Model baze podataka knjižničnog sustava u kontekstu SQL i NoSQL baza podataka - razlike, prednosti, nedostaci, specifičnosti i funkcionalnosti

Kojić, Sunčica

Sveučilište J.J. Strossmayera u Osijeku

Filozofski fakultet

Dvopredmetni diplomski studij informatologije i informacijske tehnologije

Sunčica Kojić

Model baze podataka knjižničnog sustava u kontekstu SQL i NoSQL baza podataka - razlike, prednosti, nedostaci, specifičnosti i funkcionalnosti

Diplomski rad

doc. dr. sc. Tomislav Jakopec

Osijek, 2020.

Sveučilište J.J. Strossmayera u Osijeku
Filozofski fakultet
Odsjek za informacijske znanosti
Dvopredmetni diplomski studij informatologije i informacijske tehnologije

---

Sunčica Kojić

Model baze podataka knjižničnog sustava u kontekstu SQL i NoSQL baza podataka - razlike, prednosti, nedostaci, specifičnosti i funkcionalnosti

Diplomski rad

Društvene znanosti, Informacijske i komunikacijske znanosti
Grana informacijsko i programsko inženjerstvo

Mentor: doc.dr.sc. Tomislav Jakopec

Osijek, 2020.

Prilog: Izjava o akademskoj čestitosti i o suglasnosti za javno objavljivanje

Obveza je studenta da donju Izjavu vlastoručno potpiše i umetne kao treću stranicu završnog odnosno diplomskog rada.

IZJAVA

Izjavljujem s punom materijalnom i moralnom odgovornošću da sam ovaj rad samostalno napravio te da u njemu nema kopiranih ili prepisanih dijelova teksta tuđih radova, a da nisu označeni kao citati s napisanim izvorom odakle su preneseni.
Svojim vlastoručnim potpisom potvrđujem da sam suglasan da Filozofski fakultet Osijek trajno pohrani i javno objavi ovaj moj rad u internetskoj bazi završnih i diplomskih radova knjižnice Filozofskog fakulteta Osijek, knjižnice Sveučilišta Josipa Jurja Strossmayera u Osijeku i Nacionalne i sveučilišne knjižnice u Zagrebu.

U Osijeku, datum

14. 12. 2020.

Sunčica Kojić, 0344002798
ime i prezime studenta, JMBAG

Sažetak

Rad se osvrće na dva sustava koja se koriste za rad s podacima, a ti sustavi su RDBMS i NoSQL baze podataka. Rad se sastoji od tri dijela: uvodni, teorijski i praktični dio. U uvodnom dijelu rada govori se o osnovnim pojmovima baza podataka, odnosno što su baze podataka, koje vrste postoje, što su sustavi za upravljanje bazama podataka i podatkovni modeli te koje vrste podataka postoje i koje su baze podataka bolje, odnosno praktičnije za određenu vrstu podataka. U teorijskom dijelu rada ukratko će se opisati što su relacijske baze podataka i kako su nastale, što je SQL jezik i od kojih je naredbi sastavljen. Zatim će se, također, opisati i NoSQL, odnosno ne samo SQL (engl. *Not only SQL*) baze podataka, zašto se javila potreba za nastanak istih te koje sve vrste NoSQL baza podataka postoje. Treći dio rada je praktičan te će se u tom dijelu govoriti o samoj izradi jednostavne, ali funkcionalne, kako SQL, tako i NoSQL baze podataka za knjižnični sustav. Ukratko će se opisati svaka tablica u SQL bazi podataka, odnosno kolekcija u NoSQL bazi podataka. Na kraju praktičnog dijela rada napraviti će se usporedba izrađenih baza podataka, odnosno njihove ključne razlike, prednosti, nedostaci te specifičnosti i funkcionalnosti.

Ključne riječi: RDBMS, SQL, NoSQL, baze podataka, podaci

Sadržaj

1. Uvod

U današnje vrijeme ljudi su okruženi sa sve većim brojem podataka koje je potrebno organizirati na praktičan i siguran način te ih trajno pohraniti kako bismo ih mogli brzo i precizno dohvatiti i koristiti onda kada nam isti zatrebaju. Najlakši način za organizaciju, ali i pohranu tih podataka upravo su baze podataka. Baze podataka u današnje vrijeme postale su neizostavni dio, kako poslovnog, tako i privatnog života. Pojava relacijskih baza podataka imala je značajan utjecaj na sam razvoj baza podataka, a ove baze podataka svoju popularnost zadržale su i do danas, budući da su i dalje najkorištenija vrsta.[1] Kao što samo ime kaže, u relacijskim bazama podataka, podaci su povezani u relacije, a ti isti podaci pohranjuju se u tablice te se jednom ili više podataka dodjeljuje primarni ključ koji služi za jednoznačno identificiranje atributa u toj tablici. Relacijskim bazama podataka se upravlja pomoću strukturiranog upitnog jezika (SQL) i stoga je čest slučaj da se relacijske baza još nazivaju i SQL baze podataka. Više o relacijskim bazama podataka biti će rečeno u drugom poglavlju rada.

Nadalje, takozvane alternativne baze podataka, odnosno NoSQL baze podataka označavaju relativno novu tehnologiju u radu s podacima te postoji više vrsta istih. Za razliku od tradicionalnih, relacijskih baza podataka, NoSQL baze podataka svoj uspon temelje na rješavanju određenih problema s kojima se susreću relacijske baze podataka. Također, više o NoSQL bazama podataka biti će rečeno u nastavku rada.

Tema rada usporedba je relacijske te NoSQL baze podataka. Također, za lakšu usporedbu izradit će se, kako relacijska, tako i NoSQL baza podataka za knjižnični sustav kako bismo na primjerima izrade mogli vidjeti koja baza podataka je funkcionalnija te koje su njihove razlike, prednosti, ali i nedostaci i specifičnosti. Relacijska baza podataka realizirana je pomoću MySQL RDBMS-a, dok je alat za rad bio MySQL Workbench te multi-platforma otvorenog koda Xampp koja služi za razvijanje i testiranje aplikacija na lokalnom računalu. Za izradu NoSQL baze podataka koristio se besplatan program otvorenog koda MongoDB, odnosno MongoDB Community Server te grafičko korisničko sučelje MongoDB Compass.

[1] Usp. Gaspar, Drazena; Coric, Ivica. Bridging Relational and NoSQL Databases.SAD:IGI Global, 2017. str. 1. URL: https://books.google.hr/books?hl=hr&lr=&id=DDc9DwAAQBAJ&oi=fnd&pg=PR1&dq=Bridging+Relational+and+NoSQL+Databases&ots=QnMsnmgVps&sig=un5eXEQ_E9c3Nu1WOuWLzW0GujQ&redir_esc=y#v=onepage&q=Bridging%20Relational%20and%20NoSQL%20Databases&f=false (2020-11-15)

2. Baza podataka - osnovni pojmovi

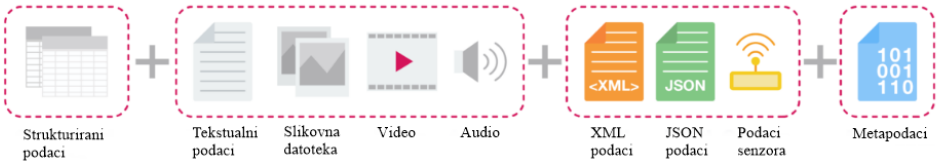
Riječ "podaci" (engl. *"data"*) potječe od latinske riječi "datum" koja u prijevodu znači "dati" (engl. *"to give"*). Podaci podrazumijevaju neku činjenicu ili događaj, odnosno oni su prikaz činjenica, figura, ideja koje mogu biti prikazane u obliku slika, brojeva te riječi. Podaci sami po sebi zapravo ne znače ništa, samo gomilu brojeva, riječi ili slika koje su izvan konteksta. Uzmimo za primjer podatke koji se sastoje od brojeva 7 i 10, a koji mogu značiti bilo što - datum, dob, dobnu skupinu ili mjesec. Međutim, podaci koji se sastoje od 7 muškaraca i 10 žena dati će mnogo više značenja pokazujući kako brojimo određene entitete u nekoj određenoj skupini, grupi ili razredu. Nakon prikupljanja podataka otvara se pitanje što učiniti s obiljem tih podataka te kako bismo ih mogli koristiti potrebno ih je organizirati, pohraniti te staviti na raspolaganje kada su potrebni, a tu se javljaju skupovi podataka, odnosno baze podataka.[2]

Baza podataka podrazumijeva organiziranu zbirku strukturiranih podataka koji su pohranjeni u vanjskoj memoriji računala, a koji su lako dostupni, kako korisnicima, tako i programima. Baze podatka se sastoje od skupa međusobno povezanih podataka koji su pohranjeni zajedno te se podaci u bazi pohranjuju u obliku koji nije ovisan o aplikacijama koje ih koriste, pa se rukovanje istima provodi isključivo kroz zajedničko i nadzirano sučelje.[3]

1.1. Vrste podataka

Kako bismo bolje razumjeli sam pojam baza podataka, potrebno je razumjeti i vrste podataka koje postoje. Iz samog naziva baza podataka možemo vidjeti kako se radi o i s podacima, a te je podatke važno pohranjivati kako bi se mogli iznova koristiti, ali isto tako i analizirati. Kada je riječ o podacima, važno je znati kako nisu svi podaci istog tipa te iste forme te možemo nabrojati tri različite vrste podataka: (1) strukturirani podaci; (2) polu-strukturirani podaci; (3) nestrukturirani podaci.[4]

[2]Usp. Berg, Kristi L.; Seymour, Tom; Goel,Richa. History Of Databases.//International Jurnal of Management & Information Systems 17, 1 (2013), str. 29. URL: https://www.researchgate.net/publication/298332910_History_Of_Databases (2020-09-13)
[3] Usp. Carić, Tonči; Buntić, Mario. Uvod u relacijske baze podataka.Zagreb: 2015., str. 2. URL: http://files.fpz.hr/Djelatnici/tcaric/Tonci-Caric-Baze-podataka.pdf (2020-09-13)
[4]Usp. Data Types: Structured vs. Unstructured data. 2019. URL: https://www.bigdataframework.org/data-types-structured-vs-unstructured-data/ (2020-09-15)



Slika 1. Tri vrste strukture podataka[5]

1.1.1. Strukturirani podaci

Kao što je već ranije rečeno, nisu svi podaci istog formata ili tipa. Prva skupina podataka su strukturirani podaci. Strukturirani podaci su oni podaci koji imaju jasno definiranu strukturu podataka te oni odgovaraju podatkovnom modelu. Podatkovni model određuje izgled podataka, kao i pravila koja takvi podaci moraju zadovoljiti, ali isto tako i koje se operacije mogu izvoditi nad takvim podacima. U strukturirane podatke ubrajamo brojeve, kao i datume, grupe riječi, brojeve koje nazivamo stringovi (npr. korisnikova adresa i sl.) te logičke vrijednosti i binarni podaci. Ovi podaci nazivaju se strukturirani podaci iz razloga jer prate definiranu formu te njihova vrijednost ima smisleni smještaj unutar neke strukture. Strukturirani se podaci skladište u relacijske baze podataka, a najčešći primjer strukturiranih podataka su SQL baze podatka.[6]

1.1.2. Polu-strukturirani podaci

Za razliku od strukturiranih podataka, polu-strukturirani podaci nemaju čvrsto definiranu strukturu. S jedne strane, neki oblici strukturiranih podataka uopće nemaju nikakvu strukturu, dok s druge strane neki imaju ali ona također nije čvrsto definirana te ne postoje točna ograničenja nad podacima. Ponekad je teško odrediti definiranu granicu između ove dvije vrste podataka te kako bi se lakše napravila razlika treba uzeti u obzir činjenicu kako strukturirani podaci imaju fiksna polja i zapise, dok nestrukturirani podaci nemaju ali imaju elemente koji pomažu pri razdvajanju tih podataka u različite hijerarhije. Primjeri polu-strukturiranih podataka uključuju JSON i XML oblike polu-strukturiranih podataka. Polu-strukturirani podaci pogodni

[5] Isto.
[6] Isto.

su za pohranu u NoSQL baze podataka budući da se kod polu-strukturiranog modela podataka podaci spremaju u obliku ključ - vrijednost.[7]

1.1.3. Nestrukturirani podaci

Kao što je već ranije rečeno, strukturirani podaci imaju čvrsto definiranu strukturu i format, a polu-strukturirani podaci nemaju strukturu i format koji je čvrsto definiran, treća vrsta podataka, odnosno nestrukturirani podaci nemaju nikakav format i strukturu. Ovi podaci su podaci koji ili nemaju unaprijed definiran model podataka ili nisu organizirani na unaprijed definiran način.[8] Budući da je od 80 do 90 % podataka koje organizacije prikupljaju i generiraju ne-strukturirano, može se reći kako su ovi podaci zapravo podaci koji se najčešće susreću.[9] Nestrukturirani podaci mogu biti tekstualni ili ne tekstualni, a uobičajeni primjeri nestrukturiranih podataka uključuju audio i video datoteke, slike ili baze podataka koje ne koriste strukturirani upitni jezik SQL te mogu biti generirani kako od strane ljudi tako i od strane strojeva. Nestrukturirani podaci pohranjuju se na više načina, a neki od njih su: (1) u aplikacijama; (2) u NoSQL bazama podataka; te (3) u skladištima podataka, a NoSQL baze podataka poput Mango DB preferirani su izbor za smještaj, upravljanje i korištenje nestrukturiranih podataka.[10]

