Usporedba graf i relacijskih baza podataka

Brica, Marko

Master's thesis / Diplomski rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:914612

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-02-05**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

USPOREDBA GRAF I RELACIJSKIH BAZA PODATAKA

Diplomski rad

Marko Brica

Osijek, 2018.

Z A H V A L A

Zahvaljujem se svojim roditeljima, Ivanu i Anđi, bratu Peri, sestri Marini i stricu Pavi što su mi omogućili studiranje u drugome gradu i što su mi bili najveća podrška tijekom moga studija.

Također, zahvaljujem se svome mentoru izv. prof. dr. sc. Krešimiru Nenadić i sumentoru Krešimiru Romić, mag. ing. comp. na pruženoj pomoći i savjetima tijekom izrade ovoga rada, kao i svim prijateljima, kolegama i ostalima koji su pomogli na bilo koji način tijekom moga studija.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 13.07.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Marko Brica
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 794 R, 26.09.2017.
OIB studenta:	70602983579
Mentor:	Izv. prof. dr. sc. Krešimir Nenadić
Sumentor:	Krešimir Romić
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Alfonzo Baumgartner
Član Povjerenstva:	Krešimir Romić
Naslov diplomskog rada:	Usporedba graf i relacijskih baza podataka
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	Opisati način rada graf baze podataka i usporediti ga s relacijskim bazama podataka. Opisati jezik za upravljanje i strukturu graf baze podataka. Na primjeru pokazati funkcionalnosti graf baze podataka korištenjem Neo4j platforme. Opisati prednosti i nedostatke s naglaskom na brzinu izvođenja upita. Sumentor: Krešimir Romić, mag.inž.rač.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	13.07.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 17.07.2018.

Ime i prezime studenta:	Marko Brica
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 794 R, 26.09.2017.
Ephorus podudaranje [%]:	9%

Ovom izjavom izjavljujem da je rad pod nazivom**: Usporedba graf i relacijskih baza podataka**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Krešimir Nenadić

i sumentora Krešimir Romić

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

U svima ljudskim djelatnostima, prilikom rada potrebno je voditi određenu vrstu evidencije. Pojava informatike i razvoj informacijskih znanosti uvelike je pomogao vođenje takvih evidencija, odnosno olakšao je vođenje takvih evidencija. Olakšanje kod ovakvih radnji je u vidu toga da se podaci koji su potrebni za obavljanje određenog posla sada mogu elektronički, korištenjem računala obrađivati i spremati u neku vrstu baze podataka. Danas, kada je informatizacija i digitalizacija prisutna u svim oblicima ljudskog života i djelovanja, svaki dan se susreću aktivnosti koje su povezane sa bazom podataka – od kupovine namirnica u trgovinama mješovite robe, preko podizanja novca na bankomatima, pa sve do pretraživanja Interneta radi razonode, obavljanja kupnje preko Interneta ili obavljanja određenih poslova.

Pored gore navedenoga, upotreba baza podataka je prisutna ponajviše u administrativnim poslovima i poslovnim informacijskim sustavima. Budući da djelatnici koji obavljaju administrativne poslove i koriste spomenute informacijske sustave, u pravilu, nisu stručnjaci iz područja informacijskih tehnologija i znanosti, pored baze podataka izrađuje se i aplikacija za pristup i obavljanje traženih radnji u bazi podataka. Takva aplikacija može biti mobilna aplikacija, aplikacija kojoj se pristupa putem Interneta ili instalirana na korisnikovo računalo.

Značajan utjecaj na razvoj baza podataka imala je pojava relacijskih baza podataka. Svoju popularnost su zadržale i do danas, te je ova vrsta u sadašnjosti najkorištenija vrsta baza podataka. Razlog tome je što relacijske baze podataka svojom strukturom i načinom rada omogućuju vrlo efikasno obavljanje raznih poslova. Kako i samo ime kaže, u relacijskim bazama podataka, podaci su povezani u relacije. Točnije, unutar baze podataka se nalaze tablice u koje se pohranjuju podaci, a jedan ili više podataka čine primarni ključ tablice koji jednoznačno identificira jedan zapis te tablice. Ukoliko je potrebno tablice se mogu povezati u relaciju korištenjem stranih (vanjskih) ključeva. Za sada će biti rečeno samo to da korištenje vanjskih ključeva utječe na performanse upita, što je u današnje doba jako bitna stavka, jer nastoji se da podaci budu dostupni u stvarnome vremenu uz što manje troškove. Više o relacijskim bazama podataka biti će rečeno u trećem poglavlju ovoga rada.

Kao rješenje gore spomenutoga problema, zadnjih godina svoj uspon su doživjele nerelacijske baze podataka (NoSQL baze podataka). NoSQL baze podataka svoj uspon temelje na tome da rješavaju određene probleme koji se susreću prilikom uporabe relacijskih baza podataka. Također, i o NoSQL bazama podataka više će biti rečeno u nastavku rada. Sada će biti spomenuto samo da postoji više vrsta NoSQL baza podataka. Jedna od njih su graf baze podataka koje će pored

relacijskih baza podataka također biti obrađene u ovome radu. Dakle, rad će imati u fokusu usporedbu relacijskih i graf baza podataka.

Rad se sastoji od više poglavlja, s pripadajućim potpoglavljima. U prvome, uvodnome poglavlju dan je uvod u rad i zadatak rada. Drugo poglavlje opisuje podatke, njihova svojstava te vrste podataka koji se mogu pohranjivati u razne baze podataka. Treće poglavlje opisuje relacijski model podataka. U njemu je opisan konceptualni i logički model (relacijska shema), SQL jezik za rad sa podacima u relacijskoj bazi podataka, te sustav za upravljanje bazom podataka. Četvrto poglavlje sadrži opis nerelacijskog modela baze podataka. U njemu su opisana svojstva nerelacijskih baza podataka. Isto tako, ukratko su opisane sve vrste nerelacijskih baza podataka. Poseban naglasak stavljen je na graf baze podataka koje se opisane detaljnije, počevši od matematičke teorije grafova, modeliranja graf baze podataka, konkretnog opisa Neo4j platforme pa sve do Cypher jezika. U zadnjem, petom poglavlju dan je opis usporedbe između relacijske i graf baze podataka. Opis usporedbe obuhvaća teorijske aspekte usporedbe, provedbu usporedbe, te analizu rezultata usporedbe.

1.1. Zadatak diplomskog rada

Opisati način rada graf baze podataka i usporediti ga s relacijskim bazama podataka. Opisati jezik za upravljanje i strukturu graf baze podataka. Na primjeru pokazati funkcionalnosti graf baze podataka korištenjem Neo4j platforme. Opisati prednosti i nedostatke s naglaskom na brzinu izvođenja upita.

2. VRSTE PODATAKA

Naziv baza podataka jasno ukazuje na to da se radi o i s podacima. U današnje vrijeme, podatak može biti bilo što i može ga generirati bilo kakav uređaj. Tako danas postoje senzori koji mjere i bilježe neke podatke (npr. vlaga u zraku, temperatura zraka, vitalne funkcije čovjeka, podaci s pretraživanja Interneta i slično). U određenim slučajevima važno je pohranjivati te podatke radi njihove ponovne upotrebe i analize tih podataka.

Važno je znati da svi podaci koji se generiraju nisu istoga tipa i da nisu u istoj formi. Neki podaci su strukturirani i pohranjuju se u relacijske baze podataka, dok su drugi nestrukturirani (dokumenti, slike, videozapise i slično). Prema nekim procjenama, količina podataka koja se godišnje generira, replicira i iskoristi udvostručavat će se svake dvije godine, tako da će se do 2020. godine doseći količina od 44 zetabajta (44 ZB), odnosno 44 milijardi terabajta (TB) podataka. [1] Moglo bi se reći da su podaci novo zlato. Zbog toga je važno znati jesu li podaci strukturirani ili nisu te na koji način se može upravljati njima i analizirati ih.

2.1. Strukturirani podaci

Kao što je ranije rečeno različiti podaci dolaze u različitim formatima i različitog su tipa. Na samim početcima skladištenja podataka, skladištili su se samo podaci koji imaju jasno definiranu strukturu. Razlog tome je što jasno definirana struktura podataka omogućuje lakšu analizu tih podataka. Za pojam strukturiranih podataka ne postoji jasno određena definicija. Pri tome problem stvara i granica između strukturiranih i polustrukturiranih podataka. Jednostavno rečeno, strukturirani podaci su oni podaci koji odgovaraju definiranom podatkovnom modelu, pri čemu taj model određuje kako podaci trebaju izgledati, pravila koja podaci moraju zadovoljiti i koje je sve operacije moguće izvoditi nad tim podacima.

Primjeri strukturiranih podataka uključuju brojeve, datume i grupe riječi i brojeva koji se nazivaju *stringovi* (npr. korisnikovo ime, adresa, i sl.). Iz toga slijedi da se strukturirani podaci zovu podaci koji prate definiranu formu i čije vrijednosti imaju smisleni smještaj unutar neke strukture. Tradicionalno, strukturirani podaci se skladište u relacijske baze podataka.

Prikupljanje ovakvih podataka se najčešće vrši iz tradicionalnih izvora kao što su upravljanje odnosima s kupcima (CRM) ili operativnih podataka o planiranju resursa poduzeća (ERP). Iako to može izgledati kao običan posao, prikupljanje strukturiranih podataka u stvarnosti je malo složeniji proces. Složenosti toga procesa pridonose novi izvori podataka koji su nastali razvojem tehnologije. Takvi izvori često generiraju podatke u stvarnom vremenu i u velikim količinama.

Ovdje se izvori podataka mogu podijelit u dvije kategorije. Prva kategorija bi bila računalno ili strojno generirani podaci, a druga kategorija bi bila podaci koje generira čovjek. Računalo ili strojno generirani podaci predstavljaju one podatke koje računalo ili stroj generira bez ljudske intervencije, dok bi podaci koje generira čovjek bili oni podaci koje generira čovjek u interakciji sa računalom. Izvori računalno ili strojno generiranih podataka su senzori, web logovi, financijski podaci, podaci u trenutku prodaje i sl., a izvori ljudski generiranih podataka su ulazni podaci (bilo koji tip podatka koji čovjek može unijeti u računalo – npr. ime, prezime, prihod, odgovor na ankete i sl.), klik podaci, podaci povezani s igrama na računalu (koriste se za analizu i unaprjeđenje računalnih igara). Za upravljanje strukturiranim podacima se najčešće koristi SQL jezik. [2]

2.2. Polustrukturirani podaci

Kako je ranije rečeno, strukturirani podaci imaju jasno definiranu strukturu i formu. Kod polustrukturiranih podataka slučaj je donekle različit. Ovdje ne postoji čvrsto definirana shema. Moglo bi se reći da je shema ovdje opcionalna. Neki oblici polustrukturiranih podataka uopće nemaju sheme, dok neki drugi imaju ali je ona labava i ne postoje točna ograničenja nad podacima.

Često je teško definirati granicu između strukturiranih i polustrukturiranih podataka. Kao granica između navedenih vrsta podataka se uzima činjenica da strukturirani podaci sadrže fiksna polja ili zapise, dok strukturirani podaci nemaju fiksna polja i zapise, ali imaju elemente koji mogu razdvojiti podatke u različite hijerarhije. Također, polustrukturirani podaci koegzistiraju između strukturiranih i nestrukturiranih podataka.

Primjeri polustrukturiranih podataka su transportni formati podataka, kao što su XML, JSON, CSV, TSV i sl. Isto tako, ukoliko je potrebno moguće je napraviti konverziju iz strukturiranih u polustrukturirane podatke. To se radi na način da se strukturirani podaci prilagode formi nekog od navedenih transportnih formata podataka. Kod polustrukturiranog modela podataka, podaci se spremaju u obliku ključ–vrijednost. Takav oblik pohrane podataka, polustrukturirane podatke čini pogodnim za pohranu u nerelacijske baze podataka (najčešće dokument baze podatka ili ključ–vrijednost baze podataka), za razliku od strukturiranog modela koji je pogodan za pohranu u relacijske baze podataka.

2.3. Nestrukturirani podaci

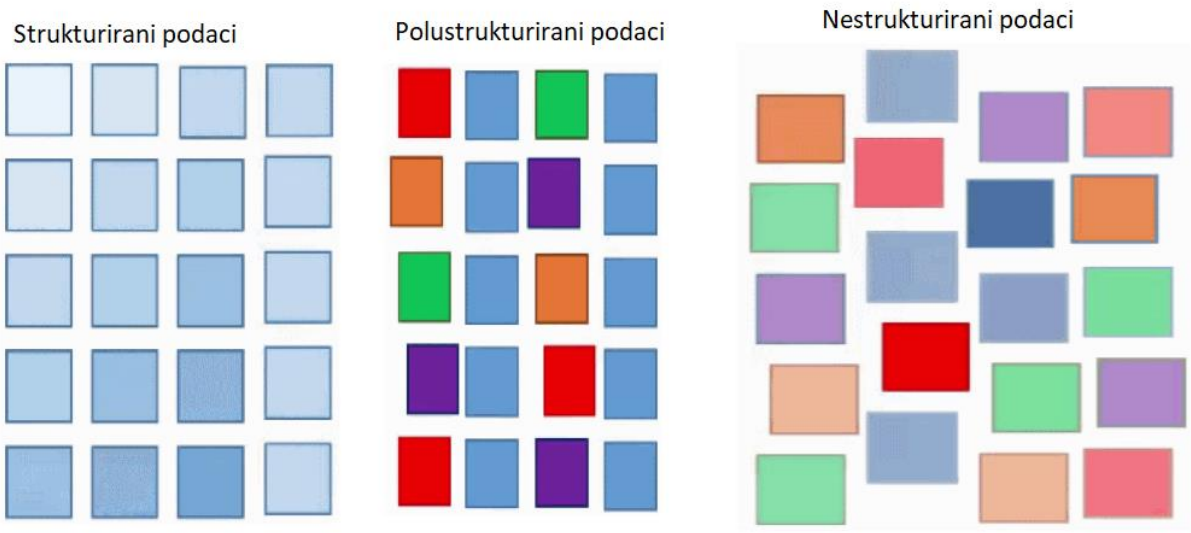
Ranije je rečeno da strukturirani podaci imaju čvrsto definiran format, a polustrukturirani imaju format koji nije čvrsto definiran. Podaci koji nemaju nikakav format se nazivaju nestrukturirani podaci. Nestrukturirani podaci su zapravo podaci koji se najčešće susreću. Donedavno tehnologija nije podržavala druge načine rada s nestrukturiranim podacima osim pohrane i ručne obrade tih podataka. Prema procjenama, više od 80% podataka u svijetu je nestrukturirano.

Neki stručnjaci su mišljenja da je pojam nestrukturiranih podataka pogrešan, jer svaki dokument može sadržavati vlastitu specifičnu strukturu ili oblikovanje u ovisnosti od aplikacije koja ga je stvorila. Međutim, ono što se nalazi unutar dokumenta je uistinu nestrukturirano. Slučajevi uporabe nestrukturiranih podatka se brzo šire. Samo za obradu teksta može se koristiti tekstualna analitika gdje se analizira nestrukturirani tekst te se potom izvlače relevantni podaci koji se zatim transformiraju u strukturirane informacije koje je moguće koristiti na različite načine.

Baš kao i ranije spomenute dvije vrste podataka i nestrukturirani podaci mogu biti generirani u realnome vremenu. Također, mogu biti generirani računalno ili strojno i podaci koje generira čovjek u interakciji sa računalom. Primjeri računalno ili strojno generiranih nestrukturiranih podataka su satelitske snimke, znanstveni podaci, fotografije i video, podaci s radara i sonara. Nestrukturirani podaci koje generira čovjek u interakciji sa računalom su podaci sa društvenih mreža, sadržaji web stranica, mobilni podaci (lokacija, SMS poruke), tekstovi unutar organizacije ili tvrtke.

Upravo zbog svega navedenoga, potrebno je pronaći način njihove pohrane. Iz svega navedenoga, može se zaključiti da relacijske baze podataka nisu dovoljne, jer podaci često nemaju nekakvu određenu strukturu te kao takvi nisu podobni za pohranu u relacijske baze podataka. Dakle, kao rješenje se nameće pohrana tih podataka u nerelacijske baze podataka. Uz nerelacijske baze podataka blisko je vezan još jedan pojam – veliki skupovi podataka (engl. *Big Data*). [2]

Na slici 2.1. prikazan je ilustrativni primjer strukturiranih, polustrukturiranih i nestrukturiranih podataka. Vidljivo je da strukturirani podaci prate određenu formu u cijelosti. Polustrukturirani podaci formu prate u određenoj mjeri, dok nestrukturirani podaci ne prate neku unaprijed određenu formu, odnosno izgledaju kao skup nasumičnih podataka.



Sl. 2.1. – Ilustrativni prikaz strukturiranih, polustrukturiranih i nestrukturiranih podataka

2.4. *Big Data* – veliki skupovi podataka

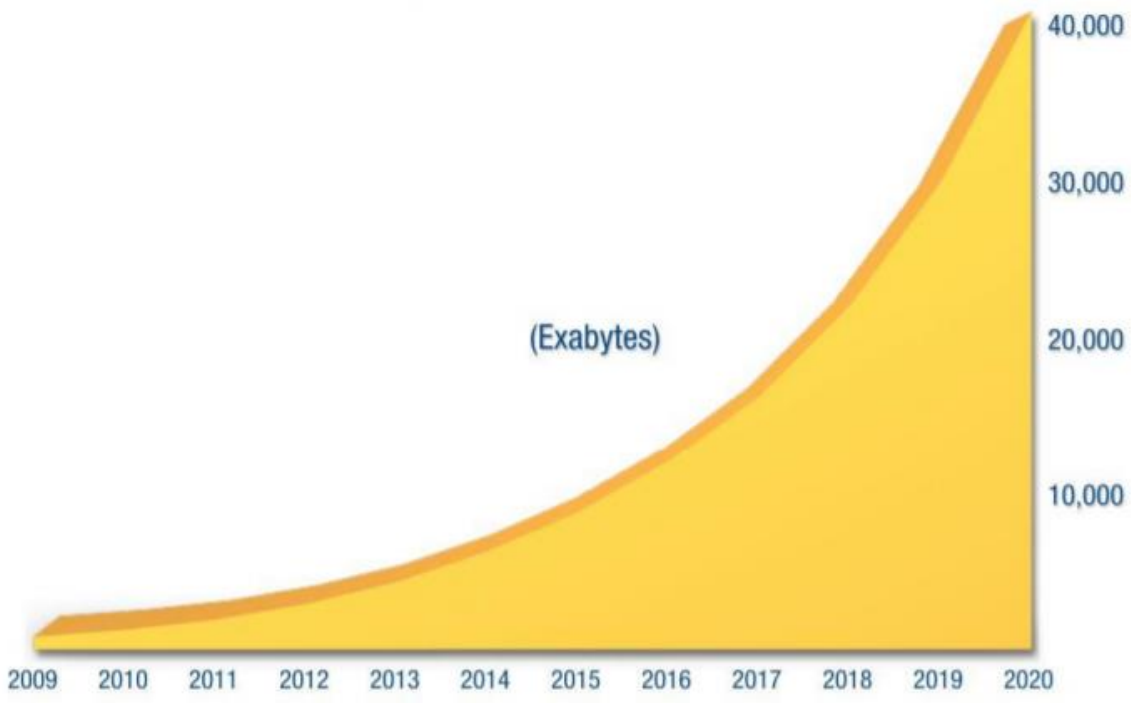
Kako je prethodno rečeno, *Big Data* pojam označava velike skupove podataka. Pod tim se misli na podatke koji pristižu velikom brzinom i u količini koja nadmašuje tradicionalne načine njihove obrade i pohrane. Ovaj pojam još uvijek nema univerzalno značenje i definiciju, a prema [3] pojam *Big Data* se definira kao proces za donošenje uvida u odlučivanje koji koristi ljude i tehnologiju kako bi se brzo analizirale velike količine različitih vrsta podataka iz različitih izvora, sa svrhom stvaranja djelotvornog znanja.

Iz gornje definicije može se uočiti da iako *Big Data* nema univerzalnu definiciju, posjeduje neke zajedničke karakteristike. Te karakteristike se u literaturi navode kao V–ovi. Različit autori navode različite V–ove, ali zajedničkih su sljedećih pet: *volume* (količina), *variety* (različitost), *velocity* (brzina), *veracity* (istinitost, vjerodostojnost), *value* (vrijednost). Spomenuti V–ovi se u literaturi nalaze i kao 3V, 4V i 5V.

2.4.1. *Volume* (količina)

Prvo svojstvo s kojim se opisuje *Big Data* je količina (engl. *volume*). Ovo svojstvo se odnosi na količinu podataka koja se generira i koja pristiže u skladišta podataka. U prošlosti, ovo je predstavljao problem, tj. prekomjerna količina podataka za skladištenje, ali s današnjim cijenama memorijskih uređaja to više ne predstavlja problem. Ono što predstavlja problem je način obrade i analize tih podataka. Količina podataka koji se generiraju raste eksponencijalno, te 90% svih dostupnih podataka proizvedeno je u zadnje dvije godine. Prema [3], 2014. godine u svijetu je bilo tri milijarde korisnika Interneta koji su koristili preko 6 milijardi uređaja (poslužitelji, osobna

računala, tableti, pametni telefoni i sl.) te je generirano oko osam exabytea podataka samo navedene godine. Naravno, određena količina tih podataka je beskorisna, ali ih je potrebno analizirati na određeni način kako bi se saznalo jesu li korisni za promatrani problem. Na slici 2.2. je prikazan porast podataka kroz vrijeme i predviđanje porasta količine podataka do 2020. godine.



Sl. 2.2. – Porast količine podataka kroz vrijeme[5]

2.4.2. *Variety* (različitost)

Drugo svojstvo važno za opis *Big Data* je različitost (engl. *variety*). Ovdje se radi o raznim izvorima podataka i raznim tipovima podataka ili točnije rečeno raznim formatima podataka. Tako podaci mogu biti podaci iz relacijske baze podataka (strukturirani), ali isto mogu biti i tekstualni dokumenti, slike, videozapisi, poruke elektroničke pošte, zvučni zapisi, podaci o financijskim transakcijama, podaci sa senzora, itd. Masovna upotreba mobilnih uređaja i „pametnih“ uređaja ostavlja svoj digitalni trag, što predstavlja podatke koji se mogu analizirati. Dugo vremena analizirali su se samo podaci koji su strukturirani. Razvoj novih tehnologija je omogućio i analiziranje nestrukturiranih i polustrukturiranih podataka. Veliki izazov je ovakav skup podataka oblikovati na smislen način kako bi se mogla izvući neka korist iz tih podataka. Prema [3], više od 95% podataka su nestrukturirani.

