Izrada baze podataka na primjeru skladišta korištenjem MYSQL 8

Rojnić, Stefano

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike

STEFANO ROJNIĆ

Izrada baze podataka na primjeru skladišta korištenjem mysql 8

Završni rad

Pula, 1.9., 2020. godine

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike

STEFANO ROJNIĆ

Izrada baze podataka na primjeru skladišta korištenjem mysql 8

Završni rad

JMBAG: 0303075824, redoviti student

Studijski smjer: Informatika

Kolegij: Baze podataka 2

Mentor: doc. dr. sc. Goran Oreški

Pula, 1.9., 2020. Godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

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

Student

_____

U Puli, rujan, 2020. godine

IZJAVA

o korištenju autorskog djela

Ja, Stefano Rojnić dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom “Izrada baze podataka na primjeru skladišta korištenjem MySQL 8” koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

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

U Puli, rujan, 2020. godine

Potpis

_____

1. UVOD

Baze podataka su zapisi raznih vrsta podataka koji su pohranjeni na nekom računalu. U računalnom svijetu služe za pohranu podataka te omogučavaju nekom račnalnom programu brzo spremanje i dohvaćanje tih podataka. Današnje baze podataka omogučavaju spremanje podataka poput slika, video i audio zapisa te čak emojija.

Cilj ovog rada je upoznati se s MySQL kroz povijest, prednosti i nedostatke, usporedbom s drugim sustavima, te novim funkcionalnostima koje donosi MySQL 8. Izraditi ću primjer baze podataka za jedno skladište koje zaprima te otprema narudžbe. Moguće je pračenje kupaca i njihovih narudžbi, te praćenje artikla koji se nalaze u skaldištu. U radu je prikazano konceptualno oblikovanje, kao i proces i pravila pretvaranja konceptualne scheme u relacijsku schemu. Sustav se može lako prilagoditi ili nadograditi, a svi korišteni alati su besplatni i dostupni za korištenje. Korišteni alat za upravljajne bazom podataka je MySQL. To je sustav otvorenog koda koji postoji već 25 godina, pokreće se na poslužitelju te ima svoj vlastiti alat za izradu relacijskih baza podataka naziva MySQL Workbench. Sustav MySQL koriste mnoge svjetske kompanije poput Wikipedia-e, Adobe-a, Paypal-a, Craiglist-a, Wordpress-a i drugih. Često se u praksi koristi u kombinaciji s PHP jezikom.

Motiv za izradu ovog rada bio mi je prijašnje poznavanje MySQL jezika, te rad s bazom podataka. S MySQL sam se upoznao prije nekoliko godina prilikom korištenja Wordpress i SMF web aplikacija, te jedne računalne igre. Wordpress i SMF su web aplikacije koje koriste MySQL bazu podataka za čuvanje podataka. Modifikacija originalne računalne igre je otvorenog koda, izrađena u C, C++ i dr. jezicima. Za spremanje podataka koristi lokalne datoteke s .ini ekstenzijama. Poboljšanjem izvornog koda može se napraviti spremanje podataka u bazu podataka putem MySQL. To omogučava brže, sigurnije i bolje čuvanje podataka uz mnoge druge mogučnosti poput izrade web aplikacije koja se spoji s bazom gdje se čuvaju podaci, te se tako putem web stranice mogu prikazati podaci iz igre.

2. MySQL

MySQL je najpopularniji SQL sustav za upravljanje bazom podataka s otvorenim kodom. Dolazi kao sastavni dio serverskih Linux distribucija, no postoje verzije za ostale operacijske sustave poput Windows i Mac. Izvodi se u pozadini osim ako nije pokrenut putem konzole. Podržan je jezik strukturiranih upita. Stvoren je 1995.-e godine od švedske tvrtke, no popularnost dobiva tek 2000-e godine. Iako je MySQL u početku svog nastojanja dominirao, kasnije nailazi na probleme s financijama. Zbog otvornog koda većina prihoda je iz donacija. Kasnije Oracle kompanije preuzima vodstvo nad MySQL. Baza podatka može sadržavati jednostavan popis podataka, sve do većih količina podataka za velike korporacije. MySQL su relacijske baze podataka, podaci se ne spremaju u jedno veliko spremište već u zasebne tablice. Od alata za izradu baze poznat je MySQL Workbench koji se može besplatno preuzeti. Pomoću alata dizajniramo izgled baze podataka, tj. shemu baze. Shema se kasnije pretvori u tablice koje koristimo za pohranu podataka. Element koji se pohranjuje u bazu podataka naziva se entitet. Entitet može biti osoba, objekt, neki događaj i sl. i opisan je atributima. Postoje različiti odnosi između entiteta koji se nazivaju relacija. Relacije se prema vrsti mogu podijeliti na jedan naprema jedan, jedan naprema više, više naprema jedan, te više naprema više. MySQL se vrlo često koristi u kombinacija sa PHP jezikom. U takvoj kombinaciji češće se koristi phpmyadmin za upravljanje podacima u bazi.

SQL dio u "MySQL" znači "Strukturirani jezik upita". SQL je najčešći standardizirani jezik koji se koristi za pristup bazama podataka. SQL je definiran ANSI / ISO SQL standardom. SQL standard razvija se od 1986. godine, a postoji nekoliko verzija. "SQL: 1999" odnosi se na standard objavljen 1999. godine, a "SQL: 2003" na trenutnu verziju norme. Koristimo frazu "SQL standard" da označimo trenutnu verziju SQL standarda u bilo kojem trenutku. (Anon, n.d.).

MySQL je softver otvorenog koda. Može se besplatno preuzeti na službenim stranicama te koristiti i mijenjati. Izvorni kod može se mijenjati uz poštivanja pravila koja su određena GPL (GNU General Public License) licencom. MySQL je najčešće korišten za spremanje podataka kod web aplikacija.

2.1 Povijest

Povijest MySQL-a seže u 1979. godinu kada je Monty Widenius, radeći za malu tvrtku TcX, stvorio alat za izvještavanje napisan na BASIC-u koji je radio na računalu od 4 MHz s 16 KB RAM-a. S vremenom je alat prepisan u C i prebačen da radi na Unixu. To je još uvijek bio samo mehanizam za pohranu na niskoj razini s prednjim dijelom za izvještavanje. Alat je bio poznat pod imenom Unireg. Neko vrijeme 1990-ih, kupci TcX-a počeli su se zalagati za SQL sučelje za svoje podatke. (S. Pachev, n.d.)

U svibnju 1996. godine nastaje prva MySQL verzija. Prvotno je bila objavljena samo za ograničenu grupu ljudi. U listopadu iste godine objavljeno je javno izdanje pod verzijom 3.11.1.

MySQL projekt su pokrenuli Michael Widenius (Monty), David Axmark i Allan Larsson. Prvih 6 godina komercijalno ga je zastupao TCX, tvrtka u vlasništvu jednog od 3 osnivača. Monty je bio u Finskoj, David i Allan u Švedskoj. (Dries Buytaert, 2010.)

2000-e godine MySQL softver postaje otvorenog koda po uvjetima GPL licence. Od tada prihodi postaju znatno smanjeni (čak 80% manje), ali popularnost masovno raste pa tako već nakon godinu dana MySQL ima 2 milijuna aktivnih instalacija. 2002.-e godine tvrtka otvara sjedište u SAD-u. Širenjem na američko tržište, već iste godine broj korisnika prelazi 3 milijuna, uz prihod od 6.5 milijuna dolara. Popularnost je konstantno rasla. Do kraja 2003.-e godine MySQL ima 4 milijuna aktivnih instalacija, a prihod tvrtke bio je skoro duplo od prethodne godine. Kasnije tvrtka mijenja poslovni model u periodičnu pretplatu umjesto jednokratne naknade za licencu. Model se pokazao uspješnim te donio prihod od 20 milijuna dolara 2004.-e godine.

2006-e godine Mårten Mickos potvrđuje da je Oracle pokušao kupiti MySQL. Izvršni direktor Larry Ellison komentirao je kako su razgovarali, no MySQL je tada bila malena kompanija s nedovoljno prihoda za Oracle. Oracle kupuje Sleepycat, tvrtku koja pruža MySQL Berkeley DB mehanizam za transakcijsko pohranjivanje. Procjenjuje se da MySQL ima 33% tržišnog udjela izmjerenog u instalacijskoj bazi i 0,2% tržišnog udjela mjerenog u prihodima (tržište baze podataka bilo je 15 milijardi dolara u 2006. godini). Sun Microsystems 2008. godine kupio je MySQL AB za otprilike jednu milijardu dolara. (Dries Buytaert, 2010.)

Nedugo nakon što je Sun Microsystems preuzeo, dva od suosnivača MySQL AB-a Michael Widenius i David Axmark odlaze. Dvije godine kasnije Oracle je kupio većinski dio dionica Sun Microsystems-a i postao jedna od vodećih svjetskih informatičkih kompanija. Danas MySQL ima preko 100 milijuna aktivnih instalacija.

2.2 Prednosti

MySQL je jedna od najpopularnijih baza podataka za web-bazirane aplikacije. Besplatna za korištenje te se često ažurira značajkama i sigurnosnim poboljšanjima. Instalacija je vrlo jednostavna, te ne koristi mnogo resursa. Tu je i sučelje koje se jednostavno koristi, a naredbe iz paketa omogućuju obradu ogromnih količina podataka. Sustav je također nevjerojatno pouzdan.

Od akvizicije, Oracle je povećao MySQL osoblje i dao mu zreliji inženjerski proces u kojem se inženjering i planiranje vode od strane Oracle-a umjesto ljudi rasutih po cijelom svijetu. MYSQL koristi InnDB kao svoj glavni mehanizam za pohranu podataka. InnoDB je također dio Oracle obitelji koja razvojne timove čini još integriranijima. Tvrtka također kod čini modularnijim. Na primjer, u MySQL 5.6 podijelili su jednu od ključnih brava na MySQL poslužitelju, LOCK_open, koja bi mogla poboljšati vrhunske performanse za više od 100%. (Leah Beatty, 2013.)

MySQL je najsigurniji i pouzdan sustav upravljanja bazama podataka koji koriste mnoga poznata poduzeća poput Facebook, Twitter i Wikipedia. Pruža veliku sigurnost podataka, posebno korisničkih lozinki koje su šifrirane posebnim algoritmom. Osim sigurnosti, MySQL sustav je skalabilan, može se nadograđivati i nadopunjavati. MySQL vrši obradu transakcije jako dobro i brzo. Novija verzija MySQL-a vrši obradu istog broja transakcija u puno kraćem vremenu. MySQL softver vrlo je lagan i ne zauzima mnogo prostora. Kada je pokrenut, on radi u pozadini osim ako nije pokrenut putem konzole. Jednostavan je za instalaciju i alati za rad su dostupni besplatno. Jedna od prednosti je MySQL zajednica. Kako se MySQL koristi u mnoštvo web aplikacija, gdje skoro svaka php aplikacija koristi MySQL bazu podataka, dostupno je mnogo dokumentacija i raznih foruma za pomoć. Kako postoji već dugi niz godina većina problem na koji se možemo susresti najvjerovatnije rješenje možemo pronaći na internetu.

2.3 Nedostaci

MySQL je baza otvorenog koda, ali nakon što je Oracle preuzeo vodstvo u praksi je to drugačije. Oracle ne prihvaća nikakve zakrpe niti objavljuje javni plan puta. Fokusira se isključivo na razvoj, odbija korisnička testiranja greški i sigurnosne zakrpe. Mnogi programeri otvorenog koda okrenuli su se MariaDB gdje je sav kod objavljen pod GPL, LPGL ili BSD licencom. Oracle nije drastično promjenio smjer u kojem MySQL ide, ali mnogi programmeri nisu zadovoljni njihovim uvjetima i ne prihvaćanjem ispravka programskih pogreški. Mnogi instalacijski paketi i operacijski sustavi prešli su na MariaDB ili ga nude uz MySQL. Veliku konkurenciju predstavlja SQLite i MariaDB. Na primjer, wamp programski paket za windows dolazi s oba sustava za upravljanje bazom podataka. Red Hat Enterprise Linux, Fedora, Slackware Linux, openSUSE i Wikimedia Foundation prešli su na MariaDB.

MySQL nije zreo kao ostali sustavi za upravljanje relacijskim bazama podataka. MySQL nije započeo kao RDBMS (sustav upravljanja relacijskim bazama podataka), ali je kasnije promijenio smjer kako bi obuhvatio više funkcionalnosti. Neki zreliji RDBMS, poput PostgreSQL-a, smatraju se još značajnijim. Opcije bliskog izvora, kao što su Oracle ili Microsoft SQL Server, su također alternative koje treba razmotriti. Mnoga velika imena prelaze na druge sustave za upravljanje bazom podataka. (Leah Beatty, 2013.)

Može se instalirati samo kao poslužitelj baze podataka, što znači da se ne može ugraditi u prijenosni program, mada se to može učiniti s malo trika, to nije pravi izbor baze podataka za tu vrstu ugrađenog sustava. MySQL ima veći broj ljudi koji konstantno rade na značajkama i promjenama interne arhitekture nego što ima MariaDB. Iz tog razloga MySQL opstaje još dugo vremena, no dosta velikih korporacija je već prešlo na MariaDB. Google, Craiglist, wikipedia, RedHat, CentOs i Fedora već su prešli na MariaDB. Tranzicija je jednostavna jer im je struktura baze podataka ista i ne treba je mijenjati ukoliko prelazimo s jednog sustava na drugi.

