Sustavi za upravljanje velikom količinom podataka s posebnim osvrtom na MongoDB

Kaleb, Luka

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET

DIPLOMSKI RAD

Sustavi za upravljanje velikom količinom podataka s posebnim osvrtom na MongoDB

Luka Kaleb

Split, srpanj 2022.

IZJAVA

kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam diplomski rad s naslovom SUSTAVI ZA UPRAVLJANJE VELIKOM KOLIČINOM PODATAKA S POSEBNIM OSVRTOM NA MONGODB izradio samostalno pod voditeljstvom prof. dr. sc. Marka Rosića. U radu sam primijenio metodologiju znanstvenoistraživačkog rada i koristio literaturu koja je navedena na kraju diplomskog rada. Tuđe spoznaje, stavove, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo u diplomskom radu na uobičajen, standardan način, citirao sam i povezao s fusnotama s korištenim bibliografskim jedinicama. Rad je pisan u duhu hrvatskog jezika.

Student

Luka Kaleb

Temeljna dokumentacijska kartica

Diplomski rad

Sveučilište u Splitu

Prirodoslovno-matematički fakultet

Odjel za informatiku

Ruđera Boškovića 33, 21000 Split, Hrvatska

SUSTAVI ZA UPRAVLJANJE VELIKOM KOLIČINOM PODATAKA S POSEBNIM OSVRTOM NA MONGODB

Luka Kaleb

SAŽETAK

Velike podatke karakterizira iznimno veliki volumen u bajtovima, širok izbor tipova podataka i iznimno velika brzina kojom se podaci generiraju, prikupljaju i obrađuju. Bezbroj je izvora iz kojih dolaze veliki podaci. To mogu biti mobilne aplikacije, društvene mreže, elektronička pošta, baze podataka klijenata, itd. Veliki podaci postali su bitan dio industrije mobilnih aplikacija, zbog svoje jednostavne izvedbe i naprednih značajki. Baza podataka zajednička je zbirka srodnih podataka. Dijelimo ih na relacijske i nerelacijske baze podataka. Relacijske ili SQL baze podataka se temelje na relacijskom modelu podataka koji je predstavio E. F. Codd 1970. godine. Transakcije relacijskih baza podataka slijede svojstva atomarnosti, konzistentnosti, izoliranosti i trajnosti kako bi se nakon i prije transakcije održala dosljednost u bazi podataka. Nerelacijske ili NoSQL baze podataka slijede BASE model transakcija s primarnim ciljem raspoloživosti podataka, uz prihvatljivu nedosljednost podataka dok se ne završe replikacije putem mreže. One su skalabilne i prilagodljive te su zato izvrsne za aplikacije s velikim skupovima podataka. One imaju jednostavnu i fleksibilnu strukturu te daju prednost performansama i skalabilnosti nasuprot dosljednosti. MongoDB je nerelacijska baza podataka čija je struktura savršeno prikladna za velike količine podataka.

Ključne riječi: veliki podaci, mobilne aplikacije, baza podataka, relacijska baza podataka, nerelacijska baza podataka, ACID svojstva, BASE model, MongoDB

Rad je pohranjen u knjižnici Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu

Rad sadrži: 64 stranica, 59 grafičkih prikaza, 0 tablica i 36 literaturnih navoda. Izvornik je na hrvatskom jeziku.

Mentor: prof. dr. sc. Marko Rosić, Prirodoslovno-matematički fakultet Sveučilišta u Splitu

Ocjenjivači: prof. dr. sc. Marko Rosić, Prirodoslovno-matematički fakultet Sveučilišta u Splitu

doc. dr. sc. Divna Krpan, Prirodoslovno-matematički fakultet Sveučilišta u Splitu

dr. sc. Tonći Dadić, v. pred, Prirodoslovno-matematički fakultet Sveučilišta u Splitu

Basic documentation card

Thesis

University of Split

Faculty of Science

Department of computer science

Ruđera Boškovića 33, 21000 Split, Hrvatska

BIG DATA MANAGEMENT SYSTEMS WITH A SPECIAL FOCUS ON MONGODB

Luka Kaleb

ABSTRACT

Big data is characterized by an extremely large volumen in bytes, a wide range of data types and an extremely high velocity at which data is generated, collected and processed. There are countless sources data comes from. These can be mobile applications, social media, e-mails, customer databases. Big data has become an essential part of the mobile application industry, due to its simple performance and advanced features. A database is a shared collection of related data. We divide them into relational and non-relational databases. Relational or SQL databases are based on the relational data model presented by E. F. Codd in 1970. Relational database transactions follow the properties of atomacity, consistency, isolation and durability in order to maintan consistency in the database after and before the transaction. Non-relational or NoSQL databases follow the BASE model with the primary goal of data availability, with acceptable data inconsistency untill network replication is complete. They are scalable and are therefore great for applications with large data sets. They have a simple and flexible structure. They prioritize performance and scalability for consistency. MongoDB is a non-relational database whose structure is perfectly suitable for large amounts of data.

Keywords: big data, mobile applications, databases, relational database, non-relational database, ACID properties, MongoDB

Thesis deposited in library of Faculty of science, University of Split.

Thesis consist of: 64 pages, 59 figures, 0 tables and 36 references

Original language: Croatian

Supervisor: Marko Rosić, Ph.D. Associate Professor of Faculty of Science, University of Split

Reviewers: Marko Rosić, Ph.D. Full professor tenure of Faculty of Science, University of Split

Divna Krpan, Ph.D. Assistent Professor of Faculty of Science, University of Split

Tonći Dadić, Ph. D. Senior Lecturer of Faculty of Science, University of Split

Sadržaj

Uvod

Da bi obavili svakodnevne poslove, ljudi sve više koriste mobilne aplikacije. U današnje vrijeme gotovo sve mobilne aplikacije imaju potrebu za pohranom velike količine podataka. Potreban je jedinstveni alat za upravljanje podacima za učinkovitu procjenu i obradu podataka u mobilnim aplikacijama. Korisnici bi, bez mogućnosti pohranjivanja podataka, trebali unijeti sve potrebne podatke prije nego što se proizvede koristan rezultat ili se izvrši bilo kakva obrada. Ako informacijski sustavi ne bi mogli pohranjivati podatke kao dio sustava, ne bi mogli funkcionirati. Obično pametni telefoni i tableti, odnosno uređaji na kojima rade mobilne aplikacije, imaju ograničen kapacitet pohrane. Pojava baza podataka i napredak u tehnologiji omogućili su pohranu velikih količina podataka na mobilne uređaje. U ovom radu opisani su sustavi za upravljanje velikom količinom podataka. Poseban osvrt odnosi se na MongoDB.

U prvom poglavlju opisano je kako su se razvijali pametni telefoni te se donosi osvrt na mobilne aplikacije, njihov razvoj, podjelu i način pohrane podataka.

Drugo poglavlje odnosi se na velike podatke i način na koji se prikupljaju i koriste.

Treće poglavlje donosi usporedbu relacijskih i nerelacijskih baza podataka te njihove prednosti i nedostatke.

U četvrtom poglavlju opisan je MongoDB koji pripada nerelacijskim bazama podataka te je napravljena usporedba s MySQL-om, relacijskom bazom podataka.

U petom poglavlju prikazan je praktični dio diplomskog rada. Aplikacija se zove NBA Stats jer prikazuje statistiku NBA igrača u sezoni 2021./2022. Praktični dio prikazuje primjenu MongoDB-a u aplikaciji.

1. Mobilne aplikacije

Pametni telefon (engl. smartphone) prijenosni je uređaj s računarskom platformom koji kombinira funkcije računala i mobilnog telefona u jednu jedinicu (Wikipedia, 2022). Izraz „pametni“ odnosi se na mogućnost uporabe mobitela kao džepnog računala (Wikipedia, 2022). Osnovna razlika između mobitela i pametnog telefona je u tome što se mobiteli koriste za upućivanje poziva i primanje poruka putem radio frekvencijskih veza. S druge strane, pametni telefoni su prijenosni uređaji koji imaju sve funkcije računala, kao i zaslon osjetljiv na dodir, pristup Internetu i operacijski sustav (Ask any difference, 2022). Tipični pametni telefon ima zaslon na dodir visoke rezolucije. Pametni telefoni podržavaju bežične komunikacijske protokole kao što su Bluetooth, Wi-Fi i satelitska navigacija.

Rani pametni telefoni primarno su plasirani na poslovno tržište. Bili su ograničeni svojim glomaznim oblikom, kratkim vijekom trajanja baterije, sporim analognim mobilnim mrežama i nezrelošću bežičnih usluga. Od 2000. do 2009. godine mobiteli koji su često sadržavali QWERTY tipkovnicu ili zaslon osjetljiv na dodir, imali pristup elektroničkoj pošti i bežičnom Internetu, počeli su osvajati tržište.

Nekoliko ključnih tehnoloških napredaka omogućilo je razvoj pametnih telefona. Eksponencijalni rast i smanjivanje MOS tranzistora do mikro razina tijekom 1990-ih i 2000-ih (kao što je predviđeno Mooreovim zakonom) omogućilo je izgradnju pametnih uređaja poput pametnih telefona, kao i prijelaz s analognih na brže digitalne bežične mobilne mreže (Edholmov zakon). Druga dva važna čimbenika koja su omogućila razvoj pametnih telefona su litij-ionska baterija, nezamjenjiv izvor energije koji omogućuje dugo trajanje baterije i razvoj zrelijih programskih platformi. Izumljena je 1980-ih, a komercijalizirana 1991 (Wikipedia, 2022).

Slika 1 − Pametni telefon (Pexels, 2022)

Ljudi preuzimaju više mobilnih aplikacija zahvaljujući poboljšanim mogućnostima pametnih telefona. Njih mogu upotrebljavati korisnici pametnih telefona. Mobilna aplikacija je računalni program dizajniran za rad na mobilnom uređaju kao što je telefon, tablet ili sat (Wikipedia, 2022). One su često u suprotnosti s aplikacijama za stolna računala koje su dizajnirane za rad na stolnim računalima i mrežnim aplikacijama koje se pokreću u mobilnim mrežnim (engl. web) preglednicima, a ne izravno na mobilnom uređaju. Aplikacije su izvorno bile namijenjene za pomoć kod elektroničke pošte, kalendara i baze podataka kontakata, ali je javna potražnja za njima prouzročila brzo širenje na druga područja, tako da sada postoje milijuni dostupnih aplikacija (Wikipedia, 2022) (Wikipedia, 2021). Razvoj mobilnih aplikacija u stalnom je porastu u prihodima i otvaranju radnih mjesta (Wikipedia, 2022). Tržište mobilnih aplikacija je tijekom 2010. godine generiralo 5,2 milijarde dolara prometa (Wikipedia, 2021). Izraz „aplikacija“, skraćenica za „softverska aplikacija“ je 2010. godine proglašena za riječ godine od strane Američkog dijalektološkog društva (Wikipedia, 2022).

Većina mobilnih uređaja prodaje se s nekoliko aplikacija u paketu s unaprijed instaliranom programskom podrškom (engl. software) kao što su mrežni preglednik, klijent elektroničke pošte, kalendar i aplikacija za kupnju glazbe ili drugih medija. Neke unaprijed instalirane aplikacije mogu se otkloniti običnim postupkom deinstaliranja, ostavljajući tako više prostora za pohranu željenih aplikacija. Aplikacije koje nisu unaprijed instalirane obično su

dostupne putem distribucijskih platformi koje se nazivaju trgovine aplikacija (engl. app stores). Njima mogu upravljati vlasnici operacijskih sustava mobilnih telefona, kao što je App Store (iOS) ili Google Play; proizvođači uređaja kao što su Galaxy Store i Huawei AppGallery; ili trećih strana, kao što su Amazon AppStore i F-Droid. Neke aplikacije su besplatne kod preuzimanja, a neke se moraju platiti. Do drugog tromjesečja 2015. godine samo Google Play i Apple trgovine pojedinačno su ostvarile zaradu od 5 milijardi dolara (Wikipedia, 2022).

Kompanija Apptopia objavila je listu najpopularnijih aplikacija temeljenu na broju preuzimanja iz Google Play i iOS trgovina aplikacija u 2021. godini. Drugu godinu zaredom na samom vrhu nalazi se TikTok. Slijede ga Instagram, Facebook i Whatsapp, a top pet najpopularnijih zaključuje aplikacija za dopisivanje koja je prethodne godine dobila na popularnosti – Telegram. Među deset najpopularnijih još su Snapchat, Zoom, Messenger, CapCut i Spotify. Kategoriju najpopularnijih zabavnih aplikacija, očekivano, predvodi Netflix, a slijede ga Youtube, Google Play Games, Disney + i Amazon Prime Video. Među poslovnim aplikacijama jasno je vidljiv trend „rada od kuće“. Tako je najpopularnija aplikacija Zoom, a slijede ju Google Meet, Whatsapp Business, Microsoft Teams i Adobe Acrobat Reader (Čizmić, 2021).

1.1. Razvoj mobilnih aplikacija

Da bismo razumjeli mobilne aplikacije, moramo znati kako doseći korisnike i koje su specifičnosti mobilnih uređaja (Mladenović, Mladenović, & Zaharija, 2018). Razvijanje aplikacija za mobilne uređaje zahtijeva razmatranje ograničenja i značajki tih uređaja. Mobilni uređaji rade na bateriju i imaju manje snažne procesore od osobnih računala, ali imaju više značajki kao što su otkrivanje lokacije i kamere (Wikipedia, 2022). Lokacija kod mobilnih telefona je poznata i vrlo precizna, dok je kod mrežnih aplikacija poznata, ali je približna (Mladenović, Mladenović, & Zaharija, 2018).

Slika 2 − Lokacija kod mobilnih i mrežnih aplikacija (Mladenović, Mladenović, & Zaharija, 2018)

Programeri moraju uzeti u obzir širok raspon veličina zaslona, specifikacija strojne opreme (engl. hardware specifications) i konfiguracija zbog intenzivne konkurencije u mobilnoj programskoj podršci (engl. mobile software) (Wikipedia, 2022).

Možda i najvažniji čimbenik za mobilnu aplikaciju je kompatibilnost uređaja. Također, vrlo je važan i izbor operacijskog sustava. Potrebno je osigurati da aplikacija radi na pametnom telefonu, tabletu i drugim uređajima (Apogaeis, n.d.).

Izazovi izgradnje mobilnih aplikacija:

1) fragmentacija operacijskih sustava
2) višestrukost proizvoda i razvojnih timova
3) konzistentno iskustvo korisnika
4) fragmentacija značajki. (Wikipedia, 2022)

Programiranje mobilnih aplikacija je barem podjednako teško kao i svako programiranje, unatoč tome što aplikacije izgledaju jednostavno (Mladenović, Mladenović, & Zaharija, 2018).

1.2. Pohrana podataka

Kritična komponenta svake aplikacije su pohranjeni podaci. Posebna razmatranja zahtijeva komponenta pohranjenih podataka mobilnih aplikacija. Korisnici bi, bez mogućnosti pohranjivanja podataka, trebali unijeti sve potrebne podatke prije nego što se proizvede koristan rezultat ili se izvrši bilo kakva obrada. Ako informacijski sustavi ne bi mogli pohranjivati podatke kao dio sustava, ne bi mogli funkcionirati. Obično pametni telefoni i tableti, odnosno uređaji na kojima rade mobilne aplikacije, imaju ograničen kapacitet pohrane.

Ako aplikacija mora pohraniti sve potrebne podatke na uređaj, tj. ako je aplikacija dizajnirana za korištenje samo izvan mreže, onda imamo ograničenu količinu podataka koja se može pohraniti. Osim u slučajevima kada se podaci generiraju na uređaju, vjerojatno će biti zastarjeli jer se ne ažuriraju dinamički iz vanjskih izvora kako se uvjeti mijenjaju. Nasuprot tome, kod aplikacija dizajniranih za rad na mreži, ograničenje pohrane podataka na uređaju ne utječe na aplikaciju jer se potrebnim podacima može pristupiti u stvarnom vremenu s poslužitelja. Aplikacija tada ovisi o bežičnoj vezi koja nije uvijek dostupna. Aplikacije koje sinkroniziraju podatke imaju vlastite izazove.

Različiti pristupi pohrani podataka u mobilnim aplikacija mogu imati ozbiljan utjecaj na korisnika. Korisnik, uz neke aplikacije poput osobnog popisa kontakata, ažurira pohranjene podatke i čuva ih na uređaju. Potencijalni problemi su što u mnogim situacijama podatke ažuriraju vanjski entiteti, a to zahtijeva pristup poslužitelju. Ponekad će se za ažuriranje podataka na poslužitelju koristiti čak i lokalno pohranjeni podaci kako bi korisnik mogao dijeliti podatke između nekoliko osobnih uređaja.

Od pozadinskih sustava koje koriste samo zaposlenici, kao što su mobilni sustavi za upravljanje zalihama do aplikacija usmjerenih na korisnike, kao što su one u mobilnoj trgovini, u mobilnom poslovanju aplikacije se koriste u različite svrhe. Pristup pohranjivanja podataka može se razlikovati od aplikacije do aplikacije, ali sve te aplikacije, bilo da ih isporučuje tvrtka ili zaposlenici, trebaju pohranjene podatke. Dizajn komponente pohranjenih podataka postaje sve važniji zato što su aplikacije za mobilne uređaje postale sve prisutnije, i za osobnu i za poslovnu uporabu. Aplikacije koje ne zadovoljavaju potrebe korisnika i tvrtki za podacima, oni neće prihvatiti. Naklonost među korisnicima i poduzećima naći će samo dobro osmišljene aplikacije koje pružaju prave podatke u pravo vrijeme.

