NoSQL baze podataka

Klarić, Antonio

Master's thesis / Diplomski rad

2020

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, University of Zagreb, Faculty of Humanities and Social Sciences / Sveučilište u Zagrebu, Filozofski fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:131:078263

SVEUČILIŠTE U ZAGREBU

FILOZOFSKI FAKULTET

ODSJEK ZA INFORMACIJSKE I KOMUNIKACIJSKE ZNANOSTI

SMJER INFORMATOLOGIJA

Ak. god. 2019/2020

Antonio Klarić

NoSQL baze podataka

Diplomski rad

Mentor: dr. sc. Vedran Juričić, doc.

Zagreb, kolovoz 2020

Izjava o akademskoj čestitosti

Izjavljujem i svojim potpisom potvrđujem da je ovaj rad rezultat mog vlastitog rada koji se temelji na istraživanjima te objavljenoj i citiranoj literaturi. Izjavljujem da nijedan dio rada nije napisan na nedozvoljen način, odnosno da je prepisan iz necitiranog rada, te da nijedan dio rada ne krši bilo čija autorska prava. Također izjavljujem da nijedan dio rada nije korišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

_____

(potpis)

Zahvaljujem se profesoru Juričiću i profesorici Radošević na pomoći i vodstvu kroz ovaj rad i fakultetsko školovanje, profesorima odsjeka na utjecaju na moj akademski razvoj, svojim kolegama i prijateljima na druženjima i savjetima, svojoj obitelji na kontinuiranoj financijskoj i emocionalnoj potpori kroz godine studiranja te svojoj djevojci na strpljenju, motivaciji i iskazanoj ljubavi.

Ovaj rad rezultat je njihovog utjecaja.

SADRŽAJ

1. UVOD

Duga je povijest ljudskog traženja optimalnog načina za organiziranje i skladištenje različitih vrsta i oblika podataka. Geneza se može pronaći u samom pisanju knjiga te kroz razvoj enciklopedija i rječnika, što prema Harrisonu (2015) označava prvu revoluciju baza podataka. Sam naziv *baza podataka* u modernijem značenju riječi ulazi u engleski opći vokabular nakon Drugog svjetskog rata, točnije potkraj 1960-ih godina, zahvaljujući nizozemskom znanstveniku i matematičaru Edsger Wybe Dijkstri, koji je razvio algoritam za pronalazak najkraćeg puta između čvorova u grafu. Računalne baze podataka u sljedećih dvadesetak godina rastu u upotrebi zahvaljujući razvoju elektroničkih računala i računalnih komponenata (najviše procesorske i skladišne moći) te novih oblika programske logike. Nakon pojavljivanja relacijskih baza podataka početkom 1980-ih, svaka značajnija baza podataka dijelila je istu vrstu arhitekture – relacijsku. Nakon jednostrane dominacije relacijskih baza podataka od 2000-ih godina pojavljuje se nagli rast u korištenju nestrukturiranih, nerelacijskih baza podataka.

Naime, u suvremenom svijetu potrebno je pronaći adekvatno rješenje za velike trendove unutar ICT (*eng. Information and communication technology*) sfere, s obzirom da su u današnje vrijeme sposobni poticati i utjecati na svaku osobu (Pethuru i Ganesh, 2018). Polet korištenja mobilnih uređaja i računalstva u oblaku pridodalo je ogromne količine novostvorenih podataka u svijetu – toliko da je od svih ukupnih podataka u svijetu u zadnje dvije godine stvoreno 90%, od čega je 70% stvoreno ne od strane organizacija ili tvrtki, već individualnih osoba (Sawant i Shah, 2013). Prema knjizi Pethuru i Ganesha (2018: 3) ICT je glavno i najprominentnije područje koje omogućuje poslovanje te se smatra kao glavna komponenta u osnaživanju svakodnevnih odluka, dogovora i postupaka. Zbog velikog uzleta novih tehnologija, postoje i novi trendovi. Prema navodu autora, prvi trend je prema očuvanju trilijunima digitaliziranih elemenata i pametnih predmeta. Svaki uređaj sistematično je povezan s drugim uređajem, kao i s udaljenom tvrtkom, platformom, aplikacijskim slojem i sl. Sveobuhvatan naziv za to je IoT (*eng. Internet of Things*). Također, trend je i očuvanje milijarde spojenih uređaja. Beskonačno more elektronike, strojeva, instrumenata, pametnih mobilnih uređaja, vozila, opreme, dronova, pumpi, odjeće i obuće, robota te ostalih uređaja u svim vertikalama industrije dizajnirani su i proizvedeni na način da imaju sposobnost spajanja, samim time i integraciju softvera i podatkovnih izvora u Oblak (*eng. Cloud*) okolini kako bi im se omogućilo spajanje. Posljednji trend je planiranje budućnosti kao milijunima

softverskih usluga. Aplikacije na razini tvrtke (*eng. Enterprise scale applications*) postaju dostupne putem mreže, javno vidljive, omogućene s API-jem, sastavljive, lagane i poliglotske usluge. Takav razvoj podloga je za stimulirajuću i održivu proizvodnju sljedeće generacije softverskih aplikacija i servisa. Hardverski resursi i imovina definiraju se pomoću zahtjeva softvera kako bi omogućili fleksibilnost, okretnost i mogućnost ekstenzija.

Cilj ovoga rada je definirati relevantne pojmove vezane za NoSQL baze podataka, objasniti trenutnu podjelu vrsta takvih nerelacijskih baza podataka te objasniti njihove prednosti i mane. Definirat će se koji su preduvjeti potrebni za njihovo optimalno korištenje te, na kraju, što je potrebno kako bi se stvorila uredna dugotrajnost u njihovom radu. Korištena literatura je većim dijelom smatrana recentnom (nakon 2010. godine), osim u dijelovima kada se referira na temeljne radove određenih područja.

2. DEFINICIJE OSNOVNIH POJMOVA

Baza podataka u hrvatskom jeziku je prema mrežnoj stranici Hrvatske enciklopedije definirana kao *organizirana zbirka logički povezanih, pretražljivih i međusobno ovisnih podataka (informacija), pohranjenih u nekom od računalno čitljivih medija.* Šimović (2010) navodi kako je baza podataka skup datoteka i međusobnih veza između slogova i datoteka. Navodi i kako su podaci unutar baze podataka višestruko povezani te im je osiguran direktan pristup, omogućavajući time pristup prema više korisnika. Mogu li se te definicije translatirati na moderne vrste računalnih baza podataka? Definirat će se prvenstveno što su to SQL (*eng. Stuctured Query Language*) baze podataka te pomoću navedenih pojmova definiciju proširiti na NoSQL (*eng. Not Only Structured Query Language*) baze podataka.

2.1. SQL

Relacijske baze podataka koriste programski jezik deriviran iz temelja jezika System R, danas poznat kao strukturirani upitni jezik ili SQL (Hanwahr, 2017). Prema definiciji Paniana (2005: 207), SQL je općenito sustav za rad na podacima te se zasniva na relacijskom modelu podataka tj. odnosu među podacima. Odnos može biti 1:1, 1:N ili N:M (gdje se 1, N i M odnose na brojnost). Nadalje, SQL se može definirati i kao ANSI (*American National Standards Institute*), standardizirani upitni jezik *(eng. Query language*) pomoću kojega korisnik može zahtijevati podatke iz baze podataka.

Kako bi imali ispravnu relacijsku bazu koja opisuje kako zadani niz podataka treba biti prikazan korisniku, britanski znanstvenik Edgar Codd (1970) objavio je znanstveni članak *Relacijski modeli podataka za velike dijeljene podatkovne baze* (poznat i kao „Coddovih 12 pravila“), koji je korjenito utjecao na daljnji razvoj baza podataka. Njegove ideje relacijskog pogleda (ili modela) podataka postale su definicija relacijske teorije na kojoj se zasniva SQL. Sumirajući njegov rad te nazivajući ga početkom druge revolucije baza podataka, Harrison (2015) navodi kao glavne koncepte baza podataka sljedeće pojmove:

- **N-torke**: one odgovaraju vrijednostima u redu, a atributi vrijednostima u stupcu, npr. četvorku s komponentama IME, PREZIME, DATUM i MJESTO može se iskoristiti za zapis neke osobe određenog imena i prezimena koja je na određeni datum rođena u određenom mjestu.

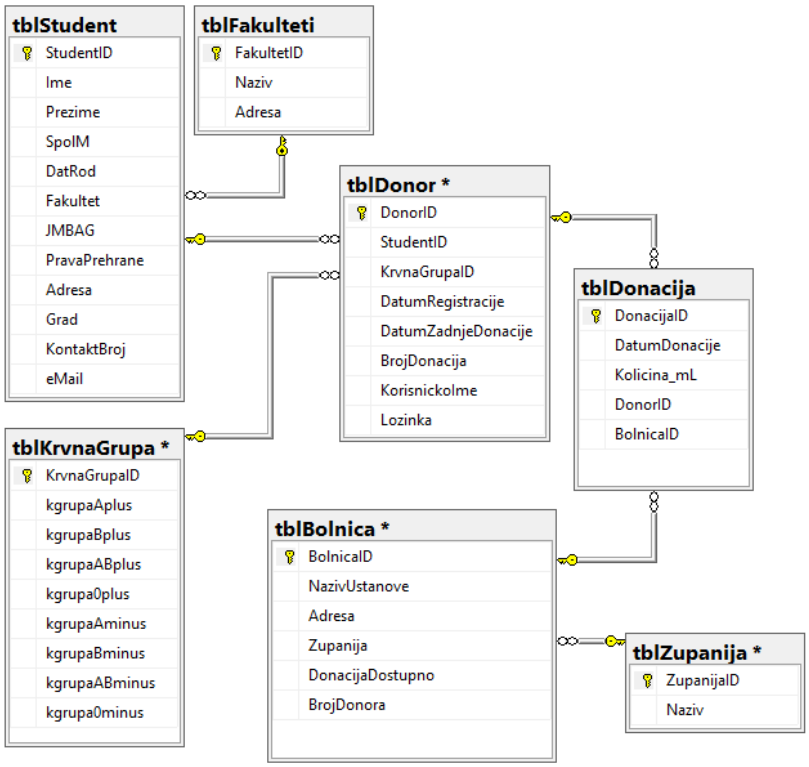
- **Relacija:** odgovara tablici u implementaciji relacijske baze, zapravo je skup određenih ntorki, npr. tablica popisOsoba.table sadrži sve vrijednosti za osobe numerirane od 1 do N.
- **Ograničenje:** ono daje konzistentnost bazi podataka. Definirana su ključna ograničenja koja određuju ntorke te relacije među ntorkama, npr. IME je ograničenja odnosno vrste podatka *string*, DATUM je vrste podatka *date*. Nadalje, red u tablici je moguće jednostavno definirati i pristupiti pomoću unikatnog ključa *(eng. Primary key)*.
- **Operacije:** vrše se na relacijama – poput pridruživanja, prikazivanja, spajanja, itd. Ove operacije uvijek vraćaju relacije, npr. postoji li potreba vidjeti sve osobe određene godine rođenja, dobiveni rezultat je prikazan u obliku tablice.

U relacijskim bazama podataka relacijski model sam po sebi ne definira način na koji baza upravlja istovremenim transakcijskim zahtjevima. Transakcija je osnovna jedinica rada nad bazom podataka te se sastoji od niza logički povezanih izmjena. Hanwahr (2017) navodi kako je definicija američkog znanstvenika, tada zaposlenika tvrtke IBM, Jim Graya u kasnim 1970-im godinama postala glavna smjernica za obrađivanje transakcija i popularno je nazvana **ACID**. Akronim predstavlja pojmove (Gray i Reuter, 1993):

- **A**tomic (Atomarnost): transakcija u bazi je nedjeljiva (po principu *sve ili ništa*, dakle ili sve tvrdnje u transakciji prolaze u bazu podataka ili niti jedna).
- **C**onsistent (Konzistentnost): baza podataka mora biti postojana odnosno dosljedna prije, za vrijeme i nakon transakcije.
- **I**solated (Izolacija): iako se mnogostruke transakcije mogu izvršavati od strane jednog ili više korisnika istodobno, jedna transakcija ne smije vidjeti rezultate drugih transakcija u postupku – rezultat more izgledati kao da su transakcije obavljane jedna za drugom.
- **D**urable (Izdržljivost): očekuje se da nakon spremanja transakcije napravljene promjene budu vidljive i nakon ozbiljnijeg pada sustava.

SQL baze podataka u potpunosti ostvaruju ACID svojstva te stoga smještaju isključivo strukturirane podatke. Vrlo su poznate po svojim transakcijskim aplikacijama. No, u situacijama kada količina podataka postane vrlo velika, interakcije unutar SQL baze podataka suočavaju se s jako velikim poteškoćama u radu (Pethuru i Ganesh, 2018).

Navedena ACID svojstva potrebna su u svakom sustavu za upravljanje bazom podataka ili menadžment sustavu (SUBP; *eng. RDBMS – relation database management system*) kako ne bi došlo do greške u radu (Deepak, 2016). Šimović (2010) ih naziva programskom podrškom. Prema tom autoru, SUBP se definira kao računalni program kojim se kontroliraju podaci i pristup podacima u bazi dok je od korisnika skrivena detaljnost fizičke pohrane podataka. Jedan od najpoznatijih sustava za upravljanje SQL bazom podataka je svakako Microsoft SQL Server (MsSQLS), za čiji je razvoj i održavanje zaslužna američka kompanija Microsoft. Primjer izgleda SQL tablice uz pomoć MsSQLS-a dan je na Slici 1.



Slika 1. Primjer SQL tablicâ i njihovog odnosa prikazan pomoću Microsoft SQL Servera (vlastiti studentski projekt)

Bez obzira na dojam zastarjelosti klasičnih SQL baza podataka u daljnjem dijelu teksta, one i dalje čine većinu korištenih baza podataka. U prilog tomu ide i činjenica kako se na njima i dalje nastavlja razvoj. Petrovečki i Kovačić (2019) u analizi razvoja nove verzije SQL Servera

2019 pojašnjavaju kako je u toj verziji proširena mogućnost pristupa podacima i drugim izvorima podataka što uključuje primjerice Oracle, MongoDB, SAP HANA ili bilo koji drugi izvor podataka koji koristi mogućnost ODBC *(engl. Open Database Connectivity)* upravljačkog programa. ODBC je standardno aplikacijsko programsko sučelje koje se koristi za pristup menadžment sustavu baze podataka. Autori smatraju kako je SQL Server 2019 stabilna platforma koja omogućuje olakšan rad s velikim podatkovnim klasterima. Prema Harrisonu (2015) veliki svjetski tvorci baza podataka primijetili su nedostatke postojećeg stanja baza podataka te su napravili svoje *„server side"* programske jezike, koristeći SQL kao podlogu. Primjeri toga su Transact-SQL i SQL server (Microsoft), PL/SQL i Oracle (Oracle), Informix (IBM), PostgreSQL i sl.

Pethuru i Ganesh (2018) također navode kako su relacijske baze podataka integralna i neophodna komponenta u IT poduzetništvu te se njihova važnost neće smanjivati. Autori navode kako sada postoje tvrtke koje nude i bazu podataka u Oblaku kao uslugu (DBaaS, od eng. *Database as a service*). Takav servis, odnosno usluga pomiče dodjelu privilegija, postavke konfiguracije, skaliranje, oblikovanje performansi, sigurnosne kopije, postavke privatnosti i kontrolu pristupa s korisnika baze podataka na operatora servisa, smanjujući općenite troškove korisnicima. Takve baze nemaju mogućnost multi-latentnosti (jedna instanca softvera radi na serveru i poslužuje više korisnika), elastične skalabilnosti i privatnosti baze.

2.2. NoSQL

Oko 2005-e godine, porastom popularnosti društvenih mreža (prvenstveno MySpacea, te u nastavku Amazona i Facebooka), postaje vrlo brzo evidentno kako klasične relacijske baze podataka, bez obzira na kontinuiranu inovativnost, neće moći podnijeti opterećenje koje im nameću spomenute masivne aplikacije na razini cijelog Interneta, ni hardverski ni softverski. Harrison (2015) naziva taj period trećom revolucijom u razvoju baza podataka. U njemu dolazimo do ubrzanog razvoja na novoj tehnologiji – nerelacijskim bazama podataka.

Što to su dakle po definiciji NoSQL baze podataka? Šimović (2010) navodi kako postoji podjela na formatizirane i ne-formatizirane baze podataka te da su ne-formatizirane baze one s ne-formatiziranim podacima. Smatra kako su to svi oni podaci koji se mogu međusobno gotovo proizvoljno povezati da bi na taj način tvorili tekstove. Raj i Deka (2018)

navode kako su NoSQL baze podataka sve one baze podataka koje su nerelacijske, distribuirane, otvorenog koda i horizontalno skalabilne (gdje je moguće dodavati nove čvorove u sustav, no o tome detaljnije u kasnijim poglavljima). Zanimljiva činjenica je da se u popularnim i znanstvenim publikacijama prilikom svakog pokušaja definicije pojma NoSQL (*Not Only Structured Query Language*) koristi negacija definicije pojma SQLa, dakle nešto poput – „*NoSQL nije (samo) relacijska baza podataka*“. Takve definicije nisu pošteđene niti ozbiljnije publikacije poput knjige Sadalage i Fowler (2013), no autori navode kako nema potrebe voditi veliku brigu o interpretaciji pojma i definirati bazu kao „ne-samo SQL“ (jer postoje iznimke toj definiciji), već se posvetiti referenci kako NoSQL označava skup baza podataka koje su otvorenog koda, razvijene početkom 21. stoljeća te u većini slučajeva ne koriste SQL. Iako je definicija da su NoSQL baze podataka nerelacijske baze direktni prijevod i doslovno znači negaciju, općenita je potreba sve definicije izražavati primjereno pojmu. Definicije također trebaju biti sažete i pregledne, te ne smiju biti cirkularne (sadržavati iste akronime) niti sadržavati pojmove koji kažu isto koliko i pojam (*Hrvatska enciklopedija – definicija*). Stoga se zapravo može smatrati da trenutno ne postoji uniformirana i potpuna definicija NoSQL-a.