Izvedba sheme i složenost upita utječu na performanse. Loše izvedena shema ili upitom koji se vrši nad bazom su primarni razlozi za pogoršanje performansi.

Na temelju izvedbe sheme i složenosti upita, performance se mogu pogoršati ili poboljšati, jer postoje MySQL baze podataka koje su optimizirane i učinkovite u posjedovanju više od milijardu redaka zbog pravilne provedbe. Na primjer, upit za neograničene/neindeksirane stupce definitivno bi rezultirao većim vremenom odgovora jer mora izvesti skeniranje cijele tablice, što je process koji zahtijeva velike resurse i dovodi do degradacije performansi. (Vanier E., Shah B., Malepati T., 2019.)

2.4 Usporedba s ostalim sustavima

Za usporedbu ću uzeti SQLite i PostreSQL.

SQLite je besplatni softver otvorenog koda. Opisuje se kao baza podataka bez poslužitelja. Baza podataka je ustvari samo jedna datoteka koja se nalazi na disku.

SQLite ima znatno manji broj podržanih tipova podataka, npr za brojeve podržava samo INT i REAL, dok MySQL raspolaže tipovima INT, BIGINT, FLOAT, DOUBLE, DOUBLE DECIMAL. Biblioteka je veličine oko 250kb, dok je MySQL poslužitelj oko 600mb. SQLite nema korisničku funkciju upravljanja, dok MySQL ima dobar sustav upravljanja s više korisnika s različitim dozvolama. Lako se ograničuje korisnikovo upravljanje bazom, moguće je ograničiti naredbe poput create, drop, delete, insert, select i update. SQLite nema ugrađeni mehanizam za provjeru autentičnosti, podacima iz baze može pristupiti bilo tko. MySQL ima dobru sigurnost, pa tako sadrži provjeru autentičnosti s korisničkim imenom, lozinkom i SSH.

Kada želimo bazu podataka koja je lako prenosiva, ušteda prostora i jednostavnost podataka i upravljanje odabrati ćemo SQLite. MySQL ćemo odabrati kada nam je potrebno puno značajki u vezi s bazom, dobre sigurnosne značajke, skalabilnost te upravljivost korisnicima i višestruke kontrole pristupa. (Edward S., 2019.)

PostgreSQL, poznat i kao Postgre ili pgsql, također je besplatan sustav za upravljanje bazama podtaka otvorenog koda. Oglašava se kao najnaprednija relacijska baza podataka otvorenog koda na svijetu. Za razliku od MySQL koja je relacijska baza podataka, PostgreSQL je objektno-relacijska baza podataka. PostgreSQL je izgrađen da bude bogat značajkama, proširljiv i drži se većine SQL:2011 standarda. U prošlosti ova dva sustava za upravljanje bazom podataka bila su uravnoteženija. Postgresove performanse čitanja bila su sporija od MySQL-a, ali je bio sposoban pisati velike količine podataka učinkovitije. U novim verzijama razlike u performansama su nepostojeće. Postgres je manje popularan od MySQL-a. Koriste ga mnogi popularni servisi poput Skype, Reddit, Instagram i drugi.

Postgres uključuje značajke poput nasljeđivanja tablice i preopterećenja funkcija, što može biti važno za određene aplikacije. Postgres je poznat po zaštiti integriteta podataka na razini transakcije. To ga čini manje ranjivim na korupciju podataka. Postgres je i dalje manje popularan od MySQL-a, pa je na raspolaganju manji broj alata trećih strana ili programera ili administratora baza podataka. Postgres se izrađuje s proširivošću, usklađenošću s standardima, skalabilnošću i integritetom podataka - ponekad na štetu brzine. Zbog toga, za jednostavne, teške tijekove rada, Postgres može biti lošiji izbor od MySQL-a. (Krasimir Hristozov, 2019.)

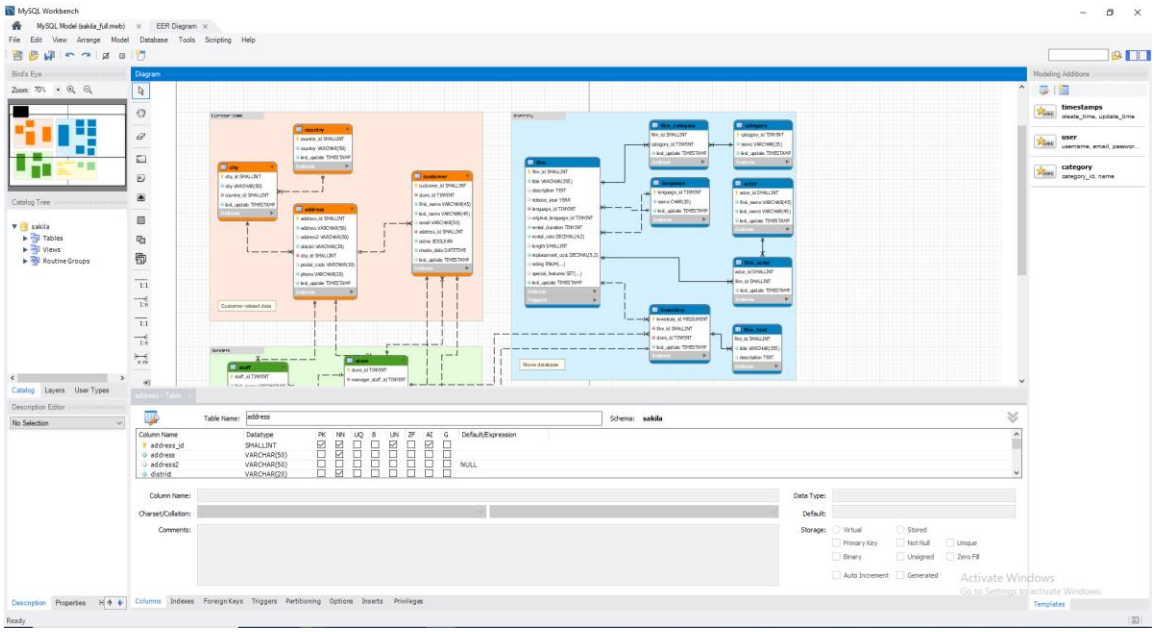
2.5 Alati za rad

MySQL Workbench je objedinjeni vizualni alat za arhitekte baza podataka, programere i DBA. MySQL Workbench pruža sučelje koje se lako koristi za obavljanje mnogih zadataka uključenih u rad s bazama podataka. Integrira razvoj SQL-a, administraciju, dizajn baze podataka, stvaranje i održavanje u jedno vizualno integrirano razvojno okruženje. MySQL Workbench dostupan je na Windowsima, Linuxu i Mac OS X. Sličan je SSMS SQL Serveru, koji se koristi za administraciju SQL Servera.

MySQL Workbench sadrži vizualne alate koji vam omogućuju zadatke poput:

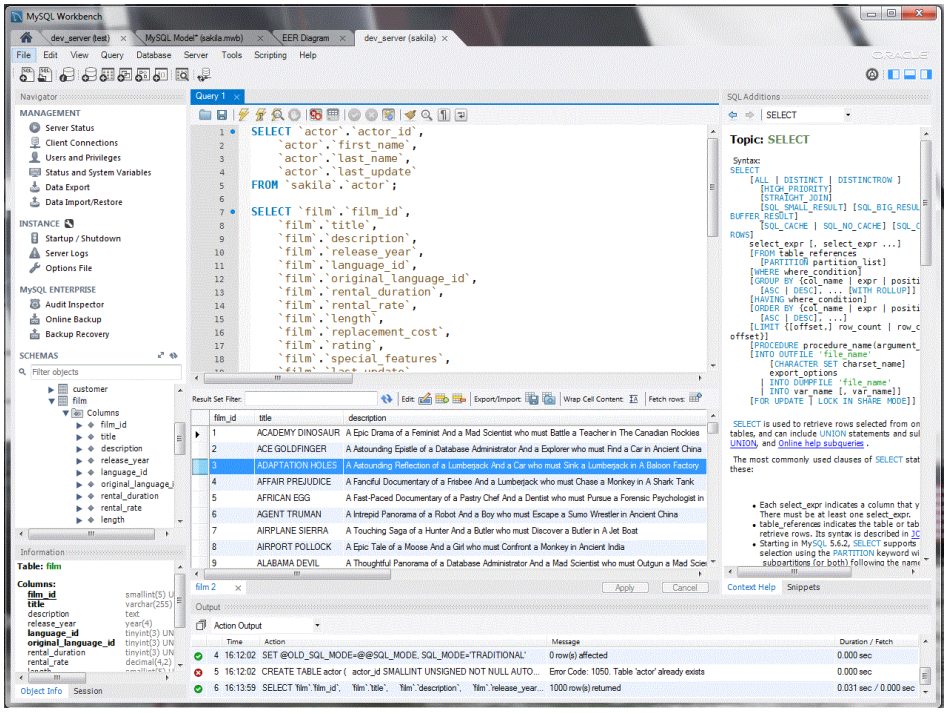
- Stvaranje i pregled baza podataka
- Stvaranje i pregled ostalih objekata baze podataka (kao što su prikazi, okidači, pohranjeni postupci itd.)
- Stvaranje, izvršavanje i optimizacija SQL upita
- Konfiguriranje poslužitelja
- Administriranje korisnika
- Izvršavanje sigurnosne kopije i oporavak
- Pregled podataka revizije
- Pregled statusa poslužitelja

MySQL Workbench omogućuje DBA, programera ili arhitekta podataka da vizualno dizajniraju, modeliraju, generiraju i upravljaju bazama podataka. Sadrži sve što modelar podataka treba za izradu složenih ER modela, inženjering unaprijed i obrnuto, a također nudi ključne značajke za izvršavanje teških zadataka upravljanja promjenama i dokumentacije koji obično zahtijevaju mnogo vremena i truda. (Anon, n.d.)



Slika 1: MySQL Workbench alat

MySQL Workbench nudi vizualne alate za stvaranje, izvršavanje i optimizaciju SQL upita. SQL Editor nudi isticanje sintakse u boji, automatsko dovršavanje, ponovnu upotrebu SQL isječaka i povijest izvršavanja SQL-a. Panel s bazama podataka omogućuje programerima lako upravljanje standardnim vezama s bazama podataka, uključujući MySQL Fabric. Preglednik objekata omogućuje trenutni pristup shemi baze podataka i objektima. (Anon, n.d.)



Slika 2: MySQL Workbench sql editor (Anon., n.d.)

Workbench alat također ima alat za dokumentiranje shema baze podataka. Služi za vizualni prikaz svih informacija o shemi, tablica i međusobnih relacija. Olakšava i smanjuje posao programeru ili administratoru baza podataka jer dokumentiranje dizajna može biti dugotrajan proces. Modeli se mogu generirati u HTML kodu ili kao običan tekst, te sadrži sve objekte i modele trenutne sesije u alatu.

3. MySQL 8

MySQL 8 objavljen je 19. Travnja 2018-e godine. MySQL 8.0 uključuje značajna poboljšanja performansi, sigurnosti i produktivnosti programera omogućujući sljedećoj generaciji web, mobilnih, ugrađenih i Cloud aplikacija. Nove promjene dogodile su se na područjima SQL, JSON i GIS. Također postavljen je novi zadani znakovni skup UTF8MB4 koji omogućava spremanje emojija. MySQL je danas najkorisnija i široko korištena platforma baze podataka otvorenog koda. Najveće, najpopularnije i najprometnije web stranice na svijetu, poput GitHub, US Navy, NASA, Tesla, Netflix, WeChat, Facebook, Zendesk, Twitter, Spotify i drugi, oslanjaju se na MySQL.

MySQL 8.0 temelji se na ovom zamahu kroz mnogo poboljšanja osmišljena kako bi omogućili inovativnim DBA-ovima i programerima da stvore i implementiraju novu generaciju web, ugrađenih, mobilnih i Cloud / SaaS / PaaS / DBaaS aplikacija na najnoviju generaciju razvojnih okvira i hardvera platforme. (Anon., n.d.)

Glavni pokretački razlozi za nadogradnju MySQL-a uključuju sljedeće.

- Značajke: Objavljena je nova značajka koja može poboljšati vaše aplikacije ili podaci, primjeri uključuju spremište dokumenata, grupnu replikaciju, i InnoDB klaster
- Izvedba: Novija verzija poboljšava stvaranje performansi vaše aplikacije bolje. Na primjer, najnovije izdanje 5.7 je mnogo puta brže od prethodnih verzija i MySQL 8 obećava da poboljšati na tome.
- Održavanje: Postoje nove značajke koje vam pomažu u održavanju sustav bolji. Primjeri uključuju novi rječnik podataka, Grupa Replikacijski i pomoćni alati poput MySQL Enterprise Backup.
- Bug fixes: Možda postoje nedostaci u starijim verzijama koji su potrebni zaobilazna rješenja ili ograničenja. Novije verzije mogu sadržavati ispravke za kritične programske pogreške kako biste mogli ukloniti zaobilazne načine i ograničenja uzrokovane nedostatkom.
- Usklađenost: Vaša platforma, standardni operativni postupci ili vanjski entiteti zahtijevaju nadogradnju radi usklađenosti.
  (Charles B., 2018.)