S obzirom na pohranjenu komponentu podataka mobilne aplikacije dolaze u tri varijante:

1. Izvanmrežne aplikacije (engl. Offline apps) pohranjuju podatke na mobilnom uređaju. Puna funkcionalnost ovih aplikacija dostupna je izvan mreže i ne moraju biti na mreži da bi funkcionirale, osim za početnu instalaciju.
2. Mrežne aplikacije (engl. Online apps) pohranjene podatke dohvaćaju s poslužitelja, tako da ovise o pristupu poslužitelju. Aplikacija se za svoju funkcionalnost oslanja na podatke pohranjene na poslužitelju, iako se neki podaci mogu pohraniti na mobilnom uređaju. U ovu kategoriju spadaju aplikacije koje se koriste za e-trgovinu. Puna funkcionalnost ovih aplikacija dostupna je samo kada su na mreži; moraju biti na mreži da bi funkcionirale.
3. Sinkronizirane aplikacije (engl. Synchronized apps) pohranjuju sve svoje podatke na mobilnom uređaju, ali pohranjeni podaci mogu se ažurirati podacima s poslužitelja kada je uređaj povremeno na mreži, iako se mogu koristiti izvan mreže. Kada su izvan mreže, ove aplikacije pružaju punu funkcionalnost.

Hibridna aplikacija, koja kombinira mrežne i izvanmrežne karakteristike, četvrta je vrsta aplikacije. Pružaju ograničenu funkcionalnost kada su izvan mreže, a punu funkcionalnost kada su na mreži. Za neke aplikacije ove vrste mogu biti ograničeni podaci pohranjeni na mobilnom uređaju, a podaci za ove aplikacije su pohranjeni na mreži. Primjer su Google karte. Ove aplikacije moraju biti na mreži kako bi se pružila potpuna funkcionalnost.

Type of app	Primary location of stored data	Currency of stored data on mobile device	Currency of stored data on server	Updating of stored data	Functionality
Offline	Mobile device	Current	NA	Device user	Offline
Online	Server	NA	Current	Device user and external entities	Online
Synchronized	Mobile device and server	Current as of last synchronization	Current as of last synchronization	Device user and external entities	Offline

Slika 3 − Karakteristike pohrane podataka u mobilnim aplikacijama (Nickerson & B., 2016)

Pri odabiru pristupa podacima za mobilne aplikacije mogu se uzeti u obzir brojni čimbenici. Na korisnika izravno utječu četiri čimbenika:

- brzina pristupa pohranjenim podacima: Koliko tehnologija za pohranu i obradu mobilnog uređaja može pružiti, toliko će brz biti pristup pohranjenim podacima kod izvanmrežnih ili sinkroniziranih aplikacija. Brzina komunikacijskog kanala i

količina podataka kojima se pristupa presudna je kod pristupa podataka mrežnim aplikacijama. Brzina pristupa kod izvanmrežnih ili sinkroniziranih i mrežnih aplikacija nije ista, što korisnici mogu primijetiti. Ono što utječe na mogućnost korisnika da pristupi podacima na vrijeme je brzina.

- dostupnost pohranjenih podataka: Pohranjeni podaci su uvijek dostupni kod izvanmrežnih i sinkroniziranih aplikacija. Za mrežne aplikacije dostupnost pohranjenih podataka ovisi o dostupnosti mrežne veze. Na pristup podacima koji su korisniku potrebni može utjecati dostupnost.
- volumen pohranjenih podataka: Volumen podataka koji se mogu pohraniti ograničen je kapacitetom memorije mobilnog uređaja kod izvanmrežnih i sinkroniziranih aplikacija. Volumen pohranjenih podataka za mrežne aplikacije može biti onoliki koliko poslužitelj može pohraniti te nije ograničen mobilnim uređajem. Izvanmrežne i sinkronizirane aplikacije ne pružaju sve podatke dostupne za mrežne aplikacije; zato korisnici aplikacije s vrlo velikim količinama podataka to mogu lako otkriti. Utjecaj na mogućnost korisnika da ima na raspolaganju sve potrebne podatke ima volumen podataka.
- aktualnost pohranjenih podataka: Pohranjeni podaci su uvijek aktualni za izvanmrežne i mrežne aplikacije. Za sinkronizirane aplikacije aktualnost pohranjenih podataka ovisi o aktivnosti baze podataka od posljednje sinkronizacije. Na korisnikovu mogućnost da ima dostupne ažurne podatke može utjecati aktualnost pohranjenih podataka. (Nickerson & B., 2016)

Type of app	Speed of stored data access	Availability of stored data	Volume of stored data	Currency of stored data
Offline	Fastest	Available at all times	Limited	Current
Online	May be limited	May not be available	Effectively unlimited	Current
Synchronized	Fastest	Available at all times	Limited	May not be current

Slika 4 − Četiri čimbenika za odabir pristupa podacima u mobilnim aplikacijama (Nickerson & B., 2016)

2. Veliki podaci

Poboljšanje industrije mobilnih aplikacija uzrokovano je radikalnim povećanjem broja mobilnih korisnika. Da bi obavili svakodnevne poslove, ljudi sve više koriste mobilne aplikacije. One troše puno podataka. Potreban je jedinstveni alat za upravljanje podacima za učinkovitu procjenu i obradu podataka u mobilnim aplikacijama (Rajput, 2021).

Velike podatke prikupljaju organizacije. Podaci se mogu dobiti za informacije i koristiti u projektima strojnog učenja, modeliranja i oni su kombinacija nestrukturiranih, polustrukturiranih i strukturiranih podataka. Glavni čimbenik u svijetu velikih podataka je brzina. Velike podatke karakterizira iznimno veliki volumen u bajtovima, širok izbor tipova podataka i iznimno velika brzina kojom se podaci generiraju, prikupljaju i obrađuju.

Doug Laney, analitičar u konzultantskoj tvrtki Meta Group Inc, prvi put je identificirao ove karakteristike 2001. godine. Nakon što je kupio Meta Group, Gartner ih je dodatno popularizirao. Nekoliko drugih V-ova kao što su istinitost, vrijednost i varijabilnost dodano je u novije vrijeme različitim opisima velikih podataka. Implementacija velikih podataka, iako oni ne predstavljaju nikakav specifičan volumen podataka, često uključuje terabajte (1024 gigabajta), petabajte (1024 terabajta), pa čak i eksabajte (1024 petabajta) podataka stvorenih i prikupljenih tijekom vremena. Poboljšanje poslovanja, pružanje bolje usluge korisnicima, kreiranje personaliziranih marketinških kampanja i poduzimanje drugih radnji koje u konačnici mogu povećati prihod i dobit neki su od razloga zašto tvrtke koriste velike podatke. Tvrtke koje koriste velike podatke, u odnosu na tvrtke koje ih ne koriste, imaju potencijalnu konkurentsku prednost jer mogu donositi brže i informiranije poslovne odluke. Tvrtke mogu upotrijebiti velike podatke za preciziranje svog marketinga, oglašavanja i promocije kako bi povećale angažman kupaca i pružile vrijedne uvide u klijente. Veliki podaci koriste se kako bi pomogli liječnicima u dijagnosticiranju bolesti i zdravstvenih stanja kod pacijenata te medicinskim istraživačima kako bi identificirali znakove bolesti. Velike podatke koriste i sljedeće organizacije:

- Veliki podaci pomažu u energetskoj industriji da plinske i naftne tvrtke identificiraju potencijalna mjesta za bušenje.

- Velike podatkovne sustave za upravljanje rizicima i analizu tržišnih podataka koriste tvrtke za financijske usluge.
- Za prevenciju kriminala, u hitnim slučajevima i inicijativu za pametne gradove također koristimo velike podatke.

Bezbroj je izvora iz kojih dolaze veliki podaci. To mogu biti mobilne aplikacije, društvene mreže, elektronička pošta, baze podataka klijenata, medicinska dokumentacija, sustavi za obradu transakcija i slično. Oblici velikih podataka također su slike, video i audio datoteke.

Najčešće citirana karakteristika velikih podataka je volumen. Široka paleta vrste podataka obuhvaćena je velikim podacima, kao što su:

- Transakcije i financijski zapisi spadaju u strukturirane podatke.
- Tekst, dokument i multimedijske datoteke spadaju u nestrukturirane podatke.
- Zapisnici mrežnih poslužitelja i prijenos podataka sa senzora spadaju u polustrukturirane podatke.

U sustavima velikih podataka možda će se morati pohranjivati i upravljati različitom vrstom podataka. Više skupova podataka koji možda nisu unaprijed integrirani često se koriste kod aplikacija za velike podatke.

Brzina se odnosi na brzinu kojom se podaci generiraju i moraju se analizirati i obraditi. Umjesto dnevnih, tjednih ili mjesečnih ažuriranja, skupovi velikih podataka ažuriraju se u stvarnom ili gotovo realnom vremenu. Analiza velikih podataka dalje se širi na strojno učenje i umjetnu inteligenciju te je zato upravljanje brzinom podataka važno.

Često se veliki podaci pohranjuju u podatkovnom jezeru (engl. data lake). Podatkovna jezera mogu podržavati različite tipove podataka i temelje se na Hadoop klasterima, uslugama za pohranu objekata u oblaku, nerelacijskim bazama podataka ili drugim velikim podatkovnim platformama, dok skladišta podataka sadrže obično samo strukturirane podatke i grade se na relacijskim bazama podataka (Bridget & J., n.d.).

Značajnim i važnim čimbenikom u razvoju mobilnih aplikacija velike podatke čini:

1. analiza korisničkog iskustva
2. dostupnost podataka u stvarnom vremenu
3. prilagođene marketinške kampanje

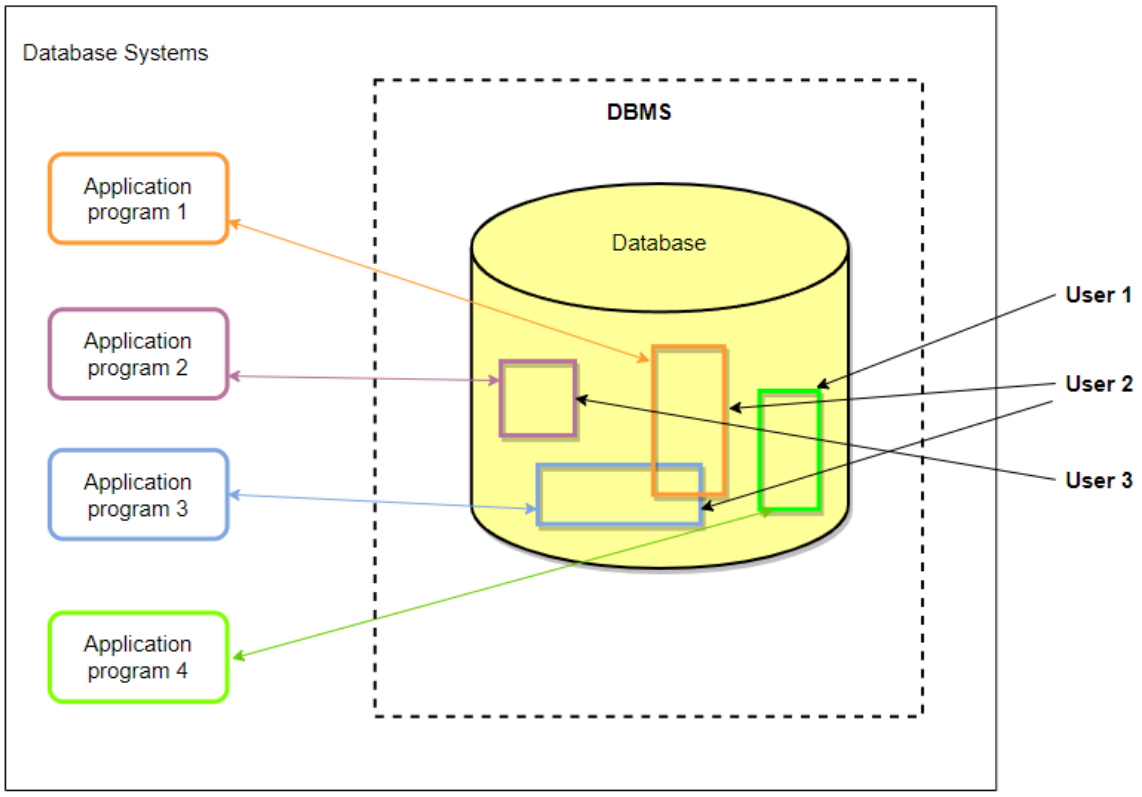
4. zahtjevi klijenata
5. pretvorbe prodaje
6. analitika društvenih mreža
7. povezani uređaji.

Kada je riječ o budućnosti mobilne aplikacije, veliki podaci uključeni su u digitalne tehnologije. Predviđa se da će biti s nama i u budućnosti, s obzirom na to da igraju ključnu ulogu u razvoju mobilnih aplikacija. Veliki podaci postali su bitan dio industrije mobilnih aplikacija, zbog svoje jednostavne izvedbe i naprednih značajki. Jedna od najboljih odluka za agencije je ulaganje u analizu velikih podataka. Ogromnu količinu podataka o lokaciji i potrebama korisnika nam također nude veliki podaci. Organizacije bi trebale učinkovito koristiti podatke dobivene analitikom velikih podataka kako bi ostale ispred konkurencije (Rajput, 2021).

3. Baze podataka

Baza podataka (engl. Database) zajednička je zbirka srodnih podataka. Podacima smatramo poznate činjenice koje možemo zabilježiti i koje imaju implicitno značenje (Educative, n.d.). Baza podataka organizira podatke u obliku tablica, pogleda (engl. view), shema, izvješća, itd (Sonal & Paramjeet, n.d.). Male baze podataka mogu se pohraniti u datotečni sustav, a velike baze podataka pohranjene su u oblaku ili smještene na računalnim klasterima (Wikipedia, 2022). Na primjer, razmotrimo imena, telefonske brojeve i adrese ljudi koje poznajemo.

Baza podataka može se promatrati kao spremište podataka koje je jednom definirano, a zatim mu pristupaju različiti korisnici kao što je prikazano na Slika 5 (Educative, n.d.).



Slika 5 – Baza podataka (Educative, n.d.)

Aplikacije su ništa drugo nego programska podrška koja različitim korisnicima omogućuju dohvaćanje ili slanje podataka iz baze podataka.

Sljedeća su svojstva baza podataka:

- Mini svijet je neki aspekt stvarnog svijeta koji predstavlja baza podataka. Ako se dogodi promjena u mini svijetu, ta promjena će se odraziti u bazi podataka.
- Baza podataka logički je koherentna zbirka podataka s nekim inherentnim značenjem. Nasumični podaci ne mogu se nazvati bazom podataka. Ako zabilježimo adrese i telefonske brojeve nasumičnih ljudi, podaci će imati neko inherentno značenje (tj. bit će adresar), dok se popis nasumičnih imena ne može smatrati bazom podataka.
- Baza podataka dizajnirana je, izgrađena i popunjena podacima za određenu svrhu. Ona ima namijenjenu grupu korisnika i unaprijed zamišljene aplikacije za koje su ti korisnici zainteresirani.

Baza podataka ima određeni stupanj interakcije s događajima u stvarnom svijetu i publikom koja je aktivno zainteresirana za njezin sadržaj. Transakcije mogu izvršiti korisnici baze podataka. Na primjer, ocjene studenta povećaju se na ispitu što uzrokuje promjenu informacija u bazi podataka. Da bi bazu podataka smatrali pouzdanom, promjene se moraju što prije odraziti u bazi podataka kako bi ona mogla ostati pravi odraz mini svijeta kojeg predstavlja.

Baze podataka mogu biti različitih veličina i složenosti. Pogledajmo primjer popisa imena i adresa koji je već spomenut. Baza podataka o imenima i adresama može se sastojati od samo nekoliko stotina unosa s jednostavnim tekstualnim informacijama.

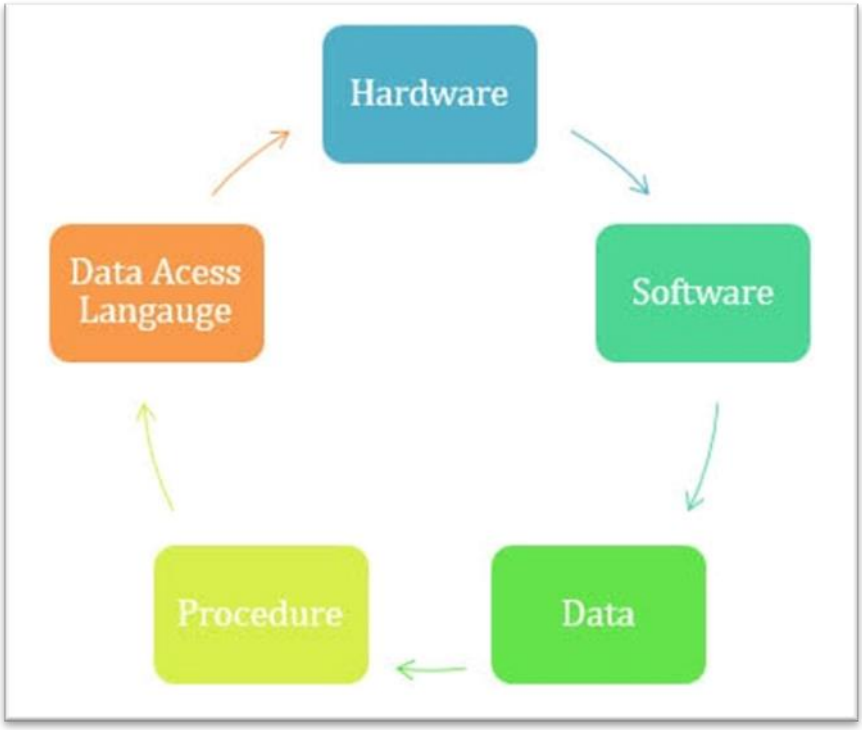
Dok baze podataka koju održava tvrtka društvenih mreža poput Twittera ili Instagrama ima milijune korisnika. Informacije o tome koji su korisnici međusobno povezani u obliku pratitelja, sadržaja koji je svaki korisnik objavio te velik broj drugih informacija mora održavati baza podataka (Educative, n.d.).