1.2. Sustav za upravljanje bazom podataka

Sustav za upravljanje bazom podataka (Data Base Management System - DBMS) programski je sustav koji omogućuje definiranje, manipulaciju, te preuzimanje i upravljanje podacima u bazi podataka, a temelji se na odabranom modelu podataka.[11] Kao i svaki drugi sustav, i ovaj ima nekoliko osnovnih zadaća koje mora obavljati, a one su:

a) zaštita od neovlaštenog korištenja objekata baza podataka,

b) očuvanje integriteta podataka,

c) omogućavanje obnove podataka u slučaju gubitka istih,

d) omogućavanje istovremenog, višestrukog pristupa istim podacima u bazi podataka,

e) omogućavanje opisa podataka metapodacima,

---

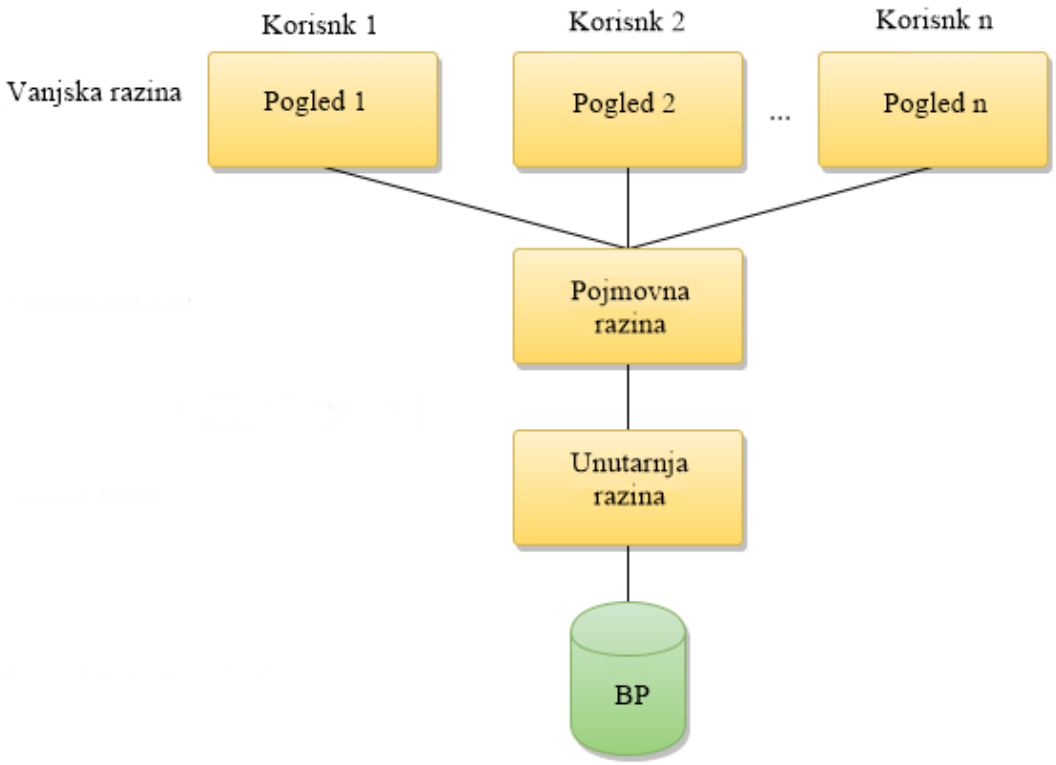
[7] Isto.
[8] Isto.
[9] Usp. Unstructured data. //MongoDB. URL: https://www.mongodb.com/unstructured-data (2020-09-15)
[10] Isto.
[11] Usp. Database Management System (DBMS).// Technopedia. URL: https://www.techopedia.com/definition/24361/database-management-systems-dbms (2020-09-20)

f) identificiranje optimalne strukture za najprikladnije upravljanje podacima,

g) opis, kao i rukovanje podacima.[12]

Nadalje, sustavi baza podataka sastoje se od složenih struktura podataka. Sustav za upravljanje bazom podataka razlikuje tri razine arhitekture baze podataka kao što je prikazano na slici 2, a one su: (1) vanjska razina; (2) pojmovna razina; te (3) unutarnja razina.[13]

Vanjska razina predstavlja korisničku aplikaciju koja služi za pristup i rad s podacima te je ova razina najbliža korisniku. Ova razina je poznata i kao razina pregleda te korisniku prikazuje samo relevantni sadržaj baze podataka, a skriva preostale detalje iz baze podataka koje određenom korisniku nisu potrebne.[14]



Slika 2. Razine arhitekture DBMS[15]

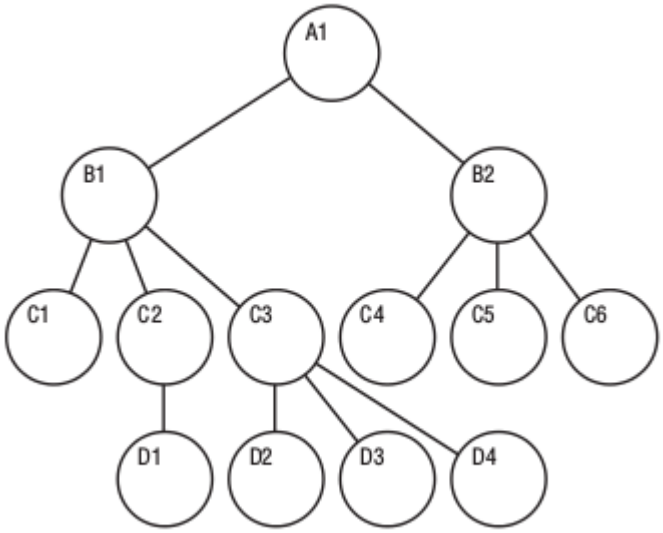
[12] Usp. Carić, Tonči; Buntić, Mario. Uvod u relacijske baze podataka.Zagreb: 2015., str. 3. URL: http://files.fpz.hr/Djelatnici/tcaric/Tonci-Caric-Baze-podataka.pdf (2020-09-13)
[13]Usp. DBMS Schemas: Internal, Conceptual, External.//Guru99. URL: https://www.guru99.com/dbms-schemas.html (2020-09-20)
[14]Isto.
[15]Singh, Chaitanya. DBMS - Three Level Architecture.//BeginersBook. URL: https://beginnersbook.com/2018/11/dbms-three-level-architecture/ (2020-11-15)

Druga razina je pojmovna, odnosno konceptualna razina. Ova razina opisuje strukturu cijele baze podataka te djeluje kao srednji sloj između fizičkog prostora za pohranu i korisničkog prikaza. Na ovoj razini definira se logička struktura baze podataka, odnosno identificira se koji se podaci trebaju pohraniti u bazu podataka, kakav je odnos među tim podacima te kakvi su tipovi podataka i veza među njima. Nadalje, treća i zadnja razina je unutrašnja, a ona je još poznata i kao fizička razina te definira fizičku strukturu pohrane. Unutrašnja razina vodi računa o tome kako i koji su podaci pohranjeni u bazu podataka te također, ova razina obuhvaća i upravljanje radnom memorijom.[16]

1.3. Podatkovni modeli

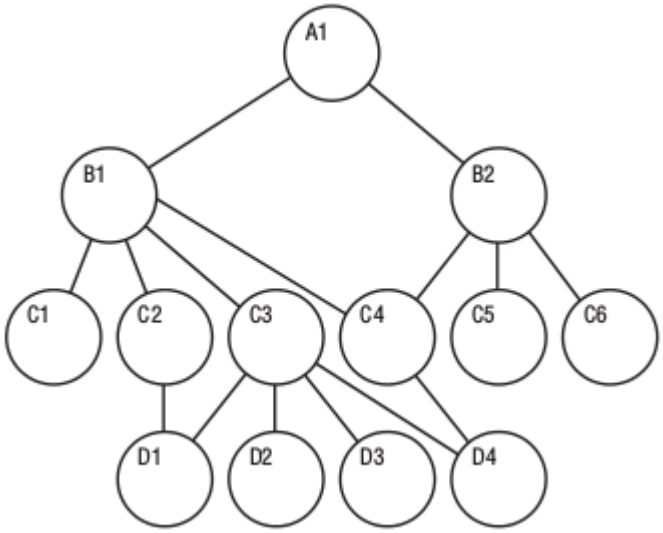
Podatkovni model skup je pravila koja služe za određivanje izgleda logičke strukture baze, a definira načine na koje će podaci biti pohranjeni i ažurirani te kako će im se pristupati u sustavu za upravljanje bazama podataka. Postoje četiri vrste podatkovnih modela, a one su: (1) hijerarhijski podatkovni model; (2) mrežni podatkovni model; (3) objektno orijentirani podatkovni model; te na kraju (4) relacijski podatkovni model.[17] Hijerarhijski model jedan je od najstarijih modela baze podataka koji je razvio IBM, a pojavio se je 60-ih godina prošlog stoljeća. U hijerarhijskom modelu podaci su organizirani obliku stabla kako je prikazano na slici 3. Podaci u hijerarhijskom modelu povezani su na način roditelj - dijete pri čemu svaki roditelj može imati više od jednog djeteta, dok je svako dijete povezano samo s jednim roditeljem.[18]

[16] Isto.
[17] Usp. Carić, Tonči; Buntić, Mario. Uvod u relacijske baze podataka.Zagreb: 2015., str. 4-10. URL: http://files.fpz.hr/Djelatnici/tcaric/Tonci-Caric-Baze-podataka.pdf (2020-09-13)
[18] Isto.



Slika 3. Hijerarhijski podatkovni model[19]

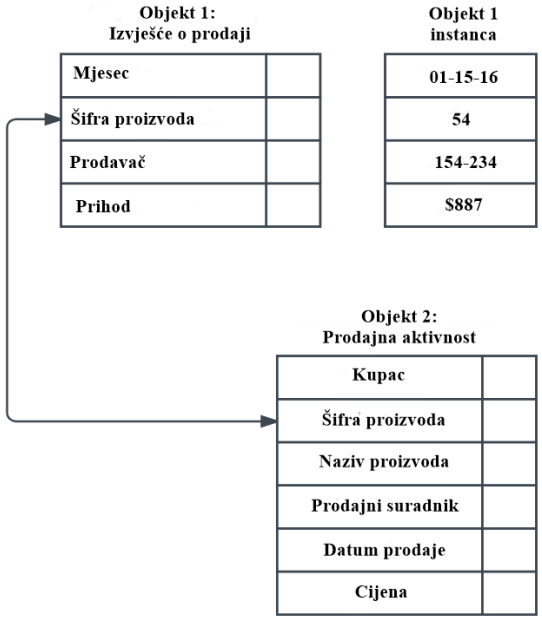
Drugi podatkovni model je mrežni, a njegov je izgled veoma sličan hijerarhijskom modelu podataka te se smatra i modificiranom verzijom hijerarhijskog modela. Međutim, razlika između ova dva modela je ta što u mrežnom podatkovnom modelu, osim što svaki roditelj može imati više od jednog djeteta, također i svako dijete može imati više roditelja kao što možemo vidjeti na slici 4.[20]



Slika 4. Mrežni podatkovni model[21]

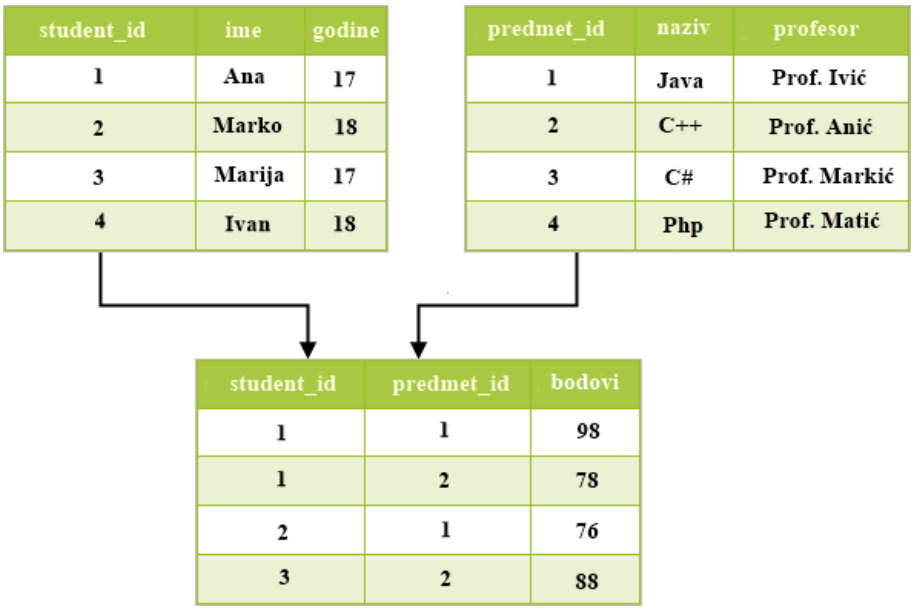
[19]Understanding the Hierarhical Database Model.//MariaDB. URL: https://mariadb.com/kb/en/understanding-the-hierarchical-database-model/ (2020-09-20)
[20] Usp. Carić, Tonči; Buntić, Mario. Uvod u relacijske baze podataka.Zagreb: 2015., str. 4-10. URL: http://files.fpz.hr/Djelatnici/tcaric/Tonci-Caric-Baze-podataka.pdf (2020-09-13)
[21] Understanding the Network Database Model.//MariaDB. URL: https://mariadb.com/kb/en/understanding-the-network-database-model/ (2020-09-13)

Objektno - orijentirani podatkovni model (Slika 5.) pojavio se je 80-ih godina te je ovaj model zasnovan na konceptu objektno - orijentiranih programskih jezika budući da koristi svojstva kao što su nasljeđivanje, višeobličje, apstrakcija te učahurivanje i modularnost. Podaci u objektno - orijentiranom modelu predstavljeni su u obliku objekata koji se, također, koriste u objektno - orijentiranom programiranju.[22] Nadalje, četvrti i zadnji podatkovni model je relacijski, a njega je 1970. godine razvio engleski matematičar Edgar Frank Codd te se ovaj model danas najčešće koristi. U ovom modelu, umjesto da se datoteke oslanjaju na odnos roditelj - dijete, omogućava se da se svaka datoteka poveže s bilo kojom drugom datotekom putem nekog određenog zajedničkog polja. Relacijski se model sastoji od tri glavne komponente, a one su: odnosi, atributi te domene. Odnosi predstavljaju tablice s redovima i stupcima, imenovani stupci relacija nazivaju se atributima te na kraju domena koja predstavlja skup vrijednosti koje atributi mogu poprimiti.[23] Relacijski podatkovni model prikazan je na slici 6.