2.4.3. *Velocity* (brzina)

Svojstvo brzine u kontekstu *Big Data* se može razmatrati s dva aspekta – brzina kojom podaci pristižu i brzina kojom podaci moraju biti obrađeni kako bi zadovoljili neke kriterije. Idealan slučaj bi bio da se podaci obrađuju u realnome vremenu, jer u poslovnim sustavima zakašnjela reakcija može značiti propuštenu priliku i financijski gubitak. Ovo i prethodna dva svojstva definiraju 3V opis *Big Data*.

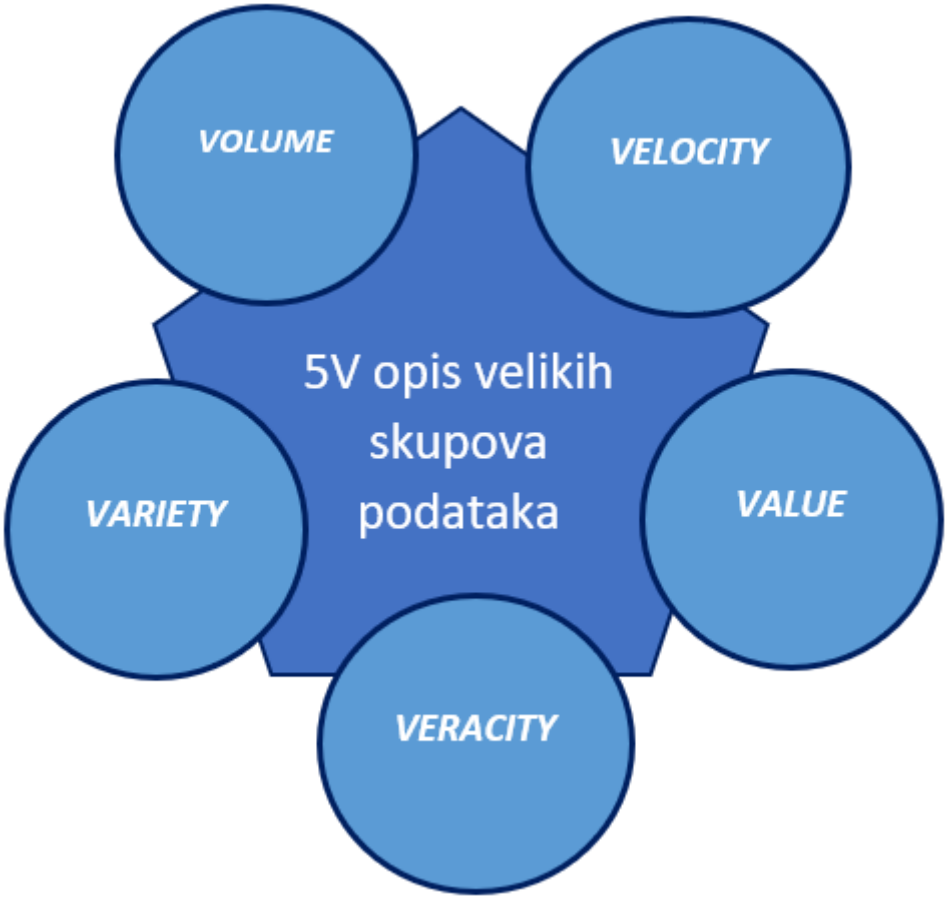
2.4.4. *Veracity* (vjerodostojnost)

Nadogradnja na 3V opis *Big Data* je 4V opis, koji čini prethodna tri svojstva i vjerodostojnost. Vjerodostojnost kod velikih skupova podataka se odnosi na kontekst, šum i abnormalnosti u podacima. Ova karakteristika odgovara na pitanje jesu li podaci koji su pohranjeni i koji se analiziraju smisleni za problem koji se proučava? Iako su sve karakteristike važne, vjerodostojnost je vjerojatno najvažnija od svih. Razlog tome je to što nitko nema koristi od analiziranja i pohranjivanja lažnih podataka, jer iz njih ne može izvući nikakvo korisno znanje, premda su podaci lažni. Kako bi se podaci održali vjerodostojnima potrebno su procesi koji vrše njihovo čišćenje i kako bi se spriječilo nakupljanje „prljavih podataka“.

2.4.5. *Value* (vrijednost)

Svojstvo vrijednost i prethodno navedena četiri svojstva čine 5V opis *Big Data*. Vrijednost kod *Big Data* se odnosi na vrijednost svakog dijela podataka. Iako je ovo zadnje svojstvo, uopće nije zanemarivo. Štoviše, može se reći da je ovo svojstvo najvažnije. Moglo bi se reći da će jedan dio podatka koji nikada ne bi imao nikakvu vrijednost biti reduciran na dio podatka koji ima vrijednost za pohranu ili procesiranje. Kada se kaže vrijednost podataka, odnosi se na vrijednost (financijsku i poslovnu) koju posjedovanje i analiza određenih podataka mogu donijeti. Dakle, riječ je o koristi koja se može ostvarit pomoću nekog skupa podataka. Pri tome, pristup velikim skupovima podataka nije dobar sve dok se ti podaci ne pretvore u određenu vrijednost, odnosno beskoristan je. [4]

Na slici 2.3. dan je ilustrativni prikaz 5V opisa *Big Data*. Svaka stavka (svako V) predstavlja jednu karakteristiku velikih skupova podataka. Unutar svake stavke navedeni su nazivi pojmova koji obilježavaju *Big Data*.



Sl. 2.3. – Ilustrativni prikaz 5V opisa velikih skupova podataka

3. RELACIJSKI MODEL BAZE PODATAKA

Kao što je ranije rečeno, trenutno su danas najzastupljenije relacijske baze podataka. Relacijske baze podataka se temelje na relacijskom modelu baze podataka. Prije nego se krene u daljnju priču o relacijskim bazama podataka, bit će dana definicija relacijske baze podataka. Važno je znati da različiti stručnjaci imaju različite definicije relacijske baze podataka, a jedna od definicija, prema [6] kaže da je baza podataka skup međusobno ovisnih podataka koji su spremljeni bez redundancije (zalihosti), pri čemu ti podaci služe jednoj ili više aplikacija na optimalan način, tako da su podaci neovisni o programu s kojim se obrađuju. Također, mora postojati kontroliran pristup do podataka.

Baza podataka je nastala kao rješenje problema održavanja istovjetnosti višestruko evidentiranih podataka i izmjena u velikom broju programa. Rješenje toga problema se odvijalo u dva koraka. Prvi korak datira iz 1965. godine kada je američki znanstvenik Charles Bachman u velikoj mjeri riješio problem fragmentacije podataka, odnosno omogućeno je formiranje integriranih datoteka koje su mogli konzumirati više odvojenih programa. C. Bachman se smatra tvorcem koncepta baze podataka, iako ovaj koncept nije imao sve odlike suvremene baze podataka.

U drugome koraku je bilo potrebno riješiti detalje organizacije podataka i manipulacije nad tim podacima. Organizacija podataka i manipulacija nad tim podacima je riješena kroz standard usvojen 1971. godine i dopunjen 1978.godine. Prema tome standardu, podaci su predstavljeni preko slogova, a integracija i uspostavljanje veza između podataka je ostvareno pokazivačkim ulančavanjem slogova u strukture kao što su stabla, liste ili mreže. Ovakve baze podataka su nazvane formatirane.

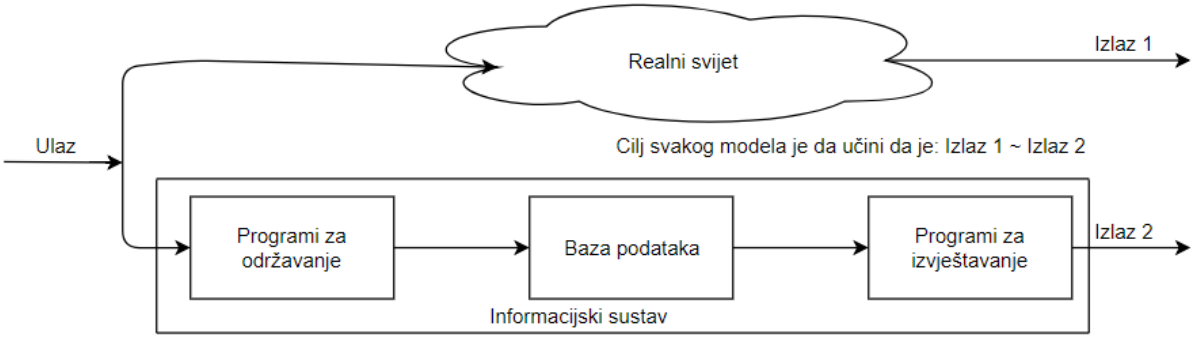
Paralelno s gore navedenim događajima, određeni broj znanstvenika je radio na ideji da se baza podataka ostvari isključivo korištenjem matematičkog koncepta relacije. Ideju u konkretnu realizaciju je pretvorio američki znanstvenik Edgar F. Codd. On je 1971. godine formulirao teorijske osnove relacijskih baza podataka, koje su pored organizacije podataka obuhvaćale i načine manipulacije nad podacima. S realizacijom prve kompletne relacijske baze podataka (System R, IBM) započeto je 1974. godine, a završena je pet godina poslije.

System R je poslužio kao prototip iz kojeg su kasnije izvedene prve komercijalne relacijske baze podatka, a na tržištu su se pojavile 1981. godine. Usporedno s time sazrijevao je i koncept standardizacije organizacije i manipulacije podataka kod relacijske baze podataka. Te aktivnosti su rezultirale nastankom standardnog SQL (engl. *Structured Query Language*) jezika za relacijske

baze podataka. On je definiran 1986. godine, a dopunjavan je 1989., 1992., 1995., 1999. i 2003. godine. [7] O samome relacijskome modelu baze podataka, kao i o SQL jeziku bit će više rečeno u nastavku.

3.1. Modeliranje relacijskog modela baze podataka

U ovome dijelu rada riječ će biti o modeliranju baze podataka. Pod pojmom modeliranja smatra se kreiranje koncepta koji pobliže opisuje procese, objekte i odnose između njih u realnome svijetu (životu). Kod modeliranja relacijskog modela baze podataka može se govoriti o konceptualnom modeliranju i o logičkom modeliranju baze podataka. Na slici 3.1. je prikazan zadatak modela i njegov odnos sa realnim svijetom. Sa slike 3.1. je vidljivo da izlaz iz modela, odnosno rezultat operacija nad modelom mora biti što je moguće više jednak rezultatu operacija u realnom svijetu.



Sl. 3.1. - Zadatak modela i njegov odnos sa realnim svijetom [8]

3.1.1. Konceptualno modeliranje baze podataka

Kada se kreće u proces modeliranja baze podatka prvo se radi koncept, odnosno konceptualni model baze podataka. Konceptualni model baze podataka opisuje strukturu podataka i predstavlja ključ razumijevanja procesa ili skupa procesa koji se žele modelirati i za koje se želi izraditi baza podataka. Ova vrsta modeliranja polazi od specifikacija informacijskih zahtjeva, koje čine zahtjevi na strukturu podataka i zahtjevi za korištenjem podataka, a rezultira izrađenim konceptualnim modelom podataka. Konceptualni model podataka je neovisan o implementaciji, kako logičkoj (sustavu za upravljanje bazom podataka) tako i fizičkoj (bazi podataka).

Dobar konceptualni model je načinjen na taj način da su podaci koji opisuju jedan objekt grupirani na jednom mjestu. Pri tome, važno ih je grupirati tako da što je više moguće budu neovisni o podatku drugog objekta. Ovakav način ima niz dobrih odlika, a neke od njih su:

- svaka konstrukcija u modelu podataka ima samo jedno značenje, koje se čita samo iz jedne konstrukcije,

- izrada modela je jednostavnija jer je u postupku modeliranja potrebno koncentrirati se samo na dijelove modela i donošenje međusobno neovisnih odluka,
- promjene u modelu podataka lako se lokaliziraju, a promjene se vrše samo na konstrukcijama koje su zahvaćene tom promjenom,
- model podataka se gradi po blokovima. [6]

Objekti iz realnog svijeta se u računalnim znanostima opisuju pomoću podataka. Zbog toga se kaže da su podaci apstrakcija realnog svijeta, odnosno sredstvo za kodiranje osobina objekata iz realnoga svijeta. Modeliranje kao postupak svođenja objekata iz realnog svijeta na određeni konačni broj podataka se sastoji iz više koraka.

Prvi korak je izbor. U ovome koraku mnoštvo objekata iz realnoga svijeta se reducira na manji skup objekata. Taj skup objekata sačinjavat će model. Objekti iz realnoga svijeta mogu biti npr. student, vozilo, artikl, proizvod i sl. Ovaj korak služi kako bi se identificirali samo neophodni objekti za promatrani problem čime se smanjuje složenost modela, a kasnije i sustava. Izbor se ne odnosi samo na objekte, nego i na njihove osobine, pa i veze između objekata.

Drugi korak u postupku modeliranja je imenovanje. Ovdje se svakom objektu iz realnog svijeta dodjeljuje ime koje ga jednoznačno opisuje. Također, i svakom svojstvu (atributu) određenog objekta se dodjeljuje ime koje ga jednoznačno opisuje. Isto tako, svaka uočena veza između objekata iz realnog svijeta dobiva ime.

Zadnji, treći korak u postupku modeliranja je klasifikacija. Kod klasifikacije se nehomogeni skup objekata i veza između njih svrstava u nehomogene klase i tipove objekata. Klasifikacija uvijek ovisi od područja primjene.

Rezultat gore navedenih koraka modeliranja je konceptualni model. Takav model sadrži sve neophodne i relevantne objekte za promatrani problem iz stvarnog svijeta. Također, sadrži i osobine tih objekata, kao i veze između pojedinih objekata. Važna stvar za ovakav model je ta da on sadrži tri neodvojive komponente – strukturu podataka, operacije nad podacima i ograničenja.

Struktura podataka i ograničenja opisuju stanje realnog sustava, odnosno predstavljaju statički opis stanja sustava. Objekti, njihova svojstva i veze između objekata i njihovih svojstava čine strukturu modela. Operacije nad podacima u modelu su zapravo operacije nad strukturom modela kojim se izražava dinamika realnog sustava. Ograničenja su pravila po kojima realni sustav funkcionira, tj. pravila koja razdvajaju dopuštena od nedopuštenih stanja realnog sustava i dio su strukture podataka, te se često ne razmatraju kao zasebna cjelina nego kao dio strukture podataka. [8]

Prethodno spomenuto može se postići korištenjem više metoda i tehnika konceptualnog modeliranja. Tehnike konceptualnog modeliranja se služe pojmovima: entitet (objekt), veza (odnos) entiteta i atribut (svojstvo ili obilježje). Kako bi se jednostavnije razumio konceptualni model, prikaz modela se vrši pomoću dijagrama. Dijagram se sastoji od više entiteta koji imaju svoja svojstva, a pri tome određeni entiteti su na neki način povezani. Takav dijagram se zove model entitet–veze, ili skraćeno E–R model, a često i E–R dijagram (skraćeno od engl. Entity–Relationship Model). E–R model promatra realni svijet kroz entitete, njihove atribute te veze između pojedinih entiteta. Može se reći da je E–R model apstrakcija realnog svijeta. [6]

3.1.1.1. Entitet

Kao što je ranije rečeno, entitet predstavlja objekt iz stvarnog svijeta. Pojam entiteta može se definirati na više načina, a prema [6] entitet se definira kao nešto što postoji i što se može identificirati u stvarnome svijetu.

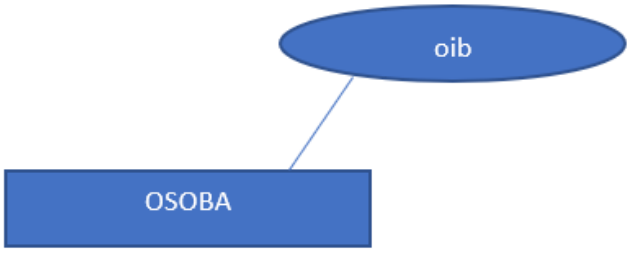
Dakle, pod pojmom entiteta se podrazumijeva sve ono što se može jednoznačno odrediti, identificirati i razlikovati. Ovako jednostavno definiran pojam entiteta pokazuje da entitet može biti bilo koji realni ili apstraktni objekt koji je interesantan za određeni problem. Svaki entitet uočen u realnom svijetu ima svoja svojstva (atribute), a vrijednosti tih svojstava omogućuju razlikovanje entiteta. Primjer entiteta su: Osoba, Vozilo, Proizvod, i sl. Prilikom imenovanja entiteta, najprikladnije je koristiti jedninu imenice koja najbolje opisuje promatrani objekt (npr. Osoba). Ukoliko je potrebno, pored imenice se može dati i riječ koja pobliže opisuje entitet (npr. Rezervni dio). Kod crtanja entiteta u E–R modelu se koristi pravokutnik, unutar kojeg se upisuje naziv tog entiteta – prikazano na slici 3.2.

Sl. 3.2. – Grafički prikaz entiteta u E–R modelu

3.1.1.2. Atribut

Kako je rečeno, entitet predstavlja objekt iz stvarnog svijeta. Ono što pobliže opisuje taj entitet su njegova svojstva ili atributi. Svaki atribut ima vrijednost. Stoga, jasno se da zaključiti da se atributi sastoje od parova naziv–vrijednost. Jedna od definicija atributa, prema [7] kaže da je to imenovana vrsta svojstva koja se ne može rastavit na dijelove bez gubitka svakog značenja.

Upravo te vrijednosti atributa omogućuju razlikovanje entiteta istog tipa. Primjeri atributa mogu biti: OIB, ime, prezime i sl. Može se primijetiti da neki atributi svake instance entiteta imaju jedinstvenu vrijednost (npr. OIB), dok neki drugi atributi mogu imati iste vrijednosti za različite instance entiteta (npr. ime). Ovdje se može govoriti o dvije vrste atributa – identifikatori i deskriptori. Identifikatori se koriste za jednoznačno identificiranje instance entiteta, a deskriptori za određivanje vrijednosti atributa koji mogu biti zajednički većem broju instanci istog entiteta. Identifikatori se još nazivaju i ključnim atributima, a deskriptori neključnim atributima. Ukoliko atribut ima svoje dodatne atribute, tada ga je bolje promatrati kao poseban entitet. Za naziv atributa se uzima riječ koja najbolje opisuje određeno svojstvo entiteta, a prilikom crtanja E–R modela atribut se označava elipsom unutar koje se piše naziv atributa te je povezana sa entitetom – prikazano na slici 3.3. [9]



Sl. 3.3. – Grafički prikaz atributa i njegova povezanost sa entitetom u E–R modelu

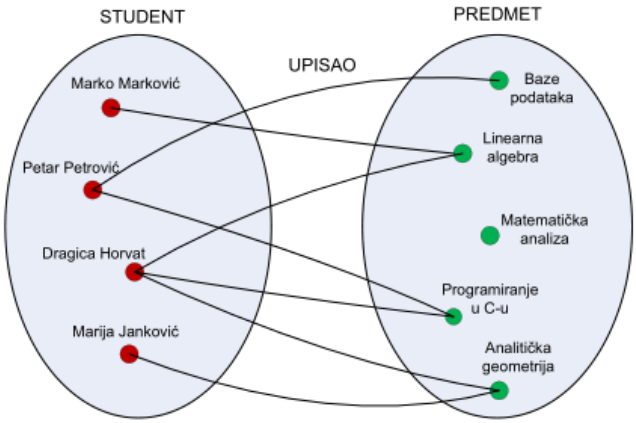
3.1.1.3. Veza

Često se objekti u stvarnome svijetu nalaze u nekakvom odnosu. Taj odnos u E–R modelu se naziva veza. Veza predstavlja spajanje dva ili više entiteta u novi entitet – vezu. Moguć je slučaj kada veza ima svoje atribute, pa se ti atributi nazivaju atributi veze. Prilikom imenovanja veze, vezi se daje ime koje pobliže opisuje odnos između entiteta koji se nalaze u toj vezi. Kod crtanja E–R modela, veza se grafički prikazuje rombom, a unutar romba se piše naziv veze – prikazano na slici 3.4.

Sl. 3.4. – Grafički prikaz veze između entiteta

U realnome svijetu, nekada su dva entiteta u određenom odnosu, a nekada više entiteta je uključeno u taj odnos. Prilikom modeliranja potrebno je i to uzeti u obzir. Veza se uvijek definira na razini

tipova entiteta, a realizira se povezivanjem pojedinih instanci entiteta određenog tipa. Broj entiteta koji su uključeni u vezu predstavlja stupanj veze. Stupanj veze može biti unarni, binarni, ternarni, ili veza višeg stupnja. Unarni (rekurzivni, involucijski) stupanj veze je poseban slučaj binarne veze u kojoj na obje strane veze sudjeluje isti tip entiteta. Najčešći stupanj veze je binarni. Binarni stupanj veze označava da se točno dva entiteta nalaze u vezi, a opisuje se kao skup uređenih parova instanci entiteta koji su povezani tom vezom. Na slici 3.5. prikazan je primjer binarne veze koja se zove UPISAO, a povezuje entitete STUDENT i PREDMET. Stanje veze prikazuje se kao skup parova, gdje svaki pojedini par označava da je dotični student upisao dotični predmet.