Postoji pet glavnih komponenti baze podataka kao što je prikazano na slici dolje:

- Strojna oprema koja se sastoji od fizičkih, elektroničkih uređaja kao što su računala, ulazno-izlazni uređaji, uređaji za pohranu, itd. Ovo nudi sučelje između računala i sustava u stvarnom svijetu.
- Programska podrška je skup programa koji se koriste za kontrolu i upravljanje cjelokupnom bazom podataka. Taj skup programa čine sama programska podrška baze podataka, operacijski sustav, mrežna programska podrška koja se koristi za

dijeljenje podataka među korisnicima i aplikacije za pristup podacima bazi podataka.

- Podaci (engl. Data) su neorganizirana i sirova činjenica koju je potrebno obraditi da bi bila smislena. Podaci se općenito sastoje od činjenica, zapažanja, brojeva, percepcija, znakova, simbola, slika, itd.
- Procedure (engl. Procedure) predstavljaju skup pravila i uputa koja vam pomažu u korištenju DBMS-a.
- Jezik pristupa bazi podataka (engl. Database Access Language) koristi se za pristup podacima iz baze podataka i unos novih podataka, ažuriranje već postojećih ili dohvaćanje potrebnih podataka iz DBMS-a. Korisnik specifične naredbe piše u jeziku za pristup bazi podataka i šalje ih bazi podataka (Peterson, 2022).

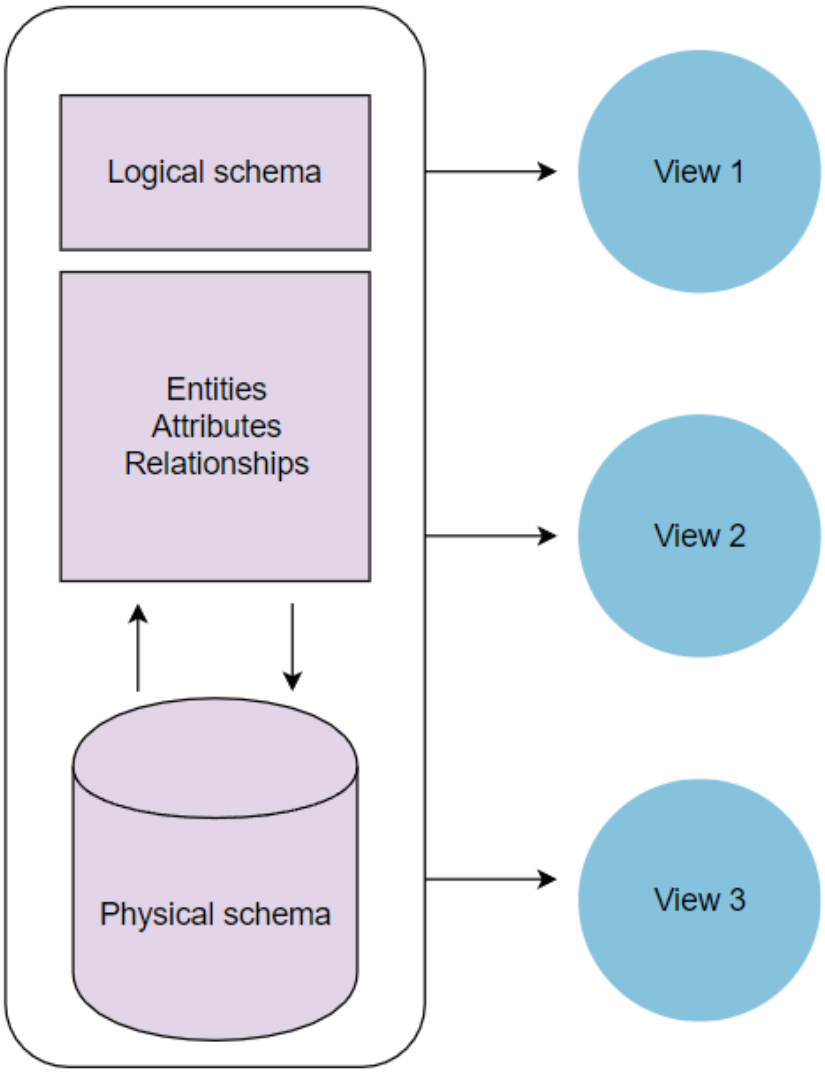


Slika 6 − Pet glavnih komponenti baze podataka (Peterson, 2022)

3.1. Sheme baza podataka

Sheme baza podataka apstraktni su dizajni koji predstavljaju pohranu podataka u bazi podataka. Sheme baza podataka opisuju kako su podaci organizirani i odnose između tablica u bazi podataka. Potrebno je unaprijed planirati shemu baze podataka kako bismo znali koje su komponente potrebne i kako će se međusobno povezati. One ne sadrže podatke, već umjesto toga opisuju oblik podataka i kako se odnose na druge tablice ili modele. Instanca sheme baze podataka je unos u bazi podataka.

Dvije su glavne vrste shema baza podataka koje definiraju različite dijelove sheme: logičke i fizičke.



Slika 7 − Shema baze podataka (Schaffer, 2022)

Sheme baza podataka uključuju:

- sve relevantne ili važne podatke
- dosljedno oblikovanje za sve unose podataka
- jedinstveni ključevi za sve unose i objekte baze podataka
- svaki stupac u tablici ima naziv i vrstu podataka.

O veličini projekta ovisi veličina i složenost shema baze podataka. Vizualni stil shema baze podataka omogućuje nam pravilno strukturiranje baze podataka i njezinih odnosa prije nego što prijeđemo na kod. Modeliranje podataka je proces planiranja dizajna baze podataka. Sheme baze podataka važne su kod projektiranja sustava za upravljanje bazom podataka i relacijskog sustava za upravljanje bazom podataka (Schaffer, 2022), koji će biti opisani dalje u radu.

3.2. Sustav za upravljanje bazom podataka

Sustav za upravljanje bazom podataka (engl. Database Management System, skraćeno DBMS) je programska podrška koja komunicira s krajnjim korisnicima, aplikacijama i samom bazom podataka kako bi prikupio i analizirao podatke. Programska podrška DBMS-a dodatno obuhvaća osnovne objekte za upravljanje bazom podataka (Wikipedia, 2022). DBMS-ovi nisu novi koncepti, a prvi put su implementirani 1960-ih. Prvi DBMS u povijesti je Integrated Data Store (IDS) Charlesa Bachmana (Peterson, 2022). Skup baze podataka, DBMS-a i pridruženih aplikacija može se nazvati sustavom baze podataka. Izraz „baza podataka“ često se slobodno koristi za označavanje bilo kojeg DBMS-a, sustava baze podataka ili aplikacije povezane s bazom podataka.

Računalni znanstvenici klasificiraju sustave upravljanja bazama podataka prema modelima baze podataka koje podržavaju. Relacijske baze podataka su postale dominantne 1980-ih. Relacijske baze podataka modeliraju podatke kao retke i stupce u nizu tablica, a velika većina koristi strukturirani jezik za upite (engl. Structured Query Language, skraćeno SQL) za pisanje i ispitivanje podataka. Početkom 21. stoljeća postale su popularne nerelacijske baze podataka, koje se još nazivaju NoSQL baze podataka jer koriste različite jezike upita (Wikipedia, 2022).

Povijest sustava upravljanja bazom podataka:

1960. – prvi DBMS Charlesa Bachmana

1970. – Codd je predstavio IBM-ov sustav upravljanja informacijama (IMS)

1976. – definiran je model odnosa entiteta poznat kao i ER (engl. Entity Relation) model od strane Petera Chena

1980. – relacijski model polako postaje široko prihvaćena komponenta baze podataka

1985. – razvijanje objektno orijentiranog DBMS-a

1990. – ugradnja objektne orijentacije u DBMS

1991. – Microsoft isporučuje MS pristup, osobni DBMS koji zamjenjuje sve druge osobne DBMS proizvode

1995. – prve aplikacije za internetske baze podataka

1997. – XML primijenjen na obradu baze podataka.

Prednosti DBMS-a su:

- DBMS nudi razne tehnike za dohvaćanje i pohranu podataka.
- DBMS balansira potrebe više aplikacija koje koriste podatke.
- Nudi jedinstvene administrativne procedure za podatke.
- Programeri aplikacija nikada nisu bili izloženi detaljima pohrane podataka.
- Nudi razne funkcije za učinkovito pohranjivanje i dohvaćanje podataka.
- Nudi sigurnost podataka i integritet.
- DBMS podrazumijeva ograničenja integriteta kako bi dobili visoku razinu zaštite od zabranjenog pristupa podacima.
- Raspoređuje istovremeni pristup podacima tako da samo jedan korisnik može pristupiti istim podacima u isto vrijeme.
- Za razvoj aplikacija potrebno je manje vremena.

Nedostaci DBMS-a

DBMS nudi mnogo prednosti, ali ima i neke nedostatke:

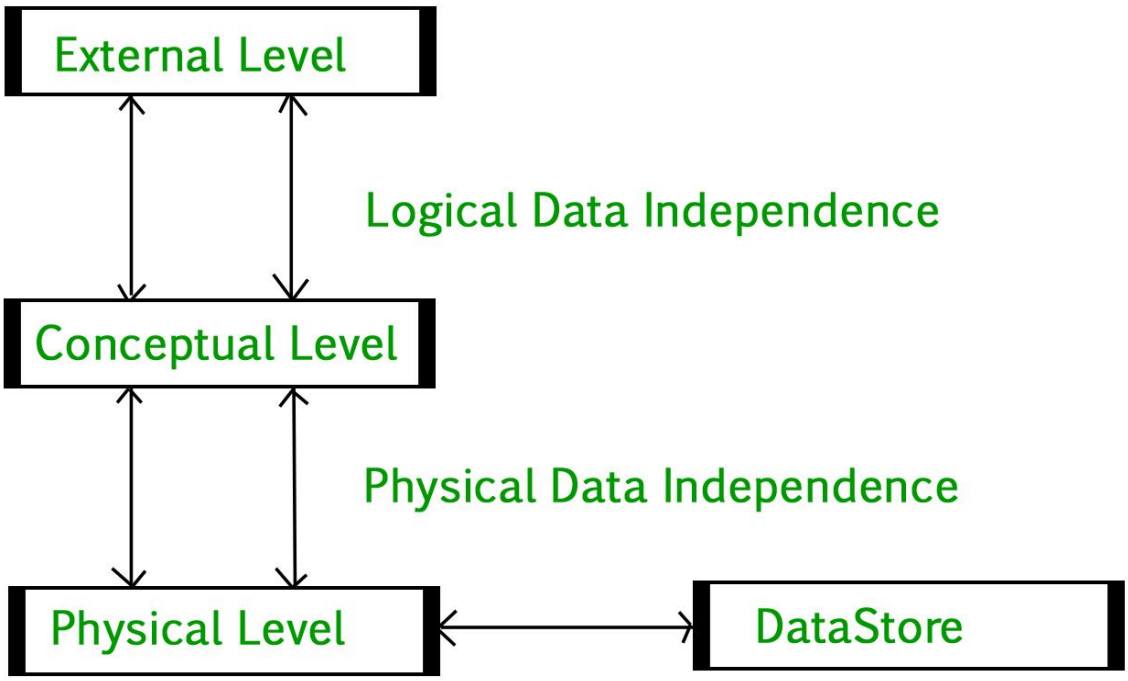
- Cijena strojne opreme i programske podrške DBMS-a je visoka što dovodi do povećanja budžeta organizacije.
- Najčešće je potrebna obuka korisnika za korištenje DBMS-a zato što su većina sustava za upravljanje bazama podataka složeni sustavi.
- DBMS ne može izvoditi sofisticirane račune.
- Ako u isto vrijeme korisnici upotrebljavaju isti program, može doći do gubitka nekih podataka.
- Podaci mogu biti integrirani u jednu bazu podataka koja se može oštetiti zbog strujnih kvarova ili ako je baza podataka oštećena na mediju za pohranu (Peterson, 2022).

3.2.1. Troslojna arhitektura u DBMS-u

DBMS troslojna arhitektura dijeli cijeli sustav u tri međusobno povezana, ali neovisna modula kao što je prikazano na slici dolje:

1. Fizička razina (engl. Physical Level): informacije o lokaciji objekata baze podataka u spremištu podataka čuvaju se na fizičkoj razini. Lokacije ovih objekata nisu poznate različitim korisnicima DBMS-a. Kod fizičke razine baze podataka opisuje se kako se podaci pohranjuju na sekundarnim uređajima za pohranu kao što su diskovi i vrpce te se daje uvid u dodatne pojedinosti pohrane.
2. Konceptualna razina (engl. Conceptual Level): podaci su na konceptualnoj razini predstavljeni u obliku različitih tablica baze podataka. Baza podataka Student, na primjer, može sadržavati tablice Student i Kolegij, koje će biti vidljive korisnicima, ali njihovi korisnici nisu svjesni njihove pohrane. Drugi naziv za konceptualnu razinu je logička shema jer opisuje kakvu vrstu podataka pohraniti u bazi podataka.
3. Vanjska razina (engl. External Level): pogled na podatke u smislu tablica konceptualne razina specificira vanjska razina. Za zadovoljavanje potreba određene

kategorije korisnika koristi se svaki pogled vanjske razine. Apstrakcija podataka je glavni fokus vanjske razine (Tuteja, 2021).



Slika 8 − Troslojna arhitektura u DBMS-u (Tuteja, 2021)

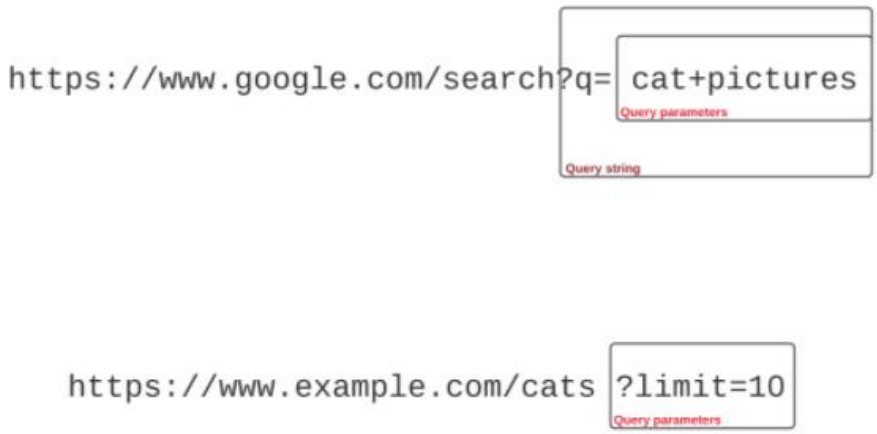
Promjene podataka na jednoj razini ne bi trebali utjecati na drugu razinu, odnosno podaci su neovisni. U ovoj arhitekturi prisutne su dvije vrste neovisnosti podataka:

1. Fizička neovisnost podataka (engl. Physical Data Independence): kad se dogodi promjena u fizičkoj lokaciji tablice i indeksa, ona ne bi trebala utjecati na konceptualnu razinu ili vanjski pogled na podatke. Većina DBMS-a je implementirala ovu neovisnost podataka.
2. Neovisnost konceptualnih podataka (engl. Conceptual Data Independence): na konceptualnoj razini sheme i shemi vanjske razine podaci moraju biti neovisni. Promjena konceptualne sheme ne bi trebala imati utjecaj na vanjsku shemu. Ako dodajemo ili brišemo atribute tablice, to ne bi trebalo imati utjecaj na korisnički prikaz tablice. Neovisnost konceptualnih podataka teže je postići u odnosu na neovisnost fizičkih podataka jer se promjene konceptualne sheme održavaju u korisnikovom pogledu (Tuteja, 2021).

3.3. Upiti u bazama podataka

Upit (engl. Query) je pitanje na vrlo visokoj razini. Kada postavljamo pitanje drugoj osobi, od nje očekujemo odgovor. Isto je i kod računala kada obavljamo upite u bazi podataka. CRUD operacije povezuju se s upitima baze podataka i bit će opisane u radu. Zahtjev za pristup podacima iz baze podataka radi manipulacije ili dohvaćanja je upit baze podataka. Izvođenje logike s informacijama koje dobijemo kao odgovor na upit nam omogućuju ti zahtjevi. Korištenje nizova upita, pisanje jezikom upita i korištenje upita po primjeru (engl. Query By Example, skraćeno QBE) neki su od različitih pristupa upitima. Primjer upita po primjeru je GraphQL, koji omogućuje da zatražimo određene podatke, tako da klijentima dajemo veću kontrolu nad informacijama koje se šalju. Uz GraphQL korisnici mogu, ni manje ni više, tražiti i primati samo određene podatke koje traže.

Na kraju usklađenog lokatora sadržaja (engl. Uniform Resource Locator, skraćeno URL) stavljaju se parametri upita (engl. Query Parameters) kao dio niza upita. Parametre koje korisnik unese u traku za pretraživanje, tražilice koriste za pronalazak rezultata pretraživanja.