3.1 Nove funkcionalnosti

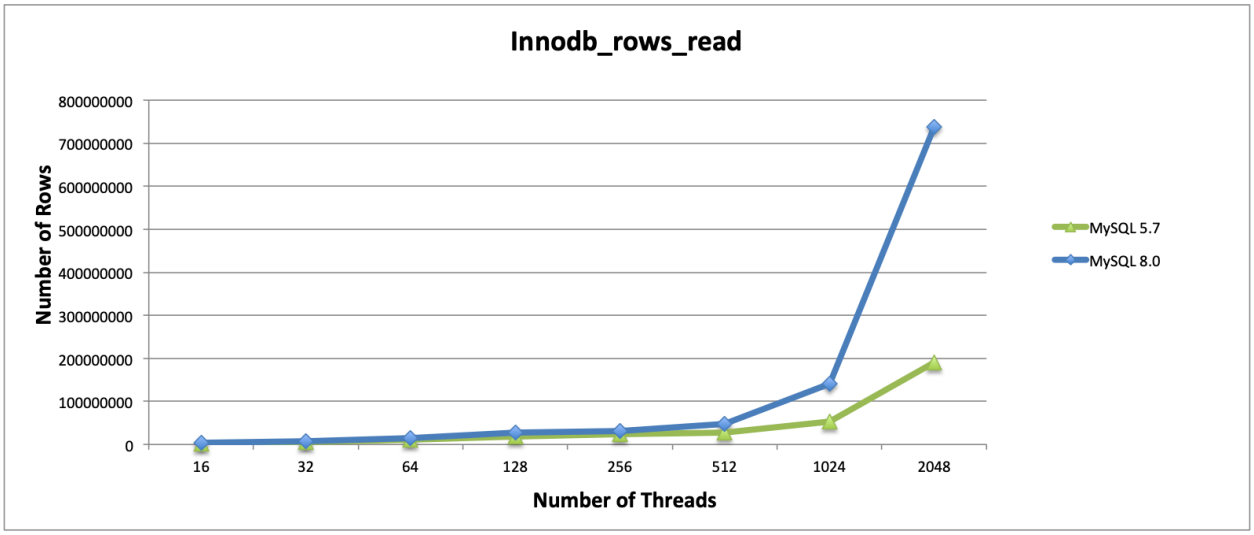
MySQL 8 donosi više od 250 novih značajki. Sve značajke mogu se pronaći u MySQL referentnom priručniku. Neke od ključnih novih poboljšanja su:

1. **SQL** - funkcije prozora, uobičajeni izrazi tablica, NOWAIT i SKIP LOCKED, silazni indeksi, grupiranje, regularni izrazi, skupovi znakova, model troškova i histogrami.
2. **JSON** - Proširena sintaksa, nove funkcije, poboljšano sortiranje i djelomična ažuriranja. Pomoću funkcija tablice JSON možete koristiti SQL stroj za JSON podatke.
3. **GIS Geography support** - Prostorni referentni sustavi (SRS), kao i SRS svjesni tipovi prostornih podataka, prostorni indeksi i prostorne funkcije.
4. **Pouzdanost** - DDL izjave postale su atomske i nesretne, meta-podaci pohranjuju se u jednom transakcijskom rječniku. Pokreće ih InnoDB.
5. **Osmotrivost -** Značajna poboljšanja sheme performansi, informativne sheme, varijable konfiguracije i evidentiranja pogrešaka.
6. **Upravljivost** - Daljinsko upravljanje, Poništavanje upravljanja prostorom tablice i novi instant DDL.
7. **Sigurnost** - Poboljšanja OpenSSL-a, nove zadane provjere autentičnosti, SQL uloge, razbijanje super privilegija, snage lozinke i još mnogo toga.
8. **Performanse** - InnoDB je značajno bolji kod čitanja / pisanja opterećenja, IO-vezanog radnog opterećenja i visokih kontroverzi "vruće točke". Dodana značajka Grupe resursa kako bi se korisnicima omogućila optimizacija za određeno radno opterećenje na određenom hardveru preslikavanjem korisničkih niti u CPU.

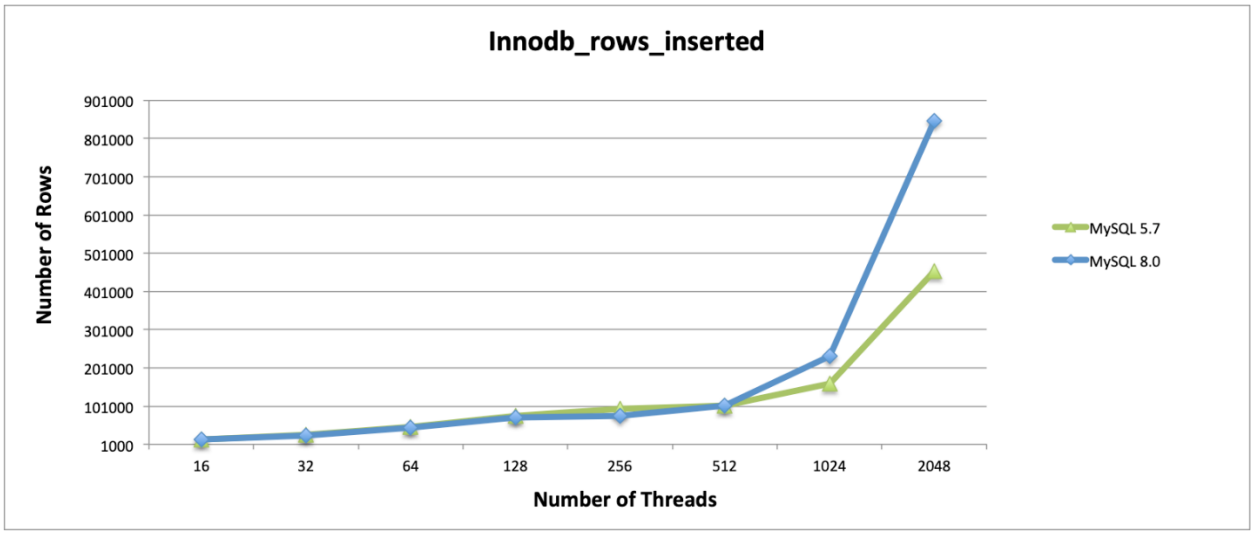
   (Geir Hoydalsvik, 2018.)

Pouzdanost MySQL-a u verziji 8 znatno je povećana. Svoje meta-podatke pohranjuje u InnoDB mehanizam. InnoDB pruža standardne značajke transakcija u skladu s ACID-om, zajedno s podrškom za strane ključeve. Sistemske tablice poput korisnika i privilegija, zajedno s tablicama rječnika podataka sada se nalaze u InnoDB. U prijašnjim verzijama postojala su dva rječnika podataka. U nekim slučajevima moguće je da oni izađu iz sinkronizacije i dođe do pada sustava. U novoj verziji postoji samo jedan rječnik podatka. MySQL 8 osigurava atomski ddl opravak s binarnim zapisnikom. Bilo koji ddl izraz će se izvršiti u potpunosti ili se neće izvršiti uopće.

Jedna od najznačajnijih funkcionalnosti MySQL 8 je performansa mehanizma za pohranu sustava InnoDB. Na slikama je prikazana Innodb read i insert operacija. Kod manjeg broja thread-ova, MySQL 5.7 i 8.0 donekle su slični. Povečanjem broja thread-ova, MySQL 8.0 nadmašuje prijašnju verziju. Obrađeno je puno više readaka za isti broj thread-ova. Isto vrijedi i za read i delete operacije.

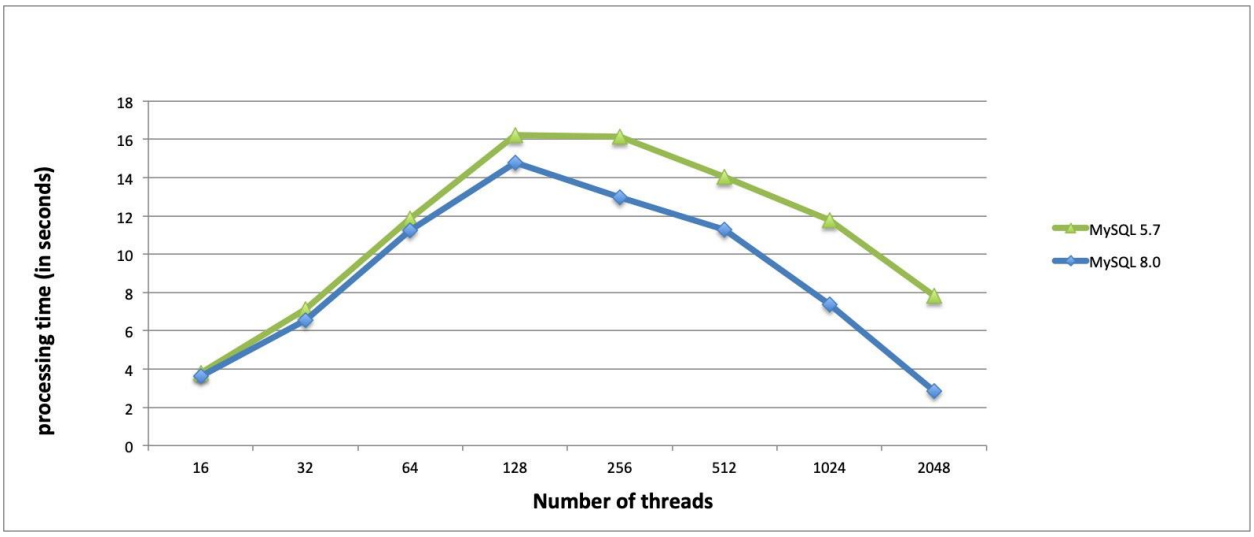


Slika 3: InnoDB read usporedba (Paul Namuag, 2019.)

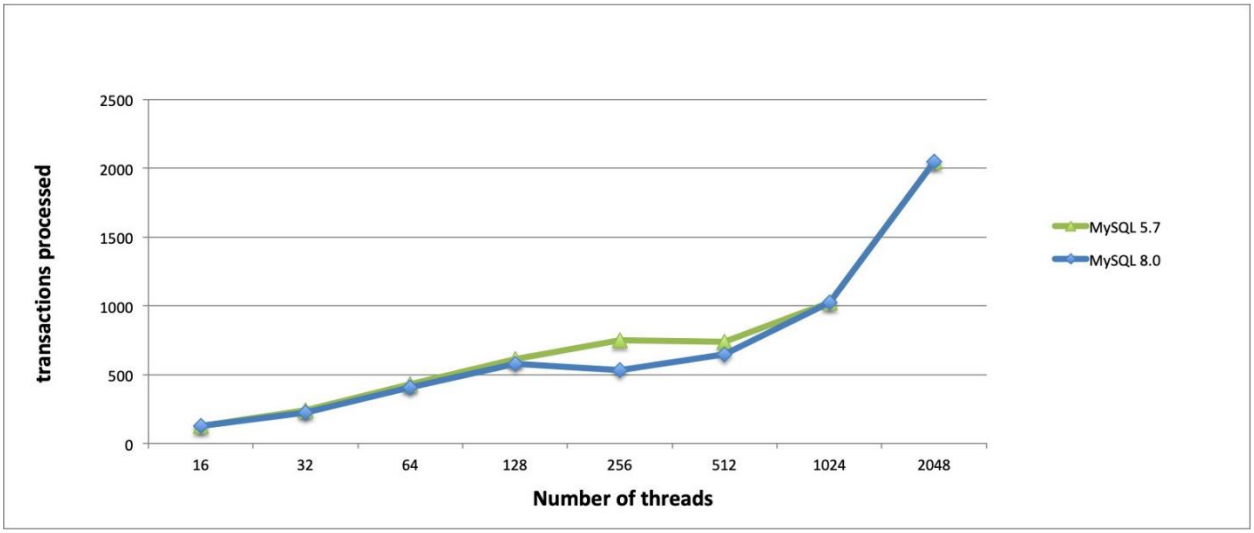


Slika 4: InnoDB insert usporedba (Paul Namuag, 2019.)

Na slikama je prikazana usporedba transakcija. MySQL 8 opet nadmašuje verziju 5.7. Na slici 6 prikazana je usporedba potrebnog vremena da se obrade transakcije. MySQL verziji 8 potrebno je znatno manje vremena za obradu transakcija.



Slika 5: Usporedba vremena obrade transakcije (Paul Namuag, 2019.)



Slika 6: Usporedba obrade transakcija (Paul Namuag, 2019.)

Slika 6 prikazuje broj transakcija koji se međusobno ne razliku. Obje verzije MySQL-a izvršiti će gotovo isti broj transakcija, ali se razlikuju kojom će brzinom završiti. MySQL 5.7 i dalje dobro podnosi mala opterećenja, ali pri vevćin opterećenja verzija 8 operacija izvršava brže.

3.2 Zastarjele funkcionalnosti

Sljedeće su značajke zastarjele u MySQL 8.0 i mogu biti ili će biti uklonjene u budućem nizu.