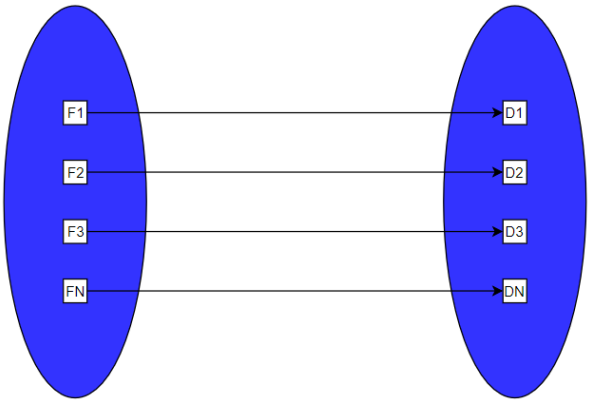


Sl. 3.5. – Stanje binarne veze i parovi povezanih instanci entiteta [10]

Odnosi između promatranih entiteta najčešće se prikazuju primjenom logike skupova i preslikavanja njihovih elemenata. Ovisno kakvo je preslikavanje u pitanju, može se govoriti o spojnosti veze. Spojnost veze opisuje ograničenja preslikavanja pojedinačnih entiteta koji sudjeluju u promatranoj vezi između entiteta E_1 i E_2. U literaturi se često za pojam spojnosti veze može naći i naziv funkcionalnost veze. Postoji više vrsta spojnosti veze, a te vrste spojnosti veze su:

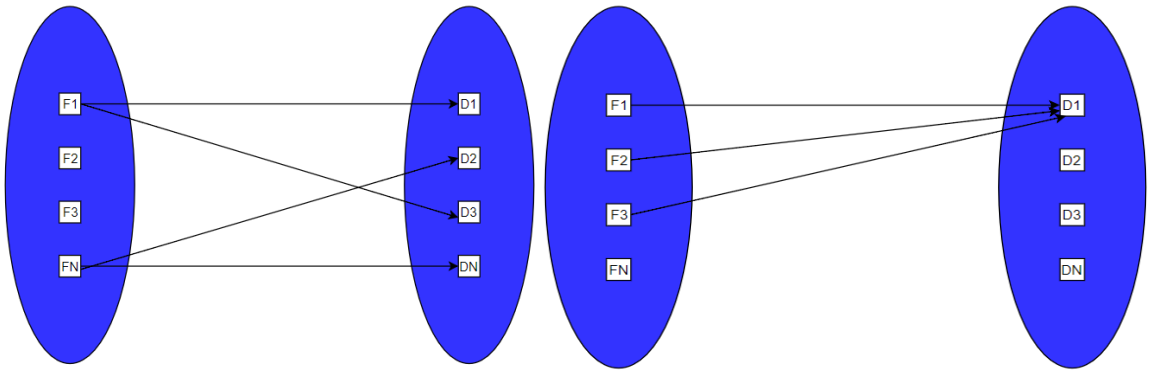
- 1:1 (jedan-prema-jedan)
- 1:M (jedan-prema-mnogo)
- M:1 (mnogo-prema-jedan)
- M:N (mnogo-prema-mnogo).

Vrsta spojnosti 1:1 označava da jedan primjerak prvog tipa entiteta može biti u vezi s najviše jednim primjerkom drugog tipa entiteta. Također jedan primjerak drugog tipa entiteta može biti u vezi s najviše jednim primjerkom prvog tipa entiteta. Preslikavanje na kojem se temelji ova vrsta spojnosti prikazano je na slici 3.6.



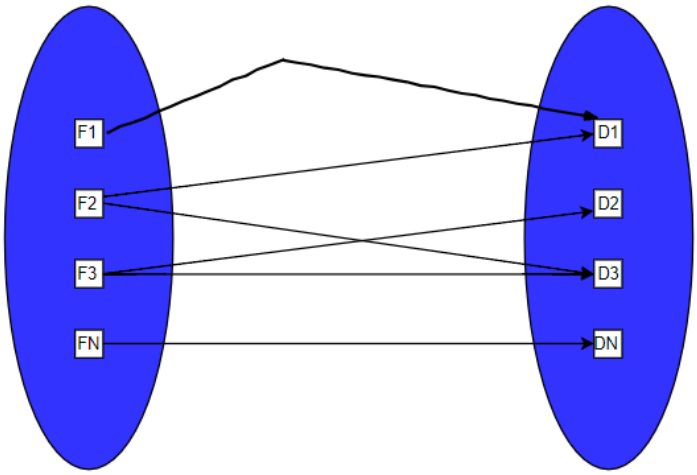
Sl. 3.6. – Spojnost veze 1:1

Spojnost veze 1:M znači da jedan primjerak prvog tipa entiteta može biti u vezi s 0, 1 ili više primjeraka drugog tipa entiteta. Istovremeno, jedan primjerak drugog tipa entiteta može biti povezan s najviše jednim primjerkom prvog tipa entiteta. Ovakva vrsta spojnosti je dosta slična spojnosti M:1. Moglo bi se reći da spomenuta dva tipa spojnosti predstavljaju istu stvar, a razlika je u tome kako se odnos između entiteta promatra. Spojnost veze 1:M i M:1 je prikazana na slici 3.7.



Sl. 3.7. – Spojnost veze 1:M (lijevo) i M:1 (desno)

Najsloženija spojnost veze je veza koja ima spojnost tipa M:N. Spojnost veze M:N znači da jedan primjerak prvog tipa entiteta može biti u vezi s 0, 1 ili više primjeraka drugog tipa entiteta. Također, jedan primjerak drugog tipa entiteta može biti u vezi s 0, 1 ili više primjeraka prvog tipa entiteta. Jedan od primjera ovakve veze bi moglo biti ako se uoče entiteti student i profesor. Jednom studentu predaje više profesora, a jedan profesor predaje za više studenata. Spojnost veze M:N prikazana je na slici 3.8. [10]



Sl. 3.8. – Spojnost veze M:N

3.1.2. Logičko modeliranje baze podataka

Jednom kada je konceptualni model gotov može se početi s logičkim modeliranjem baze podataka. U literaturi, za pojam logičkog modeliranja se može naći i pojam implementacijskog modeliranja. Logičko modeliranje rezultira izradom logičkog modela baze podataka. Ovaj model nije konačna fizička struktura podataka, pa se stoga i zove logički model. Budući da se opisuje prema pravilima određenog sustava za upravljanje bazom podataka naziva se implementacijski. Opisan je u obliku sheme baze podataka, u kojoj se koriste pojmovi relacija, redak (n–torka), primarni ključ, strani ključ, itd., ovisno o pravilima korištenog sustava za upravljanje bazom podataka. Zapravo, logičko modeliranje baze podataka se svodi na dvije stavke, a to su pretvorba konceptualnog modela podataka (izvedenog u obliku E–R modela) u logičku shemu baze podataka i doradu logičke sheme baze podataka.

Za opis logičkog modela može poslužit više modela podatka, kao što su relacijski, mrežni, hijerarhijski i drugi. U ovome radu bit će opisan relacijski model podatka, koji je ujedno i najčešće korišteni model. Osnovna kvaliteta relacijskog modela podataka je njegova jednostavnost, kako u razumijevanju, tako i u praktičnom radu. Precizno definiranje modela je omogućeno jer je nastao na matematičkim temeljima. Također, ubraja se u formalne modele, a to znači da ima dobro definirani skup koncepata za opis strukture, skup operatora za izvršenja operacija nad podacima i skup pravila integriteta podataka.

Osnovni cilj relacijskog modela je omogućiti nezavisnost podataka, dati teorijske temelje za rješenje redundancije podataka, omogućit razvoj jezika orijentiranih na skupove za obradu podataka, te dati bogat model podataka za opis i obradu jednostavnih i složenih podataka. Prva dva cilja relacijski model ostvaruje jednostavnim tabličnim prikazom i normalizacijom podataka.

Treći cilj je ostvaren upotrebom operacija relacijske algebre, koja omogućuje jednostavan rad s relacijama. Zadnji, četvrti cilj se nametnuo nešto kasnije i još nije ostvaren na zadovoljavajući način. Relacijski model opisuje isključivo logičke aspekte podataka i ne bavi se problemom fizičkog smještaja podatak u bazu podataka.

Kao što je rečeno u prethodnom odjeljku, relacijski model je temeljne koncepte preuzeo iz matematičke teorije skupova. Ti koncepti su domena, relacija i atribut. Pomoću domene definira se skup vrijednosti koji služe pri modeliranju podataka. Prema tome, relacija je skup n–torki. Uobičajeno, relacija se prikazuje dvodimenzionalnom tablicom, u kojoj jedan redak odgovara jednoj n–torki, a jedan stupac odgovara jednoj domeni. Budući da je pojam relacije preuzet iz matematike, on pretpostavlja fiksan poredak domena (stupaca tablice). Za razliku od matematike, imenovanjem stupaca u tablici, poredak tih stupaca postaje nebitan jer se taj nedostatak iz matematike otklanja kasnijom definicijom relacijske sheme baze podataka. Atribut može poprimiti jednu vrijednost iz pripadajuće domene. Ukoliko atribut ne poprimi nijednu vrijednost, dodjeljuje mu se tzv. NULL vrijednost.

Relacije, odnosno tablice u bazi podataka se opisuju relacijskom shemom. Relacijska shema se sastoji od naziva relacije i konačnog skupa atributa koji su povezani sa tom relacijom. Atribut čija vrijednost jednoznačno predstavlja jednu n–torku u relaciji se naziva primarni ključ relacije. U relacijskoj shemi, taj atribut se obično podcrta, što znači da je taj atribut primarni ključ.

Često je relacijska shema manje razumljiva od konceptualne. Razlog tome je što su entiteti i veze između entiteta pretvoreni u relacije. Ta pretvorba se odvija na način da svaki entitet iz konceptualnog modela postaje relacija u relacijskoj shemi baze podataka. Kada je riječ o atributima entiteta, ti atributi postaju atributi relacije. Pretvorba veza iz konceptualnog modela u relacijsku shemu ovisi o spojnosti veze. Veza iz konceptualnog modela će postati relacija, a samim time i tablica u bazi podataka, samo ako je spojnost veze M:N. U tome slučaju, primarni ključevi oba tipa entiteta postaju strani ključevi u relaciji koja predstavlja vezu. Ukoliko je spojnost veze 1:1, tada veza ne postaje relacija, nego primarni ključ drugog tipa entiteta ulazi u relaciju koja predstavlja prvi tip entiteta kao strani ključ. Za spojnost veze 1:M i M:1 je slična situacija kao i kod spojnosti veze 1:1. Razlika je u tome što primarni ključ entiteta koji se nalazi na strani 1, ulazi u relaciju koja predstavlja entitet sa strane M. Također, i ovdje taj primarni ključ postaje strani ključ relacije.

Relacija ima određena osnovna svojstva, od kojih prva dva proizlaze iz same definicije relacijske sheme, a druga dva iz definicije relacije, a ta svojstva su:

- relacijska shema na sadrži dva jednaka atributa, odnosno relacija ne sadrži dva jednaka stupca,
- redoslijed atributa, odnosno redoslijed stupaca je nebitan,
- relacija ne sadrži dvije jednake n–torke,
- redoslijed n–torki u relaciji je nebitan. [6]

Radi ilustracije svega gore navedenoga, bit će dan primjer koji sadrži dva tipa entiteta – student i predmet. Spomenuta dva tipa entiteta povezana su vezom "upisao“ koja ima spojnost M:N. Za potrebe ovoga primjera neka entitet Student ima sljedeće atribute: JMBAG, ime, prezime, studij, a entitet predmet neka ima atribute šifra predmeta, naziv, nositelj. Veza „upisao“ neka ima atribut datum upisa. Kao što je ranije rečeno, svaki entitet postaje relacija, a budući da je spojnost veze u ovom primjeru M:N i ta veza će postati relacija. Tako imamo sljedeće relacije koje predstavljaju relacijsku shemu baze podataka:

STUDENT(jmbag, ime, prezime, studij),

PREDMET (sifra-predmeta, naziv, nositelj),

UPISAO (datum-upisa, jmbag, sifra-predmeta).

Zadnja faza modeliranja, koja dolazi nakon izrade relacijske sheme je kreiranje fizičke sheme baze podataka. Fizička shema je zapravo tekst koji se sastoji od naredbi u nekom od jezika (npr. SQL) koji razumije sustav za upravljanje bazom podataka. Izvođenjem tih naredbi sustav za upravljanje bazom podataka stvara fizičku građu baze podataka.

3.1.2.1. Normalizacija relacijske sheme baze podataka

Nakon što se na osnovu konceptualnog modela izradio logički model, odnosno relacijska shema baze podataka moguće je pojava određenih nedostataka koje je potrebno otkloniti prije same implementacije baze podataka i njenog puštanja u rad. Česta pojava je redundancija, odnosno pojava pohranjivanja istog podatka na dva ili više mjesta. Problem kod redundantnih podataka je to što se isti podatak mora istovremeno unijeti, izmijenit ili obrisati na više mjesta. U ovakvim slučajevima se javljaju anomalije pri postavljanju upita te se mogući netočni odgovori na postavljeni upit. Postupak kojim se otklanjaju moguće anomalije naziva se normalizacija.

Normalizacija je potrebna zbog dobrog oblikovanja baze podataka. Dobro oblikovana baza podataka nije redundantna i ne pokazuje anomalije pri unosu, izmjeni, ispisu ili brisanju podataka. Normalizacija kao postupak se temelji na nekoliko formi za normiranje, točnije sedam formi za

normiranje. U praksi se koriste prve četiri forme za normiranje, a preostale tri su više od teorijskog značaja te u ovome radu neće biti razmatrane. Postupak normalizacije je moguće izbjeći ako se na početku modeliranja baze podataka dobro uoče potrebi entiteti, njihovi atributi i veze između entiteta. Ipak, dobra praksa nalaže provođenje normalizacije radi provjere i sigurnosti da je baza podataka dobro oblikovana, a pri tome je moguće otkloniti i određene nedostatke ukoliko je potrebno.

Pretvorbom konceptualnog modela u relacijsku shemu, relacija bi trebala biti najmanje u prvoj formi za normiranje (1NF). Preciznije rečeno, relacija je u 1NF ako je svaka vrijednost atributa jednostruka i nedjeljiva, odnosno ako svaki atribut sadrži samo jednu vrijednost. U svojim radovima, E. F. Codd je definirao drugu i treću formu za normiranje (2NF i 3NF), a kasnije i poboljšanu inačicu 3NF, koja se naziva Boyce–Codd-ova forma za normiranje (BCNF). 2NF je zadovoljena ako su svi neključni atributi potpuno funkcijski zavisni o bilo kojem mogućem ključu relacije. Prilikom prelaska iz 1NF u 2NF može se pojaviti tranzitivna ovisnost. Tranzitivnu ovisnost rješava 3NF, odnosno relacija je u 3NF ako nijedan neključni atribut nije tranzitivno ovisan o bilo kojem ključu relacije. U većini praktičnih slučajeva je dovoljno da je relacija u 3NF, ali opisujući ponašanje neključnih atributa ona ne rješava problem koji se javlja kod složenih primarnih ključeva. Taj problem rješava BCNF, a relacija je u BCNF ako sve funkcijske zavisnosti proizlaze iz njenog ključa. Za kraj priče o normalizaciji, važno je još reći da pravilo koje odlikuje normalizaciju je to da ako se relacija nalazi u nekoj normalnoj formi, tada zadovoljava i niže normalne forme. [6]

3.2. SQL

SQL predstavlja jezik za rad s relacijskom bazom podataka. Naziv SQL je akronim engleskog naziva ***S**tructured **Q**uery **L**anguage*, ili jednostavno prevedeno strukturirani upitni jezik. Pored SQL-a, za rad sa podacima u relacijskoj bazi podataka postoje i neki drugi jezici, primjerice QUEL i QBE. To su jezici četvrte generacije koji su zasnovani na relacijskom računu, s time da je matematička notacija zamijenjena ključnim riječima. Ključne riječi su preuzete iz govornog engleskog jezika.

SQL je razvio IBM-a u sklopu projekta System R, a nastao je iz svog prethodnika SEQUEL-a. Tvorci SQL jezika su američki znanstvenici Donald D. Chamberlin i Raymond Boyce, a prvu specifikaciju su objavili 1974. godine. Od objave prve specifikacije, jezik se tijekom vremena usavršavao. Pored IBM-a i druge kompanije koje su se bavile razvojem relacijskog modela baze podataka počelu su uključivati SQL jezik, uz male izmjene, u svoje sustave za upravljanje bazom

podataka. Rezultat toga je bio nastanak različitih „dijalekata“ SQL jezika. Zbog nastanka tih dijalekata, pojavila se potreba za standardizacijom SQL jezika, tj. izradom SQL standarda, nalik na standarde nekih programskih jezika tog vremena. [6],[11]

Sa standardizacijom, usvojeno je i načelo vertikalne kompatibilnosti, po kojemu svaki novi standard, uz poboljšanja, mora sadržavati i sve mogućnosti ranijeg standarda. Prvi standard za SQL jezik je donesen 1986. godine. On je bio rezultat kompromisa koji je već godinama vladao na tržištu. Taj standard je donio Američki nacionalni institut za standarde (ANSI, engl. **A**merican **N**ational **S**tandards **I**nstitute). ANSI standard je 1987. godine prihvatila i međunarodna organizacija za standarde (ISO, engl. **I**nternational **O**rganization for **S**tandardization), te se od tada naziva ANSI/ISO standard, koji je mijenjan tijekom vremena, a zadnja inačica standarda je objavljena 2011. godine.

Terminologija SQL jezika se nešto razlikuje od one koja je korištena u postupku modeliranja. Za pojam relacije, koristi se pojam tablice. Analogno tome, za jednu n-torku relacije kaže se red tablice, a atributi su stupci tablice. Ovakva terminologija je preuzeta iz prakse koja je prethodila standardizaciji, a rezultirala je neobičnom okolnosti da u SQL jeziku za rad sa relacijskim bazama podataka ne postoji nijedna konstrukcija koja sadrži riječ *„Relational“*. [7]

SQL jezik operacije nad podacima izvodi postavljanjem upita. Primjeri klasičnih upita koji se koriste u radu su naredbe za kreiranje baze podataka, kreiranje tablica unutar baze podataka, dodavanje podataka u tablice, izmjena postojećih podataka u tablicama, brisanje tablica ili nekih podataka iz tablice, dohvat podataka iz tablice. Također, omogućene su i neke aritmetičke i logičke operacije nad podacima, kao i definiranje „pogleda“ (virtualne tablice izvedene iz postojećih), stvaranje indeksa, te kontrola sigurnosti. Kao i drugi programski jezici, i SQL ima definirane svoje tipove podataka, aritmetičke i logičke operacije, te operacije za rad sa podacima. Tipovi podataka u SQL-u se dijele na tekstualne, numeričke i tipovi podataka za datum i vrijeme, te binarne tipove podataka. Najosnovniji i najčešće korišteni tipovi podataka u SQL-u prikazani su u tablici 3.1.

Tablica 3.1. – Prikaz najčešće korištenih SQL tipova podataka

NAZIV	OPIS	
CHAR(n)	Fiksna duljina zapisa, maksimalno 8000 znakova.	ZNAKOVNI TIPOVI
VARCHAR(n)	Varijabilna duljina zapisa, maksimalno 8000 znakova.	
TEXT	Varijabilna duljina zapisa, maksimalno 2^{31}-1 znakova.	
INT	Cijeli brojevi od -2^{31} do 2^{31}-1.	NUMERIČKI TIPOVI
BIGINT	Cijeli brojevi od -2^{63} do 2^{63}-1.	
SMALLINT	Cijeli brojevi od -2^{15} do 2^{15}.	
TINYINT	Cijeli brojevi od 0 do 255.	
DECIMAL(p,s)[1]	Decimalni brojevi od $-10^{38}+1$ do $10^{38-1.}$	
NCHAR(n)	Fiksna duljina zapisa, maksimalno 4000 znakova.	UNICODE TIPOVI
NVARCHAR(n)	Varijabilna duljina zapisa, maksimalno 4000 znakova.	
NTEXT	Maksimalno 2^{30}-1 znakova.	
DATETIME	Datum, od 01.01.1753. do 31.12.9999.	DATUM I VRIJEME
SMALLDATATIME	Datum, od 1.1.1900. do 6.6.2079, zaokruženo na minutu	
TIMESTAMP	Specijalna namjena, najčešće u milisekundama.	
BINARY(n)	Fiksna duljina binarnog zapisa, maksimalno 4000 bytes.	BINARNI TIPOVI
VARBINARY(n)	Varijabilna duljina binarnog zapisa, maksimalno 4000 bytes.	
IMAGE	Maksimalno 2^{31}-1 bytes.	

[1] p – ukupan broj znamenki (od 1 do 38), s – broj decimalnih znamenki

Tipovi podataka su važni iz razloga što svaki podatak, odnosno atribut koji se pohranjuje u tablicu mora imati definiran tip podatka. Nad tako definiranim podacima, koji imaju vrijednost i tip moguće je izvršavati naredbe u SQL jeziku. Naredbe koje pripadaju SQL jeziku moguće je podijeliti u dvije grupe. Prva grupa je jezik za definiciju podataka (DDL), a druga je jezik za manipulaciju sa podacima(DML). Pored spomenute dvije grupe, moguće je izvesti još jednu grupu naredbi koja se naziva jezik za kontrolu pristupa podacima (DCL).

DDL (engl. *Data Definition Language*) objedinjuje naredbe za definiranje resursa relacijske baze podataka. Pod resursima baze podataka se podrazumijeva:

- struktura baze podataka,
- tablice i atributi,
- tipovi podataka,
- ograničenja,
- pomoćni indeksi za direktan pristup,
- pogledi.

Efikasan sustav za upravljanje bazom podataka trebao bi omogući izvršavanje naredbi koje su prikazane u tablici 3.2. Uz naziv naredbe, dan je i kratak opis šta dotična naredba radi.

Tablica 3.2. – Popis naredbi DDL grupe naredbi

NAZIV NAREDBE	OPIS NAREDBE
CREATE DATABASE	Kreiranje baze podataka.
CREATE TABLE	Kreiranje nove tablice u bazi podataka.
NOT NULL	Osigurava da stupac ne može bit prazan.
UNIQUE	Osigurava da stupac ne sadrži duplirane vrijednosti.
PRIMARY KEY	Definira primarni ključ za tablicu.
FOREIGN KEY	Definira vanjski ključ za tablicu.
DEFAULT	Definira zadanu vrijednost za stupac.
CHECK	Potvrđuje podatak u atributu.

CREATE INDEKS	Kreira indeks za tablicu.
CREATE VIEW	Kreira dinamički podskup redova/stupaca iz jedne ili više tablica.
ALTER TABLE	Izmjena definicije tablice
DROP TABLE	Trajno briše tablicu (i podatke iz nje) iz baze podataka.
DROP INDEKS	Trajno briše indeks.
DROP VIEW	Trajno briše pogled.

Pod pojmom DML podrazumijevaju se SQL naredbe koje služe za manipulaciju nad podacima koji su pohranjeni u bazi podataka. Osnovne funkcionalnosti koje se omogućene DML jezikom su unos, dohvaćanje, izmjena i brisanje podataka. Pored samih naredni, DML obuhvaća i određene klauzule, operatore i agregatne funkcije. DML grupa naredbi je prikazana u tablici 3.3.

Tablica 3.3. – Popis naredbi DML grupe naredbi

NAZIV	OPIS NAREDBE
INSERT	Dodaje red(ove) u tablicu.
SELECT	Dohvaća atribute iz redova u jednoj ili više tablica ili pogleda.
WHERE	Ograničava selekciju redova na osnovu zadanog izraza.
GROUP BY	Grupira selektirane redove na osnovu jednog ili više atributa.
HAVING	Ograničava selekciju grupiranih redova na osnovu uvjeta.
ORDER BY	Reda selektirane redove na osnovu jednog ili više atributa.
UPDATE	Izmjenjuje vrijednost atributa u jednom ili više redova tablice.
DELETE	Briše jedan ili više redova iz tablice.
COMMIT	Trajno čuva promjene nad podacima.

