NoSql baze podataka

Hanuljak, Ivan

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

Ivan Hanuljak

NoSQL BAZE PODATAKA

Završni rad

Pula, 2020.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike

Ivan Hanuljak

NoSQL BAZE PODATAKA

Završni rad

JMBAG: 0303061325, redovni student
Studijski smjer: Sveučilišni preddiplomski studij informatike
Predmet: Baze podataka II
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija
Mentor: doc. dr. sc. Goran Oreški

Pula, kolovoz, 2020.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Ivan Hanuljak, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

_____

U Puli, _____

IZJAVA

o korištenju autorskog djela

Ja, Ivan Hanuljak dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom NoSQL baze podataka koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, _____

Potpis

_____

SADRŽAJ

1. UVOD

Baze podataka danas snažno obilježavaju naš svakodnevni život, razne životne aktivnosti, kao i poslovanje na međunarodnoj razini. Značaj baza podataka danas je najjednostavnije potvrditi činjenicom da se svakodnevno ljudi diljem svijeta redovno susreću s bazama podataka, odnosno procesima, uslugama i aktivnostima koje su njima podržane.

Svaki proces obrade i pohrane te ponovnog korištenja podataka zasniva se na postojanju određene baze. Pri tome ti podaci mogu biti brojčani i znakovni, no treba spomenuti i fotografije, video uratke i sve ostale vrste podataka koji se danas koriste.

S napretkom znanosti, tehnologije i inovacija razvijale su se i baze podataka. Rezultat takvog razvoja je postojanje tradicionalnih i naprednih, suvremenih baza podataka u današnjici. Također, danas se često govori o takozvanim alternativnim bazama podataka, u koje se svrstavaju NoSQL baze. Njihov intenzivniji razvoj i primjena javljaju se nakon pojave Interneta, suvremenog instrumenta komunikacije, a svrha njihova razvoja bilo je maksimiziranje fleksibilnosti i kvalitete u procesu pohrane velikih količina podataka.

Cilj rada je istražiti opće teorijske i praktične osnove u svezi značenja, obilježja i primjene baza podataka. Posebna pažnja posvećuje se argumentiranju njihova suvremenog značaja. Svrha rada je detaljnije analizirati NoSQL baze podataka, što je i središnja problematika rada.

Rad sadrži uvod, zaključak i tri poglavlja. U prvome poglavlju iskazuju se osnovne činjenice o bazama podataka. Sljedeće poglavlje istražuje NoSQL baze podataka identificirajući pri tome njihove osnovne vrste i značajke. Predzaključno poglavlje obrađuje primjenu, odnosno rad s NoSQL bazama podataka, te daje njihovu usporedbu ili komparativnu analizu sa SQL bazama podataka.

Za potrebe istraživanja korištene su metode analize i sinteze, induktivna i deduktivna metoda, metoda komparacije i metoda dokazivanja te apstrakcije. Rad je uređen metodom deskripcije.

2. BAZE PODATAKA

U ovome se poglavlju daje nešto širi uvod u središnju problematiku rada. U tu se svrhu detaljnije pojmovno određuju baze podataka, a poseban naglasak postavlja se na identificiranje njihova značaj, u kontekstu primjene. Naposlijetku se na jednaki način pristupa alternativnim ili suvremenim bazama podataka, koje nose naziv NoSQL.

2.1. POJMOVNO ODREĐENJE I ZNAČAJ BAZA PODATAKA

Baze podataka mogu se definirati na razne načine, no najčešće se koriste neke od osnovnih definicija ovoga pojma. One su predmet istraživanja ovoga rada, a osim pojmovnog određenja središnjeg termina ima za cilj ukazati na njegov značaj. Ono što je važno pri samom početku istaknuti jest da baze podataka danas snažno obilježavaju modernu ekonomiju, globalno društvo i način života, te sve procese koji se svakodnevno odvijaju diljem svijeta.

Kao što i sam naziv indicira, baze podataka ukazuju na višu i kompleksniju razinu rada s raznim vrstama podataka, a uvelike se razlikuju od klasičnih ili tradicionalnih programskih jezika. To je zapravo tehnologija koja se javlja oko 60-ih i 70-ih godina prošloga stoljeća, a od tada do danas snažno se razvija. Već od prvih oblika, baze podataka utjecale su na jačanje produktivnosti, kvalitete i pouzdanosti razvoja aplikacija koje se koriste za pohranjivanje i korištenje podataka u računalima (Sadalage, Fowler, 2013).

Među mnogim definicijama ovoga termina izdvaja se ona koja ističe kako je zapravo riječ o skupu međusobno povezanih podataka, koji su pohranjeni u vanjskoj memoriji računala. Ti su podaci kao takvi dostupni raznim korisnicima i aplikacijskim programima, a bilo kakva aktivnost u svezi njih provodi se korištenjem i podrškom određenog softvera (Oracle, 2020).

Sustav za upravljanje bazom podataka (engl. *Data Base Management System* – DBMS) sljedeći pojam je usko povezan uz korištenje, primjenu ili značaj baza podataka. To je poslužitelj ili server koji oblikuje fizički prikaz baze sukladno

traženoj logičnoj strukturi. Njegova se uloga očituje u obavljanju svih operacija u svezi podataka, koje se traže ili zahtijevaju od strane korisnika. Jednako tako, on služi osiguranju sigurnosti podataka i automatizaciji administrativnih poslovanja s bazom (Oracle, 2020).

Svi podaci u bazi podataka organizirani su logičnim slijedom, a pri tome se koristi neki od mogućih modela podataka, koji predstavlja skup ili niz pravila kojima se određuje izgled strukture baze podataka. Time oni čine osnovu za koncipiranje, projektiranje, ali i korištenje, odnosno implementaciju baze podataka.

2.2. VRSTE BAZA PODATAKA

Danas postoji mnoštvo bazi podataka, što zapravo svjedoči o intenzitetu njihova razvoja tijekom proteklih nekoliko dekada. Sve one podržavaju razne poslovne i ostale funkcije te potrebe suvremenoga društva i ekonomije, ali zadiru i u svakodnevni život. S obzirom na njihova obilježja i funkcionalnost, pronalaze svoju primjenu u raznim segmentima. Osnovne vrste baza podataka prikazuju se Tablicom 1.

Tablica 1. Vrste baza podataka

VRSTA	OBILJEŽJA
Relacijske baze podataka	• Obilježile su razdoblje 80ih godina 20. stoljeća; • Stavke su organizirane kao skup tablica sa stupcima i redovima, a tehnologija omogućuje najučinkovitiji i fleksibilniji način pristupa strukturiranim informacijama.
Objektno orijentirane baze podataka	• Informacije se prikazuju u obliku objekata.
Distribuirane baze podataka	• Imaju dvije ili više datoteka na različitim mjestima; • Baza podataka može biti pohranjena na više računala, smještena na jednom fizičkom prostoru ili raspršena u mreži.
Skladišta podataka	• Služi za brze upite i analize.
NoSQL baze podataka	• Nerelacijska baza podataka namijenjena obradi velikih količina informacija; • Popularnost im se razvija usporedno s razvojem Interneta.
Grafičke baze podataka	• Pohranjuje podatke u smislu entiteta i odnosima između entiteta.
OLTP baze podataka	• Brza i analitička baza podataka dizajnirana za veliki broj transakcija koje izvršava više korisnika.

Izvor: Oracle (2020.) What is Data Base. Dostupno na: https://www.oracle.com/database/what-is-database.html (27.07.2020.).

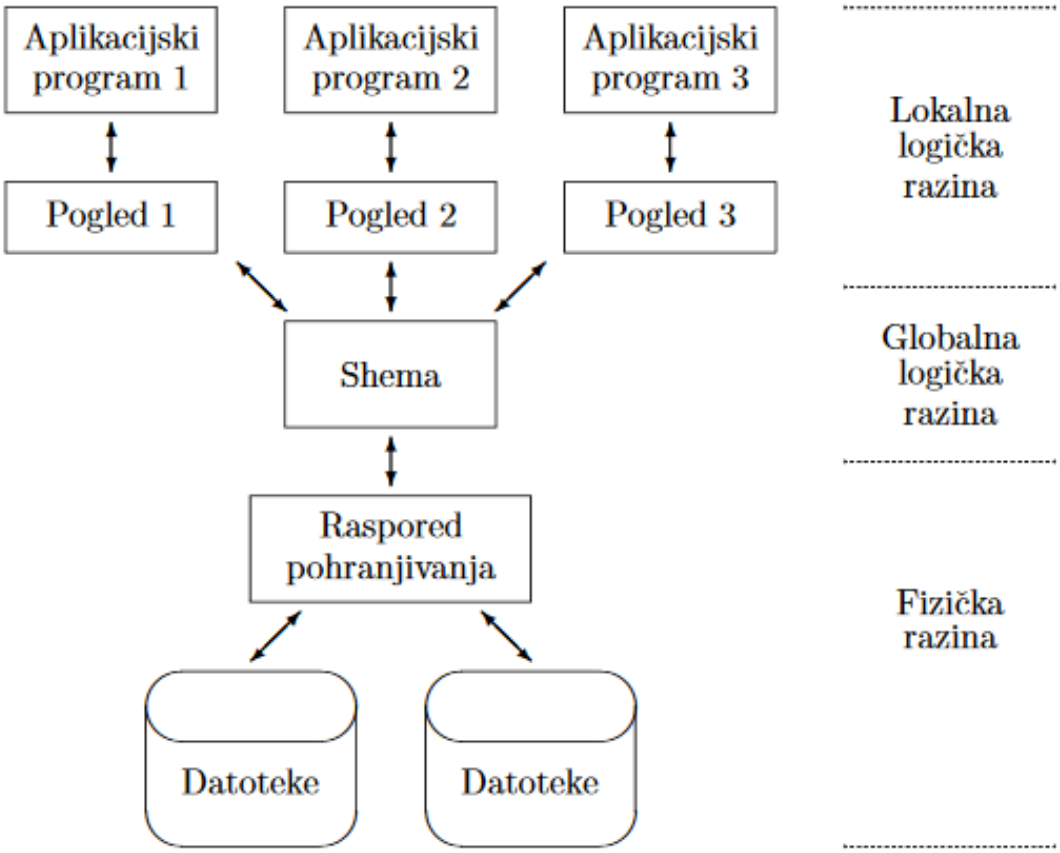
Navedene vrste baza podataka zapravo svjedoče o njihovoj evoluciji, ali i područjima primjene. Predočene vrste predstavljaju tek neke od osnovnih, dok su one ostale uglavnom specifične i prilagođene pojedinim funkcijama, znanstvenim, financijskim i ostalim potrebama.

