka. Neka istraživanja dokazala su da je količina digitalnih podataka dosegla 35 zetabajta do 2020. godine. Ako se vratimo u 2009. godinu, možemo svjedočiti nekim zanimljivostima. Facebook je objavio informacije o svom sustavu za pohranu. Sveukupna veličina slika na Facebook-u iznosila je 1.5 petabajta onda možemo samo izmisliti koliko iznosi danas. Film Avatar zauzeo je 1 petabajt sa korištenim 3D efektima [16].

Možemo zaključiti kako veličina podataka raste, pa je efikasna pohrana i pristup takvim podacima težak. Stvari se kompliciraju i kod toleriranja kvarova i kod sigurnosne kopije. Takvi podaci uključuju pokretanje paralelnih procesa. Oporavak nakon bilo kakvog kvara i pružanje rezultata u razumno kratkom periodu je kompleksno. NoSQL je prvi korak prema rješenju takvih problema [16].



Slika 1. Usporedba ACID i BASE transakcija [1]

[2] MapReduce je algoritam velike skalabilnosti te odlične tolerancije kod pojave kvarova [16].

Tablica 1. Opis ACID i BASE transakcija [3]

ACID	BASE
Atomičnost osigurava da ukoliko neki dio transakcije ne uspije, cijela transakcija je neuspješna.	Svaki zahtjev garantira odgovorom, uspješnim ili neuspješnim.
Transakcija ne može ostaviti bazu u stanju koje je nekonzistentno.	Svako stanje sustava trebalo bi se promijeniti s vremenom i kad nema nikakvih unosa.
Svaka transakcija je izolirana od druge te si time ne ometaju istovremeno izvršavanje, pa to izgleda kao da se pojavljuju jedna iza druge.	Baza može biti trenutno nekonzistentna, ali će biti eventualno konzistentna.
Sve transakcije ostaju sačuvane u bazi u slučaju prekida.	

2.1. Karakteristike NoSQL-a

NoSQL ne rješava samo problem skalabilnosti nego nudi još mnogo toga. Neke od karakteristika su:

- Baza podataka je bez sheme što znači da nema potrebe za razmišljanjem o strukturi baze, pa se tako s vremenom mogu dodavati polja ili spajati podatci [3]. Nije potrebno definirati shemu podataka te nudi heterogene strukture podataka u jednoj domeni [1].
- Brzina je jedna velika prednost NoSQL-a jer omogućuje prijenos podataka između naizmjenično povezanih uređaja u milisekundama, radilo se o maloj ili velikoj količini [3].
- Jednostavni API (eng. Application Programming Interface) nudi jednostavna sučelja za pohranu i pretraživanje podataka te manipulaciju podacima na nižoj razini. Protokoli bazirani na tekstu korišteni su uglavnom s HTTP[3] REST[4]-om s JSON-om. API omogućava da internetske baze podataka rade kao usluge na internetu [1].
- Eventualna konzistentnost jedna je od glavnih karakteristika. Da bi se postigla skalabilnost i dostupnost potrebno je imati kopije podataka na više uređaja. Kad god se dogodi promjena podataka na nekom uređaju mora se proširiti i na druge replike. Neke od promjena će se ažurirati odmah, a neke tijekom vremena te tako kopije s vremenom postanu konzistentne [1].

[3] HyperText Text Transfer Protocol Secure je internetski protokol tj. skup pravila po kojima se web stranica prenosi s poslužitelja do korisnika [17].

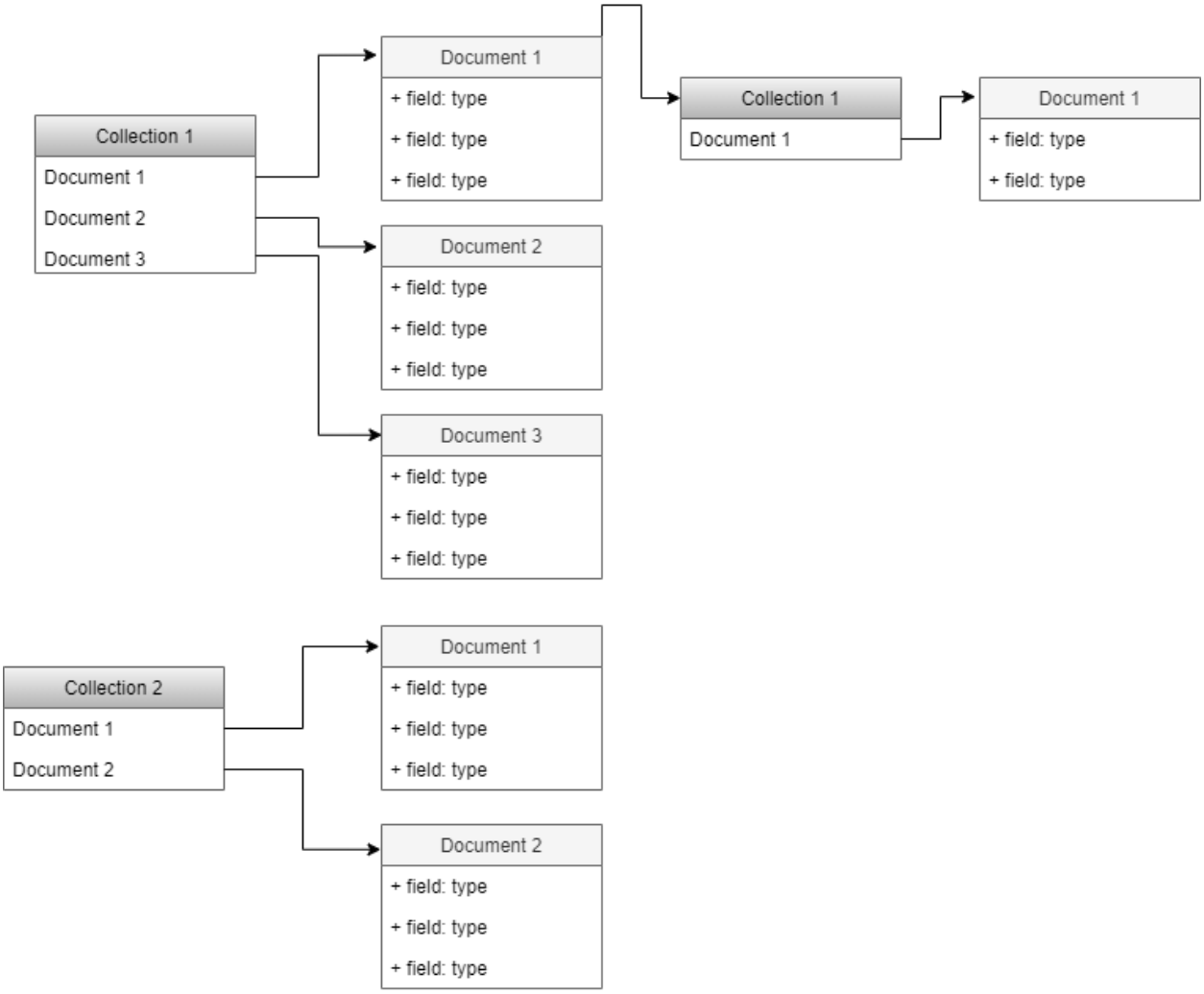
[4] REST (eng. Representional state transfer) kreiran je za upravljanje i razvoj arhitekture Web-a [18].

2.2. Vrtse NoSQL-a

Postoje 4 vrste NoSQL baza podataka, a to su dokumentne, stupčane, ključ-vrijednost i graf NoSQL baze podataka. U nastavku će biti opisana svaka od njih.

Dokumentne baze podataka

Dohvaćaju i pohranjuju podatke kao ključ-vrijednost parove u dokumente, ali vrijednosni dio je pohranjen kao dokument u JSON ili XML[5] formatu. Dokumente nije potrebno definirati kao što je slučaj kod racionalnih baza podataka gdje treba znati sve stupce u tablicama [1]. Dokumenti se nalaze unutar kolekcija. Kolekcije omogućavaju indeksiranje dokumenata ne samo sa primarnim ključevima nego i na temelju njihovih svojstava [16]. Prednost ove baze podataka je njezina fleksibilnost odnosno prihvaćanje bilo kojeg tipka podatka [15]. Ne bi se trebale koristiti za kompliciranе transakcije koje zahtijevaju više operacija [1]. Neke od dokumentnih baza podataka su MongoDB, CouchDB, RavenDB itd. (slika 2.)



Slika 2. Primjer modela dokumentne baze podataka

[5] XML je kratica za EXtensible Markup Language tj. jezika za označavanje podataka koji je jednostavno čitljiv i ljudima i računalima [19].

Stupčane baze podataka

Umjesto pohrane podataka u redovima, podaci se pohranjuju u stupcima unutar neke druge grupe/obitelji stupaca [16]. Svaki se stupac tretira zasebno. Vrijednosti u stupcima raspoređeni su jedne do drugih što pruža učinkovitost. Ne stvara prostora za null vrijednosti nego jednostavno ne sprema takve stupce kada vrijednost ne postoji za taj stupac. Podaci su također sortirani po redu ovisno o primarnom ključu koji se nekad nazivao ključem reda jer se jedinstveno označuje podatke unutar jednog reda tablice. Za primjer možemo uzeti podatke neke osobe kao što su ime, prezime, poštanski broj, grad. Ako bismo koristili stupčanu bazu podataka mogli bismo napraviti još jednu obitelj stupaca „ime“ koje bi se sastojalo od stupaca „ime“ i „prezime“ ili „lokacija“ sa stupcima „poštanski broj“ i „mjesto“ (tablica 2.). Iako dokumentne baze podataka imaju velike mogućnosti agregiranja podataka i dobre perfomanse kod pristupa podacima, složeni upiti mogu predstavljati probleme [16]. Najpoznatije stupčane baze su BigTable, Cassandra, SimpleDB itd.

Tablica 2. Primjer modela stupčane baze podataka [1]

<table>
<tr><th colspan="3">Obitelj stupaca</th></tr>
<tr><td rowspan="6">Ključ reda</td><td colspan="2">Naziv stupca</td></tr>
<tr><td>Ključ</td><td>Ključ</td></tr>
<tr><td>Vrijednost</td><td>Vrijednost</td></tr>
<tr><td colspan="2">Naziv stupca</td></tr>
<tr><td>Ključ</td><td>Ključ</td></tr>
<tr><td>Vrijednost</td><td>Vrijednost</td></tr>
</table>

Ključ-vrijednost baze podataka

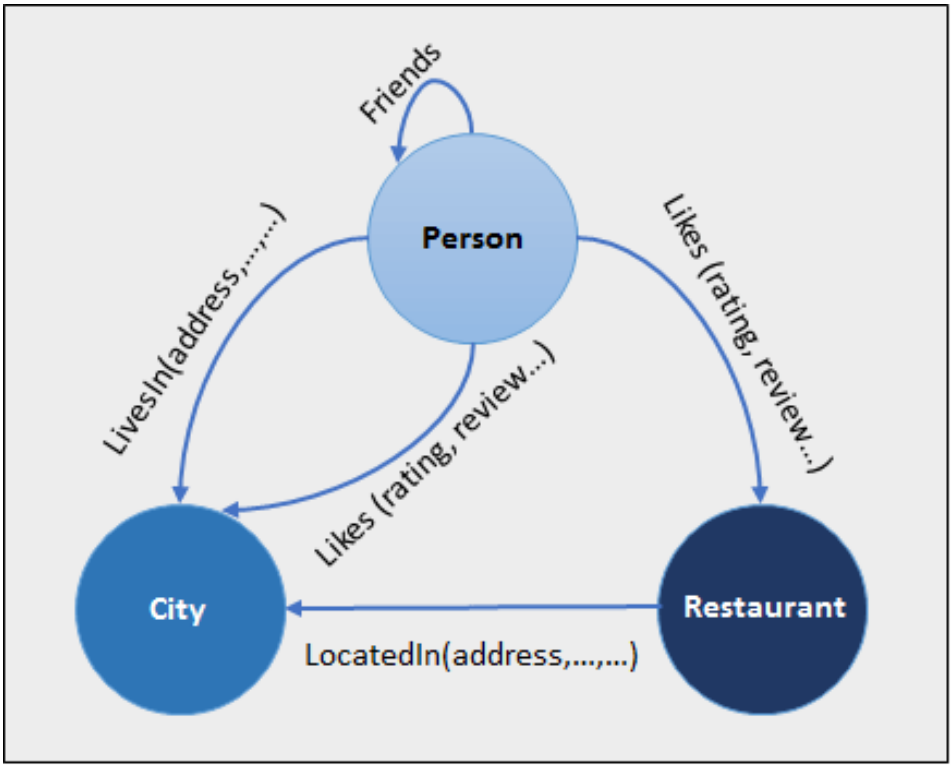
Podaci su pohranjeni kao ključ-vrijednost parovi [1]. Svaki ključ je jedinstven. Vrijednosti mogu biti bilo kojeg tipa stoga spadaju pod najpopularnije vrste baze podataka. Mogu upravljati sa jako puno podataka i velikim opterećenjem (tablica 3.) [15]. Najčešće se koriste za sadržaj u košaricama za online kupovinu, a poznate ključ-vrijednost baze podataka su Redis, Membase, Memcached, DynamoDB itd. [1]

Tablica 3. Primjer modela ključ-vrijednost baze podataka [1]

Ključ	Vrijednost
Ime	Mia
Prezime	Markanović
Spol	F
Adresa	Ilica 12
Email	mmarkanovic@gmail.com

Graf baze podataka

Sadrže entitete koji su pohranjeni kao čvorovi koji su povezani s krajevima, te svaki čvor i kraj ima jedinstveni identifikator [1]. Čvor pohranjuje informacije poput imena, adrese itd. dok krajevi predstavljaju veze između različitih čvorova (slika 3.) [15]. Koriste se za društvene mreže, a neke od poznatiji su Flock DB, Neo4J, Infinite Graph itd. [1]



Slika 3. Primjer modela graf baze podataka [1]

2.3. Prednosti i nedostaci NoSQL-a

Mnoge su prednosti korištenja NoSQL-a što ga i čini često korištenom bazom podataka za web aplikacije. Ima jednostavan dizajn sheme koja je fleksibilna te se lako može mijenjati bez ikakvog zastoja ili ometanja pružanja usluge. Ono što su još pridonosti fleksibilnosti je objektno orijentirano programiranje koje je jednostavno za korištenje. Također, NoSQL podržava i druge programske jezike važne za developere i njihove platforme. Jednako upravlja strukturiranim, nestrukturiranim i polustrukturiranim podacima. Zbog baratanja sa velikim podacima (Big Data) omogućeno je upravljanje brzinom prijenosa podataka, njihovom raznolikošću, volumenom i složenosti. Jedna od prednosti je što nema potrebe za priručnom memorijom (eng. cache) za pohranu podataka. Izbjegavanje pravila RDBMS-a omogućuje lakše implementiranje [1].

No, postoje i neki nedostatci NoSQl-a kao što su ograničene mogućnosti upita za razliku od SQL-a. RDBMS je znatno zreliji za korištenje. NoSQL ne može garantirati konzistentnost kada se istovremeno izvršava više transakcija. Problem nastaje kada se nagomila količina podataka, pa postaje teško upravljati s jedinstvenim vrijednostima jer ključevi postaju kompliciranijima. Ne može dobro upravljati s racionalnim podacima te je novim developerima teško naučiti kako NoSQL funkcionira [1].