Set znakova utf8mb3 je zastario te se koristi utf8mb4. Budući da je caching_sha2_password zadani dodatak za autentifikaciju u MySQL 8.0 i pruža super skup mogućnosti dodatka za provjeru autentičnosti sha256_password, sha256_password je zastario i bit će uklonjen u budućoj MySQL verziji. Dodatak validate_password ponovo je dopunjen za upotrebu infrastrukture komponente poslužitelja. Oblik dodatka validate_password i dalje je dostupan, ali je zastario i bit će uklonjen u budućoj verziji. Podrška AUTO_INCREMENT zastarjela je za stupce tipa FLOAT i DOUBLE (i sve sinonime). Preporučljivo je uklanjanje atributa AUTO_INCREMENT iz takvih stupaca ili pretvorba u cijeli broj. Atribut UNSIGNED zastario je za stupce tipa FLOAT, DOUBLE i DECIMAL (i bilo koje sinonime). Sintaksa FLOAT (M, D) i DOUBLE (M, D) za određivanje broja znamenki za stupce tipa FLOAT i DOUBLE (i bilo koje sinonime) nestandardno je MySQL proširenje i zastarjela sintaksa.

Za izjave SELECT upotreba odredbe INTO nakon FROM, ali ne na kraju SELECT, zastarjela je od MySQL 8.0.20. Preferira se stavljanje INTO-a na kraj izjave.

Za UNION izjave, ove dvije inačice koje sadrže INTO zastarjele su u verziji 8.0.20: Korištenje INTO prije FROM u zadnjem bloku upita izraza upita. U zagradivom slijedećem bloku izraza upita Korištenje INTO, bez obzira na njegov položaj u odnosu na FROM. (Anon., n.d.)

3.3 Uklonjene funkcionalnosti

Sljedeće stavke su zastarjele i uklonjene su u MySQL 8.0.

Sistemska varijabla innodb_locks_unsafe_for_binlog uklonjena je. READ COMMITTED razina izolacije pruža sličnu funkcionalnost.

Kod koji se odnosi na zastarjele tablice sustava InnoDB uklonjen je u MySQL 8.0.3. INFORMATION_SCHEMA prikazi koji se temelje na sistemskim tablicama InnoDB zamijenjeni su internim prikazima sustava na tablicama rječnika podataka. Zahvaćeni InnoDB pregledi INFORMATION_SCHEMA preimenovani su:

Stari nazivi	**Novi nazivi**
INNODB_SYS_COLUMNS	INNODB_COLUMNS
INNODB_SYS_DATAFILES	INNODB_DATAFILES
INNODB_SYS_FIELDS	INNODB_FIELDS
INNODB_SYS_FOREIGN	INNODB_FOREIGN
INNODB_SYS_FOREIGN_COLS	INNODB_FOREIGN_COLS
INNODB_SYS_INDEXES	INNODB_INDEXES
INNODB_SYS_TABLES	INNODB_TABLES
INNODB_SYS_TABLESPACES	INNODB_TABLESPACES
INNODB_SYS_TABLESTATS	INNODB_TABLESTATS
INNODB_SYS_VIRTUAL	INNODB_VIRTUAL

Tablica 1: Preimenovani prikazi InnoDB informacijske sheme

Uklonjene su sljedeće značajke u vezi s upravljanjem računom:

- Korištenje GRANT-a za stvaranje korisnika. Umjesto toga, koristite CREATE USER. Slijedeći ovu praksu, način rada NO_AUTO_CREATE_USER SQL nema značaja za GRANT izjave, pa se i on uklanja, a pogreška se sada zapisuje u zapisnik poslužitelja kada prisutnost ove vrijednosti za sql_mode opciju u datoteci s opcijama onemogućuje pokretanje mysqld-a.
- Upotreba GRANT-a za izmjenu svojstava računa koji nisu dodjeljivanje privilegija. To uključuje svojstva autentifikacije, SSL i ograničenja resursa. Umjesto toga, uspostavite takva svojstva u trenutku otvaranja računa pomoću CREATE USER ili ih naknadno modificirajte s ALTER USER.
- IDENTIFIED BY PASSWORD sinhranom „auth_string“ za CREATE USER i GRANT. Umjesto toga, koristite IDENTIFIEDED auth_plugin AS "auth_string" za CREATE USER

i ALTER USER, gdje je vrijednost "auth_string" u formatu kompatibilnom s imenovanim dodatkom.

- Nadalje, s obzirom na to da je IDENTIFIED BY PASSWORD sintaksa uklonjena, sistemska varijabla log_builtin_as_identified_by_password je suvišna i uklonjena je.
- PASSWORD () funkcija. Uz to, uklanjanje PASSWORD () znači da SET PASSWORD ... = PASSWORD ('auth_string') sintaksa više nije dostupna.
- Ukinuti zastarjeli ASC ili DESC kvalifikatori za odredbe GROUP BY uklanjaju se. Upiti koji su se prije oslanjali na razvrstavanje GROUP BY mogu proizvesti rezultate koji se razlikuju od prethodnih MySQL verzija. Da biste proizveli zadani redoslijed sortiranja, navedite odredbu ORDER BY
- Uklonjeni su neiskorišteni sistemske varijable date_format, datetime_format, time_format i max_tmp_tables. (Anon., n.d.)

4. Konceptualni dizajn

U ovom poglavlju razraditi ću bazu podataka na konceptualnoj razini. Konceptualna schema prva je faza kod oblikovanja baze podataka. Prikazuje se pomoću dijagrama, a sastoji se od entiteta, veza i atributa. Kod konceptualnog dizajna bavimo se odlukama koje attribute želimo čuvati u bazi, te kako ih grupirati u različite tablice.

Konceptualna shema daje zoran i jezgrovit prikaz baze koji je oslobođen tehničkih detalja. Moglo bi se reći da ta shema zapravo u prvom redu opisuje stvarni svijet o kojem želimo bilježiti podatke, a tek onda na posredan način i samu bazu. (Robert Manger, 2010.)

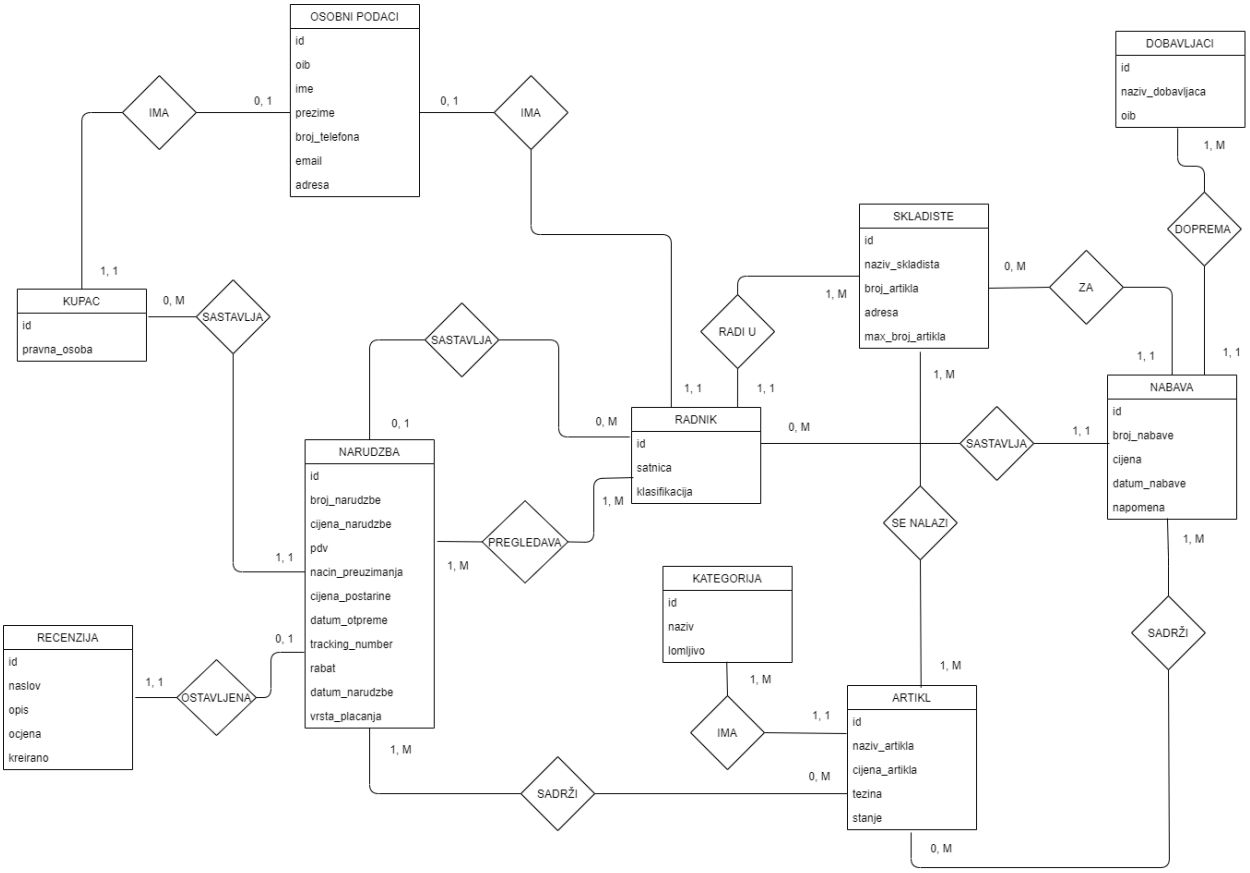
Dakle u konceptualnoj schemi cilj nije prikazivati tehničke detalje nego prikazati izgled baze na što jednostavniji i razumljiv način. Razmuljiva je ljudima svih struka te služi za komunikaciju projektanata i korisnika.

U toj komunikaciji korisnici nastoje utvrditi jesu li projektanti prepoznali sve poslovne procese, uključili sve potrebne podatke i ispravno shvatili odnose među tim podacima. (Robert Manger, 2010.)

4.1 Opis primjera

U svrhu ovog rada, izraditi ću bazu podataka za skladište trgovine. Baza će služiti za vođenje artikla koji se nalaze u skladištu te zaprimanje narudžbi istih. Istu bazu moguće je dodatno proširiti i dodati nove funkcionalnosti u budućnosti. Cilj je pregled i obrada narudžbi koje kupac izdaje te kontrola i upravljanje artiklima koji se nalaze u skladištu. Dakle, kupac koji se najvjerovatnije nalazi iza ekrana i preko webshopa naručuje, odabire artikle koje naručuje i tako sastavlja narudžbu. Tehnički, kupac ne mora sastaviti narudžbu, no može se u budućnosti vratiti i ponovo sastaviti još narudžbi te jedan kupac tako može imati više narudžbi. Kupac može biti i radnik iz skladišta te isti može sastaviti narudžbu. Narudžba u sebi sadrži artikle koje je kupac odabrao. Narudžba se sastoji samo od artikla koje je kupac odabrao, te se zato ne mora svaki artikl nalaziti u narudžbi. Radnik koji radi u nekom skladištu zaprima, tj. pregledava narudžbu. Nije obavezno da svaki radnik pregleda svaku narudžbu, iako jednu narudžbu može pregledati više radnika. Svaki radnik radi u jednom skladištu, a u istom skladištu može raditi više radnika. Svaki artikl se nalazi u jednom ili više skladišta. Svaki artikl ima određenu kategoriju u koju spada. U jednoj kategoriji može se nalaziti više artikla koji spadaju u istu, dok jedan artikl pripada samo jednoj kategoriji. Ukoliko nekog artikla fali mora se naručiti. To obavlja neki od radnika koji sastavlja nabavu za dobavljače. Ne moraju se svi radnici baviti nabavkom artikla, recimo da to rade samo radnici koji su šefovi skladišta i sl. Radnik može sastaviti više stavki nabavki ukoliko neki dobavljač ne dostavlja neki artikl pa je potrebno artikl naručiti kod drugog dobavljača ili ponovno naručuje. U nabavci se sadrže artikli koji su potrebni za nabavu, barem jedan ili više njih. Svaka nabavka je jedinstvena za svakog dobavljača, te ne mogu dva dobavljača imati istu nabavku, rade se dvije zasebna nabavke. Dobavljači mogu imati više nabavki. Također svaka stavka nabave se odnosi na jedno skladište koje na kraju može imati više stavki nabava. Na kraju kupac može, ali ne mora, ostaviti generalnu recenziju, te ocijeniti iskustvo narudžbe i dati ocjenu. Recenzija se odnosi na narudžbu, a može postojati samo jedna recenzija po narudžbi.

Skupovi entiteta koji čine bazu podataka prikazani su pravokutnicima, a nose imena kupac, narudžba, radnik, skladiste, artikl, osobni podaci, kategorija, recenzija. Svaki entitet povezan je vezom koja je prikazana u obliku romba te nosi ime u obliku glagola.



Slika 7: Konceptualna shema baze podataka

4.2 Entiteti i njihovi atributi

Modeliranje entiteta i veza promatramo:

– entiteti: stvari, bića, pojave ili događaji

– veze: odnosi među entitetima

– atributi: svojstva entiteta ili veza

Entitet je nešto o čemu želimo spremati podatke u našu bazu. Entitet može biti biće, objekt ili neki događaj. Opisani su atributima koji svaki ima svoju određenu vrstu podataka. Na primjer, atributi kupca su me, prezime, adresa, broj telefona i sl. Ako je atribut neko ime onda će biti varchar vrste podataka, ukoliko je neki cijeli broj onda je vrsta podataka int, itd. Atribut koji sam može imati svoje atribute trebamo smatrati entitetom. Na primjer entitet kupac može sadržavati atribut adresa koja ima svoje atribute kao što su ulica, država i dr. Tada atribut adresa pretvaramo u entitet s atributima ulica, država i dr. Ime entiteta zajedno sa pripadnim popisom atributa određuje tip entiteta. Atribut, ili skup atributa, čija vrijednost jednoznačno određuje primjerak entiteta je kandidat za ključ. Npr. za entitet kupac kandidat za ključ je kupac_id.