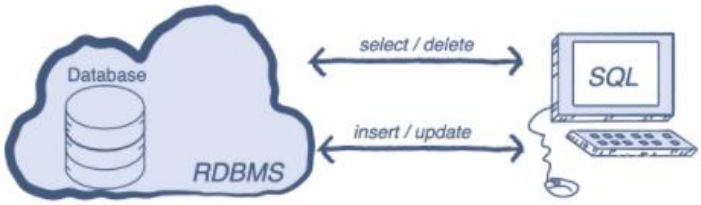


Slika 9 − Parametri upita (Kopecky, 2020)

U većini slučajeva, prilikom korištenja parametra upita nije potrebno znati stvarni jezik upita. Upit po primjeru ili QBE, formuliran je od strane računalnog znanstvenika u IBM-u 1970-ih. To je sustav za filtriranje ili pretraživanje baze podataka u kojima nije bilo potrebe za korištenjem jezika upita.

Poduzimanje radnji na bazama podataka kao što je stvaranje, čitanje, ažuriranje i brisanje stavki u bazi podataka te naprednije upite poput filtriranja i brojanja omogućuje nam jezik upita. Najpoznatiji jezik upita je strukturirani jezik upita ili SQL. Upit po primjeru koji je razvio IBM 1970-ih je zajedno odrastao sa SQL-om. SQL radi na temelju relacijskih baza podataka. Možemo pohraniti, dohvaćati i manipulirati podacima koristeći upite u relacijskom DBMS-u uz pomoć SQL-a.

U relacijskom DBMS-u podaci se pohranjuju na strukturiran način, gdje postoje odnosi između entiteta i varijabli u podacima. Shema baze podataka specificira odnos između različitih entiteta i organizaciju podataka za entitete te definira odnose (Kopecky, 2020).



Slika 10 − Strukturirani jezik upita (SQL) (Kopecky, 2020)

3.4. CRUD operacije

Osnovne operacije programiranje sustava za upravljanje relacijskim bazama podataka su CRUD (engl. create, read, update, delete) operacije. CRUD operacije koriste se za čitanje, brisanje, manipulaciju, umetanje i uređivanje podataka tablice.

Za programere je ključno razumjeti CRUD operacije jer je SQL istaknut u većini industrija. Sustavi koji pružaju trajnu pohranu potrebni su bilo kojoj organizaciji koja prati podatke (kao što su računi, podaci o plaćanju ili drugi zapis), a koja je obično organizirana u bazu podataka. Podaci u relacijskim bazama podataka organizirani su po recima i stupcima. Pomoću primarnih i stranih ključeva moguće je povezivanje s drugim tablicama. Akronim CRUD odnosi se na četiri funkcije koje koristimo za implementaciju aplikacija za trajnu pohranu i aplikacija za relacijske baze podataka, uključujući Oracle Database, Microsoft SQL Server i MySQL (Sulemani, 2021).

Letter	Operation	Function
C	Create	Insert
R	Read	Select
U	Update	Edit
D	Delete	Delete

Slika 11 − CRUD operacije (Sulemani, 2021)

Sigurnosna kontrola, kontrola transakcija, pristup i dopuštenje te optimizacija performansi su operacije temeljene na CRUD-u. Za sve što je povezano s bazom podataka i dizajnom baze podataka stalno koristimo CRUD operacije. Bez njih ništa ne mogu učiniti ni programeri programske podrške. Također, važne su i za krajnje korisnike. Registracija za mrežne stranice i stvaranje blogova ne bi bila moguća bez CRUD-a. Većina aplikacija koje koristimo dopuštaju nam dodavanje ili stvaranje novih unosa, traženje postojećih, njihovo mijenjanje ili brisanje.

Prednosti CRUD-a su:

- Sigurnosna kontrola je olakšana jer zadovoljava zahtjeve pristupa.
- Čini aplikacije skalabilnijima jer pojednostavljuje i olakšava dizajn.

Dodavanje novih redaka u tablicu nam omogućuje CREATE. Pomoću naredbe INSERT INTO možemo dodati nove retke u tablicu. INSERT INTO je ključna riječ, nakon čega slijedi naziv tablice, nazivi stupaca i vrijednosti koje treba umetnuti. Imamo dvije mogućnosti kada koristimo INSERT INTO.

```
INSERT INTO table_name
 VALUES (value1, value2, value3, ...);
```

Slika 12 − INSERT INTO primjer 1 (Sulemani, 2021)

```
INSERT INTO table_name (column1, column2, column3, ...)
 VALUES (value1, value2, value3, ...);
```

Slika 13 – INSERT INTO primjer 2 (Sulemani, 2021)

Podatke ćemo dodati u tablicu pekara, što je prikazano na slici dolje.

```
INSERT INTO menu
VALUES (1, 'croissant', 1, '2020-12-16');
If we want to add multiple rows, we do that using:
INSERT INTO menu
VALUES
(2, 'bread', 3, '2020-12-16' ),
(3, 'eclairs', 2, '2020-12-16' );
```

Slika 14 – INSERT INTO primjer 3 (Sulemani, 2021)

Svaki unos imat će jedinstveni ID i novi redci bit će dodani u tablicu menu.

Funkcija čitanja omogućuje nam dohvaćanje određenih zapisa i čitanje njihovih vrijednosti te je slična funkciji pretraživanja. Čitanje radi naredba SELECT.

Sve podatke iz tablice menu prikazat ćemo na sljedeći način:

```
SELECT * FROM menu;
```

Slika 15 – Prikaz svih podataka iz tablice naredbom SELECT (Sulemani, 2021)

Korištenjem ove naredbe neće se napraviti nikakve promjene u tablici menu, nego će se prikazati svi zapisi u tablici.

```
IF OBJECT_ID('cusp_CustomerRead') IS NOT NULL
BEGIN
    DROP PROC cusp_CustomerRead
END
GO
CREATE PROC cusp_CustomerRead
    @CustomerID int
AS
BEGIN

    SELECT CustomerID, FirstName, LastName, Email, PhoneNumber
    FROM   Customer
    WHERE  (CustomerID = @CustomerID)
END
GO
```

Slika 16 – Odabir željenih podataka naredbom SELECT (Sulemani, 2021)

Način na koji mijenjamo postojeći zapis u tablici je ažuriranje. Za izmjenu postojećih podataka u bazi podataka koristimo naredbu UPDATE. Potrebno je definirati tablicu u kojoj želimo ažurirati podatke i stupce koji će se ažurirati. Trebamo povezane vrijednosti, a ponekad i cijele retke.

Naredbu UPDATE koristimo za ažuriranje postojećeg zapisa na sljedeći način:

```
UPDATE table_name
 SET column1 = value1, column2 = value2, ...
 WHERE condition;
```

Slika 17 – Ažuriranje postojećeg zapisa naredbom UPDATE (Sulemani, 2021)

Operacija DELETE koristi se za uklanjanje zapisa iz tablice. Funkcije za brisanje jednog ili više zapisa iz baze podataka odjednom su ugrađene u SQL-u. Razlikujemo ažuriranje statusa retka (engl. soft delete) ili trajno brisanje (engl. hard delete).

```
DELETE FROM table_name WHERE condition;
```

Slika 18 – Naredba DELETE (Sulemani, 2021)

Uklanjanje jedne stavke iz tablice naredbom DELETE:

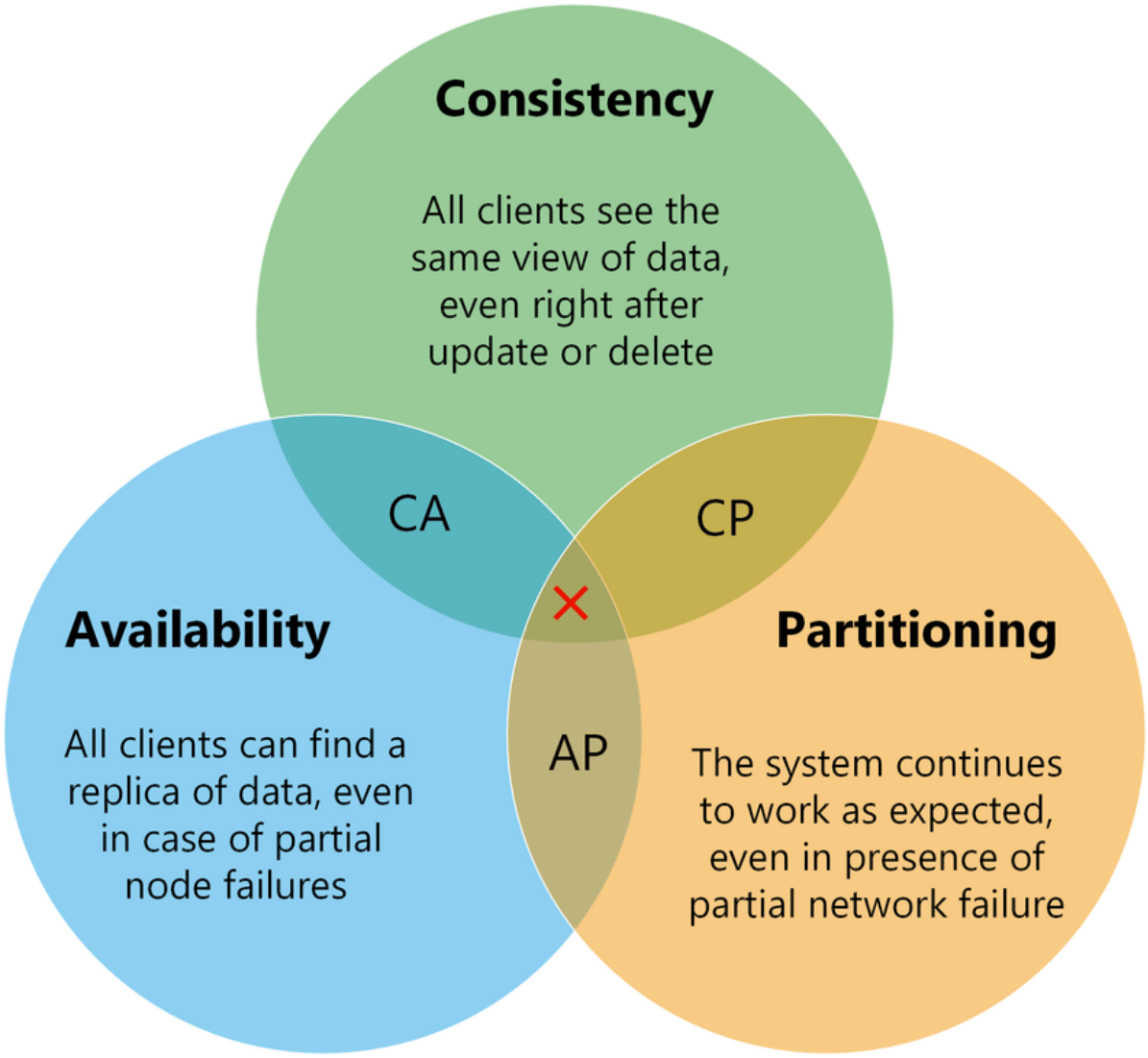
```
DELETE FROM menu WHERE item_name='bread';
```

Slika 19 − Uklanjanje jedne stavke iz tablice naredbom DELETE (Sulemani, 2021)

Slika 19 prikazuje uklanjanje samo reda sa stavkom kruh iz tablice menu.

3.5. CAP teorem

CAP teorem je predstavio informatičar Eric Brewer 1998. godine. Gilbert i Lynch su kasnije objavili Brewerovu pretpostavku, formalni dokaz njegovog teorema. CAP je skraćenica za dosljednost (engl. Consistency), dostupnost (engl. Availability) i postojanje mrežnih particija (engl. Partition tolerance). Dosljednost, dostupnost i postojanje mrežnih particija ne može biti zajamčeno od raspodijeljenog sustava cijelo vrijeme (Keboola, 2021).



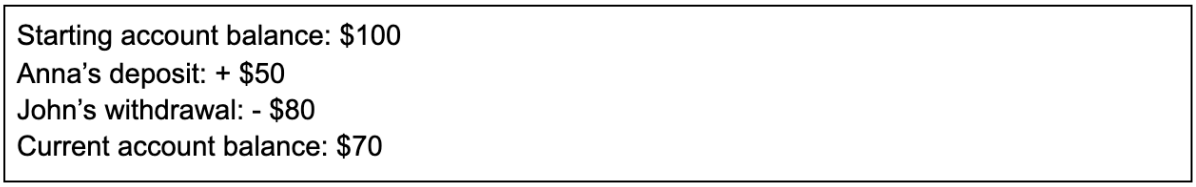
Slika 20 – CAP teorem (Khazaei, n.d.)

Moramo odlučiti o kompromisu i dati prioritet za najviše dvije karakteristike raspodijeljenih sustava koje želimo zadržati kada stvari krenu po zlu, kao što prikazuje Slika 20.

3.5.1. Dosljednost

Bez obzira koji čvor tražimo, dosljedan sustav uvijek vraća iste informacije. Zamislimo da postoji više čvorova diljem svijeta. Na tim čvorovima klijenti nešto zapisuju.

Na primjer, Anna s Filipina uplaćuje novac svom bratu Johnu na obiteljski račun, a on podiže novac s računa u Nizozemskoj (Keboola, 2021).



Slika 21 − Primjer dosljednog sustava (Keboola, 2021)

Njima nije važno što su čvorovi diljem svijeta i trebaju sinkronizirati informacije između njih, nego ih samo zanima kakvo je njihovo stanje na računu nakon zadnjih operacija. Neovisno o tome koji čvorovi prvi obrađuju tu informaciju, dosljedan sustav pokazat će iste informacije. Također, bitno je razlikovati konzistentnost u ACID-u i dosljednost u CAP teoremu. Dosljednost u CAP teoremu jamči da će podaci biti univerzalno dosljedni u svim čvorovima, dok konzistentnost u ACID-u jamči da će podaci u sustavu baze podataka biti valjani nakon transakcije (Keboola, 2021).

3.5.2. Dostupnost

Dostupan sustav će, bez pogreške, dati odgovor svakom zahtjevu za čitanje ili pisanje. On jamči da će zahtjevi za pisanje i čitanje biti obrađeni, ali ne i da će promatrati najnovije promjene, što znači da ne jamči konzistentnost. Sustavi visoke dostupnosti dupliciraju podatke na više čvorova tako da su dostupni za prihvaćanje zahtjeva za čitanje ili pisanje (Keboola, 2021).

3.5.3. Postojanje mrežnih particija

Sposobnost raspodijeljenog sustava da normalno radi u vrijeme kvarova na mreži odnosi se na postojanje mrežnih particija. Tehnički izraz kvara mreže je mrežna particija. Cijela mreža podijeljena je na manje dijelove mreže jer su jedan ili dva čvora izgubila komunikaciju između njih. Postojanje mrežnih particija osigurava da mrežne particije neće utjecati na korisnike koji sustavu šalju zahtjeve za čitanje ili pisanje. Takav sustav mora se ponašati kao da niti jedan čvor nije otkazao (Keboola, 2021).

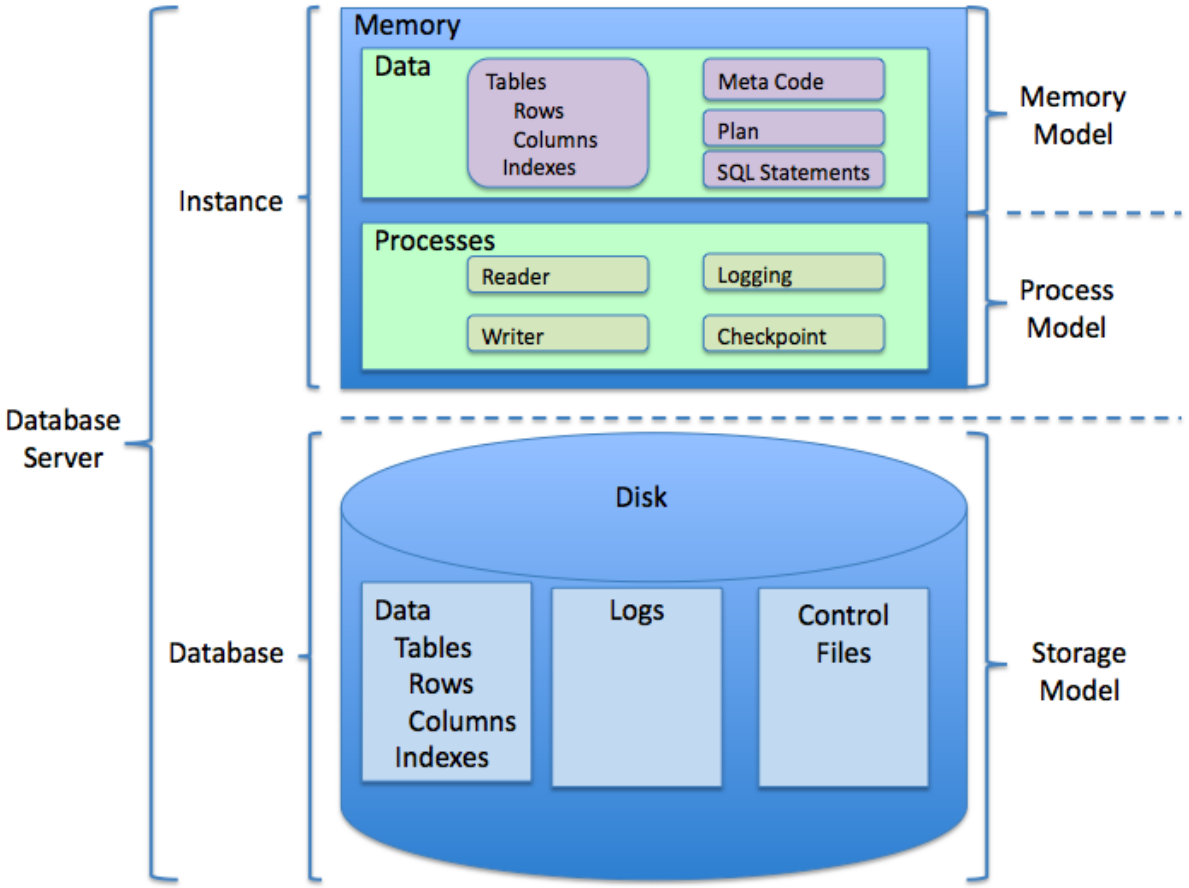
3.6. Relacijske baze podataka

„Relacijska baza podataka je digitalna baza podataka koja se temelji na relacijskom modelu podataka, kako ga je predložio E. F. Codd 1970. godine (Wikipedia, 2022).“ Relacijske baze podataka imaju strukturu i još se nazivaju SQL baze podataka. Baš poput

telefonskih imenika koji pohranjuju brojeve i adrese, relacijske baze podataka imaju unaprijed definirane sheme (Schaffer, 2022). Sustav za upravljanje relacijskom bazom podataka je sustav koji se koristi za održavanje relacijskih baza podataka. SQL se koristi za postavljanje upita i održavanje baze podataka kod mnogih relacijskih DBMS-ova (Wikipedia, 2022).