Slika 5.Objektno orijentirani podatkovni model[24]

[22] Usp. Carić, Tonči; Buntić, Mario. Uvod u relacijske baze podataka.Zagreb: 2015., str. 4-10. URL: http://files.fpz.hr/Djelatnici/tcaric/Tonci-Caric-Baze-podataka.pdf (2020-09-13)
[23] Isto.
[24] What is a Database Model.//Lucidchart. URL: https://www.lucidchart.com/pages/database-diagram/database-models (2020-11-18)

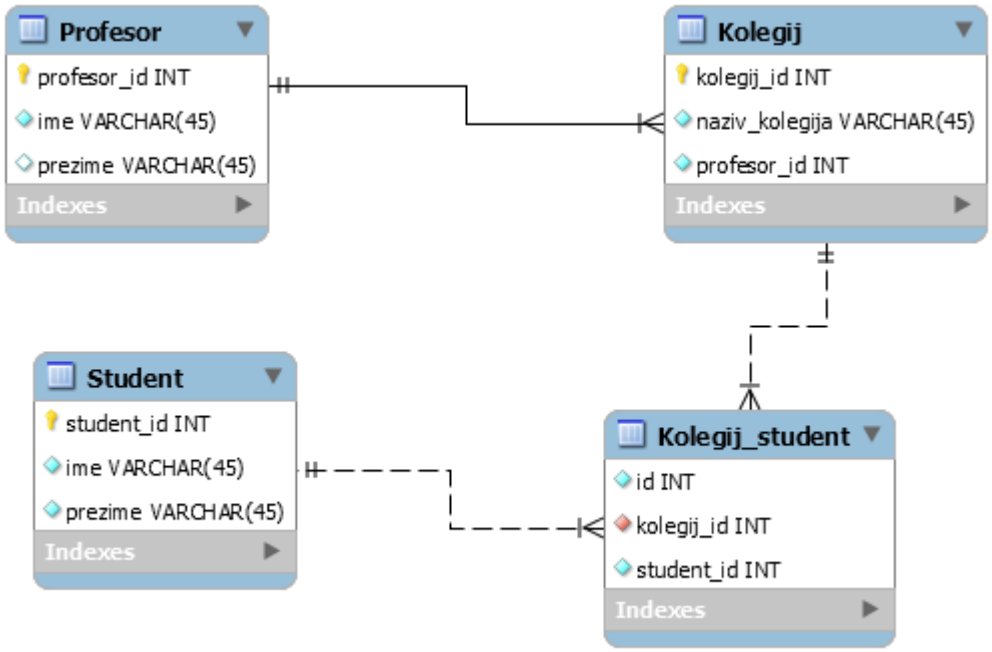


Slika 6. Relacijski podatkovni model[25]

1.4. Modeliranje podataka

Modeliranje podataka tehnika je dokumentiranja programske podrške (engl. *software*) sustava pomoću dijagrama i simbola te se koristi za predstavljanje komunikacije podataka. Kako bismo oblikovali shemu za bazu podataka koja je usklađena s pravilima relacijskog modela potrebno je služiti se jednom pomoćnom fazom koja se naziva modeliranje entiteta i veza (engl. *Entity - Relationship Modeling*). Modeliranje entiteta i veza grafički je prikaz entiteta i njihovih odnosa kao što se vidi na slici 7. Ova shema predstavlja apstrakciju realnog svijeta te zahtijeva da se svijet promatra preko tri komponente, a one su: (1) entiteti; (2) atributi; i (3) veze.[26]

[25]DBMS Database Models.//Studytonight. URL: https://www.studytonight.com/dbms/database-model.php# (2020-09-20)
[26] Usp. Carić, Tonči; Buntić, Mario. Uvod u relacijske baze podataka. Zagreb: 2015., str. 15. URL: http://files.fpz.hr/Djelatnici/tcaric/Tonci-Caric-Baze-podataka.pdf (2020-09-13)



Slika 7. Primjer ER dijagrama

Entitet je prepoznatljivi objekt iz stvarnog svijeta koji postoji, odnosno on je ključni element u svim relacijskim bazama podataka, a on može biti biće (na primjer student), objekt (na primjer auto), te pojava ili događaj (na primjer utakmica, praznik). Entitet se opisuje atributima, a atributi predstavljaju svojstva koja služe za opisivanje entiteta.[27] Ako uzmemo kuću kao primjer entiteta, atributi ovog entiteta predstavljali bi adresu, boju fasade, broj katova i sl.

Nadalje, veze prikazuju povezanost među entitetima, a postoje četiri vrste istih:

- Jedan prema jedan (1:1) - označava povezanost jedne instance jednog entiteta s jednom instancom drugog entiteta.

- Jedan prema više (1:N) - označava povezanost jedne instance jednog entiteta s više instanci drugog entiteta.

- Više prema više (M:N) - označava povezanost više instanci jednog entiteta s više instanci drugog entiteta.

[27] Usp. Carić, Tonči; Buntić, Mario. Uvod u relacijske baze podataka. Zagreb: 2015., str. 15-16. URL: http://files.fpz.hr/Djelatnici/tcaric/Tonci-Caric-Baze-podataka.pdf (2020-09-13)

- Više prema jedan (N:1) - označava povezanost više instanci jednog entiteta s jednom instancom drugog entiteta. [28]

2. Relacijske baze podataka (RDBMS)

Prije nastanka baza podataka, podaci su bili pohranjeni u dokumentima no kako se količina podataka povećavala, pristup tim informacijama bio je spor i neučinkovit. Također, rastom količine podataka, njihovo prikupljanje i čuvanje bilo je sve teže. Kao što je već rečeno, prve baze podataka bile su hijerarhijske i mrežne koje su zamišljene kao mehanizmi za pohranu, no nisu osigurale laku i normalnu tehniku pristupa podacima. Tada su nastale relacijske baze podataka kako bi se omogućila lakša obrada informacija, ali i normalna tehnika pristupanja podacima.[29]

Relacijske baze podataka jedna su od vrsta baza podataka o kojima će se govoriti u ovom radu. Relacijska baza podataka najzastupljeniji je model baze podataka već 40 godina. Ova baza temelji se na relacijskom modelu čije su temeljne ideje iznesene u članku autora E.F.Codd-a "*A relational model for large shared data"* 1970. godine, a programska podrška koja se koristi za održavanje relacijskih baza podataka je relacijski sustav upravljanja bazama podataka, odnosno *RDBMS* (engl. *Relational DataBase Management System*). Autor članka predložio je da osnovna struktura za pohranu i prikaz podataka bude tablica, a da se podacima upravlja pomoću operacija. Nadalje, u knjizi *Bridging Relational and NoSQL Databases* autori Gaspar i Coric opisuju tri osnovna "stupa" (engl. *pillars*) za koje kažu da predstavljaju temelj zajedničke arhitekture relacijskih baza, a oni su: (1) Relacijski model podataka, o kojemu se već govorilo u radu; te (2) ACID transakcije; i na kraju (3) SQL jezik.[30]

U relacijskim bazama podataka podaci se spremaju u tablice te se tablice povezuju relacijama, odnosno vezama. Kako bismo jedinstveno mogli identificirati svaki član relacije, svaka relacija mora imati nešto što se zove primarni ključ (engl. *primary key*). Primarni ključ, kako je već rečeno, koristi se kao jedinstveni identifikator za brzo raščlanjivanje podataka unutar tablice te jedna tablica ne može imati više od jednog primarnog ključa. Primarni ključ mora sadržavati

[28] Usp. Singh, Chaitanya. Entity Relationship Diagram-ER Diagram in DBMS.//BeginersBook. URL: https://beginnersbook.com/2015/04/e-r-model-in-dbms/ (2020-09-23)

[29] Usp. Ali, Wajid...[et al.].Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics.//Asian Jurnal of Research inComputer Science 4, 2. str. 1-8. (2019). URL: https://www.researchgate.net/publication/336686999_Comparison_between_SQL_and_NoSQL_Databases_and_Their_Relationship_with_Big_Data_Analytics (2020-09-23)

[30] Usp. Gaspar, Drazena; Coric, Ivica. Bridging Relational and NoSQL Databases.SAD:IGI Global, 2017. str. 1. URL: https://books.google.hr/books?hl=hr&lr=&id=DDc9DwAAQBAJ&oi=fnd&pg=PR1&dq=Bridging+Relational+and+NoSQL+Databases&ots=QnMsnmgVps&sig=un5eXEQ_E9c3Nu1WOuWLzW0GujQ&redir_esc=y#v=onepage&q=Bridging%20Relational%20and%20NoSQL%20Databases&f=false (2020-09-23)

jedinstvenu vrijednost za svaki red podataka te ne može sadržavati null vrijednost. Osim primarnog ključa, postoji još i strani ključ (engl. *foreing key*), a on služi za stvaranje odnosa između dvije tablice. Svrha stranog ključa je održavanje integriteta podataka te omogućavanje navigacije između dvije različite instance entiteta, a ovaj ključ djeluje kao unakrsna referenca između dvije tablice jer upućuje na primarni ključ druge tablice.[31]

Nadalje, kako su relacijske baze podataka veoma zastupljene u današnje vrijeme, tako postoji mnogo različitih sustava koji koriste SQL jezik. Neki od najpopularnijih sustava su Oracle, MySQL, SQL Server, PostgreSQL, IBM DB2, Microsoft Access, SQL Lite, MariaDB, Informix te Azure SQL.[32]

2.1. ACID transakcije

Transakcija baze podataka svaka je aktivnost koju provodi aplikacijski program koji čita ili ažurira bazu podataka. To je logična cjelina rada koja mora biti u potpunosti dovršena ili u potpunosti prekinuta, odnosno svaka transakcija može imati jedan od dva ishoda: počinjeno ili prekinuto.[33]

U relacijskim bazama podataka transakcije se temelje na ACID principu koji označava četiri svojstva transakcija u bazama podataka, oni su:

- Atomarnost (engl. *Atomicity*): Transakcija mora biti u potpunosti dovršena ili u potpunosti prekinuta.

- Dosljednost (engl. *Consistency*): Jednom kada se transakcija izvrši, trebala bi prijeći iz jednog dosljednog stanja u drugo.

- Izolacija (engl. *Isolation*): Transakcija se treba izvršiti izolirano od ostalih transakcija te ne smije ometati drugu transakciju.

- Trajnost (engl. *Durability*): Nakon uspješnog završetka transakcije, promjene se u bazi podataka trebaju zadržati, čak i u slučaju kvara sustava.[34]

---

[31] Usp. DBMS Keys:Candidate, Super, Primary, Foreign(Example).//Guru99. URL: https://www.guru99.com/dbms-keys.html (2020-09-26)

[32] Usp. Chand, Mahesh. What are the Most Popular Relational Databases.//C#Corner. URL: https://www.c-sharpcorner.com/article/what-are-the-most-popular-relational-databases/ (2020-09-26)

[33] Usp. Gaspar, Drazena; Coric, Ivica. Bridging Relational and NoSQL Databases.SAD:IGI Global, 2017. str. 12. URL: https://books.google.hr/books?hl=hr&lr=&id=DDc9DwAAQBAJ&oi=fnd&pg=PR1&dq=Bridging+Relational+and+NoSQL+Databases&ots=QnMsnmgVps&sig=un5eXEQ_E9c3Nu1WOuWLzW0GujQ&redir_esc=y#v=onepage&q=Bridging%20Relational%20and%20NoSQL%20Databases&f=false (2020-09-23)

[34] Usp. DBMS Transaction Management: What are ACID Properties?.//Guru99. URL: https://www.guru99.com/dbms-transaction-management.html (2020-09-26)

ACID princip osigurava da podaci u bazi podataka ostanu točni i dosljedni unatoč bilo kakvim kvarovima sustava.