ROLLBACK	Vraća podatke u početno stanje.
OPERATORI USPOREDBE	
=, <,>, <=,=>,<>	Koriste se u uvjetnim izrazima.
LOGIČKI OPERATORI	
AND/OR/NOT	Koriste se u uvjetnim izrazima.
SPECIJALNI OPERATORI	
BETWEEN	Provjerava je li vrijednost atributa u danom okviru.
IS NULL	Provjerava je li vrijednost atributa ostavljena prazna.
LIKE	Provjerava slaže li se vrijednost atributa sa unesenim *stringom*.
IN	Provjerava poklapa li se vrijednost atributa sa bilo kojom vrijednosti iz liste vrijednosti.
EXIST	Provjerava vraća li podupit kao rezultat neki od redova.
DISTINCT	Ograničava vrijednosti jedinstvenih vrijednosti.
AGREGATNE FUNKCIJE	
COUNT	Za dani stupac kao rezultat vraća broj redova koji nisu prazni.
MIN	Za dani stupac kao rezultat vraća najmanju vrijednost pronađenu u njoj.
MAX	Za dani stupac kao rezultat vraća najveću vrijednost pronađenu u njoj.
SUM	Za dani stupac kao rezultat vraća zbroj svih vrijednosti iz tog stupca.
AVG	Za dani stupac kao rezultat vraća prosječnu vrijednost svih vrijednosti iz tog stupca.

Kako je ranije rečeno, pored DDL i DML grupe naredbi, može se izvesti i DCL grupa naredbi. DCL (engl. *Data Control Language*) prije svega predstavlja naredbe za kontrolu pristupa podacima. Kontrola pristupa podacima podrazumijeva kreiranje više korisničkih profila s

određenim podacima za pristup (korisničko ime, zaporka) i pridruženim privilegijama. Privilegije se u ovom kontekstu odnose na dopuštene i moguće radnje koje može izvoditi pojedini korisnik nad podacima u bazi podataka. Kod DCL grupe naredbi razlikuju se naredbe za rad sa korisničkim profilima i naredbe za rad sa korisničkim privilegijama. Popis tih naredbi dan je u tablici 3.4. [8]

Tablica 3.4. – Popis naredbi DCL grupe naredbi

NAZIV NAREDBE	OPIS NAREDBE
NAREDBE ZA RAD SA KORISNIČKIM PROFILIMA	
CREATE USER	Kreira korisnički profil u bazi podataka.
RENAME USER	Izmjena postojećeg korisničkog profila.
DROP USER	Uklanjanje postojećeg korisničkog profila.
SET PASSWORD	Promjena postojeće pristupne zaporke za korisnički profil.
NAREDBE ZA RAD SA KORISNIČKIM PRIVILEGIJAMA	
GRANT	Dodjela privilegija određenom korisničkom profilu.
REVOKE	Oduzimanje privilegija određenom korisničkom profilu.
FLUSH PRIVILEGES	Ponovno učitavanje parametara sustava koji se nalaze u privremenoj memoriji.

3.3. Sustav za upravljanje bazom podataka

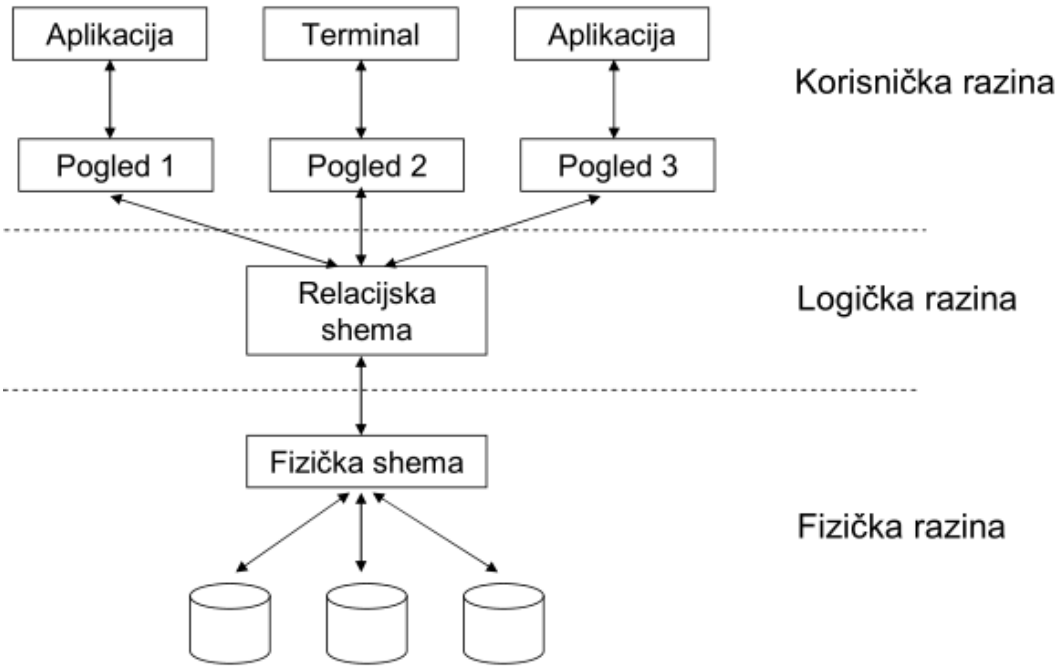
Sustav za upravljanje bazom podataka (SUBP, engl. DBMS) je ključni dio svake baze podataka. SUBP se odnosi na tehnologiju pohrane i dohvata korisničkih podataka uz najveću učinkovitost i odgovarajuće mjere sigurnosti. Tradicionalno, podaci su bili organizirani u datotečne sustave. Pojavom relacijskog modela baze podataka, SUBP je bio novi koncept u pohrani i organizaciji podataka, a istraživanja su rađena kako bi se prevladao nedostatak u tradicionalnom stilu upravljanja podacima. Prema [12], SUBP se definira kao skup programa za definiranje, administriranje i procesiranje podataka. Važno za znati je da SUBP organizira podatke u strukturu, te za podatke koji su pohranjeni u bazi podataka se može reći da su strukturirani. Više o strukturiranim podacima je rečeno u prethodnome poglavlju.

Korisnik ili korisnički program postavlja zahtjev za obavljanje neke operacije nad podacima, a SUBP ga analizira, provjerava, transformira u niz operacija koje je potrebno obaviti na fizičkoj razini, obavlja operacije te vraća rezultat. Također, SUBP pruža nezavisnost podataka. Nezavisnost podataka može biti fizička i logička. Fizička nezavisnost podataka se postiže tako što pri promijeni unutarnje (fizičke) sheme baze podataka nije potrebno mijenjati aplikacijske programe. Logička nezavisnost podataka se ogleda u tome da pri promijeni konceptualne sheme nije potrebno mijenjati aplikacijske programe. U praksi, teže se postiže logička nezavisnost. SUBP je kreiran zbog mogućnosti izvršavanja određenih zadataka, a ti zadaci koje SUBP ima su:

- zaštita baze podataka od neovlaštenog korištenja (engl. *data security*),
- sprječavanje narušavanja pravila integriteta (engl. *data integrity*),
- osigurati obnovu podataka u slučaju uništenja (engl. *recovery*),
- spriječiti štetne interferencije korisnika u višekorisničkim sustavima (engl. *concurrency*),
- omogućiti korištenje rječnika podataka (podaci o podacima),
- optimizirati sve funkcije i obavljati ih efikasno što je više moguće.

Dizajn SUBP uvelike ovisi o njegovoj arhitekturi. Arhitektura SUBP može biti centralizirana, raspodijeljena ili hijerarhijska. Također, arhitektura SUBP može biti jednoslojna ili višeslojna. Kod višeslojne arhitekture, cijeli sustav je podijeljen u više povezanih modula, koji se mogu samostalno modificirati, uređivati, mijenjati. U jednoslojnoj arhitekturi, SUBP je jedini modul koji direktno pristupa SUBP i koristi ga. Sve promjene učinjene ovdje, biti će direktno učinjene na samome SUBP. Ovakva arhitektura ne nudi alate za krajnje korisnike. Ukoliko je u pitanju dvoslojna arhitektura SUBP, tada je potrebno imati aplikaciju putem kojem se pristupa SUBP. U ovakvoj arhitekturi, aplikacijski sloj je potpuno neovisan o bazi podataka u smislu rada, dizajna ili programiranja.

Najčešće korištena arhitektura za dizajn SUBP je troslojna arhitektura. Ova arhitektura SUBP se sastoji od tri sloja. Slojevi su odvojeni na temelju složenosti korisnika i kako oni koriste podatke pohranjene u bazi podataka. Prikaz troslojne arhitekture dan je na slici 3.9.



Sl. 3.9. – Troslojna arhitektura sustava za upravljanje relacijskom bazom podataka

Sa slike 3.9. vidljivo je da se kod troslojne arhitekture radi o tri razine apstrakcije. Prva razina je fizička razina i ona predstavlja fizički prikaz i raspored podataka na jedinicama za pohranjivanje. Logička (konceptualna) razina predstavlja logički opis (shemu) cijele baze podataka. Treća, najgornja razina u troslojnoj arhitekturi SUBP je korisnička razina. U literaturi, ovaj sloj se naziva i prezentacijski sloj. Korisnički sloj omogućuje individualni korisnički pogled na podatke. Interakcija između korisnika i SUBP se odvija na ovoj razini.

Baza podataka najbolje svoju primjenu ima ako se radi u višekorisničkom okruženju. U ovakvim slučajevima veoma bitan čimbenik je SUBP. Jedna od osnovnih zadaća SUBP je sprječavanje štetnih posljedica pri transakcijama koje se odvijaju u bazi podataka. Pod pojmom transakcije se podrazumijevaju sve promjene koje se događaju nad podacima u bazi podataka, te se transakcije odvijaju u cijelosti do kraja ili se poništavaju u cijelosti. Svaka transakcija koja se odvija u bazi podataka treba zadovoljiti svojstva poznata kao ACID svojstva, a SUBP se brine da ta svojstva budu uistinu zadovoljena. U ACID svojstva spadaju: atomnost (valentnost), konzistentnost, izolacija i trajnost.

Atomnost (engl. *atomicity*) podrazumijeva skup aktivnosti nad bazom podataka pod principom „sve ili ništa“. To znači sljedeće – ili su sve aktivnosti uspješno obavljene ili nijedna aktivnost nije izvršena. Pored atomnosti transakcije, SUBP mora osigurati i atomnost svake pojedinačne operacije.

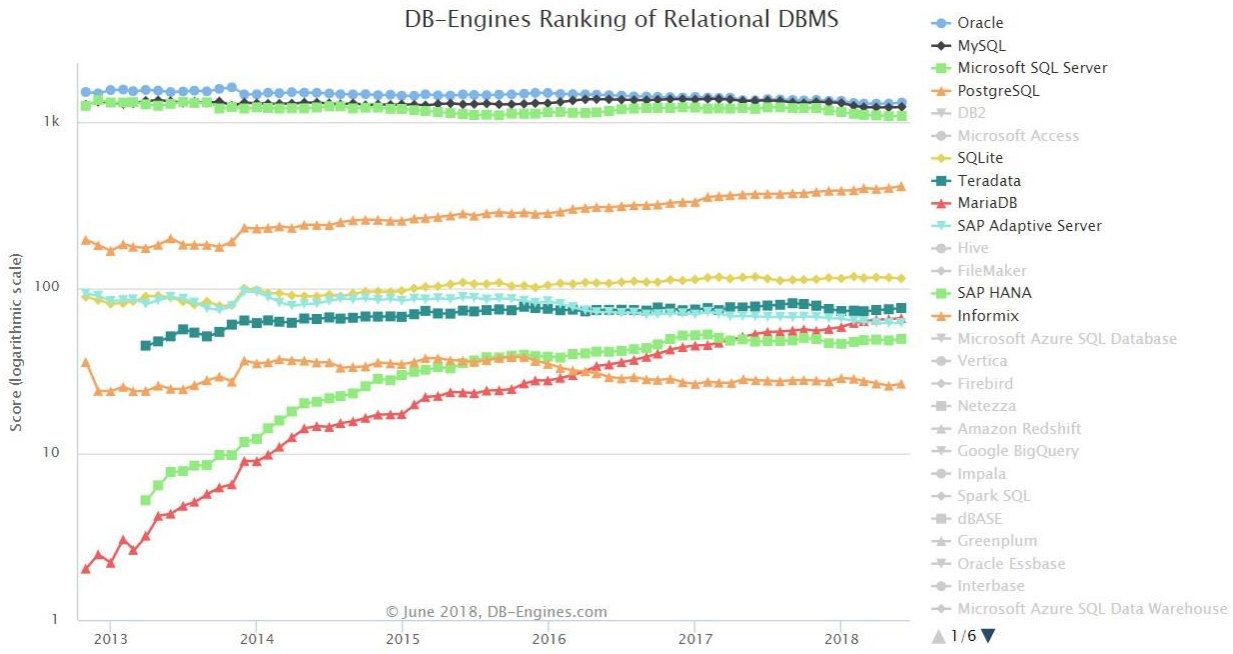
Konzistentnost (engl. *consistency*) znači da transakcija treba prevesti bazu podataka iz jednog u drugo stabilno stanje. Ukoliko tijekom izvođenja transakcije dođe do pojave greške, podaci trebaju biti vraćeni u stanje prije početka izvođenja te transakcije. Konzistentnost baze podataka je upravo najviše narušena tijekom izvođenja transakcije.

Izolacija (engl. *isolation*) predstavlja međusobnu izoliranost transakcija. To znači da kada se više transakcija izvodi u istom trenutku njihovi efekti trebaju biti međusobno izolirani. Zbog povećanja paralelizma u obradi transakcija postoje različite razine izoliranosti. Izoliranost transakcije dovodi do toga da izgleda kao da se samo jedna transakcija izvodi u tom trenutku.

Trajnost (engl. *durability*) znači da kada se transakcija završi i potvrde se promijene, efekti te transakcije ne mogu biti izgubljeni, čak i ako se neposredno po njenom okončanju dogodi neki ozbiljan otkaz sustava. Drugim riječima, svojstvo trajnosti osigurava da podaci neće biti izgubljeni. SUBP ovdje ima zadaću osigurati prijenos efekta potvrđene transakcije na bazu podataka, čak i ako je medij na kojem je pohranjena baza podataka prestao s radom.

Osiguranje ACID svojstava pomoću SQL-a se postiže upotrebom ključnih riječi *BEGIN TRANSACTION, COMMIT* i *ROLLBACK*. *BEGIN TRANSACTION* označava početak transakcije. Ključna riječ *COMMIT* označava završetak i potvrdu transakcije, dok *ROLLBACK* označava završetak transakcije kada se sve naredbe nisu uspješno izvršile. [8]

Danas postoji više sustava za upravljanje relacijskom bazom podataka, a koji će biti korišten u konkretnom informacijskom sustavu ili aplikaciji ovisi o parametrima poput raspoloživog budžeta, prirodi aplikacije pa čak i o preferencijama i navikama razvojnog tima. Internet stranica https://db-engines.com/en/ vodi evidenciju o korištenju pojedinih sustava za upravljanje bazom podataka prema njihovoj popularnosti, a podaci se osvježavaju na mjesečnoj bazi. Slika 3.10. grafički prikazuje poredak korištenja pojedinih sustava za upravljanje relacijskim bazama podataka prema popularnosti u lipnju 2018. godine. [23]



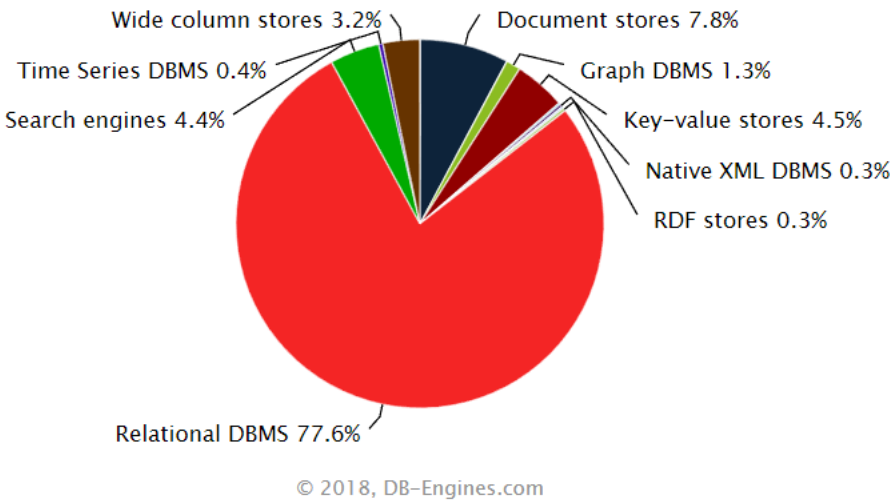
Sl. 3.10. – Popularnost pojedinih sustava za upravljanje relacijskom bazom podataka u lipnju 2018. godine [23]

4. NERELACIJSKI MODEL BAZE PODATAKA

U ovome poglavlju bit će opisan nerelacijski model baze podataka. Dok su relacijske baze podataka prema svojoj definiciji i načinu organiziranja podataka podobne za pohranu strukturiranih podataka, pojavom nerelacijskih baza podataka omogućena je pohrana i polustrukturiranih i nestrukturiranih podataka. Razlozi zašto je važno pohranjivati i takve podatke opisani su u poglavlju „Vrste podataka“. Također, NoSQL baze podataka u pravilu mogu pohraniti puno više podataka nego relacijske što ih čini pogodnim za „podršku“ velikih skupova podataka.

Prema [13] NoSQL kao pojam u računarstvu se definira kao široka klasa sustava za upravljanje bazom podataka poznatih po svojim nepridržavanjima široko korištenih relacijskih sustava za upravljanje bazom podataka, odnosno NoSQL baze podataka nisu primarno izgrađene od tablica i kao takve ne koriste SQL jezik za manipulaciju nad podacima.

Nerelacijske baze podataka se često skraćeno naziva NoSQL (engl. *Not Only SQL*) baze podataka. NoSQL je naziv koji obuhvaća nekoliko različitih tehnologija pri radu s bazama podataka. Takve baze podataka su se pojavile kao alternativa relacijskim bazama podataka, u kojima se upiti izvode prvenstveno pomoću SQL naredbi. Na tržištu su relacijske baze podataka još uvijek dominantne, a porast popularnosti NoSQL baze podataka su doživjele razvojem tehnologije koji je otkrivao manjkavosti u radu sa relacijskim bazama podataka. Kod nerelacijskih baza podataka bitno svojstvo je to da se u njih mogu pohranjivati i strukturirani podaci, samo što se ovdje podaci ne spremaju u tablice nego u neke druge objekte. U koje objekte će biti spremljen neki skup podataka ovisi o vrsti NoSQL baze podataka. Ovakav koncept pohrane podataka s vremenom se pokazao vrlo efikasnim i korisnim, a osim toga kao prednost NoSQL baza podataka se mogu istaći i dostupnost (*open source*), distribuiranost, visoke performanse, tolerancija na greške i skalabilnost. Iako su danas još uvijek relacijske baze podataka najzastupljenije na tržištu, zadnjih godina raste i popularnost NoSQL baza podataka. Udio pojedinih baza podataka, prema modelu i kategoriji u lipnju 2018. godine je prikazan na slici 4.1.



Sl. 4.1. – Udio pojedinih kategorija sustava za upravljanje bazom podataka u lipnju 2018. godine [23]

NoSQL baze podataka prednost nad relacijskim bazama podataka ostvaruju na temelju modela podataka, skalabilnosti, replikacije podataka i mogućnosti oporavka baze podataka. Prednost koju ostvaruje nerelacijski model baze podataka nad relacijskim u velikoj mjeri duguje modelu podataka. Definiranje modela podataka kod relacijskog modela je vrlo zahtijevan i nepredvidiv posao, budući da se on temelji na korisničkim zahtjevima. Promjena korisničkih zahtjeva ovaj posao čini duljim i složeniji. NoSQL baze podataka imaju dosta fleksibilniji model podataka, a za razliku od relacijskih baza podataka, podaci se ne spremaju u tablice, nego u različite objekte.

Jedna od najbitnijih prednosti NoSQL baza podataka u odnosu na relacijske baze podataka je skalabilnost. Jednostavno rečeno, skalabilnost bi bilo svojstvo sustava koje predstavlja mogućnost povećanja bez izmjene osnovnih osobina ili funkcija. Budući da se broj korisnika, pa samim time i količina podataka svakim danom sve više i više povećava, važno je voditi brigu o načinu i kapacitetima za pohranu tih podataka. Za tu svrhu razvijena su dva načina skaliranja – vertikalno i horizontalno skaliranje. Vertikalno skaliranje se može definirati kao dodavanje resursa serveru na kojem su pohranjeni podaci. Horizontalno skaliranje predstavlja dodavanje novih servera i pohranu podataka na više različitih servera. Vertikalna skalabilnost je podržana i kod relacijskih baza podataka i zasnovana je na centraliziranom pristupu, što sustav čini nesigurnijim u slučaju kvara. NoSQL baze podataka omogućuju horizontalno skaliranje. Povećanjem podataka koje treba pohraniti, jednostavno se u sustav dodaje novi server. Na ovaj način, ako dođe do kvara jednog servera podacima se i dalje može pristupiti koristeći druge servere unutar tog skupa servera.

Također, kao i kod relacijskih baza podataka, i kod nerelacijskih baza podataka replikacija se zasniva na principu *master–slave*. To znači da se podaci najprije pohranjuju na glavnom serveru

(*master*), a zatim se repliciraju na „podređene“ servere (*slave*). Podaci se mogu replicirati istodobno, tj. u trenutku pohrane na *master*, pohranjuju se i na sve *slave* servere. Ovakav način se naziva sinkrona replikacija, dok način gdje se podaci prvo pohranjuju na *master*, potom se pohranjuju u red, te nakon nekog vremena repliciraju na *slave* servere se naziva asinkrona replikacija.

Kolikogod suvremeni serveri bili pouzdani, uvijek se mora očekivati da će doći do kvara servera. Kvar servera može značiti gubitak podataka koji su pohranjeni na tome serveru. Zbog tog razloga potrebno je izraditi strategiju oporavka baze podatka u slučaju kvara. Ovo je kod NoSQL baza podataka znatno olakšano, budući da su podaci pohranjeni na više fizičkih servera koji se nalaze u hijerarhiji. Nakon što se kvar dogodi, sustav za upravljanje bazom podataka pregledava zapisnik aktivnih transakcija, te ponovo pokreće one transakcije koje su se izvodile neposredno prije pojave kvara. Konzistentnost podataka je osigurana na način da se isti postupak odvija na nekoliko servera.