Neke od popularnih relacijskih baza podataka su:

- MySQL
- Oracle
- MS SQL Server
- SQLite
- PostgreSQL
- MariaDB (Schaffer, 2022).



Slika 22 − Struktura relacijske baze podataka (Wikipedia, 2022)

E. F. Cood je izumio pojam relacijske baze podataka u IBM-u 1970. godine. U svom istraživačkom radu „Relacijski model podataka za velike dijeljene podatke banaka“ prvi je put uveo taj pojam. Pojam je postupno počeo opisivati širu klasu sustava baza podataka, zbog toga što nijedna komercijalna implementacija relacijskog modela ne odgovara svim Codd-ovim pravilima, za koji u najmanju ruku vrijedi:

1. Korisniku se podaci moraju predstaviti kao relacije (kao zbirka tablica pri čemu se svaka od tablica sastoji od redaka i stupaca, tj. prezentacija u obliku tablica).
2. Relacijski operatori su omogućeni za manipulaciju podacima u obliku tablica.

Proizvod koji predstavlja prikaz podataka kao skup redaka i stupaca najčešća je definicija relacijskog DBMS-a, čak i ako se ne temelji striktno na relacijskoj teoriji. Obično, prema ovoj definiciji, relacijski DBMS proizvodi ne implementiraju svih 12 Codd-ovih pravila (Wikipedia, 2022).

Prednosti modela relacijske baze podataka su:

- jednostavnost
- Strukturna neovisnost – relacijska baza podataka ne bavi se strukturom, nego samo podacima što može poboljšati performanse modela.
- Jednostavnost za korištenje – tablice koje se sastoje od redaka i stupaca prilično su prirodne i jednostavne za razumijevanje, i zato je relacijski model u DBMS-u jednostavan.
- Neovisnost podataka – moguće je mijenjati strukturu baze podataka bez potrebe za mijenjanjem bilo koje aplikacije.
- Skalabilnost – da bi se poboljšala upotrebljivost baze podataka, treba povećati bazu podataka.

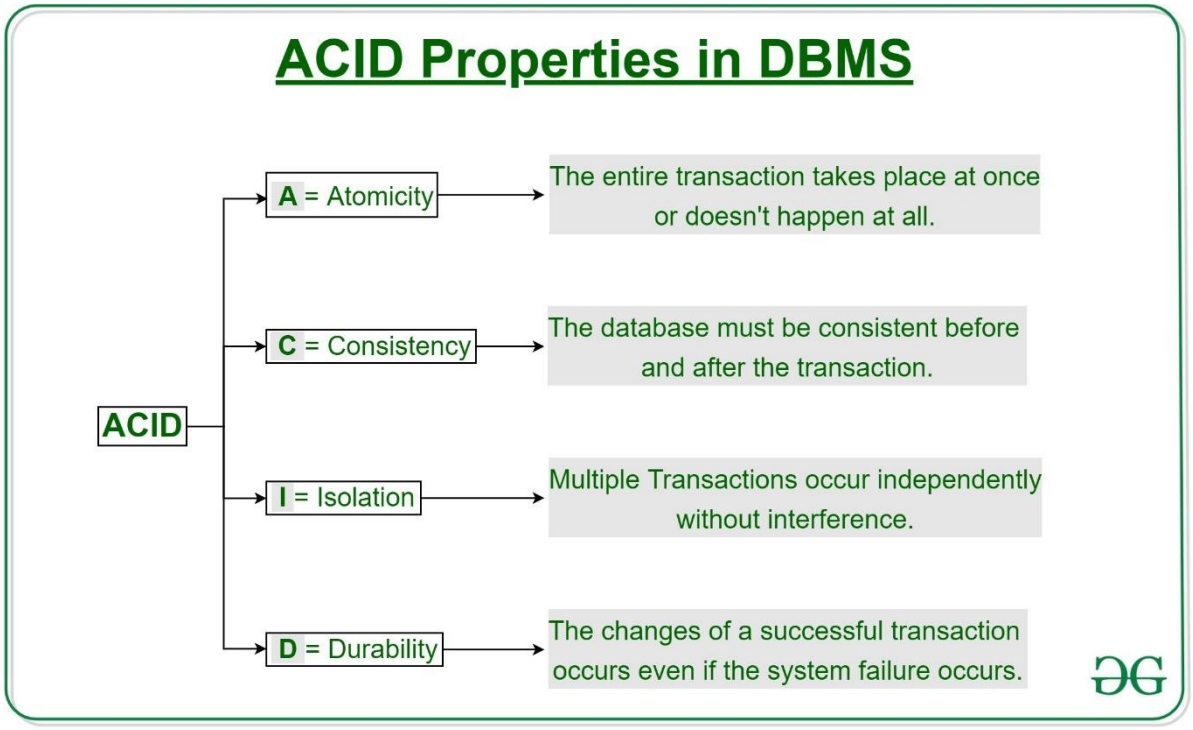
Nedostaci modela relacijske baze podataka su:

- Kod nekih relacijskih baza podataka ne mogu se prekoračiti ograničenja na duljinu polja.
- Kako količina podataka raste, relacijske baze podataka mogu postati složene, a odnosi između dijelova podataka postaju kompliciraniji.

- Složeni sustavi relacijskih baza podataka mogu dovesti do izoliranih baza podataka u kojima se informacije ne mogu dijeliti s jednog sustava na drugi (Peterson, Guru99, 2022).

3.6.1. ACID transakcije

Jedna logička jedinica rada koja pristupa i eventualno mijenja sadržaj baze podataka naziva se transakcija. Pomoću operacije čitanja i pisanja transakcije pristupaju podacima. Slijede se određena pravila kako bi se nakon i prije transakcije održala dosljednost u bazi podataka. Ta svojstva nazivaju se ACID svojstva (Kaur, 2022).



Slika 23 − ACID svojstva (Kaur, 2022)

Atomarnost (engl. atomacity) osigurava da se cijela transakcija odvija odjednom ili da se uopće ne dogodi. Transakcije se ne događaju djelomično, tj. nema sredine. Svaka transakcija izvodi se ili se uopće ne izvršava i smatra se jednom jedinicom. Atomarnost uključuje dvije operacije:

- Prekini (engl. abort) – promjene u bazi podataka nisu vidljive ako se transakcija prekine.
- Izvrši (engl. commit) – vidljive su napravljene promjene ako se transakcija izvrši.

Drugi naziv za atomarnost je „pravilo sve ili ništa“.

Transakciju T koja se sastoji od T1 i T2 ćemo razmotriti:

Prijenos 100 s računa X na račun Y

Before: X : 500	Y: 200
Transaction T	
T1	**T2**
Read (X) X: = X − 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

Slika 24 − Prijenos s računa X na račun Y (Kaur, 2022)

Ako transakcija ne uspije prije završetka T2, a nakon završetka T1 (recimo, nakon write(x), a prije write (y)), iznos koji je oduzet od X, nije dodan Y. To će rezultirati nedosljednim stanjem baze podataka. Kako bi se osigurala ispravnost stanja baze podataka transakcija se mora izvršiti u cijelosti.

Konzistentnost (engl. consistency) znači da se ograničenja integriteta moraju održavati tako da baza podataka bude dosljedna prije i nakon transakcije. Prije i nakon transakcije ukupni iznos mora ostati isti. Na gornjem primjeru, ukupni iznos prije transakcije je 700 (X = 500, Y = 200), što mora ostati i nakon transakcije (X = 400, Y = 300). Baza podataka je konzistentna. U slučaju da se T1 završi, a T2 ne uspije javlja se nedosljednost baze podataka.

Izoliranost (engl. isolation) omogućuje da se više transakcija može izvršavati istovremeno, a da ne dođe do nedosljednosti stanja baze podataka. Transakcije se odvijaju bez smetnji, neovisno jedna o drugoj. Sve dok se određena promjena u toj transakciji ne zapiše u memoriju ili ne preda, promjene koje se događaju u određenoj transakciji neće biti vidljive niti jednoj drugoj transakciji. Stanje nakon istovremenog izvršavanja transakcija bit će isto kao i stanje da su se transakcije izvršavale serijski, odnosno nekim redoslijedom.

Neka je X = 500, Y = 500. Razmotrit ćemo dvije transakcije T i T".

T	T''
Read (X) X: = X*100 Write (X) Read (Y) Y: = Y – 50 Write (Y)	Read (X) Read (Y) Z: = X + Y Write (Z)

Slika 25 – Transakcije T i T" (Kaur, 2022)

Neka je T izvršeno do Read (Y), a zatim počinje T". Dolazi do preplitanja operacija, kao rezultat tog izvršavanja, zbog čega T" čita ispravnu vrijednost X, ali netočnu vrijednost Y i zbroj izračunat od strane T":

T": (X + Y = 50 000 + 500 = 50 500)

Na kraju transakcije zbroj neće biti isti kao na početku transakcije:

T": (X + Y = 50 000 + 450 = 50 450)

Zbog gubitka od 50 jedinica, baza podataka je nedosljedna. Promjene bi trebale biti vidljive tek nakon što su napravljene u glavnoj memoriji i transakcije se moraju odvijati izolirano.

Trajnost osigurava da nakon što transakcija završi izvršenje, ažuriranja i izmjene baze podataka budu pohranjene i zapisane na disk te da traju i ako dođe do kvara sustava. Ažuriranja se pohranjuju u nepromjenjivu memoriju i postaju trajna. Dakle, učinci transakcije se nikada ne gube.

U cjelini, ACID svojstva pružaju mehanizam za osiguravanje ispravnosti i konzistentnosti baze podataka tako da svaka transakcija predstavlja skupinu operacija koje djeluju kao jedna jedinica, proizvode dosljedne rezultate, djeluju izolirano od drugih operacija i ažuriranja koja čine da su trajno pohranjeni (Kaur, 2022).

3.6.2. Primarni i strani ključ

Kako bi osigurali da su podaci u određenom stupcu jedinstveni koristimo primarni ključ (engl. primary key). Stupac primarnog ključa ne može imati NULL vrijednost. To je stupac koji je specifično generiran od strane baze podataka prema definiranom redoslijedu ili već postojeći stupac. Također, njegova vrijednost se ne može izbrisati iz roditeljske tablice.

Stupac ili skupina stupaca u tablici relacijske baze podataka koja pruža vezu između podataka u dvije tablice naziva se strani ključ (engl. foreign key). Strani ključ je stupac

koji upućuje na stupac druge tablice. Odnosi se na polje u tablici koje je primarni ključ druge tablice (Geeks for geeks, 2020).

3.6.3. Vrste odnosa

Odnosi u relacijskim bazama podataka nam omogućavaju da spriječimo suvišne podatke. Razlikujemo tri vrste odnosa između tablica:

1. jedan prema više (engl. one-to-many)
2. više prema više (engl. many-to-many)
3. jedan prema jedan (engl. one-to-one).

Najčešća vrsta odnosa je jedan prema više. To znači da red u tablici A može imati mnogo podudarnih redaka u tablici B, dok red u tablici B može imati samo jedan podudarni redak u tablici A. Ako je samo jedan od povezanih stupaca primarni ključ ili ima jedinstveno ograničenje, stvara se odnos jedan prema više.

U odnosu više prema više, red u tablici A može imati više podudarnih redaka u tablici B, i obrnuto. Definiranjem spojne tablice (engl. junction table) stvaramo takav odnos. U spojnoj tablici primarni ključ sastoji se od stranih ključeva iz tablice A i tablice B.

Najrjeđa vrsta odnosa je jedan prema jedan. To znači da red u tablici B ne može imati više od jednog podudarnog reda u tablici B, i obrnuto. Ako su oba povezana stupca primarni ključevi ili imaju jedinstvena ograničenja, stvara se odnos jedan prema jedan, što nije čest slučaj (docs.microsoft.com, 2022).

3.6.4. Normalizacija baze podataka

Tehnika dizajna baze podataka koja smanjuje redundantnost podataka i eliminira nepoželjne karakteristike kao što su anomalije umetanja, ažuriranja i brisanja je normalizacija. Dijeljenje većih tablica na manje tablice i povezivanje pomoću odnosa definirani su pravilima normalizacije. Svrha normalizacije je osigurati logičko pohranjivanje podataka i eliminirati ponavljajuće podatke. Edgar Codd, izumitelj relacijskog modela, predložio je teoriju normalizacije podataka uvođenjem prve normalne forme, a teorija je proširena drugom i trećom normalnom formom. Raymond F. Boyce i Codd su se udružili kako bi razvili Boyce-Codd normalnu formu.

Popis normalnih formi u SQL-u:

- 1NF (prva normalna forma)
- 2NF (druga normalna forma)
- 3NF (treća normalna forma)
- BCNF (Boyce-Codd normalna forma)
- 4NF (četvrta normalna forma)
- 5NF (peta normalna forma)
- 6NF (šesta normalna forma)

Normalizacija se najbolje postiže u trećoj normalnoj formi u većini slučajeva. Normalizaciju baze podataka objasnit ćemo kroz primjer videoteke koja održava bazu podataka iznajmljenih filmova. Sve informacije se pohranjuju u jednu tablicu, bez ikakve normalizacije u bazi podataka, kao što je prikazano na Slika 26.

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

Slika 26 – Baza podataka iznajmljenih filmova (Peterson, Guru99, 2022)

Svaka ćelija ili tablica u prvoj normalnoj formi treba sadržavati jednu vrijednost, a svaki zapis mora biti jedinstven.

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

Slika 27 − Tablica u 1NF (Peterson, Guru99, 2022)

U drugoj normalnoj formi imamo dva pravila: da bude u 1NF i da primarni ključ s jednim stupcem funkcionalno ne ovisi ni o jednim podskupu odnosa ključa kandidata.

Ako želimo da naša baza podataka bude u 2NF, moramo podijeliti gornju tablicu.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Slika 28 − Tablica 1 (Peterson, Guru99, 2022)

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Slika 29 − Tablica 2 (Peterson, Guru99, 2022)

Baza podataka podijeljena je u dvije tablice,− Tablica 1 i− Tablica 2. Podatke o članovima sadrži− Tablica 1, a podatke o iznajmljenim filmovima sadrži− Tablica 2. Prvi stupac u tablici je primarni ključ (Membership_ID) za prvu tablicu. Sada smo ostvarili jedinstveno identificiranje zapisa.

Uzrokovanje promjena bilo kojeg stupca koji nije ključ, kada se mijenja stupac koji nije ključ, naziva se tranzitivna funkcionalna ovisnost (engl. transitive functional dependencies). Promjenom drugog stupca (Full Names) koji nije primarni ključ u tablici 1, možemo dovesti do promjene u četvrtom stupcu (Salutation).

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3rd Street 34	Mr.
3	Robert Phil	5th Avenue	Mr.

Change in Name

May Change Salutation

Slika 30 − Tranzitivna funkcionalna ovisnost (Peterson, Guru99, 2022)

Treća normalna forma, kao i druga normalna forma, ima dva pravila: da bude u 2NF i da nema tranzitivnih funkcionalnih ovisnosti. Moramo podijeliti tablicu da bismo 2NF tablicu premjestili u 3NF.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION ID
1	Janet Jones	First Street Plot No 4	2
2	Robert Phil	3rd Street 34	1
3	Robert Phil	5th Avenue	1

Slika 31 − Tablica 1 u 3NF (Peterson, Guru99, 2022)

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Slika 32 − Tablica 2 u 3NF (Peterson, Guru99, 2022)

SALUTATION ID	SALUTATION
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

Slika 33 − Tablica 3 u 3NF (Peterson, Guru99, 2022)

Naša tablica je u 3NF jer nema tranzitivnih ovisnosti. Primjer je sad na razini koja se ne može dalje dekomponirati kako bi se postigle više normalne forme normalizacije u DBMS-u (Peterson, Guru99, 2022).

3.7. Nerelacijske baze podataka

Nerelacijske baze podataka, za razliku od relacijskih baza podataka, pohranjuju podatke na način koji nije tablični. Nerelacijske baze podataka se još zovu i NoSQL baze podataka. NoSQL (engl. Not only SQL) znači ne samo SQL, jer nerelacijske baze podataka možemo koristiti sa ili bez SQL-a. Skalabilne su i prilagodljive te su zato izvrsne za aplikacije s velikim skupovima podataka. Nerelacijske baze podataka pružaju tipove podataka koji su izgrađeni za odgovarajuće modele podataka i optimizirani za veće performanse i visoko funkcionalne API-je. Niz različitih modela podataka koriste za pristup i upravljanje vrstama podataka (Schaffer, Educative, 2021).