2.3. Big Data i NewSQL

Spominjući NoSQL ne smije se izostaviti i termin Big Data. Prema autorima Pethuru i Ganesh (2018), to je opći pojam koji opisuje ogromne količine podataka koje nisu pohranjene u relacijskoj formi u tradicionalnim SQL bazama podataka. Programeri velikih web aplikacija imaju potrebu konstantno i vrlo brzo inkorporirati različite vrste podataka u bazu kako bi oplemenili svoje aplikacije zbog potreba tržišta, što tradicionalne relacijske baze podataka nisu u mogućnosti pratiti. Glavne karakteristike su pohrana podataka u magnitudi petabyte-a ($1\,PB = 10^{15}$ byte) i exabyte-a ($1\,EB = 10^{18}$ byte). Postoje različite strukture podataka (strukturirane, polu-strukturirane i manje-strukturirane). Nadalje, Big Data sadrži više tipova izvora podataka (senzori, strojevi, mobilni uređaji, društvene mreže, zapisnici, operativni podaci, itd.). Naposljetku, podaci su ovisni o vremenu (u realnom vremenu ili vrlo blizu realnog vremena), što znači da su prikupljeni podaci relevantni prema vremenskim zonama te je moguće izvući i vremensku komponentu kao podatak.

Michael Stonebreaker, pionir Ingres i Postgres relacijskih baza podataka 2007. godine sa svojim istraživačkim timom objavljuje rad u kojemu predlažu novi dizajn arhitekture baze podataka, što se danas naziva NewSQL (Harrison, 2015). Naziv NewSQL pripada skupini modernih baza podataka koje imaju skalabilne performanse NoSQL baza podataka namijenjenih za radno opterećenje u mrežnim transakcijskim procesima dok također sadržavaju i ACID svojstva (Pethuru i Ganesh, 2018). U cilju im je pružiti konzistentnost transakcija i ogromnu skalabilnost. Aplikacije se koriste SQL bazom podataka kako bi pokretale visoki broj paralelnih transakcija za unos i promjenu podataka. NewSQL baze programiraju se od početka stoga nemaju nikakve poteškoće u migriranju nasljednih (*eng. Legacy*) projekata s tehnologije na tehnologiju. Primjeri NewSQL baza su SAP HANA, NuoDB, MemSQL, Google Spanner, CocroachDB i Hyper (Pethuru i Ganesh, 2018).

3. TEMELJNI TEOREMI

Kako bi svaka baza podataka pohranila informacije na siguran način mora zadovoljiti nekoliko kriterija. Podacima mora pružiti:

- **Povjerljivost** (*eng. Confidentiality*) – samo autorizirani korisnici ili sustavi mogu pristupiti određenim podacima.
- **Integritet** (*eng. Integrity*) – točnost i konzistentnost podataka za vrijeme njihovog životnog vijeka.
- **Dostupnost** (*eng. Availability*) – podaci trebaju biti dostupni u bilo kojem traženom trenutku.

Većina poslovnih relacijskih SQL baza podataka poput Oracle i MsSQL baza imaju snažne sigurnosne značajke integrirane u njih. One poštuju ACID svojstva koja jamče sigurnu i pouzdanu transakciju unutar baze podataka. SUBP također sadrže razne sigurnosne značajke poput sigurnosti vezane za korisničku ulogu, kontrolu pristupa putem razina korisničkih mogućnosti, kriptiranih poruka, podrške za red i stupac pristup kontroli itd. Takve sigurnosne značajke zahtijevaju značajne troškove u obliku naknade proizvođaču menadžment sustava i ograničavaju brzinu pristupa podacima[*].

Kako je već ranije spomenuto, NoSQL baze ne prate striktno ACID svojstva. Prvotno je nastao CAP teorem, osmišljen od strane američkog znanstvenika dr. Edwina A. Brewera 2000 godine, poznat i kao Brewerov teorem. Prema Deepak (2016), Pethuru i Ganesh (2018) i Gačić (2017), akronim predstavlja termine:

- **Konzistentnost** (*eng. Consistency*) – postoji minimalno jedna kopija repliciranih podataka koja se podudara s izvornom, budući da su raspoređene kopije na različitim čvorovima.
- **Dostupnost** (*eng. Availability*) – podaci su redovno dostupni za ažuriranje bez vremenske odgode te neovisno o greškama u internoj komunikaciji između repliciranih podataka.
- **Tolerancija particioniranja** (*eng. Partition-tolerance*) – dostupnost podataka ne ovisi o mrežnim podjelama čak i kada je onemogućena komunikacija među

[*]SQL Vs NoSQL - *Exact differences and know when to use NoSQL and SQL*.

čvorovima. Tolerancija je u mogućnosti neovisnog funkcioniranja svake zasebne particije.

Navedene komponente potrebne su za uspješnu implementaciju sustava koji prenosi podatke putem dijeljene mreže. Minimalno dvije od tri komponente (pravilo 2/3) teorema moraju biti zadovoljene kako bi se ostvarila kompatibilnost i neometan rad sustava.

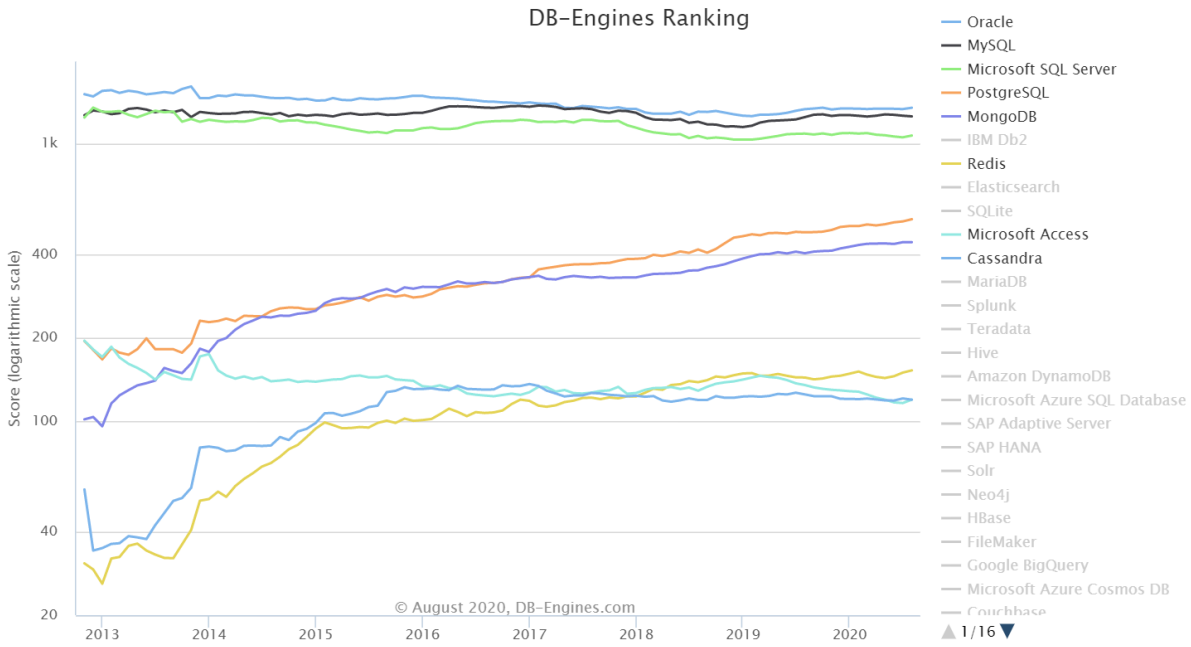
Prema CAP teoremu, nastao je skup svojstava u NoSQL bazama podataka poznat kao **BASE** (akronim za eng. ***B**asically **a**vailable, **S**oft state, **E**ventually consistent*). Deepak (2016) također smatra kako se implementacija BASE svojstava unutar NoSQL baza podataka treba evaluirati uspoređujući s ACID svojstvima, što se u praksi često navode kao oprečni teoremi. Prema BASE svojstvima, percepcija dostupnosti podataka ne ovisi o dostupnosti cijelog sustava, već o pojedinim čvorovima, čineći podatke dostupnima većinu vremena, pa i na privremenu štetu konzistentnosti podataka (ako se dogodi ispad pojedinog čvora, samo ti podaci nisu dostupni u tom trenutku). Također, stanje sustava može se promijeniti čak i kada nema upita, s obzirom na to da ažuriranje može doći s ažuriranjem čvora kojem pripada baza podataka (Sambolek, 2015 prema Cesar, 2018). Gačić (2017) objašnjava *meko stanje* kao dozvoljenost privremene netočnosti i mogućnosti mijenjanja podataka koji trenutno nisu u upotrebi. Konzistentnost podataka u BASE modelu odvija se u konačnici, s obzirom na to da prema zahtjevima sustava konzistentnost podataka nije potrebno održavati u svakom koraku transakcije. Pethuru i Ganesh (2018) BASE teorem objašnjavaju na način kako će baza podataka u nekom trenutku klasificirati i indeksirati svoj sadržaj (tj. svoje pohranjene podatke) kako bi se unaprijedilo prikazivanje podataka ili informacija sadržanih u nekom tekstu ili objektu.

Prema svakodnevnim primjerima u radu Clarence, Aravindh i Shreeharsha (2012), *„Basically available“* označava da je dio sustava online, dok se dio sustava isključuje iz mreže. Primjerice, u Amazon online trgovini moguće je u zadanom trenutku pristupiti njihovoj kolekciji informatičke opreme, dok je primjerice kolekcija glazbenih uređaja privremeno nedostupna. *„Soft state“* metoda koristi se zadnjim dostupnim podacima prije isključivanja servera kako bi napravila zaključak o stvarnom stanju sustava. Primjerice, ako je zadnji zapis u bazi kako postoji deset komada određenog artikla, no taj podataka je trenutno nedostupan, pretpostavlja se i dalje kako postoji količina od deset komada te kupac može proslijediti svoju narudžbu za odabrani artikl. Treći dio metode, naziva se „naposljetku dosljedno“ (*eng. „Eventually consistent“*) što označava da ukoliko se na stranici objavi kako

je primjerice deset komada artikla dostupno na tržištu Europe, utoliko se iste informacije može prikazati nešto kasnije na području Amerike, no sve će informacije naposljetku biti konzistentne. Sawant i Shah (2012) navode odličan primjer koristeći se primjerom Google mrežne tražilice. Prilikom pristupa tražilici i pretrage željenog termina dobivamo veoma velik broj stranica pretrage, no samo jedna stranica je zapravo vidljiva i dostupna (*basically available*). Ostatak stranica je u mekom stanju (*soft state*) i trenutno je još u procesu sastavljanja, iako mi kao korisnici ne vidimo taj pozadinski proces. U trenutku kada otvorimo prvu stranicu pretrage, ostatak podataka postaje naposljetku konzistentan.

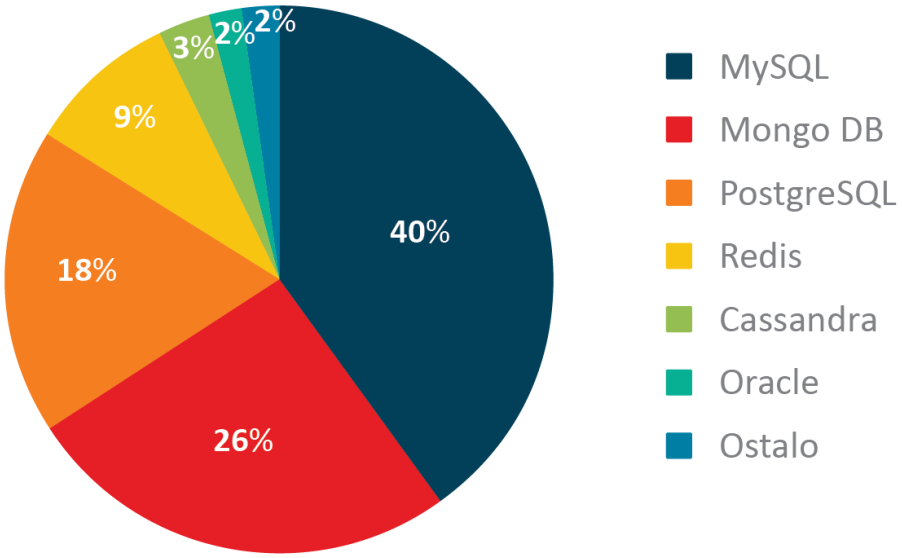
4. USPOREDBA POPULARNOSTI

Trenutno na globalnoj razini ne postoji objektivan pokazatelj utemeljen na realnim podacima korištenja svih baza podataka, no postoje određene mrežne stranice koje prate trendove kretanja prema određenim kriterijima. Porast popularnosti korištenja NoSQL baza podataka svakako je evidentan (Slika 2.), no relacijske baze podataka i dalje su u brojčanom vodstvu (Monnappa, 2019).



Slika 2. Graf popularnosti korištenja baza podataka prema stranici DB-engines
Preuzeto s https://db-engines.com/en/ranking_trend. Datum pristupa: 13.08.20.

Primjer (Slika 2.) je preuzet sa stranice koja prebrojava količinu pretraga određenih upita na web tražilicama (Google, Yahoo, Yandex) kao i opći interes na forumima (StackOverflow) ili oglasima za posao (LinkedIn). Prema primjeru kompanije ScaleGrid, koja je napravila anketno istraživanje na IT konferenciji DeveloperWeek 2019 godine održanoj u San Franciscu, može se dobiti daljnji uvid u trendove korištenja baza podataka (Slike 3. i 4.). Prema njihovim rezultatima, NoSQL baze podataka koriste se u **39.52%** slučajeva, dok se SQL baze koriste **60.48%.** Prema popularnosti njihovih tipova rezultati su opisani u Slici 3.

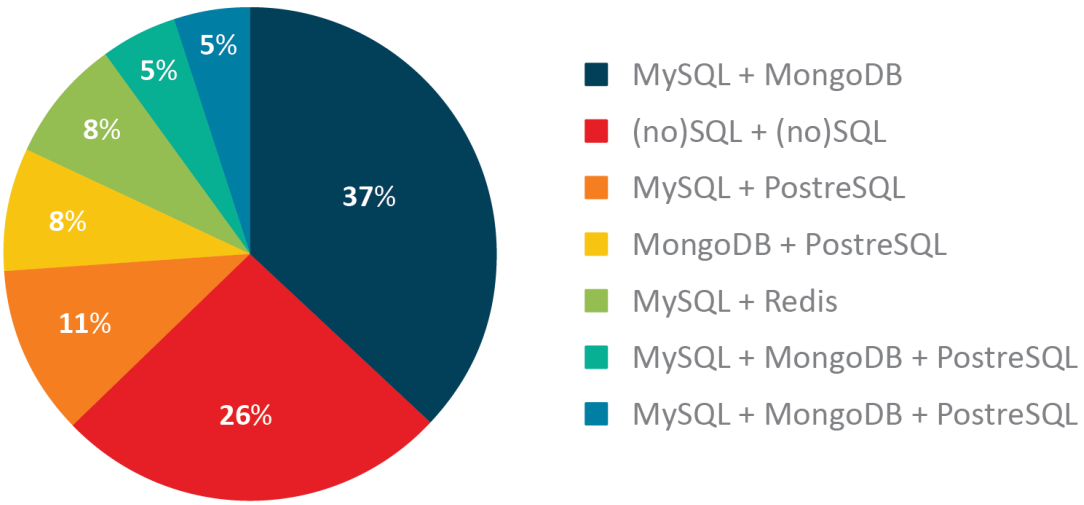


Slika 3. Prikaz popularnosti korištenja baza podataka prema istraživanju ScaleGrid-a

Nadalje, prema rezultatima istraživanja ispitanici se u čak **44.3%** slučajeva odlučuju koristiti brojčano više baza podataka, ponekad čak i različite vrste. **75.6%** ispitanika prilikom rada koristi kombinaciju relacijskih i nerelacijskih baza podataka te njihove sustave za upravljanje. Tako je postotak korištenja kombinacija baza podataka:

- SQL + NoSQL: **75.6%**
- SQL + SQL: **14.6%**
- NoSQL + NoSQL: **9.8%**

Od tih **75.6%** radi se o sljedećim kombinacijama, prikazano na Slici 4*:



Slika 4. Prikaz vrsta kombinacija NoSQL i SQL baza podataka

*2019 Database trends – *SQL vs. NoSQL, top databases, single vs. multiple database use.*

Može se zaključiti kako trendovi povećanja korištenja nerelacijskih baza podataka ne idu na štetu već postojećih baza, već se koriste kao specifično rješenje uz postojeće stanje. Razvojni programeri zbog troškova implementacije i kompleksnosti postojećih sustava često nisu u mogućnosti izvršiti tranziciju na primjerice nerelacijske oblike baza podataka. Stoga postoji zatvorena petlja koja „stare“ relacijske sustave održava relevantnima čak i kada postoji veća potreba za unaprjeđenjem sustava van relacijske strukture (Monappa, 2019). U praksi, preporučuje se korištenje NoSQL baza podataka ne kao zamjenu za SQL baze podataka, već kao nadopunu ili proširenje, ovisno o specifičnim potrebama razvijanog sustava.