Ranije je spomenuto da se podaci kod nerelacijskih baza podataka ne pohranjuju u tablice nego u neke druge objekte. Ti objekti mogu biti dokumenti, parovi ključ–vrijednost, grafovi i stupci. Na osnovu tipa objekta u koji se pohranjuju podaci, razlikuje se više vrsta nerelacijskih baza podataka. Vrste nerelacijskih baza podataka i njihova svojstva biti će opisani u nastavku rada, a poseban naglasak biti će stavljen na graf baze podataka. [14]

4.1. Vrste NoSQL baza podataka

Budući da NoSQL baze podataka nemaju striktno definiranu shemu i model podataka, odnosno imaju više njih, s vremenom su se pojavili razni tipovi ili vrste NoSQL baza podataka. Svaka od tih vrsta pogodna je za neke određene slučajeve. Ono što je zajedničko svim NoSQL bazama podataka je to da ne slijede relacijski model. Kada se priča o NoSQL bazama podataka, obično se podrazumijeva skupina sljedeće četiri vrste baza podataka – baze podataka temeljene na parovima ključ–vrijednost, dokument baze podataka, stupčaste baze podataka i graf baze podataka.

4.1.1. Ključ–vrijednost baze podataka

Model ključ–vrijednost baze podataka (engl. *Key–value*) je jedan od najjednostavnijih modela nerelacijskih baza podataka. Kako i samo ime kaže, upravljanje i pohrana podataka kod ovog modela se temelji na identifikatorima koji se nazivaju ključevi. Osim za identificiranje, ključevi služe i za indeksiranje. Ustvari, model ključ–vrijednost pohrane podataka je složenija varijanta pohrane podataka u asocijativno polje. Asocijativno polje predstavlja apstraktni tip podatka sličan

tipu polje, ali posjeduje ublažene zahtjeve na tipove vrijednosti koji se pohranjuju. Ovdje je moguće pohraniti vrijednosti različitih tipova unutar istog asocijativnog polja. Iz toga slijedi da je asocijativno polje kolekcija podataka tipa ključ–vrijednost, gdje se svaki ključ pojavljuje najviše jednom. Ključ–vrijednost model pohrane podataka poznat je po odličnim performansama i mogućnosti skaliranja.

S obzirom na to koji način pohrane podataka koriste, ključ–vrijednost baze podataka moguće je podijeliti na dva tipa – tip pohrane koji pohranjuje podatke samo u memoriji računala i tip koji dugoročno čuva podatke na trajnim spremištima podataka. Prvi tip pohrane karakterizira to da se podaci gube gašenjem računala, a koristi se u slučajevima kada je korisniku potreban brzi pristup podacima. Kod drugog tipa pohrane, podaci ostaju trajno pohranjeni na jedinicama za pohranu podataka, a prednost mu je brzina pristupa podacima korištenjem memorije.

Iz svega gore navedenoga, vidljivo je da ključ–vrijednost baze podataka zahtijevaju minimalni skup zahtjeva za uređenje i raspored podataka. Glavni zahtjev koji mora zadovoljit je to da svaka vrijednost unutar istog imenika mora imati jedinstveni ključ. Pod pojmom imenika se podrazumijeva skup parova ključ–vrijednost koji su logički povezani. Također, vrijednosti koje se pohranjuju ne moraju nužno biti istog tipa i taj tip podatka baza podataka ne mora znati unaprijed. Još jedno svojstvo koje krasi ovu vrstu NoSQL baze podataka je jednostavnost operacija. Operacije koje su dozvoljene nad pohranjenim podacima su postavljanje ključa, dohvat ključa i brisanje ključa iz baze podataka.

Kao i kod relacijskih baza podataka, i ovdje postoji više implementacija. Implementacija označava zajednički naziv za sve sustave za upravljanje ključ–vrijednost bazom podataka. Sve sustave za upravljanje ključ–vrijednost bazama podataka karakterizira jednostavnost, brzina i skalabilnost.

Jednostavnost se odlikuje u tome da je model podataka vrlo jednostavan, a za rad sa podacima nije potrebna prethodna definicija sheme baze podataka, kao ni definicija tipa podatka. Uz to, pojavi li se potreba za novim atributima unutar objekta, baza podataka ne mora znati ništa o tome. Ovdje je dovoljno samo uključiti te atribute u programski kod. Jednostavan model pohrane podataka također osigurava i veću brzinu izvođenja pojedinih naredbi u odnosu na druge baze podataka. Veća brzina je omogućena zahvaljujući ranije spomenutim tipovima pohrane. Također, skalabilnost je temeljena na repliciranju podataka.

Ključ–vrijednost baze podataka se koriste za pohranu podataka o korisničkim profilima, informacija o sesiji i statusnih poruka. Također, koriste se kod web trgovina i telekom imenika. Najpoznatiji predstavnici ove vrste NoSQL baza podataka su Riak, Redis i Voldemort. [15]

4.1.2. Dokument baze podataka

Kod ključ–vrijednost baza podataka, za svaki ključ se vraća vrijednost vezana uz taj ključ. Dokument baze podataka funkcioniraju na malo drugačiji način. Naime, dokument baza podataka svoju upotrebu nalazi u slučajevima kada je potrebna fleksibilnost ključ–vrijednost baze podataka. Takvi slučajevi uglavnom sadrže upravljanje složenijim podatkovnim strukturama, a važna stavka je i indeksiranje podataka te njihovo pretraživanje.

Dokument baze podataka imaju jedinstveni identifikator, kao i kod ključ–vrijednost baza podataka, ali ovdje ga nije potrebno koristi za operacije nad podacima unutar baze. Zapravo, dokument baze podataka predstavljaju složeniji model ključ–vrijednost baza podataka. Ta složenost kod ovog modela proizlazi iz činjenice da se vrijednosti spremaju u dokument koji se može pretraživati.

Riječ dokument, u nazivu ovoga modela ne upućuje na neki poseban format dokumenta, odnosno u dokument baze podataka moguće je pohraniti dokumente kreirane u nekom od alata za obradu teksta, HTML dokumente i sl. Općenito rečeno, dokument koji je pohranjen u dokument bazu podataka predstavlja samo–opisujuću stablastu strukturu, kao što su XML ili JSON formati. Pojam samo–opisujuća struktura znači da unutar takvih struktura postoje oznake koje razdvajaju elemente i prema njima je složena hijerarhija unutar tog dokumenta. Ranije u radu je rečeno da XML i JSON formati zapisa podataka spadaju u polustrukturirane podatke, pa se iz te činjenice lako da zaključit da su dokument baze podataka pogodne za pohranu polustrukturiranih podataka.

Dodavanjem novog dokumenta u dokument bazu podataka, sustav za upravljanje bazom podataka automatski će indeksirati taj dokument. To dalje znači da se svi dokumenti unutar baze mogu pretražiti. Baza podataka vrlo lako da odgovor koji dokumenti sadrže traženo svojstvo koje je interesantno u nekom trenutku. Također, pored dokumenata koji sadrže to svojstvo kao odgovor dobije se i lokacija tog svojstva unutar pojedinog dokumenta. Lokaciju toga svojstva unutar dokumenta moguće je dobit zahvaljujući specijalnom ključu koji se naziva put po dokumentu. Na osnovu toga specijalnog ključa vrši se obilazak stablaste strukture.

Kao i kod ostalih NoSQL baza podataka, ni ovdje shema podataka nije strogo definirana. Pohranu podataka je moguće izvršiti bez znanja baze podataka o strukturi, tipu ili značenju tih podataka.

Izmjena sheme podataka koja se radi kod relacijskih baza podataka pri promjeni strukture ovdje je nepotrebna. Također, moguće je naknadno promijeniti strukturu podataka čak i ako je sustav već u produkciji. Iako dokument baze podataka nemaju strogo definiranu shemu podataka, one se temelje na polimorfnoj shemi koja predstavlja organizacijsku shemu po kojoj je dopušteno dodavanje različitih tipova dokumenata unutar jedne kolekcije. Kolekcija dokumenata se može predočit kao lista međusobno sličnih dokumenata.

Operacije koje je moguće izvoditi u dokument bazama podataka su dodavanje, brisanje, ažuriranje zapisa, te dohvat podataka. Budući da postoji više implementacija dokument baza podataka, ne postoji standardni jezik za manipulaciju nad podacima. Ipak, princip i koncept svakog jezika je isti, neovisno o implementaciji. Također, skalabilnost ove vrste NoSQL baza podataka je temeljena na horizontalnom skaliranju. Dokument baze podataka se najviše koriste u web aplikacijama, računalnim igrama, web trgovinama i umrežavanju. Najpoznatiji predstavnici dokument baza podataka su DocumentDB, MongoDB, OrientDB i CouchDB. [15]

4.1.3. Stupčaste baze podatka

Stupčaste baze podataka (engl. *Column Store Database*) su NoSQL baze podataka koje podatke pohranjuju koristeći model orijentiran ka stupcima. Ovaj tip baze podataka se smatra NoSQL bazom podataka jer koristi različit model pohrane podataka u odnosu na relacijsku bazu podataka. Pohrana podataka u stupčaste baze podataka je sklonija pohrani podataka u stupce nego u redove. Pohrana po stupcima kod dosta upita ubrzava rad, budući da se ne moraju analizirati nepotrebni podaci u redcima, nego se direktno radi sa stupcima.

U odnosu na relacijske baze podataka, logički model je sljedeći: tablica je skup redova sa samo jednom strukturom, redak je skup stupaca, a stupac je skalarna vrijednost iz jedne domene. [16] Vidljivo je da stupčaste baze podataka dijele neke sličnosti s ključ–vrijednost, dokument pa čak i relacijskim modelom baze podataka. Stupčaste baze podataka se mogu koristiti i za manje skupove podataka, ali prvenstveno su predviđene za pohranu velikih skupova podataka.

Pojedini redci u stupčastoj bazi podataka jedinstveno se mogu identificirati preko identifikatora, slično kao primarni ključ u relacijskim bazama podataka. Jedan redak predstavlja jedan entitet, a atributi tog entiteta su zapisani po stupcima. Stupac jedinstveno određuje njegovo ime, a vrijednost koja se može pohranit u stupac mora biti jedan od primitivnih tipova podataka. Kod stupčastih baza podataka vrlo važna stvar je dobro osmisliti ključ retka. Dobro osmišljen ključ retka trebao bi opisivati zapis i njegov sadržaj, a omogućuje brzi pronalazak toga zapisa unutar baze podataka.

Ključevi redaka se također koriste i za pravilnu distribuciju podataka između servera. Ukoliko je ključ retka loše osmišljen, moguće je da se dogodi situacija gdje jedan server zaprima većinu zahtjeva. Ovakva situacija znatno usporava rad cijelog sustava.

Unutar stupca je moguće da svaka vrijednost sadrži više različitih inačica te vrijednosti. Zbog toga, vrijednosti unutar stupca se označavaju vremenskim oznakama. To znači da vrijednost koja je prva dodana neće biti promijenjena, nego će se samo na kraj zapisa dodavati nove vrijednosti sa različitim vremenskim oznakama. Kada se govori o vremenskim oznakama vrijednosti, uz njih se uvijek veže i pojam sažimanja. Sažimanje predstavlja proces spajanja podataka kroz vrijeme i brisanje starih podataka. Obrisani će biti samo oni podaci koji su dobili oznaku za brisanje. Stupčaste baze podatka se najviše koriste kod pohrane i analize velikih skupova podataka, a najpoznatije implementacije ove vrste NoSQL baza podataka su Bigtable, Cassandra, HBase, Vertica i Druid. [15]

Kako je ranije rečeno, NoSQL baze podataka možemo podijeliti na četiri vrste. Do sada, u radu su ukratko opisane tri vrste. Četvrta vrsta NoSQL baza podataka – graf baze podataka bit će detaljnije opisana u nastavku rada.

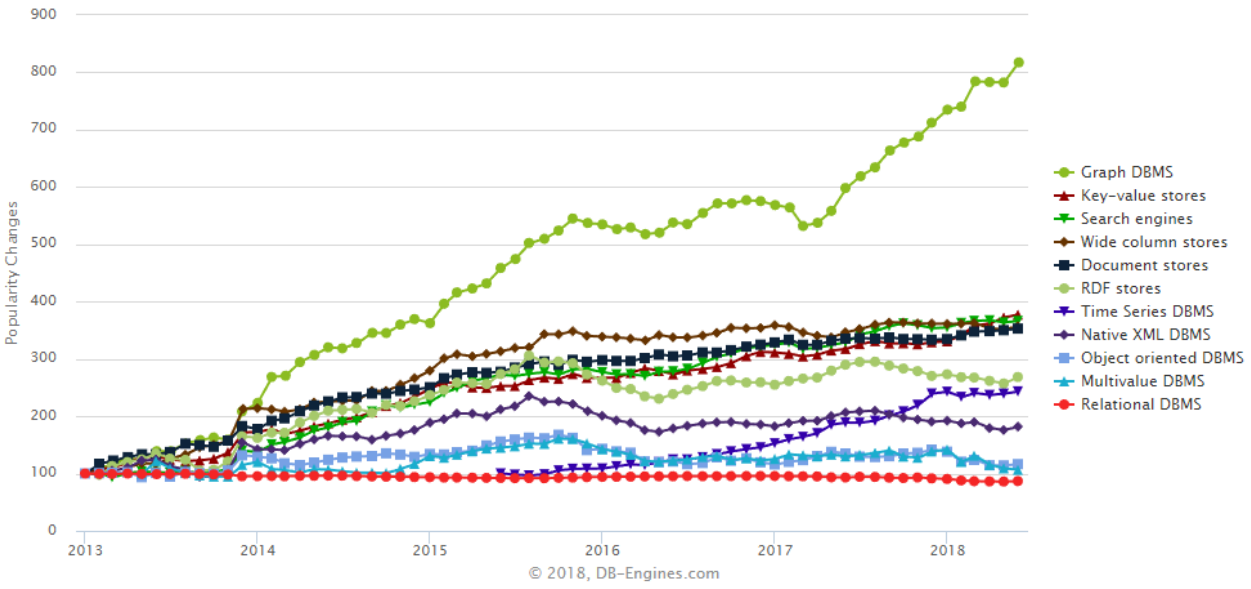
4.2. Graf baze podataka

Ranije u radu je rečeno da su graf baze podataka jedna od vrsta NoSQL baza podataka. O svim NoSQL bazama podataka može se reći puno više nego što je napisano u ovom radu. Ipak, budući da se rad temelji na usporedbi relacijskih i graf baza podataka, detaljnije će biti opisane samo graf baze podataka. Često se u literaturi može naći da graf baze podataka predstavljaju alternativu relacijskome modelu baze podataka, pa čak i da su graf baze podataka relacijske baze podataka nove generacije. Razlog zašto je to tako vjerojatno leži u činjenici što se oba tipa baze podataka temelje na sličnim matematičkim modelima. Budući da se temelje na matematičkim modelima, to ih čini međusobno više usporedivima u odnosu na druge vrste baza podataka. Matematički model na kojemu se temelje graf baze podataka je graf, odnosno teorija grafova. Iz toga se može izvući i definicija graf baza podataka koja kaže da je graf baza podataka ona baza podataka kod koje se podaci pohranjuju u strukturu grafa.

Povećanjem složenosti i količine podataka njihovo modeliranje u relacijsku shemu je znatno otežano. Budući da je model graf baze podataka sličan relacijskom modelu baze podataka, graf baze podataka se nameću kao prirodnije rješenje za ovaj problem. Tehnologija graf baze podataka omogućuje potpuno iskorištavanje potencijala suvremenih podataka, kako u jednostavnim, tako i

u složenijim situacijama. Dok kod relacijske baze podataka postoji unaprijed definirana shema podataka, kod graf baza podataka ne postoji unaprijed definirana shema podataka kao takva. Točnije rečeno, bilo koja shema je odraz unesenih podataka. Kako se unose raznovrsniji podaci, shema raste u skladu s tim.

Graf baze podataka svakodnevno povećavaju svoju zastupljenost na tržištu, prvenstveno zahvaljujući sposobnosti pohrane i upravljanju velikom količinom podataka. Ova vrsta NoSQL baza podataka svoju primjenu je pronašla i na društvenim mrežama (Facebook, Twitter) i u tražilicama (Google). Također, graf baze podataka se mogu koristiti kod prikaza povezanosti više osoba, određivanja najbliže udaljenosti između dvije lokacije, sustava preporuke ili kod vizualizacije podataka koji su prethodno pohranjeni. Najpoznatiji sustavi za upravljanje graf bazama podataka su Neo4j, OrientDB, InfiniteGraph, FlockDB i dr. U ovome radu biti će riječi samo o Neo4j platformi, ali prije toga biti će pojašnjena teorija grafova na kojoj se temelje graf baze podataka. Na slici 4.2. je prikazan trend rasta pojedinih vrsta baza podataka u razdoblju od 2013. do 2018. godine. Sa slike 4.2. je jasno vidljivo da najveći trend porasta korištenja imaju graf baze podataka.



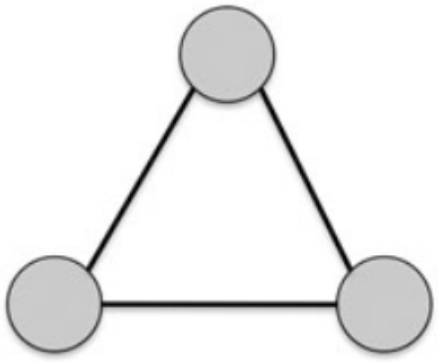
Sl. 4.2. – Trend porasta korištenja pojedinih vrsta baza podataka [23]

4.2.1. Teorija grafova i model grafa

Teorija grafova je samostalna i važna oblast matematike. Grafovi su zanimljivi jer se pomoću njih na jednostavan način mogu modelirati različiti složeni problemi iz prakse. Ti problemi su obično postavljanje prometnica, električnih mreža, računalnih mreža, grafovi koji pokazuju interakciju na

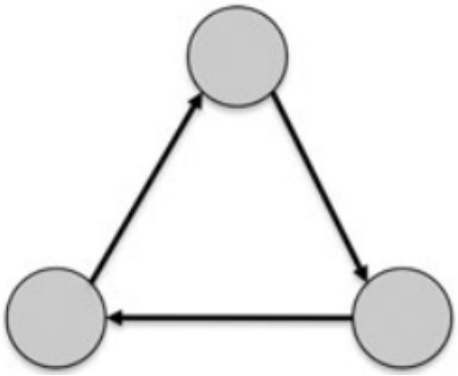
društvenim mrežama i slično. Teorija grafova i model grafa predstavljaju temelje za graf baze podataka, odnosno matematički opis modela graf baze podataka. Zbog toga je jako važno poznavati i razumjeti terminologiju iz područja grafova. U ovome dijelu rada biti će opisani osnovni pojmovi iz matematičke teorije grafova. Navedeni pojmovi su korisni kod modeliranja i rada sa graf bazom podataka.

Svaka priča o teoriji grafova započinje definicijom grafa, a prema [17] graf *G* je definiran kao trojka koja se sastoji od skupa vrhova *V(G),* skupa bridova *E(G)* i relacijama kojima bridovi povezuju dva vrha (ne nužno različitih). Neformalno rečeno, graf je apstraktni matematički model koji je predstavljen točkama (ili kružnica) i linija između njih. Točke (ili kružnice) su vrhovi grafa, a linije između njih su relacije kojima bridovi povezuju vrhove grafa. Za vrhove grafa u literaturi se može naći još i pojam čvor. Uobičajeni način prikaza grafa je slika u ravnini, a slika 4.3. predstavlja prikaz jednog jednostavnog grafa koji sadrži tri vrha (čvora) i tri brida.



Sl. 4.3. – Jednostavan prikaz grafa

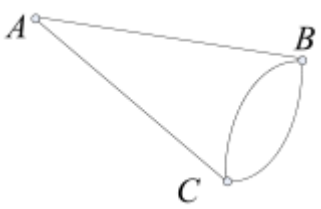
Prema navedenoj definiciji grafa, graf se predstavlja kao uređena trojka sastavljena od vrhova, bridova i relacija između vrhova. U praksi je uglavnom slučaj da se graf predstavi kao skup vrhova i bridova. Također, grafom se može smatrati i skup koji je sastavljen od samo jednog vrha, gdje brid počinje i završava u tome vrhu i na taj način čini petlju. Pored toga, bridovi koji povezuju dva vrha često imaju i usmjerenje. Usmjerenje je realizirano na način da linija koja predstavlja brid na svome kraju ima strelicu koja označava smjer. Jedan takav graf je prikazan na slici 4.4.



Sl. 4.4. – Jednostavan prikaz grafa s usmjerenim bridovima

Obzirom na to jesu li bridovi usmjereni ili ne, može se govoriti o dvije vrste grafova. Te vrste su usmjereni graf i neusmjereni graf. Iz samog naziva lako se da zaključiti da neusmjereni graf ima bridove koji nisu usmjereni. Isto tako, da se zaključiti da onaj graf koji ima usmjerene bridove ustvari je usmjereni graf. Shodno tome, slika 4.3. predstavlja neusmjereni graf, a slika 4.4. predstavlja usmjereni graf. Usmjereni graf u matematici se naziva digraf. Jednostavno rečeno, neusmjereni graf je uređeni par skupova *(V,E)* gdje elementi skupa *V* predstavljaju vrhove, a elementi skupa *E* predstavljaju bridove. Kod crtanja, odnosno prikaza neusmjerenog grafa, raspored vrhova i bridova u prostoru nije važan, ali je važno da vrhovi grafa budu povezani istim bridovima.

Osnovna razlika između usmjerenih i neusmjerenih grafova je ta da je brid u usmjerenom grafu uređeni par vrhova (čvorova). Primjera radi, za brid X = (A,B) se kaže da povezuje vrh A s vrhom B, s tim da to nužno ne znači da povezuje i vrh B s vrhom A. Umjereni graf je moguće dobiti iz neusmjerenog ukoliko se svaki brid zamijeni sa druga dva brida – jedan u jednom smjeru i drugi u drugom smjeru. Broj grana kojima je neki vrh krajnja točka se naziva stupanj vrha (ili stupanj čvora). Ukoliko su neka dva vrha povezana s više od jednim bridom, tada se za taj graf kaže da je multigraf. Multigraf je prikazan na slici 4.5.