Sve u svemu, možemo zaključiti da NoSQL nije rješenje baš za sve aplikacije. Stoga mnogi proizvodi ne izbacuju potpuno ACID transakcije nego ih koriste sukladno s NoSQL bazom podataka. Možemo se složiti da je NoSQL svakako jedna velika tehnološka promjena kao i promjena u načinu razmišljanja i rješavanju problema koji se nađe pred developerima [3].

3. Oracle NoSQL baza podataka

Za izradu praktičnih primjera korištena je besplatna usluga Oracle NoSQL Database Cloud Service kojoj se može pristupiti na Oracle službenoj stranici. Nudi vrlo jednostavno stvaranje aplikacija koristeći ključ-vrijednost model. Omogućava pouzdanost, fleksibilnost, skalabilnost i visoke perfomanse što je potaklo mnoge kompanije na korištenje poput Airbus-a, Turkish Notary Union-a, BizDoc-a, Cisco-a, Yahoo-a itd. [2]

Neku od slučajeva upotrebe NoSQL bate podataka [4]:

- Personalizacija - stvaranje bogatog, personaliziranog korisničkog iskustva sa brzim odgovorima (JSON podatkovna podrška i automatsko skaliranje) temeljenim na kreiranom sadržaju od strane individualnih korisnika.
- Otkrivanje prijevare u stvarnom vremenu - brzo vrijeme odgovora dok se simultano procesiraju transakcije putem POS uređaja.
- Mobilne aplikacije - izgradnja modernih, skalabilnih i responzivnih mobilnih aplikacija
- Industrija igara - status igrača i same igre je u stvarnom vremenu dok se paralelno dohvaćaju i ažuriraju podaci igrice jer se tijekom igranja sprema jako puno podataka (pohranjivanje rezultata, ubijanje protivnika, vrijeme zadnjeg igranja, skupljanje bodova, podaci o profilu korisnika itd.) koji uvijek moraju biti brzo dostupni (u milisekundama).
- Internet of Things (IoT) - pojednostavljeno backend pohranjivanje i analiza velikih podataka za koje su potrebne brze operacije pisanja i čitanja,

3.1. O usluzi

U potpunosti upravljiva usluga Oracle NoSQL Database Cloud Service namijenjena je za operacije koje zahtijevaju predvidljive i brze odgovore za jednostavne upite. Developerima omogućuje fokus na sam razvoj aplikacije, a manje brige oko klaster usluga[6] ili nadzora sustava, podešavanja, dijagnosticiranja i skaliranja. Nakon stvaranja Oracle Cloud računa korisnik je u mogućnosti stvarati tablice i specificirati zahtjeve za protok i pohranu tablica. Određen je kapacitet putem jedinica za čitanje i pisanje za protok podataka i vrijednost u GB za spremanje jedinica [5].

Developeri se mogu povezati na Oracle NoSQL Database Cloud Service kako bi radili s tablicama pomoću druge Oracle usluge tj. konzole Oracle Cloud Infrastructure Console koja se može preuzeti na njihovim službenim stranicama [5].

3.2. Ključne značajke

Neke od ključničih značajki Oracle NoSQL Database Cloud Service-a su [6]:

- Developeri nemaju potrebu za administriranjem poslužiteljskih podataka ili infrastrukture i sigurnosti jer Oracle održava hardver i softver.
- Nakon izrade aplikacije i spajanja s uslugom developeri mogu odmah krenuti sa čitanjem i pisanjem podataka jer Oracle izvodi upravljanje bazom podataka, upravljanje pohranom podataka, visoku dostupnost i skalabilnost kako bi se izgradila aplikacija visokih perfomansi.

[6] Kluster usluga eng. *cluster service* jer sistemska komponenta koja se koristi za kontroliranje klastera na pojedinom čvoru, olakšava komunikaciju između drugih softverskih komponenti, osigurava konstantnost klastera kroz sve čvorove i barata obavijestima

- Aplikacije vraćaju podatke predvidljive latencije[7].
- Cloud Service usluge nudi jednostavne CRUD API-eve kako bi se tablice lako kreirale i kako bi se lako upravljalo podacima u njima.
- Podržava modeliranje bazirano na shemi i bez nje (JSON).
- Oracle NoSQL Database Cloud Service koristi više dostupnih domena AD (eng. *Availabillity Domain*) ili domene grešaka FD (eng. *Fail Domain*) kako bi u slučaju da je jedna od tih domena nedostupna korisnički podaci i dalje bili dostupni putem druge domene.
- Podaci su kriptirani AES-om 256 (eng. *Advanced Encryption Standard*) u mirovanju i HTTPS protokolom u trenutku prijenosa.
- Developeri mogu pristupiti podacima sa SQL upitima.
- Nudi mogućnost postavljanja vremena življenja redova u tablicama nakon čijeg će isteka ti redovi postati nedohvatljivi.
- Nudi opciju ažuriranja dijelova JSON dokumenta jer se takva ažuriranja pojavljuju na serveru, pa bi potrošila kapacitet protoka.
- Mogućnost sekundarnog indeksiranja poboljšava performanse na više putanja koristeći indeks.
- Podržava ACID transakcije.
- Povećanjem opterećenosti aplikacije poveća se njihova sigurnost propusnosti kako bi održali dosljedno korisničko iskustvo, a u suprotnom, kada se smanjuje opterećenost mogu smanjiti sigurnost propusnosti kako ni snizili operativne troškove.

3.3. Koncepti

3.3.1. Tablica

Tablica je kolekcija redova gdje svaki od njih sadrži podatke aplikacije. Svaki red sastoji se od ključa i polja vrijednosti koji se definiraju na početku tijekom kreiranja tablice. Kao što je navedeno gore, svaka tablica ima svoje spremište koje je unaprijed određeno kapacitetom ovisno radi li se o besplatnoj verziji Cloud Service-a ili ne, te koliko smo mi sami odredili kapaciteta za svaku od tablica [7].

Big Data

Oracle NoSQL Database Cloud Service podržava sve tri vrste velikih podataka poznatih pod imenom „big data“ [7]:

1. Strukturirani – podaci koji mogu biti organizirani i pohranjeni u tablicama sa predefiniranim strukturama poput relacijskih tablica koje su jednostavne za upravljanje i analiziranje. Primjer: podaci sa kreditnih kartica.
2. Nestrukturirani – podaci koji ne mogu biti strukturirani ili pohranjeni u strukturiranim tablicama. Oracle NoSQL Database Cloud Service nudi definiranje tablica s JSON podacima u redovima kako bi se pohranile te vrste podataka. Primer: multimedija, rezultati Google pretraživanja.
3. Polustrukturirani – podaci koji ne mogu stati u relacijske tablice, ali mogu biti organizirane poput njih, u stupcima i redovima. Oracle NoSQL Database Cloud Service to rješava pomoću ključ-vrijednost parova. Primjer: XML datoteka.

[7] Latencija je vrijeme koje prođe od trenutka kad je poslan zahtjev za podacima do trenutka dok se oni ne pojave. [10]

Vrste podataka

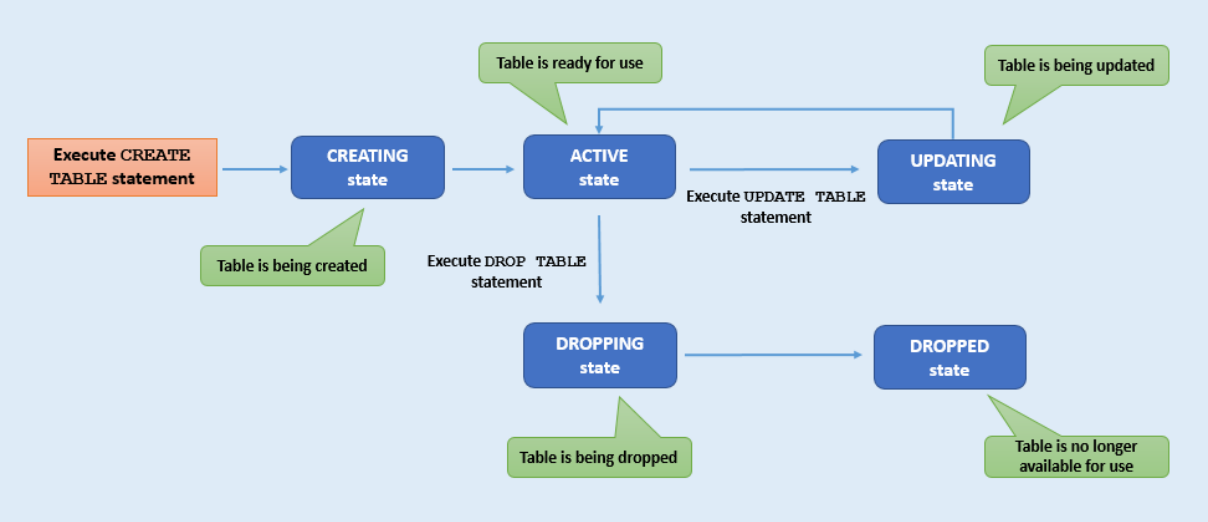
Tablica je kreirana korištenjem jezika definicije podataka eng. Data Definition Language koji služi za definiranje vrste podataka i primarnih ključeva. Oracle NoSQL Database Cloud Service ima široku lepezu vrste podataka koja se može koristiti (tablica 4.).

Tablica 4. Vrste podataka i njihovi opisi [8]

Vrste podataka	Opis
BINARY	Slijed od 0 ili više bitova.
FIXED_BINARY	Niz bitova fiksne veličine.
BOOLEAN	True ili false.
FLOAT	64-bitni realan broj dvostruke preciznosti.
LONG	32-bitni realan broj-
INTEGER	64-bitni cijeli broj.
STRING	Slijed Unicode znakova.
NUMBER	Decimalan broj proizvoljne preciznosti.
TIMESTAMP	Točnost u vremenu s preciznošću pohranjeno i upravljano u UTC (Coordinated Universal Time).
ENUM	Nabrajanje prezentirano kao niz znakova čije su vrijednosti simbolički identifikatori koji su pohranjeni kao male integer (cjelobrojne) vrijednost u uređenoj predefiniranoj listi.
ARRAY	Uređena kolekcija fiksne duljine od 0 ili više podataka. Podaci koji nisu definirani kao JSON ne mogu biti NULL vrijednosti.
MAP	Neuređena kolekcija 0 ili više ključ-vrijednost parova gdje su svi ključevi stringovi i jedinstveni, a sve vrijednosti istog tipa. Ključ-vrijednost parovi se nazivaju polja gdje je ključ ime tog polja, a dodijeljene vrijednosti su vrijednosti polja.
RECORD	Fiksna kolekcija 0 ili više ključ-vrijednost parova gdje su kao i kod map vrste podataka svi ključevi jedinstveni i stringovi.
JSON	Svi valjani JSON podaci.

Stanja tablice

Svaka tablica prolazi pojedina različita stanja od kreiranja do brisanja. Tablica koja je u stanju DROPPING ne može prijeći u stanje ACTIVE, dok tablica u stanju ACTIVE može prijeći u stanje UPDATING (slika 4. i tablica 5.) [8].



Slika 4. Opis stanja tablica i njenog životnog ciklusa [8]

Tablica 5. Opisi stanja tablice [8]

Stanja tablice	Opis
CREATING	Slijed od 0 ili više bitova.
UPDATING	Niz bitova fiksne veličine.
ACTIVE	True ili false.
DROPPING	64-bitni realan broj dvostruke preciznosti.
DROPPED	32-bitni realan broj.

Indeksiranje

Aplikacije mogu kreirati indekse na bilo kojem polju podataka koji ima vrstu podatka koja dozvoljava indeksiranje uključujući i JSON podatke.

Kapacitet

Tijekom kreiranja tablica potrebno je odrediti kapacitet propusnosti i pohrane za svaku tablicu kako bi se limitirale operacije čitanja i pisanja. Veličina mjesta pohrane koju tablica može koristiti je limitirana kapacitetom pohrane [9].

Za procjenu kapaciteta potrebno je poznavati sljedeće pojmove [9]:

- Jedinica pisanja eng. Write Unit *(*WU*)* – jedna jedinica pisanja definirana je kao protok do1 KB podataka u sekundi. Operacija pisanja je bilo koji Oracle NoSQL Database Cloud Service API poziv koji rezultira umetanjem, ažuriranjem ili brisanjem zapisa. NoSQL tablica ima limitiranu vrijednost pisanja koja predstavlja broj jedinica pisanja koje mogu biti korištene svake sekunde. Primjer: zapis veličine manje od 1 KB zahtjeva 1 WU za operaciju pisanja dok zapis od 1.5 KB zahtjeva 2 WU.
- Jedinica čitanja eng. Read Unit *(*RU*)* – definira se kao protok do 1 KB u sekundi za konačno konzistentne operacije čitanja. NoSQL tablica ima limitiranu vrijednost čitanja koja predstavlja broj jedinica čitanja koje mogu biti korištene svake sekunde. Primjer: zapis veličine od 1.5 KB za konačno konzistentnu operaciju čitanja zahtjeva 2 RU i 4 RU za apsolutnu konzistentnu operaciju čitanja.
- Kapacitet pohrane – jedna jedinica pohrane je 1 GB pohrane podataka.
- Apsolutna konzistentnost – podaci koji se vraćaju moraju biti najnoviji podaci upisani u bazu podataka.
- Konačna konzistentnost - podaci koji se vraćaju ne moraju biti najnoviji podaci upisani u bazu podataka jer ako nema novih promjena nad podacima onda će svi pristupi tim podacima vratiti posljednje promjene.

Faktori koji utječu na kapacitet [9]:

- Veličina zapisa – povećanjem zapisa povećavaju se i jedinice čitanja i pisanja.
- Konzistentnost podataka i apsolutna konzistentna čitanja su dvostruko veći trošak od konačno konzistentnih čitanja.
- Sekundarno indeksiranje – kada je postojeći zapis u tablici modificiran, ažuriranje sekundarnih indeksa koristi jedinice pisanja. Ukupni rezervirani trošak protoka za operaciju pisanja je suma korištenih jedinica pisanja koje se koriste pri upisu u tablicu i ažuriranjem lokalnih sekundarnih indeksa.
- Modeliranje podataka – kod JSON-a bez sheme svaki se dokument samopopisuje što dodaje „overhead“ meta podataka na sveukupnu veličinu zapisa dok sa tablicama fiksirane sheme iznos je 1 bajt.
- Uzorak upita – cijena operacija upita ovisi o broju dohvaćenih redova, veličini izvornih podataka, broju predikata, projekcija i indeksa. Najjeftiniji upiti specifiraju *shard* i indeksirane ključeve kako biti dopustili sustavu upravljanje primarnim i sekundarnim indeksima.

3.3.2. Distribucija i komadanje

Tijekom kreiranja tablica najprije je bitno odrediti primarne i „shard“ ključeve jer oni pomažu efikasno distribuirati podatke. Oni se kreiraju samo kad se kreira i sama tablica te ostaju sve dok postoji tablica i ne mogu biti promijenjeni ili obrisani [7].

Potrebno je kreirati jedan ili više primarnih ključeva pri kreiranju same tablice. Primarni ključ jedinstveno identificira red u tablici. Kod jednostavnih CRUD operacija koriste se takvi ključevi kako bi se dohvatio određeni red za čitanje ili izmjenu [7].