2.2. SQL jezik

SQL, odnosno strukturirani upitni jezik (engl. *Structured Query Language*) predstavlja kod koji se obično koristi za rad s relacijskom bazom podataka. SQL, na temelju jezika SEQUEL (engl. *Structured English Query Language*), nastao je u sklopu projekta System R od strane IBM tvrtke. Nakon objave SQL jezika dolazi do njegovog usavršavanja tijekom vremena. Međutim kako su se druge kompanije, osim IBM-a, također bavile razvojem relacijskog modela baze podataka počele su uvrštavati SQL jezik, uz male promjene, u svoje sustave za upravljanje bazom podataka te je nastala potreba za standardizacijom SQL standarda. Tako je 1986. godine donesen prvi standard za SQL jezik od strane Američkog nacionalnog instituta za standarde (*ANSI*, engl. *American National Standards Institute*), a godinu dana kasnije isti standard prihvaća i međunarodna organizacija za standarde (*ISO*, engl. *International Organization for Standardization*). Prihvaćanjem standarda od strane dviju organizacija, on dobiva naziv *ANSI/ISO* standard, a njegova zadnja inačica objavljena je 2011. godine.[35]

SQL jezik sastavljen je od skupa naredbi koji se koriste za manipulaciju, pretraživanje te definiranje podataka, a te naredbe možemo svrstati u sljedeće tri kategorije: (1) *DDL* (engl. *Data Definition Language*)- naredbe za opis podataka; (2) *DML* (engl. *Data Manipulation Language*) - naredbe za manipuliranje podacima; (3) *DQL* (engl. *Data Query Language*) - naredbe za postavljanje upita; te (4) *DCL* (engl. *Data Control Language*) - naredbe za kontrolu pristupa podacima.[36]

- Naredbe za opis podataka: koriste se kreiranje, mijenjanje te brisanje postojećih objekata baze podataka kao što su prikazi, sheme, indeksi i sl. U ovo grupu naredbi pripadaju naredba CREATE koja gradi novu tablicu, naredba ALTER koja postojeću tablicu ili određeni podatak u tablici te naredba DROP koja služi za brisanje tablice, indeksa ili prikaza.[37]

[35] Usp. Manger, Robert. Baze podataka. Zagreb:Element, 2014., str. 29. URL: http://jadran.izor.hr/~dadic/EKO/baze-podataka.pdf (2020-09-26)

[36] Usp. SQL | DDL, DQL, DML, DCL and TCL Commands, 26. kolovoza 2019. URL: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/ (2020-12-07)

[37] Usp. Data Definition Language (DDL).// Technopedia. URL: https://www.techopedia.com/definition/1175/data-definition-language-ddl (2020-09-27)

- Naredbe za manipuliranje podacima: omogućuje različite operacije kao što je umetanje, mijenjanje te brisanje podataka u tablicama. Ova skupina naredbi uključuje naredbe kao što su UPDATE, INSERT te DELETE.[38]

- Naredbe za postavljanje upita: omogućuje dohvaćanje podataka iz baze podataka, a osnovna naredba za zadavanje upita u SQL jeziku je SELECT naredba.[39]

- Naredbe za kontrolu pristupa podacima: uključuje naredbe GRANT i REVOKE koje se uglavnom bave pravima, dozvolama te drugim kontrolama sustava baze podataka.[40]

Nadalje, SQL, kao i drugi upitni jezici, ima svoje definirane tipove podataka, a oni se dijele na: (1) tekstualne; (2) numeričke; (3) tipovi podataka za vrijeme i datum; te (4) binarni tipovi podataka; i na kraju (5) logički tipovi podataka. U Tablici 1 prikazani su najosnovniji te najčešće korišteni tipovi podataka u SQL-u.

Tablica 1. - Prikaz najčešće korištenih SQL tipova podataka.[41]

NAZIV	OPIS	
CHAR(n)	Fiksna duljina zapisa, maksimalno 8000 znakova.	TEKSTUALNI TIPOVI PODATAKA
VARCHAR(n)	Varijabilna duljina zapisa, maksimalno 8000 znakova.	
TEXT	Varijabilna duljina zapisa, maksimalno 2^{31} -1 znakova.	
NCHAR	Fiksna duljina zapisa, maksimalno 4000 znakova.	UNCODE TIPOVI PODATAKA
NVARCHAR(n)	Varijabilna duljina zapisa, maksimalno	

[38] Usp. Usp.Usp. Data Manipulation Language (DML).// Technopedia. URL: https://www.techopedia.com/definition/1179/data-manipulation-language-dml (2020-09-27)

[39] Usp. Query Language.// Technopedia. URL: https://www.techopedia.com/definition/3948/query-language (2020-09-27)

[40] Usp. SQL | DDL, DQL, DML, DCL and TCL Commands, 26. kolovoza 2019. URL: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/ (2020-12-07)

[41] Drkusic, Emil. Learn SQL: SQL Data types, 2020. URL: https://www.sqlshack.com/learn-sql-sql-data-types/ (2020-09-27)

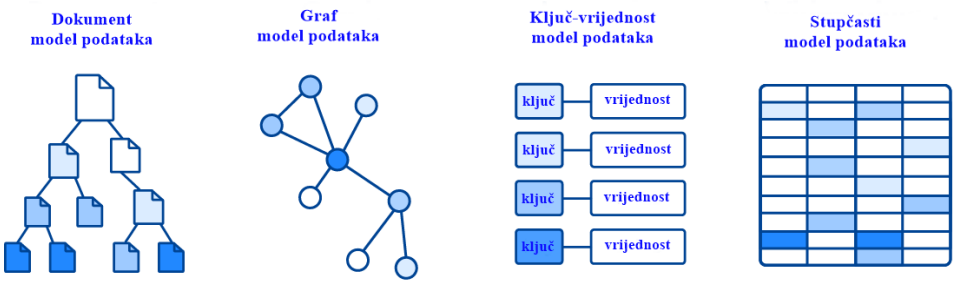
	4000 znakova.	
NTEXT	Maksimalno 2^{31}-1 znakova.	
INT	Cijeli brojevi od -2^{31} do 2^{31}-1.	NUMERIČKI TIPOVI PODATAKA
BIGINT	Cijeli brojevi od -2^{63} do 2^{63}-1.	
SMALLINT	Cijeli brojevi od 0 do 255.	
DECIMAL	Decimalni brojevi od $-10^{38}+1$ do 10^{38}-1.	
DATETIME	Datum, od 1.1.1753. do 31.12. 9999.	DATUM I VRIJEME
SMALLDATETIME	Datum, od 1.1.1900. do 6.6.2079., zaokruženo na minutu.	
TIMESTAMP	Specijalna namjena, najčešće u milisekundama.	
BINARY(n)	Fiksna duljina binarnog zapisa, maksimalno 8000 bytes.	BINARNI TIPOVI PODATAKA
VARBINARY(n)	Varijabilna duljina binarnog zapisa, maksimalno 8000 bytes.	
IMAGE	Maksimalno 2^{31}-1bytes.	
BIT	Cijeli broj koji može biti 0, 1 ili NULL	LOGIČKI TIPOVI PODATAKA

3. NoSQL (Not only SQL) baza podataka

Pojam NoSQL, koji označava "ne samo SQL" (engl. *not only SQL*), po prvi se put pojavio 1998. godine kada je Carlo Strozzi napravio svoju relacijsku bazu podataka otvorenog koda koja nije koristila SQL jezik te ju je nazvao "*Strozzi NoSQL*". Ova vrsta baze podataka pohranjuje svoje tablice u obliku *ASCII* datoteka te ne koristi SQL upite već se bazom upravlja pomoću shell skripte, odnosno tekstualnih datoteka s naredbama koje omogućuju razne operacije u UNIX

sustavima.[42] Iako se NoSQL po prvi put spominje 1998. godine, onakav kakvog ga danas znamo počeo se upotrebljavati tek 2009. godine kada je Eric Evens na konferenciji koja se održala u San Franciscu iskoristio izraz NoSQL za porast korištenja ne relacijskih baza podataka objašnjavajući kako je potrebno pronaći alternativno rješenje za rješavanje problema jer relacijske baze podataka nisu dovoljne.[43]

Nadalje, za razliku od relacijskih baza podataka koje se temelje na tablicama i koriste SQL jezik za postavljanje upita, ne relacijske baze podataka ne koriste SQL jezik za postavljanje upita te se temelje (Slika 8.) na dokument modelu podataka (engl. *document database*), ključ- vrijednost modelu (engl. *key-value stores*), graf modelu podataka (engl. *graph database*) i stupčastim modelom podataka (engl. *column - oriented databases, wide-column stores*).[44]



Slika 8. Vrste NoSQL baza podataka[45]

- Dokument model podataka: pohranjuje podatke u JSON ili XML formatu, a svaki dokument ima vlastiti jedinstveni ključ koji se koristi za dohvaćanje podataka. Primjer takvih baza: Mango DB i CouchDB.

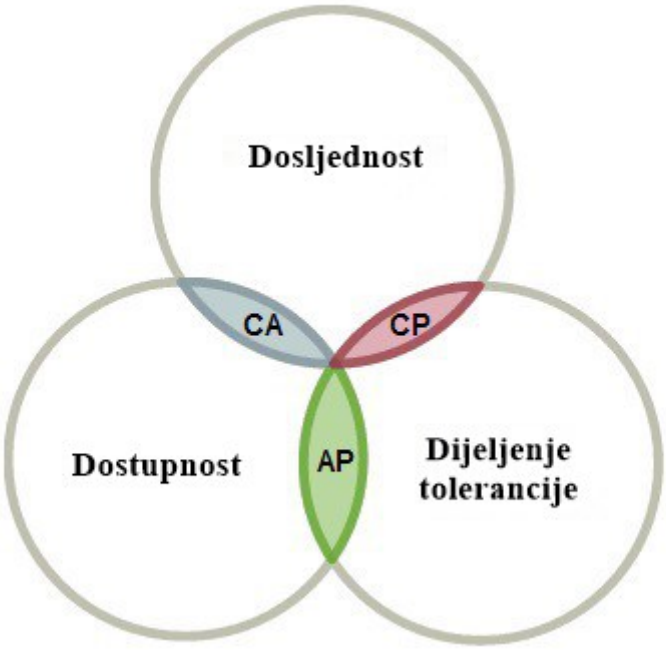
- Ključ - vrijednost model: najjednostavnija vrsta NoSQL baze podataka koja pohranjuje podatke kao kolekciju parova ključeva i pripadajućih vrijednosti. Primjeri takvih baza: Redis i Riak.

- Graf model podataka: ova baza u obliku grafa usmjerena je na odnos između podataka te se svaki element pohranjuje kao čvor. Primjeri takvih baza: Neo4j i AllegroGraph.

[42]Usp. Why NoSQL? Advantages over relational databases, 30. listopad 2018. URL: https://oracle-patches.com/en/databases/3689-why-nosql (2020-10-01)
[43] Usp. Strauch, Christof. NoSQL Databases. Stuttgart: Stuttgart Media Uneversity, 2017., str. 2. URL: https://www.christof-strauch.de/nosqldbs.pdf (2020-10-01)
[44]Usp. Understanding the Different Types of NoSQL Databases. //MongoDB. URL: https://www.mongodb.com/scale/types-of-nosql-databases (2020-10-01)
[45] NoSQL Technologies. //DB BEST Technologies. URL: https://www.dbbest.com/technologies/nosql-databases/ (2020-10-01)

- Stupčasti model podataka: pohranjuje podatke u stupce, a neke od karakteristika ovog modela su sažimanje podataka, skalabilnost te brzina učitavanja podataka. Primjeri takvih baza: BigTable i Cassandra.[46]

Nadalje, za razliku od SQL-a koji radi na ACID principu, NoSQL, zbog nepostojanja sheme te distribuiranosti pristupa, djeluju prema CAP teoremu i BASE principu. CAP teorem, prikazan na slici 9, govori kako je nemoguće zadovoljiti sljedeća tri svojstva istovremeno u distribuiranim sustavima: (1) dosljednost (engl. *Consistency*); (2) dostupnost (engl. *Availability*); te (3) dijeljenje tolerancije (engl. *Partition Tolerance*).[47]



Slika 9. CAP teorem[48]

Osim CAP teorema, uz NoSQL veže se i BASE princip koji se primarno orijentira na dostupnost podataka, za razliku od ACID principa koji je orijentiran na dosljednost podataka. Nadalje, svojstva BASE principa su:

- U osnovi dostupno (engl. *Basically Available*): označava da nam je sustav dostupan čak i ako su neki dijelovi sustava nedostupni.

[46]Usp. What is NoSQL?.//MongoDB. URL: https://www.mongodb.com/nosql-explained (2020-10-01)
[47] Usp.Hussain, Sadequl. SQL and NoSQL Databases Features and Differences. 28. ožujka 2019. URL: https://www.mssqltips.com/sqlservertip/5980/sql-and-nosql-database-features-and-differences/ (2020-10-01)

[48] Nazrul, Syed Sadat. CAP Theorem and Distributed Database Management Systems. 24. travnja 2018. URL: https://towardsdatascience.com/cap-theorem-and-distributed-database-management-systems-5c2be977950e (2020-10-01)

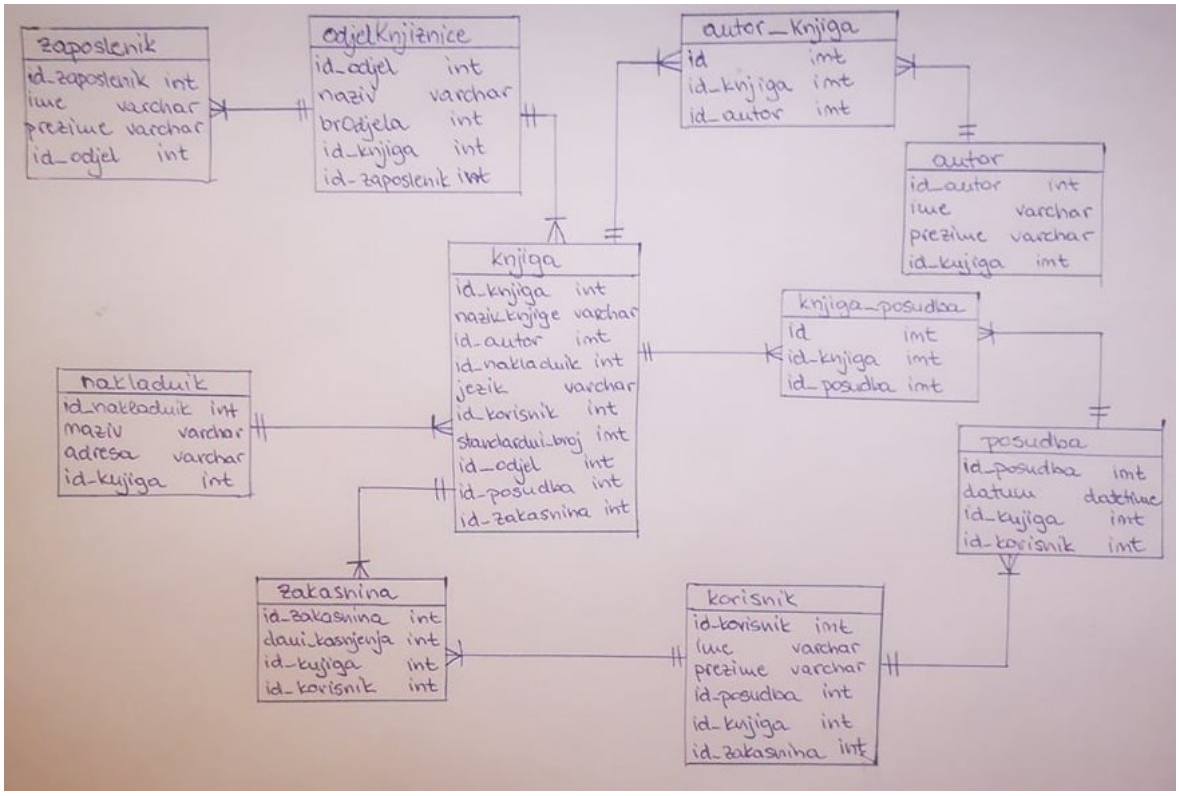
- “Meko” stanje (engl. *Soft state*): označava da se stanje sustava može promijeniti čak i bez unosa novih podataka zbog modela dosljednosti.