Nerelacijske baze podataka vrlo su učinkovite u analizi velike količine nestrukturiranih podataka (Peterson, Guru 99, 2022). Ako programeri žele raditi promjene u bazi podataka, mogu ih napraviti u hodu, bez utjecaja na aplikacije koje koriste bazu podataka što je jedna od prednosti nerelacijskih baza podataka (Matillion, 2022).

3.7.1. Usporedba relacijskih i nerelacijskih baza podataka

U eri velikih računala i poslovnih aplikacija, mnogo prije interneta, oblaka (engl. cloud), mobilnih uređaja i današnjih masovno interaktivnih tvrtki rođeni su relacijski DBMS-ovi. Nadogradnja poslužitelja, odnosno procesora, memorije i pohrane jedini je način da se povećava kapacitet relacijskih baza podataka. Tijekom proteklog desetljeća značajno se promijenio način na koji se mrežne aplikacije bave podacima. Više korisnika pristupa tim podacima i prikuplja se više podataka nego ikada prije.

Nerelacijske baze podataka pojavile su se kao rezultat porasta mrežnih aplikacija i eksponencijalnog rasta interneta (Couchbase, n.d.). Za Googleov BigTable se vjeruje da je prvi nerelacijski sustav baze podataka (Altarade, n.d.).

Pet trendova zbog kojih se tvrtke okreću nerelacijskim bazama podataka, a koji predstavljaju tehničke izazove koji su preteški za većinu relacijskih baza podataka su:

1. Više kupaca ide na Internet.
2. Internet povezuje sve.
3. Veliki podaci postaju sve veći.

4. Aplikacije se sele u oblak.
5. Svijet je postao mobilan (engl. mobile) (Couchbase, n.d.).

Za relacijske baze podataka, koje su temeljene na shemi i teže ih je skalirati, izazov su skalabilnost i performanse. Relacijske baze podataka nisu optimizirane za visoke performanse u aplikacijama u kojima se masivni podaci (engl. massive data) često pohranjuju i obrađuju, unatoč tome što nude dosljednost. Zbog visokih performansi, velike skalabilnosti i lakoće pristupa nerelacijske baze podataka stekle su veliku popularnost, iako im nedostaju značajke koje pružaju dosljednost i pouzdanost (Altarade, n.d.).

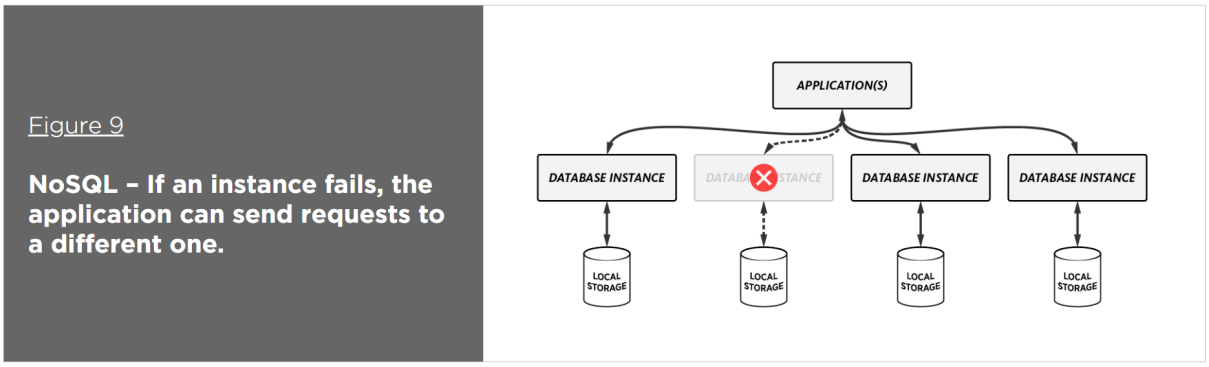
U usporedbi s tradicionalnim, relacijskim bazama podataka, nerelacijske baze podataka imaju brojne prednosti. Temeljna razlika je što nerelacijske baze podataka nemaju shemu, imaju jednostavnu i fleksibilnu strukturu (Altarade, n.d.). Model podataka kod relacijskih baza podataka je fiksiran i definiran statičnom shemom. To je problem jer programeri moraju modificirati shemu ili zatražiti „promjenu sheme“ od administratora baze podataka kako bi promijenili model podataka. Nerelacijske baze podataka ne definiraju statički kako se podaci moraju modelirati i podržavaju agilni razvoj (Couchbase, n.d.).

Proširenje je lakše i jeftinije kada radimo s nerelacijskim bazama podataka, nego kod rada s relacijskim bazama podataka. To je zato što se proširenje radi horizontalnim skaliranjem i raspodjelom opterećenja na sve čvorove, a ne tipom vertikalnog skaliranja koji se obično radi sa sustavima relacijskih baza podataka.

Značajke pouzdanosti, koje su izvorno podržane od strane sustava relacijskih baza podataka, ne podržavaju nerelacijske baze podataka. Nerelacijske baze podataka daju prednost performansama i skalabilnosti nasuprot dosljednosti. Programeri moraju implementirati svoj vlastiti kod kako bi podržali značajke pouzdanosti i konzistentnosti, što dodaje više složenosti sustavu (Altarade, n.d.).

Dostupnost postaje glavna briga jer se sve više angažmana kupaca odvije online putem mrežnih i mobilnih aplikacija. Bez iznimke, ove kritične aplikacije moraju biti dostupne 24 sata na dan i 7 dana u tjednu. Izazov relacijske baze podataka, koje su raspoređene na jednom fizičkom poslužitelju ili se oslanjaju na grupiranje sa zajedničkom pohranom, je omogućavanje dostupnosti 24 sata u danu, 7 dana tjedno. Baza podataka postaje nedostupna ako se implementira kao jedan poslužitelj i ne uspije, ili kao klaster i dijeljena pohrana ne uspije. Dok nerelacijske baze podataka dijele i raspodjeljuju podatke na više instanci baze podataka bez zajedničkih resursa. Radi visoke dostupnosti, podaci se mogu

replicirati na jednu ili više instanci. Nerelacijske baze podataka ne zahtijevaju zasebnu programsku podršku za replikaciju, one su automatske i ugrađene. Ako čvor ne uspije, automatski prelazak na grešku osigurava da baza podataka može nastaviti sa slanjem zahtjeva drugom čvoru (Couchbase, n.d.).



Slika 34 – Slanje zahtjeva drugom čvoru (Couchbase, n.d.)

Dostupnost povećava implementacija baze podataka u više podatkovnih centara i također, pomaže pri oporavku od katastrofe te ima prednost povećanja performansi jer se sva čitanja i upisivanja mogu izvršiti u najbližem podatkovnom centru, čime se smanjuje kašnjenje (Couchbase, n.d.).

Slika 35 prikazuje usporedbu na razini baze podataka, a ne različitih sustava upravljanja bazom podataka koji implementiraju oba modela (Altarade, n.d.).

Feature	NoSQL Databases	Relational Databases
Performance	High	Low
Reliability	Poor	Good
Availability	Good	Good
Consistency	Poor	Good
Data Storage	Optimized for huge data	Medium sized to large
Scalability	High	High (but more expensive)

Slika 35 – Usporedba nerelacijskih i relacijskih baza podataka (Altarade, n.d.)

3.7.2. BASE model

Fleksibilnost i fluidnost s lakoćom manipuliranja podacima omogućio je porast popularnosti nerelacijskih baza podataka te je osmišljen novi model baze podataka koji održava ta svojstva. Akronim BASE znači:

1. uglavnom dostupno (engl. Basically Available)
2. meko stanje (engl. Soft State)
3. postepena konzistentnost (Eventually Consistent).

Nerelacijske baze podataka modelirane BASE-om osigurat će dostupnost podataka širenjem i repliciranjem kroz čvorove klastera (engl. cluster) baze podataka, umjesto da to postane obvezno za trenutnu dosljednost.

Vrijednosti podataka mogu se promijeniti tijekom vremena zbog nedostatka neposredne konzistentnosti. Delegirajući odgovornost na programere, BASE model prekida s konceptom baze podataka koja obvezuje vlastitu dosljednost.

Ne znači da BASE model nikada ne postiže trenutnu dosljednost, zato što je ne obvezuje. Očitanja podataka su moguća, iako možda ne održavaju stvarnost, sve dok ne postigne trenutnu dosljednost.

Programeri bi trebali biti vrlo dobro upućeni u konzistentne podatke i paziti na dosljedne podatke s obzirom na labavu konzistentnost BASE modela (Geeks for geeks, 2022).

3.7.3. Vrste nerelacijskih baza podataka

Četiri su glavne vrste nerelacijskih baza podataka:

- ključ-vrijednost baze podataka
- široko stupčaste baze podataka
- dokumentne baze podataka
- baze podataka usmjerene prema grafovima (Schaffer, Educative, 2021).

Ključ-vrijednost baze podataka, koje se još nazivaju i skladištem ključeva i vrijednosti, jedna su od najjednostavnijih vrsta nerelacijskih baza podataka koje spremaju podatke kao grupu parova sastavljenu od ključa i vrijednosti. Idealne su za procese kao što su upravljanje sesijama za mrežne aplikacije, korisničke sesije za velike online igre i online košarice za kupnju jer su vrlo skalabilne i mogu podnijeti veliku količinu prometa. Primjeri su Amazon DynamoDB i Redis.

Podaci se kod široko stupčastih baza podataka ne pohranjuju u retke i stupce, nego samo u stupce. Idealne su za podršku aplikacijama velikih podataka u stvarnom vremenu jer su visoko skalabilne. Primjeri su BigTable, Apache Cassandra i Scylla (Matillion, 2022).

Podaci se čuvaju u zbirkama dokumenata kod dokumentnih baza podataka i obično se koriste XML, JSON i BSON formati. Koliko god želimo podataka možemo pohraniti u jedan zapis, u bilo kojoj vrsti podataka koju želimo (Peterson, Guru 99, 2022). Primjeri dokumentnih baza podataka su MongoDB, Document DB i ApacheCouchDB (Matillion, 2022).

Teoriju grafova za pohranjivanje, mapiranje i postavljanje upita koriste baze podataka usmjerene prema grafovima. Također, koriste se za analizu međupovezanosti. Baze podataka usmjerene prema grafovima mogu se koristiti i za rudarenje podataka o kupcima s društvenih mreža (Peterson, Guru 99, 2022). Primjeri su Datastax Enterprise Graph i Neo4J (Matillion, 2022).

	Storage Type	Query Method	Interface	Programming Language	Open Source	Replication
Cassandra	Column Store	Thrift API	Thrift	Java	Yes	Async
MongoDB	Document Store	Mongo Query	TCP/IP	C++	Yes	Async
HyperTable	Column Store	HQL	Thrift	Java	Yes	Async
CouchDB	Document Store	MapReduce	REST	Erlang	Yes	Async
BigTable	Column Store	MapReduce	TCP/IP	C++	No	Async
HBase	Column Store	MapReduce	REST	Java	Yes	Async

Slika 36 – Usporedba različitih nerelacijskih sustava za upravljanje bazom podataka (Altarade, n.d.)

4. MongoDB

MongoDB je nerelacijska ili NoSQL baza podataka koja koristi dokumente umjesto tablica ili redaka za pohranu podataka, a razvijen je 2007. godine. Manipulirati podacima možemo u jednoj operaciji baze podataka što nam omogućuje ovakav model.

Dokumenti i datoteke nalik JSON-u (engl. JavaScript Object Notation) podržani za JavaScript su MongoDB dokumenti. Promjena strukture tijekom vremena olakšana je zato što polja dokumenata mogu varirati. MongoDB ne zahtijeva unaprijed definiranu shemu baze podataka, pa se zato smatra da su bez sheme. Ne mora se definirati fiksna struktura zato što MongoDB nema shemu. Mogu ga koristiti administratori i programeri jer je jednostavan za učenje i korištenje. Za sve operacijske sustave kao što su Mac, Linux i Windows i glavne programske jezike, MongoDB ima podršku.

MongoDB nam omogućuje da ispunimo zahtjeve za brzinom i pohranom te nudi veću pouzdanost i učinkovitost. Ima visoku dostupnost, horizontalno skaliranje i geografsku distribuciju s obzirom da je raspodijeljena baza podataka. Za sve ljude koji žele skalirati i brzo razvijati MongoDB je izvrsno rješenje. Omogućuje suradnju velikom broju članova tima i podržava brzi iterativni razvoj.

MongoDB je nerelacijska baza podataka čija je struktura savršeno prikladna za velike količine podataka. Sustavi za upravljanje sadržajem, analitika kupaca, integracija podataka u stvarnom vremenu, upravljanje podacima o proizvodima, mobilnost i skaliranje neki su od najčešćih primjera upotrebe MongoDB-a.

MongoDB ima sljedeća ograničenja:

- Na temelju načina na koji želite pristupiti podacima, možda moramo redovito ažurirati svoje dokumente jer MongoDB ne podržava spajanja.
- Parovi ključ-vrijednost mogu dovesti do velike upotrebe memorije što može rezultirati redundantnošću podataka.
- 16MB je ograničenje veličine dokumenata.
- Transakcije mogu postati kompliciranе zbog toga što se ne poštuju striktno ACID svojstva.

- Ne može implementirati logiku na razini baze podataka jer ne podržava pohranjene procedure.

4.1. Usporedba MySQL-a i MongoDB-a

MySQL je relacijska baza podataka i postoji već neko vrijeme. MongoDB postao je popularan zbog svoje skalabilnosti i zahtjeva za raznolikošću.

MySQL je sustav upravljanja relacijskim bazama podataka (RDBMS) otvorenog koda (engl. open source) koji pohranjuje podatke u tablice i retke. Za prijenos i pristup podacima koristi SQL. Pojednostavljuje upite i korelaciju pomoću JOIN operacije. Podržava višenitnost (engl. multithreading) i slijedi arhitekturu klijent-poslužitelj.

Ima stabilnost, veliku zajednicu i opsežno testiranje. Za mnoge programske jezike uključujući C, Java, C++, Python i PHP, MySQL je dostupan. Mrežne stranice s velikim prometom, aplikacije za e-trgovinu, skladištenje podataka i aplikacije za bilježenje najčešći su slučajevi upotrebe MySQL-a.

MySQL ima sljedeća ograničenja:

- Ako imate podatke koji se ne uklapaju u niti jednu tablicu, morat ćete ponovno dizajnirati strukturu baze podataka kako bi je prilagodili jer su podaci pohranjeni u tablicama.
- Teško je upravljati bazom podataka jer mora biti raspoređena na više poslužitelja.
- Kod baza s velikom količinom podataka MySQL postaje manje učinkovit zbog problema sa skaliranjem.
- Osjetljivost na napade SQL injekcije (engl. SQL injection attacks).

4.1.1. Prikaz podataka

MySQL predstavlja podatke u recima i tablicama, dok MongoDB predstavlja podatke kao JSON dokumente. Dokument je svaki pojedinačni objekt koji pohranimo u MongoDB-u. Oni žive u zbirkama (engl. collections), a zbirke žive unutar baza podataka.

First Name	Last Name	Employee_ID	Status
John	Doe	1234	Active

Slika 37 − Prikaz podataka u MySQL-u (Sulemani, Educative, 2021)

```
{
First Name: 'John',
Last Name: 'Doe',
Employee_ID: 1234,
Status: 'Active'
}
```

Slika 38 − Prikaz podataka u MongoDB-u (Sulemani, Educative, 2021)

4.1.2. Umetanje podataka

Opcija za ugnježđivanje (engl. nesting) ili ugrađivanje (engl. embedding) podataka nije podržana u MySQL-u. Korištenje JOIN-ova može rezultirati većim tablicama s nepotrebnim poljima, što baš i nije poželjno. Ponekad je korištenje naredbe JOIN zahtjevno za performanse i dugotrajno. Možemo ugraditi povezane podatke u MongoDB-u. Ako smatramo da bi dokument mogao previše narasti, imamo mogućnost referenciranja podataka iz drugog dokumenta.

```
{
    id: 13,
    name: ‘John Doe’,
    age: 23,
    address: {
        City: 'New Jersey,
        Street: 'London',
        Zip_code: 9876
    }
}
```

Slika 39 − Umetanje podataka u MongoDB-u (Sulemani, Educative, 2021)

4.1.3. Jezik upita

MongoDB koristi MongoDB jezik upita (engl. Mongo Query Language, skraćeno MQL), dok MySQL koristi SQL.