Posebni „shard“ ključevi distribuiraju podatke kroz klastere kako bi se postigla veća učinkovitost te pozicioniraju zapise koji lokalno dijele „shard“ ključ radi lakšeg snalaženja i pristupa zapisima. Takvi zapisi nalaze se na istoj lokaciji i može im se pristupiti automatski. Kreiranje ključeva utječe na skaliranje i propusnost. Kada zapisi dijele „shard“ ključ može se obrisati više redova u tablici ili dohvatiti podskup redova u jednoj operaciji. Dobro dizajnirani *shard* ključevi zahtijevajući manje ciklusa za postavljanje ili dohvaćanje podataka s jednog komada eng. shard mogu poboljšati perfomanse. Određuju pohranu na istom komadu kako bi se olakšali učinkoviti upiti za vrijednosti ključa. Ukoliko se ne postavi „shard“ ključ, Oracle će koristiti primarne ključeve za komadanje. Savjetuje se izbjegavanje ključeva koji imaju nekoliko jedinstvenih vrijednosti kako bi podaci bili distribuirani na najučinkovitiji način kroz komade, shard-ove [8].

Kada odabiremo „shard“ ključ dobro je uzeti u obzir sljedeće [8]:

- Polja niske kardinalnosti grupiraju zapise zajedno na nekoliko komada (shard-ova). Zbog toga ti komadi zahtijevaju rebalansiranje podataka što povećava pojavu problema sa samim komadima. Zato bi svaki „shard“ ključ trebao imati visoku kardinalnost. Dobar izbor su npr. ID brojevi poput JMBAG, OIB i slično.
- Samo oni objekti koji dijele „shard“ ključ mogu sudjelovati u transakciji. Ako se koriste ACID transakcije koje obuhvaćaju više zapisa, onda je dobro koristiti takvu vrstu ključa.
- Kada su *shard* ključevi dobro raspoređeni onda niti jedan komad tj. *shard* ne ograničava kapacitet sustava.
- Upiti bi trebali biti usmjereni na određeni komad radi učinkovitosti i perfomansa. U suprotnom će se upit primijeniti na sve komade.

Dosljednost čitanja također je jedna od stavki koju je potrebno uzeti u obzir tijekom kreiranja tablica. Ona određuje različite nivoe fleksibilnosti ovisno o tome koja je kopija podataka korištena za ispunjavanje operacije čitanja. Nivoi konzistentnosti koje Oracle NoSQL Database Cloud Service sadrži su EVENTUAL i ABSOLUTE kao što je navedeno gore. Aplikacija može odrediti apsolutnu konzistentnost s garancijom da će sve operacije čitanja vratiti najnovije upisane vrijednosti za kreirane ključeve te takva konzistentnost rezultira sa višim troškovima. Konačna konzistentnost omogućava aplikaciji vraćanje vrijednosti puno brže čak i kad nije najnovija te se bi se trebala koristiti samo kad je potrebno [7].

3.4. Korištenje Oracle NoSQL baze podataka kroz primjere

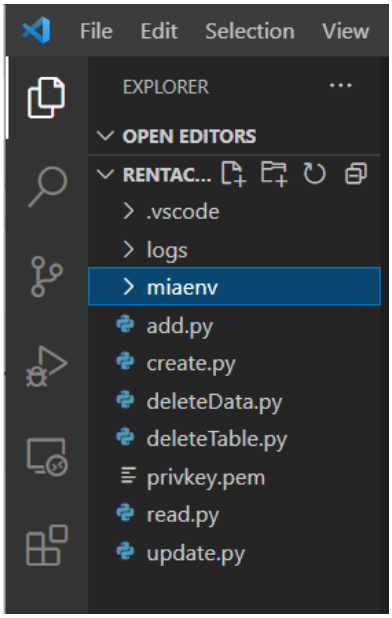
3.4.1. Spajanje na Oracle NoSQL Database Cloud Service

Kako bi se povezali s aplikacijom za koju se baza podataka izrađuje potrebno je osigurati sigurnu vezu. Za početak je potrebno izraditi Oracle Cloud račun koji nudi besplatno korištenje Cloud Service-a trideset dana. Svi uvjeti za daljnje povezivanje zahtijevaju jedinstvenog korisnika čiji će se podaci unutar Oracle Cloud računa koristiti. Povezivanje se može ostvariti pomoću Jave, Python-a, Node.js-a te Go-a. U ovom završnom radu korišten je Python unutar Visual Studio Code-a te će u nastavku biti opisane upute za uspješno pozivanje.

Za početak je potrebna instalacija softverskog razvojnog okruženja NoSQL Database Python SDK-a. Za primjere je napravljena mapa na lokalnom računalu s nazivom RentACar unutar koje

se nalaze python datoteke koje će se koristiti za CRUD upite (slika 5.). Unutar te mape kreirano je virtualno okruženje izvršavanjem naredbe **python -m venv miaenv** gdje je miaenv naziv tog okruženja. Venv je modul koji nudi izolaciju od ostatka sustava te je tako moguće instalirati dodatne pakete samo unutar tog okruženja tj. mape u kojoj je pokrenut. Pokreće se naredbom **.\miaenv\Scripts\Activate**. Python SDK zahtjeva Python verziju ne stariju od verzije 2.7.5 te ne mlađu od 3.5. Da bi se instalirao treba pokrenuti naredbu **pip install borneo**. Također zahtjeva kreiran račun na Oracle Cloud-u uz preplatu na Oracle Cloud Service. Cloud Service je stoga siguran zbog autentifikacije sa jedinstvenim identitetom i povezivanjem aplikacije sa specifičnim korisnikom. Nudi mogućnost kreiranja dodatnih korisnika ili grupa što za primjere ovog završnog rada nije bilo potrebno, ali se može proučiti na sljedećem linku Oracle-ovih službenih stranica. Da bi se Oracle NoSQL Database Cloud Service i Python SDK uspješno povezali u terminalu pokrećemo naredbu **pip install oci** kako bi instalirali Oracle Cloud Infrastructure Python SDK [13].

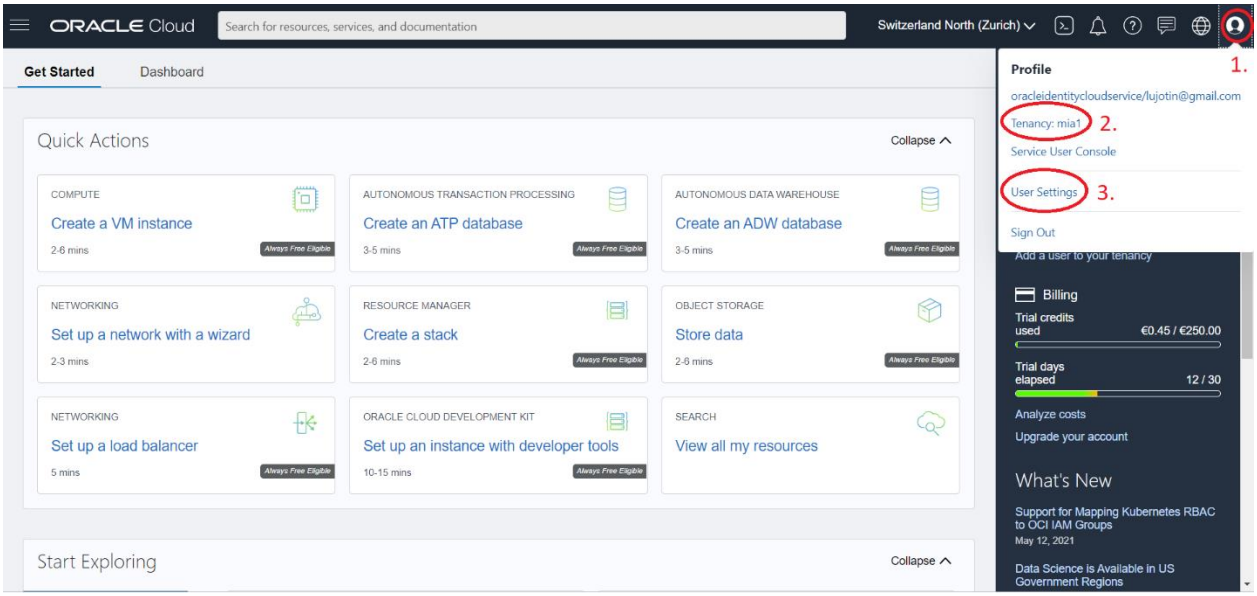
Da bismo radili upite u Pythonu potrebno je na početku svakog koda dodati nekoliko linija kako bismo se povezali s bazom podataka na Oracle Cloud Service-u. U liniji 1 potrebno je iz modula **borneo** unijeti klasu **SignatureProvider** (slika 6.). Klasi je potrebno nekoliko parametara kako bismo uspostavili vezu s aplikacije na uslugu. Te parametre možemo pronaći na Oracle Cloud-u. Prvi parametar **tenant_id** naći ćemo ako kliknemo na ikonu označenu sa brojem 1 na slici 7. Zatim će nam se prikazati padajuća lista gdje ćemo kliknuti na gumb „**Tenancy**“ (broj 2 na slici 7.). Kopirat ćemo „**tenancy ID**“ (slika 8.) i zalijepiti ga kao parametar u kod (broj 1 na slici 6). Sljedeći ID također ćemo pronaći na padajućoj listi kad kliknemo na ikonu profila u gornjem desnom kutu. Biramo gumb **„User settings**“ (broj 3 na slici 7.). Opet kopiramo sadržaj kao na slici 9. te zalijepimo pod parametar **user_id** u Pythonu (broj 2 na slici 8.). Parametar **fingerprint** nalazi se ispod podataka o korisniku. Kliknemo na opciju **API Key**s na lijevoj navigacijskoj traci gdje će nam se prikazati traženi „fingerprint“ koji moramo kopirati u parametar pod brojem 4 u Pythonu (slike 8. i 10.). Zadnji parametar koji nam je ostao odnosi se na privatni ključ. Njega treba generirati tako što ćemo prvo kliknuti na gumb „**Add API Key**“. Otvara nam se dijaloški okvir za dodavanje novog ključa (slika 11.). Izaberemo opciju „**Generate API Key Pair**“ te kliknemo na gumb „**Download Private Key**“. Ključ možemo spremiti bilo gdje na računalo, a u našem slučaju spremili smo ga u mapu RentACar s ostalim datotekama s primjerima. No, tu još nismo gotovi. Sljedeći korak je generiranje ključa u **Windows PowerShell**-u koristeći naredbu **openssl genrsa -out privatekey.pem 2048**. U parametar **private_key** stavljamo generirani privatni ključ zajedno s njegovom putanjom. Kad smo stavili sve potrebne nam parametre za klasu SignatureProvider moramo još stvoriti nekoliko objekata. U objekt **regions** spremit ćemo pristupnu točku regionalne mreže prema Oracle Cloud-u gdje se nalazi baza s kojom ćemo se povezati [13]. U našem slučaju parametar regije je **EU_ZURICH_1**. Kreirat ćemo objekt **handle** koji će sadržavati sve što smo do sad kreirali te će poslati zahtjev na uslugu. Time smo završili sve postavke, pa tako u nastavku možemo raditi upite nad bazom.



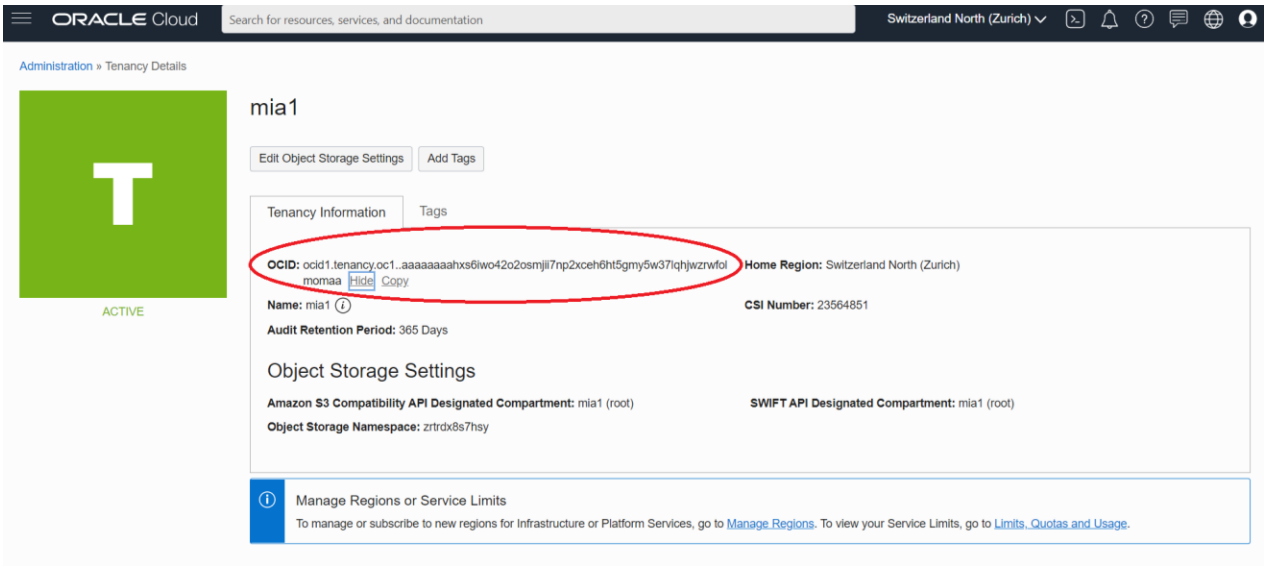
Slika 5. Mapa RentACar sa svojim datotekama

```
from borneo.iam import SignatureProvider
from borneo import NoSQLHandleConfig, Regions, NoSQLHandle

at_provider = SignatureProvider(
    1 tenant_id='ocid1.tenancy.oc1..aaaaaaaahxs6iwo42o2osmjii7np2xceh6ht5gmy5w37lqhjwzrwfolmomaa',
    2 user_id='ocid1.user.oc1..aaaaaaaa4lclgrmctruprix5ebbhcwpjzqzryzy5rnu5ummsj44rnwai7ppq',
    3 private_key='privkey.pem',
    4 fingerprint='b3:f5:0e:cf:de:e8:39:a6:34:cf:a6:65:99:ef:d2:75'
)
region = Regions.EU_ZURICH_1
config = NoSQLHandleConfig(region, at_provider)
handle = NoSQLHandle(config)
```

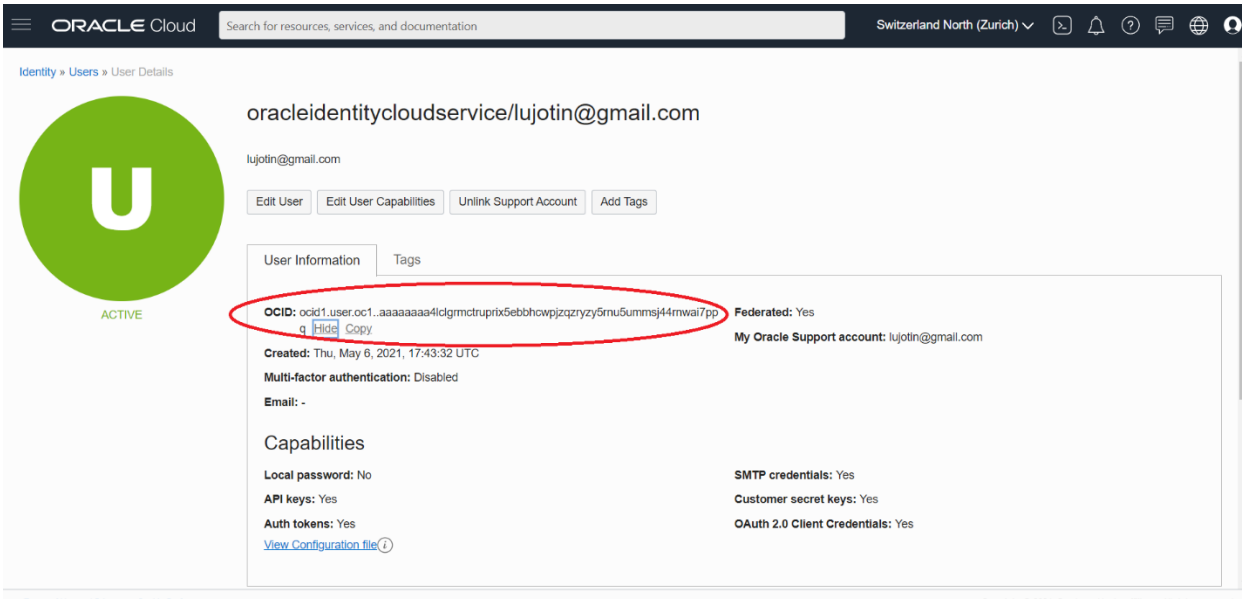
Slika 6. Kod u Pythonu za povezivanje aplikacije sa Oracle Cloud-om