5. VRSTE NOSQL BAZA PODATAKA

Nastavno na (ne)definiciju same NoSQL baze podataka, može se detaljnije definirati njihovu osnovnu podjelu. Prema velikoj većini citiranih radovima postoje četiri vrste nerelacijskih baza podataka, detaljnije objašnjene u nastavku, te poredane prema kriteriju kompleksnosti:

1. **Ključ-vrijednost baze podataka**
2. **Dokument baze podataka**
3. **Stupčane baze podataka**
4. **Graf baze podataka.**

Vrijedno je spomenuti kako Deepak (2016) navodi razvrstavanje NoSQL baze prema vrsti, no opisuje ih kao orijentaciju, primjerice Orijentiranost ključ-vrijednost. Svaka od tih tipova baza podataka dizajnirana je kako bi skladištila jako velike količine podataka kao i ostavila prostor za buduće tipove podataka. Izbor baze ovisi o vrsti podataka koje je potrebno pohraniti, njihovoj veličini, kao i kompleksnosti sustava (Pethuru i Ganesh, 2018). Nadalje, NoSQL sustavi ne zahtijevaju meta podatke. Podaci koji su pohranjeni su samo-opisni podaci koji sadrže imena i vrijednosti u zajedničkoj strukturi (Elmasri i Navathe, 2016).

5.1. Ključ-vrijednost baze podataka

Ključ-vrijednost (*eng. Key-Value DB*) baze podataka su najjednostavnije NoSQL baze podataka (Olivera, 2019). Zasnivaju se na modelu temeljenom na parovima oblika ključ-vrijednost, koji su u odnosu 1:1. Vrijednost može biti bilo kojeg tipa podataka, poput niza, cijelog broja, JSON strukture ili primjerice matrice. Ključevi su primarni deskriptori koji referenciraju na neke podatke (Pethuru i Ganesh, 2018). Svaki ključ u tablici mora biti jedinstveni no zato mogu biti u raznim formatima, primjerice u obliku URL-a (*eng. Uniform Resource Locator*), niza znakova u obliku nekog koda i sl. (Stojanović, 2016). Ovakve baza podataka po strukturi podsjećaju na rječnike te su zbog nje idealne za pohranu primjerice korisničkih profila (Olivera, 2019). Također, koriste se za aplikacije kojima je potrebna ogromna količina jednostavnih podataka poput senzornih očitavanja ili visoko promjenjivih podataka poput burzovnih kvota (Pethuru i Ganesh, 2018). Iz perspektive baze podataka „vrijednost“ je arbitraran i beznačajan pojam koji baza samo pohranjuje, ne znajući kakvi se

podaci nalaze unutar. Odgovornost je aplikacije shvatiti i iskoristiti podatke koji su pohranjeni unutar vrijednosti (Pethuru i Ganesh, 2018).

Prema Clarence, Aravindh i Shreeharsha, (2012) primjeri ključ-vrijednost baze podataka su DynamoDB, Riak, Berkeley DB, Redis, Voldemart, Tokyo Tyrant te Oracle NoSQL.

Glavne značajke prema Pethuru i Ganesh (2018) su:

- Podatkovni oblik: oblik ključ-vrijednost parova vrlo je fleksibilan oblik s obzirom na to da se ne nameće ikakva struktura nad podacima.
- Skalabilnost i pouzdanost: postiže se implementirajući particioniranje podataka, replikaciju podataka na više servera te njihov automatski oporavak.
- Ekonomičnost: skaliraju pomoću servera namijenjenih za široku upotrebu te mogu narasti na bilo koju razinu bez potrebe za ikakvim redizajnom baze. Budući da ne zahtijevaju kompleksni upitni jezik, lako je migrirati aplikaciju iz jednog sustava u drugi bez potrebe za novom arhitekturom sustava ili prepravljanjem koda.
- Brzina: moguće je istodobno izvršiti velik broj upita.

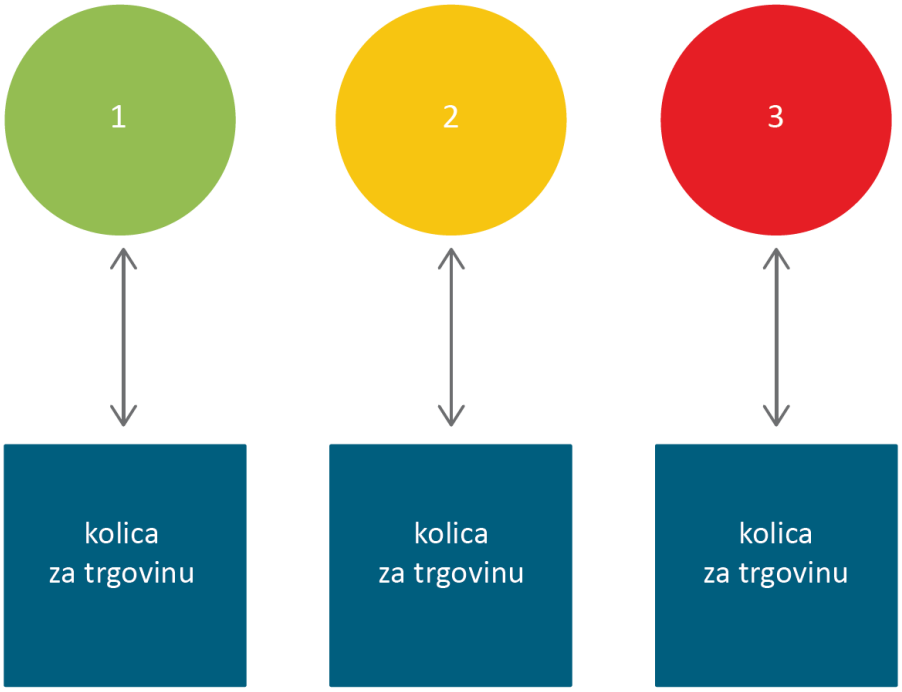
Ograničenja:

- Nema povezanosti kod mnogostrukih setova podataka.
- Multi-operacijske transakcije: ako se pohranjuje velik broj ključeva i postoji poteškoća u pohrani jednog od njih, nije moguće poništiti ostatak operacije.
- Upiti nad podacima prema „vrijednosti“ nisu mogući: traženje ključa temeljeno na nekoj informaciji pronađenoj u „vrijednost“ dijelu ključ-vrijednost para nije moguće.
- Operacije prema grupama: kako su operacije ograničene na jedan ključ u jednom trenutku, ne postoji način kako izvršavati program s više ključeva u istom trenutku.
- Budući da ne postoji stupčana struktura u bazi, ažuriranje dijela vrijednosti ili vršenje upita nad bazom za specifičnom informacijom nije jednostavno te brzo kao u usporedbi sa SQL bazama podataka.
- Povećanjem količine podataka postaje izazovno generirati tisuće novih primarnih ključeva. U tom slučaju potreban je dobro osmišljen kompleksan algoritam za generiranje niza znakova.

Primjeri učestalih operacija (Stojanović, 2016):

- get (*key*): Dohvaća vrijednost pomoću pridruženog ključa.
- put (*key*, *value*): Stvara novi par ključ-vrijednost ili pridružuje novu vrijednost već postojećem ključu.
- delete(*key*): Briše par pomoću vrijednosti ključa – ako on ne postoji, vraća grešku.
- execute(*key*): Poziva operaciju nad vrijednosti pomoću ključa.

Za razliku od relacijskih baza podataka, kako bi se pronašao specifični objekt potrebno je poznavati samo vrijednost ključa, s obzirom na to da ne postoje redovi i stupci, već samo parovi objekata (Pethuru i Ganesh, 2018).



Slika 5. Primjer korištenja ključ-vrijednost baze podataka

Na Slici 5. prikazana je vrijednost ključa (1,2 i 3) te pripadajućih kolica za trgovinu, u kojima se mogu nalaziti različiti predmeti. Ključevi nemaju interakciju s ostalim ključevima, niti pripadajući setovi podataka (kolica za trgovinu) s ostalim podacima, no u njima se mogu nalaziti u potpunosti različiti setovi podataka.

5.2. Dokument baze podataka

Poput samog naziva, naglasak je na dokumentima. Ovaj tip predstavlja najpopularniju vrstu NoSQL baze podataka. Svaki zapis i njegovi okolni podaci tretiraju se kao pojedinačni dokumenti, poznati i kao polu-strukturirani podaci (Deepak, 2016). Dokumenti koji su pohranjeni i primljeni iz skladišta baze podataka mogu biti u formatu XML, JSON ili BSON, te su dokumenti unutar baze obično slični jedno drugome. Smještaju se u kolekcije, usporedivo tablicama relacijskih baza podataka, ako jedan dokument odgovara retku tablice (Stojanović, 2016). Postoji sličnost i s ključ-vrijednost bazama podataka, no umjesto vrijednosti imamo kompleksnije podatke u obliku dokumenata (Pethuru i Ganesh, 2018). Dokumenti se dakle nalaze u strukturi podatkovnog hijerarhijskog stabla koji su samo-opisni i sastoje se od skalarnih vrijednosti, mapa i kolekcija. Svaki primarni ključ uparuje se s kompleksnijom podatkovnom strukturom (u ovom slučaju dokumentom) koji sadrži više različitih ključ-vrijednost parova ili ključ-niz parova, ili čak ugniježđene (*eng. nested*) dokumente (Slika 6.). Dokumenti se tretiraju kao jedna cjelina i dijeljenje dokumenta na njegove sastavne ime/vrijednost parove se izbjegava (Olivera, 2019). Idealna primjena bila bi u obliku kataloga proizvoda, koji može prikazati individualne proizvode ali ne i srodne proizvode. Drugi primjeri korištenja su u slučajevima kada je potrebna implementacija sustava za upravljanje sadržajem ili primjerice platforme za mrežne dnevnike (*eng. Blog*), analitiku ili e-kupovinu (Pethuru i Ganesh, 2018).

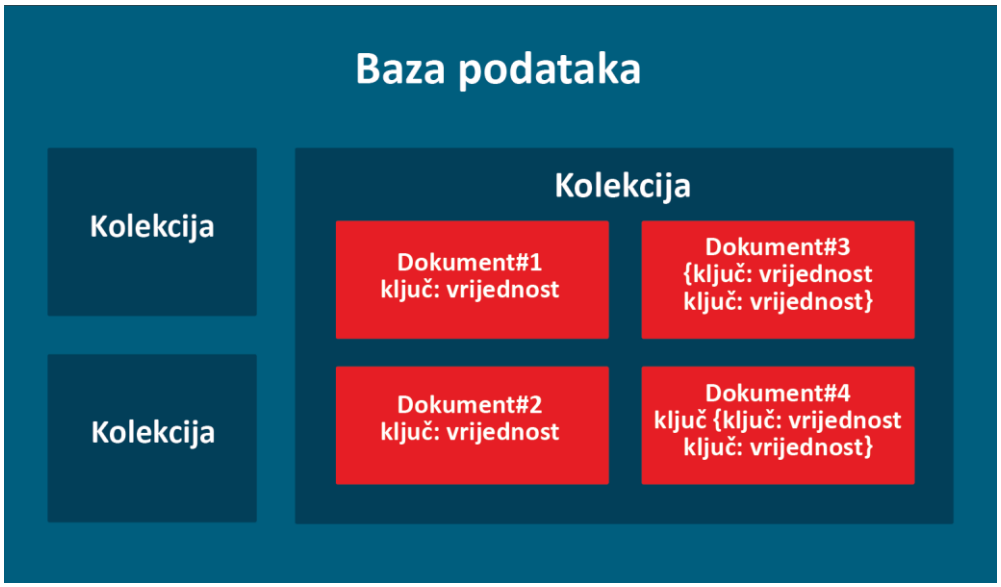
Glavne značajke:

- Podatkovni oblik: kolekcija dokumenata najčešće u formatu XML i JSON, no može se odnositi i na Microsoft Word ili PDF (*eng. Portable Document Format*) dokumente.
- Dinamičke sheme: svaki dokument u nekoj kolekciji može imati drugačiju strukturu.
- Skalabilnost: ostvaruje se horizontalnim skaliranjem.
- Replikacija: čuvanje kopija podataka na više servera.
- Brze performanse zapisivanja: dokumentne baze podataka prioritiziraju dostupnost zapisivanja naspram stroge konzistentnosti podataka. Na taj način osigurava se velika brzina zapisivanja podataka čak i u slučaju greške u hardveru ili nedostupnosti mreže.
- Implementira ACID transakcije i primjenjuje SUBP karakteristike: omogućuje indeksiranje dokumenata temeljeno na njihovom primarnom identifikatoru i svojstvima, a do neke mjere podupire i upitne transakcije.

Indeksiranjem smatramo striktno komponentu orijentiranu prema performansama baze podataka, gdje poboljšava kvalitetu odgovora na upite i ažuriranja podataka, dok usporava odgovor na dodavanje podataka i njihovo brisanje (Codd, 1970).

Općenite operacije nad dokumentima vrše se:

- Pretrage dokumenata vrše se s: *Field*, *Range* te regularnim izrazima.
- Pomoću upita koji mogu uključivati JavaScript funkcije.
- Indeksiranjem koje može biti izvršeno na bilo kojem polju.



Slika 6. Izgled sheme Dokument baze podataka

Postoji nekoliko vrsta takvih baza podataka ovisno o tekstualnom formatu kojim se koriste:

- **XML** dokument baze podataka
- **JSON** dokument baze podatka.

5.2.1. XML dokument baze

XML (*eng. Extensible Markup Language*) dokumenti formirali su prve dokumentne baze podataka. XML jezik sadrži mnoštvo standarda i alata koji pomažu s autorizacijom, validacijom, pretragom i transformacijom XML dokumenata. U svom istraživanju, Ken, Wai-Choi i Kup-Sze (2013) ustanovili su kako su XML baze podataka idealan odabir u menadžmentu kliničkih bolničkih podataka zbog brzine izvršavanja upita, kao i zbog skalabilnosti i fleksibilnosti. Najpoznatije XML baze podataka su eXist (otvorenog pristupa tj. koda), BaseX i MarkLogic (komercijalna baza podataka) (Olivera, 2019). Prema Harrisonu (2015) alati i standardi koji pripomažu obradi XML dokumenata su:

- **XPath**: sintaksa za dohvat specifičnih elemenata iz XML dokumenta, pruža jednostavan i prikladan način za filtraciju dokumenata pomoću zamjenskih znakova i oznaka.
- **XQuery**: upitni jezik za rad s upitima nad XML dokumentima, znan i kao „SQL XML-a". Povezan je s **XQuery Update**, koji pruža mogućnost promjene dokumenta.
- **XML shema**: poseban predložak dokumenta koji opisuje sve elemente koji mogu biti prisutni u specifičnim klasama XML dokumenta. Služimo se shemom kako bi validirali točnost formata XML dokumenta ili pomogli programu da prilagodi XML dokument u željeni format.
- **XSLT** *(eng. Extensible Sheet Language Transformations)*: jezik za transformaciju XML dokumenta u alternativne formate poput HTML-a.
- **DOM:** objektno orijentiran API koji programi koriste za interakciju s XML-om ili sličnim strukturama dokumenata.

XML dokument definira sadržaj podataka, no ne nosi informaciju o prikazu podataka (nema meta-podataka). XML ne sadrži stroga pravila o upotrebi elemenata i atributa. Elementi mogu sadržavati tekstualni zapis, druge elemente ili atribute, no mogu biti i bez sadržaja. Elementi za razliku od atributa mogu sadržavati mnogostruke vrijednosti, te se mogu prikazati u hijerarhijskoj strukturi (također za razliku od elemenata). XML shema detaljno opisuje strukturu XML dokumenta te se koristi za njegovu validaciju (Simović, Varga i Oreški, 2017).