- Eventualna dosljednost (engl. *Eventual Consistency*): označava kako će sustav s vremenom postati konzistentan.[49]

4. Izrada relacijske i NoSQL baza podataka

4.1. Izrada relacijske baze podataka

Na samom početku izrade relacijske baze podataka potrebno je osmisliti cjelokupnu shemu baze podataka (Slika 10.) te ju je zatim potrebno implementirat pomoću grafičkog alata za dizajniranje baza podataka MySQL Workbench. Uz grafički alat MySQL Workbench potrebno je koristiti Xampp alat, multi-platforma otvorenog koda, odnosno platformu pogodnu za razvijanje te testiranje aplikacija na lokalnom računalu, a omogućuje Apache poslužitelj, PHP MySQL, MariaDB i druge aplikacije na lokalnom računalu.

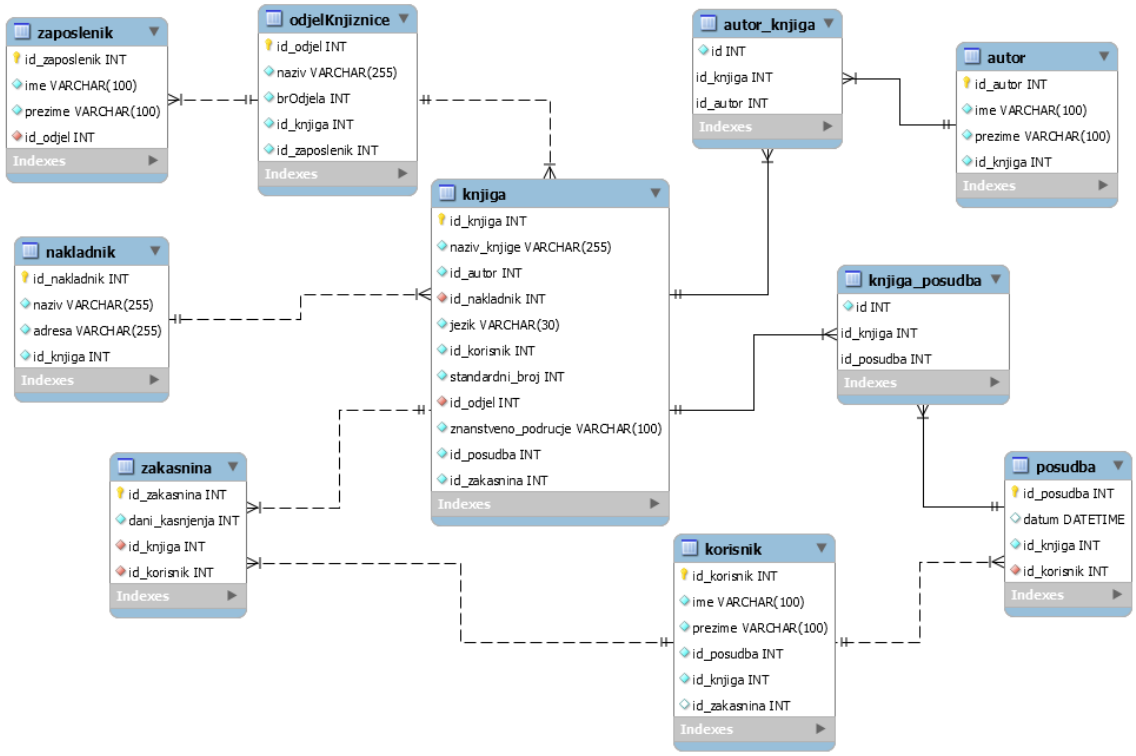


Slika 10. ERA za relacijski model

[49] Usp.Hussain, Sadequl. SQL and NoSQL Databases Features and Differences. 28. ožujka 2019. URL: https://www.mssqltips.com/sqlservertip/5980/sql-and-nosql-database-features-and-differences/ (2020-10-01)

4.2. Relacijska baza podataka “Model knjižnice”

Slika 11. prikazuje grafički prikaz ER modela koji je izrađen u MySQL bazi podataka koristeći alat MySQL Workbench. Baza podataka pod nazivom “Model knjižnice” sastoji se od sveukupno 10 tablica te s odgovarajućim atributima koji su u nastavku objašnjeni. Također, svaki atribut ima označen tip podataka koje određeno polje može primiti. SQL skripta nalazi se u prilogu na kraju rada.



Slika 11. ER model baze podataka "Model knjižnice"

4.2.1. Tablica “knjiga”

Tablica knjiga služi za upisivanje novih knjiga ili uređivanje postojećih. Primarni ključ ove tablice je “id_knjiga”, a vanjski ključevi su “id_korisnik” koji se veže na tablicu “korisnik” i “id_odjel” koji vežu tablicu “odjelKnjiznice”. Atribute entiteta “knjiga” možemo vidjeti u tablici 2.

Tablica 2. - “knjiga”

Ključ	Atribut	Tip podataka	Opis
PK	id_knjiga	int	Šifra knjige
	naziv_knjige	varchar	Naziv knjige
	id_autor	int	Šifra autora knjige
	id_nakladnik	int	Šifra nakladnika knjige
	jezik	varchar	Jezik knjige
VK	id_korisnik	int	Šifra korisnika koji je podigao knjigu
	standardni_broj	int	Jedinstveni identifikator knjige
VK	id_odjel	int	Šifra odjela na kojemu se knjiga nalazi
	znanstveno_podrucje	varchar	Naziv znanstvenog područja kojemu knjiga pripada
	id_posudba	int	Šifra posudbe knjige
	id_zakasnina	int	Šifra zakasnine knjige

4.2.2. Tablica ”knjiga_posudba”

Tablica “knjiga_posudba” naziva se još i tablica sjecište budući da povezuje tablice “knjiga” i “posudba”. Svaka knjiga može se više puta posuditi, a svaka posudba može imati istovremeno više knjiga koje su posuđene. Nadalje, kako većina sustava za upravljanje baza podataka ne podržava više naprema više vezu između tablica, potrebno je stvoriti treću tablicu s odnosno jedan naprema više koja spaja te dvije tablice. Primarni ključ tablice “knjiga_posudba je “id” te su vanjski ključevi “id_knjiga” koji se povezuje s tablicom “knjiga” i “id_posudba” koji povezuje s tablicom “posudba”. Atributi entiteta “knjiga_posudba” vidljivi su u tablici 3.

Tablica 3. - “knjiga_posudba”

Ključ	Atribut	Tip podataka	Opis
PK	id	int	Šifra
VK	id_knjiga	int	Šifra knjige
VK	id_posudba	int	Šifra posudbe

4.2.3. Tablica “posudba”

Tablica “posudba” sadrži podatke o posudbi knjiga te služi za unos novi posudbi. Primarni ključ ove tablice je “id_posudba”koji se veže za vanjski ključ “id_posudba” u tablici “knjiga_posudba”. Vanjski ključ ove tablice je “id_korisnik” koji se veže na primarni ključ tablice “korisnik”. Popis atributa entiteta vidljiv je u tablici 4.

Tablica 4. - “posudba”

Ključ	Atribut	Tip podataka	Opis
PK	id_posudba	int	Šifra posudbe knjige
	datum	datetime	Datum kada je knjiga posuđena
	id_knjiga	int	Šifra posuđene knjige
VK	id_korisnik	int	Šifra korisnika koji je posudi knjigu

4.2.4. Tablica “autor”

Tablica “autor” sadrži osnovne podatke o autorima koji su napisi određenu knjigu te služi za unos novih autora. Primarni ključ tablice je “id_autor” koji se veže na vanjski ključ “id_autor” u tablici “autor_knjiga”. Atribute entiteta “autor” vidljivi su u tablici 5.

Tablica 5. - “autor”

Ključ	Atribut	Tip podataka	Opis

PK	id_autor	int	Šifra autora
	ime	varchar	Ime autora knjige
	prezime	varchar	Prezime autora knjige
	id_knjiga	int	Šifra knjiga određenog autora

4.2.5. Tablica ”autor_knjiga”

Kao tablica “knjiga_posudba, tako je i ova tablica sjecište te služi za povezivanje dvije tablice koje imaju vezu više naprema više. Povezuje dvije tablice, odnosno tablice “knjiga” i “autor”. Primarni ključ ove tablice je “id” te su vanjski ključevi “id_knjiga” koji se povezuje s tablicom “knjiga” i “id_autor” koji povezuje s tablicom “autor”. Atributi entiteta “autor_knjiga” vidljivi su u tablici 6.

Tablica 6. - “autor_knjiga”

Ključ	Atribut	Tip podataka	Opis
PK	id	int	Šifra
VK	id_knjiga	int	Šifra knjige
VK	id_autor	int	Šifra autora koji je napisao knjigu

4.2.6. Tablica “nakladnik”

Tablica “nakladnik” sastoji se od osnovnih podataka o nakladnicima te služi za unos novih nakladnika. Primarni ključ tablice je “id_nakladnik”, a vanjski ključ je “id_knjiga” koji služi za povezivanje s primarnim ključem “id_knjiga” u tablici “knjiga”. Prikaz atributa entiteta “nakladnik” vidljiv je u tablici 7.

Tablica 7. - “ nakladnik”

Ključ	Atribut	Tip podataka	Opis
PK	id_nakladnik	int	Šifra nakladnika
	naziv	varchar	Puni naziv nakladnika (nakladničke kuće)
	adresa	varchar	Adresa nakladnika
VK	id_knjiga	int	Šifra knjige

4.2.7. Tablica "korisnik"

Tablica "korisnik" sadrži podatke o korisnicima koji su učlanjeni u knjižnicu te služi za unos novih korisnika. Primarni ključ tablice je "id_korisnik". Atributi koje sadrži entitet "korisnik" nalaze se u tablici 8.

Tablica 8. - "korisnik"

Ključ	Atribut	Tip podataka	Opis
PK	id_korisnik	int	Šifra korisnika
	ime	varchar	Ime korisnika knjižnice
	prezime	varchar	Prezime korisnika knjižnice
	id_posudba	int	Šifra posudbe knjige
	id_knjiga	int	Šifra knjige koja je posuđena
	id_zakasnina	int	Šifra zakasnine posuđene knjige

4.2.8. Tablica "odjelKnjiznice"

Tablica pod nazivom "odjelKnjiznice" sadrži podatke o odjelu u knjižnici na kojemu se određena knjiga nalazi te služi za unos novih knjiga u odgovarajuće odjele. Primarni je ključ tablice "id_odjel". Popis atributa entiteta "odjelKnjiznice" vidljiv je u tablici 9.

Tablica 9. - ”odjelKnjiznice”

Ključ	Atribut	Tip podataka	Opis
PK	id_odjel	int	Šifra odjela knjižnice
	naziv	varchar	Naziv odjela knjižnice
	brOdjela	int	Broj odjela knjižnice
	id_knjiga	int	Šifra knjige koja se nalazi u određenom odjelu
	id_zaposlenik	int	Šifra zaposlenika koji radi na određenom odjelu

4.2.9. Tablica “zakasnina”

Tablica “zakasnina” sadrži podatke posuđenosti knjiga te služi za unos podataka kako o korisnicima koji su posudili knjigu, tako i o vremenu posuđenosti određene knjige. Primarni ključ tablice “zakasnina” je “id_zakasnina”, a vanjski ključevi su “id_knjiga” koja se povezuje s primarnim ključem tablice “knjiga, te “id_korisnik” koji se povezuje s primarnim ključem tablice “korisnik”. Atributi entiteta “zakasnina” navedeni su u tablici 10.

Tablica 10. - “zakasnina”

Ključ	Atribut	Tip podataka	Opis
PK	id_zakasnina	int	Šifra zakasnine
	dani_kasnjnja	int	Broj dana kašnjenja povrata knjige
VK	id_knjiga	int	Šifra knjige koja ima zakasninu
VK	id_korisnik	int	Šifra korisnika koji ima zakasninu

4.2.10. Tablica “zaposlenik”

Tablica “zaposlenik” sadrži podatke o zaposlenicima koji rade u knjižnici. Primarni ključ tablice je “id_zaposlenik”, a vanjski ključ tablice je “id_odjel” koji se povezuje s primarnim ključem tablice “odjelKnjiznice”. Popis atributa entiteta “zaposlenik” vidljiv je u tablici 11.