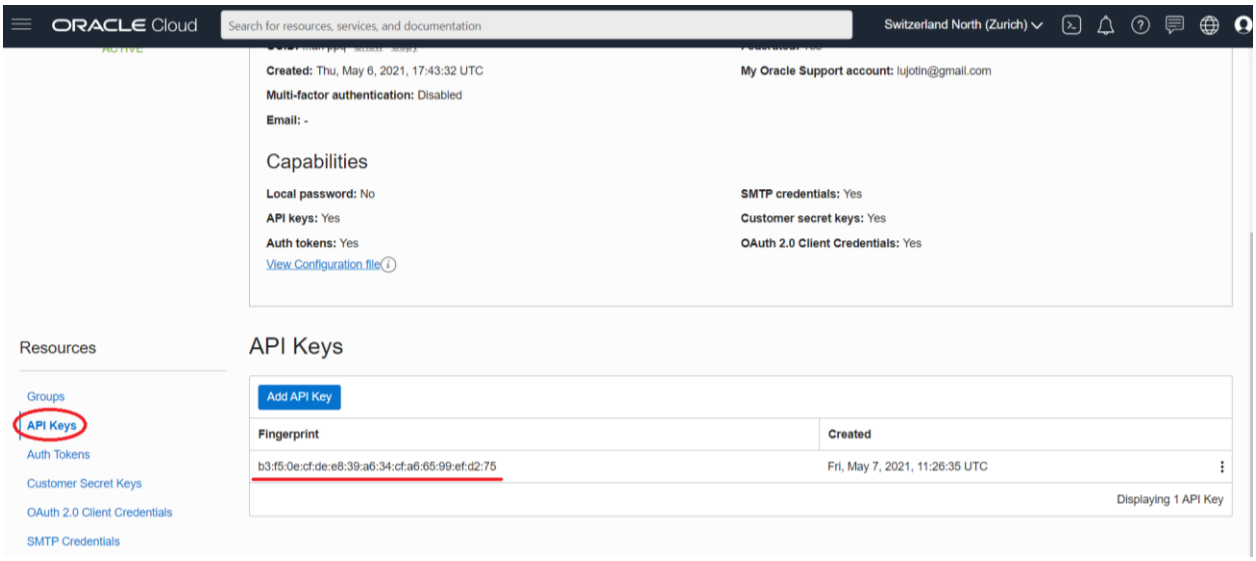
Slika 7. Prikaz Oracle Cloud korisničkog sučelja



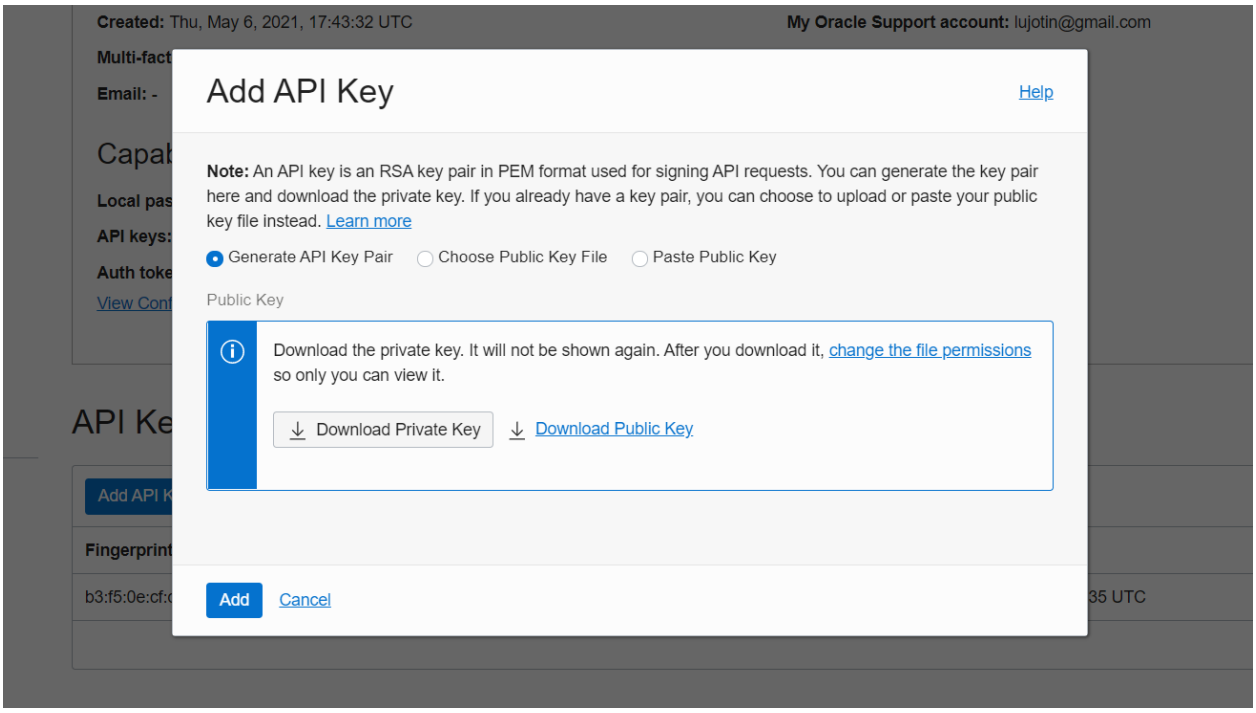
Slika 8. Tenancy ID



Slika 9. User settings



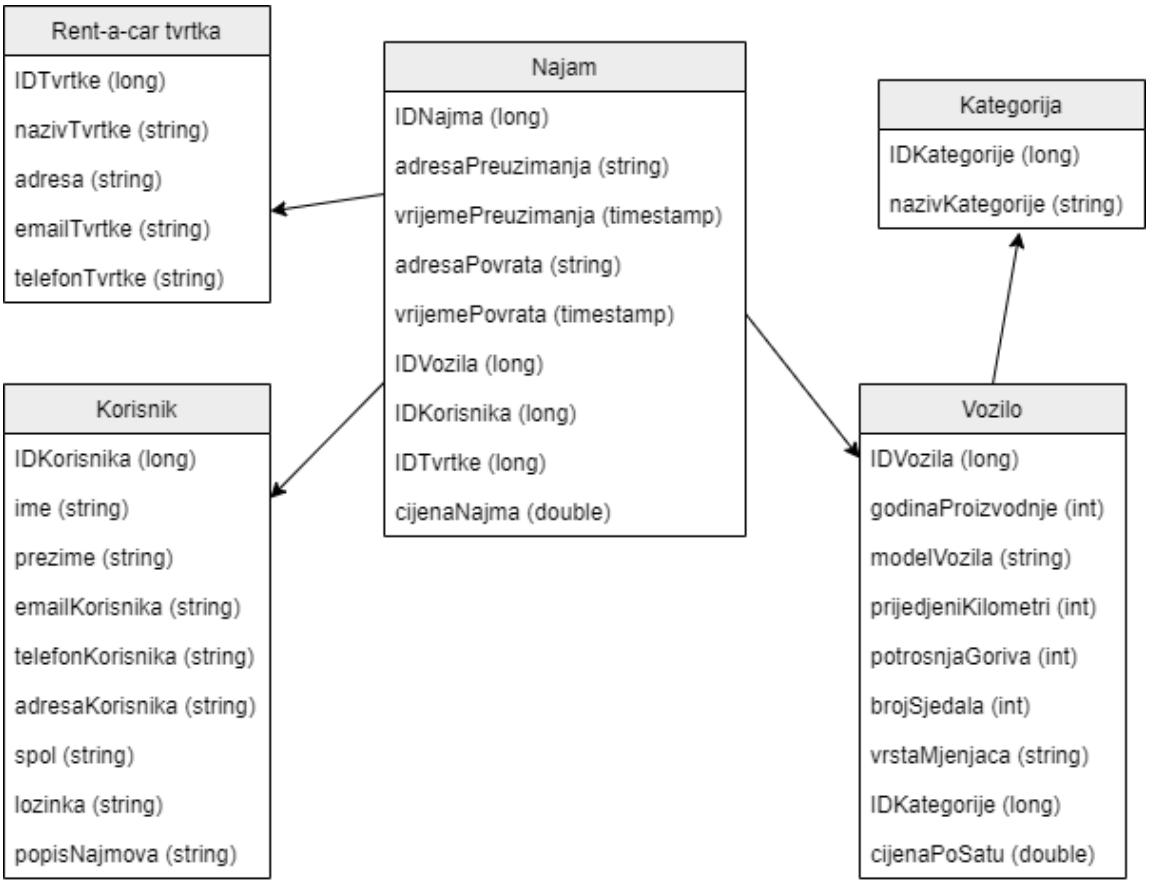
Slika 10. Fingerprint



Slika 11. Dijaloški okvir za dodavanje novog API ključa

3.4.2. Model baze podataka

Kao model za izradu praktičnih primjera u Oracle NoSQL Database Cloud Service-u uzeta je tvrtka za iznajmljivanje vozila. Sastoji se od pet tablica. Tablice su zapravo kolekcije redova gdje svaki red sadrži zapis podataka. Svaka od tih tablica ima polja za primarni ključ i polja s ostalim podacima. Na slici 12. možemo vidjeti svih pet tablica te njihove atribute odnosno u Oracle Cloud-u zvat ćemo ih stupci. Baza podataka će prikupljati podatke o najmu vozila. Tablica **najam** stoga ima stupce **IDNajma (long), adresaPreuzimanja, vrijemePreuzimanja, adresaPovrata (string), vrijemePovrata (timestamp), IDVozila (long), IDKorisnika (long), IDTvrtke (long) i cijenaNajma (double)**. Sadržavati će sve informacije kako bi znali koje je vozilo unajmljeno, kada i gdje te tko ga je iznajmio. Da bi znali o kojem se korisniku, vozilu i tvrtki za najam radi potrebne su nam još tablice za njih. Tablica **korisnik** sadrži **IDKorisnika (long), ime (string), prezime (string), emailKorisnika (string), telefonKorisnika (string), adresaKorisnika (string), spol (string), lozinka (string)** i **popisNajmova (string)**. Tablica **RentACarTvrtka** ima **IDTvrtke (long), nazivTvrtke (string), adresa (string), emailTvrtke (string)** i **telefonTvrtke (string)**. Zatim nam treba još tablica s podacima o **vozilu** koja ima stupce **IDVozila (long), godinaProizvodnje (int), modelVozila (string), prijedjeniKilometri (int), potrosnjaGoriva (int), brojSjedala (int), vrstaMjenjaca (string), IDKategorije (long) i cijenaPoSatu (**double). Svako vozilo spada u neku kategoriju, pa je tako još napravljena tablica **kategorija** koja ima **IDKategorije (long)** i **nazivKategorije (string)** za svaku od kategorija.



Slika 12. Model baze podataka

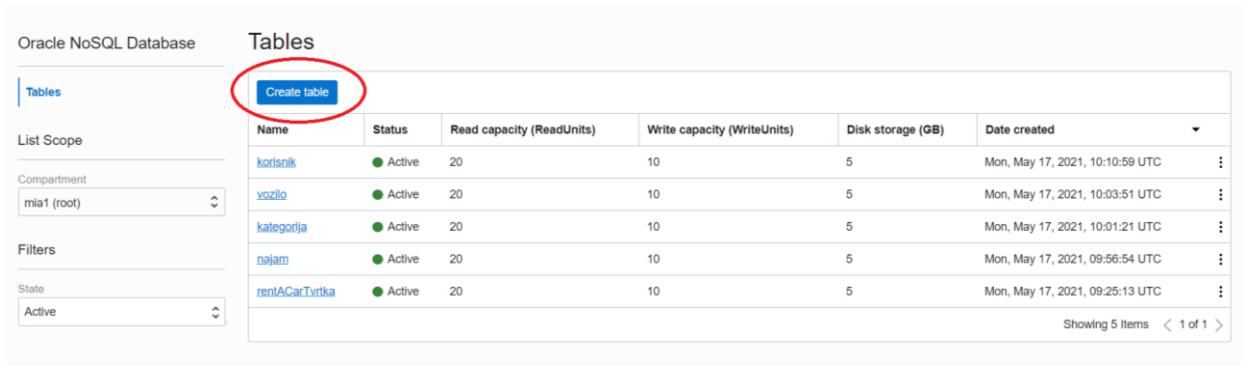
3.4.3. CRUD upiti

CRUD je skraćenica za CREATE, READ, UPDATE i DELETE (kreiranje, čitanje, ažuriranje i brisanje). Spada u jednostavne upite koji se mogu koristiti pri upravljanju bazom podataka. U sljedećim primjerima biti će prikazano kako oni izgledaju. Sve jednostavne upite moguće je napraviti u Oracle NoSQL Database Cloud Service-u ili u samoj aplikaciji koristeći Python programski jezik.