XMLprimjer.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<glazbenik>
   <glazbenik_ime>Rolling Stones</glazbenik_ime>
   <albumi>
      <album>
         <album_ime>Tatoo You</album_ime>
         <datum_izdavnja>1981</datum_izdavnja>
         <zanr>Hard Rock</zanr>
      </album>
      <album>
         <album_ime>A Bigger Bang</album_ime>
         <datum_izdavnja>2005</datum_izdavnja>
         <zanr>Rock</zanr>
      </album>
      <album>
         <album_ime>Blue and Lonsome</album_ime>
         <datum_izdavnja>2016</datum_izdavnja>
         <zanr>Chicago Blues</zanr>
      </album>
   </albumi>
</glazbenik>
```

Slika 7. Primjer XML dokumenta

XML jezik (Slika 7.) pisan je u oznakama < > koje se razlikuju od drugih označnih jezika koji nemaju pred-definirana imena koja mogu koristiti i gdje razvojni programeri moraju poštovani određena pravila za ispravno definiranje XML dokumenta. Razvojni programeri imaju slobodu definirati oznake prema svojim željama. Skup podataka može se smatrati XML dokumentom ako je strukturiran prema XML specifikacijama. Svi XML dokumenti moraju imati korijenski element koji se ponaša kao nadređeni elementima u nižim razinama (primjer na Slici 5.) Elementi su osjetljivi na mala i velika slova, te se vrijednosti atributa navode u kosim zagradama. (Simović, Varga i Oreški, 2017).

5.2.2. JSON dokument baze

Kao što i sam naziv kaže, očekuje se da se podaci agregiraju i spremaju u oblik dokumenta koristeći format JSON *(eng. JavaScript Object Notation*). Harrison (2015) navodi kako su oko 2004. godine mnoge mrežne stranice počele nuditi interaktivni sadržaj zahvaljujući razvoju AJAXa (*eng. Asynchronous JavaScript and XML*) u kojemu je JavaScript unutar preglednika komunicirao direktno s pozadinskim slojem šaljući poruke u XML formatu. XML je uskoro naslijedio JSON, koji je samo-opisni format poput XML-a s više kompaktnosti i uskom integracijom u JavaScript jeziku. Elmasri i Navathe (2016) JSON opisuju kao skup podatkovnih objekata zasnovan na tekstualnom obliku kako bi se mogli

prenositi među klijentima i serverima preko Interneta. Može se smatrati alternativom XML-u te zastupa objekte koristeći atribut-vrijednost parove. Svaki JSON dokument se također može smatrati i objektom unutar aplikacije (Pethuru i Ganesh, 2018). Dokumentna baza podataka izgledom podsjeća na redove u menadžment sustavu za upravljanje relacijskom bazom podataka (RDBMS). Sadrži jedan ili više ključ-vrijednost parova, ugniježđenih dokumenata i nizova. Nizovi mogu sadržavati kompleksne hijerarhijske strukture. Kolekcija (*eng. Data bucket*) je grupa dokumenata koja dijeli određenu zajedničku svrhu (što svojim oblikom također nalikuje tablici u RDBMS). Iako poželjnije, dokumenti u istoj kolekciji ne moraju biti iste vrste (Olivera, 2019). Primjerice, JSON dokument može uzeti sve podatke pohranjene u jednom redu koji pokriva dvadeset tablica relacijske baze podataka te ih agregirati i spremiti u jedan dokument/objekt (Pethuru i Ganesh, 2018). Rezultat toga je fleksibilni podatkovni model u kojemu je jednostavno razlikovati tražene elemente. Primjeri JSON baze podataka su MongoDB, CouchDB, OrientDB, DocumentDB itd. O MonguDB-u i njegovim sustavima za upravljanje bazom podataka detaljnije je razrađeno u poglavlju 9. *(MongoDB – primjeri korištenja*). Prema Harrisonu (2015) JSON dokumenti sastoje se od malog seta jednostavnih konstrukata, (vidljivih na Slici 8.):

- Vrijednosti: mogu biti Unicode nizovi, standardno formatirani brojevi (uključujući znanstvenu notaciju), Boolean, matrice i objekti.
- Objekti: sastoje se od jednog ili više parova vrijednosti i naziva u {1:1} formatu „ime“:“vrijednost“ omeđene vitičastim zagradama { } i razmaknute zarezom.
- Matrice: sastoje se od liste vrijednosti omeđene uglatim zagradama [ ] i također razmaknute zarezom.

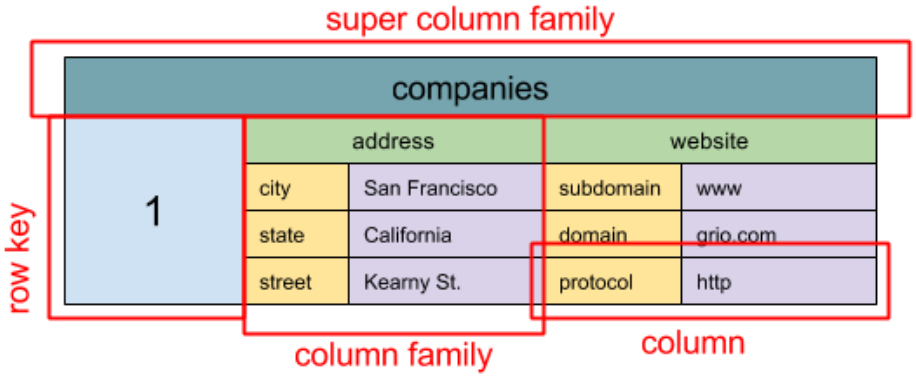
JSONprimjer2.json