Kandidat za ključ je atribut ili skup atributa čije vrijednosti jednoznačno određuju primjerak entiteta zadanog tipa. Tip entiteta može imati više mogućih izbora za ključ, ali ne mogu postojati dva različita entiteta istog tipa s istim vrijednostima kandidata za ključ.

Dva različita člana ili primjerka entiteta ne mogu imati isti ključ. Ključ je jedinstven za svakog člana (primjerka) entiteta. Kada imamo više kandidata za ključ, možemo birati te jednim proglasiti primarnim ključem. (Robert Manger, 2010.)

Primjer jednog entiteta iz iz konceptualne sheme bio bi NARUDZBA s atributima, to izgleda ovako:

NARUDZBA (id, broj_narudzbe, cijena_narudzbe, pdv, nacin_preuzimanja, datum_otpreme, rabat, cijena_postarine, tracking_number, sifra_posiljke, datum_narudzbe, vrsta_placanja)

4.3 Veze i njihovi atributi

Pomoću veza između dva entiteta naglašavamo da su ti entiteti u međusobnom odnosu. Veza je međusobni odnos entiteta.

Veza se uvijek definira na razini tipova entiteta, no realizira se povezivanjem pojedinih primjeraka entiteta dotičnih tipova. (Robert Manger, 2010.)

Npr. za entitete artikl i skladište njihova veza je ‘se nalazi’. Tom vezom izražava se činjenica da se artikl nalazi u skladištu.

Veze koje se nalaze u primjeru iz poglavlja 4.1:

- SASTAVLJA između kupac i narudzba
- SASTAVLJA između radnik i nabava
- PREGLEDAVA između radnik i narudzba
- SADRŽI između narudzba i artikla
- SADRŽI između nabava i artikl
- RADI U između radnik i skladiste
- SE NALAZI između artikl i skladiste
- OSTAVLJENA između recenzija i narudzba
- IMA između artikl i kategorija
- IMA između kupac i osobni podaci
- IMA između radnik i osobni podaci
- ZA između nabava i skladiste
- DOPREMA između nabava i dobavljaci

4.4 Funkcionalnosti veza i kardinalnost

Shema E-R dijagrama može definirati ograničenja kojih se sadržaj baze mora pridržavati. Kardinalnost određuje ograničenja pridruživanja entiteta. Kardinalnost ili omjer kardinalnosti označava broj entiteta kojim se drugi entitet može pridružiti putem skupa odnosa. Način na koji veza povezuje primjerke entiteta ima svojstva funkcionalnosti i obaveznog članstva, tj. kardinalnosti.

Poznavanje tih svojstava važno je da bi se veza u idućoj fazi projektiranja ispravno prikazala u relacijskoj shemi. Ako promatramo vezu između dva entiteta imenovanim E1 i E2, Funkcionalnost te veze je svojstvo koje kaže je li za odabrani primjerak entiteta jednog tipa moguće jednoznačno odrediti povezani primjerak entiteta drugog tipa. Drugim riječima, funkcionalnost je svojstvo koje kaže može li se veza interpretirati kao preslikavanje (funkcija) iz skupa primjeraka entiteta jednog tipa u skup primjeraka entiteta drugog tipa. (Robert Manger, 2010.)

Veza se može promatrati u dva smjera iz entiteta E1 do entiteta E2 i obratno, postoje četiri vrste funkcionalnosti prikazane u tablici.

OZNAKA	NAZIV	OPIS
1:1	Jedan-naprama-jedan	Jedan primjerak od E_1 može biti povezan najviše s jednim primjerkom od E_2. Također, jedan primjerak od E_2 može biti povezan najviše s jednim primjerkom od E_1.
1:M	Jedan-naprama-mnogo	Jedan primjerak od E_1 može biti povezan s više primjeraka od E_2. Istovremeno, jedan primjerak od E_2 može biti povezan najviše s jednim primjerkom od E_1.
M:1	Mnogo-naprama-jedan	Jedan primjerak od E_1 može biti povezan najviše s jednim primjerkom od E_2. Istovremeno, jedan primjerak od E_2 može biti povezan s više primjeraka od E_1.
M:M	Mnogo-naprama-mnogo	Jedan primjerak od E_1 može biti povezan s više primjeraka od E_2. Također, jedan primjerak od E_2 može biti povezan s više primjeraka od E_1.

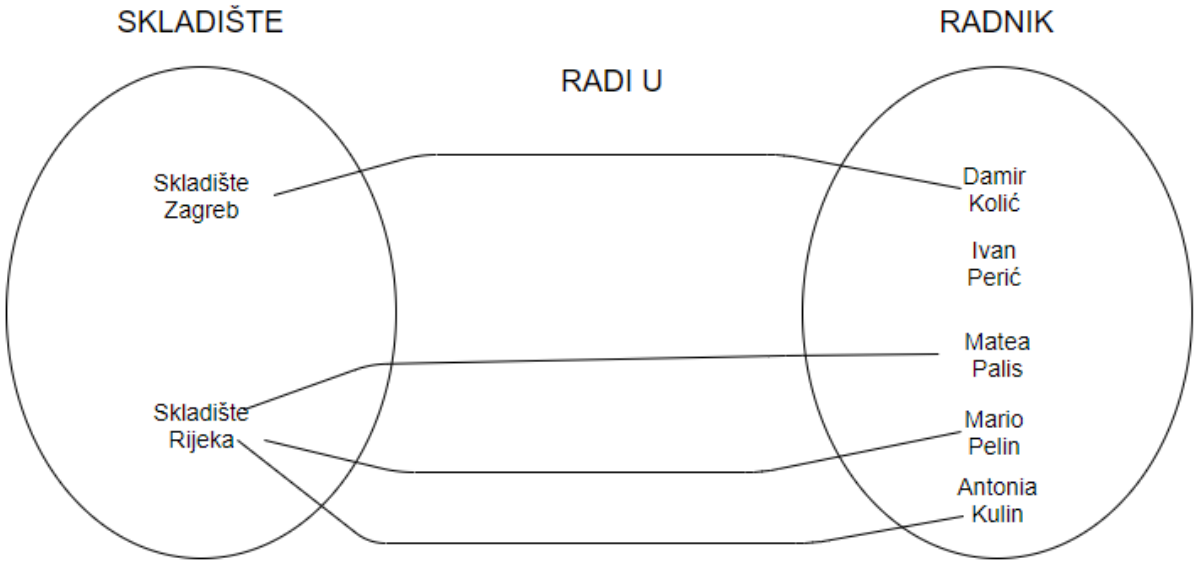
Tablica 2: Vrste funkcionalnosti za vezu između tipova entiteta E1 i E2 (Robert Manger, 2010.)

Kao primjer za funkcionalnost 1:M navodim vezu RADI U između entiteta SKLADIŠTE i RADNIK. Radnik može raditi u samo jednom skladištu, dok skladište može imati više radnika koje radi u istom.

Kažemo da E1 ima obavezno članstvo u toj vezi ako svaki primjerak od E1 mora sudjelovati u vezi, dakle mora biti povezan barem s jednim primjerkom od E2. Analogno se definira i obaveznost članstva za E2. (Robert Manger, 2010.)

Pogledajmo vezu se nalazi između artikl i skladište. Tip entiteta ARTIKL nema obavezno članstvo jer se artikl ne mora nalaziti u skladištu.

Jedan primjerak entiteta radnik može biti povezan samo s jednim primjerkom entiteta skladište. U ovom slučaju nije moguće da jedan radnik radi u više skladišta istovremeno. Jedan primjerak entiteta skladište može istovremeno biti povezan s više primjerka entiteta narudžba. U jednom skladištu može raditi više radnika.



Slika 8: Veza s funkcionalnošću 1:M

5. Logički dizajn

U ovom poglavlju baviti ću se projektiranjem baze podataka na logičkoj razini. Obraditi ću svojstva relacijskog modela, izradom relacijske scheme te kasnije dotjerivanjem, odnosno normalizacijom iste. Cilj je stvoriti relacijsku shemu koja će opisivati logčku strukturu baze sukladno pravilima relacijskog modela.

5.1 Relacijski model

Relacijska schema manje je razumljiva krajnjim korisnicima jer su entiteti i veze između entiteta pretvoreni u relacije pa se teško razlikuje jedan od drugog. Može se više-manje izravno implementirati pomoću današnjih DBMS-a, te zahvaljujući današnjem softveru, od relacijske sheme do njezine konačne implementacije vrlo je kratak put. Relacijska shema sastoji se od popisa atributa i njihovih domena. Koncept instance relacije odgovara pojmu vrijednosti varijable. Kako se vrijednost varijable može mijenjati, tako se i sadržaj relacije s vremenom može mijenjati kako se relacija ažurira. Shema relacije se ne mijenja.

Relacijski model koristi zbirku tablica za predstavljanje podataka i odnosa među tim podacima. Svaka tablica ima više stupaca, a svaki stupac ima jedinstveno ime. Tablice su također poznate kao odnosi. Relacijski model primjer je modela zasnovan na zapisu. Modeli zasnovani na zapisima tako su nazvani jer je baza podataka strukturirana u zapisi fiksnog formata nekoliko vrsta. Svaka tablica sadrži zapise određene vrste. Svaka vrsta zapisa definira fiksni broj polja ili atributa. Stupci tablice odgovaraju atributima vrste zapisa. Relacijski model podataka najčešće je korišten model podataka, a velika većina postojećih sustava baza podataka temelji se na relacijskom modelu. (Abraham Silberschatz, Henry F. Korth, S. Sudarshan, 2010.)

5.2 Relacija, atribut, n-torka

Relacijski model zahtijeva da se baza podataka sastoji od skupa pravokutnih tablica – relacija. Svaka relacija ima svoje ime. Stupac relacije sadrži vrijednost atributa. Za svaki atribut je definiran tip dozvoljenih podataka. Vrijednosti jednog atributa su podaci iste vrste. Moguće je da postoji atribut koji nema vrijednost.

Znači, za atribut je definiran tip ili skup dozvoljenih podataka koji se zovu domena atributa. Vrijednost atributa mora biti jednostruka i jednostavna, što znači da se ne ponavlja i ne rastavlja na dijelove. Jedan redak relacije obično predstavlja jedan primjerak entiteta ili bilježi vezu između dva ili više primjeraka. Taj redak nazivamo n-torka. U jednoj relaciji ne smiju postojati dvije jednake n-torke, naime relaciju tumačimo kao skup n-torki. Broj atributa se zove stupanj relacije, a broj n torki je kardinalnost relacije. (Robert Manger, 2010.)

Relaciju tumačimo kao skup n-torki stoga u jednoj relaciji ne smiju postojati dvije n-torke.

Broj atributa se zove stupanj relacije, a broj n-torki je kardinalnost relacije.

Tablica prikazuje primjer relacije OSOBNI PODACI s atributima ID, OIB, IME, PREZIME, BROJ_TELEFONA, EMAIL.

ID	OIB	IME	PREZIME	BROJ TELEFONA	EMAIL
1	056486564	DARKO	PEJIĆ	095648564	Darko001@mail.com
2	546853150	PETAR	RADIĆ	0995646864	rapetar@gmail.com
3	231847645	DENIS	PULIĆ	0975468246	puldenis@gmail.com

Tablica 3: Primjer relacije s osobnim podacima

5.3 Kandidat za ključ, primarni ključ

Ključ K relacije R je podskup skupa atributa od R sa sljedećim svojstvima:

1. Vrijednosti atributa iz K jednoznačno određuju n-torku u R. Dakle u R ne mogu postojati dvije n-torke s istim vrijednostima atributa iz K.

2. Ako iz K izbacimo bilo koji atribut, tada se narušava svojstvo 1.

Ta su svojstva „vremenski neovisna“, u smislu da vrijede u svakom trenutku bez obzira na povremene unose, promjene i brisanja n-torki. Budući da su sve n-torke u R međusobno različite, K uvijek postoji. Naime, skup svih atributa zadovoljava svojstvo 1. Izbacivanjem suvišnih atributa doći ćemo do podskupa koji zadovoljava i svojstvo 2. (Robert Manger, 2010.)

Sve vrste podataka mogu funkcionirati kao primarni ključ. Osobni identifikacijski brojevi, JMBAG, email adrese mogu funkcionirati kao primarni ključ, podaci samo moraju biti jedinstveni. Na primjer, promatrajući tablicu 3 relacija s osobnim podacima, atribut OIB čini kandidat za ključ jer je oib osobni identifikacijski broj te je za svaku osobu posebno određen i različit. Atributi ime i prezime nisu ključ jer se mogu ponavljati. Može se dogoditi da relacija ima više kandidata za ključ. Recimo u relaciji s osobnim podacima ključ bi mogao biti oib i broj telefona. Tada od dva kandidata za ključ odabiremo jedan za primarni ključ. Atribut broj telefona bi mogao biti ključ no može se desiti da broj telefona promijeni korisnika ili sl. te zato ne čini ključ. Zbog sigurnosti kako se oib možda u budućnosti neće koristiti kao što se jmbg više ne koristi, primarni ključ je id. Vrijednost primarnog ključa nikada nesmije biti neupisana, prazna.