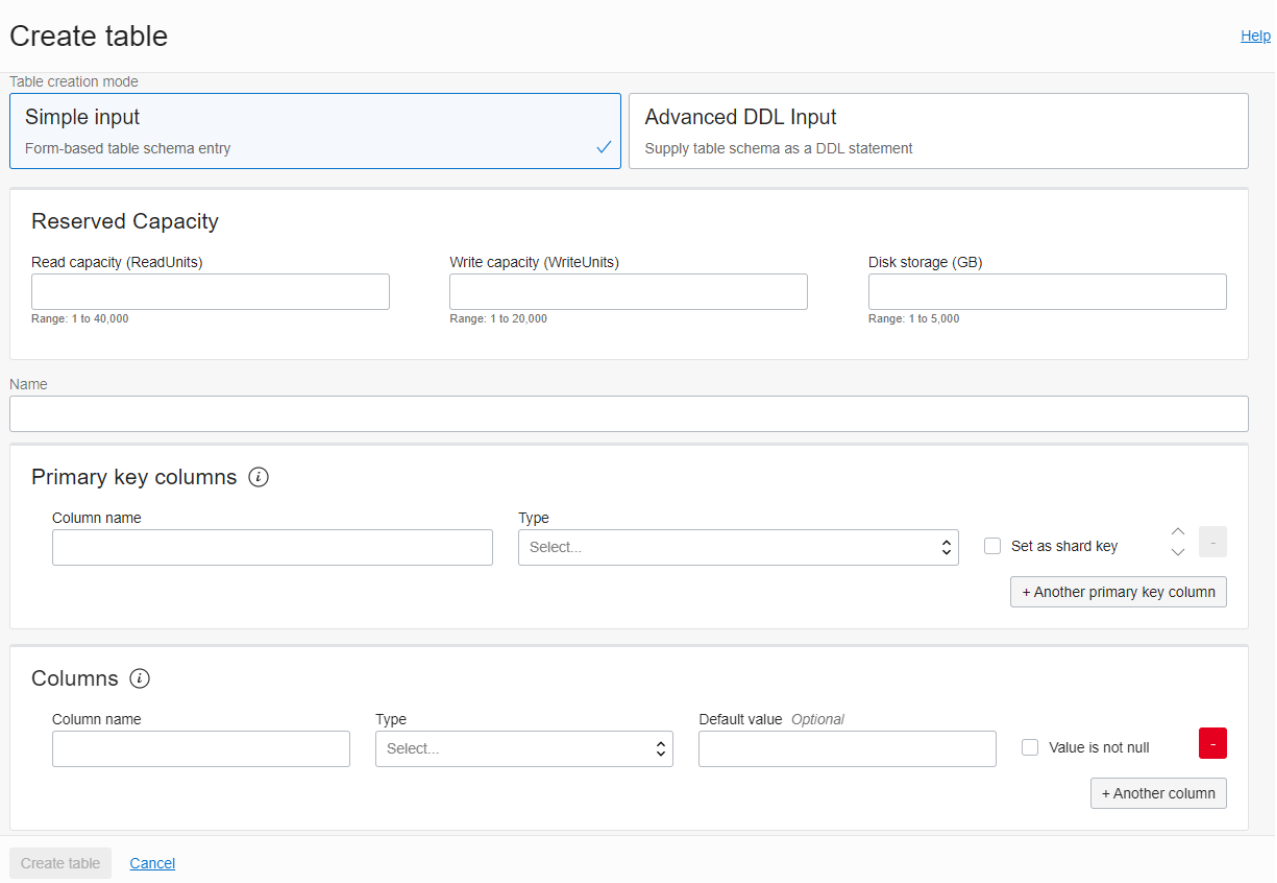
1. CREATE

Kreiranje tablica u Oracle NoSQL Database Cloud Service-u

Klikom na plavi gumb „Create table“ iz slike 13. otvara se novi dijaloški okvir za unos naziva i tipova polja pojedine tablice kao na slici 14. Tablici je potrebno odrediti kapacitet dodjeljujući vrijednost jedinica za čitanje i pisanje te pohrane na disku u GB. Zatim se unosi naziv i tip polja primarnog ključa. Primarni ključ je moguće još označiti kao „shard“ ključ ili dodati još jedan primarni ključ. Zatim se dodaju ostala polja tablice kojima se također određuje naziv i tip te se nudi mogućnost postavljanja da vrijednosti ne budu null odnosno da će se tražiti obavezan unos vrijednosti u to polje ili postavljanja zadane vrijednosti. Nakon završetka unosi potrebno je kliknuti na plavi gumb „Create table“ kako bi tablica bila kreirana i prešla u aktivno stanje.

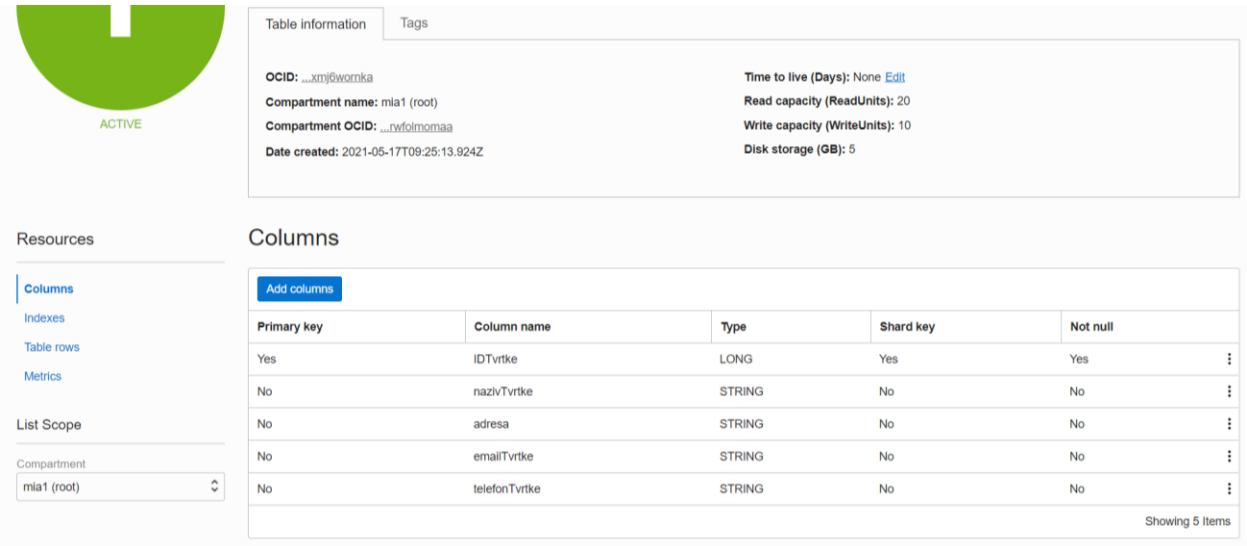


Slika 13. Prikaz svih tablica i gumb „Create table“



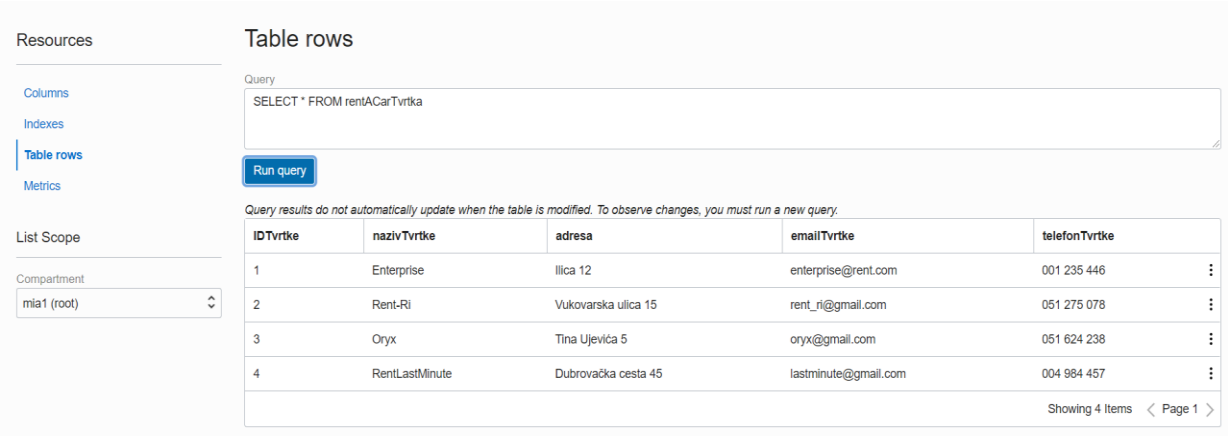
Slika 14. Dijaloški okvir za unos stupaca u tablicu

Klikom na naziv tablice otvaraju se svi podaci o njoj. Na slici 15. pod opcijom „Columns“ možemo vidjeti sve stupce koje smo prethodno kreirali. Za naknadno dodavanje stupaca možemo kliknuti na gumb „Add columns“ iznad prikaza svih stupaca.



Slika 15. Prikaz unesenih naziva stupaca i njihovih tipova podataka u tablicu RentACarTvrtka

Ukoliko odaberemo opciju „Table rows“ na lijevoj navigacijskoj traci prikazuju nam se stupci zajedno s njihovim vrijednostima (slika 16.). Da bi smo unijeli vrijednosti potrebno je koristiti Python.



Slika 16. Opcija ‘Table rows’ i sve unesene vrijednosti tablice rentACArTvrtka

Kreiranje tablica u Pythonu

Na slici 17. prikazan je kod koji je potrebno pokrenuti da bi se kreirala željena tablica. Iz modula **borneo** potrebna nam je klasa **TableRequest** koji će se koristiti za kreiranje, ažuriranje i brisanje tablica. Obavezno mora imati metodu **set_statement()** čiji je parametar varijabla **statement** koja sadrži upit za kreiranje tablice **rentACarTvrtka**. U upitu se navodi naziv tablice koja će se kreirati te unutar zagrade se nalaze nazivi stupaca zajedno sa pripadajućim tipovima podataka. Na kraju se unutar zagrade dodaje parametar **primary key** u čijoj će se zagradi nalaziti naziv prethodno navedenog stupca koji će biti primarni ključ. Potrebno je i uključiti i **TableLimits** koji će se postaviti unutar **TableRequest**-a metodom **set_table_limits()** koja ima tri parametra odnosno vrijednosti jedinica za čitanje, pisanje i limit diska za pohranu u GB. U objekt **request**

spremit ćemo navedeni zahtjev i limite, a u **result** navodimo metodu **do_table_request()** čiji su parametri objekt **request** te vrijeme završetka u milisekundama i interval za operaciju čekanja. Metoda kombinira izvršavanje operacije sa čekanjem na izvršenje. Rezultat metode biti će rezultat zahtjeva tablice.

Slika 17. Primjer kreiranje tablice sa nazivima stupaca i njihovim tipovima podataka u Pythonu

Dodavanje vrijednosti stupcima u Pythonu

Da bismo mogli dodavati vrijednosti stupcima unutar tablice moramo iz modula **borneo** povući klasu **PutRequest** kako bi se dodao jedan red (slika 18.). Ona zahtjeva postavljanje naziva tablice metodom **set_table_name()** kako bi se znalo o kojoj tablici se radi. Ta vrijednost spremit će se u objekt **request** koji će se koristiti zajedno s metodom **set_value** za postavljanje vrijednosti stupaca. Njezini parametri su naziv stupca te njegova vrijednost. Metoda **put()** postavlja red u tablicu. Stvara novi red ili ga postavlja preko postojećeg. Vrijednost koju će koristiti je unutar je objekta **PuRequest** i mora imati primarni ključ i ostala potrebna polja. Da bi se podaci pravilno unijeli moraju odgovarati shemi tablice.

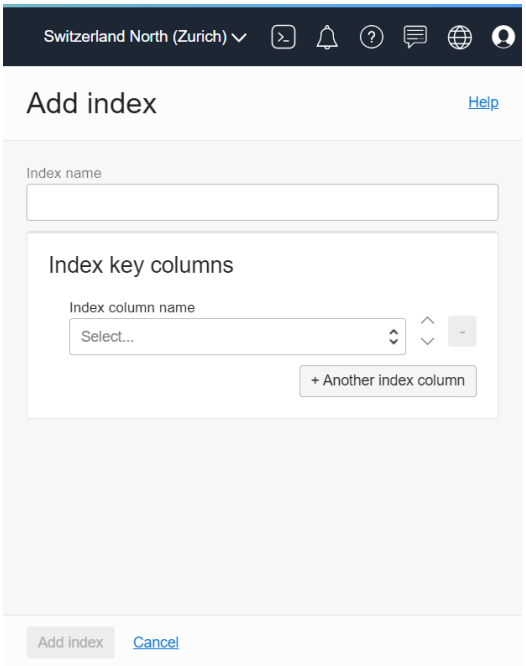
```python
from borneo.iam import SignatureProvider
from borneo import NoSQLHandleConfig, Regions, NoSQLHandle

at_provider = SignatureProvider(
    tenant_id='ocid1.tenancy.oc1..aaaaaaaahxs6iwo42o2osmjii7np2xceh6ht5gmy5w37lqhjwzrwfolmomaa',
    user_id='ocid1.user.oc1..aaaaaaaa4lclgrmctruprix5ebbhcwpjzqzryzy5rnu5ummsj44rnwai7ppq',
    private_key='privkey.pem',
    fingerprint='b3:f5:0e:cf:de:e8:39:a6:34:cf:a6:65:99:ef:d2:75'
)
region = Regions.EU_ZURICH_1
config = NoSQLHandleConfig(region, at_provider)
handle = NoSQLHandle(config)

from borneo import PutRequest

request = PutRequest().set_table_name('kategorija')

request.set_value({'IDKategorije': 10, 'nazivKategorije': 'automobil'})
result = handle.put(request)

```

Slika 18. Primjer dodavanje vrijednosti stupcima tablice u Pythonu

Kreiranje indeksa u Oracle NoSQL Database Cloud Service-u

Indeksiranje je alternativni način dohvaćanja redova tablice. Obično koristimo primarni ključ kako di dohvatili red tablice kojoj pripada taj ključ. Indeksiranjem možemo vrlo učinkovito dohvatiti retke na temelju polja koja nisu dio primarnog ključa. U Oracle Cloud-u indeks možemo dodati tako da odaberemo „**User settings**“ te izaberemo tablicu. Nakon toga biramo opciju '**Indexes**' na lijevoj navigacijskoj traci. Zatim je potrebno kliknuti na gumb „**Add index**“ (slika 19.). Otvorit će se dijaloški okvir za unos imena novog indeksa te odabir naziva stupca tablice nad kojim želimo napraviti indeks (slika 20.).

Slika 19. Opcija „Indexes“ i gumb „Add indeks“



Slika 20. Dijaloški okvir za unos novog indeksa

Kreiranje indeksa u Pythonu

U Pythonu kao i kod kreiranja tablica koristimo varijablu **statement** u koju ćemo spremiti upit. Na slici 21. je primjer upita za kreiranje indeksa nad tablicom **korisnik** za polje **ime**. Kreirani indeks pojavit će se u Oracle Cloud-u pod opcijom „**Indexes**“ (slika 22.).

```
create.py > ...
4   at_provider = SignatureProvider(
5       tenant_id='ocid1.tenancy.oc1..aaaaaaaahxs6iwo42o2osmjii7np2xceh6ht5gmy5w37lqhjwzrwfolmomaa',
6       user_id='ocid1.user.oc1..aaaaaaaa4lclgrmctruprix5ebbhcwpjzqzryzy5rnu5ummsj44rnwai7ppq',
7       private_key='privkey.pem',
8       fingerprint='b3:f5:0e:cf:de:e8:39:a6:34:cf:a6:65:99:ef:d2:75'
9   )
10  region = Regions.EU_ZURICH_1
11  config = NoSQLHandleConfig(region, at_provider)
12  handle = NoSQLHandle(config)
13
14  # table
15  from borneo import TableLimits, TableRequest
16
17  statement = 'CREATE INDEX IF NOT EXISTS imeID1 ON korisnik(ime)'
18  request = TableRequest().set_statement(statement).set_table_limits(
19      TableLimits(20, 10, 5))
20
21  result = handle.do_table_request(request, 40000, 3000)
```

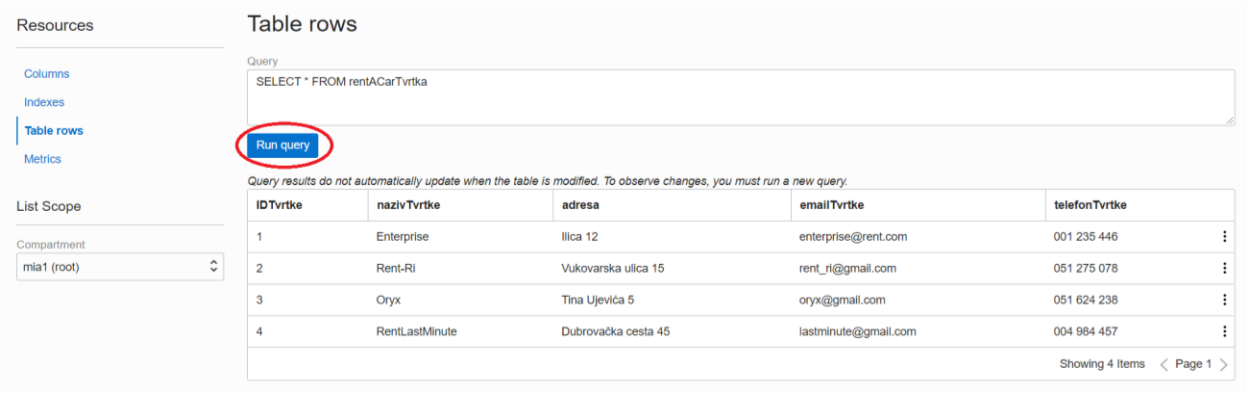
Slika 21. Primjer kreiranja indeksa u Pythonu