U suvremeno doba poseban značaj i interes plijene alternativne ili nove baze podataka. Skupine takvih su primjerice sljedeće (Oracle, 2020):

- Baze podataka otvorenog koda – to je sustav baze podataka čiji je izvorni kod otvorenog koda, a neki od primjera su SQL ili NoSQL baze podataka, o kojima više slijedi uskoro;
- *Cloud* baze podataka ili baze podataka u oblaku – predstavljaju skup strukturiranih ili nestrukturiranih podataka, koji se nalaze na privatnoj, javnoj ili hibridnoj platformi računalstva u oblaku;
- Multimodel baza podataka – kao što naziv ukazuje, one integriraju i kombiniraju različite vrste modela baza podataka u jedinstven integrirani zadnji dio. Upravo zbog toga prikladne su za primanje različitih vrsti podataka;
- Baze podataka s vlastitim pogonom – definiraju se kao najnovije i najnaprednije baze podataka, koje imaju vlastito pokretanje, a temelje se na oblaku i koriste strojno učenje za automatizaciju podešavanja baze podataka, sigurnosti, sigurnosnih kopija, ažuriranja i ostalih rutinskih zadataka upravljanja, koje tradicionalno izvode administratori baze podataka.

Neovisno o vrsti baze podataka, sve one imaju sličnu arhitekturu, nešto specifičnijih i različitijih performansi. Ona integrira tri osnovna dijela ili elementa (Slika 1.).

Slika 1. Arhitektura baze podataka



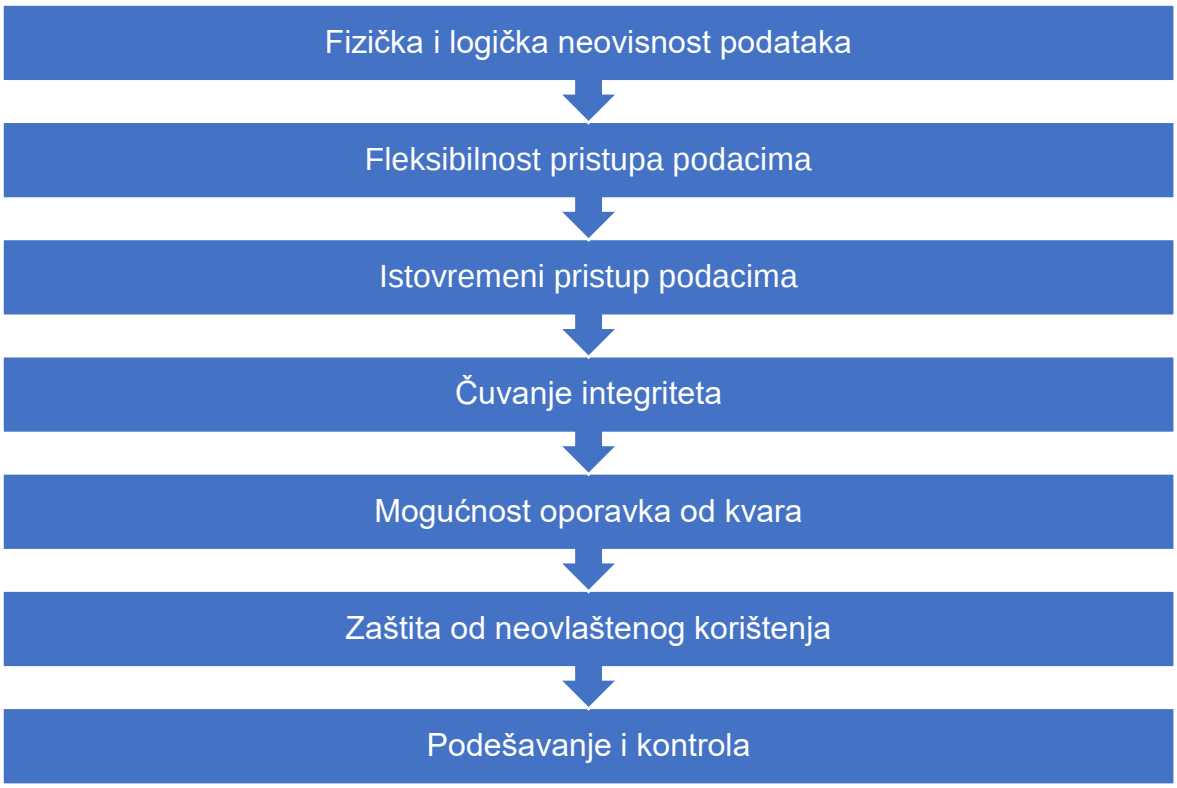
Izvor: Manger, R. (2003.) Baze podataka. Dostupno na: http://jadran.izor.hr/~dadic/EKO/baze-podataka.pdf (27.07.2020.).

Primarno se navodi fizička razina, koju vide samo sistemski programeri. To je fizički prikaz i raspored podataka na jedinicama vanjske memorije. Sljedeći element arhitekture baze podataka je globalna logička razina ili sloj, koja podrazumijeva strukturu baze, odnosno njezinu shemu ili formu. U konačnici se navodi lokalna logička razina, odnosno predodžba o dijelu baze kojeg koristi neka konkretna aplikacija.

2.3. CILJEVI BAZA PODATAKA

Neovisno o vrsti baze podataka koja se u praksi koristi, svrha njezina korištenja je postizanje konkretnih ciljeva. Oni s mogu predočiti na sljedeći način (Slika1.).

Slika 2. Ciljevi korištenja baza podataka



Izvor: Manger, R. (2003.) Baze podataka. Dostupno na: http://jadran.izor.hr/~dadic/EKO/baze-podataka.pdf (27.07.2020.).

Fizička nezavisnost podataka, kao jedan od brojnih ciljeva njihova korištenja, odnosi se na razdvajanje logičke i stvarne fizičke građe baze. Ukoliko se fizička građa promijeni, naprimjer podaci se prepišu na neki drugi disk, to ne bi smjelo zahtijevati promjene u postojećim aplikacijama.

Logička nezavisnost podataka nadovezuje se na prethodno. Ukoliko se promijeni logička definicija baze, kao na primjer uvede se novi zapis, također se upućuje na nepostojanje obveze promjene aplikacije.

Fleksibilnost pristupa podacima podrazumijeva da korisnik u bazama podataka može slobodno prebirati podatke i uspostavljati veze među njima. Istovremeni pristup podacima, kao sljedeći cilj, odnosi se na funkciju baze da omogućuje korisnicima da istovremeno koriste određene podatke, a da pri tome ne smetaju jedan drugome.

Čuvanje integriteta je sljedeći cilj, a njima se ukazuje da se automatski čuvaju korektnost i konzistencija podataka. Osim njega, treba spomenuti i mogućnost oporavka od kvara, odnosno pouzdanu zaštitu u slučaju kvarova i greški hardvera i softvera.

Zaštita od neovlaštenog korištenja odnosi se na postojanje mogućnosti ograničavanja pristupa i prava korištenja podataka. Zadovoljavajuća brzina pristupa podrazumijeva dostatnu brzinu operacija s podacima, u skladu s njihovim obilježjima, potrebama korisnika i funkcijama pojedinih vrsti baza podataka. U konačnici se izdvajaju i mogućnost podešavanja te kontrole, u smislu stalnih briga o performansama, učinkovitosti i efikasnosti.

3. NOSQL BAZE PODATAKA I SQL BAZE PODATAKA

Nakon osnovnih podataka o značenju, ulozi i funkcijama, te vrstama baza podataka, detaljnije se zadire u središnju problematiku rada. U skladu s time analiziraju se NoSQL baze podataka te se daje njihova usporedna analiza sa SQL bazama podataka.

3.1. OSNOVNO O NOSQL BAZAMA PODATAKA

Relacijske baze podataka, o kojima je ukratko bilo riječi u prethodnom dijelu rada, koriste se već gotovo pola stoljeća. Dok su u inicijalnim fazama razvoja predstavljale inovaciju i novitet u računalstvu, danas predstavljaju standardne sustave za rad s podacima, posebice kada je riječ o velikoj količini istih.

Osnovno obilježje ovih baza podataka je utemeljenost na relacijskom modelu podataka. Njihov intenzivniji razvoj i primjena javljaju se s razvojem i popularizacijom Interneta, kao i novih tehnoloških izazova u domeni operacijskih sustava, programskih jezika i softverskih sustava. U suvremeno doba jedan od vodećih izazova je količina podataka koja se koristi u razne svrhe, a koju je teško obrađivati, posebice kada je vrijeme ograničeno za te radnje (Stojanović, 2016).