Primarni Ključ je stupac ili skup stupaca pomoću kojih se svaki redak može jedinstveno identificirati. Oni su velike važnosti, te je svaki jedinstven i neponovljiv. U bazi podataka ne smiju postojati dva identična primarna ključa. Primarni ključ je većinom niz jedinstvenih brojeva, gdje se u svakom retku u tablici vrijednost primarnog ključa povećava za jedan. MySQL sadrži funkciju auto_increment za primarne ključeve, te se onda svakom novom kreiranom retku u tablici primarni ključ automatski zadaje, te je svaki sljedeći za jednu vrijednost veći.

5.4 Pretvaranje konceptualne u relacijsku schemu

U ovom poglavlju obraditi ću kako se iz cijele konceptualne sheme dobiva relacijska shema.

Krećemo od pretvorbe entitea i atributa.

Svaki tip entiteta prikazuje se jednom relacijom. Atributi tipa postaju atributi relacije. Jedan primjerak entiteta prikazan je jednom n-torkom. Primarni ključ entiteta postaje primarni ključ relacije. (Robert Manger, 2010.).

Na primjer, tip entiteta artikl prikazuje se relacijom: ARTIKL (šifra artikla, naziv artikla, količina, težina, stanje, cijena artikla).

Tako ćemo sve entitete prikazati sa relacijom,

KUPAC	id, adresa, pravna_osoba
OSOBNI PODACI	id, oib, ime, prezime, broj_telefona, email
NARUDZBA	id, broj_narudzbe, cijena_narudzbe, pdv, nacin_preuzimanja, cijena_postarine, datum_otpreme, tracking_number, rabat, datum_narudzbe, nacin_placanja
RADNIK	id, satnica, klasifikacija
SKLADISTE	id, adresa, broj_artikla, max_broj_artikla
ARTIKL	id, naziv_artikla, količina, cijena_artikla, težina, stanje
KATEGORIJA	id, naziv, lomljivo
RECENZIJA	id naslov, opis, ocjena, kreirano
NABAVA	id, broj_nabave, cijena, datum_nabave, napomena
DOBAVLJACI	id, naziv_dobavljaca, oib

Tablica 4: Pretvorba entiteta u relacije iz baze podataka

Sljedeće je pretvorba veza jedan naprama mnogo.

Ako tip entiteta E1 ima obavezno članstvo u vezi s tipom E2 koja ima funkcionalnost M:1, tada relacija za E1 treba uključiti primarne atribute od E2. (Robert Manger, 2010.)

Na primjer, promotrimo konceptualnu schemu sa slike 7. Veza “ostavljena” između recenzija i narudzba, svaka recenzija mora pripadati nekom kupcu, te se zbog veze “ostavljena” u relaciju recenzija ubacuje ključ relacije narudzba_id. Taj ključ nazivamo strani ključ jer je on prepisan iz relacije u neku drugu relaciju.

Relacija recenzija izgledati će ovako:

RECENZIJA (šifra_recenzije, naslov, opis, ocjena, kreirano, narudzba_id)

NARUDŽBA	id, broj_narudzbe, cijena_narudzbe, pdv, nacin_preuzimanja, cijena_postarine, datum_otpreme, tracking_number, rabat, datum_narudzbe, nacin_placanja, kupac_id, radnik_id
RADNIK	id, satnica, klasifikacija, skladiste id, osobni podaci id
SKLADIŠTE	id, Naziv_skladišta, adresa, broj_artikla, max_broj_artikla
ARTIKL	id, naziv_artikla, količina, cijena_artikla, težina,stanje, kategorija_id
KUPAC	id, pravna_osoba, osobni podaci id
RECENZIJA	id, naslov, opis, ocjena, kreirano, narudzba_id
NABAVA	id, broj_nabave, cijena, datum_nabave, napomena, radnik_id, dobavljaci id
DOBAVLJACI	id, naziv_dobavljaca, oib
OBRADA_NARUDZBE	radnik_id, skladiste_id
STAVKA_ARTIKLI	narudzba_id, artikl_id, popust
SADRZAJ_SKLADISTA	artikl_id, skladiste_id
STAVKA_NABAVE	artikl_id, nabava id, kolicina artikla

Tablica 5: Relacije sa stranim ključevima nakon pretvorbe veza jedan naprama mnogo

Veze koje imaju kardinalnost M:M prikazuju se posebnom relacijom koja se sastoji od primarnih atributa za oba tipa entiteta.

Uzmimo za primjer vezu “sadrzaj_skladista” prikazana relacijom:

sadrzaj_skladista (narudžba id, artikl id)

Jedan primjerak od entiteta artikl može biti povezan s više primjerka entiteta narudžbe, i obrnuto. Dakle jedan artikl može se nalaziti u više narudžbi, te jedna narudžba može se sastojati od više artikla. Ključ relacije “sadrži” sastoji se od kombinacije atributa narudzba_id i artikl_id. Da bismo pronašli sve artikle koje neka narudžba sadrži, pretražujemo relaciju “sadrzaj_skladista” te

izdvajamo sve n-torke koje odgovaraju vrijednosti narudzba_id. Svaka od izdvojenih n-torki sadrži artikl_id koji se nalazi u narudžbi. Isto tako i obratno, kako bi pronašli sve narudžbe u kojoj se nalazi artikl, pretražujemo relaciju "sadrzaj_skladista" te izdvojimo n-torke s odgovarajućom vrijednosti za artikl_id. Svaka od izdvojenih n-torki otkriva artikl_id jednog od traženog artikla.

Iz sheme se ne vide tipovi atributa. Kod određivanja tipova atributa gleda se ime atributa te se prema imenu tog atributa određuje tip atributa. Ponekad se tip atributa ne može odrediti iz imena atributa. Izrađuje se rječnik podataka te se on priloži uz shemu. Rječnik podataka je tablica u kojoj su zapisani svi atriubuti, za svakog od njih je definiran tip i definiran kratki opis. Kod određivanja tipova atributa treba paziti na ograničenja tog tipa. Na primjer, ukoliko za atribut cijena postavimo da je tipa integer, kasnije se cijena može zapisati i prikazati samo u cijelom broju, bez točke ili zareza, te nije moguće izraziti cijenu u lipama. Zato za atribut cijena odredimo tip decimal.

Rječnik podataka stvaramo jer se iz konceptualne sheme ne vide tipovi atributa, i ponekad se samo iz imena atributa ne može lako odrediti njihovo značenje. Rječnik podataka je tablica u kojoj su popisani svi atributi, isti atribut je upisan samo jednom, te je za svaki atribut definiran tip i opisano značenje. Stvara se tako da prođemo svim relacijama iz sheme i upišemo u tablicu. Zatim se atributima na što razumljiviji način opiše značenje i odredi atribut.

Kod određivanja tipova treba uzeti u obzir da današnji DBMS-ovi očekuju da će atributi biti cijeli ili realni brojevi, znakovi ili nizovi znakova, datumi ili novčani iznosi. Znači, za svaki atribut treba odrediti tip u jednom od ovih oblika, s time da taj tip možemo preciznije podesiti uvođenjem raznih ograničenja. (Robert Manger, 2010.)

IME ATRIBUTA	TIP	OPIS
Nacin preuzimanja	Niz znakova	Osobno preuzimanje ili poštom
Cijena narudzbe	Decimalni broj	Određuje cijenu narudžbe
Pdv	Cijeli broj	Opisuje visinu pdv-a
Cijena postarine	Niz znamenki	Cijena poštarine
Rabat	Mali cijeli broj	Određuje rabat na narudžbu
OIB	Cijeli broj	Označava oib kupca
Naziv skladista	Niz znakova	Ime skladišta
Adresa	Niz znakova	Opisuje adresu
Broj artikla	Cijeli broj	Određuje broj artikla
Naziv artikla	Niz znakova	Naziv koji opisuje artikl
Cijena artikla	Decimalni broj	Cijena određenog artikla
Količina	Cijeli broj	Opisuje količinu artikla
Ime	Niz znakova	Ime osobe

prezime	Niz znakova	Prezime osobe
Datum	Niz znakova	datum
Tezina	Niz znakova	Težina artikla u kg, gr ili sl.
Klasifikacija	Niz znakova	Određuje vrstu tj. privilegije nekog radnika
Satnica	Decimalni broj	Cijena radnog sata radnika
Max broj artikla	Cijeli broj	Maksimalni broj artikla
Datum narudzbe	Datum	Datum sastavljajna narudžbe
Vrsta placanja	Niz znakova	Način plaćanja (Gotovina, kartica, na rate)
Email	Niz znakova	Email kupca
Naslov	Niz znakova	Naslov recenzije ili emaila
Opis	Niz znakova	Tekst recenzije ili emaila
Ocjena	Mali cijeli broj	ocjena recenzije
Kreirano	Datum	Datum kreiranja recenzije
Napomena	Niz znakova	Dodatni tekst kao napomena
Nacin_preuzimanja	Niz znakova	Određuje način preuzimanja, osobno ili poštom
Datum_otpreme	Datum	Datum slanja pošiljke
Tracking_number	Niz znamenki	Šifra za praćenje pošiljke
Lomljivo	Boolean	Određuje da li su artikli u kategoriji lomljivi
Stanje	Niz znakova	Opisuje stanje artikla
Pravna osoba	Boolean	Označuje je ili nije pravna osoba

Tablica 6: Rječnik podataka za bazu podataka

6. Normalizacija

Normalizacija je metoda za dizajniranje relacijske baze podataka. Cilj je izrada sheme koja omogućuje pohranu podataka izbacujući višak kako bi pristup tim podacima bio lakši. Primjenom normalizacije podaci su spremljeni na pravo mjesto. Ako su vi podaci na jednom mjestu, dohvaćanjem različitih ali međusobno povezanih podataka, znači odlazak na mnogo različitih mjesta što usporava process. Ažuriranje podataka je brže jer se obavlja samo na jednom mjestu. Podrazumijeva se da su sve relacijske baze normalizirane.

Sa normaliziranim modelom podataka podatak je spremljen na jedno mjesto a podaci o jednom entitetu spremljeni su zajedno (Koraljka B., Damir M., n.d.)

Dotjerivanje je potrebno zato što polazna relacijska shema može sadržavati određene nepravilnosti. Te nepravilnosti treba otkloniti prije nego što krenemo u projektiranje na fizičkoj razini. Sam postupak dotjerivanja zove se normalizacija. (Robert Manger, 2010.)

Obično se susrećemo s 3 normalne forme koje su dovoljne za svakodnevne potrebe, no teorija normalizacije tu ne staje.

Normalne forme:

- 1. normalna forma - svi entiteti moraju imati jedinstveni identifikator (ključ) koji se može sastojati od jednog ili više atributa. Svako polje u tablici mora sadržavati samo jednu vrijednost (atributi moraju biti jednostavni – ne smiju biti sastavljeni od više atributa)
- 2. normalna forma - svi atributi koji nisu dio ključa moraju u potpunosti ovisiti o njemu
- 3. normalna forma - svi atributi koji nisu dio ključa ne smiju biti međusobno ovisni

Osnovna pravila normaliziranih tablica:

- svako polje tablice trebalo bi predstavljati jedinstven tip informacije
- svaka tablica mora imati primarni ključ koji se sastoji od jednog ili više polja u tablici
- za svaku jedinstvenu vrijednost primarnog ključa mora postojati jedna i samo jedna vrijednost u bilo kojem stupcu podataka, a ta se vrijednost mora odnositi na subjekt tablice
- mora postojati mogućnost promjene bilo kojeg polja (osim polja u primarnom ključu) bez utjecaja na bilo koje drugo polje

(Koraljka B., Damir M., n.d.)

Svaka normalna forma predstavlja određeni „zahtjev na kvalitetu“ koji bi relacija trebala zadovoljavati. Što je normalna forma viša, njezini zahtjevi su stroži. (Robert Manger, 2010.)

Normalizaciju postižemo dijeljenjem podataka u više tablica. Umjesto jedne tablice imamo više tablica koje imaju određenu svrhu. Na taj način baza se lako proširuje i može sadržavati više vrsta podataka. Podacima se lakše pristupa i manipulira jer su razdvojeni na više jedinstvenih, smislenih tablica.

6.1 Prva normalna forma

Prva normalna forma (1NF) je zapravo izmišljeni pojam zbog drugih modela podataka i ne predstavlja posebne zahtjeve na relaciju. 1NF ima svojstva koja su već ugrađena u relacijskom modelu, te zato niti ne može postojati relacijska baza podataka koja nije u prvoj normalnoj formi.

Vrijednost atributa unutar relacije (sadržaj jedne kućice unutar tablice) mora biti:

- jednostruka - u jednu kućicu ne može se upisati više vrijednosti, nego najviše jedna vrijednost
- jednostavna - upisana vrijednost ne smije biti složena, nego takva da ju sama baza smatra nedjeljivom

Dakle, kažemo da je relacija u 1NF, jer su vrijednosti njezinih atributa jednostruke i nedjeljive.