Tablica 11. - “zaposlenik”

Ključ	Atribut	Tip podataka	Opis
PK	id_zaposlenik	int	Šifra zaposlenika
	ime	varchar	Ime zaposlenika
	prezime	varchar	Prezime zaposlenika
VK	id_odjel	int	Šifra odjela na kojemu određeni zaposlenik radi

4.3. Izrada NoSQL baze podataka

MongoDB besplatan je program otvorenog koda, odnosno baza podataka koja je orijentirana prema dokumentima. Za izradu NoSQL baze podataka prvo je potrebno instalirati MongoDB Community Server i MongoDB Compass, odnosno grafičko korisničko sučelje. Nakon instalacije, potrebno je otvoriti Naredbeni redak (engl. *command prompt*) i upisati naredbu za provjeru uspješnosti instalacije:

```
mongo --version
```

zatim naredbu:

```
mongod
```

kako bismo pokrenuli server te ostaviti otvoren ovaj Windows terminal u pozadini budući da služi za spajanje na server. Zatim je potrebno otvoriti novi Naredbeni redak (engl. *MongoDB shell*), koji služi za komuniciranje s MongoDB-om, te upisati naredbu:

```
mongo --host localhost:27017
```

kako bismo se spojili na MongoDB Community Server i otvoriti grafičko korisničko sučelje MongoDB Compass koji je instaliran na lokalnom računalu. Kada je sve pokrenuto, možemo započeti s kreiranjem kolekcija.

4.4. NoSQL baza podataka Model knjižnice

Za kreiranje baze podataka u Naredbenom retku (engl. *MongoDB shell*) koristi se naredba *use ime_baze.* Baza podataka za potrebe ovog diplomskog rada nazivati će se Model knjižnice tako da je naredba za kreiranje baze podataka *use model_knjižnice*. NoSQL baza podataka Model knjižnice (Slika 12.) sastoji se od osam kolekcija koje se mogu gledati kao ekvivalenti tablicama u relacijskim bazama podataka.

Kolekcija “knjiga” - omogućuje uvid u glavne atribute knjiga koje se nalaze u knjižnici, odnosno možemo vidjeti naziv, autora i nakladnika knjige, jezik na kojemu je napisana, korisnika koji je knjigu posudio, standardni broj (*ISBN*) knjige, odjel na kojemu se određena knjiga nalazi, te datum kada je posuđena, kao i zakasninu ukoliko korisnik kasni s vraćanjem knjige.

Kolekcija “autor” - omogućuje uvid u glavne atribute autora kao što su ime i prezime autora, te naziv knjige koju je određeni autor napisao.

Kolekcija “korisnik” - sadrži glavne atribute o korisnicima, kao što su ime i prezime korisnika, naziv posuđene knjige, datum kada je korisnik posudio knjigu te dane zakasnine ukoliko je korisnik zaboravio vratiti knjigu do određenog datuma.

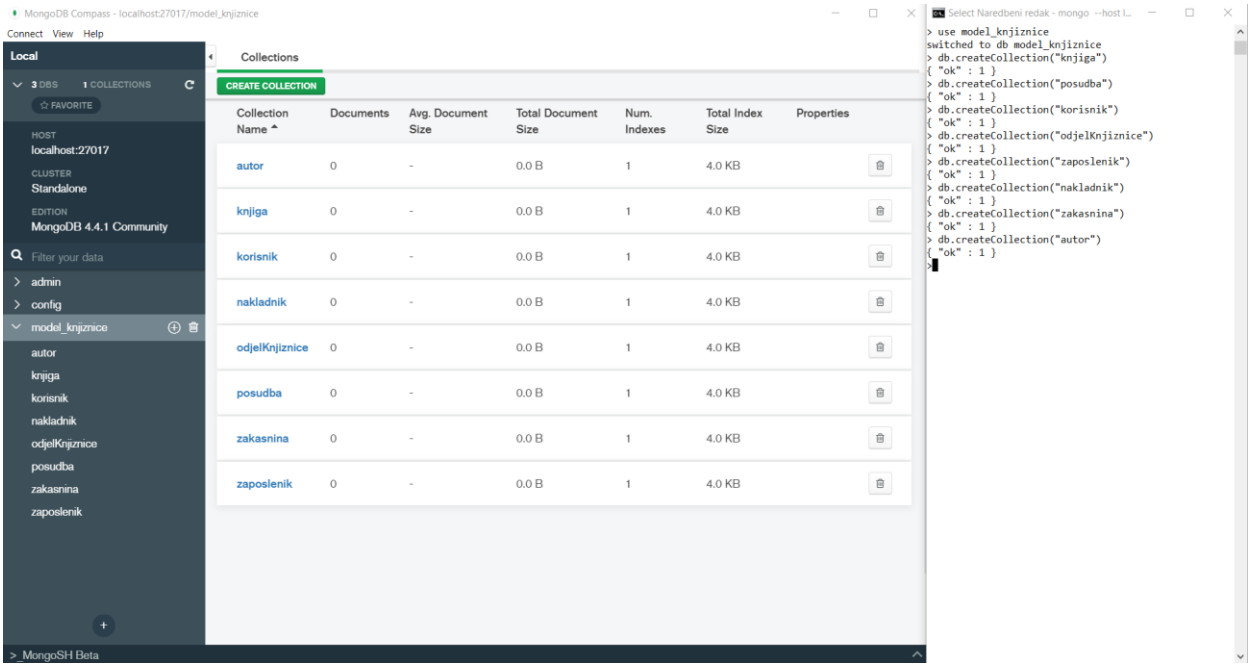
Kolekcija “nakladnik” - omogućuje uvid u glavne atribute nakladnika kao što su naziv nakladničke kuće te adresa, kao i naziv knjige.

Kolekcija “odjelKnjiznice” - sadrži glavne atribute o odjelima knjižnice, odnosno naziv i broj odjela na kojemu se određena knjiga nalazi, naziv knjige te ime i prezime zaposlenika koji radi na određenom odjelu.

Kolekcija “posudba”- sastoji se od atributa kao što su datum kada je knjiga posuđena, naziv knjige te ime i prezime korisnika koji je posudio knjigu.

Kolekcija “zakasnina” - sastoji se atributa kao što su dani kašnjenja s povratkom ukoliko korisnik nije na vrijeme vratio knjigu, naziv knjige koja ima zakasninu te ime i prezime korisnika koji nije vratio knjigu do određenog datuma.

Kolekcija “zaposlenik” - omogućuje uvid u glavne atribute zaposlenika koji rade na određenim odjelima, odnosno ime i prezime zaposlenika te naziv i broj odjela na kojemu zaposlenik radi.



Slika 12. Prikaz izrade baze podataka model_knjiznice i kolekcija

5. Usporedba NoSQL i relacijskih baza podataka

U ovom će poglavlju biti napravljena usporedba relacijskih i NoSQL baza podataka gdje će se opisati njihove razlike, prednosti, nedostaci, specifičnosti te funkcionalnosti.

Glavne razlike između relacijskih i NoSQL baza podataka su tip baze podataka, njihova fleksibilnost i struktura, skalabilnost te upitni jezik. Relacijske baze podataka tablično su orijentirane sa unaprijed strogo definiranom shemom podataka koja ograničava fleksibilnost istih. Osim toga, relacijske baze podataka imaju vertikalnu skalabilnost što znači da se povećava opterećenje na jednom poslužitelju povećanjem komponenta poput RAM-a, SSD-a ili CPU-a. Također, za definiranje i manipulaciju podacima u relacijskim bazama podataka koristi se strukturirani jezik upita (SQL). S druge strane, NoSQL baze podataka nazivaju se još i ne samo relacijskim bazama podataka ili distribuiranim bazama podataka te imaju dinamičnu shemu što

omogućava veliku fleksibilnost ovih baza podataka. Budući da NoSQL baze podataka nemaju strogo definiranu shemu, one mogu imati više strukturiranih oblika što ovisi o vrsti modela podataka koji se koristi (graf model, dokument model, stupčasti model te ključ/vrijednost model). Skalabilnost NoSQL baza podataka je horizontalna što omogućava dodavanje novih poslužitelja ili raspodjelu na više poslužitelja ukoliko se poveća opterećenje. Na kraju, NoSQL baze podataka nemaju deklarativni jezik upita, odnosno za svaki model koristi se drugačiji standardni upitni jezik kao na primjer deklarativni jezik JSON u dokument modelu baza podataka.[50] Osim navedenih razlika, relacijske i NoSQL baze podataka razlikuju se i u upitima koji su pojašnjeni dalje u tekstu. Također, podaci se u relacijskim bazama podataka spremaju u tablice i retke, dok se u NoSQL bazama podataka podaci spremaju u kolekcije i dokumente.

Za kreiranje baze podataka u SQL-u koristi se naredba *create database* uz koju se upisuje naziv baze podataka. Prikaz naredbe *create database* vidljiv je na slici 13.

```
create database model_knjiznice;
```

Slika 13. Naredba *create database* u SQL-u

U NoSQL-u (MongoDB) prilikom kreiranja baze koristi se naredba *use* uz koju se dodaje ime baze podataka, a primjer naredbe vidljiv je na slici 14.

```
> use model_knjiznice
switched to db model_knjiznice
```

Slika 14. Naredba *use* u NoSQL bazi podataka.

Kako bismo kreirali tablicu u relacijskoj bazi podataka potrebno je koristiti naredbu *create table* uz koju se upisuje i ime tablice. Također, odmah pri kreiranju tablice potrebno je unijeti sve atribute u tablicu, kao i njihove tipove podataka te odrediti primarne i vanjske ključeve. Prikaz kreiranja tablice u SQL-u nalazi se na slici 15.

[50] Usp. Smallcombe, Mark. SQL vs NoSQL: 5 Critical Differences. 19. svibnja 2020. URL: https://www.xplenty.com/blog/the-sql-vs-nosql-difference/ (2020-10-14)

```
create table autor(
      id_autor            int not null auto_increment,
      ime                  varchar(100) not null,
      prezime              varchar(100) not null,
      id_knjiga            int not null,
      primary key(id_autor)
      );
```

Slika 15. Prikaz naredbe *create table* u SQL-u.

NoSQL baze podataka ne koriste tablice, već kolekcije. Za razliku od SQL-a u kojemu je potrebno unaprijed definirati atribute te njihove tipove podataka, u NoSQL-u je dovoljno samo kreirati kolekciju, a atributi i tipovi podataka se unose kasnije. Za kreiranje kolekcije koristi se naredba *db.createCollection()* te se u zagradu upisuje naziv entiteta, odnosno kolekcije. Slika 16. prikazuje naredbu za kreiranje kolekcije u NoSQL bazama podataka.

```
> db.createCollection("autor")
{ "ok" : 1 }
```

Slika 16. Naredba za kreiranje kolekcije u NoSQL-u.

Nadalje, nakon što su tablice i kolekcije kreirane potrebno je unijeti podatke u iste. U SQL-u tablice se popunjavaju naredbom *insert* koja je prikazana na slici 17., te se može unijeti samo jedan zapis ili više zapisa odjednom. U NoSQL-u, kolekcije se popunjavaju naredbom *insertOne* te se u zagradu upisuju zapisi koje želimo unijeti u kolekciju. Također, ako želimo unijeti više zapisa odjednom, to se može učiniti naredbom *insertMany.* Naredbe *insertOne* i *insertMany* prikazane su na slici 18.

```
insert into autor (ime, prezime, id_knjiga)
values            ("Meša", "Selimović", 1),
                  ("Ivana", "Brlić-Mažuranić", 2),
                  ("Ivo", "Andrić", 3);
```

Slika 17. Naredba *insert* u SQL-u.

```
  db.autor.insertOne({ime:"Ivo", prezime:"Andrić", knjiga:{naziv_knjige:"Prokleta avlija
"}})
{
        "acknowledged" : true,
        "insertedId" : ObjectId("5fccabd88834a998489b6a0f")
}
> db.autor.insertMany([{ime:"Meša", prezime: "Selimović", knjiga:{naziv_knjige:"Tvrđava"
}}, {ime:"Ivana", prezime:"Brlić-Mažuranić", knjiga:{naziv_knjige:"Regoč"}}])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("5fccabdc8834a998489b6a10"),
                ObjectId("5fccabdc8834a998489b6a11")
        ]
}
```

Slika 18. Naredbe za popunjavanje kolekcija u NoSQL-u.

Kada u SQL-u želimo ažurirati tablicu ili određeni zapis u tablici koristimo naredbe *alter* i *update* (Slika19.). Prilikom korištenja naredbe *alter,* na primjer ažuriranje tablice, upisuje se naziv tablice koju želimo ažurirati te dodajemo atribut i tip podataka. Također, za ažuriranje koriste se i naredbe *set*, *add* i *where*. Naredba *set* koristi se kako bismo odredili koje entitete i atribute želimo ažurirati, naredba *add* koristi se za dodavanje atributa u postojeću tablicu te naredbom *where* filtriramo skup rezultata koji uključuje samo one zapise koji ispunjavaju navedeni uvjet. Prilikom ažuriranja zapisa koristi se naredba *update* te također, upisujemo što želimo ažurirati i na kojem mjestu.

```
alter table korisnik
add godina_rodjenja DATETIME;

update odjelKnjiznice
set brOdjela = 3
where id_odjel = 1;
```

Slika 19. Naredbe *alter* i *update* u SQL-u.

U NoSQL-u za ažuriranje, kako tablice, tako i određenog zapisa, koristi se naredba *updateMany()* te u zagradu upisujemo što i gdje želimo ažurirati. Međutim, prilikom ažuriranja potrebno je koristiti operatore $set i $gt. Pomoću operatora $set mijenjamo vrijednost polja, odnosno zapisa navedenom vrijednošću, dok operator $gt odabire one dokumente kod kojih je vrijednost polja veća od vrijednosti koju smo naveli u naredbi. Slika 20. prikazuje naredbu updateMany() u NoSQL-u.

```
> db.korisnik.updateMany({ }, {$set: {Godina_rodjenja: new Date()}})
{ "acknowledged" : true, "matchedCount" : 3, "modifiedCount" : 3 }
>
> db.odjelKnjiznice.updateMany({brOdjela: {$gt: 2}}, {$set: {naziv: "Odjel za mlade"}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

Slika 20. Naredba *updateMany()* u NoSQL-u.

Nadalje, sljedeće naredbe su naredbe za dohvaćanje određenog ili svih zapisa u bazi podataka. U SQL-u se za dohvaćanje podataka koristi naredba *select* pomoću naredbe *from* koja se koristi za određivanje iz koje tablice želimo odabrati podatke. Naredba *select * from ime_tablice* dohvaća sve podatke koji se nalaze u tablici, a ukoliko želimo dohvatiti samo određeni podatak, moramo specificirati iz koje tablice želimo dohvatiti isti te koje atribute želimo vidjeti. Primjer dohvaćanja jednog i svih podataka u SQL-u prikazan je na slici 21. U NoSQL-u se za dohvaćanje određenog ili svih zapisa koristi naredba *find()*. Kada koristimo naredbu *db.nazivKolekcije.find()* ispisuju nam se svi podaci iz navedene kolekcije. Prilikom korištenja iste naredbe s određenim parametrima, ispisuju nam se samo parametri koje smo napisali i identifikacijski broj (id) dokumenta. Prikaz naredbi za dohvaćanje u NoSQL-u vidljiv je na slici 22.