Nova ograničenja, izazovi i potrebe stvorili su osnovu za stvaranje novih baza podataka, na osnovu kojih se podaci organiziraju na temelju jednostavnijeg i slobodnijeg protoka podataka. Kao takve, ove baze ne zahtijevaju definiciju sheme podataka, a često se nazivaju i alternativnim bazama.

Pojam NoSQL prvi puta se u javnosti ili praksi javlja krajem prošloga stoljeća, a predstavljen je kao baza podataka otvorenog koda. Jedno od njezinih temeljnih odrednica jest da ona zapravo ne koristi SQL kao jezik upita, već se njezino funkcioniranje usmjerava na obradu podataka preko skripti tablica, koje se mogu različito kombinirati. Riječ je o alternativnom načinu pohrane podataka (Sadalage, Fowler, 2013).

Ove baze podataka su dinamičke, što znači da je tip i broj atributa nekog entiteta "otvoren", odnosno moguće ga je po potrebi mijenjati neovisno o utjecaju na ostale podatke. U tome se očituju i osnovne koristi ili pozitivne funkcionalnosti ovih baza podataka.

Korištenje NoSQL baza podataka nam pomaže u jednostavnijem prikazivanju struktura podataka u nekom programskom jeziku za neke slične strukture podataka neke takve baze podataka. Na ovaj se način pojednostavljuje interakcija između baze podataka i aplikacija.

Osim ovih značajki ili karakteristika predmetnih baza podataka, treba spomenuti i njihovu namjenu, te strukturu koja odgovara potrebi da se raspoređuju na veći broj računala gdje svaki korisnik radi odvojeno sa svojom skupinom podataka.

Sumirajući navedeno, može se istaknuti kako ne postoji univerzalna i sveobuhvatna definicija NoSQL baza podataka, već se naglasak postavlja na razmatranje njihovih obilježja ili karakteristika. Osnovne karakteristike ovih sustava su (Sadalage, Fowler, 2013):

- Ne koristi se relacijski model;
- Dobro se izvodi na klasterima;
- Otvoreni izvor;
- Izgrađeno za web potrebe 21. Stoljeća;
- Bez sheme;
- Najvažniji rezultat porasta NoSQL-a je postojanost poliglota.

S obzirom na svoja obilježja, ove baze podataka osiguravaju korisnicima suvremeno, inovativno i kompleksno baratanje podacima, posebice kada je riječ o većim količinama. Od 2009. godine do danas razvilo se više različitih NoSQL rješenja, a svako od njih ima različite pristupe organizaciji podataka, a time i njihovoj obradi. U nastavku rada predstavljaju se neke od osnovnih.

3.2. VRSTE NOSQL BAZA PODATAKA

U ovome dijelu poglavlja obrađuju se četiri osnovne vrste ovih baza podataka, odnosno četiri popularna NoSQL rješenja. To su (Stojanović, 2016):

- Ključ-vrijednost sustavi (engl. *Key-vallue systems*);
- Graf kao model podataka (engl. *Graph as a data model*);
- Dokument baze podataka (engl. *Document databases*);
- Stupičaste baze podataka (engl. *Columnar databases*).

Danas postoji veliki broj komercijalnih rješenja ovih vrsti baza podataka. Neke od popularnijih navode se u nastavku (Tablica):

Tablica 2. Komercijalna rješenja pojedinih vrsti NoSQL baza podataka

VRSTE BAZA PODATAKA	**KOMERCIJALNI PRIMJERI RJEŠENJA**
• Ključ-vrijednost sustavi (engl. *Key-vallue systems*)	• Riak; • Tokyo Cabinet and Kyoto Cabinet; • Redis; • Memcached;
• Graf kao model podataka (engl. *Graph as a data model*)	• InfiniteGraph; • Neo4j; • OrientDB; • Sparksee.
• Dokument baze podataka (engl. *Document databases*)	• MarkLogic; • MongoDB; • CouchDB; • CrateDB.
• Stupičaste baze podataka (engl. *Columnar databases*)	• Cassandra; • HBase;

Izvor: Le, J. (2019.) An Introduction to Big Data: NoSQL. Dostupno na: https://medium.com/cracking-the-data-science-interview/an-introduction-to-big-data-nosql-96b882f35e50 (30.07.2020.).

Detaljnije o svakom od ovih vrsti rješenja slijedi u nastavku.

3.2.1. KLJUČ-VRIJEDNOST SUSTAVI

Ključ-vrijednost sustavi koriste jednostavne modele podataka, koji su nalik rječniku, a sastoje se od parova oblika ključ i vrijednost. Jednom tekstualnom ključu se pridružuje jedna vrijednost bilo kojeg tipa. Za razliku od SQL baza, ove baze podataka nemaju upitni jezik.

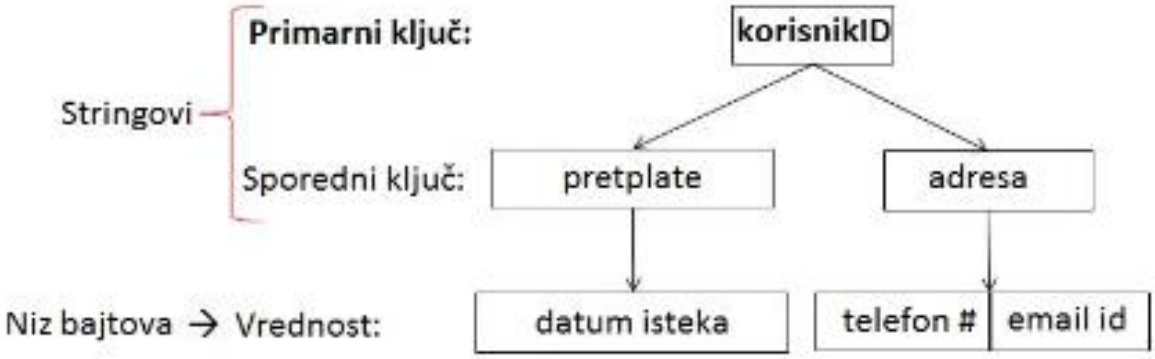
Kao takve one omogućuju tek pronalazak, dodavanje i uklanjanje parova na osnovu ključa i podržavaju tri operacije, odnosno (Stojanović, 2016):

- GET (ključ) – vraća vrijednost pridruženu zadanom ključu;
- PUT (ključ, vrijednost) – dodaje novi par (ključ, vrijednost) ili pridružuje novu vrijednost postojećem ključu;
- DELETE (ključ) – uklanja par (ključ, vrijednost), a ukoliko ključ ne postoji može dojaviti grešku.

U narednom primjeru ključ može biti u različitim formatima, kao što mogu i njegove pridružene vrijednosti biti raznolike. Primjerice one mogu predstavljati slike, zvukove, internetske stranice, razne dokumente i slično. Funkcioniranje ovih baza temelji se na dva osnovna pravila. Svaki ključ koji se nalazi u tablici mora biti jedinstven, a upit može biti baziran samo na ključu, ne i na njegove vrijednosti.

Danas postoje brojni industrijski primjeri ili varijacije ovi baza podataka, a u nastavku se prikazuje jedna od njih (Slika 3.).

Slika 3. Oracle ključ-vrijednost baza podataka



Izvor: Mijalković, S. (2013.) NoSQL baze podataka. Dostupno na: http://poincare.matf.bg.ac.rs/~vladaf/Courses/Matf%20MNSR/Prezentacije%20Individualne/Mijalkovic_NoSQL_baze_podataka.pdf (28.07.2020.). Str. 5.

Korištenje ovih baza podataka generira konkretne prednosti ili koristi, kao i sve ostale vrste. One se mogu razmatrati kroz sljedeće funkcije ili doprinose (Mijalković, 2013):

- Jednostavnost korištenja,
- Velika skalabilnost;
- Jednostavnost operacija i modeliranja podataka.

U usporedbi s relacijskim odnosno standardnim SQL bazama podataka treba istaknuti da je obrada podataka znatno jednostavnije. Kod SQL bazi podataka upiti mogu biti puno kompleksniji, s obzirom da ove baze sadrže "*join*" operacije koje se onda povezuju s drugim tablicama. Osim toga, SQL upiti mogu sadržavati druge, SQL upite, što dodatno otežava samu optimizaciju.

3.2.2. GRAF KAO MODEL PODATKA

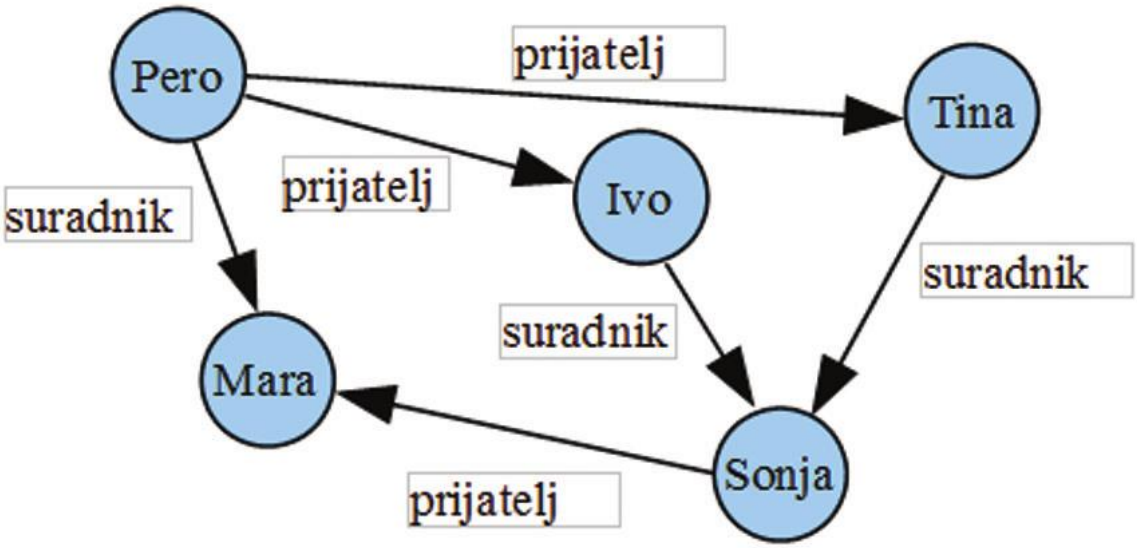
Graf baze podataka, kao što i sam naziv ukazuje, koriste se u aplikacijama kod kojih su značajni odnosi među podacima i iskazivanje tih odnosa. Ovi sustavi specificirani su eksplicitno, dok je kod relacijskih baza podataka implicitni i nefleksibilni odnos podataka.