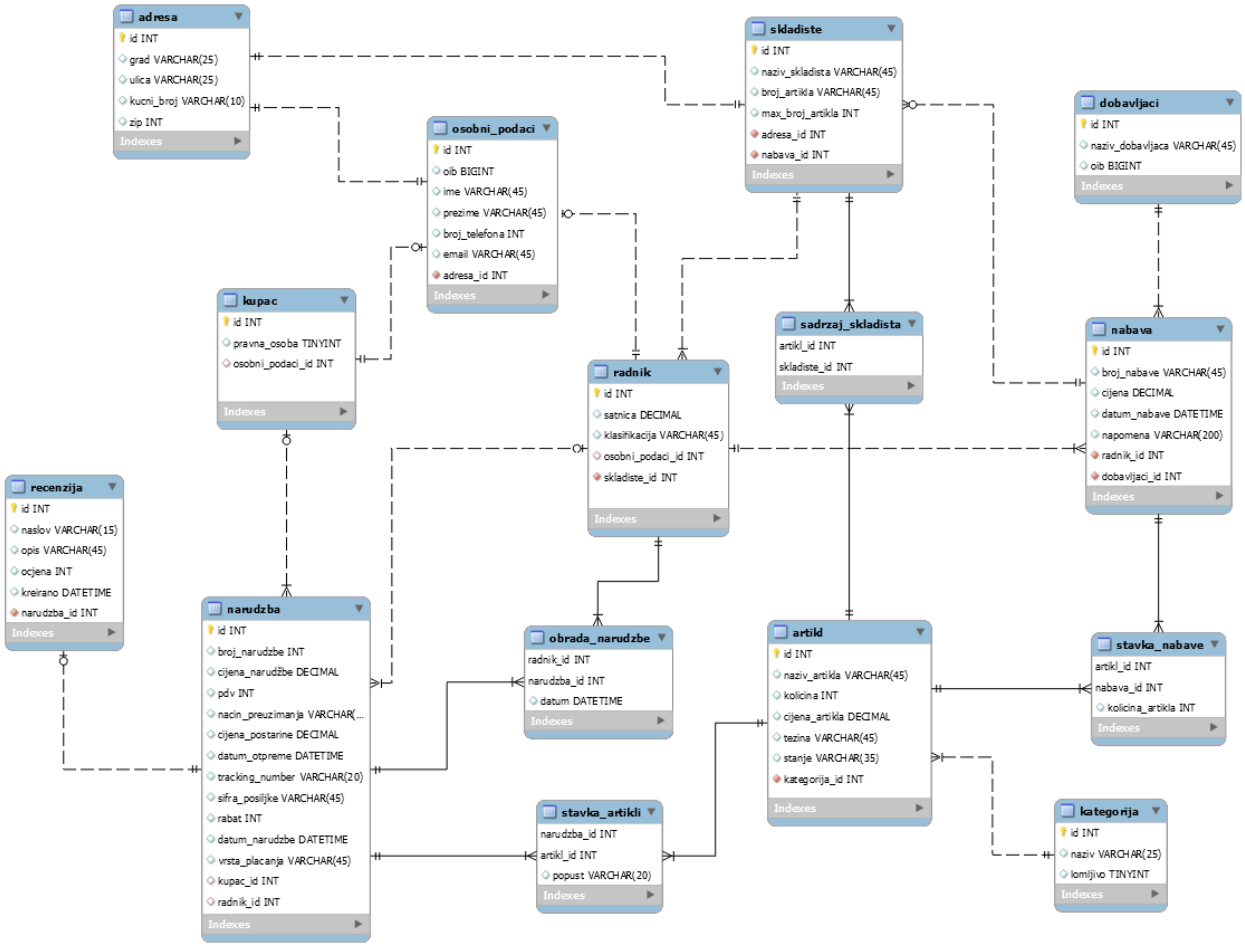
Da bismo nenormalizirane podatke pohranili u relacijskoj bazi, moramo ih prevesti u 1NF. Dakle, uvođenjem dovoljnog broja relacija i atributa moramo eliminirati sve ponavljajuće skupine i podzapise.

Postupak prevođenja izgleda otprilike ovako:

- Uvodi se jedna osnovna relacija i onoliko pomoćnih relacija koliko ima ponavljajućih skupina.
- Fiksni dio zapisa prikazuje se kao jedna n-torka u osnovnoj relaciji, s time da je svaki podzapis rastavljen na nekoliko zasebnih atributa.
- Svaka pojava ponavljajuće skupine u zapisu prikazuje se kao zasebna n-torka u odgovarajućoj pomoćnoj relaciji. Zbog čuvanja veze s polaznim zapisom u tu se n-torku prepisuje i neki od identifikacijskih podataka iz fiksnog dijela zapisa.

(Robert Manger, 2010.)

Za primjer uzmimo tablicu s osobnim podacima. Atribut adresa može sadržavati više različitih podataka kao što su ulica, kučni broj, grad i sl. Isti atribut se nalazi u talici skladište. Zato uvodimo još jednu relaciju s nazivom adresa i dodajemo joj attribute id, grad, ulica, kučni broj, zip koji opisuju tu relaciju.



Slika 9: logička shema nakon prevođenja u 1NF

6.1.1 Parcijalna ovisnost

Ako u relaciji postoje atributi koji nisu potpuno funkcionalno ovisni o primarnom onda za njih kažemo da su parcijalno ovisno o tom ključu. U relaciji “sadrži” atribut cijena artikla je potpuno funkcionalno ovisan o primarnom ključu (šifra artikla, šifra narudžbe). Zato atributi adresa, poštarina, pdv i rabat nisu potpuno funkcionalno ovisni o primarnom ključu, jer su ovisni samo o šifra narudžbe ne i o šifra artikla

Parcijalna ovisnost smatra se nepoželjnim svojstvom. Naime ona može uzrokovati teškoće kod manipuliranja s podacima, zato se uvodi pojam druge normalne forme. (Robert Manger, 2010.)

6.2 Druga normalna forma

Relacija je u drugoj normalnoj formi (oznaka: 2NF) ako je svaki njezin neprimarni atribut potpuno funkcionalno ovisan o primarnom ključu. Drugim riječima, relacija je u 2NF ako u njoj nema parcijalnih ovisnosti atributa o primarnom ključu. Definicija 2NF zaista ima smisla jedino ako je primarni ključ relacije složen. Relacija s jednostavnim ključem (dakle ključem koji se sastoji samo od jednog atributa) automatski je u 2NF. (Robert Manger, 2010.)

Relacijska shema je u drugoj normalnoj formi ako svaki atribut zadovoljava da se pojavljuje u kandidatu za ključ ili nije djelomično ovisan o ključu kandidata.

U ovom primjeru baza podatka već se nalazi u 2NF jer je u 1NF, svi njeni neključni atributi funkcionalno zavise od primarnog ključa. Baza nebi bila u 2NF kada relacija KUPAC nebi postojala već bi njezini atributi bili atributi relacije NARUDŽBA. Tada neki atributi relacije NARUDŽBA nebi funkcionalno zavisili od primarnog ključa te bi se relacija razdvojila u dvije, narudžba i kupac.

6.2.1 Tranzitivne ovisnosti

Tranzitivna ovisnost je korak u treću normalnu formu i smatra se nepoželjnim svojstvom. Da bi shemu preveli u 3NF, shema ne smije sadržavati tranzitivne ovisnosti. Tranzitivna ovisnost je situacija kada postoji stupac u tablici koji direktno ne opisuje primarni ključ. Svakoj vrijednosti jednog atributa mora odgovarati samo jedna vrijednost drugog atributa.

Ako relacije nisu normalizirane, dolazi do teškoća kod unosa, promjene i brisanja podataka. Bez obzira o kojoj normalnoj formi je riječ, teškoće su otprilike slične. Slično kao i parcijalna ovisnost, i tranzitivna se ovisnost smatra nepoželjnim svojstvom. Naime, i ona može uzrokovati slične teškoće kod manipuliranja s podacima. (Robert Manger, 2010.)

Brzi način za ući u treći normalni oblik je pogledati sva polja u tablici i pitati opisuju li sva primarni ključ. (B. Bulger, J. Greenspan, D. Wall, 2004.)

6.3 Treća normalna forma

Relacijska shema je u trećoj normalnoj formi (3NF) ako nema neprimarnih atributa koji ovise o primarnom ključu. Znači da je shema u 2NF i ne sadrži tranzitivne ovisnosti. Prevođenje u 3NF slično je kao kod 2NF. Prekidaju se tranzitivne ovisnosti tako da se relacija prepolovi, tj. razbije u barem dvije relacije. Moramo pripaziti da prilikom razbijanja relacije ne dođe do guitka informacija.

- Uz polaznu relaciju dodaje se onoliko novih relacija koliko ima različitih atributa koji se pojavljuju kao srednji atributi u tranzitivnim ovisnostima.
- Iz polazne relacije izbacuju se i prebacuju u nove svi oni atributi koji su tranzitivno ovisni.
- Pritom u jednu novu relaciju idu oni atributi u čijim se tranzitivnim ovisnostima pojavljuje isti srednji atribut.
- Uz prebačene atribute, u novu se relaciju prepisuje i odgovarajući srednji atribut pa on postaje ključ u toj novoj relaciji.

Kako bi provjerili treću normalnu formu na primjeru baze podataka, uzet ćemo nekoliko tablica s atributima iz primjera baze. Promatrajući tablice ispod gledamo postoji li tranzitivna ovisnost, tj. postoji li redak u tablici koji ne opisuje primarni ključ.

id	Naziv_skladista	Broj_artikla	Max_broj_artikla
551	Skladište Zagreb 01	1500	1600
552	Skladište Rovinj	150	230
553	Skladište Zadar	130	290

Tablica 7: Tablica SKLADISTE s atributima

id	Naziv artikla	Količina	Cijena artikla	Težina	Stanje
11	TV Samsung S08UZ12	11	3899.99	5kg	Uredu
12	Slušalice airpods	13	760	590gr	uredu

Tablica 8: Tablica ARTIKL s atributima

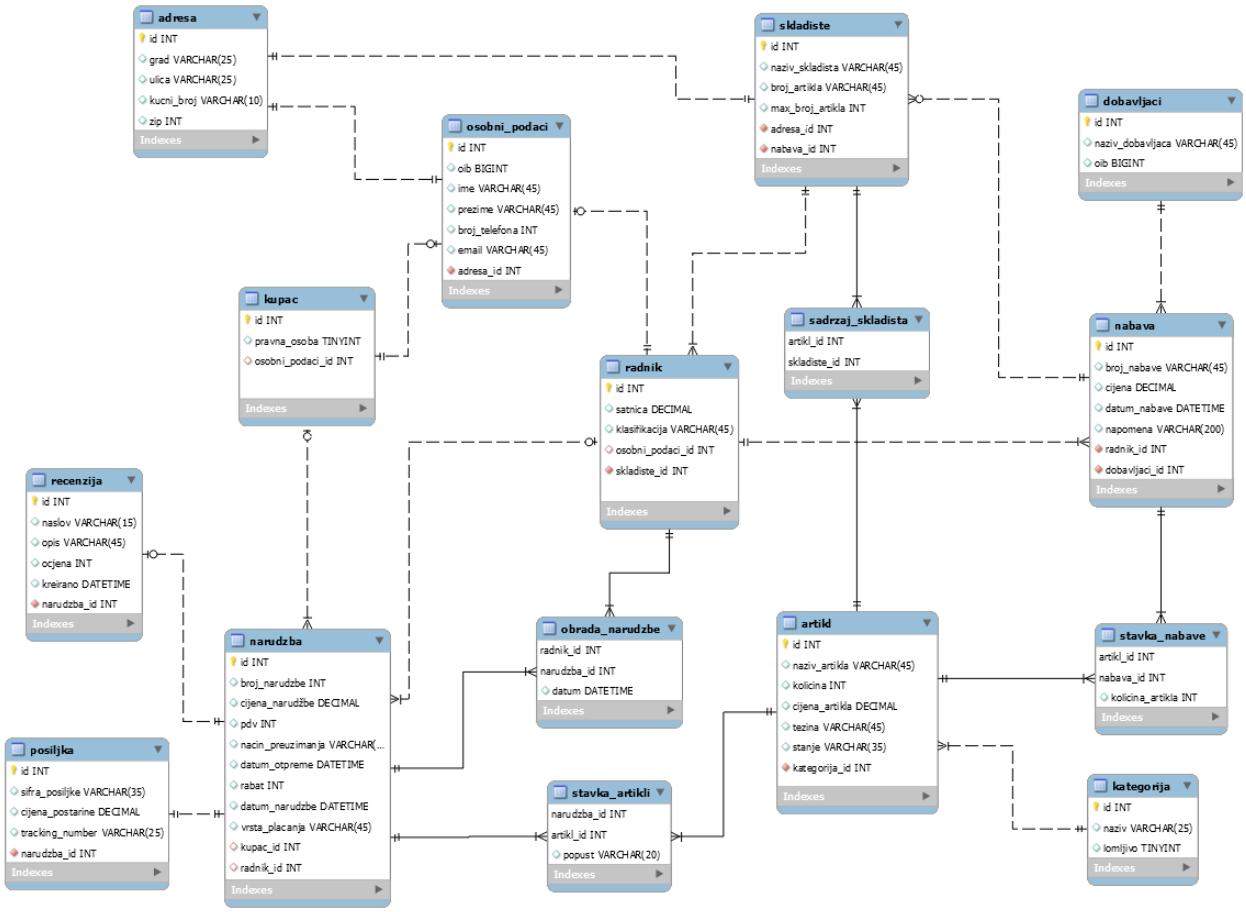
šifra narudžbe	cijena narudžbe	pdv	način preuzimanja	cijena poštarine	datum otpreme	tracking number	šifra pošiljke	rabat	datum narudžbe	vrsta plaćanja
115466	250	25	osobno preuzimanje	0	2020-08-10 22:30:59	0	A4756	0	2020-08-11 11:10:19	gotovina
112456	359	25	poštom	40	2020-08-11 20:10:19	BC74635E	B77126	0	2020-08-12 10:10:59	gotovina
228546	1250	25	poštom	40	2020-08-13 11:33:49	AY4756S	A11514	0	2020-08-14 21:30:19	kartice

Tablica 9: Tablica NARUDŽBA s atributima

Promotrimo relaciju narudžba. Relacijska shema nije u 3NF jer se u tablici nalazi atribut koji ovisi o drugom atributu. Atributi cijena poštarine i tracking number vezani su uz pošiljku, tj. o atributu šifra pošiljke jer je ujedno taj atribut jedinstven i kandidat za ključ. Tablicu ćemo dekompozirati u dvije; narudžba i pošiljka. Obje tablice su sada u 3NF i izgledaju ovako:

NARUDZBA (id, broj_narudzbe, cijena_narudzbe, pdv, nacin_preuzimanja, datum_otpreme, rabat, datum_narudzbe, vrsta_placanja)

POSILJKA (id, cijena_postarine, tracking_number, sifra_posiljke)



Slika 10: Logička shema nakon prevođenja u 3NF

Najpoželjniji normalni oblik koji se može dobiti je boyce – codd normalni oblik, skraćeno BCNF. Takav oblik normalizacije eliminira svu suvišnost koja se temelji na funkcijskim ovisnostima. Postupak prevođenja je sličan kao kod 2NF ili 3NF. Relacija se razbija na više manjih tako da se pritom prekinu nepoželjne ovisnosti. Shema baze podataka je u BCNF ako nema funkcijskih ovisnosti, odnosno svaka determinanta relacije je ujedno i kandidat za ključ.