Slika 22. Prikaz kreiranog indeksa

2. READ

Ispis svih redaka tablice u Oracle NoSQL Database Cloud Service-u

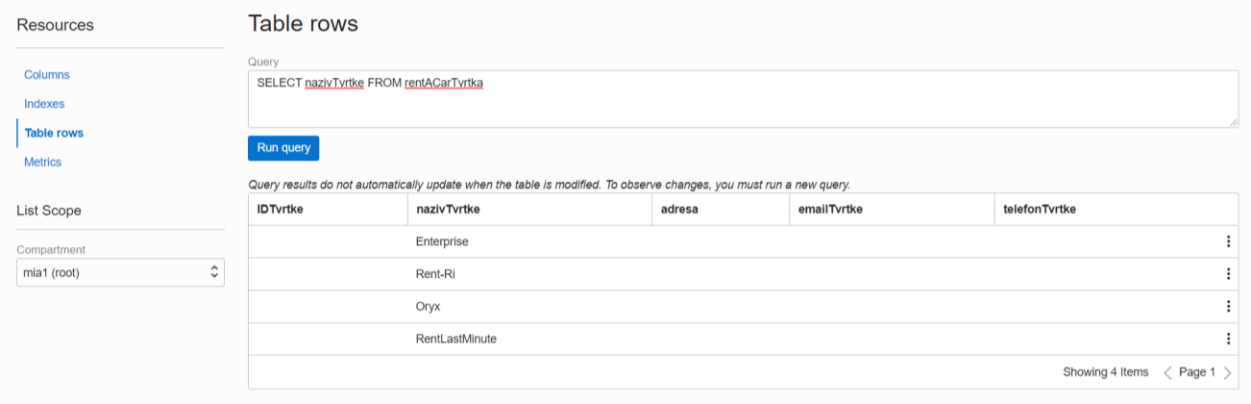
Ukoliko želimo doći do svih unesenih podataka pojedine tablice, morat ćemo u Oracle Cloud-u koristiti SQL upit. Najprije odaberemo tablicu čije podatke želimo dohvatiti. Na lijevoj navigacijskoj traci odaberemo „**Table rows**“. Unutra polja „**Query**“ napišemo upit za ispis svih podataka te kliknemo na gumb „**Run query**“ (slika 23.) kako bi se ispod prikazali podaci.



Slika 23. Primjer ispisa svih redaka tablice u Oracle NoSQL Database Cloud Service-u

Ispis podataka jednog ili više stupaca u Oracle NoSQL Database Cloud Service-u

Ponekad ne želimo dohvatiti sve podatke neke tablice nego samo podatke jednog stupca ili više njih. Unutra polja „**Query**“ izvest ćemo **SELECT** upit tako što ćemo navesti nazive stupaca odvojene zarezom čije podatke želimo dohvatiti. Na primjeru sa slike 24. ispisali smo sve nazive tvrtke za najam vozila koja postoje u tablici navodeći naziv stupca **nazivTvrtke** u upitu zajedno sa nazivom tablice.



Slika 24 .Primjer ispisa podataka jednog stupca u Oracle NoSQL Database Cloud Service-u

Ispis jednog retka tablice u Pythonu

Dohvaćanje pojedinog retka u tablici moguće je izvesti koristeći Python (slika 25.). Biti će nam potrebna klasa **GetRequest** iz modula **borneo**. Potrebno je postaviti naziv tablice sa **set_table_name()** te primarni ključ retka kojeg želimo ispisati sa metodom **set_key()**. Nakon uspješne operacije slanja zahtjeva uz pomoć metode **get_value()** moguće je doći do traženih vrijednosti. Pokretanjem datoteke naredbom **python read.py** ispod u terminalu biti će ispisane vrijednosti retka čiji je primarni ključ jednak onom navedenom u metodi **set_key()**.

```
read.py > ...
1   from borneo.iam import SignatureProvider
2   from borneo import NoSQLHandleConfig, Regions, NoSQLHandle
3
4   at_provider = SignatureProvider(
5       tenant_id='ocid1.tenancy.oc1..aaaaaaaahxs6iwo42o2osmjii7np2xceh6ht5gmy5w37lqhjwzrwfolmomaa',
6       user_id='ocid1.user.oc1..aaaaaaaa4lclgrmctruprix5ebbhcwpjzqzryzy5rnu5ummsj44rnwai7ppq',
7       private_key='privkey.pem',
8       fingerprint='b3:f5:0e:cf:de:e8:39:a6:34:cf:a6:65:99:ef:d2:75'
9   )
10  region = Regions.EU_ZURICH_1
11  config = NoSQLHandleConfig(region, at_provider)
12  handle = NoSQLHandle(config)
13
14  from borneo import GetRequest
15
16  request = GetRequest().set_table_name('rentACarTvrtka')
17
18  request.set_key({'IDTvrtke': 1})
19  result = handle.get(request)
20  print(result.get_value())

TERMINAL  DEBUG CONSOLE  PROBLEMS  OUTPUT                    11: python

PS C:\Users\MM\Desktop\RentACar> & c:/Users/MM/Desktop/RentACar/miaenv/Scripts/Activate.ps1
(miaenv) PS C:\Users\MM\Desktop\RentACar> python read.py
{'IDTvrtke': 1, 'nazivTvrtke': 'Enterprise', 'adresa': 'Ilica 12', 'emailTvrtke': 'enterprise@rent.com', 'telefonTvrtke': '001 235 446'}
```

Slika 25. Primjer ispisa jednog retka tablice u Pythonu

3. UPDATE

Promjena limita jedinica u Pythonu

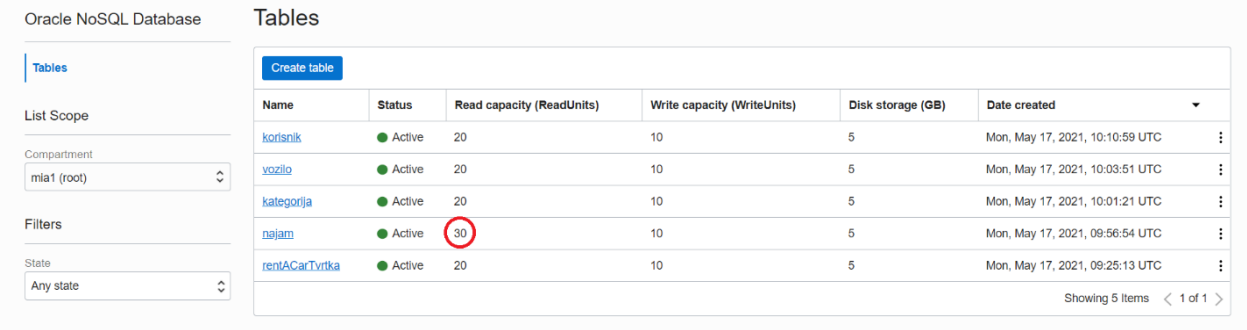
Kako je navedeno i u prethodnim primjerima za ažuriranje podataka potrebno je iz modula **borneo** uključiti **TableRequest**. Ukoliko se ne koristi tipični DDL upit, potrebno je navesti naziv

tablice pomoću metode **set_table_name()** koje se sprema u objekt **request** (slika 26.). Zatim se uz tu varijablu koristi metoda **set_table_limits()** koja se koristi i kod kreiranja tablice u čiju se zagradu unose parametri koji se mijenjaju (limit jedinice čitanja, pisanja i diska za pohranu). U konačnici će promjena limita biti vidljiva u popisu svih postojećih tablica u Oracle cloud-u pored naziva tablice čiji smo limit mijenjali (slika 27.).

update.py > ...

```python
from borneo.iam import SignatureProvider
from borneo import NoSQLHandleConfig, Regions, NoSQLHandle

at_provider = SignatureProvider(
    tenant_id='ocid1.tenancy.oc1..aaaaaaaahxs6iwo42o2osmjii7np2xceh6ht5gmy5w37lqhjwzrwfolmomaa',
    user_id='ocid1.user.oc1..aaaaaaaa4lclgrmctruprix5ebbhcwpjzqzryzy5rnu5ummsj44rnwai7ppq',
    private_key='privkey.pem',
    fingerprint='b3:f5:0e:cf:de:e8:39:a6:34:cf:a6:65:99:ef:d2:75'
)
region = Regions.EU_ZURICH_1
config = NoSQLHandleConfig(region, at_provider)
handle = NoSQLHandle(config)

from borneo import TableLimits, TableRequest

request = TableRequest().set_table_name('najam')
request.set_table_limits(TableLimits(30, 10, 5))
result = handle.table_request(request)

```

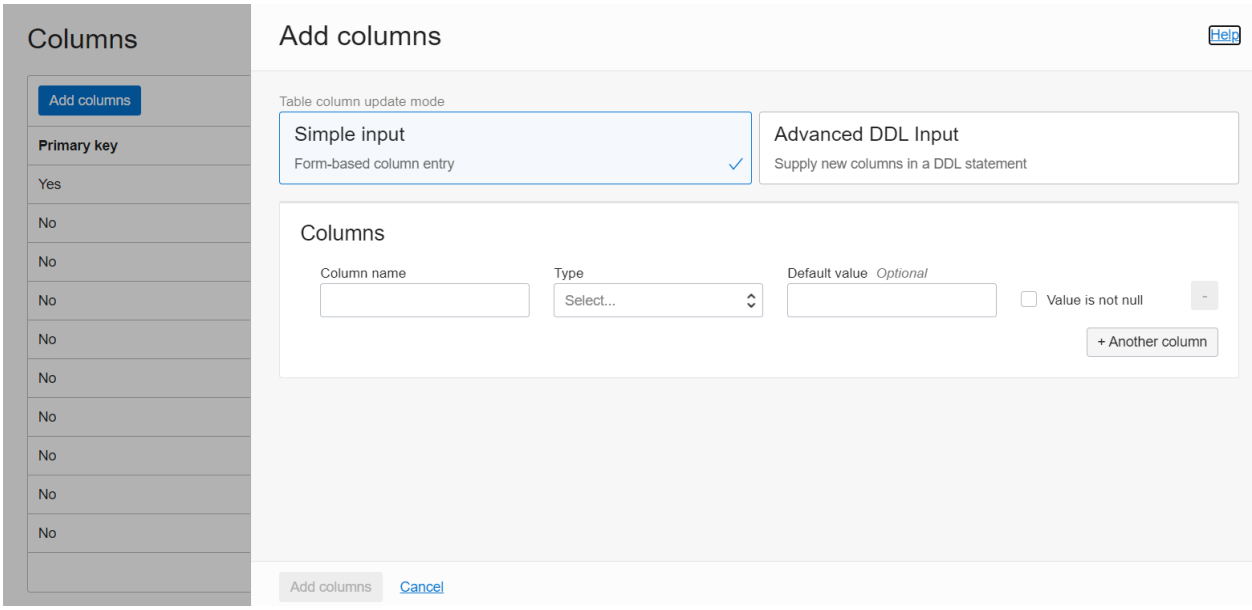
Slika 26. Primjer promjene limita jedinica u Pythonu



Slika 27. Rezultat promjene jedinice čitanja

Dodavanje stupaca u Oracle NoSQL Database Cloud Service-u

Da bismo dodali novi stupac u Oracle Cloud-u potrebno je kliknuti na naziv tablice, a zatim na gumb „**Add columns**“ (broj 1. na slici 30.) gdje se pojavljuje dijaloški okvir za unos podataka o stupcu kao što su naziv stupca, tip podatka, uključivanje opcije „not null“ te postavljanje zadane vrijednosti ukoliko je ta opcija uključena (slika 28.).



Slika 28. Dijaloški okvir za dodavanje novih stupaca u tablicu

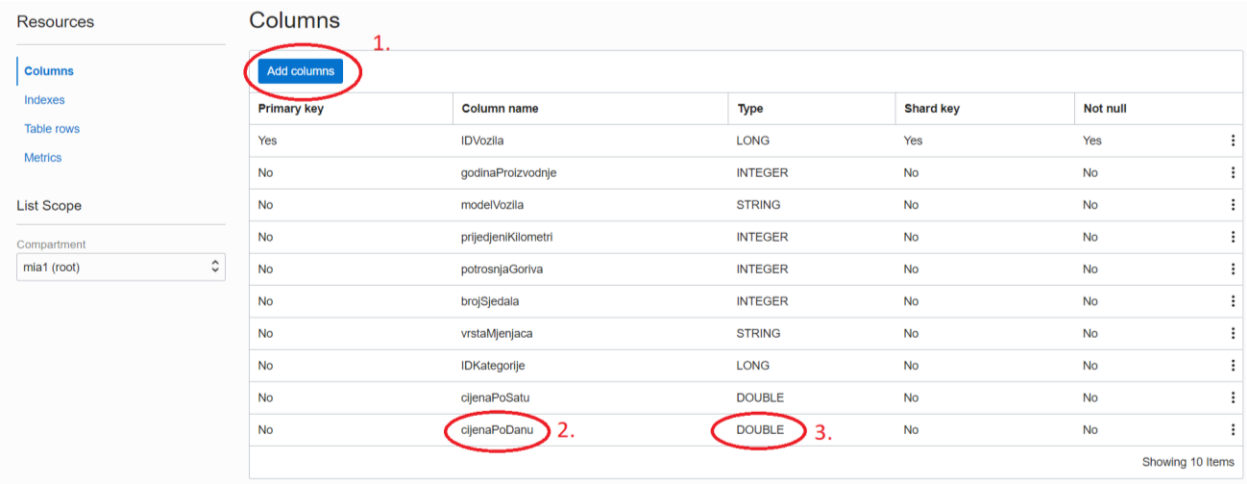
Ažuriranje tablica u Pythonu

Osim promjene limita u Pythonu je moguće i mijenjati ostale podatke u tablici poput dodavanja novih stupaca (slika 29.) ili brisanje postojećih. Za dodavanje stupca koristit ćemo DDL upit koji spremamo pod varijablu **statement**. U upitu navodimo naziv tablice koju ažuriramo, a u zagrade naziv stupca koji dodajemo i njegov tip. U objekt **request** spremamo klasu **TableRequest** koju smo prethodno uključili te metodu uz nju **set_statement()** čiji je parametar varijabla u kojoj je spremljen upit. Naziv stupca i tip podatka koji smo unijeli možemo vidjeti u Oracle Cloud-u kao na slici 30. pod brojevima 2 i 3.

update.py > ...

```
1   from borneo.iam import SignatureProvider
2   from borneo import NoSQLHandleConfig, Regions, NoSQLHandle
3
4   at_provider = SignatureProvider(
5       tenant_id='ocid1.tenancy.oc1..aaaaaaaahxs6iwo42o2osmjii7np2xceh6ht5gmy5w37lqhjwzrwfolmomaa',
6       user_id='ocid1.user.oc1..aaaaaaaa4lclgrmctruprix5ebbhcwpjzqzryzy5rnu5ummsj44rnwai7ppq',
7       private_key='privkey.pem',
8       fingerprint='b3:f5:0e:cf:de:e8:39:a6:34:cf:a6:65:99:ef:d2:75'
9   )
10  region = Regions.EU_ZURICH_1
11  config = NoSQLHandleConfig(region, at_provider)
12  handle = NoSQLHandle(config)
13
14  from borneo import TableRequest
15
16  statement = 'ALTER TABLE vozilo(ADD cijenaPoDanu DOUBLE)'
17  request = TableRequest().set_statement(statement)
18
19  result = handle.do_table_request(request, 40000, 3000)
20
```