Naime, graf baze podataka koriste se primjerice kada je važno povezati dva ili više entiteta te prikazati odnose među njima. Jedan od primjera može biti iskazivanje odnosa između korisnika, narudžbi, artikla i proizvoda. Točnije, ovom se bazom iskazuje što je neki kupac naručio.

Treba pri tome naglasiti da takve informacije daju i relacijske baze, no one se čine neprikladnim i neefikasnim u slučaju kada se želi istaknuti primjerice koji je kupac naručio proizvod 2015. Vrijeme za dobivanje te informacije korištenjem relacijske baze podataka je znatno duže nego na primjeru graf baze podataka.

Ovaj se primjer može dočarati Slikom 4.

Slika 4. Graf baza podataka



Izvor: Stojanović, A. (2016.) Osvrt na NoSQL baze podataka – četiri osnovne tehnologije. Polytechnic & Design. Vol. 4. No. 1. Str. 48.

Baze podataka bazirane na ovom modelu, kao što je i prikazano, eksplicitno čuvaju i prikazuju veze među čvorovima. Za izvještavanje o odnosima među entitetima dovoljno je pratiti putanje među njima. Moguće je predstaviti tko je prijatelj Perinog prijatelja, pri čemu se prati veza od Pere do Tine, pa od Tine do Sonje, pa od Sonje do Mare. Graf baze podataka su za ovakva pitanja efikasnije jer uočavaju direktne veze među čvorovima i štede vrijeme te napore.

3.2.3. DOKUMENT BAZE PODATAKA

Među svim NoSQL bazama podataka, dokument baze podataka su najpopularnije te imaju široku primjenu. Osnovni razlog tome očituje se u činjenici da je model intuitivan, a osnovni element je dokument, koji se određuje kao uređeni skup ključeva s pridruženim vrijednostima.

Način funkcioniranja temelji se na smještaju dokumenata u kolekcije, a pri čemu svi dokumenti imaju dinamičke sheme. To znači da svaki dokument u kolekciji može imati drugačiju strukturu, a to je osnovno razlikovno svojstvo u odnosu na relacijske baze podataka.

Kao ogledni primjer može poslužiti JavaScript notacija (Stojanović, 2016):

{ime: “Pero”, prezime: “Perić”}

Navedeni primjer iskazuje odnosne ključeve i pridružene vrijednosti. Ključevi su ime i prezime, a vrijednosti Pero i Perić. Treba istaknuti da kod ovog modela dokumenti mogu imati hijerarhijsku strukturu pri čemu se određuje pripadnost nekog dokumenta onom drugom. Kompleksniji primjer takve strukture može se dočarati kako slijedi (Stojanović, 2016):

{autor: “Pero Perić”,
knjige: [{ naziv: “Knjiga 1”, godina: 2012 },
{ naziv: “Knjiga 2”, godina: 2013 }],
ostalo: { izdavači: [“Addison- Wesley”, “Prentice Hall”, “Wiley”],
područja: [“Računarstvo”, “Matematika”, “Statistika”]}

Kako bi se u ovome sustavu razlučila značenja i odnosi koriste se kolekcije. Na primjer neka kolekcija može sadržavati knjige, druga popis studenata, treća predmete te tako redom dalje.
Daje se konkretizirati kako ovaj model podataka u praksi korisnicima omogućuje raspoređivanje podataka na više servera. Osim toga, omogućeno je i indeksiranje za bržu podatkovnu obradu, te replikaciju ili čuvanje kopija.

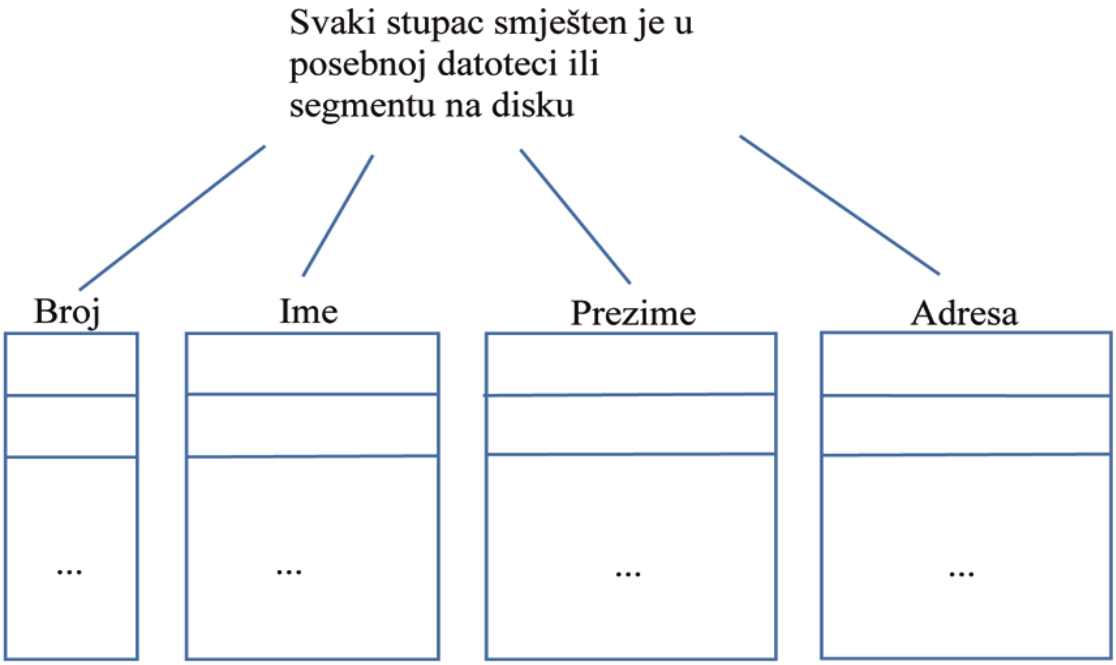
3.2.4. STUPIČASTE BAZE PODATAKA

Stupičaste baze podataka temelje se na stupcima, koji se mogu promatrati kao stupci na primjeru tablica. U usporedbi s relacijskim bazama podataka treba istaknuti kako se ovi sustavu koriste, između ostaloga, za potrebe pristupa odabranim stupcima neovisno o drugima. To je ujedno i temeljna prednost ovih sustava i baza podataka koje se na njima zasnivaju (AWS, 2020).

Sljedeća značajka osniva se na efikasno učitavanje vrijednosti u memoriju stupaca. To znači da vrijednosti su dio nekog stupca postaju smještene na jednom cjelovitom prostoru. Primjerice, ako neka tablica zauzima određenu količinu prostora, primjerice 20 GB, ona je raspoređena nejednako u okviru svakog stupca. Tako će stupac A imati 5 GB, stupac B 12 GB, a stupac C 3 GB. Nastavno tome, ako upit koristi jedan od tih stupaca, na primjer stupac B, tada se koristi samo 12 GB, a ne čitav prostor.

Podaci se u ovome sustavu za svaki stupac smještaju u zasebnu datoteku (Slika 5.).

Slika 5. Stupičasti sustav



Izvor: Stojanović, A. (2016.) Osvrt na NoSQL baze podataka – četiri osnovne tehnologije. Polytechnic & Design. Vol. 4. No. 1. Str. 51.

U skladu s danim prikazom, vrijednosti za svaki red stupca postavljene su jedna do druge. Učitavanje stupaca u ovome sustavu je particioniranje, agregacija, specijalni tipovi kolekcija te spremište za datoteke.

Particioniranje podrazumijeva raspodjelu podataka na više servera s ciljem maksimiziranja efikasnosti u radu s velikom količinom podataka. Agregacija se odnosi na kombiniranje i transformaciju dokumenata, odnosno postupke filtriranja, grupiranja, sortiranja i ostalog. Specijali tipovi kolekcija nadalje omogućuju spremanje dokumenata čije vrijeme je isteklo, dok spremišta za datoteke označavaju mogućnost spremanja ili pohranjivanja velikih datoteka (AWS, 2020).

Kao što je već i naznačeno, razlika između stupičastih sustava i relacijskih baza podataka odnosi se i na nemogućnosti korištenja operacije pridruživanja ili „*join*“. To je razlog veće skalabilnosti za distribuirane sustave. Među osnovne prednosti stupičastih sustava, odnosno ovih NoSQL baza podataka iskazuje se u njihovoj prikladnosti za obradu iznimno velikih količina podataka.

Konkretizira se da se relacijska baza podataka koristi za spremanje redaka podataka, a obično su to transakcijske aplikacije, dok je stupčana baza podataka optimizirana za brzo pretraživanje stupaca podataka, obično u analitičkim aplikacijama. Spremanje u stupcu smatra se važnim čimbenikom u izvedbi analitičkih upita jer drastično smanjuje sveukupne potrebe ulaza/izlaza diska i smanjuje količinu podataka koju je potrebno učitati s diska.