Sl. 4.5. – Multigraf

Također, matematička teorija grafova poznaje i pojam težinskih grafova. Težinski graf je onaj graf čiji bridovi imaju težinu. Za pojam težine se u literaturi može naći pojam cijena. Baš kao i bestežinski grafovi, tako i težinski grafovi mogu biti usmjereni ili neusmjereni. U suprotnome, ako graf nema petlji ni višestrukih bridova za takav graf kažemo da je jednostavan. Niz čiji su članovi naizmjenice vrhovi i bridovi se naziva šetnja u grafu. Ako su svi bridovi šetnje međusobno različiti, onda se to zove staza. Staza na kojoj su svi vrhovi različiti se naziva put. Zatvorena staza pozitivne duljine čiji su vrhovi, osim krajeva, međusobno različiti zove se ciklus. [18]

Operacija koja se često izvodi nad grafovima je obilazak grafa. Ova operacije je od velike koristi kod traženja određenog čvora. Stoga, može se reći da obilazak grafa predstavlja i pretraživanje grafa. U teoriji grafova poznate su dvije vrste pretraživanja grafa. Jedna je pretraživanje grafa po širini, a druga pretraživanje grafa po dubini. Ove vrste pretraživanja grafa se nazivaju algoritmi pretraživanja grafa.

Algoritam pretraživanja grafa u širinu predstavlja obilazak grafa razinu po razinu. Spomenuti algoritam započinje u određenom vrhu koji se naziva početni vrh. Nastavlja se tako da se pronađu svi susjedni vrhovi toga početnoga vrha. Zatim se prelazi na traženje „susjedovih susjeda“. Ovaj postupak ide sve dok se ne pronađu svi vrhovi dostupni iz početnog vrha.

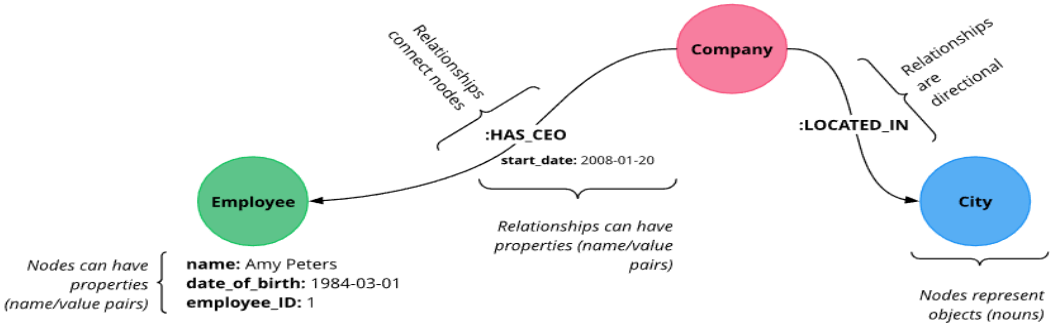
Drugi algoritam, algoritam pretraživanja u dubinu radi na sličan način kao i pretraživanje u širinu. Razlika je u tome što ovaj algoritam nastoji uvijek ići „dublje“ u graf. Pretraživanje se vrši na način da za svaki susjedni vrh *v* vrha *u*, pretraživanje u dubinu rekurzivno se poziva i provjerava susjede vrha *v* prije nego se vrati i provjeri preostale susjede vrha *u*.

4.2.2. Modeliranje graf baze podataka

Modeliranje kao postupak je definiran u poglavlju koje govori o relacijskim bazama podataka. Baš kao i kod relacijskih i graf bazu podataka prije same implementacije potrebno je modelirati. Postupak modeliranja kao rezultat će imati izrađen model graf baze podataka. Na osnovu toga modela, kasnije se izrađuje graf baza podataka. Osnova za modeliranje graf baze podataka je graf model podataka sa svojstvima (engl. *property graph data model*).

Prije svega, važno je reći da živimo u povezanom svijetu. U takvom svijetu, samo baza podataka koja nativno obuhvaća veze može učinkoviti pohraniti i obrađivati veze, a takve bazu su graf baze podataka. Pristupanje čvorovima i vezama u graf bazama podataka je učinkovita, vremenski konstantna operacija koja omogućuje brzi prijelaz milijuna veza u sekundi po jezgri.

Graf model podataka je osnova za izradu graf baze podataka. Ovaj model podataka se temelji na matematičkom modelu grafa, a sadrži povezane entitete koje imaju atribute. I ovdje su entiteti objekti iz realnog svijeta, a predstavljeni su čvorovima grafa. Svaki od njih ima svoj identifikator (oznaku) i svojstva, odnosno atribute u obliku parova ključ–vrijednost. Više čvorova može imati istu oznaku (jedu ili više). Bridovi grafa u graf modelu podataka sa svojstvima predstavljaju veze. Vezama se povezuju dva čvora, tj. entiteta iz realnog svijeta. Svaka veza u modelu ima usmjerenje, identifikator, tip, te početni (izvorišni) i završni (odredišni) čvor. Također, veza može imati svojstva, i ta svojstva su uglavnom određene kvantitativne vrijednosti iz realnog svijeta. Ovakav model podataka je uglavnom dostatan za opis velike većine slučajeva iz realnog svijeta. Jedan primjer graf modela podataka sa svojstvima dan je na slici 4.6.



Sl. 4.6. – Graf model podataka sa svojstvima [20]

Prvi korak kod modeliranja je upoznavanje s domenom problema. To znači da se trebaju dobro razumjeti predmet modeliranja te veze i pravila po kojima su objekti u tom predmetu povezani. Kod relacijskih baza podataka ovo se ostvarivalo izradom konceptualnog modela, odnosno E–R dijagrama. Slična situacija je i kod modeliranja graf baze podataka. Ovdje se entiteti iz relacijskog modela mogu smatrati čvorovima grafa, atributi su svojstva čvora, a koncept stranog (vanjskog) ključa se realizira pomoću veze između dva čvora. Ovdje veza može biti jednosmjerna ili dvosmjerna. Kod modeliranja relacijskog modela svakom entitetu i vezi daju su imena. U slučaju graf baze podataka ta imena se predstavljaju oznakom (engl. *labels*). Oznake se koriste za grupiranje čvorova u skupove, a jedan čvor može imati više oznaka. Također, ove oznake se indeksiraju zbog ubrzavanja pretraživanja, odnosno ubrzavanje pronalaženja čvor u grafu. Nakon početne analize domene modeliranja te identificiranja čvorova i veza kojima su povezani čvorovi prelazi se na usavršavanje toga modela. Usavršavanje modela podrazumijeva dodatnu provjeru jesu li svi entiteti iz promatrane domene predstavljeni čvorovima, jesu li atributi čvorova potpuni te jesu li sve veze koje postoje u stvarnosti između čvorova prikazane na modelu grafa i sadrže li

ispravna svojstva. Nakon što se završi proces usavršavanja modela dobije se potpuniji i točniji model grafa, koji je spreman za fizičku implementaciju.

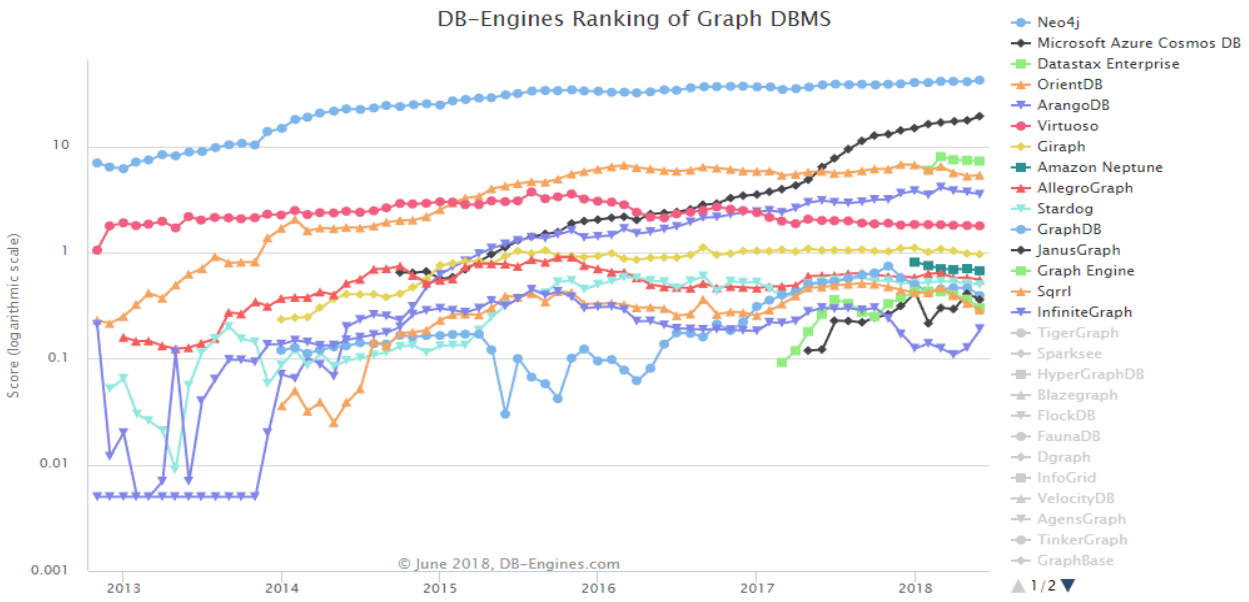
Kod modeliranja relacijskog modela, potrebna je pretvorba konceptualnog modela u logički model, te normalizacija toga logičkog modela. Ovaj korak kod modeliranja graf baze podataka nije potreban. To znači da konceptualni model koje je predstavljen u obliku grafa se kao takav i pohranjuje u graf bazu podataka. Na taj način se povećava povezanost između stvarne situacije i one koja je prikazana konceptualnim modelom grafa. Zapravo, može se reći da su graf baze podataka bez sheme, a to znači da nije potrebno definirati čvor prije nego se u njega spreme podaci. Upravo ovo svojstvo omogućava obavljanje promjena na vrlo lak način.

Relacijske baze podataka su prvenstveno bile izrađene zbog olakšavanja posla popunjavanja i pohrane formulara u papirnatom obliku i tabličnih struktura. Taj posao one rade vrlo dobro i još uvijek su temelj većine aplikacija. Ipak, takve aplikacije moraju biti visoko strukturirane, budući da one pohranjuju podatke u već predefinirane tablice s predodređenim stupcima i u retke pohranjuju mnogo informacija iste vrste. Današnji zahtjevi korisnika su malo drugačiji, odnosno traže više od toga – više mogućnosti, više podataka, više agilnosti, više brzine i više veza. Ironično u svemu tome je što relacijske baze podataka nisu učinkovite u rješavanju relacija (povezanosti) između entiteta, osobito u sustavima gdje su česte izmjene podataka. Prilikom povećanja količine podataka i njihove složenosti i različitosti relacijske baze podataka su opterećene velikim spojenim tablicama koje znaju omesti rad i daljnji razvoj. Kao rješenje ovoga problema, nameće se upotreba graf baza podataka.

Ostale NoSQL baze podataka pohranjuju skupove nepovezanih dokumenata, stupaca i vrijednosti. Iako i one imaju svoje prednosti u odnosu na relacijske baze podataka, njihova nepovezana konstrukcija otežava ispravno korištenje odnosa između podataka. [14], [19], [20], [21]

4.2.3. Neo4j

Baš kao i relacijske baze podataka, tako i graf baze podataka imaju sustav za upravljanje bazom podataka, odnosno sustav za upravljanje graf bazom podataka. Kod graf baza podataka jedan od najpoznatijih i najčešće korištenih sustava za upravljanje graf bazom podataka je Neo4j. Upravo iz toga razloga Neo4j je odabran za korištenje i u ovom radu. Dijagram popularnosti pojedinih sustava za upravljanje graf bazom podataka, s kojeg se jasno vidi da Neo4j zaista i je najzastupljeniji na tržištu dugi niz godina, prikazan je na slici 4.7.



Sl. 4.7. – Popularnost pojedinih sustava za upravljanje graf bazom podataka u lipnju 2018. godine [23]

Neo4j sustav za upravljanje graf bazom podataka je razvila istoimena tvrtka. Izgradnja prvog prototipa Neo4j sustava za upravljanje graf bazom podataka je započeta 2000. godine kao pokušaj otklanjanja nedostataka u performansama kod relacijskih baza podataka. Prva ikad razvijena inačica Neo4j se pojavila 2002. godine, 2010. godine je izašla inačica Neo4j 1.0. Trenutna inačica, 3.0 datira iz 2016. godine. S vremenom su se u Neo4j sustav za upravljanje graf bazom podataka implementirale dodatne funkcionalnosti, tako da je Neo4j danas poznat kao platforma za povezan podatke, a sustav za upravljanje graf bazom podataka čini jedna modul te platforme. Za razvoj Neo4j platforme korišten je programski jezik Java. Još jedna stvar koja krasi ovaj sustav za upravljanje graf bazama podataka je mogućnost uvoza podataka iz CSV datoteka.

Neo4j je od temelja rađen kako bi bio graf baza podataka, a to znači da je njegova arhitektura dizajnirana za optimiziranje brzog upravljanja, pohranu i pronalazak čvorova i veza između njih. Stoga, za odnose između čvorova, kod Neo4j se kaže da su „građani prvog reda“. Kod relacijskih baza podataka, operacija spajanja (engl. *Join*) degradirat će eksponencijalno s brojem veza koje sudjeluju u tome spajanju. Odgovarajuća akcija u Neo4j izvodi se kao navigacija od jednog do drugog čvora, i time je izvedba ove operacije linearna. Ovakav pristup pri pohrani i izvođenju upita nad vezama između entiteta pruža veoma visoku uspješnost. Budući da je većina pretraživanja graf baze podataka lokalizirana u širem susjedstvu čvora, ukupna količina podataka pohranjenih u bazi podataka neće utjecati na vrijeme izvođenja operacije. Ovako decidirano upravljanje memorijom i visoko skalabilni i memorijski učinkovite operacije uvelike pridonose performansama.

Pristup modelu graf baze podataka je uvelike olakšan u odnosu na relacijske baze podataka. Pod tim se misli da Neo4j omogućuje dosljednu upotrebu istog modela kroz koncepciju, dizajn, implementaciju, pohranu i vizualizaciju tih podataka. Glavna prednost ovoga je što je omogućeno sudjelovanje svim sudionicima iz domene problema u razvojnom ciklusu. Pored toga, moguće je model domene razvijati neprekidno dok se mijenjaju zahtjevi, bez skupih promjena sheme i migracija.

Neo4j omogućuje pohranu 10^{12} (bilijuna, engl. *trillions*) entiteta, te je kao takav pogodan za pohranu najvećih skupova podataka koji su zamislivi. Za upotrebu u produkcijskim okruženjima moguće ih je koristi kao skalabilne skupove servera tolerantne na greške. Upravo zbog velike skalabilnosti, Neo4j skupovi servera zahtijevaju samo par desetaka strojeva, a ne stotine ili tisuće kao što je slučaj kod nekih drugih sustava. Takvim zahtjevima za strojevima se značajno umanjuju troškovi i operativna složenost. [24]

Vrlo važna značajka, koja izdvaja Neo4j u odnosu na druge NoSQL baze podataka je i ta što Neo4j podržava ACID svojstva. ACID svojstva su opisanu u dijelu o relacijskim bazama podataka. Kada su u pitanju NoSQL, pa tako i graf baze podataka modeli dosljednosti podataka su često nevjerojatno različiti u odnosu na model dosljednosti podataka kod relacijskog modela. Također, model dosljednosti podataka se može razlikovati i između NoSQL vrsta baza podataka. Pored spomenutog ACID modela dosljednosti podataka, drugi najčešće i najzastupljeniji model dosljednosti je BASE (akronim od engl. ***B**asic **A**vilability, **S**oft–state, **E**ventual consistency*). Navedena dva modela dosljednosti podataka u graf bazama podataka, odnosno Neo4j koegzistiraju, te je na taj način moguće ostvariti određene prednosti prilikom transakcija u bazi podataka. Kod Neo4j graf baze podataka, ACID model dosljednosti se koristi prvenstveno za sigurno i dosljedno pohranjivanje podataka.

U mnogim domenama i slučajevima upotrebe, ACID transakcije su previše pesimistične. Preciznije rečeno, pesimističnost kod ACID transakcija znači da se puno više brine o sigurnosti podataka nego što to domena problema zapravo zahtjeva. Kod NoSQL, pa tako i Neo4j graf baza podataka ACID transakcije su manje „moderne“ nego kod relacijske baze podataka. Razlog tome je što su s vremenom pojedine baze podataka olabavile zahtjeve za neposrednu dosljednost podataka, pravovremenost i točnost podataka, a sve to zbog ostvarivanja drugih prednosti poput skaliranja i otpornosti. Jasno je da se na ovakav način gubi dosljednost, ali pouzdanost sustava i podataka u njemu je osigurana BASE modelom. BASE model također zadovoljava fleksibilnost

koju pruža NoSQL model baze podataka i slični pristupi upravljanja i pohrane nestrukturiranih podataka.

Osnovna dostupnost (engl. *Basic Avilability*) predstavlja svojstvo koje znači da baza podataka radi i da joj je moguće pristupiti većinu vremena. Ovo se postiže korištenjem visoko distribuiranog pristupa upravljanju bazom podataka. Umjesto da održava jedan veliki skup pohranjenih podataka i usredotoči se na toleranciju grešaka u toj jedinici za pohranu, NoSQL baze podataka šire svoje podatke preko mnogih sustava za pohranu s visokim stupnjem replikacije. Značajka mekano stanje (engl. *soft–state*) označava da jedinice za pohranu ne moraju biti konzistentne za pisanje, tj. pohranu podataka, niti različite replike uvijek moraju biti međusobno konzistentne. Moguća dosljednost (engl. *Eventual consistency*) znači da sustav pokazuje dosljednost u nekoj kasnijoj točki. Zapravo, ovo je jedini zahtjev koji NoSQL baza podataka postavlja u vezi s dosljednošću podataka. To znači da zahtijeva da podaci u nekom budućem trenutku, u odnosu na trenutak pohrane, konvergiraju u dosljedno stanje. Međutim, nije zajamčeno kada i u kojem trenutku će se dogodit ta konvergencija podataka u dosljedno stanje. Ovakav zahtjev predstavlja potpuno odstupanje u odnosu na ACID model konzistentnosti koji zabranjuje da se transakcija izvrši sve dok prethodna transakcija nije izvršena i dok podaci u bazi podataka nisu postali konzistentni. BASE model nije prikladan za svaki slučaj, ali predstavlja fleksibilnu alternativu, pa čak i nadopunu ACID modelu dosljednosti, pri čemu se ne zahtjeva strogo pridržavanje relacijskog modela. [25],[26]

4.2.4. Cypher upitni jezik

Kako bi operacije nad podacima bile moguće, sustav za upravljanje (graf) bazom podataka treba podržavati rad s jednim ili više upitnih jezika koji će izvoditi te operacije. Jezik pomoću kojeg je moguće manipulirati s podacima u Neo4j graf bazi podataka je Cypher Query Language, ili skraćeno Cypher. Cypher, baš kao i SQL manipulaciju nad podacima obavlja izvođenjem upita. Isto tako, Cypher je deklarativni upitni jezik. Deklarativan znači da se fokusira na aspekte rezultata, a ne na metode ili načine za dobivanje rezultata, tako da je čitljiv čovjeku i ekspresivan. Konkretno, deklarativni jezik uključuje davanje opširnih uputa o tome koji će zadatak biti izvršen, ali izostavlja specifičnosti o tome kako će biti izvršen.

Pored deklarativnih jezika, postoje i imperativni upitni jezici. Za upotrebu je puno jednostavniji deklarativni jezik, jer u slučaju korištenja imperativnog jezika potrebno je opsežno poznavanje jezika i duboko tehničko razumijevanje pojedine fizičke implementacije, te kao takvi imperativni jezici su skloniji ljudskoj pogrešci. [22]

Budući da je vrlo čitljiv, upiti napisani u Cypher jeziku su veoma laki za održavanje, čime je pojednostavljeno održavanje aplikacija. Upitni jezik Cypher je inspiriran SQL jezikom, što se može vidjeti i po nazivima određenih naredbi i klauzula koje su identične onima u SQL jeziku. Također, određeni dijelovi sintakse su preuzeti iz SQL jezika. Sintaksa Cypher jezika pruža poznati način za pronalaženje uzoraka u grafu. Pored toga što je inspiriran SQL jezikom, koncept podudaranja uzoraka preuzeo je iz SPARQL, odnosno Cypher jezik od baze podataka zatraži pronalazak podataka koji odgovaraju određenom uzorku (engl. *pattern*). Značajna stvar kod Cypher jezika je da su njegove naredbe kraće u odnosu na SQL jezik. To ne znači da će vrijeme izvršavanja te naredbe biti kraće u odnosu na istu naredbu napisanu u SQL jeziku, ali znači da je manja mogućnost ljudske greške prilikom pisanja složenijih upita. Kod modeliranja graf baze podataka je rečeno da je izrađeni model ono što se pohranjuje u graf bazu podataka. Praktično, to znači da se upotrebnom Cypher jezika stvara i fizička implementacija toga modela. Dok se kod SQL jezika radi o tablicama i spajanju tih tablica, kod Cypher jezika se radi o odnosima između entiteta (čvorova). Baš kao što je prirodnije raditi graf model, tako je prirodnije upotrebljavati i Cypher jezik jer potječe od nacrtanog modela grafa. Ovo svojstvo Cypher jezik čini dosta čitljivim i razumljivijim čak i nekome tko nije stručnjak u području (graf) baza podatka. Jednostavno rečeno, model grafa i upiti na taj graf su lice i naličje iste stvari. Pravi upitni jezik u bazi podataka omogućuje pronalazak obje strane. Jezik Cypher je lako koristiti jer se podudara s načinom na koji se intuitivno opisuje graf pomoću dijagrama.

Kako bi bilo moguće izvršavati operacije nad podacima u graf bazi podataka korištenjem Cypher jezika, on treba, uz naredbe podržavati tipove podataka, operatore, te određene funkcije. Što se tiče samih tipova podataka, kod Cypher jezika oni se dijele u tri kategorije – svojstveni, strukturni i kompozitni tipovi podataka. Svojstveni tipovi podataka mogu biti vraćeni kao rezultat Cypher upita, korišten kao parametar, pohranjen kao svojstvo i konstruirani sa Cypher literalima. Strukturni tipovi podataka samo mogu biti vraćeni kao rezultat Cypher upita, dok kompozitni tipovi podataka mogu biti vraćeni kao rezultat Cypher upita, korišteni kao parametar, konstruirani sa Cypher literalima, ali ne mogu biti pohranjeni kao svojstvo. U tablici 4.1. su prikazani tipovi podataka u Cypher jeziku.