```
select * from knjiga;

select naziv_knjige, id_autor, jezik, id_posudba
from knjiga;
```

Slika 21. Dohvaćanje podataka u SQL-u.

```
> db.knjiga.find()
{ "_id" : 1, "naziv_knjige" : "Prokleta avlija", "autor" : { "ime" : "Ivo", "prezime" :
"Andrić" }, "nakladnik" : "Školska knjiga", "jezik" : "hrvatski", "korisnik" : { "ime" :
 "Ivo", "prezime" : "Ivić" }, "standardni_broj" : -60743, "odjelKnjiznice" : { "naziv" :
 "Odjel za mlade", "brOdjela" : 4 }, "posudba" : { "datum" : ISODate("1970-01-01T00:00:0
1.990Z") }, "zakasnina" : { "dani_kasnjenja" : 5 } }
>
>
> db.knjiga.find({}, {naziv_knjige: "Prokleta avlija", autor: {ime: "Ivo"}, jezik: "hrva
tski"})
{ "_id" : 1, "autor" : { "ime" : "Ivo" }, "naziv_knjige" : "Prokleta avlija", "jezik" :
"hrvatski" }
>
```

Slika 22. Naredbe za dohvaćanje podataka u NoSQL-u.

Zadnja naredba korištena prilikom izrade ovih dviju baza podataka je naredba za brisanje zapisa iz tablica, odnosno kolekcija. U SQL-u, naredba *truncate* koristi se za brisanje svih podataka iz određene tablice, također, ukoliko želimo obrisati cijelu tablicu koristi se naredba *drop table*. Kako bismo izbrisali određeni podatak iz tablice koristi se naredba *delete* pomoću naredbe *where* kako bismo specificirali gdje se nalazi podatak koji želimo izbrisati. Prikaz naredbi za brisanje jednog, odnosno svih podataka vidljiv je na slici 23.

```
truncate zaposlenik;

delete from korisnik
where id_korisnik = 2;
```

Slika 23. Naredbe za brisanje podataka iz tablica u SQL-u.

U NoSQL-u se za brisanje jednog ili svih podataka koristi naredba *deleteMany()*. Kako bismo izbrisali sve podatke iz određene kolekcije, potrebno je u obične zagrade upisati vitičaste zagrade te ih ostaviti prazne. Za brisanje određenog zapisa, u vitičastoj zagradi specificiramo parametar koji nam pomaže pri lociranju podatka u kolekciji. Na slici 24. prikazane su naredbe za brisanje podataka iz kolekcija. Za brisanje cijele kolekcije koristi se naredba:

db.nazivKolekcije.drop()

```
> db.zaposlenik.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 1 }
>
> db.korisnik.deleteMany({_id: 1})
{ "acknowledged" : true, "deletedCount" : 1 }
```

Slika 24. Naredbe za brisanje podataka iz kolekcija.

Iz prethodnih primjera možemo vidjeti kako se manipulacija podacima ne razlikuje previše osim u sintaksi te da je u SQL-u prilikom kreiranja tablica potrebno odmah definirati atribute s određenim tipovima podataka, dok se u NoSQL-u (MongoDB) to radi u trenutku unosa podataka u kolekcije.

6. Rasprava

Nakon završetka teorijskog dijela ovog rada, započela je izrada relacijske i NoSQL baze podataka kako bi se napravila usporedba istih. Prva na redu bila je relacijska baza podataka za koju je bilo potrebno osmisliti tablice koje su uredno logički povezane te skicirati dijagram baze. Izrada relacijske baze podataka odvijala se brzo te bez previše problema. Alati koje je bilo potrebno instalirati za izradu relacijske baze podataka su grafički alat MySQL Workbench te alat Xampp s kojim se povezuje na server. Nakon instalacije slijedio je sam proces izrada baze podataka pomoću navedenih alata. Prvo su napravljene tablice s odgovarajućim atributima i tipovima podataka, a zatim je izrađen ER model. Problemi na koje se nailazilo tijekom izrade relacijske baze podatak upravo su same relacije između tablice. Odnosno, tablica "zaposlenici" se nije prikazivala tijekom izrade ER modela, a kada se prikazivala nije imala relaciju s tablicom "odjelKnjiznice". Problem se riješio brisanjem tablice "zaposlenik" te ponovo stvaranja iste sa pravilnim primarnim i vanjskim ključevima. Izrada relacijske baze podataka trajala je sveukupno dva dana. Nakon izrade ER modela, u tablice su uneseni podaci kako bi se mogli provoditi upiti potrebni za bolju usporedbu relacijskih baza podataka sa NoSQL bazama podataka.

Nakon završetka izrade relacijske baze podataka slijedila je izrada NoSQL baze podataka. Prije početka izrade NoSQL baze podataka bilo je potrebno istražiti koji su programi i alati potrebni za izradu dokument orijentiranih baza podataka te na koji se način izrađuje ista. Izrada NoSQL baze podataka odvijala se pomoću MongoDB-a te, kao što je ranije u radu navedeno, bilo je potrebno instalirati MongoDB Community Server i MongoDB Compass na osobnom računalu. Nakon instalacije, koja je prošla bez većih problema, slijedilo je povezivanje na server te izrada NoSQL baze podataka. Prilikom izrade dolazilo je do problema tijekom, kako kreiranja baze podataka, tako i postavljanja upita budući da se sintaksa i upitni jezik za NoSQL baze podataka razlikuje od strukturiranog upitnog jezika SQL. Nakon kreiranja baze, kreirane su sve potrebne kolekcije te su uneseni podaci u iste kako bi se mogli provoditi upiti koji su potrebni za usporedbu ovih dviju baza podataka. Nadalje, problemi na koje se nailazilo tijekom kreiranja NoSQL baze podataka bile su sintaksne greške tijekom postavljanja upita. Odnosno javljale su se greške koje su ukazivale na nedostatak zagrada ili dvotočaka, a problem se riješio pažljivijim pisanjem upita. Izrada NoSQL baze podataka trajala je sveukupno pet dana. Unatoč manjim problemima izrada baze podataka, kako relacijske, tako i NoSQL, završila je uspjehom.

6.1. Prednosti i nedostaci

6.1.1. SQL

Prednosti:

Kao jedna od prednosti SQL-a, odnosno relacijskih baza podataka je jednostavnost, kako jednostavnost razvoja baze podataka, tako i korištenje iste. Druga prednost je upitni jezik. Zahvaljujući strukturiranom upitnom jeziku (SQL) ta jednostavnost vidljiva je i tijekom filtriranja podataka, odnosno omogućeno je listanje velikog broja podataka, ali i prikaz samo onih podataka koje smo zatražili određenim upitom. Nadalje, zahvaljujući skalabilnosti relacijske baze podataka pruža se fleksibilnost iste, odnosno omogućena je jednostavnost dodavanja, uklanjanja te mijenjanja željenih podataka bez velikog utjecaja na cjelokupnu bazu podataka. Osim toga, zbog dobro definiranih standarda, relacijske baze podataka omogućuju sigurnost, ali i stabilnost iste. Zadnja prednost ovih baza podataka je unaprijed definirana shema koja nam omogućava lakše upravljanje podacima.

Nedostaci:

Iako se smatra kao prednost radi lakšeg upravljanja podacima, unaprijed definirana shema može biti i veliki nedostatak tijekom izrade same baze, odnosno ukoliko tijekom izrade iste odlučimo dodati novu tablicu, moramo mijenjati cijelu bazu podataka kako bismo uskladili njene relacije. Također, jedan od nedostataka je strogo definirana struktura kod relacijskih baza podataka koja prati ERA model te tijekom unošenja podataka u tablice mora se paziti na tip, veličinu i vrstu podataka koju unosimo u zadanu tablicu, za razliku od NoSQL-a gdje se mogu unijeti bilo kakvi podaci budući da ova baza podataka prima podatke bez obzira na njihovu strukturu.

6.1.2. NoSQL

Prednosti:

NoSQL baze podataka nude nekoliko prednosti u odnosu na relacijske baze podataka, a jedna od tih prednosti upravo je mogućnost upravljanja velikim količinama, kako strukturiranih, tako i polu-strukturiranih i ne-strukturiranih podataka. Budući da nema strogo definirane sheme, NoSQL baze podataka nude veću fleksibilnost koja omogućava lakši razvoj, ali i korištenje iste. Također, za razliku od SQL-a gdje se sve tablice moraju posebno kreirati sa svojim zadanim

atributima i tipovima podataka, u NoSQL-u ukoliko koristimo kolekciju koja ne postoji, ona će biti automatski stvorena.

Nedostaci:

Nedostatak NoSQL baza podataka je kompleksnost tijekom unošenja podataka u istu, odnosno vrijeme pisanja pojedinih naredbi. Uzmimo za primjer kolekciju "knjiga" u kojoj se nalazi 10 atributa, odnosno dokumenata. U relacijskim bazama podataka tijekom unosa podataka u tablice dovoljno je, za određene atribute, unijeti identifikacijski broj (ID) što olakšava i smanjuje vrijeme unosa podataka. S druge strane, u NoSQL bazama podataka tijekom unosa podataka u kolekciju potrebno je posebno navoditi sve podatke kao što je vidljivo na primjeru ispod teksta budući da NoSQL baze podataka nemaju primarne i vanjske ključeve.