Kao i druge NoSQL baze podataka, ova baze podataka orijentirana je na stupce dizajnirane za skaliranje „korištenjem“ distribuiranih klastera jeftinog hardvera za povećanje propusnosti. To ih čini prikladnima za skladištenje podataka i obradu velike količine podataka.

4. RAD S NOSQL BAZOM PODATAKA I KOMPARATIVNA ANALIZA NOSQL SUSTAVA TE RELACIJSKIH BAZA PODATAKA

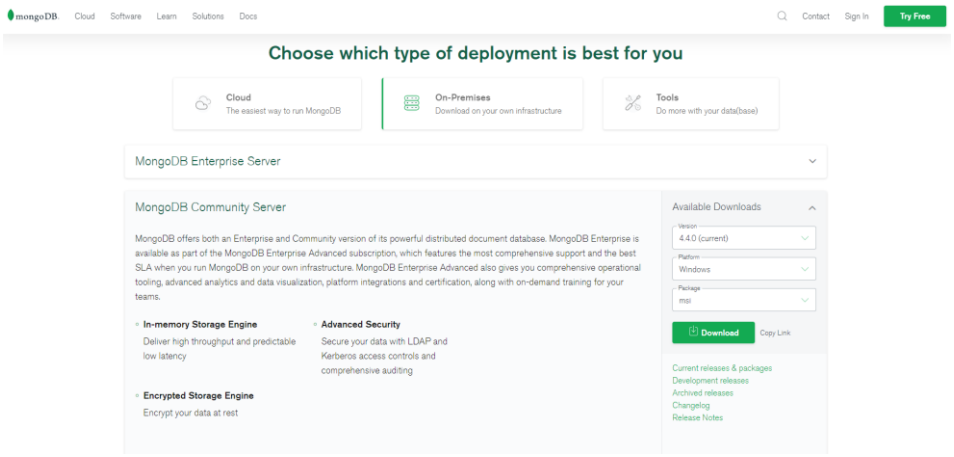
U ovome dijelu rada daje se primjer rada s jednom od NoSQL baza podataka. Osim toga, sumirajući prethodne značajke ovih baza podataka, daje se prikaz komparativne analize s relacijskim bazama podataka.

4.1. RAD S NOSQL BAZOM PODATAKA

Kao ogledni primjer za potrebe analize rada s NoSQL bazom podataka uzima se i u ovome radu predočava MongoDB. Riječ je o sustavu za pohranjivanje dokumenata otvorenog koda, te o jednom od vodećih NoSQL sustava za upravljanje bazama podataka. On djeluje na način da se aplikacije izrađuju i održavaju na način da se osigurava agilnost i skalabilnost sustava.

Prije pokretanja MongoDB-a potrebno je preuzeti i instalirati MongoDB community server besplatno sa www.mongodb.com stranice bez potrebne dodatne registracije. Na stranici se može pronaći i vodič za instalaciju koji dodatno objašnjava korake i postupak instalacije.

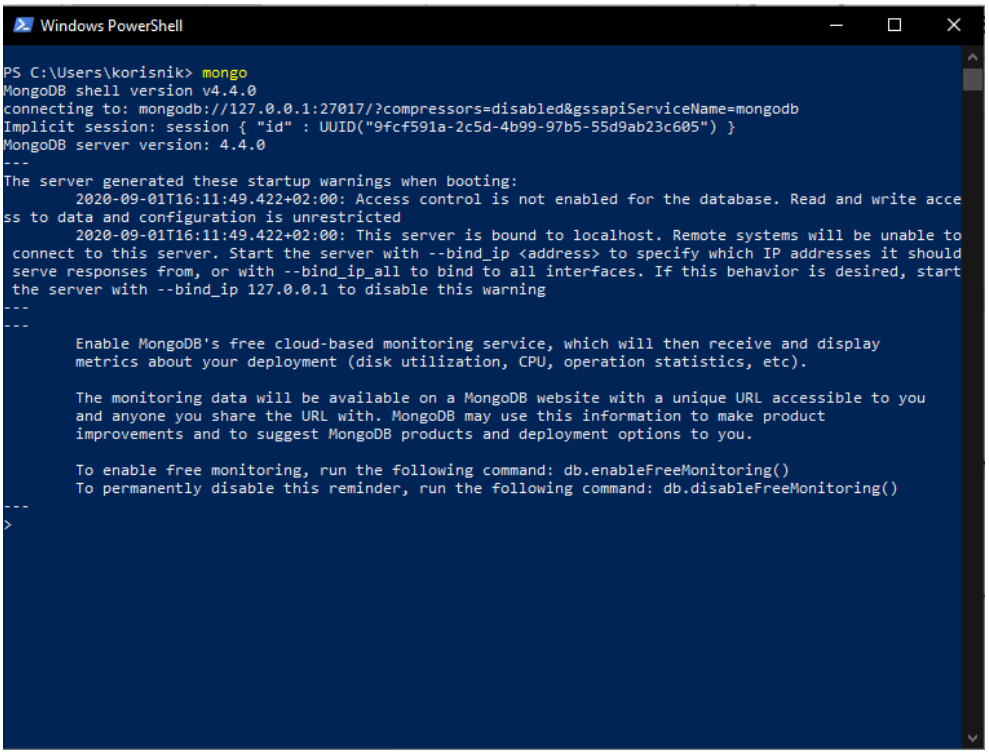
Slika 6. Prikaz stranice za odabir verzije MongoDB za instalaciju na osobnom računalu.



Izvor: Izbor verzije MongoDB za instalaciju.
https://www.mongodb.com/try/download/community

Nakon instalacije potrebno je pokrenuti MongoDB servis preko Windows Powershella upisivanjem „mongo“ u naredbeni redak.

Slika 7. Windows PowerShell prozor sa pokrenutim MongoDB servisom



Kako bi se prezentirao način rada s ovim sustavom, istražuju se sljedeći segmenti ili aktivnosti:

4.1.1 DODAVANJE PODATAKA;

4.1.2 CRUD OPERACIJE;

- NAČIN POHRANE PODATAKA (CREATE);
- ČITANJE PODATAKA (READ);
- AŽURIRANJE PODATAKA (UPDATE);
- BRISANJE PODATAKA (DELETE);

4.1.3 REFERENCIRANJE PODATAKA.

4.1.4 INDEKSIRANJE PODATAKA.

- JEDINSTVENI INDEKSI

4.1.5 AGREGIRAJUĆI I MAP-REDUCE PRISTUPI AGREGACIJA MONGODB SUSTAVA.

4.1.1. DODAVANJE PODATAKA

Naredbom „use“ se spajamo na postojeću bazu podataka, no ona nam isto služi i za kreiranje nove baze podataka ukoliko ona već ne postoji pod tim nazivom.

Slika 8. Naredba za dodavanje nove baze podataka u MongoDB

```
> use trgovina
switched to db trgovina
>
```

4.1.2. CRUD OPERACIJE

Crud operacije se odnose na osnovne operacije: umetanja, čitanja, ažuriranja i brisanja. Sada kad smo dodali novu bazu podataka u MongoDB možemo započeti raditi s istom.

- **Način pohrane podataka (Create)**

Podaci se pohranjuju u ovome sustavu u BSON formatu dokumenta. To su analogno strukturirani objekti u objektno orijentiranim programskim jezicima (Slika 9.).

Slika 9. Prikaz pohrane podataka u MongoDB

```
> db.proizvodi.insertOne({name: "Mlijeko", price: 4.99})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5f4e6e9ca4a4448d3fe453e8")
}
>
```

U ovome se sustavu svi dokumenti pohranjuju unutar takozvanog *collectiona*. To je zapravo grupa dokumenata koja se grupira na temelju zajedničkih svojstava.

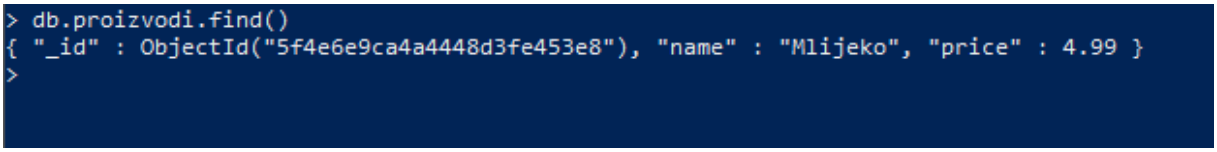
Naredba *db.collection.insertOne* (umjesto *collection* se upisuje naziv baze koji je zadan naredbom „*use*“) služi za dodavanje podataka u kolekciju uz pomoć JSON objekta (koji se nalazi u vitičastim zagradama) koji sadrži polja vrijednosti ključa, u ovome slučaju ime i cijenu proizvoda. Nakon upisa podataka MongoDB ispisuje potvrdu o upisu podatka u bazu te tim podacima automatski generira jedinstveni *ID* što se može i vidjeti na Slici 9.

Moguće je dodati i više kolekcija odjednom sa naredbom *db.collection.insertMany*.

- **Čitanje podataka (Read)**

Čitanje podataka u ovome sustavu temelji se na upitima, kojima se nastoji obuhvatiti jedan ili više podataka, a pri tome se koriste točni kriteriji obuhvata. U ovome sustavu oni mogu imati jedan *Collection* dokument, a sam upit integrira detaljizirane uvjete, koje dokument mora zadovoljiti. Za ovu aktivnost koristi se *db.collection.find()* naredba (MongoDB, 2020). Primjer čitanja podataka u ovome sustavu prikazuje se Slikom 10. Za čitanje jedne kolekcije koristi se naredba *db.collection.findOne()*, koja ispisuje prvi dokument koji se podudara sa traženim nazivom.