Odabir podataka u MySQL-u prikazan je dolje. Oznaka * znači da će se prikazati svi podaci iz odabrane tablice.

```
Select * from employee;
```

Slika 40 − Odabir podataka u MySQL-u (Sulemani, Educative, 2021)

U tablici pod nazivom „employee“ umetnut ćemo polja „employee_id“, „department“ i „status“ s vrijednostima 12, „Sales“ i „Active“.

```
INSERT INTO employee (employee_id, department, status)
VALUES (12, 'Sales', 'Active');
```

Slika 41 − Umetanje podataka u MySQL-u (Sulemani, Educative, 2021)

Ažuriranje podataka u MySQL-u prikazano je na slici dolje. U tablici koja se naziva „employee“ promijenili smo vrijednost atributa „department“ u „Finance“ za točno odabrani redak u tablici gdje je ID jednak 14.

```
UPDATE employee SET department = 'Finance' WHERE employee_id = 14;
```

Slika 42 − Ažuriranje podataka u MySQL-u (Sulemani, Educative, 2021)

Odabir podataka u MongoDB-u

```
db.find.employee()
```

Slika 43 − Odabir podataka u MongoDB-u (Sulemani, Educative, 2021)

U MongoDB-u podatke ćemo umetnuti na način da ključu pridružimo vrijednost. Ključu „employee_id“ pridodana je vrijednost „12“, ključu „department“ pridodana je vrijednost „Sales“, a ključu status „Active“.

```
db.employee.insert ({employee_id:'12', department.Sales', status:'Ac
tive'})
```

Slika 44 − Umetanje podataka u MongoDB-u (Sulemani, Educative, 2021)

Ažuriranje podataka u MongoDB-u

```
db.employee.update({employee_id::{$eq:14}},{$set{ department:'Financ
e'}},{multi:true})
```

Slika 45 − Ažuriranje podataka u MongoDB-u (Sulemani, Educative, 2021)

4.1.4. Optimizacija indeksa

Indeksi za optimizaciju koriste se kod obje baze podataka. MySQL pretražuje cijelu tablicu ako ne pronađe relevantan indeks za upit. Ako nema indeksa, MongoDB pretražuje svaki dokument u kolekciji.

Također, MongoDB je nerelacijska ili NoSQL baza podataka i zato je brži od MySQL-a.

4.1.5. Replikacija

Paralelnu reprodukciju iz nekoliko glavnih baza podataka nam kod MySQL-a omogućuje master-slave i master-master replikacija. Dok, MongoDB podržava replikaciju, automatske izbore i ugrađeno dijeljenje. Automatski izbori omogućuju postavljanje sekundarnih baza podataka koje preuzimaju kontrolu ako primarna baza podataka ne uspije, a dijeljenje (engl. sharding) omogućuje horizontalno skaliranje.

MySQL je vertikalno skalabilan, što znači da možemo povećati opterećenje na jednom poslužitelju povećanjem radne memorije (engl. Random Access Memory, skraćeno RAM) ili specifikacija središnje jedinice (engl. Central Processing Unit, skraćeno CPU). MongoDB je horizontalno skalabilan, što znači da možemo stvoriti MongoDB klaster s više poslužitelja dodavanjem više poslužitelja u bazu podataka.

4.2. Kada koristiti MySQL ili MongoDB

Nema jasnog pobjednika kada je u pitanju odabir između MySQL-a i MongoDB-a jer oba služe različitim područjima. Ovisno o našim potrebama i ciljevima projekta odabrat ćemo koju bazu podataka želimo koristiti.

MySQL je dobar izbor ako:

- Naša baza podataka neće se puno povećavati, a tek smo započeli svoj posao.
- Imamo fiksnu shemu baze podataka ili se naša struktura podataka neće mijenjati dugo vremena.
- Imamo visoku stopu transakcija.
- Glavni prioritet je sigurnost podataka.
- Treba nam bolja podrška.

Ako radimo s aplikacijom koja ima strukturirane podatke s jasnom shemom ili zahtijeva transakcije u više redova, onda je MySQL dobar izbor.

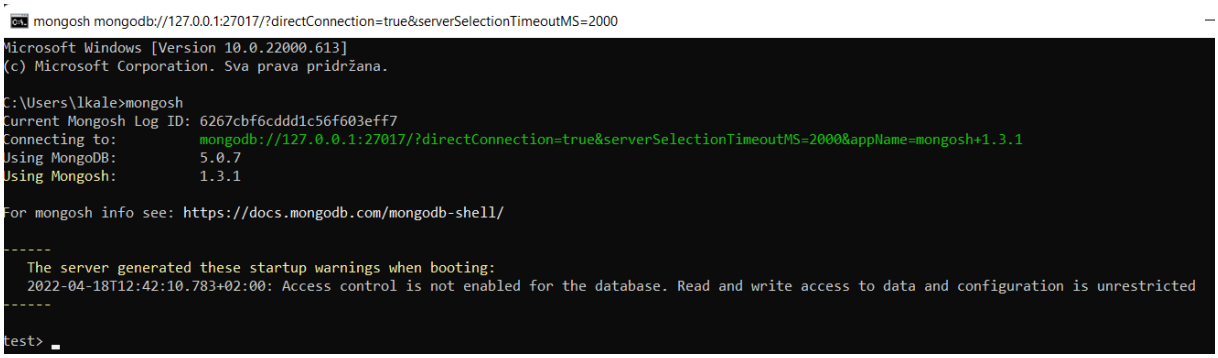
MongoDB je dobar izbor ako:

- Želimo visoku dostupnost podataka uz trenutni i automatski oporavak podataka.
- Želimo smanjiti cijenu migracije sheme i radimo s nestabilnom shemom.
- Usluge se temelje na oblaku.
- Cilj nam je ubrzanje razvoja.

Ako radimo s analitikom u stvarnom vremenu, mobilnim aplikacijama, Internetom stvari (engl. Internet of Things), gdje možda imamo strukturirane ili nestrukturirane podatke koji imaju potencijal za brzi rast, onda je MongoDB pravi izbor (Sulemani, Educative, 2021).

5. Primjena MongoDB u aplikaciji NBA Stats

Prije izrade projekta potrebno je instalirati MongoDB i Mongosh. Mongosh je prevoditelj naredbenog retka (engl. shell) koji omogućuje kreiranje upita i pristup bazi podataka. U terminalu je potrebno upisati mongosh. Koristi se verzija 5 MongoDB-a i verzija 1 Mongosha (Web Dev Simplified, 2021).

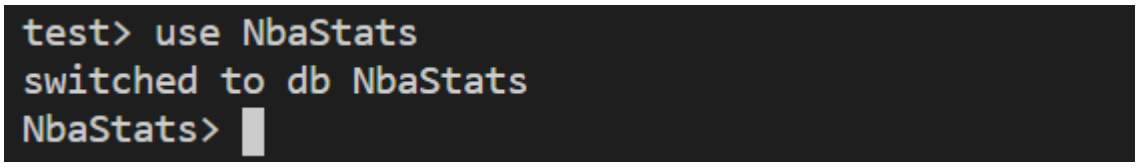


Slika 46 − Mongosh (izvor: autor)

Baza podataka koja se trenutno koristi je test. Naredba show dbs koristi se za pregled baza podataka.

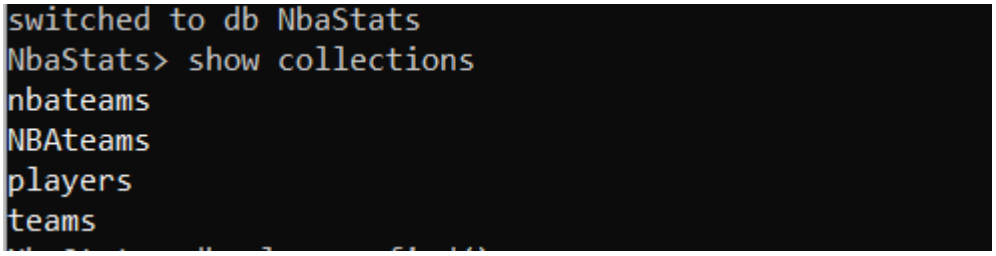
Slika 47 − Naredba show dbs (izvor: autor)

Naredbu use, nakon koje slijedi naziv baze podataka, koristi se ako se želi izgraditi pristup i koristiti bazu podataka.



Slika 48 − Naredba use (izvor: autor)

Naredba show collections služi za prikaz kolekcija koje imamo.



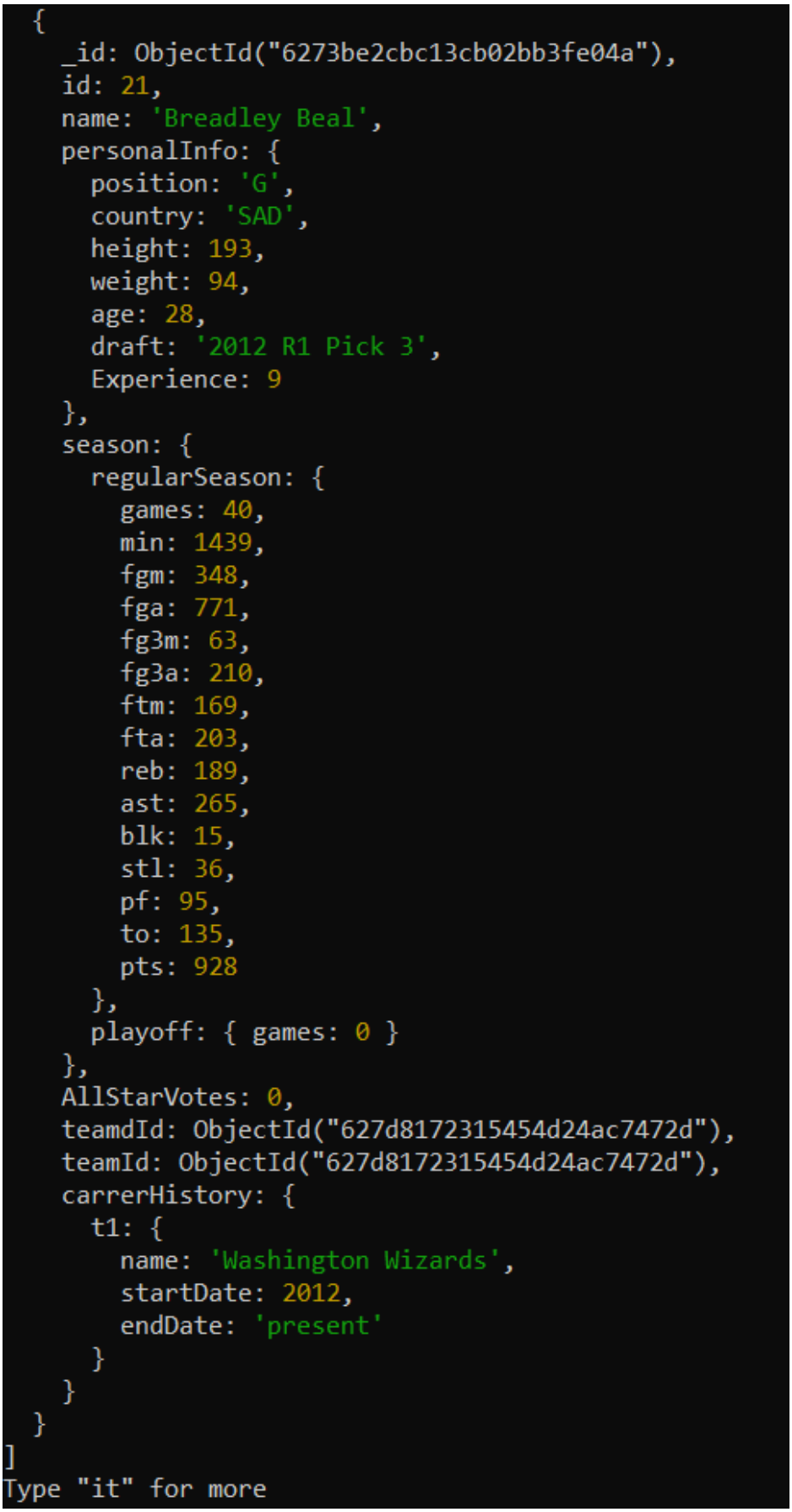
Slika 49 − Naredba show collections (izvor: autor)

Metoda find vraća sve što se nalazi unutar baze podataka.

```
NbaStats> db.players.find()
[
  {
    _id: ObjectId("625d47b9182b4be8f6d15fbf"),
    id: 1,
    name: 'Giannis Antetokounmpo',
    personalInfo: {
      position: 'F',
      country: 'Greece',
      height: 211,
      weight: 110,
      age: 27,
      draft: '2013 R1 Pick 15',
      Experience: '8 years',
      birthdate: 'December 6, 1994'
    },
    season: {
      regularSeason: {
        games: 67,
        min: 2204,
        fgm: 689,
        fga: 1245,
        fg3m: 71,
        fg3a: 242,
        ftm: 553,
        fta: 766,
        reb: 778,
        ast: 388,
        blk: 91,
        stl: 72,
        pf: 212,
        to: 219,
        pts: 2002
      },
      playoff: {
        games: 2,
        min: 74,
        fgm: 21,
        fga: 39,
        fg3m: 1,
        fg3a: 6,
        ftm: 17,
        fta: 29,
        reb: 34,
        ast: 12,
        blk: 4,
        stl: 1,
        pf: 8,
```

Slika 50 – Find metoda (izvor: autor)

Potrebno je upisati „it“ ako se žele prikazati svi igrači u bazi podataka.



Slika 51 − Naredba it (izvor: autor)

MongoDB automatski generira ID u istom trenutku kad se umetne zapis. Programeri ne moraju brinuti o njihovom generiranju. Svaki igrač ima ime, osobne informacije, sezonu, itd.

Budući da ne postoji shema, možemo dodati što god želimo u svoje zbirke. Također, ne moraju imati neke stupce ili ista polja (Web Dev Simplified, 2021). Ali, shemu je moguće definirati pomoću $jsonSchema validatora.

```
NbaStats> db.createCollection("NBAteams", { validator: { $jsonSchema: {required: ["name", "conference", "founded", "court", "city", "coach"]
g"}, founded: {bsonType: "int"}, court: {bsonType: "string"}, city: {bsonType: "string"}, coach: {bsonType: "string"}  }  } }))
{ ok: 1 }
NbaStats>
```

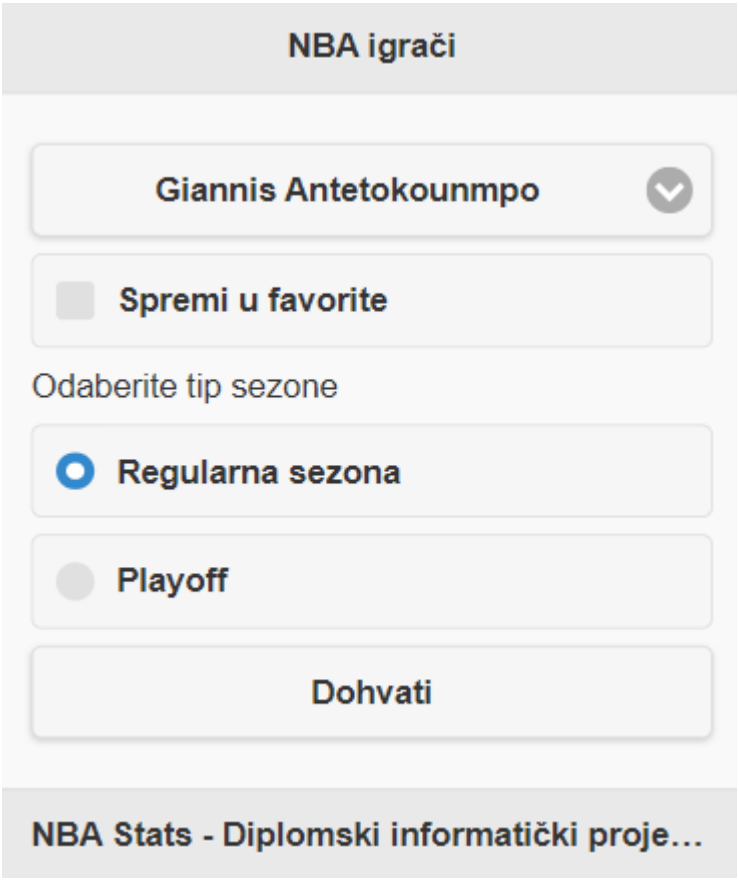
Slika 52 − Kreiranje zbirke uz $jsonSchema (izvor: autor)

Ako se želi izbrisati, umetnuti ili pretražiti dokument koji nije u skladu s definiranom shemom, prikazat će se greška.

```
NbaStats> db.NBAteams.insertOne({name: "Aaaa", city: "bbbb"})
Uncaught:
MongoServerError: Document failed validation
Additional information: {
  failingDocumentId: ObjectId("628168032dd3449cbf50e154"),
  details: {
    operatorName: '$jsonSchema',
    schemaRulesNotSatisfied: [
      {
        operatorName: 'required',
        specifiedAs: {
          required: [ 'name', 'conference', 'founded', 'court', 'city', 'coach' ]
        },
        missingProperties: [ 'coach', 'conference', 'court', 'founded' ]
      }
    ]
  }
}
```