Slika 29. Primjer dodavanja stupca tablici u Pythonu

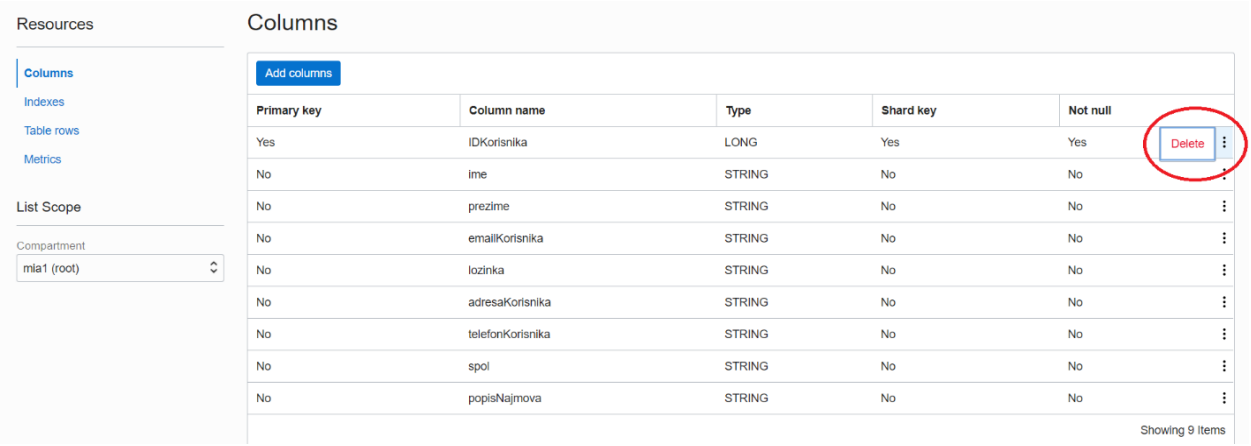


Slika 30. Rezultat dodavanja stupca u Pythonu i gumb „Add columns“

4. DELETE

Brisanje stupca u Oracle NoSQL Database Cloud Service-u

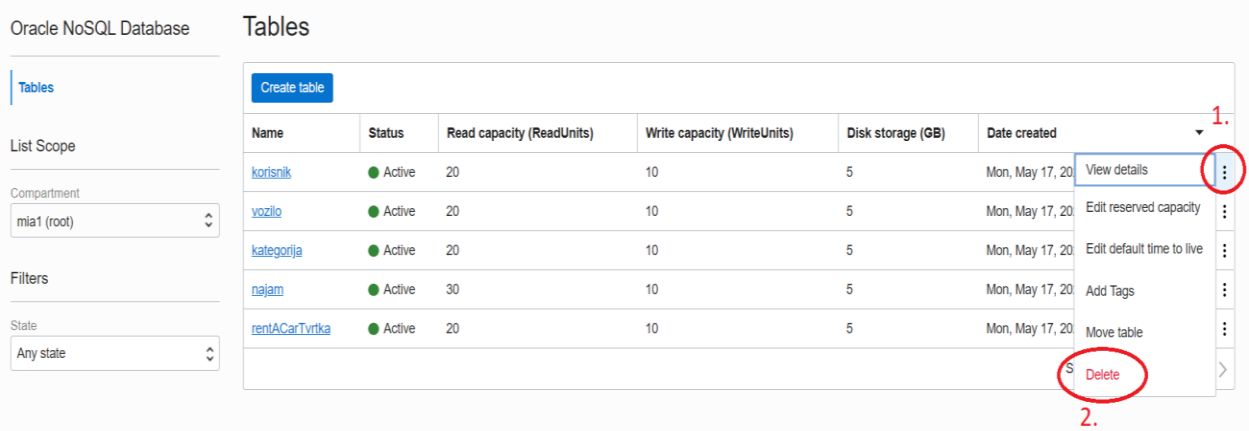
Da bi obrisali stupac u Oracle Cloud-u moramo odabrati stupac tako što kliknemo na njegov naziv. Pod opcijom „**Columns**“ prikazuju se svi stupci tablice gdje se zdesna klikom na tri točkice pored podataka o pojedinom stupcu otvara gumb „**Delete**“ za brisanje tog stupca. (slika 31.)



Slika 31. Primjer brisanja stupaca u Oracle NoSQL Database Cloud Service-u

Brisanje tablice u Oracle NoSQL Database Cloud Service-u

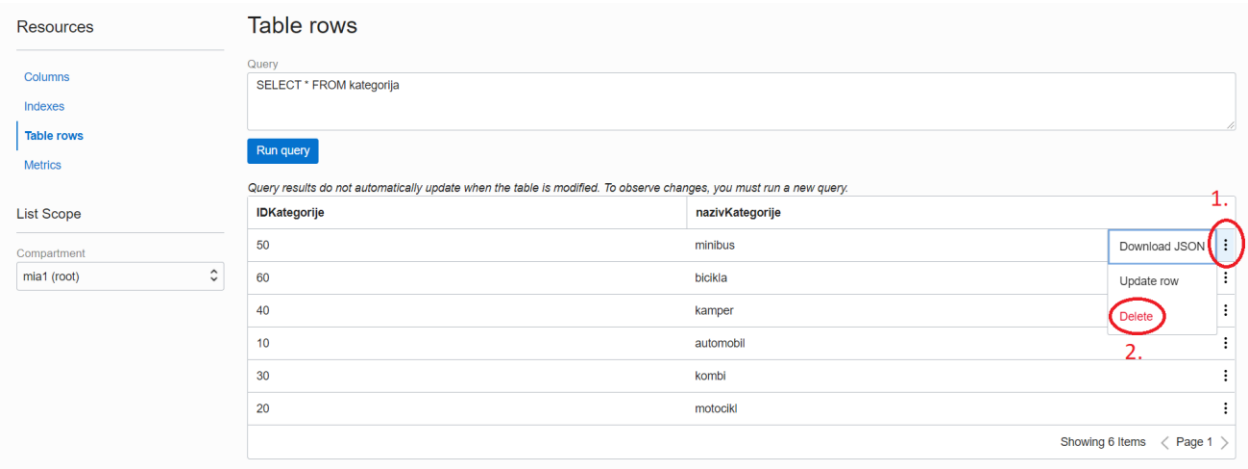
Tablicu je vrlo lako moguće obrisati u Oracle Cloud-u tako što pored naziva stupca na desnoj strani kliknemo na tri točkice (broj 1 na slici 32.) te će nam se ponuditi padajuća lista. Uz razne opcije koje se nude u padajućoj listi, opcija brisanja nalazi se na samom kraju (broj 2 na slici 32.).



Slika 32. Primjer brisanja tablice u Oracle NoSQL Database Cloud Service-u

Brisanje podatka u Oracle NoSQL Database Cloud Service-u

Za brisanje unesenog podatka određene tablice potrebno je kao i kod brisanja stupca prvo odabrati tablicu tako što na popisu svih tablica kliknemo na njezin naziv. Umjesto opcije „**Columns**“ biramo opciju „**Table rows**“. Tako će nam se prikazati svi uneseni podaci (redovi) odabrane tablice. Opet ćemo kliknuti na tri točkice pored reda koji želimo ukloniti te će nam se ponuditi padajuća lista s gumbom za brisanje na dnu (slika 33.).



Slika 33. Primjer brisanja podatka u Oracle NoSQL Database Cloud Service-u

Brisanje stupaca u Pythonu

Kao i kod kreiranja i ažuriranja tablica potrebna nam je klasa **TableRequest**. U varijablu **statement** spremit ćemo DDL upit za brisanje jednog stupca u tablici. Zatim taj upit šaljemo u zahtjevu koristeći sada već poznatu metodu **set_statement()** (slika 34.).

deleteData.py > ...

```python
from borneo.iam import SignatureProvider
from borneo import NoSQLHandleConfig, Regions, NoSQLHandle

at_provider = SignatureProvider(
    tenant_id='ocid1.tenancy.oc1..aaaaaaaahxs6iwo42o2osmjii7np2xceh6ht5gmy5w37lqhjwzrwfolmomaa',
    user_id='ocid1.user.oc1..aaaaaaaa4lclgrmctruprix5ebbhcwpjzqzryzy5rnu5ummsj44rnwai7ppq',
    private_key='privkey.pem',
    fingerprint='b3:f5:0e:cf:de:e8:39:a6:34:cf:a6:65:99:ef:d2:75'
)
region = Regions.EU_ZURICH_1
config = NoSQLHandleConfig(region, at_provider)
handle = NoSQLHandle(config)

from borneo import TableRequest

statement = 'ALTER TABLE vozilo(DROP cijenaPoDanu)'
request = TableRequest().set_statement(statement)

result = handle.do_table_request(request, 40000, 3000)
```

Slika 34. Primjer brisanja stupca u Pythonu

Brisanje podatka u Pythonu

Za brisanje jednog reda u Pythonu koristimo klasu **DeleteRequest** iz modula **borneo** (slika 35.). Brisanje je uspješno samo ako red postoji. Obavezni parametri su ime tablice i primarni ključ reda koji se želi obrisati. U objekt **request** spremit će se ime tablice koje vraća metoda **set_table_name()**. Zatim koristimo metodu **set_key()** koja vraća primarni ključ, a njezini parametri su naziv polja koji se koristi za primarni ključ te njegova vrijednost. Metoda **delete()** kao parametar ima prethodni zahtjev sa potrebnim parametrima. Rezultat metode biti će informacije o obrisanom stupcu.

deleteData.py > ...

```python
from borneo.iam import SignatureProvider
from borneo import NoSQLHandleConfig, Regions, NoSQLHandle

at_provider = SignatureProvider(
    tenant_id='ocid1.tenancy.oc1..aaaaaaaahxs6iwo42o2osmjii7np2xceh6ht5gmy5w37lqhjwzrwfolmomaa',
    user_id='ocid1.user.oc1..aaaaaaaa4lclgrmctruprix5ebbhcwpjzqzryzy5rnu5ummsj44rnwai7ppq',
    private_key='privkey.pem',
    fingerprint='b3:f5:0e:cf:de:e8:39:a6:34:cf:a6:65:99:ef:d2:75'
)
region = Regions.EU_ZURICH_1
config = NoSQLHandleConfig(region, at_provider)
handle = NoSQLHandle(config)

from borneo import DeleteRequest

request = DeleteRequest().set_table_name('kategorija')
request.set_key({'IDKategorije': 50})

result = handle.delete(request)
```

Slika 35. Primjer brisanja podatka u Pythonu

Brisanje tablice u Pythonu

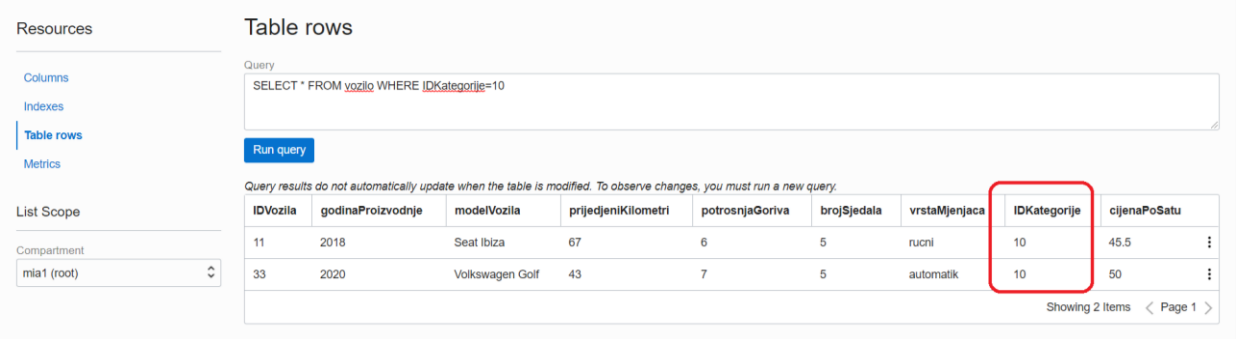
Brisanje tablice također se vrši pomoću DDL upita koji se sprema u varijablu **statement** koji će biti parametar metode **set_statement()** (slika 36.). Te se to sve šalje u zahtjevu.

```python
from borneo.iam import SignatureProvider
from borneo import NoSQLHandleConfig, Regions, NoSQLHandle

at_provider = SignatureProvider(
    tenant_id='ocid1.tenancy.oc1..aaaaaaaahxs6iwo42o2osmjii7np2xceh6ht5gmy5w37lqhjwzrwfolmomaa',
    user_id='ocid1.user.oc1..aaaaaaaa4lclgrmctruprix5ebbhcwpjzqzryzy5rnu5ummsj44rnwai7ppq',
    private_key='privkey.pem',
    fingerprint='b3:f5:0e:cf:de:e8:39:a6:34:cf:a6:65:99:ef:d2:75'
)
region = Regions.EU_ZURICH_1
config = NoSQLHandleConfig(region, at_provider)
handle = NoSQLHandle(config)

from borneo import TableRequest

statement = 'DROP TABLE proba'
request = TableRequest().set_statement(statement)

result = handle.do_table_request(request, 40000, 3000)
```

Slika 36. Primjer brisanja tablice u Pythonu

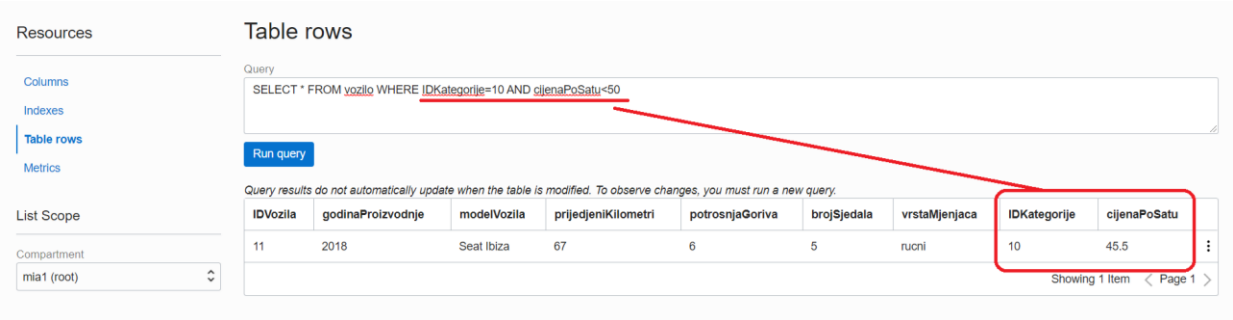
3.4.4. Složeni upiti

Jedna velika razlika između SQL i NoSQL baza podataka je u korištenju složenih upita. Nad NoSQL tablicama nije moguće raditi JOIN upite odnosno spajati dvije ili više tablica. Stoga složenije upite možemo raditi samo nad jednom tablicom u bazi. Oracle NoSQL Database Cloud Service ima opciju pisanja upita koja se može pronaći pod resursom „**Table rows**“ nakon što se odabere tablica s kojom se želi raditi. Unutar praznog polja za unos teksta („**Query**“) potrebno je napisati željeni upit čiji će se razultat ispisati ispod nakon klika na gumb „**Run query**“. Na primjeru tvrtke za najam vozila koristena je klauzula **WHERE** te operator jednako kako bi se ispisala sva vozila čiji je **IDkategorije** jednak 10 odnosno tražila su sve sva vozila koja spadaju pod kategoriju **automobil** (slika 37.). Korišten je **SELECT** upit i znak „*****“ kako bi se prikazali svi nazivi stupaca i njihove vrijednosti koji zadovoljavaju uvjet jednakosti.