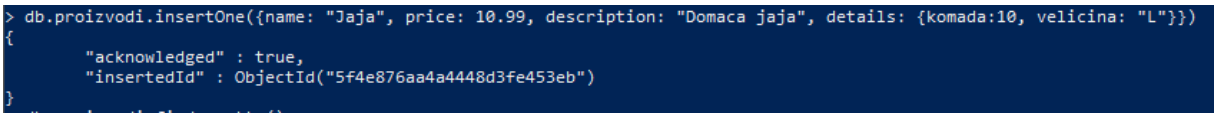
Slika 10. Čitanje podataka u MongoDB sustavu



Ako se naredbi ne dodaju argumenti u zagradi, kao u primjeru na Slici 10, MongoDB će ispisati sve podatke u izabranom *collectionu*.

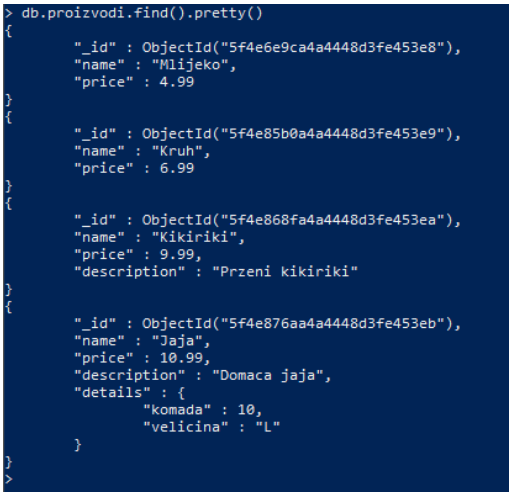
Nadalje, korištenje ovog sustava zahtijeva razradu modeliranja podataka. Oni se modeliraju na način definiranja točne strukture dokumenta za pohranu podataka. Oni dokumenti koji se pohranjuju unutar jednog *collectiona* imaju sličnu strukturu. U načelu postoje dva pristupa povezivanja dokumenta, reference i ugrađeni dokumenti (MongoDB, 2020). Reference se pri tome odnose na poveznice među dokumentima, dok su ugrađeni dokumenti primjer umetanja jednog dokumenta u neki drugi (Slika 11.).

Slika 11. Umetanje dokumenta



Umetanje dokumenta u *collection* je moguć korištenjem vitičastih zagrada i dodavanja novog ključa, u ovom slučaju „*details*“ koji se sastoji od „komada“ i „velicina“.

Slika 12. Čitanje podataka uz naredbu „pretty“

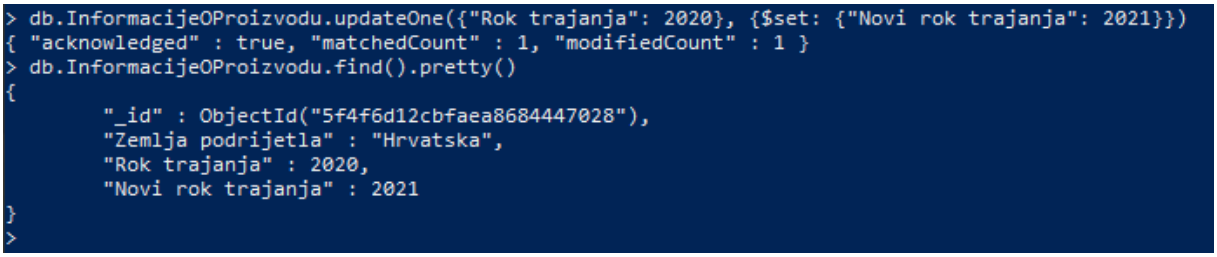


Prikaz podataka sa naredbom *db.collection.find().pretty()* koja podatke sortira za lakše čitanje. Na Slici 12. vidimo *collection* odnosno proizvod „Jaja“ sa umetnutim dokumentom „*details*“ koji sadrži količinu i veličinu proizvoda.

- **Ažuriranje podataka (Update)**

Update naredba služi za izmjenu i ažuriranje podataka iz kolekcije (engl. *collection*). Moguće je izmjeniti jedan dokument ili više njih sa naredbom *db.collection.updateOne* odnosno *db.collection.updateMany*. *Update* koristi tri argumenta pri pozivanju, prvi argument za pronalazak dokumenta (*filter*), drugi argument koji opisuje promjenu odnosno što se mijenja (*data*), a treći nam služi kao pomoć pri konfiguriranju tog procesa (*options*). Postoji i treća naredba koja služi za zamjenu nekog dokumenta novim, a piše se *db.collection.replaceOne*.

Slika 13. Ažuriranje dokumenta



Pronađen je uz pomoć filtera koji pretražuje polje Rok trajanja s vrijednošću 2020. U dokument je izmijenjen rok trajanja uz pomoć naredbe *db.collection.updateOne*.

- **Brisanje podataka (Delete)**

Isto kao i sa prethodnim naredbama, sa *Delete* možemo obrisati jedan dokument ili više njih sa naredbom *db.collection.deleteOne*, odnosno *db.collection.deleteMany*.

Slika 14. Brisanje dokumenta

```
> db.InformacijeOProizvodu.deleteOne({"Rok trajanja": 2020})
{ "acknowledged" : true, "deletedCount" : 1 }
>
```

Brisanje jednog dokumenta sa ključem „Rok trajanja“ i vrijednosti 2020.

4.1.3. REFERENCIRANJE PODATAKA

Referenciranje nam služi u izradi shema između dokumenata, iako to u Mongu nije potrebno kao kod ostalih baza podataka. Referenciranje se postiže kada polje jednog dokumenta referencira na jedinstveni identifikator drugog dokumenta.

Slika 15. Referenciranje veze jedan naprema jedan

```
> db.osoba.insertOne({ime: "Josip", godine: 23, placa: 6000})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5f4fd529cbfaea8684447033")
}
> db.auti.insertOne({model: "Mercedes", cijena: 200000, vlasnik: ObjectId("5f4fd529cbfaea8684447033")})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5f4fd58ccbfaea8684447034")
}
```

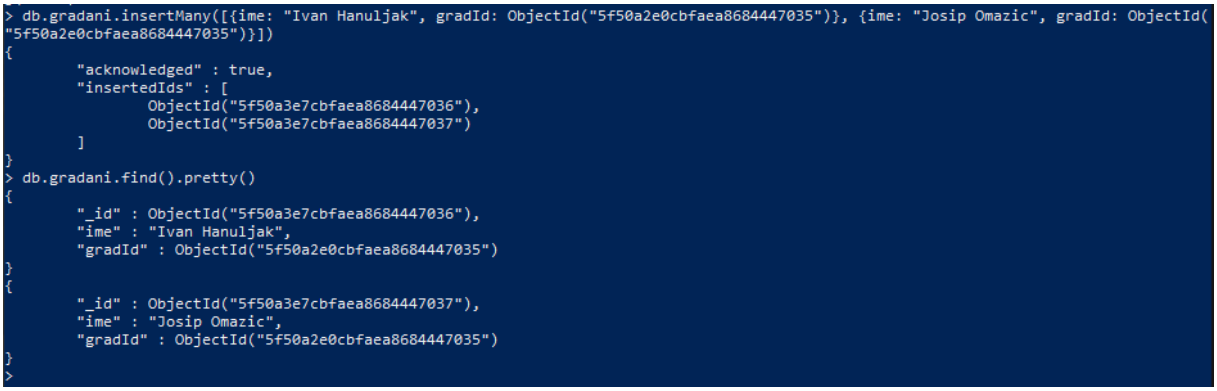
Veza jedan naprema jedan koristi se kada jednom dokumentu pripada samo jedan dokument iz druge kolekcije i s njim je povezan. Ova veza može biti zamišljena tako da jedna osoba može posjedovati samo jedan automobil, a jedan automobil je u vlasništvu samo jednog vlasnika, a to je prikazano u primjeru na slici 15.

Slika 16. Referenciranje veze jedan prema više

```
> db.gradPodaci.insertOne({ime: "Djakovo", koordinati: {a: 50, b: 34}})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5f50a2e0cbfaea8684447035")
}
```

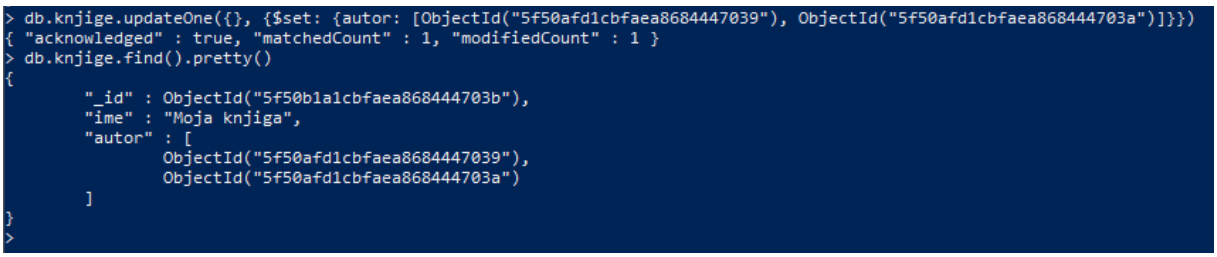
Veza jedan prema više koristi se kada jednom dokumentu pripada više dokumenata iz druge kolekcije. Ova veza se može pojasniti na primjeru grada. Jedan grad može imati više stanovnika, ali stanovnik može biti samo iz jednoga grada.

Slika 17. Referenciranje veze jedan prema više, drugi dio



Referenciranje podataka više prema više koristi se kada više dokumenata sadrži pripadajuće veze s više dokumenata iz druge kolekcije. Za ostvarivanje ovakvih veza koriste se redovi vrijednosti.