```
{   "_id":{"$oid": "5f50fba26fbbcf23081cbbb3"},
    "naslov": "Inception",
    "godina": {"$numberInt": "2010"},
    "glumci":
    [   "Leonardo DiCaprio",
        "Ken Watanabe",
        "Joseph Gordon-Levitt",
        "Marion Cotillard",
        "Ellen Page",
        "Tom Hardy",
        "Cillian Murphy",
        "Tom Berenger",
        "Michael Caine"     ],
    "zanr":["Znanstvena Fantastika"]    }
```

Slika 8. Primjer JSON dokumenta

5.3. Stupčane baze podataka

Ove baze podataka pohranjuju podatke u obliku takozvanih obitelji stupaca (*eng. Column Family stores*) koji se mogu promatrati i kao stupci tablice. Posjeduju redove i brojeve stupaca koji su pridruženi s ključem reda. Stojanović (2016) navodi kako su vrijednosti jednog stupca smještene na određenom, cjelovitom prostoru diska, čineći učitavanje jednog stupca efikasnim. Po potrebi, iz jednog stupca mogu se sastaviti redovi, poput tablica relacijskih baza podataka. Računalne operacije i neke mogućnosti značajno se razlikuju ovisno o sustavu za upravljanje bazom podataka koji se koristi. Slika 9. daje primjer stupčane baze podataka u kojoj imamo osnovnu jedinicu – stupac (u paru vrijednost-ključ), grupiranu unutar obitelji stupaca, kojoj je pridružen broj reda. Sve vrijednosti obitelji stupaca tada se grupiraju u jedinstvenu, super-obitelj stupaca. Primjeri stupčanih baza podataka su Apacheov HBase i Cassandra te Kudu.



Slika 9. Primjer stupčane baze podatka.
Preuzeto s https://blog.grio.com/2015/11/sql-nosql-a-brief-history.html.
Datum preuzimanja: 28.08.20

Značajke:

- Podatkovni oblik: stupac, odnosno obitelj stupaca s kojom se može upravljati i polu-strukturiranim tipom podataka s obzirom na to da svaki red može imati vlastitu shemu.
- Kompresija: stupčane pohrane su vrlo efikasne u kompresiji podataka i/ili particioniranju, s obzirom na to da su svi podaci istog tipa.
- Agregiranje upita: zbog njihove strukture, stupčane baze podataka rade posebice dobro s agregacijskim upitima (poput SUM, COUNT, AVG i sl.), pogotovo ako se podaci nalaze unutar istog stupca.
- Skalabilnost: stupčaste baze su veoma skalabilne. Jako su pogodne za izrazito velik broj paralelnih procesa, koji uključuju rasprostranjenost podataka preko iznimno velikog broja uređaja (klastera) – ponekad i njih tisuće.

- Brzina učitavanja i izvršavanja upita: imaju mogućnost izrazito brzog učitavanja. Tablica s milijardu redova može se učitati u nekoliko sekundi.
- Konzistentnost: sadrži mješavinu ACID svojstava.

Ograničenja:

- Postupno povećanje učitavanja podataka: čitanje podataka traje manje nego njegovo zapisivanje. Ako se koriste u OLTP (*Online Transaction Processing)* bazama, stvaraju veliki napor u operacijama nad jednim redom. U stupčanoj bazi podataka dohvat podataka unutar jednog reda podrazumijeva sastavljanje reda iz svih stupaca tablice.
- Upiti samo na određeni broj redaka i određeno broj stupaca: čitanje specifičnih podataka zahtjeva više vremena od predviđenog.
- Ako postoji potreba za više polja unutar jednog stupca, zapisivanje novih podataka u takvom obliku također bi trajalo duže od predviđenog.

Zapisi u bazu podataka puno su brži od čitanja, stoga je jedno od područja korištenja i analiza podataka u realnom vremenu. Zapisivanje događaja u realnom vremenu je također jedan od slučajeva upotrebe. Primjer korištenja je Facebook-ova baza podataka Apache Cassandra – stupčana baza podataka s mogućnošću skladištenja milijardi stupaca po redu, no nije u mogućnosti pohraniti nestrukturirane podatke ili izvršiti upitne transakcije s kraja na kraj (*eng. end-to-end*) (Pethuru i Ganesh, 2018).

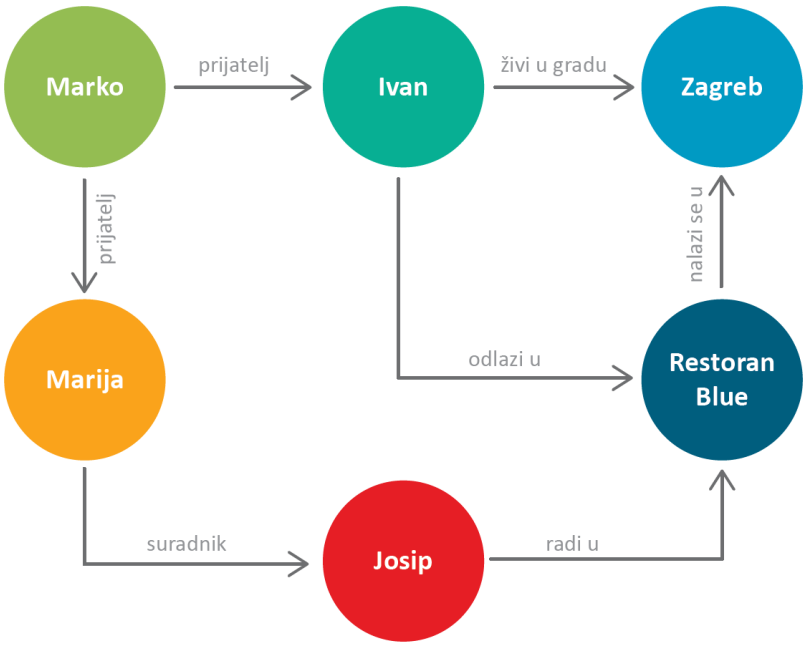
5.4. Graf baze podataka

Ovaj tip baze podataka koristan je u aplikacijama u kojima je odnos među podacima od najveće važnosti. Pomoću nje prikazuje se pohrana čvorova *(eng. Node*) ili entiteta i lukova odnosno veza među tim čvorovima. Čvorovi stoga predstavljaju podatke. Ove baze podataka su ekspresivne strukture s kolekcijom čvorova i veza koje ih povezuju, te se koriste kako bi pohranile informacije o korištenju mreže, poput društvenih konekcija, temeljeno na matematičkoj teoriji grafova (Olivera, 2019). Među-povezani podaci vrlo se lako mogu pohraniti i prikazati. Baza podataka pristupa svojim podacima pomoću upita nad prolaskom ili putom po grafu (*eng. Traversal*), navigirajući od početnog do krajnjeg čvora pomoću algoritma. Realna primjena ovih baza je u modeliranju velikog broja scenarija, poput konstrukcije svemirskih raketa, u izgradnji transportnih sustava (cestovnih i vlačnih), jednostavnijih opskrbnih lanaca i logistike, mrežnih i IT operacija, sustava za preporuku unutar tražilica u realnom vremenu, u otkrivanju zloupotrebe i prevare i slično. Društvena

mreža Facebook je jedan od slavnih primjera – u njoj je mreža među ljudima prikazana poput društvenog grafa te daje unikatnu snagu platformi (Harrison, 2015). Primjeri Graf baza podataka su Neo4J, Apache Giraph, Infinite Graph, OrientDB, FlockDB.

Glavni dijelovi grafa, vizualno prikazani na Slici 10. su:

- **Čvorovi**: predstavljaju entitet (osobu, mjesto, predmet, kategoriju ili bilo koji drugi podatak).
- **Lukovi:** predstavljaju vezu među čvorovima te se mogu indeksirati. Uvijek imaju početni čvor, krajnji čvor, smjer kretanja te vrstu (Pethuru i Ganesh, 2018). Jednosmjerni su ili dvosmjerni te postoji zadano ograničenje na količinu rubova. Mogu se shvatiti i kao ključ-vrijednost parovi koji se mogu dodijeliti čvorovima i vezama među njima (Stojanović, 2016).
- **Svojstva**: informacije o odnosima među čvorovima (najčešće kvantitativne oznake, poput težine, troška, udaljenosti, ocjene, vremenskog intervala i slično). Skup ključ-vrijednost parova sadrži podatke smještene u dotični čvor, koji se može indeksirati. Neki autori svojstva lukova ne navode kao glavne elemente, poput Pethuru i Ganesh (2018), već smatraju kako su to inherentna svojstva lukova.



Slika 10. Primjer logike unutar Graf baze podataka
Primjer čvora (vrijednosti „Marko“ i „Ivan“), svojstva (prijatelj) te luka (konekcija između ta dva entiteta).

I čvorovi i lukovi nose određena svojstva – svojstva čvorova su slična onima u relacijskoj bazi podataka/JSON dokumentu, dok su svojstva lukova tip, snaga ili povijest veze (Pethuru i

Ganesh, 2018). Prema Harrisonu (2015) graf baze podataka imaju snažne teorijske temelje u matematičkoj teoriji grafova. Matematička notacija pripisuje se za:

- Definiranje i dodavanje odnosno micanje čvorova iz grafa.
- Izvedbu operacija za pronalazak susjednih tj. obližnjih čvorova.

Postoji središnje pravilo – „nema slomljenih veza“ (*eng. no broken links*). Lukovi uvijek moraju imati početni i krajnji čvor. Brisanje jednog luka nije moguće bez brisanja njegovih pridruženih konekcija. Na visokoj razini, Graf baze mogu se kategorizirati u dvije vrste:

1. Graf baze (u realnom vremenu): Izvodi transakcijsku mrežnu održivost grafa u realnom vremenu. Slično je s mrežnim transakcijskim procesnim bazama (OLTP) u području RDBMS.
2. Grafni računski stroj (eng. *Batch mode*): Izrađuje graf analizu van mreže u skupini (*eng. Batch*) kroz niz koraka. Sličan je mrežnom analitičkom procesu (OLAP) za analizu podataka na skupini, poput rudarenja podataka (*eng. Data mining*).

Značajke:

- Skalabilnost kompleksnosti podataka.
- Fokusiranost na međupovezanost: Čak i kada količina podataka raste eksponencijalno, ove baze se ističu u upravljanju velikom količinom čvorova i kratkim vremenom izvršavanja upita nad njima.
- Postoji velik broj upitnih jezika.

Ograničenja:

- Nedostatak istodobnosti (*eng. Concurrency*) na visokim performansama: u mnogim slučajevima, ove baze podataka pružaju mnogostruko čitanje i jednostruko zapisivanje vrste transakcija, što kao posljedicu usporava istodobnost i performanse, ograničavajući paralelizam *(eng. Threaded parallelism).*
- Nedostatak formalnog jezika: nedostatak etabliranog i standardiziranog deklarativnog jezika je u ovo doba problem. Neoj4 preporučuje korištenje Cypher-a dok Oracle radi na vlastitom rješenju.
- Nedostatak paralelizma (paralelne obrade): Bitna stavka je činjenica da particioniranje grafa predstavlja poteškoću. Većina ne pruža „shared nothing“ paralelne upite na vrlo velikim grafovima, stoga je paralelizam veliki problem.

6. ARHITEKTURE SUSTAVA

Najveći izazov u korištenju NoSQL-a je skaliranje na određenu veličinu. Skaliranje, negdje zvano i particioniranje, se može raditi horizontalno (*eng. Scale out*) ili vertikalno (*eng. Scale up*). Vertikalno skaliranje podrazumijeva centralizirani pristup koji se oslanja na sve veće servere s većom kvantitetom resursa - poput memorije, procesorske moći, skladišta i I/O (*eng. Input/Output*) kapaciteta. Horizontalno skaliranje ili skaliranje prema vani označava distribuirani pristup koji balansira veći broj fizičkih i/ili virtualnih servera koji se dodaju skupini (klasteru) servera prilikom vremena izvršavanja (*eng. Runtime*) (Pethuru i Ganesh, 2018). Budući da se danas skaliranje NoSQL baza podataka izvršava isključivo horizontalno, to jest dodavanjem novih uređaja (servera) na njihovu mrežu, razvijene su određene tehnike izgradnje arhitekture sustava koje pripomažu tom postupku. Prema razradi Clarence, Aravindh i Shreeharsha (2012), osnovne tehnike su:

- Gospodar-Rob replikacija (*eng. Master-slave replication*)
- Komadanje (*eng. Sharding*)
- Model Dynamo.

Gospodar-rob replikacija

Jedan server se označava kao gospodar/ravnatelj te se svi zapisi u bazu podataka događaju primarno i isključivo na njemu. Svako ažuriranje u sustavu propagira se i na ostale servere – njegove podanike odnosno robove. Ako postoji označeni server kao „gospodar“ i tri „podanika“, zapis se događa isključivo na glavnom serveru dok svi uređaji – njih četiri, mogu odgovarati na upite čitanja podataka. Prema Pethuru i Ganesh (2018) dokument baza MongoDB primjer je korištenja ovakve arhitekture, dok je arhitektura stupčane baze podataka Cassandra bez ovakve replikacije (*eng. Masterless*), no o tome više u daljnjim poglavljima.

Komadanje (*eng. Sharding*)

Ova tehnika omogućava popunjavanje nedostataka prethodne tehnike – ako postoji više istovremenih zahtjeva za zapisivanjem u bazu podataka, to jest više nego što jedan server može podnijeti. U ovoj tehnici kompletno izolirane strukture stavljene su na jedan server prema abecednom redu. Primjerice, imena osoba su raspoređena na A-M na jednom serveru te N-Z na drugom serveru. Teorijski, mogli bismo dodati za svako slovo abecede vlastiti server te omogućiti trideset puta brži pristup upisivanju podataka. Problem u ovom modelu javlja se

prilikom dodavanja novog servera kada se cijeli sustav mora ugasiti kako bi se omogućio prijenos podataka na novi server.

Model Dynamo

Nastavak popunjavanja nedostataka kada komadanje podataka preko većeg broja servera nije optimalno, nalazi se u tehnici Dynamo. Amazonov osmišljen model ima mogućnost zadržavanja dijela sustava na mreži prilikom dodavanja novog uređaja. Postoji nekoliko pristupa tom modelu, temeljenom na BASE teoremu.

6.1. Prednosti korištenja NoSQL-a

Za web aplikacije koje upravljaju velikom količinom nestrukturiranih podataka, ne mogu se koristiti SQL baze podataka, poput primjerice stranica društvenih mreža. Te aplikacije imaju dva glavna zahtjeva – skalabilnost i dostupnost. Taj zahtjev ispunjava NoSQL baza podataka, no bez robusne sigurnosti koju pruža relacijska baza. Prema autorima Pethuru i Ganesh (2018) te Olivera (2019) prednosti NoSQL baza podataka su:

- **Elastična skalabilnost**: NoSQL baze podataka namijenjene su horizontalnom skaliranju koristeći široko dostupnu opremu (cjenovno pristupačni hardver).
- **Ekonomičnost**: NoSQL baze se mogu jednostavno instalirati na cjenovno pristupačni hardver (*eng. Commodity harware*) u skupinama/klasterima ako je potrebno izvršiti veći broj transakcija ili za veću količinu podataka. To znači da se za razliku od SQL baza podataka može procesuirati više transakcija i skladištiti više podataka po mnogostruko manjoj cijeni.
- **Primjenjivost za Big Data:** ogromne količine nestrukturiranih ili polu-strukturiranih podataka relativno lako se koriste unutar specifično odabrane NoSQL baze podataka.
- **Dinamične lako oblikovljive sheme**: NoSQL baze podataka ne zahtijevaju shemu kako bi počele raditi na podacima. U relacijskim bazama podataka potrebno je prvotno definirati shemu, što čini sustav bazično kompleksnijim, s obzirom na to da je potrebno mijenjati shemu svaki put kada se zahtjevi baze promjene. To ukratko znači da svaka kontrola kvalitete podataka mora biti odrađena unutar aplikacije. Vrijedno je spomenuti kako nisu sve NoSQL baze podataka bez sheme te također mogu imati nepogodnosti ako se organizacija podataka ne izvrši uredno. NoSQL baze ne moraju se izraditi s ciljem na

umu, već mogu imati fokus na dostupnost podataka, poput DynamoDB i Cassandra baza podataka. Kao primjer mogu se navesti platforme za oglašavanje.

- **Auto Sharding**: NoSQL baze podataka automatski i prirodno (*eng. Native*) raspoređuju podatke preko arbitrarnog broja servera unutar skupine (klastera) servera bez potrebe za pomoći aplikacija te bez potrebe da aplikacija uopće bude svjesna o kakvom se sastavu servera radi. Zbog toga, serveri se mogu dodavati i uklanjati bez potrebe za privremenom obustavom rada aplikacije.
- **Repliciranje**: većina NoSQL baza podataka podržava automatsku replikaciju baze kako bi održali dostupnost u slučaju dugotrajnijeg nedostatka električne energije ili planiranog perioda održavanja. Sofisticiranije NoSQL baze podataka su u potpunosti samoodržive, pružajući automatski povratak (*eng. Failover*) i obnavljanje, kao i mogućnost distribucije baze podataka preko nekoliko geografskih regija kako bi se oduprli ispadima unutar određene regije te kako bi se omogućila lokalizacija podataka. Također, pohranjuju i više kopija istih podataka unutar skupine (klastera) kako bi se osigurala visoka dostupnost podataka.
- **Integrirano među-spremanje (*eng. Caching*)**: Mnoge NoSQL tehnologije imaju odlične mogućnosti integriranog i transparentnog spremanja podataka u među-memoriji, održavajući često korištene podatke u memoriji sustava što je duže moguće kako bi se izbjegla potreba za posebnim slojem među-spremnika (nečim što je u relacijskim bazama podataka razvijano na posebnom serveru).
- **Otvorenog koda:** autori Dileep et al. (2016) navode kako su velika većina sustava za upravljanje bazom otvorenog koda, čineći ih dostupnim svima te vrlo lako oblikovljivim za specifične potrebe svakog razvojnog inženjera. Na njihovom razvoju sudjeluje cijela zajednica razvojnih programera neovisno o njihovoj pozadini ili geografskoj lokaciji, često se sastajući i na raznim vrstama konferencija te MeetUp-ova.