Tablica 4.1. – Tipovi podataka u Cypher jeziku

NAZIV	OPIS ili PODTIP	
Number	Abstraktni tip podatka, podtipovi integer i float.	SVOJSTVENI TIPOVI PODATAKA
String	Za pohranu niza znakova.	
Boolean	Binarni tip podatka.	
Point	-	
Temporal types	Date, Time, LocalTime, DateTime, LocalDateTime, Duration	
Node	Sadrži Id, Labels, Map	STRUKTURNI TIPOVI PODATAKA
Relationships	Sadrži Id,Type, Map, Id početnog i završnog čcora	
Path	Alternativni slijed čvorova i veza.	
List	heterogene, uređene zbirke vrijednosti, od kojih svaka ima svojstvo, strukturni ili kompozitni tip	KOMPOZITNI TIPOVI PODATAKA
Maps	heterogene, neuređene zbirke (ključ, vrijednost) parova, gdje je ključ tipa String, a vrijednost ima bilo koje svojstvo, strukturni ili kompozitni tip	

Sljedeća stavka koju jezik Cypher podržava su operatori. Prema [27] operatori u Cypher jeziku se dijele na generalne, matematičke, operatore usporedbe, binarne, *string*, *list*, regularne izraze, *string matching*. Operatori u Cypher jeziku su prikazani u tablici 4.2.

Tablica 4.2. – Operatori u Cypher jeziku

NAZIV GRUPE OPERATORA	OPERATORI
Generalni operatori	DISTINCT, ., [ ]
Matematički operatori	+, -, *, /, %, ^
Operatori usporedbe	=, < >, <, >, <=, >=, IS NULL, IS NOT NULL
Binarni (boolean) operatori	AND, OR, XOR, NOT
String operatori	+
List operatori	+, IN, [x], [x…y]
Regularni izrazi	=~
String matching	STARTS WITH, ENDS WITH, CONTAINS

Operator *null* se koristi za prikaz nedefiniranih/nepostojećih vrijednosti. Ukoliko su dvije vrijednosti *null*, to ne znači da su one iste, nego znači da u tim vrijednostima nedostaje konkretna vrijednost. Aritmetički izrazi, usporedbe i funkcije poziva će vratiti vrijednost *null* ukoliko je bilo koji argument *null*. Također, Cypher jezik podržava i određene grupe funkcija. Funkcije koje su podržane u Cypher jeziku se dijele na matematičke, agregacijske, *string*, funkcije za veze, funkcije za trajanje, privremene (engl. *temporal*) funkcije, prostorne funkcije i funkcije za putanje. Svi navedeni tipovi podataka, funkcije i operatori ne bi imali svoju primjenu bez naredbi pomoću kojih se kreiraju upit u Cypher jeziku. Cypher jezik podržava naredbe kao i SQL jezik, s tim da se neke drugačije zovu, neke se zovu isto, a neke i ne postoje u SQL jeziku. Popis naredbi i klauzula u Cypher jeziku dan je u tablici 4.3.

Tablica 4.3. – Popis naredbi i klauzula u Cypher jeziku

NAZIV NAREDBE/KLAUZULE	OPIS NAREDBE/KLAUZULE
MATCH	U njoj se navodi uzorak prema kojem se vrši pretraživanje.
RETURN	Definira što treba sadržavati rezultat upita.
ORDER BY	Definira poredak (uzlazno ili silazno) vraćenog rezultata; dolazi nakon RETURN
LIMIT	Ograničava broj redova u rezultatu upita.
SKIP	Definira od kojeg reda treba početi prikazivati redove rezultata upita.
UNION	Kombiniranje rezultata više različitih upita u jedan skup.
OPTIONAL MATCH	Ekvivalent vanjskom spajanju u SQL.
START	Koristi se za pronalazak ishodišne točke od koje kreće pretraživanje.
WHERE	Filtrira rezultate dobivene sa MATCH i OPTIONAL MATCH.
AGGREGATION	Ekvivalentno GROUP BY u SQL.
CREATE	Kreiranje čvora ili veze.
MERGE	Kreira elemente traženog uzorka ukoliko n ne postoji u bazi podataka.
SET	Za ažuriranje oznaka čvorova i veza i svojstava čvorova i veza elemenata graf baze podataka.
DELETE	Za brisanje elemenata graf baze podataka.
REMOVE	Brisanje svojstva i oznake elementa graf baze podataka

U tablici 4.3. se navedeni najkorištenije naredbe u Cypher jeziku. Kako je ranije rečeno, a sada se može i vidjeti iz tablice 4.3., nazivi naredbi u Cypher jeziku većinom su isti kao i u SQL jeziku. Budući da Cypher rezultat upita dohvaća tako što traži uzorak u grafu za to mu je potrebna naredba

MATCH. Moglo bi se reći da je MATCH naredba temelj pri radu s Neo4j graf bazom podataka. U toj naredbi se navede uzorak po kojemu će biti izvršeno pretraživanje graf baze podataka. Također, upit koji se upiše u MATCH naredbi predstavlja putanju. Dakle, Cypher jezik funkcionira tako što, kada se pokrene određeni upit, pretraživanje baze podataka se vrši s nekim od algoritama pretraživanja grafa. Neki od najčešće korištenih algoritama pretraživanja grafa koji se koriste u Neo4j graf bazi podataka su algoritam pretraživanja u širinu, algoritam pretraživanja u dubinu, Dijkstrin algoritam, algoritam tri boje i Ullmanov algoritam. [14], [22], [27]

Iz svega do sada navedenoga u ovome radu, jasno je vidljivo da relacijske baze podataka i graf baze podataka dijele određena svojstva. Ta svojstva su sličnost u modelima, povezanosti podataka, upitnom jeziku i sl. Ovakva sličnost između relacijskog i graf modela baze podataka daje dobre temelje za njihovu međusobnu usporedbu po raznim parametrima. Jedna takva usporedba je dana u sljedećem poglavlju.

5. USPOREDBA I ANALIZA USPOREDBE

U ovome poglavlju je riječ o usporedbi relacijskih baza podataka i graf baza podataka, kao i o analizi rezultata te usporedbe. Za predstavnika relacijske baze podataka u ovoj usporedbi je uzet Microsoft SQL Server, a za predstavnika graf baze podataka već spomenuti Neo4j. U usporedbi su razmatrane sličnosti i razlike koje obuhvaćaju teorijske aspekte, razlike između odgovarajućeg modela podataka, razlike u upitima – izgledu naredbi, duljini naredbe i sl., te kao završi dio analize dan je prikaz i usporedba vremena izvršavanja pojedinih upita kako bi se mogli izvući zaključci provedene usporedbe. Također, kao uvod u ovo poglavlje dan je kratki prikaz rezultata i zaključaka iz radova na ovu temu koju su provodili stručnjaci iz ovog područja. Prije samog prikaza tih rezultata, dan je prikaz kratke usporedbe između Neo4j graf baze podataka i relacijskih baza podataka koju je izradila tvrtka Neo4j, Inc. [28] Usporedba je dana u tablici 5.1.

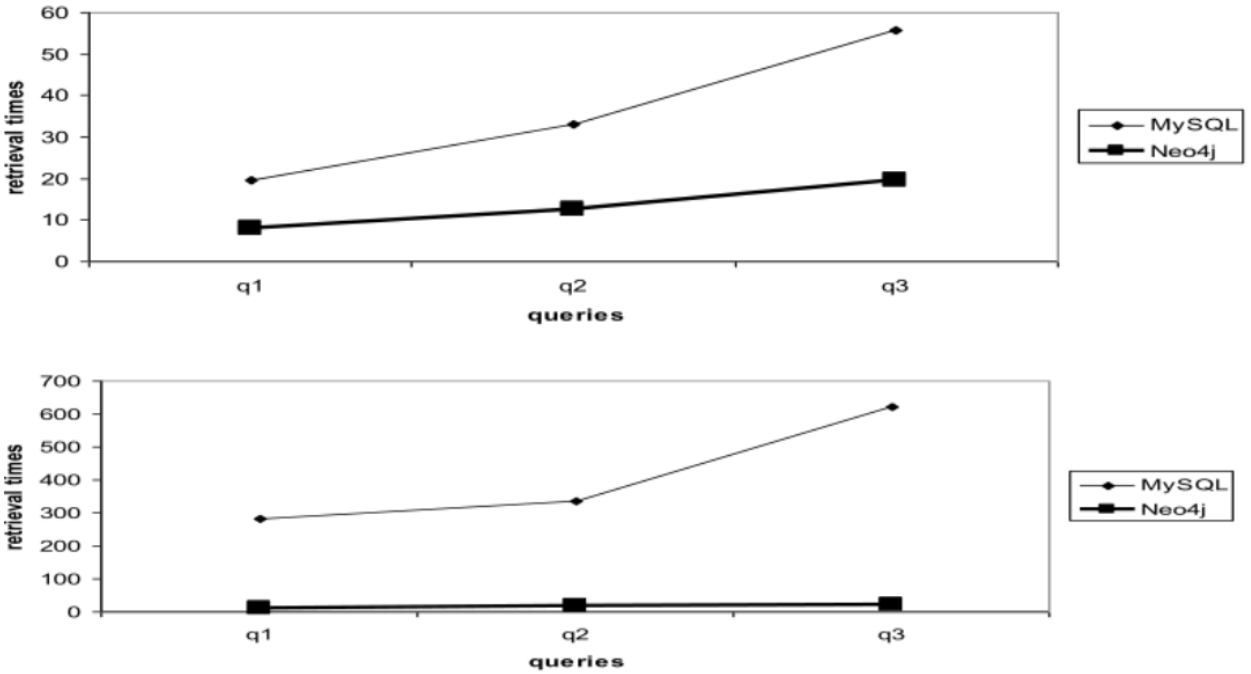
Tablica 5.1. – Usporedba Neo4j graf baze podataka i relacijskih baza podataka prema Neo4j, Inc.

Parametar usporedbe	Relacijska baza podataka	Neo4j graf baza podataka
Pohrana podataka	Podaci su pohranjeni u fiksne, unaprijed definirane tablice koje se sastoje od stupaca i redaka s povezanim podacima koji se često odvajaju između tablica, što smanjuje učinkovitost upita.	Struktura graf baza podataka sa svojstvom slobodnog indeksa rezultira bržim transakcijama i obradom podataka.
Modeliranje podataka	Model baze podataka se mora razviti modeliranjem i potrebno ga je prevesti iz konceptualnog u logički pa tek onda u fizički model. Sve vrste podataka i izvori podataka moraju bit unaprijed poznati, pa sve promjene zahtijevaju stanku u radu za implementaciju promjena.	Fleksibilan proces modeliranja i model podataka koji rezultira nepostojanjem neusklađenosti između logičkog i fizičkog modela. Vrste podataka i izvori podataka mogu se dodati ili mijenjati u bilo kojem trenutku, što dovodi do dramatično kraćih vremena razvoja i prave agilne iteracije.
Upitni jezik	SQL: Upitni jezik kojim se povećava složenost s porastom broja	Cypher: Nativni upitni jezik za graf baze podataka koji pruža

	JOIN operacija potrebnih za povezivanje podataka.	najučinkovitiji i najekspresivniji način za opis povezanosti upitima.
Performanse upita	Performanse obrade podataka padaju sa brojem i dubinom *JOIN* – ova.	Procesiranje grafa osigurava nimalo latentnosti i izvođenje u realnom vremenu, bez obzira na broj ili dubinu odnosa.
Podrška za transakcije	Podrška za ACID transakcije koje zahtijevaju poslovne aplikacije za dosljedne i pouzdane podatke.	Zadržava podršku za ACID transakcije zbog dosljednosti i pouzdanosti podataka tokom cijelog vremena – savršeno za velike *enterprise* sustave.
Obrada pri skaliranju	Skaliranje kroz replikaciju i povećanje arhitekture je moguće ali je skupo. Složene veze podataka nisu pogodne za skaliranje.	Model grafa prirodno (inherentno) skalira upite temeljene na uzorku. Skaliranje arhitekture omogućuje integritet podataka putem replikacije.
Učinkovitost podatkovnih centara	Konsolidacija poslužitelja je moguća, ali je skupo za arhitekturu. Skaliranje arhitekture je skupo u smislu kupnje, korištenja energije i vremena upravljanja.	Podaci i veze se pohranjuju prirodno zajedno sa poboljšanjem performansi kako složenost i skaliranje raste. To dovodi do konsolidacije poslužitelja i nevjerojatno učinkovitog korištenja sklopovlja.

U prilog tvrdnjama koje su dane u tablici iznad idu i rezultati nekih istraživanja i znanstvenih radova. Jedno takvo istraživanje je i [29]. U tom radu je rađena usporedba između relacijske MySQL baze podataka i Neo4j graf baze podataka. Također, autori spomenutog rada konstatiraju u radu da današnje potrebe za pohranom i obradom podataka nadilaze mogućnosti relacijskih baza podataka, iako su relacijske baze podataka još uvijek dobre u određenim segmentima. Usporedba je provedena prema nekoliko različitih kriterija – zrelost, stabilnost, sigurnost i fleksibilnost, koji mogu biti temelj za odlučivanje koju vrstu baze podataka implementirati u određeni sustav. Ustanovljeno je da su relacijske baze podataka zrelije i stabilnije od graf baza podatka, a razlog tome je što su relacijske baze podataka na tržištu već par desetljeća, dok su graf baze podataka relativno nova vrsta baza podataka. Što se tiče sigurnosti, utvrđeno je da Neo4j još uvijek nema

mehanizme za upravljanje sigurnošću i ograničenja za više korisnika, nego se te stavke realiziraju na sloju aplikacije. Suprotno tomu, relacijske baze podataka podržavaju spomenute mehanizme već jako dugi niz godina. Unatoč tomu što su relacijske baze podataka zrelije i sigurnije, one posjeduju jedan veliki nedostatak u odnosu na graf baze podataka. Taj nedostatak je fiksna shema podataka, što praktično znači da je otežana nadogradnja ovih baza podataka. Nadogradnja graf baza podataka je vrlo jednostavna. Kao eksperimentalni dio istraživanja mjerena su vremena izvršenja pojedinih upita u milisekundama. Na osnovu tih mjerenja, autori su zaključili da graf baze podataka daju bolje rezultate, odnosno brže su u usporedbi s relacijskim bazama podataka. Bolje performanse se posebno očituju što je veći broj objekata u bazi. Na slici 5.1. se nalazi grafički prikaz rezultata ovog istraživanja. [29] Gornji graf prikazuje rezultate mjerenja za upite koji su se provodili na uzorku od 100 objekata, a donji prikazuje rezultate mjerenja vremena za upite koji su se provodili na uzorku od 500 objekata u bazi podataka.

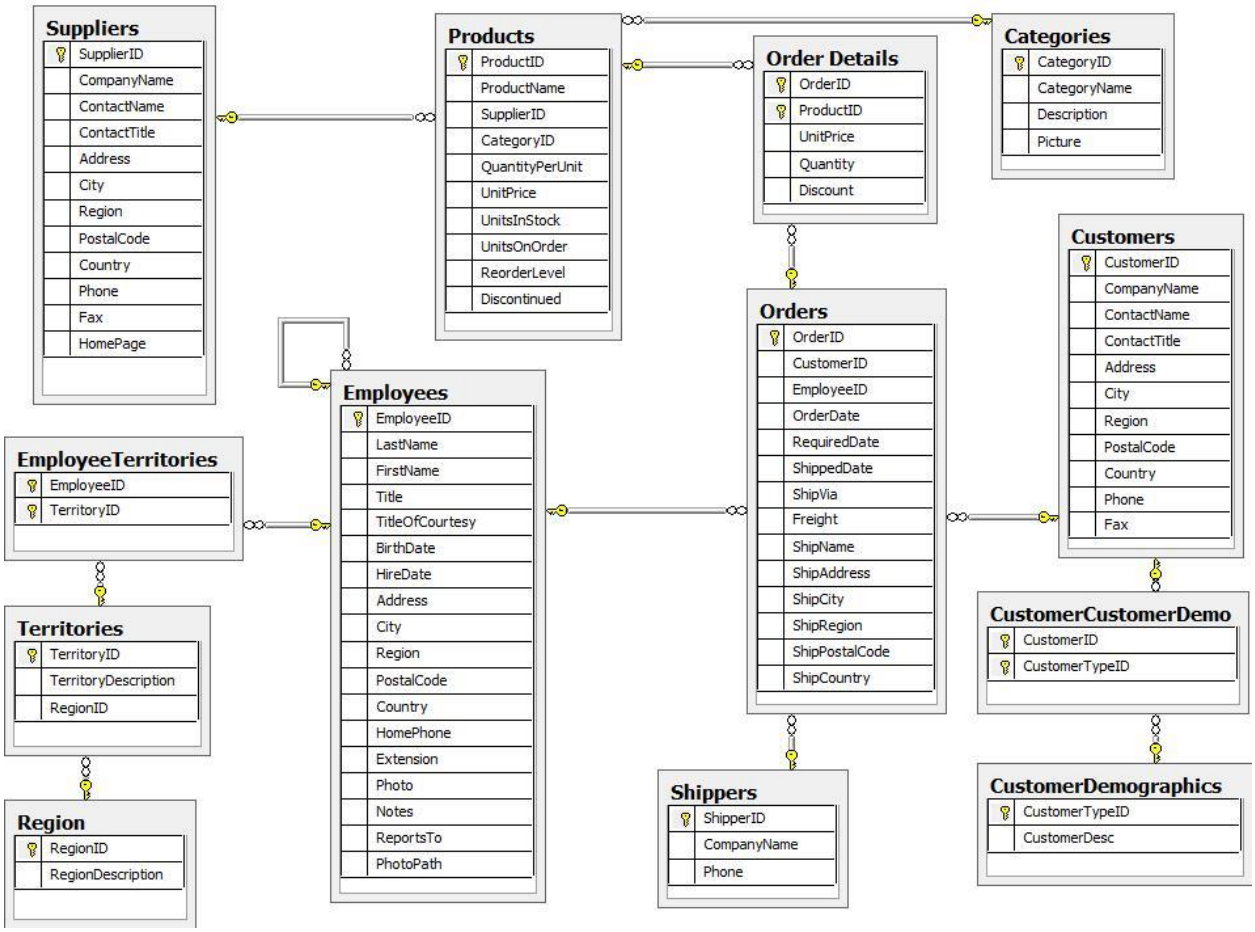


Sl. 5.1. – Rezultati usporedbe prema [29]

Kako je ranije rečeno, relacijske i graf baze podataka se razlikuju i u modelu koji nastane kao rezultat procesa modeliranja. Što se samog procesa modeliranja tiče, kod relacijskih baza podataka ono se odvija u tri koraka – izrada konceptualnog modela, njegova konverzija u logički model te na kraju izrada fizičkog modela koji se temelji na logičkom modelu. Kod graf baza podataka, nije potrebno prevoditi konceptualni model u logički, pa tek onda raditi fizički model. Naime, izrada konceptualnog modela kod graf baza podataka je temelj za izradu fizičkog modela. Jednostavno rečeno, model grafa koji je nacrtan na papiru biti će identičan izgledu grafa koji se dobije nakon

implementacije toga modela. Samim time, pisanje upita za pronalazak podataka koji su interesantni u nekom trenutku je uvelike olakšano u odnosu na relacijske baze podataka. Također, kod graf baza podataka nije potrebno raditi normalizaciju sheme podataka.

Usporedba koja je provedena u ovome radu temelji se na istom podatkovnom skupu. Podatkovni skup koji je korišten u usporedbi je *NorthWind* podatkovni skup, a moguće ga je preuzeti na [30]. Inače, to je najkorišteniji podatkovni skup za prikaz relacijskih baza podataka, ali ga je moguće prikazati i kao graf bazu podataka. Zbog toga razloga je izabran kako bi bio temelj za usporedbu. Konceptualni model podatkovnog skupa *NorthWind* je prikazan na slici 5.2. [31]



Sl. 5.2. – Konceptualni model *NorthWind* baze podataka [31]

Budući da je *NorthWind* podatkovni skup zapravo relacijska baza podataka, na lak način ga je moguće uvesti u MS SQL server, dok je uvoz u Neo4j moguć na način uvoza podataka iz CSV datoteka. Također, Neo4j je prikladan za pohranu svih vrsta podataka – strukturiranih, polustrukturiranih i nestrukturiranih podataka. Prije samog uvoza podataka, tj. njihove pohrane u Neo4j graf bazu podataka potrebno je željene podatke samo transformirati (prilagoditi ih) kako bi odgovarali CSV formatu datoteke.

Prilikom kreiranja graf modela na osnovu relacijskog modela, potrebno je imati na umu da je redak u relacijskoj bazi podataka ekvivalentan čvoru u graf bazi podataka, te da je naziv tablice ekvivalentan oznaci čvora u graf modelu. U graf bazi podataka ne postoje *null* vrijednosti. U relacijskoj inačici, za praćenje hijerarhije zaposlenika u tablici *„Employees“* u stupcu *„ReportsTo“* moguć je slučaj postojanja *null* vrijednost. Kod graf baze podataka taj odnos se jednostavno ne definira. Kod relacijskih baza podataka podaci su spremljeni u tablice, a kod graf baza podataka ekvivalent tablici je oznaka čvora. Redak tablice je ekvivalentan čvoru u grafu, a stupac u tablici je ekvivalentan svojstvu čvora. Veze se kod relacijske baze podataka realiziraju preko stranih ključeva, odnosno ograničenja. Moglo bi se reći da veze u graf bazama podataka dobivaju potpuni smisao, jer se veza prikazuje kao veza i s njom je moguće izvoditi određene upite – kreiranje, brisanje, dodavanje svojstava u vezu i sl. Usporedba u ovome radu obuhvaća osnove upite za dohvat, uređivanje, brisanje i dodavanje novih podataka u bazu podataka. Također, obuhvatit će i neke dodatne upite, a svi upiti koji su korišteni prilikom usporedbe prikazani su u tablici 5.2.

Tablica 5.2. – Upiti korišteni u usporedbi

OZNAKA UPITA	OPIS UPITA
Q1	Dodavanje novoga reda/Dodavanje novog čvora.
Q2	Dohvat svih podataka iz tablice/Dohvat svih čvorova.
Q3	Ažuriranje podataka u redu/Ažuriranje podataka u čvoru.
Q4	Brisanje reda/Brisanje čvora.
Q5	Dohvat određenih podataka.
Q6	Dodavanje atributa u red/Dodavanje svojstva u čvor.
Q7	Dohvat broja redaka/čvorova sa određenim svojstvom.
Q8	Dohvat prosječne vrijednosti određenog svojstva iz tablice/čvora.
Q9	Spajanje tablica i ekvivalentni upit u Cypher jeziku.

Što se tiče samih upitnih jezika, jedan upit u SQL jeziku može biti mnogo dulji nego njegov ekvivalent u Cypher jeziku. To nužno ne znači da će se taj upit izvodit dulje nego isti takav upit u Cypher jeziku, ali znači da je veća mogućnost pojave greške prilikom pisanja upita. Osim toga, kraći upiti povećavaju jednostavnost razumijevanja, pa je samim time i održavanje olakšano. Sintaksa i vrijeme izvršavanja pojedinih upita, koji su definirani prema tablici 5.2., u SQL i Cypher jeziku dana je u tablici 5.3.