Slika 18. Referenciranje veze više prema više



Veza više prema više može biti objašnjena na primjeru pisca i knjiga. Pisac može imati više knjiga, kako bi se izbjeglo mijenjanje podataka o piscu za svaku knjigu odnosno dokument, referencom povezujemo pisca sa njegovim ključem.

4.1.4. INDEKSIRANJE PODATAKA

Za svrhu poboljšanja performansi operacija dohvaćanja, izmjenjivanja i brisanja podataka u nekoj kolekciji koristi se indeksiranje. Indeksi nam ne služe kao zamjena za kolekciju nego su korišteni kao proširenje za brže izvršavanje navedenih operacija. Taj dodatak je točnije sortirana lista vrijednosti pomoću koje Mongo tada može brže pretraživati podatke za indeksirano polje. Svako indeksirano polje sadrži pokazivač na cijeli dokument unutar neke kolekcije (engl. *collection*). Indeks ima najbolju upotrebu na poljima koja se često koriste za pretraživanje podataka, a polje koje je uvijek indeksirano je „*_id*“.

Pretraživanje podataka se može izvršavati na dva načina; pretraživanje kolekcije (engl. *Collection scan*) te pretraživanje indeksa (engl. *index scan*). Pretraživanje kolekcije traje dok se ne pretraži cijela kolekcija i dok se ne pronađu traženi dokumenti.

- **Jedinstveni indeksi**

Jedinstveni indeksi su indeksi koji se izrađuju na jednom polju u kolekciji. Ova vrste indeksa koristi vrijednosti jednog polja kako bi se izradila sortirana lista vrijednosti polja u padajućem (engl. *descending order*) ili rastućem (engl. *ascending*) obliku. Padajući oblik označava da najveća vrijednost zauzima prvo mjesto dok ostale idu prema najmanjoj koja je posljednja, dok god rastućeg oblika najmanja vrijednost zauzima prvo mjesto, a ostale vrijednosti idu redom prema najvećoj.

Slika 19. Izrada jedinstvenog indeksa na primjeru artikla

```
> db.trgovina.createIndex({artikli: 1}) {"automatskiKreiranaKolekcija": false, "brojIndexaPrij
e": 1, "brojIndexaPoslije": 2, "Dobro": 1}
```

Sintaksa za izradu jedinstvenog indeksa je *createIndex().* Indeks se izrađuje na jedno polje u primjeru na slici 19., indeksirano polje biti će artikli u kolekciji trgovina. Vrijednost koja se dodaje polju može biti -1 za sortiranje u padajućem obliku, dok je vrijednost koja se dodaje polju 1 za sortiranje u rastućem obliku. Za brisanje indeksa koristi se ista sintaksa kao na slici 19., samo se umjesto operacije *createIndex()* piše operacija *dropIndex().*

4.1.5. AGREGIRAJUĆI I MAP-REDUCE PRISTUPI AGREGACIJA MONGODB SUSTAVA

Agregirajući cjevovod je pristup koji podrazumijeva transformacije dokumenata, odnosno alternativu *map-reduce* funkciji. *Map-reduce* je obrazac za procesiranje podataka u korisne agregirane rezultate. Osim toga, ovaj sustav ima niz agregirajućih funkcija koje izvršavaju jednostavne operacije nad podacima, a one su u ovome slučaju ograničene, što nekad koristi jednostavnim operacijama koje se provode na primjeru skupa podataka.

U suvremeno doba zamijećena je popularnost dileme i istraživačkih pitanja u svezi automatizacije prebacivanja podataka iz relacijskih baza podataka u MongoDB bazu. Može se istaknuti kako za tu svrhu danas već postoje određeni alati, no također popularno je i razvijanje individualnih skripti ove transformacije. Način na koji se rade dana prebacivanja podataka odnosi se na to da su relacijski sustav i MongoDB pokrenuti paralelno, a podaci se premještaju inkrementalno.

4.2. USPOREDBA SQL I NOSQL NAREDBI

SQL je lagani deklarativni jezik. Veoma je moćan i postao je međunarodni standard, iako većina sustava implementira suptilno različite sintakse.

NoSQL baze podataka koriste upite izgleda JavaScript-a s argumentima sličnim JSON-u. Osnovne operacije su jednostavne, ali ugniježđeni JSON može postati sve zamršeniji za složenije upite.

Tablica 3. Usporedba SQL I NoSQL naredbi.

SQL	NoSQL
Kreiranje nove tablice (Create)	

`CREATE TABLE knjiznica (`
`  id int,`
`  ime.pisca varchar(25),`
`  naziv.knjige varchar(25),`
`);`
	`db.createCollection("knjiznica")`

Umetanje novih podataka (Insert)	

`INSERT INTO knjiga (`
`  id, naslov, pisac`
`)`
`VALUES (`
`  '12345678910',`
`  'Moja knjiga',`
`  'Ivan Hanuljak i Josip Omazic'`
`);`
	`db.book.insert({`
	`  ID: "12345678910",`
	`  naslov: "Moja knjiga",`
	`  pisac: "Ivan Hanuljak i Josip Omazic"`
	`});`

Ažuriranje podataka (Update)	

`UPDATE knjiga`
`SET cijena = 35.99`
`WHERE id = '12345678910'`
	`db.knjiga.update(`
	`  { id: '12345678910' },`
	`  { $set: { cijena: 35.99 } }`
	`);`

Računanje zbroja (Count)	

`SELECT COUNT(1) FROM knjiga`
`WHERE id_izdavaca = 'BrojStranica001';`
	`db.knjiga.count({`
	`  "ime.izdavaca": "BrojStranica"`
	`});`

Ispis zbroja (Aggregate)	

`SELECT format, COUNT(1) AS zbroj`
`FROM knjiga`
`GROUP BY format;`
	`db.knjiga.aggregate([`
	`  { $group:`
	`  {`
	`  _id: "$format",`
	`  total: { $sum: 1 }`
	`  }`
	`  }`
	`]);`

Brisanje broja stranica knjige (Delete)	

`DELETE FROM knjiga`
`WHERE id_izdavaca = 'BrojStranica001';`
	`db.knjiga.remove({`
	`  "ime.izdavaca": "BrojStranica"`
	`});`

Prikaz knjiga (Find)	

`SELECT LOCATE("knjiga", knjiga)`
`FROM knjiznica;`
	`db.knjiznica.find()`
	`/`
	`db.knjiznica.find().pretty()`

4.3. KOMPARATIVNA ANALIZA NOSQL SUSTAVA I RELACIJSKIH BAZA PODATAKA

U prethodnom poglavlju bilo je usporedne razrade različitosti između ovih baza podataka. One se sistematizirano mogu prikazati na sljedeći način, Tablicom 4.

Tablica 4. Entiteti i razlike između NoSQL i relacijskih baza podataka

Relacijske baze podataka	NoSQL baze podataka
• Baza podataka	• Baza podataka
• Tablica	• *Collection*
• Redak	• Dokument
• Indeks	• Indeks
• Funkcija *join*	• Referenca ili ugrađeni (insertirani) dokument

Izvor: Peroković, M. (2014.) Primjena NoSQL baza podataka na sustav za upravljanje dokumentima. Dostupno na: https://bib.irb.hr/datoteka/714321.1-maperokov-primjenaNoSQLnaDMS.pdf (29.07.2020.).

Treba istaknuti da se razlike među ovim bazama očituju u modelu podataka, agregaciji i integraciji s aplikacijama. Najveće razlike su upravo u modelu podataka. NoSQL sustav ne zahtijeva strogu shemu prije umetanja podataka, niti promjenu te sheme.

Kod NoSQL baza podataka podaci se pohranjuju na način da se pohrana ne može preslikati na korištenje objekata kod objektno-orijentiranih programskih jezika, no to je moguće kod relacijskih baza podataka. Kod NoSQL baza podataka također nema spajanja tablica kod upita, kao što je to prikazano u prethodnom dijelu teksta. Agregacija podataka bitna je zbog grupiranih i statističkih informacija.

Neki NoSQL sustavi nemaju ove sposobnosti, pa se koriste zaobilazna rješenja agregacije, a djelom je o tome već i bilo riječi. U konačnici, integracija s aplikacijama ukazuje na sljedeći skup razlika između NoSQL i tradicionalnih relacijskih baza podataka.

Korištenje NoSQL baza podataka i pohrane dokumenata ima brojne prednosti. Kao što je već i istaknuto, dokumenti koji su ugrađeni s ostalim dokumentima dohvaćaju se tek jednim upitom prema bazi, dok je kod relacijskih baza potrebno spajati tablice podataka. U NoSQL sustavu također je dokument fizički pohranjen kao jedan dokument i time on zahtjeva samo jedno čitanje, dok relacijske baze zahtijevaju više čitanja uslijed većeg broja fizičkih lokacija na disku.

Kod NoSQL sustava sučelje je implementirano u obliku metoda u specifičnom programskom jeziku, bez zasebnog jezika implementacije, koji je prisutan kod relacijskih baza podataka koje imaju SQL. Dokumenti koji su pohranjeni u NoSQL bazu podataka odgovaraju modelu i strukturi podataka u objektno orijentiranim programskim jezicima, što pojednostavljuje integraciju.

Ovime se potvrđuje kako NoSQL baze podataka, s obzirom na svoja obilježja, koja u konačnici generiraju njihovu namjenu, optimalno zadovoljavaju suvremene potrebe i načine izrade aplikacija. Konkretnije, oni bolje zadovoljavaju potrebe za upravljanje velikim podacima.