```
db.knjiga.insertOne({_id: 1, naziv_knjige: "Prokleta avlija", autor: {ime: "Ivo", prezime: "Andrić"}, nakladnik: "Školska knjiga", jezik: "hrvatski", korisnik: {ime: "Ivo", prezime: "Ivić"}, standardni_broj: 978-953-0-60764-4, odjel: {naziv:"Odjel za mlade", brOdjela: 4}, posudba:{datum: new Date(2020-11-19)}, zakasnina: {dani_kasnjenja: 5}})
```

7. Zaključak

Rad je prikazao dva različita pristupa kada je riječ o upravljanu podacima, a to su relacijske baze podataka te tehnologije NoSQL. Cilj rada bio je osmisliti i izraditi jednostavnu, ali funkcionalnu bazu podataka za vođenje knjižnice, a zatim ju realizirati u relacijskoj, odnosno SQL te NoSQL bazi podataka. Također, napravljena je usporedba ovih dviju baza podataka, kako teorijski, tako i na konkretnom primjeru.

Iako izrađeni modeli nisu dovoljno složeni kako bi prikazali detaljnije razlike u performansama, ipak je moguće vidjeti osnovne teorijske razlike osnovnih funkcija i svojstava pojedine baze, način pristupanja radu s ovim alatima, osnovne principe te sintaksu za pisanje kako MongoDB, tako i relacijskih baza podataka. Nadalje, kod relacijskih baza podataka podaci se smještaju unutar relacija koje imaju strogo definiranu strukturu te se shema podataka mora unaprijed osmisliti prije same izrade baze, dok NoSQL baze podataka nemaju strogo definiranu shemu te su puno fleksibilnije i omogućavaju izbor između raznih modela podataka. Također, relacijske baze podataka najzastupljeniji su model baza podataka, a zbog strogo definirane strukture pružaju veću sigurnost u očuvanju podataka. S druge strane, iako su NoSQL baze podataka nova tehnologija s određenim nedostacima, one su ipak pogodnije za rad s polu-strukturiranim i ne-strukturiranim podacima što nije slučaj kod relacijskih baza podataka.

Izbor baze podataka ovisi ponajviše o njenim potrebama. S jedne strane, ukoliko je riječ o aplikaciji koja će se koristiti samo unutar neke određene organizacije tada će relacijska, odnosno SQL baza podataka vrlo vjerojatno zadovoljiti većinu njenih potreba. S druge strane, NoSQL baze podataka će imati više smisla ukoliko se radi o bazi podataka kod koje se očekuje nagli porast, kako u broju korisnika, tako i u količini podataka.

Za izradu modela baze podataka za knjižnicu ne može se reći koja od ovih dviju tehnologija baza podataka bi bila bolja i praktičnija budući da se dosta razlikuju. Zaključno, na osnovu izrađenih primjera vidimo kako svaka baza podataka ima svoje određene specifičnosti, funkcionalnosti, prednosti, ali i nedostatke, stoga je potrebno prilikom odabira razmotriti sve navedeno i donijeti odluku prilikom implementacije modela baze podataka knjižničnog sustava u bilo koju knjižnicu.

Literatura:

1. Ali, Wajid...[et al.].Comparison between SQL and NoSQL Databases and Their Relationship with Big Data Analytics.//Asian Jurnal of Research inComputer Science 4, 2, str. 1-8. (2019). URL: https://www.researchgate.net/publication/336686999_Comparison_between_SQL_and_NoSQL_Databases_and_Their_Relationship_with_Big_Data_Analytics (2020-09-23)

2. Berg, Kristi L.; Seymour, Tom; Goel,Richa. History Of Databases.//International Jurnal of Management & Information Systems 17, 1 (2013), str. 29. URL: https://www.researchgate.net/publication/298332910_History_Of_Databases (2020-09-13)

3. Carić, Tonči; Buntić, Mario. Uvod u relacijske baze podataka. Zagreb: 2015., str. 2-16. URL: http://files.fpz.hr/Djelatnici/tcaric/Tonci-Caric-Baze-podataka.pdf (2020-09-13)

4. Chand, Mahesh. What are the Most Popular Relational Databases.//C#Corner. URL: https://www.c-sharpcorner.com/article/what-are-the-most-popular-relational-databases/ (2020-09-26)

5. Data Types: Structured vs. Unstructured data. 2019. URL: https://www.bigdataframework.org/data-types-structured-vs-unstructured-data/ (2020-09-15)

6. Database Management System (DBMS).// Technopedia. URL: https://www.techopedia.com/definition/24361/database-management-systems-dbms (2020-09-20)

7. Data Definition Language (DDL).// Technopedia. URL: https://www.techopedia.com/definition/1175/data-definition-language-ddl (2020-09-27)

8. Data Manipulation Language (DML).// Technopedia. URL: https://www.techopedia.com/definition/1179/data-manipulation-language-dml (2020-09-27)

9. DBMS Schemas: Internal, Conceptual, External.//Guru99. URL: https://www.guru99.com/dbms-schemas.html (2020-09-20)

10. DBMS Database Models.//Studytonight. URL: https://www.studytonight.com/dbms/database-model.php# (2020-09-20)

11. DBMS Keys:Candidate, Super, Primary, Foreign(Example).//Guru99. URL: https://www.guru99.com/dbms-keys.html (2020-09-26)

12. DBMS Transaction Management: What are ACID Properties?.//Guru99. URL: https://www.guru99.com/dbms-transaction-management.html (2020-09-26)

13. Drkusic, Emil. Learn SQL: SQL Data types, 2020. URL: https://www.sqlshack.com/learn-sql-sql-data-types/ (2020-09-27)

14. Gaspar, Drazena; Coric, Ivica. Bridging Relational and NoSQL Databases.SAD:IGI Global, 2017., str.1-12. URL: https://books.google.hr/books?hl=hr&lr=&id=DDc9DwAAQBAJ&oi=fnd&pg=PR1&dq=Bridging+Relational+and+NoSQL+Databases&ots=QnMsnmgVps&sig=un5eXEQ_E9c3Nu1WOuWLzW0GujQ&redir_esc=y#v=onepage&q=Bridging%20Relational%20and%20NoSQL%20Databases&f=false (2020-09-23)

15. Hussain, Sadequl. SQL and NoSQL Databases Features and Differences. 28. ožujka 2019. URL: https://www.mssqltips.com/sqlservertip/5980/sql-and-nosql-database-features-and-differences/ (2020-10-01)

16. Manger, Robert. Osnove projektiranja baza podataka. Zagreb: Sveučilište u Zagrebu, Sveučilišni računski centar (SRCE), 2010., str. 3. URL: https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/d310_polaznik.pdf (2020-09-13)

17. Manger, Robert. Baze podataka. Zagreb:Element, 2014., str. 5-29. URL: http://jadran.izor.hr/~dadic/EKO/baze-podataka.pdf (2020-09-26)

18. Nazrul, Syed Sadat. CAP Theorem and Distributed Database Management Systems. 24. travnja 2018. URL: https://towardsdatascience.com/cap-theorem-and-distributed-database-management-systems-5c2be977950e (2020-10-01)

19. NoSQL Technologies. //DB BEST Technologies. URL: https://www.dbbest.com/technologies/nosql-databases/ (2020-10-01)

20. Query Language.// Technopedia. URL: https://www.techopedia.com/definition/3948/query-language (2020-09-27)

21. Singh, Chaitanya. Entity Relationship Diagram-ER Diagram in DBMS.//BeginersBook. URL: https://beginnersbook.com/2015/04/e-r-model-in-dbms/ (2020-09-23)

22. Singh, Chaitanya. DBMS - Three Level Architecture.//BeginersBook. URL: https://beginnersbook.com/2018/11/dbms-three-level-architecture/ (2020-11-15)

23. Smallcombe, Mark. SQL vs NoSQL: 5 Critical Differences. 19. svibnja 2020. URL: https://www.xplenty.com/blog/the-sql-vs-nosql-difference/ (2020-10-14)

24. SQL | DDL, DQL, DML, DCL and TCL Commands, 26. kolovoza 2019. URL: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/ (2020-12-07)

25. Strauch, Christof. NoSQL Databases. Stuttgart: Stuttgart Media Uneversity, 2017., str. 2. URL: https://www.christof-strauch.de/nosqldbs.pdf (2020-10-01)

26. Unstructured data. //MongoDB. URL: https://www.mongodb.com/unstructured-data (2020-09-15)

27. Understanding the Different Types of NoSQL Databases. //MongoDB. URL: https://www.mongodb.com/scale/types-of-nosql-databases (2020-10-01)

28. Understanding the Hierarhical Database Model.//MariaDB. URL: https://mariadb.com/kb/en/understanding-the-hierarchical-database-model/ (2020-09-20)

29. Understanding the Network Database Model.//MariaDB. URL: https://mariadb.com/kb/en/understanding-the-network-database-model/ (2020-09-20)

30. What is NoSQL?.//MongoDB. URL: https://www.mongodb.com/nosql-explained (2020-10-01)

31. What is a Database Model.//Lucidchart. URL: https://www.lucidchart.com/pages/database-diagram/database-models (2020-11-18)

32.Why NoSQL? Advantages over relational databases, 30. listopad 2018. URL: https://oracle-patches.com/en/databases/3689-why-nosql (2020-10-01)

POPIS SLIKA

Slika 47. Naredbe za brisanje podataka iz tablica u SQL-u

POPIS TABLICA

Prilog 1.- SQL skripta:

```
create database model_knjiznice default character set utf8;

use model_knjiznice;


create table knjiga (
	id_knjiga			int not null auto_increment,
	naziv_knjige			varchar(255) not null,
	id_autor			int not null,
	id_nakladnik			int not null,
	jezik				varchar(30) not null,
	id_korisnik			int not null,
	standardni_broj			int not null,
	id_odjel			int not null,
	znanstveno_podrucje		varchar(100) not null,
	id_posudba			int not null,
	id_zakasnina			int not null,
primary key(id_knjiga),
CONSTRAINT `Constr_odjelKnjiznice1_fk`
FOREIGN KEY `odjelKnjiznice1_fk` (`id_odjel`) REFERENCES `odjelKnjiznice`
(`id_odjel`)
ON DELETE CASCADE ON UPDATE CASCADE
)engine=InnoDB;


create table knjiga_posudba(
	id				int not null,
	id_knjiga			int not null,
	id_posudba			int not null,
primary key(id_knjiga, id_posudba),
CONSTRAINT `Constr_knjiga1_fk`
FOREIGN KEY `knjiga1_fk` (`id_knjiga`) REFERENCES `knjiga` (`id_knjiga`)
```

```
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT `Constr_posudba_fk`
FOREIGN KEY `posudba_fk` (`id_posudba`) REFERENCES `posudba` (`id_posudba`)
ON DELETE CASCADE ON UPDATE CASCADE
)engine=InnoDB;

create table posudba(
      id_posudba                int not null auto_increment,
      datum                     datetime,
      id_knjiga                 int not null,
      id_korisnik               int not null,
primary key(id_posudba),
CONSTRAINT `Constr_korisnik1_fk`
FOREIGN KEY `korisnik1_fk` (`id_korisnik`) REFERENCES `korisnik` (`id_korisnik`)
ON DELETE CASCADE ON UPDATE CASCADE
)engine=InnoDB;

create table autor(
      id_autor              int not null auto_increment,
      ime                   varchar(100) not null,
      prezime               varchar(100) not null,
      id_knjiga             int not null,
primary key(id_autor)
)engine=InnoDB;

create table autor_knjiga(
      id                    int not null,
     id_knjiga              int not null references knjiga(id_knjiga),
     id_autor               int not null references autor(id_autor),
primary key(id_knjiga, id_autor)
```

```
)engine=InnoDB;

create table nakladnik(
	id_nakladnik		int not null auto_increment,
	naziv			varchar(255) not null,
	adresa			varchar(255) not null,
	id_knjiga		int not null,
primary key(id_nakladnik),
CONSTRAINT `Constr_knjiga_fk`
FOREIGN KEY `knjiga_fk` (`id_knjiga`) REFERENCES `knjiga` (`id_knjiga`)
ON DELETE CASCADE ON UPDATE CASCADE
)engine=InnoDB;

create table korisnik(
	id_korisnik		int not null auto_increment,
	ime			varchar(100) not null,
	prezime			varchar(100) not null,
	id_posudba		int not null,
	id_knjiga		int not null,
	id_zakasnina		int,
primary key(id_korisnik)
)engine=InnoDB;

create table odjelKnjiznice(
	id_odjel		int not null auto_increment,
	naziv			varchar(255) not null,
	brOdjela		int not null,
	id_knjiga		int not null,
	id_zaposlenik		int not null,
primary key (id_odjel)
```

```
)engine=InnoDB;


create table zakasnina(
     id_zakasnina            int not null auto_increment,
      dani_kasnjenja         int not null,
      id_knjiga              int not null,
      id_korisnik            int not null,
primary key(id_zakasnina),
CONSTRAINT `Constr_knjiga2_fk`
FOREIGN KEY `knjiga2_fk` (`id_knjiga`) REFERENCES `knjiga` (`id_knjiga`)
ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT `Constr_korisnik2_fk`
FOREIGN KEY `korisnik2_fk` (`id_korisnik`) REFERENCES `korisnik` (`id_korisnik`)
ON DELETE CASCADE ON UPDATE CASCADE
)engine=InnoDB;


create table zaposlenik(
       id_zaposlenik          int not null auto_increment,
       ime                    varchar(100) not null,
       prezime                varchar(100) not null,
       id_odjel               int not null,
primary key(id_zaposlenik, id_odjel),
CONSTRAINT `Constr_odjelKnjiznice_fk`
FOREIGN KEY `odjelKnjiznice_fk` (`id_odjel`) REFERENCES `odjelKnjiznice`
(`id_odjel`)
)engine=InnoDB;


insert into autor (ime, prezime, id_knjiga)
values ("Meša", "Selimović", 1), ("Ivana", "Brlić-Mažuranić", 2),
         ("Ivo", "Andrić", 3);
```

```
insert into knjiga(naziv_knjige, id_autor, id_nakladnik, jezik, id_korisnik,
standardni_broj, id_odjel, znanstveno_podrucje, id_posudba, id_zakasnina)

values("Prokleta avlija", 1, 1, "hrvatski", 1, 978-953-0-60764-4, 1, "nepoznato",
2, 1);


insert into korisnik (ime, prezime, id_posudba, id_knjiga, id_zakasnina)

values ("Mato", "Matić", 1, 1, 1), ("Ivo", "Ivić", 2, 1, 5);


insert into odjelKnjiznice(naziv, brOdjela, id_knjiga, id_zaposlenik)

values("Odjel za odrasle", 4, 1, 1);


insert into zaposlenik(ime, prezime, id_odjel)

values("Ana", "Anić", 3), ("Ivana", "Ivić", 2);


alter table korisnik

add godina_rodjenja DATETIME;


update odjelKnjiznice

set brOdjela = 3

where id_odjel = 1;


select * from knjiga;


select naziv_knjige, id_autor, jezik, id_posudba

from knjiga;


truncate zaposlenik;


delete from korisnik

where id_korisnik=2;
```

Prilog 2. - NoSQL skripta

```
use model_knjiznice

db.createCollection("knjiga")

db.createCollection("posudba")

db.createCollection("korisnik")

db.createCollection("odjelKnjiznice")

db.createCollection("zaposlenik")

db.createCollection("nakladnik")

db.createCollection("zakasnina")

db.createCollection("autor")

db.autor.insertOne({ime:"Ivo", prezime: "Andrić", knjiga: {naziv_knjige: "Prokleta avlija"}})

db.autor.insertMany([{ime:"Meša", prezime: "Selimović", knjiga: {naziv_knjige: "Tvrđava"}}, {ime:"Ivana", prezime: "Brlić-Mažuranić", knjiga: {naziv_knjige:"Regoč"}}])

db.knjiga.insertOne({_id: 1, naziv_knjige:"Prokleta avlija", autor:{ime: "Ivo", prezime:"Andrić"}, nakladnik: "Školska knjiga", jezik: "hrvatski", korisnik: {ime: "Ivo", prezime: "Ivić"}, standardni_broj: 978-953-0-60764-4, odjelKnjiznice: {naziv:"Odjel za mlade", brOdjela: 4}, posudba: {datum: new Date(2020-11-19)}, zakasnina: {dani_kasnjenja: 5}})

db.korisnik.insertMany([{ime: "Mato", prezime: "Matić"}, {ime: "Ivo", prezime: "Ivić"}, {ime: "Marija", prezime: "Marić"}])

db.odjelKnjiznice.insertOne({naziv: "Odjel za odrasle", brOdjela: 4, zaposlenik: {ime: "Ivana", prezime: "Ivanović"}})

db.zaposlenik.insertOne({ime: "Ivana", prezime: "Ivanović", odjelKnjiznice: {naziv: "Odjel za odrasle", brOdjela: 4}})

db.korisnik.updateMany({ }, {$set: {Godina_rodjenja: new Date()}})

db.odjelKnjiznice.updateMany({brOdjela: {$gt: 2}}, {$set: {naziv: "Odjel za mlade"}})
```

```
db.knjiga.find()

db.knjiga.find({}, {naziv_knjige: "Prokleta avlija", autor: {ime: "Ivo"}, jezik:
"hrvatski"})

db.zaposlenik.deleteMany({})

db.korisnik.deleteMany({_id: 1})
```