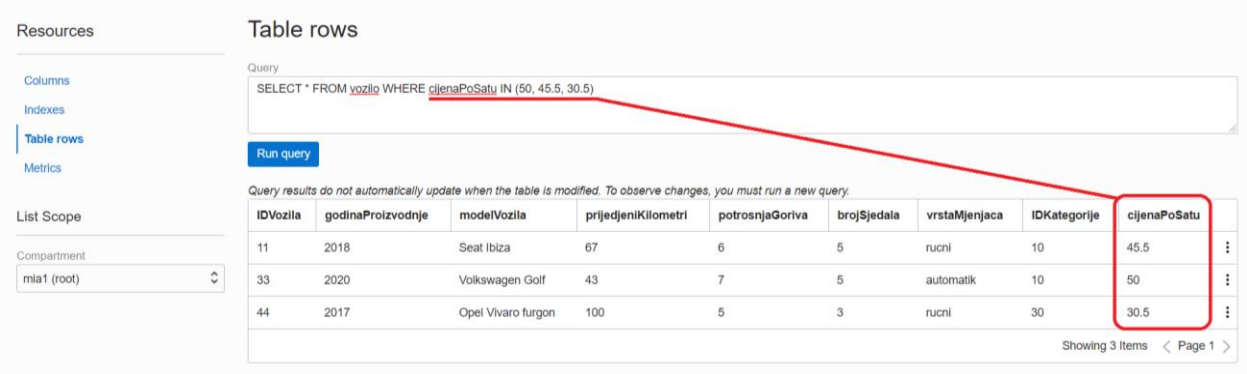
Slika 37. Korištenje WHERE klauzule s operatorom jednako

Moguće je i korištenje logičkih operatora AND, OR i NOT. Na slici broj 38. izvršili smo **SELECT** upit i koristili logički operator **AND** kako bi ispisali vozila koji će zadovoljiti dva uvjeta: moraju imati **IDKategorije** jednak 10 (automobil) i **cijenaPoSatu** mora biti manja od **50** kuna što je navedeno unutar klauzule **WHERE**.



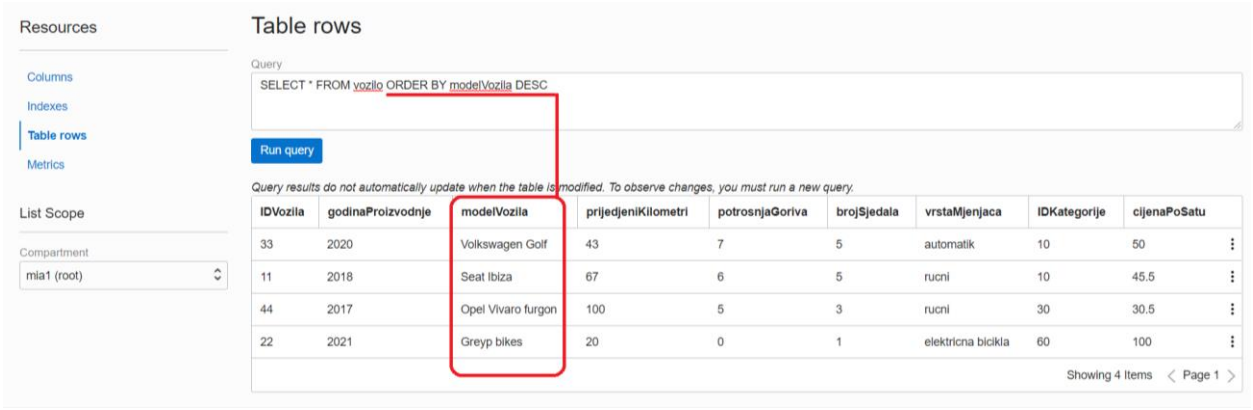
Slika 38 Korištenje WHERE klauzule sa logičkim operatorom AND

Umjesto višestruke upotrebe logičkog operatora OR može se upotrijebiti klauzula **IN** unutar čije se zagrade stavljaju vrijednosti koje se žele pronaći. Na slici 39. u **SELECT** upitu smo zadali da **cijenaPoSatu** mora biti **50, 40.5 ili 30.5** kuna što smo naveli unutar klauzule IN stoga je Oracle Cloud ispisao sve podatke o vozilima koja zadovoljavaju zadani uvjet.



Slika 39. Korištenje IN klauzule

Ukoliko želimo da nam se prilkom ispisa podaci ispisuju u padajućem ili rastućem redoslijedu, potrebno je na kraju **SELECT** upita koristiti **ORDER BY** klauzulu. Unašem primjeru sa slike 40. podatke o vozilima ispisali se po abecedi po modelu vozila, ali u padaćujem redoslijedu što smo rješili stavljajući **DESC** (eng. descending) na samom kraju upita. Za ispisivanje u obrnutom redoslijedu potrebno je samo izostaviti zadnji dio ili staviti ASC (eng. ascending) umjesto prethodno korištenog DESC-a.



Slika 40. Korištenje ORDER BY klauzule sa padajućim redoslijedom

Zaključak

Svatko tko se do sad susreo s bazama podataka, SQL mu je sigurno poznat. SQL se drži čvrste strukture kad su sheme tablica u pitanju te su vertikalno skalabilne što znači da jedino možemo poboljšati perfomanse ulažući u skupe hardvere. SQL je bio odličan jer je smanjio pojavu dupliciranog sadržaja što je pomoglo kod troškova pohrane. No, početkom 2000-tih kada troškovi pohrane nisu više bili skupi pojavila se potreba za novim vrstama baza podataka. Pojavljuje se NoSQL skup tehnologija koji može upravljati podacima velikog obujma. Kreatori riječi NoQSL mislili su na "nerelacijski" ili " ne RDBMS" model te ostali pri tom nazivu. Također neki su htjeli spasiti originalni naziv zamjenjujući ga se "Not Only SQL". Sve u svemu možemo zaključiti kako se termin NoSQL odnosi na sve baze podataka koje ne prate tradicionalna RDBMS pravila te da se odnose na velike podatke kojima se pristupa i upravlja s Weba. Stoga NoSQL nije jedan proizvod ili jedna tehnologija nego kolekcija ponekad različitih, a ponekad i povezanih koncepata o načinu pohrane i upravljanja podacima [16]. Neke od prednosti NoSQL-a su te što se ne drži strogo unaprijed definirane strukture tablica, pa je tako fleksibilnija za izmjene bez ometanja pružanja usluge. Također, nudi mogućnost upravljanja sa raznim vrstama podataka. Kad je riječ o podacima velikog obujma, NoSQL se bez problema nosi s upravljanjem brzine prijenosa, njihovim volumenom itd. Možemo zaključiti kako izbjegavanje praćenja pravila RDBMS-a olakšava implementaciju. S druge strane ima ograničenja kad su upiti u pitanju. Ne mogu se izvoditi neki složeniji upiti poput JOIN-a kao kod SQL-a. Postaje problem upravljati s jedinstvenim ključevima kad se nagomilaju podaci jer tada ključevi postaju kompliciraniji. Novijim developerima teško je shvatiti funkcioniranje NoSQL-a, pa je potrebno veliko iskustvo da bi se donijela dobra odluka hoće li se koristi relaicjske ili nerelacijske baze podataka. Ukoliko su odabir nerelacisjke baze, NoSQL, treba opet birati između široke lepeze vrsta baza koje se nude poput stupčanih, ključ-vrijednost, dokumentnih i graf baza podataka. Kako postoji više vrsta NoSQL baza, tako imamo i puno proizvoda. Neki od njih su MongoDB, Cassandra, DynamoDB, Redis, Oracle itd.

U ovom završnom radu korištena je Oracle NoSQL Database kojoj se moglo pristupiti na Oracle Cloud Service-u uz probno besplatno razdoblje od mjesec dana. To je usluga koja podržava ključ-vrijednost vrste baza podataka garantirajući fleksibilne transakcije. Nudi bazu kojom je moguće potpuno upravljati i koja je dizajnirana za operacije koje zahtijevaju predvidljiv i brz odgovor na jednostavne upite. Prikladna je za aplikacije poput IoT-a (Internet of Things), online kupovine i slično [5]. Oracle nudi veliki izbor tipova podataka koji se mogu koristiti. Svakoj tablici može se dodijeliti primarni ili „shard" ključ ovisno o potrebi te je moguće odrediti veličine jedinica pisanja, čitanja i kapaciteta pohrane za pojedinu tablicu. U radu je prikazano kako se nakon kreiranja računa na Oracle Cloud- može povezati baza s vlastitom aplikacijom na lokalnom računalu. Kroz praktične primjere za model podataka tvrtke za najam vozila prikazani su neki od jednostavnijih CRUD upita te malo složenijih. Za upite je osim Oracle Cloud-a korišten i programski jezik Python. Alat se pokazao vrlo jednostavnim za korištenjem. No, rezultat tomu su zapravo i sami jednostavni upiti koji se izvršavaju nad tablicama. Tako dolazi do izražaja jedna veća razlika između NoSQL-a i SQL-a, a to je nemogućnost spajanja više tablica ili nemogućnost korištenja nekih klauzula za filtriranje podataka u tablicama (npr. LIKE). Može se zaključiti kako je Oracle NoSQL Database vrlo sličan relacijskim bazama što se dizajna tablica tiče osim što su dodane značajke poput jedinica za određivanje kapaciteta pohrane. Svaki red u tablici identificiran je ključem te ima vrijednost neke proizvoljne veličine. Oracle NoSQL Database potpuno je upravljiva usluga, pa se developeri mogu više posvetiti stvaranju inovativnih aplikacija umjesto da se opterećuju upravljanjem servera, pohrane i mrežne infrastrukture.

Podaci svakodnevno rastu i već su dosegli ogroman volumen stoga će potreba za NoSQL-om još više rasti. Proizvodi će se sve više unaprjeđivati kako bi podnijeli dinamična opterećenja kod zahtjevnih aplikacija. S podacima velikog obujma čini se teško upravljati no, NoSQL ovdje znatno olakšava stvari te njegova fleksibilnost kad su modeli podataka i prihvaćanje svih vrsta podataka u pitanju veliki je plus. Smanjeni su i troškovi za upravljanje takvim podacima [20]. Zaključak je da NoSQL nije samo sadašnjost nego i budućnost čemu svjedoči sve veća želja za njegovom implementacijom, a i sami smo svjedoci velikog porastu broja aplikacija u svakom spektru naših života. Oracle je našao svoje mjesto na takvom tržištu te svakodnevno unaprjeđuje svoje usluge. U zadnjih nekoliko godina razvijao je svoju strategiju u oblaku, pa je tako baza podataka postala dio Oracle-ove infrastrukture u oblaku te je 2020. godine izbacio uslugu Oracle NoSQL Database Cloud Service koja je korištena u ovom radu. Oracle sve više svojih tehnologija prenosi u oblak kako bi slijedio trend omogućavanja baze podataka kao usluge. Cilj je pomoći developerima koji se oslanjaju na infrastrukturu temeljenu u oblaku [21]. Time je Oracle zasigurno dobio svoje mjesto među najpoznatijim i najkorištenijim uslugama bilo riječ o bazama podataka ili nekim od drugih usluga.

Literatura

[1] NoSQL Tutorial: Types of NoSQL Databases, What is & Example. guru99. Preuzeto 19.4.2021. sa https://www.guru99.com/nosql-tutorial.html

[2] Oracle NoSQL Database Cloud Service, oracle. Preuzeto 3.3.2021. sa https://www.oracle.com/database/nosql-cloud.html

[3] Vaish, Gaurav, Getting Started with NoSQL. Packt Publishing Ltd, Birmingham (UK), 2013.

[4] Oracle NoSQL Database - Use Case, oracle. Preuzeto 4.3.2021. sa https://www.oracle.com/a/ocom/docs/database/oracle-nosql-use-cases.pdf

[5] Oracle NoSQL Database Cloud Service: About the Service, Oracle. Preuzeto 4.3.2021. sa https://docs.oracle.com/en/cloud/paas/nosql-cloud/csnsd/nosql-database-cloud.html#GUID-88373C12-018E-4628-B241-2DFCB7B16DE8

[6] Oracle NoSQL Database Cloud Service: Key Features, oracle. Preuzeto 4.3.2021. sa https://docs.oracle.com/en/cloud/paas/nosql-cloud/csnsd/key-features.html

[7] Oracle NoSQL Database Cloud Service: Cloud Concepts, Oracle. Preuzeto 5.3.2021. sa https://docs.oracle.com/en/cloud/paas/nosql-cloud/csnsd/cloud-concepts.html

[8] Oracle Cloud Infrastructure Documentation: Table Design, Oracle. Preuzeto 13.6.2021. sa https://docs.oracle.com/en-us/iaas/nosql-database/doc/table-design.html#GUID-833B2B2A-1A32-48AB-A19E-413EAFB964B8

[9] Oracle NoSQL Database Cloud Service: Estimating capacity, Oracle. Preuzeto 5.3.2021. sa https://docs.oracle.com/en/cloud/paas/nosql-cloud/csnsd/estimating-capacity.html

[10] Latencija (elektronika), Wikipedia. Preuzeto 4.3.2021. sa https://hr.wikipedia.org/wiki/Latencija_(elektronika)

[11] TechAmerica Foundation paper (2012). Demystifying Big Data: A practical guide to transforming the business of Government. Preuzeto 5.3.2021. sa file:///D:/IoT/Razvoj%20web%20aplikacija/Demistyfying%20Big%20Data.pdf

[12] Oracle NoSQL Databse Python SDK: Working With Tables, nosql-python-sdk. Preuzeto 15.6.2021. sa https://nosql-python-sdk.readthedocs.io/en/latest/tables.html%23%20

[13] Oracle NoSQL Databse Python SDK : Installation, nosql-python-sdk. Preuzeto 15.6.2021. sa https://nosql-python-sdk.readthedocs.io/en/latest/installation.html

[14] Oracle Cloud Infrastructure Documentation : NoSQL Database, Oracle. Preuzeto 19.5.2021. sa https://docs.oracle.com/en-us/iaas/nosql-database/index.html

[15] NoSQL Databases : Defined and Explained, Scnsoft. Preuzeto 25.5.2021. sa https://www.scnsoft.com/blog/nosql-databases

[16] Tiwari Shashank, Proffesional NoSQL, John Wiley & Sons, Kanada, 2011.

[17] HTTPS, Wikipedia. Preuzeto 14.6.2021. sa https://hr.wikipedia.org/wiki/HTTPS

[18] Representational state transfer, Wikipedia. Preuzeto 14.6.2021. sa https://en.wikipedia.org/wiki/Representational_state_transfer

[19] XML, Wikipedia. Preuzeto 14.6.2021. sa https://hr.wikipedia.org/wiki/XML

[20] SQL vs NoSQL Databases- Is NoSQL the future of Databases, Huawei. Preuzeto 15.6.2021. sa https://forum.huawei.com/enterprise/en/sql-vs-nosql-databases-is-nosql-the-future-of-databases/thread/660677-893

[21] Oracle NoSQL comes to the cloud, TechTarget. Preuzeto 3.7.2021. sa https://searchdatamanagement.techtarget.com/news/252481104/Oracle-NoSQL-database-comes-to-the-cloud

Popis slika

Popis tablica