4.4. OPĆA NAMJENA NOSQL BAZA PODATAKA

Istraživanjem je potvrđeno da se NoSQL baze podataka često i uglavnom koriste za pohranu velikih podataka, čija popularnost raste u suvremeno doba. Riječ je o sasvim novoj vrsti baza podataka koja je sve popularnija na primjeru poslovanja Web kompanija. Kao temeljne prednosti njihova korištenja navode se jednostavnija skalabilnost i poboljšane performanse u odnosu na tradicionalne relacijske baze podataka.

In-memory svojstvo ovih baza ukazuje na pohranu podataka u memoriji računala kako bi im se njima brže pristupio pa je važno istaknuti da one štede vrijeme i napore. Korisna rješenja ove baze daju i u kontekstu računalstva u oblaku, koje je danas sve popularnije.

NoSQL sustavi velikih podataka obrađuju stotine tisuća transakcija u sekundi što je danas bitna odrednica moderne ekonomije i globalnog društva. Može se istaknuti da one podržavaju potrebe ubrzanog načina života i poslovanja na međunarodnoj razini. Pri analizi prikladnosti i namjene ovih sustava, stručnjaci diljem svijeta ukazuju na nekoliko temeljnih odrednica. To su (Feinleib, 2012):

- Automatizirana skalabilnost – ova rješenja trebaju biti beskrajno skalabilna na potpuno automatizirani način. To znači da se ne treba trošiti vrijeme na preokupaciju čvorovima, klasterima i operacijama skaliranja. Sve to treba se odvijati automatski;
- Problematika gubitka podataka – memorijski čvorovi često se sruše i kad se to dogodi gube se svi podaci koji su na njima pohranjeni. Ova rješenja moraju se posebice baviti tim problemom. Sukladno tome, važno je da ona imaju trajne mogućnosti pohrane, automatsko prebacivanje i sigurnosne kopije. Procesi moraju biti u potpunosti automatizirani i trebaju jamčiti kontinuitet podataka u slučaju kvara i nultu stopu gubitka podataka;
- Beskompromisne performanse – očituju se u visokoj uspješnosti obrade podataka;

- Nulto upravljanje – pored stvaranja baze podataka, svi ostali operativni zadaci (nadogradnje softvera, klasteriranje, oporavak neuspjeha) trebaju biti u potpunosti automatizirani i ne zahtijevati nikakve akcije sa strane njihovih korisnika.

Vidljivo je kako postoje brojne koristi ovih baza podataka, koji svjedoče o njihovoj sve većoj popularnosti i prednostima nad tradicionalnim relacijskim bazama podataka. Uslijed njihova značaja, danas postoji veliki broj ovakvih rješenja, a pri njihovu odabiru važno je brinuti o stvarnim potrebama korisnika, ali i obilježjima tih rješenja.

5. ZAKLJUČAK

NoSQL je sinonim za baze podataka koje ukazuju da se ne koristi samo SQL. To je ujedno i osnovna razlika ovih baza podataka i tradicionalnih relacijskih baza. NoSQL mogu se definirati i kao pristupi ili sustavi dizajniranju baza podataka koje pružaju mogućnost upravljanja velikim količinama podataka.

Kod upravljanja, odnosno obrade podataka na primjeru ovih baza postoji mogućnost korištenja raznih modela podataka, odnosno ključeva, vrijednosti dokumenata, stupaca i grafikon formata. U skladu s obilježjima ovih baza podataka, a koje su predmet istraživanja u ovome radu, moguće je zaključiti kako su to zapravo alternative tradicionalnim relacijskim bazama podataka, kod kojih se podaci smještaju u tablice, a shema podataka pažljivo je osmišljena prije nego što se baza podataka izgradi.

Osnovna namjena, kao što je i istaknuto jest obrada velikih količina podataka. Time ove baze odgovaraju zahtjevima suvremenog doba i društva, posebice potrebama Web kompanija. Popularnost ovih baza rezultirala je velikim brojem komercijalnih rješenja ili varijacija, među kojima je i MongoDB, čiji rad je detaljnije predstavljen u okviru ovoga istraživanja.

Osnovna obilježja ili karakteristike NoSQL baza podataka odnose se na fleksibilnost sheme, odnosno model podataka, agregaciju i integriranje s aplikacijama. Najveće razlike očituju se u modelu podataka, a pri tome NoSQL sustav ne zahtijeva strogu shemu prije umetanja podataka, niti promjenu te sheme.

U kontekstu pohrane podataka, kod NoSQL baza podataka ne može se preslikati pohrana na korištenje objekata kod objektno-orijentiranih programskih jezika, što je pak moguće kod relacijskih baza podataka. Kod ovih baza podataka nema spajanja tablica kod upita, što nije praksa kod relacijskih baza. To obilježje štedi vrijeme i napore.

Neki NoSQL sustavi nemaju agregacijske sposobnosti, pa se koriste zaobilazna rješenja. Integracija s aplikacijama ukazuje na sljedeći skup razlika između NoSQL i tradicionalnih relacijskih baza podataka.

Korištenje NoSQL baza podataka i pohrane dokumenata ima brojne prednosti. Kao što je već i istaknuto, dokumenti koji su ugrađeni s ostalim dokumentima dohvaćaju se tek jednim upitom prema bazi, dok je kod relacijskih baza potrebno spajati tablice podataka. U NoSQL sustavu također je dokument fizički pohranjen kao jedan dokument i time on zahtjeva samo jedno čitanje, dok relacijske baze zahtijevaju više čitanja uslijed većeg broja fizičkih lokacija na disku.

Kod NoSQL sustava sučelje je implementirano u obliku metoda u specifičnom programskom jeziku, bez zasebnog jezika implementacije, koji je prisutan kod relacijskih baza podataka koje imaju SQL. Dokumenti koji su pohranjeni u NoSQL bazu podataka odgovaraju modelu i strukturi podataka u objektno orijentiranim programskim jezicima, što pojednostavljuje integraciju.

Ovime se potvrđuje kako NoSQL baze podataka, s obzirom na svoja obilježja, koja u konačnici generiraju njihovu namjenu, optimalno zadovoljavaju suvremene potrebe i načine izrade aplikacija. Konkretnije, oni bolje zadovoljavaju potrebe za upravljanje velikim podacima.

LITERATURA

Knjige i članci:

- Sadalage, P. J, Fowler, M. (2013.) NoSQL. A Brief Guide to Emerging World of Polyglot Persistence – Distilled. U.S. Pearson Education, Inc.
- Stojanović, A. (2016.) Osvrt na NoSQL baze podataka – četiri osnovne tehnologije. Polytechnic & Design. Vol. 4. No. 1. Str. 44.-54.

Internet izvori:

- AWS (2020.) What is Columnar Database. Dostupno na: https://aws.amazon.com/nosql/columnar/ (29.07.2020.)
- Feinleib, D. (2012.) Big Data and NoSQL: Five Key Insights. Dostupno na: https://www.forbes.com/sites/davefeinleib/2012/10/08/big-data-and-nosql-five-key-insights/#90001c64245f (29.07.2020.)
- Le, J. (2019.) An Introduction to Big Data: NoSQL. Dostupno na: https://medium.com/cracking-the-data-science-interview/an-introduction-to-big-data-nosql-96b882f35e50 (30.07.2020.)
- Manger, R. (2003.) Baze podataka. Dostupno na: http://jadran.izor.hr/~dadic/EKO/baze-podataka.pdf (26.07.2020.)
- Mijalković, S. (2013.) NoSQL baze podataka. Dostupno na: http://poincare.matf.bg.ac.rs/~vladaf/Courses/Matf%20MNSR/Prezentacije%20Individualne/Mijalkovic_NoSQL_baze_podataka.pdf (28.07.2020.)
- Oracle (2020.) What is database. Dostupno na: https://www.oracle.com/database/what-is-database.html (27.07.2020.)
- Peroković, M. (2014.) Primjena NoSQL baza podatakana sustav za upravljanje dokumentima. Dostupno na: https://bib.irb.hr/datoteka/714321.1-maperokov-primjenaNoSQLnaDMS.pdf (29.07.2020.).

POPIS SLIKA

POPIS TABLICA

SAŽETAK

NoSQL baze podataka javljaju se u novije vrijeme, a prate razvoj tehnologije, potrebe društva i ekonomije. javljaju se kao alternativa tradicionalnih relacijskih baza podataka, a posebnu popularnost zauzimaju s pojavom i razvojem Interneta. Danas su značajni segment Web kompanija.

Osnovna svrha im je obrada velikih količina podataka. S obzirom na svoja obilježja poprimaju znatne prednosti nad relacijskim bazama podataka. Primarno se navode ušteda vremena i napora, a kao i veća fleksibilnost. Njihova namjena određena je obilježjima modela podataka, načina agregacije i integracije s aplikacijama.

Danas postoji veliki broj komercijalnih rješenja, koja se javljaju u svim vrstama ovih baza. Radom se istražuju obilježja ovih baza podataka, područja namjene i koristi koje one donose.

Ključne riječi: NoSQL baze podataka, relacijske baze podataka, podaci.

SUMMARY

NoSQL databases are emerging the contemporary time. They are following development of technology, the needs of society and the modern economy. NoSQL appear as an alternative to traditional relative databases. They take special popularity with the advent and demarcation of the Internet. Today, they are an important segment of Web companies.

Their main purpose is to process large amounts of data. Given to the main characteristics of those databases determinates their usage. Primarily they save time and effort, as well as greater flexibility in practice. Their designation marked model data, arrangements, and application integrations.

Today, there is a large number of commercial solutions, which occurs in all types of these databases. The paper investigates the characteristics of NoSQL databases, popularity, types and usage.

Keywords: NoSQL databases, relational databases, data