6.2. Nedostaci korištenja NoSQL-a

NoSQL baze podataka nemaju pouzdanost funkcija kakvu imaju relacijske baze podataka (odnosno, ne podržavaju ACID). To također znači kako NoSQL baze omogućuju konzistentnost u performansama i skalabilnosti. Kako bi podržali ACID svojstva, razvojni programeri moraju implementirati vlastiti kod, čineći postojeći sustav kompleksnijim. Na taj način mogu smanjiti broj sigurnih aplikacija koje čine transakcijske upite. Nadalje, NoSQL nije kompatibilan sa SQL-om. Neki NoSQL menadžment sustavi koriste strukturirani upitni jezik, što znači da će biti potrebno koristiti svoj manualni upitni jezik, čineći sustav sporijim i kompleksnijim. NoSQL baze podataka su relativno mlade u odnosu na relacijske baze podataka, što znači da su manje stabilnije i imaju manje funkcionalnosti. Prema Pethuru i Ganesh (2018) sustavi temeljeni na NoSQL bazama podataka prate CAP teorem, s ciljem izgradnje arhitekture koja za glavni cilj ima dostupnost. Moguće je poštovati samo dvije od tri smjernice teorema, čineći NoSQL baze lošim odabirom za izgradnju arhitekture aplikacije koja zahtjeva visoku razinu konzistentnosti, poput financijskih transakcija ili poziva u telekomunikacijskim kompanijama.

6.3. Sigurnost podataka

U knjizi Pethuru i Ganesh (2018) navode kako se u NoSQL bazama podataka oko sigurnosti podataka vodi velika zabrinutost. Većina baza podataka ne pruža ugrađene sigurnosne postavke u samoj bazi, već je na odgovornosti razvojnih programera da ih implementiraju u podatkovnom međusloju (*eng. middleware*). U dizajnu baze sigurnost podataka nije osmišljena kao prioritet, već brzina pristupa podacima. Nedostaju im sigurnosne značajke poput autentifikacije (određivanje identiteta nekog subjekta), autorizacije (odobrenja subjektu na pristup određenom dijelu podataka) te integritet. Budući da nemaju fiksnu i dobro definiranu shemu poput relacijskih baza podataka, nije moguće segregirati privilegije korisnicima za upit na tablici, redu ili stupci. Na taj način može doći do grešaka u konzistentnosti baze, s obzirom na to da je lako višestruko replicirati iste podatke. NoSQL baze podataka kako autori navode, naslijedile su sve poteškoće relacijskih baza podataka zajedno sa svojima, zahvaljujući svim naprednim značajkama koje donose. Heterogenost baze podataka uvelike otežava očuvanje integriteta podataka od potencijalnog napada. Nadalje, sami pohranjeni podaci nisu pod enkripcijom, već su spremljeni u običnom tekstualnom obliku. Neki od češćih vrsta napada na nerelacijsku bazu podataka su injekcijski napadi

pomoću JSON dokumenata, matrice, sheme, RESTa (*eng. Representational state transfer,* arhitekturni stil koji je najzastupljeniji u izradi internetskih aplikacija) koji se događaju zbog manjkavo spojenih protokola i mehanizama. Primjer injekcijskog napada je upit pomoću operatora *$where* unutar MongoDB-a koji može primiti kompleksne JavaScript funkcije kako bi filtrirao podatke. Napadač pod tu funkciju može umetnuti proizvoljan kôd kod operatora kako bi srušio bazu podataka. Također, kako bi se uspješno izvršio napad, osoba mora biti upoznata sa sintaksom, podatkovnim modelom i programskim jezikom u kojemu je pisana ciljana baza podataka. Prema istraživanju Shahriar i Haddad (2017) u NoSQL bazama prepoznati nedostatci teže se uniformirano sprječavaju s obzirom na to da ne postoji industrijski standard za autentifikaciju i autorizaciju korisnika te enkripciju podataka.

7. OBRAZOVANJE

Baza podataka smatra se osnovom informacijskog i ekspertnog sustava. Projektiranje uređaja za registriranje baze podataka i određivanje konfiguracije i performansi računalnog hardvera tijesno je povezano s namjenom, pa tako i sadržajem baze podataka (Šimović, 2010). Nadalje, korištenje baze podataka podrazumijeva specijalno školovano osoblje, po nazivu zanimanja *Administrator baze podataka*. Zaduženje zaposlenika je za bazu podataka u cjelini što podrazumijeva odgovornost za resurse baze podataka, njeno održavanje u radnom stanju i upravljanje. Konkretna zaduženja su primjerice – konceptualni i programski razvoj baze podataka i svrha njene primjene. Administrator također određuje prava korisnicima, zadužen je za osiguravanje pouzdanosti i integriteta podataka te njihovu zaštitu i tajnost.

Iako NoSQL puni petnaestu godinu postojanja, znanstvene publikacije tek u nekoliko zadnjih godina počinju pratiti rast i razvoj NoSQL baza podataka, posebice izvorni radovi na hrvatskom jeziku (Prilog 1). Zainteresiranost struke, ali i šire javnosti u domeni tehnologije uz potrebu za kompleksnijim rješenjima u svakodnevnom životu daje dodatan polet razvoju diskusije na temu ne-klasičnog strukturiranja podataka. Klasično obrazovan kadar u tom smislu ne postoji u cijelosti, postavljajući na tržište rada visoku raznolikost u kvaliteti. Pedagoške kompetencije ne moraju nužno biti zadovoljene kako bi se znanje o nekom području prenosilo, ne ostavljajući isključenim iz toga ni znanje o nerelacijskim bazama podataka. Nadalje, događaji poput lokalno organiziranih MeetUp-ova* (sastanci ljudi slične struke ili interesa na tjednoj ili mjesečnoj razini), kompanijskih ShowOff-ova (mini konferencija na razini kompanije ili odjela) te konferencija raznih veličina na lokalnoj, regionalno ili globalnoj razini pridonose diseminaciji specifičnog znanja van formalne strukture, prikupljajući visoku koncentraciju stručnjaka različitih područja na istom mjestu. Porastom popularnosti društvenih mreža i njihovim sveopćim društvenim prihvaćanjem, događa se također određena tranzicija u modelu klasičnog obrazovanja. Društvene mreže primjerice pružaju studentima više mogućnosti za međusobnu komunikaciju, učenje, diskusije, razmjenu informacija, reflektiranje naučenog te im omogućavaju preuzimanje materijala za učenje. Također, daju im platformu za pružanje povratne informacije i upoznavanje s novim informacijama putem Interneta (Volungevičienė, Teresevičienė, i

*MeetUp-ovi omogućuju članovima određene zajednice mogućnost česte interakcije bez potrebe za velikim planiranjem. Takvi društveni događaji su s obzirom na utrošeno vrijeme i trud u organizaciju u izrazitom kontrastu s konferencijama namijenjenim razvojnim programerima (Dileep et al., 2016).

Mejerytė-Narkevičienė, 2015 prema Đurica et al., 2018). Društvena mreža Facebook u tom kontekstu svakako je jednom od najvećih predmeta proučavanja. Manasijević et al. (2016) prema Đurica et al. (2018) navodi kako sveučilišta kao obrazovne institucije prepoznaju nezainteresiranost studenata za dosadašnji oblik prijenosa znanja. Dio odgovornosti pripada i nastavničkom iskustvu učitelja, koji primjenjuju takve tradicionalne nastavne metode. Smatra se kako je odgovornost upravo nastavnika na pronalasku efikasnijeg načina obrazovanja – jednog koji će poticati učenika na predavanju na aktivnije i motiviranije sudjelovanje. Facebook pruža društveno okruženje omogućujući studentima da surađuju s predavačima i kolegama i olakšavaju proces stjecanja znanja (Çoklar, 2012 prema Đurica et al., 2018). U istraživanju Đurica et al. (2018) o razlozima upotrebe Facebook društvene mreže među studentima, rezultati pokazuju da većina studenata svoje korisničke Facebook profile koristi za međusobnu komunikaciju i razvoj društvenih odnosa. Kao glavne prednosti uporabe Facebooka u obrazovne svrhe navode lakšu i bržu komunikaciju oko ispitnih zadataka i grupnih projekata te razmjenu materijala s predavanja i/ili relevantne multimedije.

Zbog velike brzine promjena u informacijskim tehnologijama te nemogućnosti hrvatskog obrazovnog sustava da bude u korak s tehnologijom, događa se pomak sa strukture klasičnog visokoškolskog obrazovanja na vremenski kraće, ali intenzivnije oblike obrazovanja, poput visoko-specifičnih tečajeva ili posebnih oblika postupaka certificiranja (čiji se certifikati primjerice priznaju globalno, ponekad na određeni vremenski period) koji ne moraju nužno za uvjet imati visokoškolsko obrazovanje niti su u sustavu visokog školstva. Na taj način bilo koja osoba sposobna zadovoljiti tražene uvjete uz traženu financijsku naknadu može imati pristup znanju o primjerice – nerelacijskim bazama podataka.

U Hrvatskoj, u višeškolskom i visokoškolskom obrazovanju, na razini preddiplomskog i diplomskog studija postoji nekoliko primjera podučavanja NoSQL baza podataka. Kroz proučavanje nastavnih planova i programa ustanova te pretragu njihovog silabusa putem ključnih riječi *„NoSQL“, „nerelacijsk*“* te provjerom mrežnih stranica za ishodima učenja kolegija, vidljiv je skroman broj primjera formalne edukacije po pitanju nerelacijskih baza podataka (Prilog 2). Rezultati pokazuju kako se na Fakultetu Elektrotehnike i Računalstva (FER) Sveučilišta u Zagrebu održava kolegij „Napredni modeli i baze podataka“ na razini diplomskog studija na smjeru Računalstvo. Prirodoslovno-matematički fakulteti u Zagrebu i Splitu te njihov ekvivalent u Osijeku na svojim smjerovima Informatike također nude znanje o NoSQL-u u obliku jednog kolegija. Ostale sastavnice zagrebačkog i splitskog sveučilišta

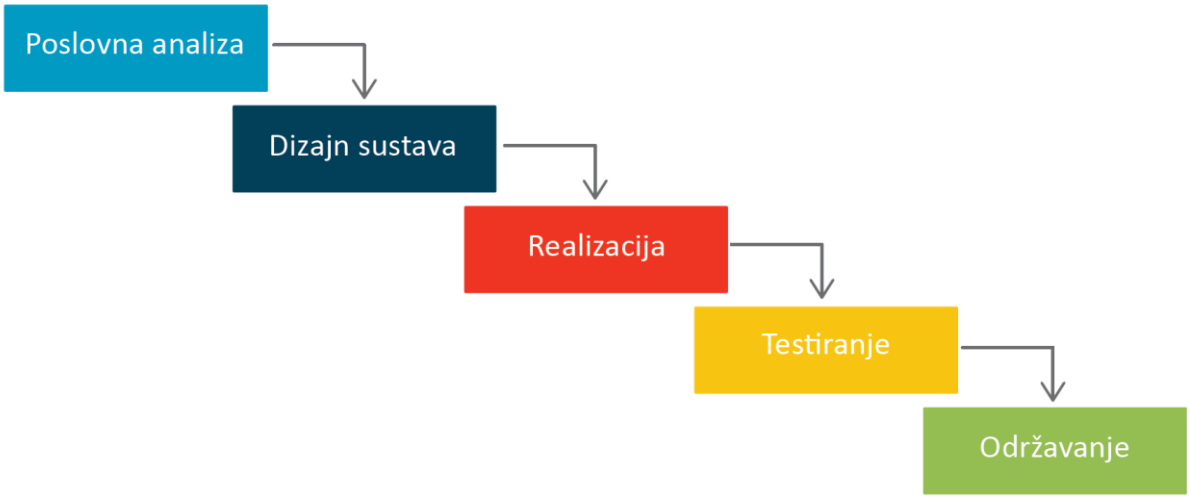
nemaju vidljiv kurikulum sa spomenutim ključnim riječima. Tehničko veleučilište u Zagrebu, doduše, na specijalističkom studiju Informatike u ponudi ima i kolegij „NoSQL i napredna Big Data analitika“ što ga čini svakako najspecifičnijim kolegijem vezanim za znanje o NoSQL-u. Sveučilište u Rijeci odijela za informatiku unutar kolegija „Sustavi za podršku odlučivanju“ također dodiruje područje nerelacijskih baza podataka. Veleučilište u istom gradu nositelj je kolegija sličnog naziva. Također, u Osijeku na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija studentima se nude dva različita kolegija na sveučilišnoj i specijalističkoj razini. Nadalje, na Fakultetu organizacije i informatike u Varaždinu u popisu kolegija ne postoji kolegij s primjerenim ishodima znanja. Potrebno je spomenuti kako SRCE (Sveučilišni Računski Centar u Zagrebu) redovito u svojoj ponudi ima radionice i objavljivane članke na temu NoSQL-a.

Može se sumirati kako u Hrvatskoj postoji deset fakulteta koji u nastavnom planu za akademsku godinu 2020/21. imaju u ponudi tek jedan[*] kolegij vezan za NoSQL baze podataka. Ti kolegiji su većinom namijenjeni studentima sveučilišnog diplomskog studija na prvoj godini te su izborni. Nose opterećenje od M=5.3 ECTS-a, što ukazuje na pretpostavljeno veliko radno opterećenje studenta. No, bez obzira na slabiju obrazovnu strukturu podučavanja, vidljiv je sve veći broj radova – što završnih, diplomskih (uključujući i ovaj), preglednih, što istraživačkih na temu NoSQL-a, označavajući disbalans između potrebe za formalnim obrazovanjem i količine objavljenih radova.

[*]Fakultet elektrotehnike, računalstva i informacijskih tehnologija (FERIT) u Osijeku jedina je iznimka s dva kolegija.

8. METODOLOGIJE RADA

Magnituda implementacije utječe ne samo na tehnologiju, već i na pristup radu. Rad s tradicionalnim, relacijskim bazama podataka najčešće zahtijeva rad po metodologiji Waterfall (*hrv. vodopad*). Prema Bassil (2012) to je jedna od temeljnih metodologija rada u životnom ciklusu razvoja softvera *(eng. SDLC – software development life cycle*). Može se definirati kao sekvencijalni model razvoja računalnog programa koji se sastoji od faza ili koraka koje dizajneri sustava i razvojni programeri moraju pratiti i završiti jednu po jednu kako bi došli do željenog rezultata i isporučili završeni proizvod (Bassil, 2012). Waterfall se sastoji od ukupno pet (5) faza (Slika 11.).

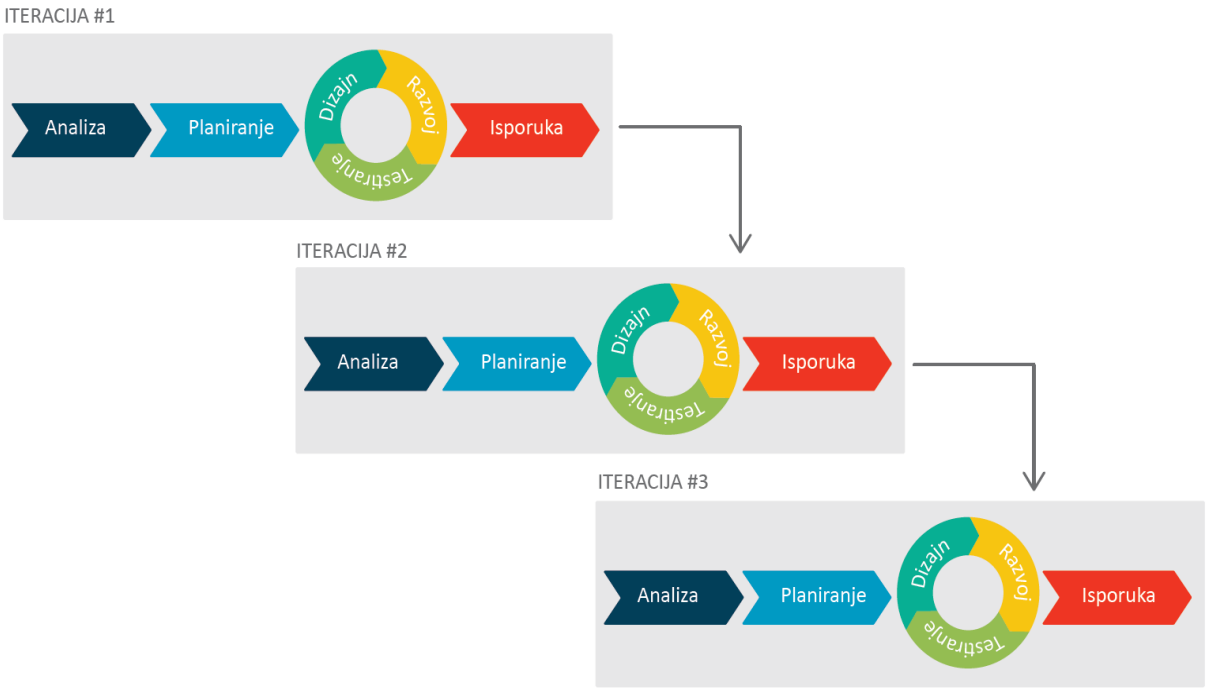


Slika 11. Prikaz tijeka razvoja softvera primjenom Waterfall metodologije

Cilj prve faze je doći do potpunog i jasnog opisa ponašanja sustava potrebnog za razvoj. Zahtjeva analizu sustava i poslovnu analizu kako bi se ustanovili funkcionalni i nefunkcionalni zahtjevi. Funkcionalni zahtjevi obično se definiraju korisničkim slučajevima upotrebe (*eng. Use case*) koji opisuju korisničku interakciju sa sustavom. Potrebno je definirati svrhu, cilj, funkcije, programske atribute, korisničke karakteristike, funkcionalne specifikacije, interakcijske zahtjeve i zahtjeve na bazi podataka. Nefunkcionalni zahtjevi za razliku od funkcionalnih označavaju niz kriterija, ograničenja i zahtjeva nametnutih na dizajn i programske operacije. U fazi dizajna kreće proces planiranja i osmišljavanja optimalnog programskog rješenja. Razvojni programeri i dizajneri sustava moraju se dogovoriti oko dizajna programske arhitekture, konceptualne sheme i logičkog dijagrama baze podataka i dizajna GUI-a (*eng. Graphic User interface*). Faza implementacije odnosno realizacije odnosi

se na proizvodnju prethodno dogovorenih poslovnih zahtjeva i specifikacija iz koncepta u izvedeno poslovno rješenje (produkcijsku okolinu). Faza testiranja poznatija i kao faza verifikacije i validacije označava proces provjere je li proizvedeno rješenje u skladu s originalnim korisničkim zahtjevima. Na kraju, faza održavanja se odvija za cijelo vrijeme životnog ciklusa sustava u svrhu dodatne razrade procesa, ispravki novonastalih grešaka, poboljšanja performansi i kvalitete sustava (Bassil, 2012). U praksi, projekti se redovito suočavaju s problemima i nedostatcima koji rezultiraju značajnim povredama rokova i procijenjenih troškova, te ponekad i potpunim neuspjehom. Mnogi projekti se ne isporučuju na vrijeme, nisu unutar dogovorenog budžeta te ne sadrže sve unutar dogovorenih funkcionalnih zahtjeva (Bassil, 2012).

Zahvaljujući svojoj fleksibilnosti sheme, NoSQL baze podataka vrlo su popularan odabir, pogotovo u agilnom razvoju računalnih programa (Störl et al., 2017). Postoje određeni izazovi kada struktura već pohranjenih podataka u produkcijskoj bazi podataka više ne odgovara rezultatima koje očekuje ažurni aplikacijski kod. Tada se očekuje migracija *legacy* podataka. Mnogi razvojni timovi oslanjaju se na vlastite migracijske skripte, što je skupocjen proces u kontekstu troška radnih sati (odnosno čovjek/dana), te je također podložan greškama (Störl et al., 2017). U visoko kompetitivnom tržištu, organizacije trebaju vrlo brzo prepoznati i reagirati na promjene u poslovnoj strukturi. Agilni razvojni proces (Slika 12.) obećava brzo usvajanje promjena prema korisničkoj potrebi. Jedan od načina je držati troškove prilagodbe niskima. Podatkovne strukture bez sheme (NoSQL) daju mogućnost razvojnim programerima da budu u korak s vječno evoluirajućom, visoko zahtjevnom grupom korisnika (Shermin, 2013). Prema Edeki (2015) metodologija agilnog razvoj softvera široko je rasprostranjena i prihvaćena u zajednici razvojnih programera. Agilni razvoj pruža mnogostruke prednosti nad Waterfall metodologijom. Agilni razvoj pokušava pojednostaviti proces planiranja i procjene razlažući velike zahtjeve u manje, individualne zadatke (*eng. Tasks*) ili iteracije. Analiza manjih zadataka pruža razvojnom timu (posebice poslovnom analitičaru zaduženom za analizu poslovnog procesa) mogućnost točnijih procjena vremena i truda potrebnog za implementaciju potrebne promjene (*eng. Feature*) u sustavu.