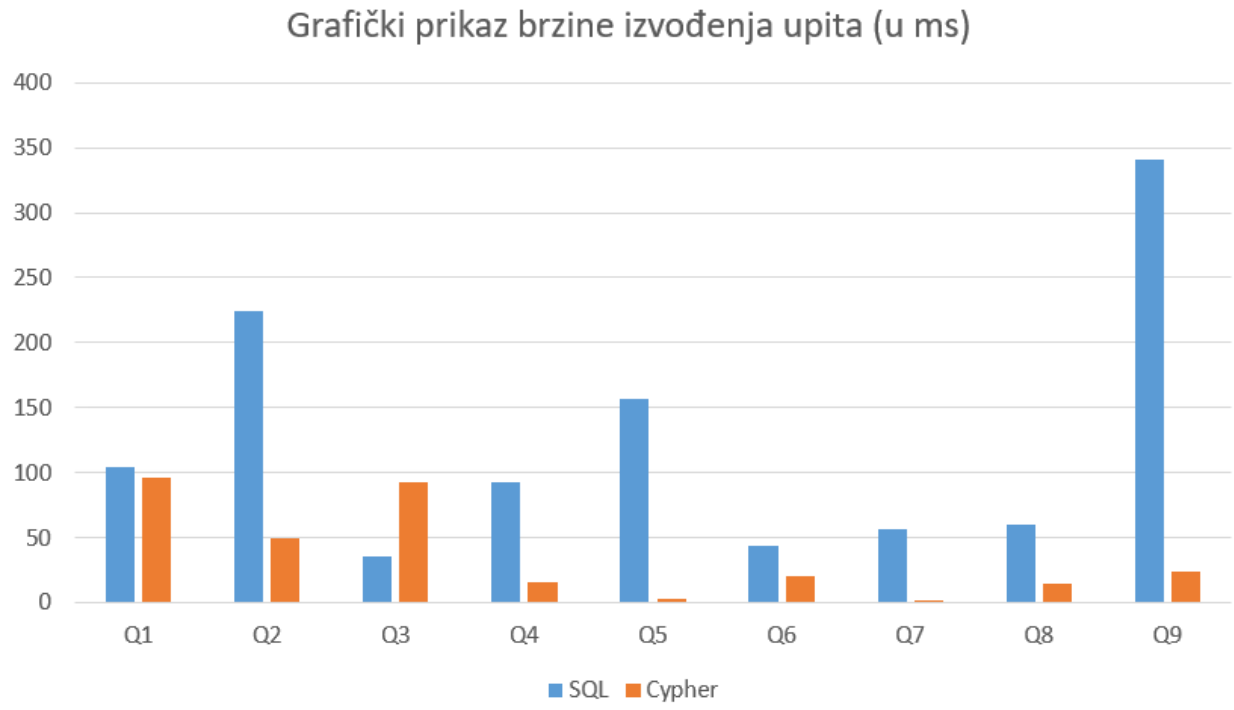
Tablica 5.3. – Sintaksa i vremena izvršavanja pojedinih upita u SQL i Cypher jeziku

Oznaka upita	**Upit u SQL jeziku**	**Upit u Cypher jeziku**
Q1	SELECT * FROM Products;	MATCH (p:Product) RETURN p;
	Vrijeme izvršavanja: 104 ms	Vrijeme izvršavanja: 96 ms
Q2	INSERT INTO Products VALUES ('Proizvod', '11','3', '300', '30.00', '10', '0','0','0');	CREATE (p:Product{productName:"Proizvod", supplierID:3,categoryID:3, quantityPerUnit:300, unitPrice:30.00,unitsInStock:10, unitsOnOrder:0,reorderLevel:0, discontinued:0})
	Vrijeme izvršavanja: 224 ms	Vrijeme izvršavanja: 49 ms
Q3	UPDATE Customers SET PostalCode = '31000' WHERE City = 'Berlin';	MATCH (c {city:'Berlin'}) SET c.postalCode='31000' RETURN c.city;
	Vrijeme izvršavanja: 36 ms	Vrijeme izvršavanja: 92 ms
Q4	DELETE FROM [Order Details] WHERE Quantity > 50;	MATCH (o:Orders{freight:100}) DELETE o;
	Vrijeme izvršavanja: 92 ms	Vrijeme izvršavanja: 16 ms

Q5	SELECT * FROM Products WHERE CategoryID = 8;	MATCH (p:Product {categoryID:8}) RETURN p;
	Vrijeme izvršavanja: 157 ms	Vrijeme izvršavanja: 3 ms
Q6	ALTER TABLE Customers ADD email VARCHAR (50);	MATCH (c { fax: '030-0076545' }) SET c.email = 'abc@degfh.com' RETURN c;
	Vrijeme izvršavanja: 44 ms	Vrijeme izvršavanja: 20ms
Q7	SELECT COUNT(City) FROM Customers;	MATCH (c:Customer) RETURN COUNT (c.city);
	Vrijeme izvršavanja: 56 ms	Vrijeme izvršavanja: 2 ms
Q8	SELECT AVG(UnitPrice) FROM Products	MATCH (p:Product) RETURN AVG (p.unitPrice);
	Vrijeme izvršavanja: 60 ms	Vrijeme izvršavanja: 15 ms
Q9	SELECT Orders.OrderID, Customers.CustomerID, Orders.OrderDate FROM Orders INNER JOIN Customers ON Orders.CustomerID=Customers.CustomerID;	MATCH (c:Customer {customerID:"ALFKI"}) OPTIONAL MATCH (c)--> (x) RETURN (x);
	Vrijeme izvršavanja: 341 ms	Vrijeme izvršavanja: 24 ms

Pogledom u tablicu 5.3. može se zaključiti da je sintaksa upita u SQL i Cypher jezicima uvelike slična. Razlog tome je taj što je Cypher nastao po uzoru na SQL jezik. Također, sama vremena izvođenja pojedinih upita variraju u velikoj mjeri. Ipak, vremena izvođenja upita treba uzeti s rezervom, budući da je analiza izvođena na prijenosnome računalu, a ne na fizičkome serveru namijenjenome samo za bazu podataka. To znači da brzina izvođenja upita ovisi i o drugim pozadinskim procesima pokrenutima na prijenosnom računalu u trenutku mjerenja vremena. Jasnija slika o rezultatima usporedbe brzina izvođenja pojedinih upita se može vidjeti prema slici 5.3. na kojoj su vremena izvođenja dana u obliku dijagrama. Prema slici 5.3. se može zaključiti da se SQL pokazao brži jedino kod ažuriranja podataka, dok je najveća razlika u vremenu kod

spajanja tablica i ekvivalentnog upita u Cypher jeziku koji se realizira pomoću naredbe OPTIONAL MATCH.



Sl. 5.3. – Grafički prikaz brzina izvođenja upita (brzine su iskazane u milisekundama)

Budući da graf baze podataka omogućuju i direktnu pohranu veza u strukturu grafa, odnosno u graf bazu podataka, u nastavku je dan jedan jednostavan primjer (Programski kod 5.1.) kreiranja veze, tj. Cypher upit pomoću kojeg se može kreirati veza sa svojstvima između dva čvora.

Programski kod 5.1. – Sintaksa upita za kreiranje veze u Cypher jeziku

```
MATCH (cust:Customer), (cc:CreditCard)
CREATE (cust) – [r: DO_SHOPPING_WITH {shop_date: „12/12/2014“, price: 5500}] -> (cc)
RETURN r
```

Iz primjera 5.1. se vide neki neuobičajeni znakovi kod pisanja upita. Prvenstveno se misli na znak ->. Naime, rečeno je da veza u graf bazi podataka ima usmjerenje, a upravo taj znak definira usmjerenje veze. Konkretno, u ovome primjeru početni (izvorišni) čvor veze je čvor *Customer* s oznakom *cust*, a završi (odredišni) čvor je čvor *CreditCard* s oznakom *cc*. Dakle, veza je usmjerena od čvora *Customer* prema čvoru *CreditCard*. Isto kao čvor, i veza ima svoje ime i oznaku koji se pišu unutar uglatih zagrada. U primjeru iznad, oznaka veze je *r*, a naziv veze je *DO_SHOPPING_WITH*. Svojstva veze se definiraju neposredno nakon imena veze, unutar

vitičastih zagrada i to na način da se navede naziv svojstva a zatim vrijednost toga svojstva. Naziv svojstva i vrijednost svojstva se odvajaju dvotočkom, a svojstva se međusobno odvajaju zarezom.

6. ZAKLJUČAK

Zadatak rada je usporediti graf baze podataka s relacijskim bazama podataka. Prilikom usporedbe nastojalo se obuhvatiti što više aspekata iz područja baza podataka kako bi usporedba bila opširnija i detaljnija. Opširnija i detaljnija usporedba za rezultat ima jasnije i bolje sagledavanje i razumijevanje problema. Ipak, poseban naglasak je stavljen na brzinu izvođenja pojedinih upita u relacijskoj bazi podataka koristeći SQL jezik i njima ekvivalentnih upita u graf bazi podataka koristeći Cypher jezik.

Prva vidljiva razlika između relacijskih i graf baza podataka je već na samome početku, odnosno u procesu modeliranja baze podataka. U oba slučaja podloga za izradu modela je preuzeta iz matematike, odnosno relacijske baze podataka se temelje na relacijskoj algebri, a graf baze podataka se temelje na teoriji grafova. Relacijski model baze podataka zahtijeva prvo izradu konceptualnog modela. Nakon što je konceptualni model izrađen, njega je potrebno prevesti u relacije koristeći definirana pravila. Također, relacije je potrebno normalizirati, a to zahtjeva još dodatnog posla prije same implementacije baze podataka.

Nasuprot modeliranju relacijskog modela, proces modeliranja graf baze podataka zahtijeva nešto manje posla. Prvenstveno se to očituje pri izradu modela koji će poslužiti za kasniju implementaciju graf baze podataka. Viđeno je da kod graf baza podataka proces modeliranja ne zahtjeva izradu konceptualnog modela pa njegovo prevođenje u logički model. Kod graf baza podataka model grafa koji se nacrta na papiru (ili u nekom od alata za modeliranje i crtanje) je dovoljan za početak implementacije te graf baze podataka. Točnije rečeno, model kakav je nacrtan odgovara shemi u kojoj su pohranjeni podaci u bazi podataka. Isto tako, dodavanje novih objekata u graf bazu podataka je znatno lakše nego kod relacijskih baza podataka jer ne zahtijeva izmjenu modela. Samim time, graf baze podataka su puno fleksibilnije po tome pitanju od relacijskih baza podataka.

Nadalje, razlike su vidljive i u upitnim jezicima sa kojim se manipulira podacima koji su pohranjeni u bazi podataka. Kod relacijskih baza podataka koristi se standardizirani SQL jezik koji je podržan od strane svih proizvođača sustava za upravljanje relacijskom bazom podataka. Malo drugačija priča je kod graf baza podataka. Naime, upitni jezik koji se koristi u sustavu za upravljanje graf bazom podataka još uvijek uvelike ovisi od proizvođača, premda ne postoji standard u tome polju. To ne znači da u budućnosti neće biti toga standarda jer se ulažu znatni napori od strane Neo4j kompanije kako bi Cypher postao standardni jezik za graf baze podataka. Upravo spomenuti Cypher jezik je obrađen u ovome radu iz razloga što je ona upitni jezik koji je

podržan u Neo4j sustavu za upravljanje graf bazom podataka. Također, vidljiva je sličnost između Cyphera i SQL jezika. Pored toga što su oba jezika deklarativni, veliku sličnost dijele u nazivima samih naredbi, pa donekle i u sintaksi tih naredbi. Ono što je glavna razlika između ova dva jezika je što Cypher radi na malo drugačiji način od SQL. Točnije rečeno, Cypher jezik radi na principu pronalaženja uzorka u grafu.

Kako je ranije rečeno, naglasak pri ovoj usporedbi je stavljen na brzinu izvođenja upita. Razlog zbog kojeg je naglasak stavljen upravo na ovaj aspekt je taj što je današnje društvo vrlo dinamično, što znači da se podaci generiraju brže i više nego ikada u prošlosti, a pri tome se jako razlikuju u formatima. Iz prethodnih rečenica se može zaključiti da je vrijeme jako bitan faktor, pa tako i vrijeme izvođenja upita znatno može olakšati izbor koju vrstu baze podataka koristiti. Također, prema [29] se može zaključit da graf baze podataka pokazuju znatnu prednost na ovome polju u odnosnu na relacijske. Ta prednost sve više i više dolazi do izražaja što je skup podataka veći.

Usporedba u ovome radu se temeljila na mjerenju vremena izvršavanja pojedinih upita. Analizom rezultata te usporedbe se jasno da zaključiti kako je Cypher jezik znatno brži od SQL jezika, te kao takav ima prednost nad SQL jezikom. U ovoj usporedbi jedini upit u SQL jeziku koji se kraće izvršavao nego u Cypher jeziku je ažuriranje podataka. Pri tome treba imati na umu da su Neo4j, pa tako i Cypher jezik građeni za rad sa veoma velikim skupovima podataka, te da je u ovoj usporedbi korišten veći skup podataka za očekivati bi bilo da bi Cypher jezik i tu imao bolje rezultate. Također, da je korišten veći skup podataka, razlike u vremenima izvođenja bi bile još i veće u korist Cypher jer bi se onda donekle ostvario cjelokupni potencijal koji nudi Neo4j graf baza podataka i Cypher jezik.

Iz svega gore navedenoga, može se zaključit kako graf baze podataka imaju jako puno prostora za napredak i za očekivati je da će one u budućnosti igrati jako važnu ulogu na tržištu. Premda su još uvijek relacijske baze podataka dominantne na tržištu, njihov udio na tržištu bi se mogao smanjiti u budućnosti. Vjerojatno jedan od glavnih razloga zbog kojih relacijske baze podataka još uvijek dominiraju tržištem se može naći u činjenici da su sustavi velikih kompanija rađeni u dobu kada NoSQL, pa tako i graf baze podataka nisu ni postojale ili su još uvijek bile relativno neistražene tehnologije. Budući da ti sustavi sadrže jako puno podataka, od kojih su neki vrlo povjerljivi i važni za poslovanje organizacije, prelazak sa relacijskih baza podataka na npr. graf baze podataka i nije tako trivijalan i jednostavan zadatak i upravo zbog toga relacijske baze podataka su još uvijek dominantne na tržištu, iako za određene slučajeve i nisu najbolji izbor.

LITERATURA

[1] C. McLellan, The Internet of things and big dana: Unlocking the power, ZDNet, 2015., dostupno na: https://www.zdnet.com/article/the-internet-of-things-and-big-data-unlocking-the-power/ [14.5.2018.]

[2] J.Hurwitz, A.Nugent, dr.F.Halper, M.Kaufman, Big Data For Dummies, John Wiley & Sons, Inc., Hoboken, New York, 2013.

[3] J.R. Kalyvas, M.R. Overly, Big Data – A Business and Legal Guide, CRC Press, Boca Raton, Florida, 2014.

[4] F. Iafrate, From Big Data to Smart Data – Volume 1, John Wiley & Sons, Inc., Hoboken, New York, 2015.

[5] J. Gantz, D. Reinsel, The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far Fast, EMC Corporation, 2012., dostupno na: https://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf [18.5.2018.]

[6] M. Varga, Baze podataka, Društvo za razvoj informacijske pismenosti, Zagreb, 1994.

[7] V. Blagojević, Relacione baze podataka, ICNT, Beograd, 2006.

[8] M. Veinović, G. Šimić, A. Jevremović, I. Franc, Baze podataka, Univerzitet Singidunum, Beograd, 2013.

[9] T. Teorey, S. Lightstone, T. Nadeau, Database Modeling & Design, Morgan Kaufman Publishers, San Francisco, 2006.

[10] R. Manger, Osnove projektiranja baza podataka – Priručnik za polaznike, SRCE, Zagreb, 2010.

[11] R. Manger, Baze podataka – skripta, Prirodoslovno – matematički fakultet Zagreb, Zagreb, 2003.

[12] A. G. Taylor, SQL For Dummies, John Wiley & Sons, Inc., Hoboken, New York, 2013.

[13] G. Vaish, Getting Started with NoSQL, Packt Publishing Ltd., Birmingham, 2013.

[14] M. Šestak, Graf baze podataka – završni rad, Fakultet organizacije i informatike, Varaždin, 2014.

[15] N. Tomić, NoSQL tehnologije i primjene – Diplomski rad, Prirodoslovno – matematički fakultet, Zagreb, 2016.

[16] J. Celko, Joe Celko's Complete Guide To NoSQL, Morgan Kaufman Publishers, Waltham, 2014.

[17] D. B. West, Intoduction to Graph Theory, Pearson Education Inc., 2001.

[18] Osnovni pojmovi teorije grafova, Fakultet elektrotehnike i računarstva, Zagreb, dostupno na: https://www.fer.unizg.hr/_download/repository/Osnovni_pojmovi-teorija_grafova.pdf [5.6.2018.]

[19] I. Robinson, J. Webber, E. Eifrem, Graph Databases, O'Reilly Media Inc., Sebastopol, 2015.

[20] What is a Graph Database?, Neo4j, Inc., dostupno na: https://neo4j.com/developer/graph-database/#property-graph [8.6.2018.]

[21] Graph Dana Modeling Guidelines, Neo4j, Inc., dostupno na: https://neo4j.com/developer/guide-data-modeling/ [8.6.2018.]

[22] O. Panzarino, Learning Cypher, Packt Publishing Ltd., Birmingham, 2014.

[23] DB – Engines Ranking, DB – Engines, dostupno na: https://db-engines.com/en/ [11.6.2018.]

[24] Operations Manual, Neo4j, Inc., dostupno na: https://neo4j.com/docs/operations-manual/3.4/introduction/ [13.6.2018.]

[25] B.M. Sasaki, Graphdatabases for Beginners: ACID vs. BASE Explained, Neo4j, Inc., 2015., dostupno na: https://neo4j.com/blog/acid-vs-base-consistency-models-explained/ [14.6.2018.]

[26] M. Chapple, Abandoning ACID in Favor of BASE in Database Engineering, Lifewire,2017., dostupno na: https://www.lifewire.com/abandoning-acid-in-favor-of-base-1019674 [14.6.2018.]

[27] Neo4j Cypher Refcard 3.4, Neo4j, Inc., dostupno na: https://neo4j.com/docs/cypher-refcard/current/ [15.6.2018.]

[28] Neo4j vs. RDBMS and other NoSQL, Neo4j, Inc., dostupno na: https://neo4j.com/product/#comparison [18.6.2018.]

[29] S. Batra, C. Tyagi, Comparative Analysis of Relational And Graph Databases, International Journal od Soft Computing and Engineering, Volume 2, str. 509 – 512, svibanj 2012.

[30] https://code.google.com/archive/p/northwindextended/downloads [20.6.2018]

[31] Tutorial: Import Dana Into Neo4j, Neo4j, Inc., dostupno na: https://neo4j.com/developer/guide-importing-data-and-etl/ [21.6.2018.]

SAŽETAK

U svim ljudskim djelatnostima je potrebno voditi određenu vrstu evidencije. Razvoj informacijskih tehnologija je taj posao znatno olakšao. Olakšanje ju prvenstveno u vidu baza podataka u koje se pohranjuju potrebni podaci zbog daljnje upotrebe. Upotreba baza podataka kao primarnog mehanizma za vođenje evidencije i pohrane podataka u poslovanju je postala stvarna potreba suvremenog društva. Također, upotreba baza podataka za pohranu podataka svakim danom bilježi sve veći i veći trend porasta. Donedavno, standard u pohrani podataka su bile relacijske baze podataka. U zadnjih par godina svoj procvat upotrebe doživljavaju i baze podataka koje se ne temelje na relacijskome modelu – nerelacijske ili NoSQL baze podataka. Pojam NoSQL baza podataka obuhvaća više vrsta baza podataka: ključ–vrijednost baze podataka, dokument baze podataka, stupčaste baze podataka i graf baze podataka. U ovome radu, pored relacijskih baza podataka detaljnije su obrađene samo graf baze podataka, odnosno napravljena je njihova međusobna usporedba. Usporedba obuhvaća proces modeliranja baze podataka, sustav za upravljanje bazom podataka, upitne jezike koji se koriste kod relacijskog i kod graf modela baze podataka. Isto tako, prikazane su glavne sličnosti i razlike između relacijskog i graf modela baze podataka, kao i primjeri upita u SQL i Cypher jeziku, te njihova vremena izvršavanja.

Ključne riječi: usporedba, SQL, relacijska baza podataka, nerelacijska baza podataka, NoSQL, graf baza podataka, Neo4j, Cypher, upit, sustav za upravljanje bazom podataka,

ABSTRACT

In all human activities it is necessary to keep a certain kind of records. Development of information technologies has made that job much easier. Relief is primarily in the database in which data is stored for future use. Using database as a primary mechanism for keeping record and storing data in business has become real need of modern society. In addition usage of the database for data storage grows higher every day. Until recently, standard in data storage was relational databases. In the last few years, databases which are not based on relational model, non–relational or NoSQL databases started booming in its use. The term, NoSQL databases includes more types of databases: key–value databases, document databases, column–oriented databases and graph databases. In this thesis, along relational databases, only graph databases are addressed with more details, that is, ther compared between themselves. Comparison includes the process of modelling the database, database management system, query language which are used for relational and graph databases. Also, main similarities and differences between relational and graph model databases are displayed, as are examples of the query in SQL and Cypher language, as well as needed execution time.

Keywords: comparison, SQL, relational database, non–relational database, NoSQL, graph database, Neo4j, Cypher, query, database management system

ŽIVOTOPIS

Marko Brica, rođen 25. 12. 1992. godine u Slavonskom Brodu. Osnovnu školu završava u Odžaku (Bosna i Hercegovina) u Osnovnoj školi Vladimira Nazora 2007. godine. Nakon završene osnovne škole upisuje Srednju školu Pero Zečević u Odžaku i završava četverogodišnje školovanje 2011. godine, čime stječe zvanje Tehničar za računarstvo. Po završetku srednjoškolskog obrazovanja upisuje preddiplomski stručni studij elektrotehnike, smjer informatika na Elektrotehničkom fakultetu (danas Fakultet elektrotehnike, računarstva i informacijskih tehnologija) u Osijeku kojeg uspješno završava u rujnu 2014. godine i stječe akademski naziv stručni prvostupnik inženjer elektrotehnike (bacc. ing. el.). Na istome Fakultetu upisuje Sveučilišni diplomski studij računarstva, modul Programsko inženjerstvo 2015. godine.

U Osijeku, 30. lipnja 2018.

_____

Marko Brica