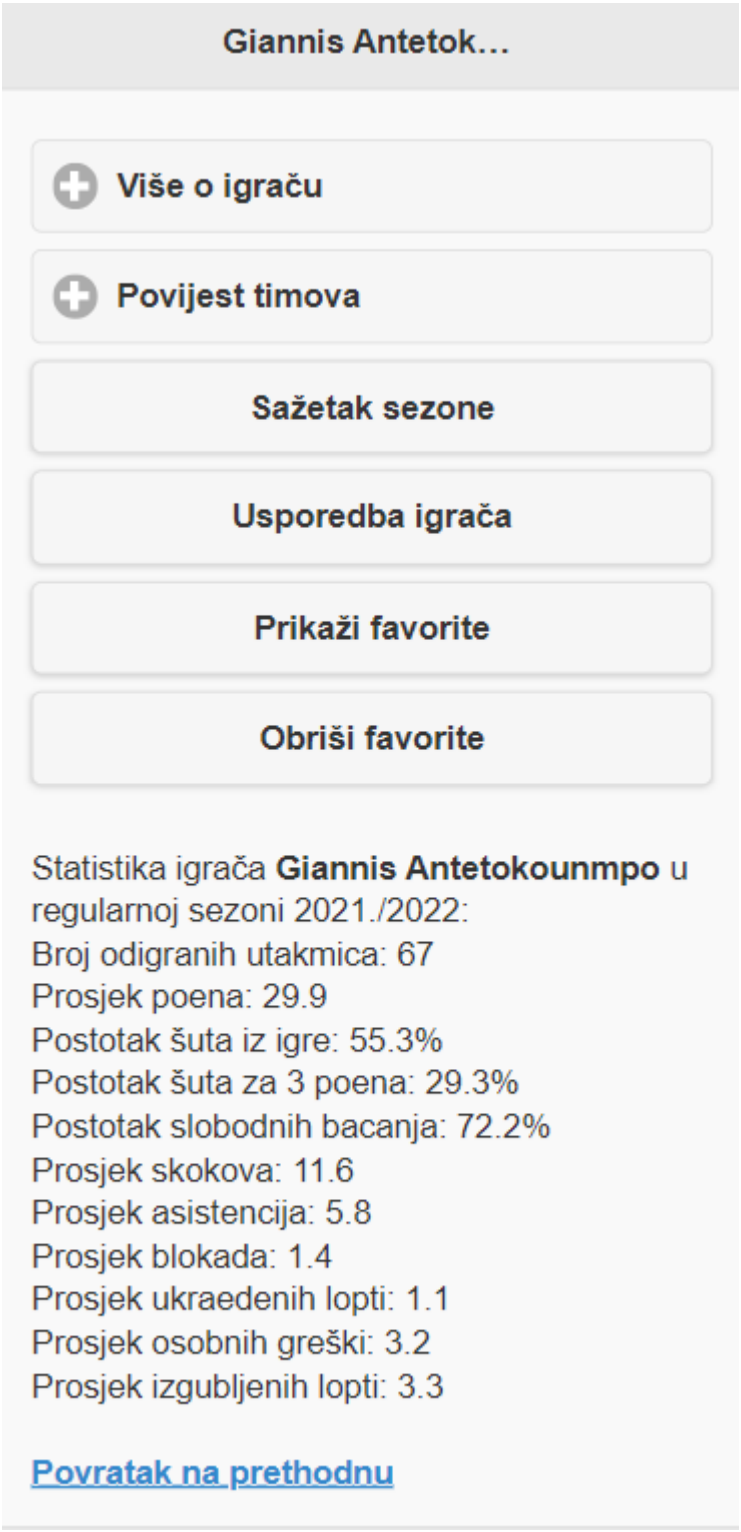
Slika 53 − Greška prilikom provjere valjanosti (izvor: autor)

Na prvoj stranici aplikacije NBA Stats može se odabrati jednog od NBA igrača kao što prikazuje slika dolje. Ako se odabere opcija spremi u favorite, odabrani igrač bit će spremljen kao favorit. Postoji mogućnost odabira između prikaza statistike za regularnu sezonu ili playoff.



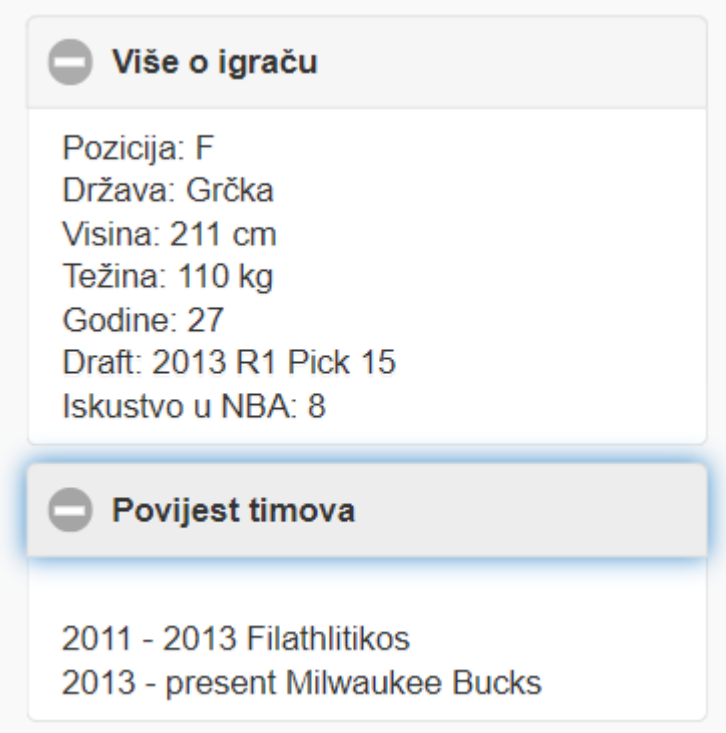
Slika 54 − Odabir igrača (izvor: autor)

Nakon odabira igrača, otvara se nova stranica koja prikazuje statistiku igrača u sezoni 2021./2022.



Slika 55 – Prikaz statistike u sezoni 2021./2022. (izvor: autor)

Osim statistike igrača, u sklopivim sadržajima (engl. collapsible set) više o igraču i povijest timova može se saznati više informacija o igraču i sve timove za koje je dosad nastupao.



Slika 56 − Više informacija o igraču (izvor: autor)

Osim tih informacija, može se pregledati i brisati igrače koji su spremljeni kao favoriti, pogledati sažetak sezone odabranog igrača ili ga usporediti s nekim drugim igračem. Ako se izabere usporedba s drugim igračem otvorit će se nova stranica.

Nakon što se odabere igrač za usporedbu, koji ne smije biti jedino već odabrani igrač, ispisat će se statistika u tablici. Crvenom bojom je obojana statistika igrača koji je bio uspješniji u pojedinoj kategoriji. Giannis Antetokounmpo je imao veći prosjek skokova u sezoni od Kevina Duranta i zato je njegov prosjek skokova obojen crvenom bojom.

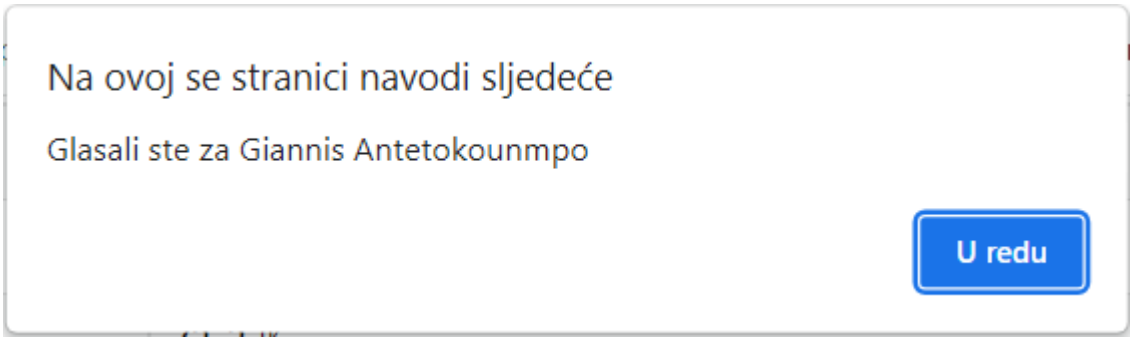
Igrači	Giannis Antetokounmpo	Kevin Durant
Prosjek minuta	32.9	37.2
Prosjek poena	29.9	29.9
Postotak šuta	55.3 %	51.8 %
Šut za 3	29.3 %	38.3 %
Slobodna bacanja	72.2 %	91 %
Prosjek skokova	11.6	7.4
Prosjek asistencija	5.8	6.4
Prosjek ukradenih lopti	1.1	0.9

Slika 57 – Usporedba igrača (izvor: autor)

Nakon prikazane statistike može se odabrati za kojeg igrača želimo glasati da nastupi na All Star utakmici.

Slika 58 – Glasanje za All Star igrača (izvor: autor)

Nakon što se glasa za igrača, ispisat će se poruka i broj glasova u bazi podataka će se tom igraču povećati za jedan.



Slika 59 – Glas za Giannisa Antetokounmpa (izvor: autor)

Zaključak

Poboljšanje poslovanja, pružanje bolje usluge korisnicima, poduzimanje drugih radnji koje u konačnici mogu povećati prihod i dobit neki su od razloga zašto tvrtke koriste velike podatke. Bezbroj je izvora iz kojih dolaze veliki podaci. To mogu biti mobilne aplikacije, društvene mreže, elektronička pošta i slično. Baza podataka zajednička je zbirka srodnih podataka. Relacijske ili SQL baze podataka se temelje na relacijskom modelu podataka koji je predstavio E. F. Codd 1970. godine. Nerelacijske ili NoSQL baze podataka su skalabilne i prilagodljive te su zato izvrsne za aplikacije s velikim skupovima podataka. Nerelacijske baze podataka pojavile su se kao rezultat porasta mrežnih aplikacija i eksponencijalnog rasta interneta. Relacijske baze podataka nisu optimizirane za visoke performanse u aplikacijama u kojima se masivni podaci često pohranjuju i obrađuju, unatoč tome što nude dosljednost. Zbog visokih performansi, velike skalabilnosti i lakoće pristupa nerelacijske baze podataka stekle su veliku popularnost, iako im nedostaju značajke koje pružaju dosljednost i pouzdanost. U usporedbi s tradicionalnim, relacijskim bazama podataka, nerelacijske baze podataka imaju brojne prednosti. Temeljna razlika je što nerelacijske baze podataka nemaju shemu, imaju jednostavnu i fleksibilnu strukturu. Proširenje je lakše i jeftinije kada radimo s nerelacijskim bazama podataka. To je zato što se proširenje radi horizontalnim skaliranjem i raspodjelom opterećenja na sve čvorove, a ne tipom vertikalnog skaliranja koji se obično radi sa sustavima relacijskih baza podataka. Nerelacijske baze podataka daju prednost performansama i skalabilnosti nasuprot dosljednosti. Dostupnost postaje glavna briga jer se sve više angažmana kupaca odvija online putem mrežnih i mobilnih aplikacija. BASE model održava fleksibilnost i fluidnost s lakoćom manipuliranja podacima. Nerelacijske baze podataka modelirane BASE-om osigurat će dostupnost podataka širenjem i repliciranjem kroz čvorove klastera baze podataka, umjesto da to postane obvezno za trenutnu dosljednost. MongoDB koristi dokumente umjesto tablica ili redaka za pohranu podataka. MongoDB nam omogućuje da ispunimo zahtjeve za brzinom i pohranom te nudi veću pouzdanost i učinkovitost. Struktura MongoDB-a je savršeno prikladna za velike količine podataka. Velika količina podataka je raspodijeljena na veliki broj čvorova raspodijeljenog sustava. U projektu su prikazane naredbe za korištenje i prikaz baze podataka, kreiranje i pregled kolekcija te prikaz podataka u MongoDB-u. Također je prikazano kako možemo definirati shemu.

Bibliografija

Altarade, M. (n.d.). *Toptal*. Dohvaćeno iz The Definitive Guide to NoSQL Databases: https://www.toptal.com/database/the-definitive-guide-to-nosql-databases

Apogaeis. (n.d.). Dohvaćeno iz 6 Biggest Challenges in Mobile App Development: https://www.apogaeis.com/blog/6-biggest-challenges-in-mobile-app-development/

Ask any difference. (2022). Dohvaćeno iz Difference Between Cellphone and Smartphone (With Table): https://askanydifference.com/difference-between-cellphone-and-smartphone-with-table/

Bridget, B., & J., B. S. (n.d.). Dohvaćeno iz Big data: https://www.techtarget.com/searchdatamanagement/definition/big-data

Couchbase. (n.d.). Dohvaćeno iz Why NoSQL Databases?: https://www.couchbase.com/resources/why-nosql

Čizmić, M. (31. prosinac 2021). *Zimo*. Dohvaćeno iz Bez velikih iznenađenja: Ovo su najpopularnije mobilne aplikacije u 2021. godini: 8) https://zimo.dnevnik.hr/clanak/bez-velikih-iznenadjenja-ovo-su-najpopularnije-mobilne-aplikacije-u-2021-godini---690438.html

docs.microsoft.com. (14. veljača 2022). Dohvaćeno iz How to define relationships between tables in an Access database: https://docs.microsoft.com/en-us/office/troubleshoot/access/define-table-relationships

Educative. (n.d.). Dohvaćeno iz What is a Database?: https://www.educative.io/courses/database-design-fundamentals/B6VQBZ6NnnW

Geeks for geeks. (28. ožujak 2020). Dohvaćeno iz Difference between Primary Key and Foreign Key: https://www.geeksforgeeks.org/difference-between-primary-key-and-foreign-key/

Geeks for geeks. (10. veljača 2022). Dohvaćeno iz ACID Model vs BASE Model For Database: https://www.geeksforgeeks.org/acid-model-vs-base-model-for-database/?ref=rp

Kaur, A. (26. travanj 2022). *Geeks for geeks*. Dohvaćeno iz ACID Properties in DBMS: https://www.geeksforgeeks.org/acid-properties-in-dbms/?ref=leftbar-rightbar

Keboola. (20. kolovoz 2021). Dohvaćeno iz What is the CAP theorem?: https://www.keboola.com/blog/cap-theorem

Khazaei, H. (n.d.). *Research gate*. Dohvaćeno iz Visualization of CAP theorem: https://www.researchgate.net/figure/Visualization-of-CAP-theorem_fig2_282679529

Kopecky, C. (31. kolovoz 2020). *Educative*. Dohvaćeno iz What is database query? SQL and NoSQL queries explained: https://www.educative.io/blog/what-is-database-query-sql-nosql

Matillion. (2. lipanj 2022). Dohvaćeno iz Database Schema: Types, Examples, and Benefits: https://www.matillion.com/resources/blog/the-types-of-databases-with-examples

Mladenović, M., Mladenović, S., & Zaharija, G. (2018). *Loomen*. Dohvaćeno iz Programiranje mobilnih aplikacija: https://loomen-raz.carnet.hr/course/view.php?id=9914

Nickerson, R. C., & B., M.-D. F. (srpanj 2016). *Scielo*. Dohvaćeno iz Selecting a Stored Data Approach for Mobile Apps: https://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-18762016000300004

Peterson, R. (27. siječanj 2022). *Guru 99*. Dohvaćeno iz What is database? Definition, Meaning, Types with Example: https://www.guru99.com/introduction-to-database-sql.html

Peterson, R. (4. lipanj 2022). *Guru99*. Dohvaćeno iz Relational Data Model in DBMS: https://www.guru99.com/relational-data-model-dbms.html

Peterson, R. (16. travanj 2022). *Guru99*. Dohvaćeno iz What is Normalization in DBMS (SQL)? 1NF, 2NF, 3NF, BCNF: https://www.guru99.com/database-normalization.html

Pexels. (travanj 2022). Dohvaćeno iz Smartphone: https://www.pexels.com/search/smartphone/

Rajput, M. (19. siječanj 2021). Dohvaćeno iz The Role of Big Data in Mobile App Development: https://bigdata-madesimple.com/role-big-data-mobile-app-development/

Schaffer, E. (11. siječanj 2021). *Educative*. Dohvaćeno iz What is a relational database? A deep dive: https://www.educative.io/blog/relational-database-deep-dive

Schaffer, E. (2022). *Educative*. Dohvaćeno iz The complete guide to System Design : https://www.educative.io/blog/complete-guide-to-system-design

Sonal, T., & Paramjeet, D. (n.d.). *Geeks for geeks*. Dohvaćeno iz Introduction of DBMS: https://www.geeksforgeeks.org/introduction-of-dbms-database-management-system-set-1/

Sulemani, M. (7. travanj 2021). *Educative*. Dohvaćeno iz CRUD operations explained: Create, read, update, delete: https://www.educative.io/blog/crud-operations

Sulemani, M. (2. veljača 2021). *Educative*. Dohvaćeno iz MongoDB vs MySQL: Which database to use: https://www.educative.io/blog/mongodb-vs-mysql

Tuteja, S. (28. lipanj 2021). *Geeks for geeks*. Dohvaćeno iz Introduction of 3-Tier Arhitecture in DBMS: https://www.geeksforgeeks.org/introduction-of-3-tier-architecture-in-dbms-set-2/

Web Dev Simplified. (28. rujan 2021). Dohvaćeno iz MongoDB Crash Course: https://www.youtube.com/watch?v=ofme2o29ngU

Wikipedia. (9. lipanj 2021). Dohvaćeno iz Mobilne aplikacije : https://hr.wikipedia.org/wiki/Mobilne_aplikacije

Wikipedia. (7. lipanj 2022). Dohvaćeno iz Smartphone: https://en.wikipedia.org/wiki/Smartphone

Wikipedia. (4. veljača 2022). Dohvaćeno iz Pametni telefon: https://hr.wikipedia.org/wiki/Pametni_telefon

Wikipedia. (11. svibanj 2022). Dohvaćeno iz Mobile app: https://en.wikipedia.org/wiki/Mobile_app

Wikipedia. (2. lipanj 2022). Dohvaćeno iz Mobile app development: https://en.wikipedia.org/wiki/Mobile_app_development

Wikipedia. (20. svibanj 2022). Dohvaćeno iz Database: https://en.wikipedia.org/wiki/Database#Database_management_system

Wikipedia. (21. svibanj 2022). Dohvaćeno iz Relational database: https://en.wikipedia.org/wiki/Relational_database

Skraćenice

DBMS	*Database Management System*	sustav za upravljanje bazom podataka
SQL	*Structured Query Language*	strukturirani jezik za upite
QBE	*Query By Example*	upit po primjeru
URL	*Uniform Resource Locator*	usklađeni lokator sadržaja
MQL	*Mongo Query Language*	MongoDB jezik upita
RAM	*Random Access Memory*	radna memorija
CPU	*Central Processing Unit*	središnja jedinica