Slika 12. Prikaz tijeka razvoja softvera primjenom agilne metodologije

Projektni menadžer tada može točnije prikazati razinu završenosti u postotcima, čije se brojke onda lako uspoređuju s originalnim rokovima utvrđenima u fazi planiranja. Agilni proces je dizajniran kako bi omogućio razvojnom programeru bolje planiranje vlastitog vremena potrošenog u ciklusu razvoja proizvoda. Za svaki zadatak u iteraciji, programer daje procjenu vremena koliko smatraju da im je potrebno za izvršiti navedeni zadatak, te nakon završetka unose realno vrijeme rada. Neki od alata koji se koriste za evidenciju vremena su JIRA, Trello, ASANA, YouTrack, TaskWorld, Monday i sl., a podatke unutar takvih alata analiziraju poslovni analitičari. Edeki (2015) nadalje navodi kako je drugi cilj agilne metodologije uključivanje ključnih dionika (naručitelja) u životni ciklus razvoja. Čestom komunikacijom razvojnog tima s dionicima omogućuje se rad na unaprjeđenju izvornih ideja što rezultira s proizvodom više kvalitete nakon razvojnog ciklusa. Na taj način proizvod se drži na pravom putu razvoja, bez obzira na promjenu željenog ishoda. Sheme relacijskih baza podataka potrebno je promijeniti prije promjena na aplikacijskoj razini, limitirajući mogućnosti dodavanja ili ažuriranja njenih karakteristika. U razvojnom procesu NoSQL baza podataka fokus je na domenskom dizajnu uz pomoć agilnog razvoja, te ih je također u većini slučajeva lakše postaviti i isporučiti od relacijskih baza (Ramllari, 2013). NoSQL baze podataka lako se uklapaju u moderan razvoj aplikacija. Manje je vremena potrebno za prelazak iz faze konceptualizacije do faze implementacije, posebno u kompanijama koje su usmjerene na brze odgovore prema zahtjevima promjenjivog tržišta. To posebice vrijedi u

slučajevima kada se aplikacije od početka dizajniraju kao aplikacije unutar Oblaka (*eng. Cloud-native*). Situacija je puno kompleksnija kada se aplikacija s lokalne relacijske baze podataka migrira na rješenje NoSQL-a unutar Oblaka što utječe na poslovni sloj i na sloj dohvata podataka gdje su potrebne još dodatne prilagodbe (Ramllari, 2013). Proces održavanja, pogotovo prilikom strogo definiranih ugovora o razini usluge, često nalaže visoku dostupnost podataka i razinu performansi u periodu 24/7, što je moguće ostvariti automatskim komadanjem podataka, balansiranjem opterećenja, menadžmentom vremena prebacivanja u slučaju pogreške – svim značajkama NoSQL baza podataka. (Dileep et al. 2016).

9. MONGODB

MongoDB je vrsta dokument baze podataka. Clarence, Aravindh i Shreeharsha (2012) navode kako sadrži mnoštvo dodatnih mogućnosti te je sličan relacijskoj bazi podataka. Pethuru i Ganesh (2018) definiraju MongoDB kao višeplatformsku dokument-orijentiranu bazu podataka otvorenog koda, napisanu u C++ jeziku. Harrison (2015) objašnjava kako je MongoDB JSON orijentirana dokumentna baza podataka, iako interno koristi binarno kodiranu varijantu JSON-a zvanu BSON. Takav format omogućuje smanjeno vrijeme sintaksne analize (parsiranja) naspram JSON-a te bogatiju podršku podatkovnim tipovima poput datuma i binarnih podataka. Prema službenoj dokumentaciji na mrežnim stranicama pohranjuje podatke koristeći se dokumentima BJSON formata koji se lako prilagođavaju objektno orijentiranom programiranju (OOP). Upotrebljava JavaScript kao jezik za rad s bazom podataka. Sadrži dinamične sheme koje mogu rasti rastom aplikacije ne zahtijevajući skupe migracije sustava. Uključuje automatsko horizontalno skaliranje, prirodno preslikavanje i automatski povratak (*eng. Failover*), bogate upite s potpunom indeksnom potporom, čime postiže brže vrijeme produktivnosti te je stvoren za agilni razvoj. Harrison (2015) također navodi kako u njegovoj arhitekturi postoje Gospodar čvorovi – čvorovi koji sadrže specijalizirane supervizorske funkcije, koordiniraju aktivnosti ostalih čvorova te vode bilješke o trenutnom stanju klastera unutar baze podataka. Autor smatra MongoDB jednom od vodećih NoSQL baza podataka zbog „*developer-friendly*“ ekosustava i arhitekture te se koristi u mnogim vrhunskim mrežnim stranicama velikih dimenzija.

9.1. MongoDB i ACID svojstva

Prema dubinskoj analizi ACID svojstva mogu se do određene mjere poštovati unutar MongoDB-a. Navedeni rad Deepaka (2016) može se sumirati prema vrijednostima akronima:

- **Atomarnost:** MongoDB pruža svojstvo atomarnosti na razini pojedinačnog dokumenta prilikom operacije zapisivanja ili ažuriranja podatka unutar dokumenta s obzirom na to da se zapisivanje događa ili u cijelosti, ili se poništava. Također, atomarnost ne postoji na razinama višestrukih dokumenata ili kolekcija.
- **Konzistentnost:** vrlo je snažna u primarnom serveru, čak i prilikom repliciranja konfiguracije seta, no druga razina čvorova može garantirati tek naknadnu konzistentnost (*eventually consistent*). MongoDB onemogućuje čitanje podataka sa sekundarnih servera zbog mogućnosti nekonzistentnosti podataka, no te postavke se

mogu promijeniti. Prema CAP teoremu, nije moguće osigurati konzistentnost i dostupnost podataka u istom trenutku.

- **Izolacija:** ovo svojstvo moguće je osigurati pomoću operacija poput *$isolation* i paterna poput *„update if current“*.
- **Izdržljivost:** MongoDB pruža razvojnim programerima mogućnost konfiguracije operacije zapisa na aplikaciju tek nakon puštanja na podatkovni dnevnik na disku pomoću parametara *sncdelay* i *journalCommitInterval*, što je slično modelu korištenom u relacijskim bazama podataka za osiguravanje izdržljivosti. Ako je primarni cilj MongoDB baze izdrživost, moguće je izgubiti dio performansi sustava zbog zapisa podataka na velik broj servera.

S navedenim se slaže i Harrison (2015: 131) navodeći kako MongoDB pruža konzistentnost na razini jednog dokumenta. Prilikom ažuriranja dokumenta zaključan je za čitanje i dodatno ažuriranje u drugim sesijama. Konzistentnost na višoj razini se postiže pomoću lokota (*eng. Locks*) koji se koriste kako bi se osiguralo da dva istovremena ažuriranja ne mogu simultano modificirati dokument te kako čitatelj ne može vidjeti nekonzistentan prikaz podataka. Lokoti se pridružuju na razini dokumenta, osiguravajući da je kolekcija spremljena pomoću stroja za pohranjivanje *(eng. Storage engine).*

9.2. Arhitektura MongoDB-a

U MongoDB-u podržava se komadanje (*eng. Sharding*) kako bi se s ciljem visoke razine dostupnosti omogućile sposobnosti horizontalnog širenja i replikacije podataka. U njegovoj arhitekturi svaki *Shard* implementiran je zasebnom MongoDB bazom podataka te sami *Shardovi* nisu svjesni svoje uloge unutar šireg sustava. Potrebno je postaviti zaseban, konfiguracijski MongoDB server kako bi se upravljalo meta-podacima s kojima se odlučuje na koji način se podaci distribuiraju kroz cijeli sustav. Usmjerni proces je odgovoran za odabir puta kroz mrežu kako bi zahtjev išao prema određenom *Shard* serveru. Kolekcija JSON dokumenata za proces komadanja zahtjeva poseban ključ (*eng. Shard key*) koji čine jedan ili više indeksiranih atributa s pomoću kojeg će se odlučiti distribucija dokumenata preko skupine *Shardova* (Harrison, 2015: 110-111). Postoje razni mehanizmi u tom procesu, poput particioniranja temeljenog na opsegu, gdje je svakom *Shardu* osiguran specifičan opseg vrijednosti posebnih ključeva. Drugi mehanizam particioniranja je pomoću kontrolnog identifikacijskog broja (*eng. Hash key*), gdje se ti ključevi distribuiraju pomoću funkcije kontrolnog identifikacijskog broja prema *shard* ključu. Komadi su uvijek u kombinaciji s

replikacijom podataka kako bi se omogućila skalabilnost i dostupnost u produkcijskoj MongoDB bazi podataka.

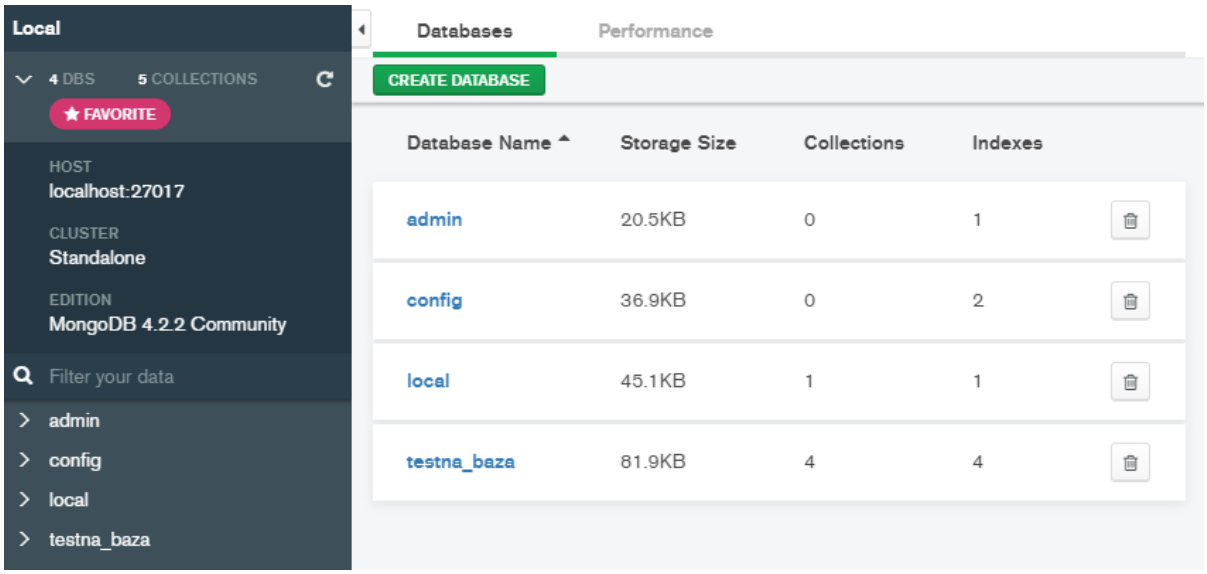
9.3. MongoDB Atlas i MongoDB Compass

Nastavljajući analizu službene dokumentacije na mrežnim stranicama, vidljivo je kako je MongoDB Atlas je mrežni program usmjeren na spajanje na Oblak te sadrži lokalnu administraciju, dok je MongoDB Compass lokalni program za administraciju MongoDB dokumentne baze podataka. Oni nisu jedini GUI za administraciju MongoDB baze, no jedni su od najpopularnijih. Ovi sustavi omogućuju:

- Statistiku u stvarnom vremenu koja daje provjeru ključnih metrika servera te operacija u bazi. Moguće je lako doći do pravih informacija u baznim operacijama i shvatiti koje su najaktivnije kolekcije podataka.
- provjeru u jednom pogledu na koji se server spaja. Moguće je navigirati među razinama instanci, baze podataka i kolekcije s filtriranjem svog prostora imena (*eng. Namespace*).
- Analizu dokumenata i prikaz bogate strukture unutar kolekcija kroz intuitivni GUI.
- Brzu vizualizaciju i istraživanje kreiranih shema kako bi shvatili frekventnost, tip i raspon polja u svom setu podataka.
- Vizualizaciju, shvaćanje i rad nad Geo-spacijalnim podacima. Mogu se konstruirati sofisticirani upiti te je rezultat moguće prikazati i grafički i u obliku skupa JSON dokumenata.
- Modifikaciju postojećih dokumenata s većim pouzdanjem koristeći intuitivni vizualni editor, ili unos novih dokumenata i kloniranje ili brisanje već postojećih.
- Shvaćanje tipa i veličine vlastitih indeksa, njihove primjene i posebnih mogućnosti.
- Stvaranje pravila validacijske sheme pomoću pametnog editora koji auto-sugerira komponente pravila. Moguće je vidjeti trenutne rezultate s pregledom u stvarnom vremenu i promjeni pravila ako je potrebno.
- Bolje editiranje s validacijom individualnih BSON tipova.
- Konekcije s obzirom na replikaciju setova omogućuju kontinuirano korištenje tijekom replikacije konfiguracije seta i promjena te pružaju dodatne informacije o spojenom klasteru servera.
- Jednostavan pristup i upravljanje izvršenim upitima. Moguće je i sačuvati često izvršavane upite pod omiljene.

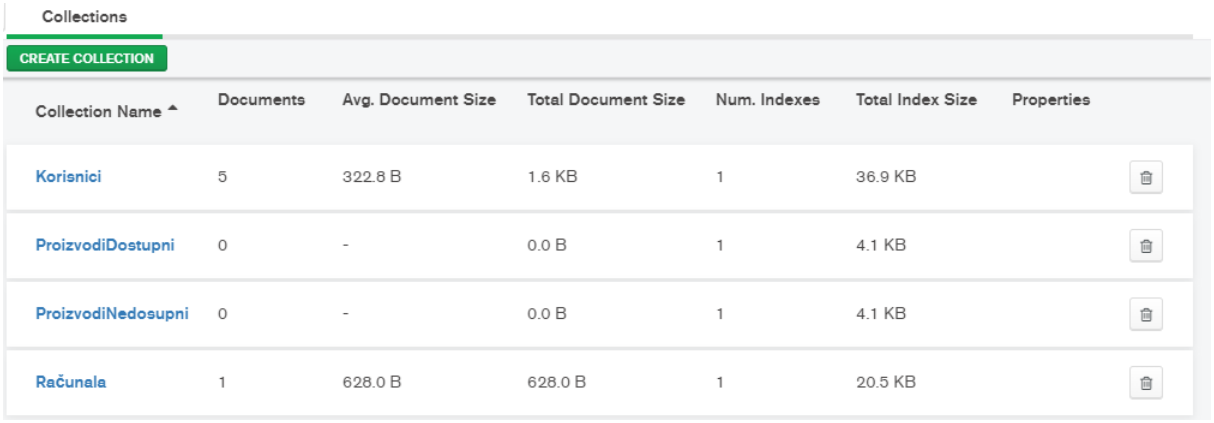
9.4. Primjer korištenja

Želimo primjerice dokumentnu bazu od početka iskoristiti za kolekciju i administraciju podataka u vlastitoj firmi, čija je djelatnost nabava i prodaja stolnih računala i računalnih komponenata. Želimo imati i registrirane korisnike na našoj mrežnoj stranici kako bi ostvarili popuste prilikom kupovine te kako bismo mogli detaljnije analizirati podatke i potrebe naših klijenata, poput njihovog mjesta stanovanja, dobi (*DoB, eng. Date of Birth*), zanimanja, popisa kupljenih uređaja i sl. Također, u ponudi primjerice imamo i prethodno sastavljena računala. Iako svako sastavljeno računalo ima donekle isti popis komponenata, bavimo se i sklopovima ostale informatičke opreme, njihovim softverom i sl. te su nam potrebne tablice koje je lako dopuniti s podacima, kakvi god oni bili u budućnosti, u kombinaciji s promjenjivim podacima korisnika. Za potrebe ovog rada napravljena je na MongoDB Compass-u verziji *4.2.2. Community edition* lokalna baza/server na *localhost:27017* bez dodatne autentifikacije (Slika 13.).



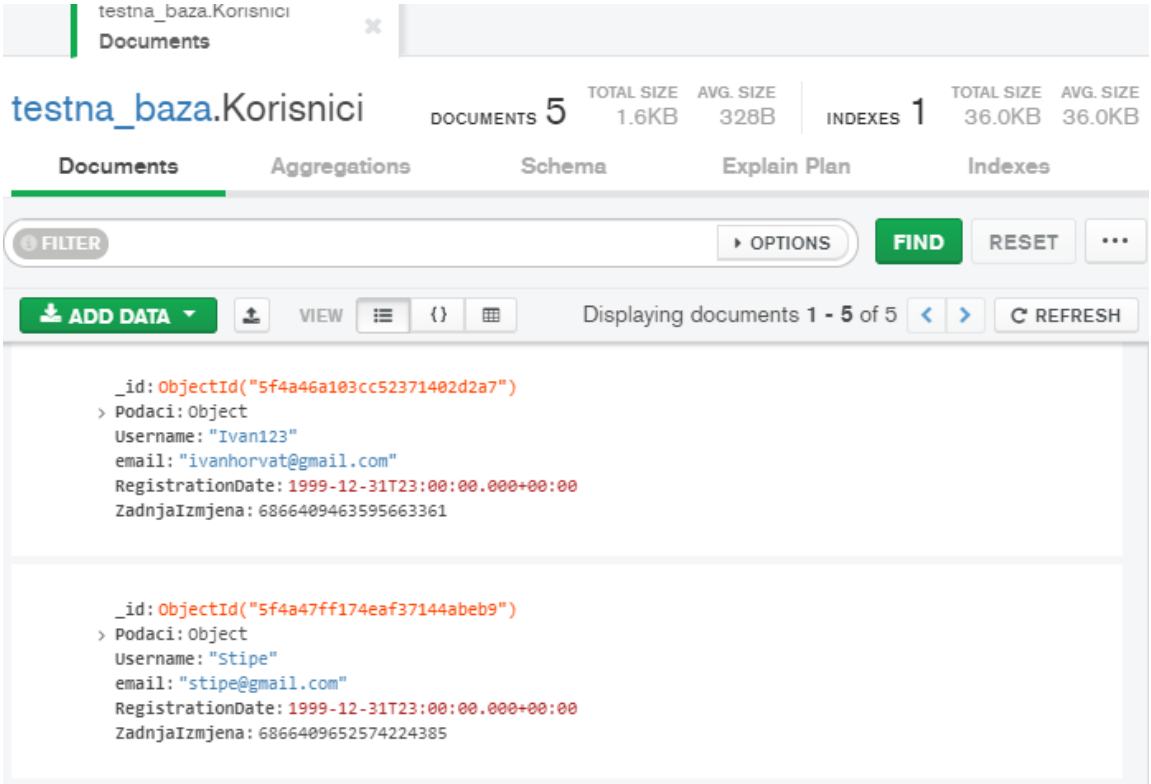
Slika 13. Prikaz popisa servera unutar MongoDB Compassa

Nakon spajanja ponuđene su nam sve baze dokumenata koje imamo na određenoj lokaciji, odnosno serveru. Za primjere napravili smo bazu pod imenom *testna_baza* dokumenata. Prema osnovnim podacima, u njoj se nalaze četiri kolekcije dokumenata bez dodatno definiranih indeksa. Za potrebe rada naše fiktivne kompanije postavili smo kolekciju dokumenta korisnika, odnosno svih osoba koje će se registrirati na našu stranicu. Imamo kolekciju dostupnih proizvoda trenutno u našem asortimanu te proizvoda koje je potrebno naručiti jer nisu dostupni. Pod posebnom kolekcijom nalaze se prethodno složena računala.



Slika 14. Prikaz kolekcije dokumenata

Unutar kolekcije korisničkih profila (Slika 14.), imamo velik broj korisnih informacija navedenih u sučelju. Naime, u okvirima oko samih podataka, dostupan nam je broj dokumenata unutar kolekcije, njihova ukupna veličina te prosječna veličina u byte-ima. Dokumente je moguće prenijeti s računala ili ručno pisati. Svakom dokumentu prilikom ručnog definiranja automatski je generiran *ObjectId*, to jest unikatni ključ. Uz klasičan uvoz podataka, Compass nudi i ručni unos podataka na vrlo jednostavan način – u obliku redova, JSON strukture dokumenta ili čak tablica. Unutar dokumenta mogu se definirati objekti, varijable s pripadajućim vrijednostima (Slika 15.) te njihovi tipovi podataka (*Array, Boolean, Decimal, Object, String, Int32, Int64,(New) Date, Timestamp, itd.*)



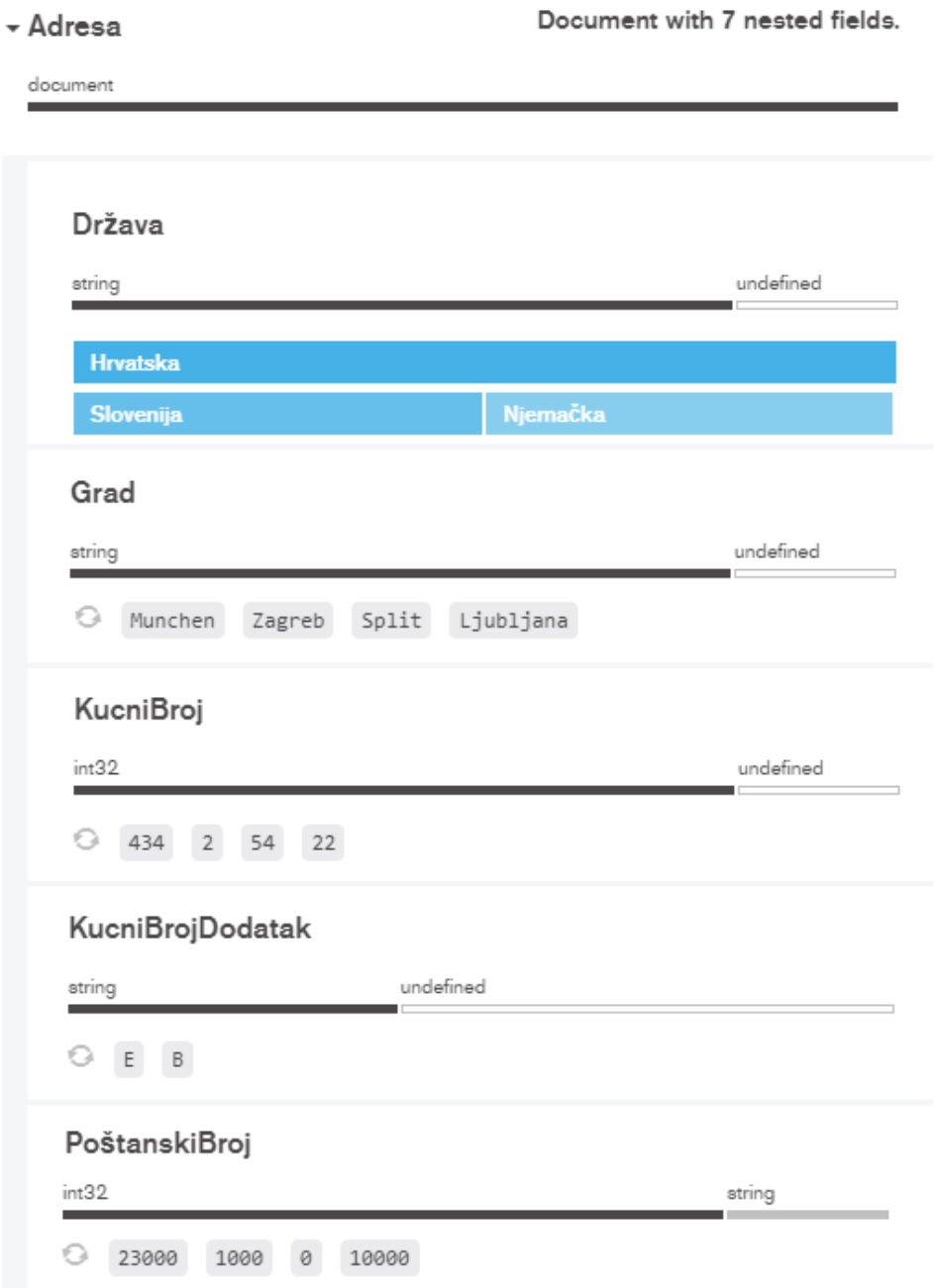
Slika 15. Prikaz dokumenata u pojedinačnoj kolekciji

Primjer vrijednosti: `{ "Država": "Hrvatska" }`

U polju *Filter* moguće je definirati željeni upit pomoću vrlo jednostavne sintakse za koju nije potrebno prethodno znanje programiranja.

Primjer jednostavnog upita: `{'Podaci.Adresa.Država': 'Hrvatska'}`

Moguće je napraviti i analizu sheme svake kolekcije dokumenata (Slika 16.) što omoguće vrlo jasan i detaljan grafički uvid u nazive varijabli, tipove njihovih podataka, te njihovu distribuciju. Primjerice, ako je neki podatak *Undefined*, označava kako nije definiran unutar same strukture dokumenata te kolekcije.



Slika 16. Prikaz analize sheme kolekcije dokumenata

Ako je recimo neki podatak drugačijeg tipa (Slika 17.), poput *PoštanskiBroj:(string)* jednostavno ga je identificirati te Compass sam nudi upit prema traženom dokumentu kako bismo prilagodili dokument.

Slika 17. Identifikacija različitih vrsta podataka pomoću analize sheme

Navedeni primjeri (prema Slikama 13-17.) vrlo su osnovne funkcionalnosti MongoDB-a unutar Compass GUI-a. Cilj je prikazati početak rada s MongoDB Compass-om, osnovni način unosa podataka u bazu, osnovne upite nad podacima jednostavnog JSON formata te način agregacije podataka kroz analizu sheme.

10. ZAKLJUČAK

NoSQL baze podataka relativno su nove baze podataka, čija je namjena proizašla iz sve većih zahtjeva ICT područja i Big Data. One su nerelacijske baze podataka čija je svrha organizacija i analiza različitih tipova podataka izrazito visokog volumena. Horizontalno su skalabilne na široko dostupnom hardveru (virtualnim i fizičkim serverima) što ih čini ekonomski isplativijima u usporedbi s tradicionalnim relacijskim bazama. Diskutabilno su jednostavne za korištenje te se zasnivaju na CAP i nastavno BASE teoremu koji im omogućuju fleksibilnost u radu s dinamičnim količinama podataka. Prema glavnoj podjeli dijele se na Ključ-vrijednost, Dokument, Stupčane i Graf baze podataka. Ključ-vrijednost najjednostavnije su NoSQL baze podataka. Vrijednosti različitih vrsta podataka uparuju se s odgovarajućim unikatnim ključevima. Primjenu pronalaze u izradi baza podataka za trgovinske lance, senzorna očitanja i sl. Dokument baze podataka imaju nekoliko vrsta dokumenata s kojima raspolažu (XML, JSON i BSON) te su jedan od najpopularnijih izbora nerelacijske baze podataka, koristeći MongoDB i pripadajuće sustave za menadžment baze podataka. Stupčane baze podataka ključevima reda pridružuju numerirane stupce i redove. Primjer korištenja je Apache Cassandra koju koristi Facebook. Graf baze podataka sastoje se od čvorova i lukova koji imaju određena svojstva. Svaki čvor mora biti spojen s drugim čvorom pomoću luka. Ovakve baze koriste se kada su prioritet odnosi među podacima, poput opskrbnih lanaca i logistike ili sustava za prepoznavanje prevare. Zbog spomenute mogućnosti horizontalnog skaliranja nerelacijske baze podataka služe se određenim tehnikama koje pripomažu postupku izrade odgovarajuće arhitekture sustava (tehnike poput Gospodar-rob replikacija, komadanje, model Dynamo). U malim i velikim programskim kompanijama razvoju NoSQL baza podataka od početnih koraka najviše pogoduje agilna metodologija razvoja sustava zbog istaknutih prednosti nad klasičnim modelima razvoja softvera (primjerice Waterfall metodologijom). U agilnoj metodologiji cijeli tijek razvoja podijeljen je na manje zadatke. Razvojni programer planira vlastito vrijeme potrebno za njihovo izvršavanje, na temelju čije procjene poslovni analitičari i voditelji projekta rade daljnje prilagodbe rokova i dizajna sustava u sukladnosti s korisnicima. Za posao razvojnog programera ili administratora baze podataka, očekuje se stručno obrazovan kadar. U Republici Hrvatskoj, u akademskoj godini 2020/21 prema analizi nastavnih planova moguće je steći formalno obrazovanje u području nerelacijskih baza podataka na deset fakulteta, s ponudom od ukupno 11 kolegija.

11. LITERATURA

1. Bassil, Y. (2012) *A simulation model for the Waterfall software development life cycle.* International Journal of Engineering & Technology (iJET). Vol. 2, No. 5.
2. Cesar, J. (2018) *Aplikacije temeljene na NoSQL i relacijskim bazama podataka*. Diplomski rad. Fakultet organizacije i informatike Varaždin.
3. Clarence, J. M. T. Aravindh, S. Shreeharsha, A. B. (2012) *Comparative study of the New Generation, Agile, Scalable, High Performance NOSQL databases*. International Journal of Computer Applications. Volume 48, No. 20.
4. Codd, F. E. (1970) *Relational model of data for large shared data banks*. Communications of the ACM. Vol 13 (6), 1970. pp. 377-387. Association for Computing Machinery, Inc.
5. Deepak, C. G. (2016) *A critical comparison of NOSQL databases in the context of ACID and BASE*. Culminating Projects in Information Assurance. St. Cloud State University, Minnesota.
6. Dileep, K. G. et al. (2016*) Changing landscape from RDBMS to operational NoSQL systems*. International Journal of Computer Science and Information Security (IJCSIS). Vol. 14, No. 10.
7. Đurica, N. et al. (2018) *Percepcije studenata o važnosti upotrebe Facebooka za akademske svrhe*. Croatian Journal of Education. Vol.20; No.4/2018, pp: 1059-1087.
8. Edeki, C. (2015) *Agile software defvelopment methodology*. European Journal of Mathematics and Computer Science Vol. 2 No. 1.
9. Elmasri, R. Navathe, S. B. (2016) *Fundamentals of Database systems*. 7th edition. Pearson, NewYork.
10. Gačić, J. (2017) *NoSQL baze podataka*. Diplomski rad. Prirodoslovno-matematički fakultet. Sveučilište u Zagrebu.
11. Gray, J. Reuter, A. (1993) *Transaction Processing: concepts and techniques.* pp 1-20. Morgan Kaufmann Publishers. San Francisco, CA.
12. Hanwahr, C. N. (2017) *"Mr. Database" - Jim Gray and the history of database Technologies*. Springer International Publishing AG.
13. Harrison, G. (2015) *Next generation databases. NoSQL, NewSQL and Big Data.* Apress Media, LLC. Sprincer Science+Business Media Finance Inc., NewYork.
14. Hrvatska Enciklopedija, mrežno izdanje. *Baza podataka*. Leksikografski zavod Miroslav Krleža, 2020 URL: https://bit.ly/2sflvoM. Datum pristupa: 10.01.2020.

15. Hrvatska enciklopedija, mrežno izdanje. *Definicija.* Leksikografski zavod Miroslav Krleža, 2020. URL: https://bit.ly/3ho1wrO Datum pristupa 22.7.2020
16. Ken K. L. Wai-Choi, T. Kup-Sze, C. (2013). *Alternatives to relational database: comparison of NoSQL and XML approaches for clinical data storage*. Computer Methods and Programs in Biomedicine. Volume 110, Issue 1, Pp. 99-109.
17. Monnappa, A. (2019) *The rise of NoSQL and why it should matter to You*. URL: https://bit.ly/36Hpkls. Zadnje mijenjano: 03.12.2019. Datum pristupa: 14.01.2020.
18. Olivera, L. (2019) *Everything you need to know about NoSQL*. https://dev.to/lmolivera/everything-you-need-to-know-about-nosql-databases-3o3h. Zadnje mijenjano: 05.06.19. Datum pristupa: 15.01.2020.
19. Panian, Ž. (2005) *Hrvatsko engleski informatičko enciklopedijski rječnik.* Europapress holding, Zagreb.
20. Pethuru, R. Ganesh, C. D. (2018*) A deep dive into NoSQL databases: the uses and applications*. Advances in computers volume 109. Academic Press, Elservier. Pp. 1-48.
21. Petrovečki, M. Kovačević, Ž. (2019) *Istaknute značajke SQL Servera 2019*. Pregledni rad. Politehnika i dizajn, Vol. 7, No. 1, 2019. Tehničko Veleučilište Zagreb.
22. Ramllari, L. (2013*) Extending a methodology for migration of the database layer to the Cloud considering relational database schema migration to NoSQL*. University of Stuttgart. Institute of Architecture of Application Systems. Master Thesis.
23. Raj, P. Deka, G. C. (2018) *A Deep Dive into NoSQL Databases: The use cases and applications*. First Edition. Academic Press. Oxford, UK.
24. Sadalage, P. J. Fowler, M. (2013) *NoSQL distilled: a brief guide to emerging world of polyglot persistance*. Addison-Wesley. Pearson Education, Inc.
25. Sawant, N. Shah, H. (2013) *Big Data Application Arhitecture Q & A. A problem solution approach*. Apress Media LCC. NewYork. Springer Science+Business.
26. Shahriar, H. Haddad, S. H. (2017) *Security Vulnerabilities of NoSQL and SQL Databases for MOOC Applications*. International Journal of Digital Society (IJDS). Volume 8. Iss 1.
27. Shermin, M. (2013) *An Access Control Model for NoSQL Databases*. Electronic Thesis and Dissertation Repository. University of Western Ontario.
28. Simović, V. Varga, M. Oreški, P. (2017) *Standard Languages for Creating a Database to Display Financial Statements on a Web Application*. International Journal of Computer, Electrical, Automation, Control and Information Engineering. Vol:11, No:2. Pp. 117-120.
29. Stojanović, A. (2016) *Osvrt na NOSQL baze podataka – četiri osnovne tehnologije*. Politehnika i dizajn, Vol. 4, No. 1, 2016. Tehničko Veleučilište Zagreb.

30. SQL Vs NoSQL - *Exact Differences And Know When To Use NoSQL And SQL*. URL: https://www.softwaretestinghelp.com/sql-vs-nosql/ Zadnje mijenjano: 02.08.2020. Datum pristupa: 15.08.2020.
31. Störl, U. Müller, D. Klettke, M. Scherzinger, S. (2017) *Enabling Efficient Agile Software Development of NoSQL-backed Applications*. Datenbanksysteme für Business, Technologie und Web (BTW 2017). Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn.
32. Šimović, V. (2010). *Uvod u informacijske sustave*. 2. Dopunjeno i izmijenjeno izdanje. pp. 162-195. Golden Marketing – tehnička knjiga. Učiteljski fakultet. Sveučilište u Zagrebu. Zagreb, Hrvatska.
33. *2019 Database Trends – SQL vs. NoSQL, Top Databases, Single vs. Multiple Database Use*. URL: https://scalegrid.io/blog/2019-database-trends-sql-vs-nosql-top-databases-single-vs-multiple-database-use/ Zadnje mijenjano: 08.03.2019. Datum pristupa: 13.08.2020.

HRVATSKI POJMOVNIK

BP – Baza Podataka

SUBP – Sustav Upravljanja Bazom Podataka

OLTP – On-line Transakcijski Proces

ENGLESKI POJMOVNIK

AJAX – Asynchronous JavaScript and XML

ANSI – American National Standards Institute

BSON – Binary JSON

DB – Database

DBaaS – Database as a Service

DBMS – Database Menagement System

GUI – Graphical User Interface

ICT – Information And Communication Technology

IoT – Internet of Things

JSON – JavaScript Object Notation

MsSQLS – Microsoft SQL Server

NoSQL – NotOnly Structured Query Language

ODBC – Open Database Connectivity

OLTP – Online Transaction Processing

PDF – Portable Document Format

RDBMS – Relational Database Menagement System

REST – Representational state transfer

SDLC – Software Development Life Cycle

SQL – Structured Query Language

URL – Uniform Resource Locator

XML – EXtensible Markup Language

POPIS SLIKA

PRILOZI

PRILOG 1 – pregled podjela radova na temu NoSQL-a

Pregled je napravljen prema podacima preuzetima u Nacionalnom repozitoriju završnih i diplomskih radova ZIR (pretraga prema upitu: „NoSQL“), datuma 28.07.20.

PODJELA PREMA USTANOVAMA I ODJELIMA

Sveučilište u Zagrebu(49)
Sveučilište Josipa Jurja Strossmayera u Osijeku(6)
Sveučilište u Rijeci(4)
Sveučilište Jurja Dobrile u Puli(4)
Veleučilište u Rijeci(3)
Sveučilište u Dubrovniku(2)
Međimursko Veleučilište u Čakovcu(2)
Sveučilište u Splitu(1)
Veleučilište u Požegi(1)
Visoko učilište Algebra(1)

PODJELA PREMA PODRUČJIMA

Tehničke znanosti(38)
Društvene znanosti(32)
Prirodne znanosti(3)

PODJELA PREMA VRSTI RADA

diplomski rad(41)
završni rad(30)
specijalistički diplomski stručni(2)

PODJELA PREMA KLJUČNIM RIJEČIMA

NoSQL(34)	JavaScript(8)
MongoDB(17)	nosql(8)
SQL(15)	JSON(7)
baza podataka(9)	Big Data(5)
baze podataka(9)	Firebase(5)

PRILOG 2 – tablica pregleda silabusa fakulteta u RH u akademskoj godini 2020/21.

lokacija	naziv visokog učilišta	smjer/modul	oblik	razina	razina	vrsta	NAZIV kolegija	ECTS
Zagreb	Fakultet elektrotehnike i računalstva	Programsko inženjerstvo i informacijski sustavi, Računalno inženjerstvo, Računalna znanost	sveučilišni	diplomski	1. god	izborni	Napredni modeli i baze podataka	5
	Prirodoslovno-matematički fakultet	Računarstvo i matematika	sveučilišni	diplomski	1. god	izborni	Napredne baze podataka	5
	Fakultet strojarstva i brodogradnje							
	Filozofski fakultet	Informacijske znanosti						
	Tehničko veleučilište Zagreb	Informatika	specijalistički	diplomski	1. god	izborni	NOSQL i napredna big data analitika	6
Split	Fakultet elektrotehnike, strojarstva i brodogradnje	*svi smjerovi*						
	Prirodoslovno-matematički fakultet	Matematika - Statistika i računarstvo	sveučilišni	diplomski	1. god	izborni	Raspodijeljene i nerelacijske baze podataka	5
Varaždin	Fakultet organizacije i informatike	Informacijsko i programsko inženjerstvo, Baze podataka i baze znanja	sveučilišni	diplomski	1. god	izborni	Teorija baza podataka	5
Rijeka	Veleučilište u Rijeci	Informacijske tehnologije u poslovnim sustavima	specijalistički	diplomski	1. god	obavezan	Upravljanje bazama podataka	4
	Svučilište u Rijeci - odijel za informatiku	Poslovna informatika i Informacijski i komunikacijski sustavi	sveučilišni	diplomski	1. god	obavezan	Sustavi za podršku odlučivanju	6
Osijek	Fakultet elektrotehnike, računarstva i informacijskih tehnologija	Računarstvo	sveučilišni	diplomski	1. god	obavezan	Računarstvo usluga i analiza podataka	6
		Informatika	specijalistički	diplomski	1. god	izborni	Oblikovanje baza podataka	5
	Odjel za matematiku - Sveučilište Josipa Jurja Strossmajera	Matematika i računarstvo	sveučilišni	diplomski	1. god	izborni	Dizajniranje i modeliranje baza podataka	6
Pula	Sveučilište Juraj Dobrila u Puli	Informatika, Nastavni smjer informatike						
Šibenik	Sveučilište u Zagrebu - Šibenik	Studij energetske učinkovitosti i obnovljivih izvora						
Zadar	Informacijske znanosti	Informacijske znanosti						

NoSQL baze podataka

SAŽETAK

Nerelacijske, NoSQL baze podataka trenutno najbolje odgovaraju rastućim trendovima ICT industrije za skladištenje i administraciju sve veće količine podataka u svijetu. Predstavljaju odmak od tradicionalnih SQL baza podataka zbog svojih mogućnosti lake horizontalne skalabilnosti (hardverski i virtualno), boljih performansi upita te prilagodljivosti arhitekture specifičnim poslovnim potrebama, no s određenim ustupcima po pitanju sigurnosti podataka (popust autorizacije i autentifikacije korisnike). S obzirom na tip podataka koji se pohranjuje, NoSQL baze podataka obično dijelimo na četiri tipa: Ključ-vrijednost baze podataka, Dokument baze podataka (s podjelom na XML i JSON), Stupčane baze podataka i Graf baze podataka. Svaka od njih koristi se u specifičnim ulogama no temelje se na zajedničkim svojstvima teorema CAP i nasljedno BASE, koji im osiguravaju dostupnost podataka i njihovu konzistentnost. U postupku razvoja NOSQL baze podataka za moderne aplikacije i sustave najčešće se koristi agilna metodologija rada, čineći isporuku završenog proizvoda i njegovo održavanje manje kompleksnim. MongoDB je primjer dokument baza podataka s pripadajućim menadžment sustavom za njeno upravljanje, zvanim MongoDB Compass, detaljnije razrađenim u ovom radu.

Ključne riječi: NoSQL baza podataka, sigurnost, agilna metodologija, MongoDB Compass

NoSQL databases

SUMMARY

Non-relational NoSQL databases are best fitted for growing trends in ICT industry regarding their needs for warehousing and administration of ever growing data in the world. They represent a certain detachment from the traditional SQL databases in terms of their easy horizontal scalability (both with hardware and virtually), better query performance and architecture adaptability for specific needs, but with certain drawbacks in terms of data security (user authorization and authentication, for example). Regarding the type of data they store, four types of NoSQL databases can be named: Key-Value databases, Document databases (with further distinction for XML and JSON/BSON data types), Column-family databases and Graph databases. Every one of them is used in specific roles, but they have mutual properties in CAP and BASE theorems, which ensure overall availability and consistency of stored data. In development of NoSQL databases for modern applications and systems agile methodology is most often used, making the deployment and maintenance of the finished product less complex. MongoDB is an example of document database with his management system called MongoDB Compass, which is in this thesis explained in more detail.

Key words: NoSQL database, security, agile metodology, MongoDB Compass