Definicija Boyce-Coddove normalne forme zasnovana je na pojmu determinante. Determinanta je atribut ili kombinacija atributa u nekoj relaciji o kojoj je neki drugi atribut u istoj relaciji potpuno funkcionalno ovisan. Relacija je u Boyce-Codd-ovoj normalnoj formi ako je svaka njezina determinanta ujedno i kandidat za ključ. Primjeri relacija koje su u 2NF i 3NF, a nisu u BCNF zapravo su vrlo rijetki. (Robert Manger, 2010.)

Za tablicu se kaže da je BCNF (Boyce-Coddov normalan oblik) ako mogu biti sve relacijske tablice identificiran pomoću jednog primarnog ključa iz glavne tablice. (S. Challawala, J. Lakhatariya, C. Mehta, K. Patel, 2017.)

7. Fizička građa baze

Fizička shema baze su zapravo naredbe u SQL-u ili nekom drugom jeziku koji razumije sustav za upravljanje bazom podataka. Izvođenjem tih naredbi sustav za upravljanje bazom podataka stvara fizižku građu baze. Najvažnija SQL naredba u fizičkoj shemi je create table. Tom jednom naredbom definirana je relacija s nazivima i tipovima atributa. Moguće je zadati koji atribut je primarni ključ, te postaviti auto_increment funkciju za automatsku dodjelu novog primarnog ključa u svakom novom retku. Također je moguće zadati koji atribut smije ili ne smije imati neupisane vrijednosti (prazan stupac). Generirana shema ima .sql ekstenziju.

Baza podataka se pohranjuje na jednom ili više uređaja za pohranu. Pohranjuje se na magnetski disk, ili sve češće na flash memoriju. Mediji na kojima se može čuvati baza podataka su predmemorija, glavna memorija, flash memorija, magnetski disk, optički disk i magnetska vrpca. Predmemorija je najbrži, ali relativno mali i skup oblik pohrane. Glavna memrija može biti od nekoliko gigabajta do nekoliko tisuća gigabajta. Obično je premalena ili preskupa za pohranu cijele baze. Zahtjeva stalno napajanja, u slučaju da ostane bez napajanja sustav pada i sadržaj se može izgubiti. Naprotiv, flash memorija čuva podatke ako ostane bez napajanja. Najčešći medij za pohranu baze podataka je magnetski disk. Imaju puno veći prostor za pohranu podataka. Pohranjeni podaci na disku ostat će sačuvani i bez napajanja. Može se desiti kvar na disku i uništiti podatke ali je to jako rijetko, rjeđe od pada sustava. Optički disk se rijetko koristi zbog male memorije za pohranu. Može se koristiti za prijenos ili arhiviranje baze podataka isto kao i magnetska vrpca. Magnetska vrpca je jeftinija, puno sporija i ujedno s optičkim diskom rijetko se koristi.

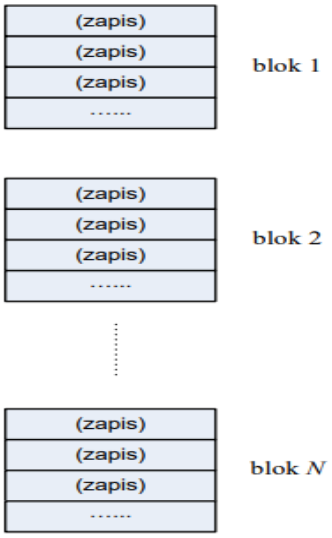
Vrijedi napomenuti da programeri baza podataka obraćaju pažnju na efekte predmemorije prilikom dizajniranja struktura podataka i algoritama za obradu upita. (Abraham Silberschatz, Henry F. Korth, S. Sudarshan, 2010.)

Fizičke karakteristike uređaja za pohranu igraju glavnu ulogu u načinu pohrane podataka, posebno zato što je pristup slučajnom dijelu podataka na disku mnogo sporiji od pristupa memoriji: Pristup disku traje desetke milisekundi, dok pristup memoriji traje desetinu mikrosekunde. (Abraham Silberschatz, Henry F. Korth, S. Sudarshan, 2010.)

7.1 Elementi fizičke baze

Važno je znati da operacijski sustav računala dijeli vanjsku memoriju u jednako velike blokove (sektore). Svaki blok jednoznačno je zadan svojom adresom. Datoteka je konačni niz zapisa (slogova) istog tipa pohranjenih u vanjskoj memoriji. Tip zapisa zadaje se kao uređena n-torka osnovnih podataka (komponenti), gdje je svaki osnovni podatak opisan svojim imenom i tipom (cijeli ili realni broj, znak, niz znakova itd.). Tipične operacije koje se obavljaju nad datotekom su: ubacivanje novog zapisa, promjena postojećeg zapisa, izbacivanje zapisa ili pronalaţenje zapisa gdje zadani osnovni podaci imaju zadane vrijednosti. (Robert Manger, 2010.).

Za primjer, promotrimo relaciju kupac koja sadrži podatke o kupcu. Kako bi ju pohranili na disk svaku njezinu n-torku pretvaramo u zapis, zatim te zapise poredamo po nekom redoslijedu. Svaki zapis je fiksne duljine, te u sebi ima vrijednosti osnovnih podataka. Kod pretvorbe relacije u datoteku nužno je uvesti fizičke detalje, na primjer duljinu nekog podatka u bajtovima, međusobne redoslijede i sl. Datoteka ne mora imati ključ kao što mora relacija, no ako je datoteka nastala na sliku relacije onda ona ima ključ koji je isti kao ključ relacije. Cijela baza je izgrađena kao skup datoteka. Kod relacijskih baza podataka, svaka relacija je prikazana jednom datotekom. Osim osnovnih datoteka, baza može imati i dodatne pomoćne datoteke poput indeksnih.



Slika 11: Datoteka sastavljena od blokova u kojima su zapisi (Robert Manger, 2010.)

Zapisi su raspoređeni po blokovima. Zapisi su manji od bloka, pa se u jedan blok sprema više zapisa. Zapis ne prelazi između blokova. Kako su cijeli zapisi spremljeni u blok i ne prelaze granicu, moguće je da dio bloka ostane neiskorišten.

7.2 Organizacija datoteka

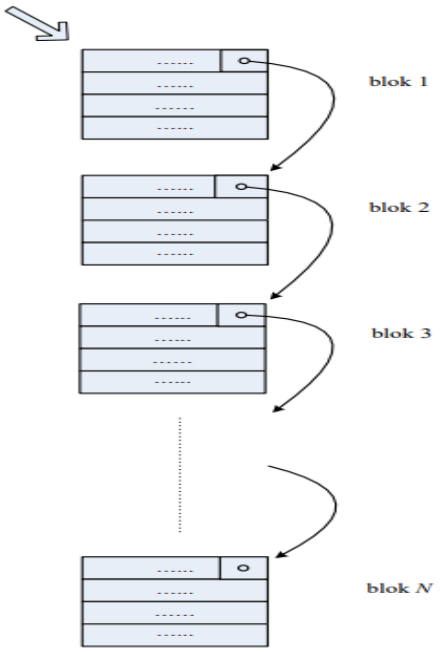
Baza podataka fizički je preslikana kao datoteka. Datoteke se nalaze u memoriji računala, tj. na diskovima. Datoteka je organizirana logički kao niz zapisa.

Datoteka sadrži zapise unutar blokova. Način na koji se ti blokovi međusobno povezuju unutar iste datoteke određen je posebnim pravilima koji se nazivaju organizacija datoteka.

Većina relacijskih baza podataka ograničava veličinu zapisa da ne bude veća od veličine bloka, radi pojednostavljenja upravljanja međuspremnikom i upravljanja slobodnim prostorom. Veliki se objekti često pohranjuju u posebnu datoteku (ili zbirku datoteka), umjesto da se pohrane s ostalim (kratkim) atributima zapisa u kojima se pojavljuju. Zatim se u zapis koji sadrži veliku pohranjuje (logički) pokazivač na objekt. (Abraham Silberschatz, Henry F. Korth, S. Sudarshan, 2010.)

Jednostavna datoteka

Kao adresu cijele datoteke pamtimo adresu prvog bloka. Prednost jednostavne organizacije je da se kod nje lagano ubacuju, izbacuju i mijenjaju zapisi. No mana je da bilo kakvo traženje, na primjer traženje zapisa sa zadanom vrijednošću primarnog ključa, zahtijeva sekvencijalno čitanje blok po blok. To znači da će jedno traženje u prosjeku zahtijevati čitanje pola datoteke pa vrijeme traženja linearno raste s veličinom datoteke. (Robert Manger, 2010.)



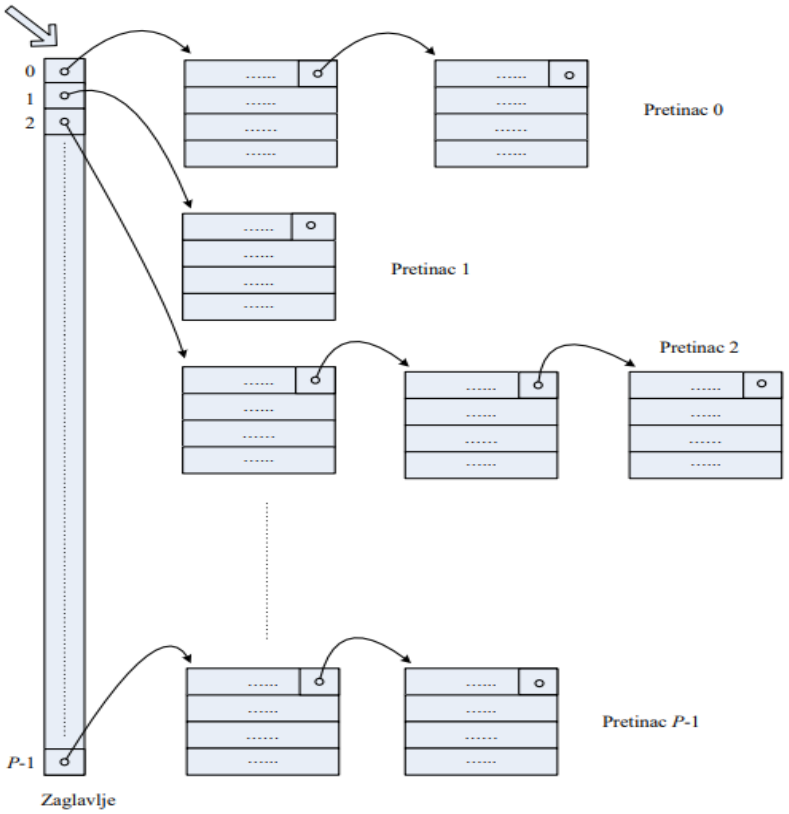
Slika 12: Građa jednostavne datoteke (Robert Manger, 2010.)

Datoteka se sadrži od blokova koji su međusobno povezani. Blokovi su poredani u listu, i svaki blok ima pokazivač na idući blok. Ovakva ogranizacija datoteke dobra je gdje nema puno pretraživanja nego se podaci samo dodaju, brišu ili je datoteka manje veličine.

Hash-datoteka

U pretincu p nalaze se zapisi datoteka koji su označeni rednim brojevima. Svaki pretinac ima jedan ili više blokova. Pokazivač na pretinac smješta se u zaglavlje prvog bloka. Adresa zaglavlja se pamti kao adresa cijele datoteke. Prednost je da na osnovi ključa imamo gotovo izravni pristup. Pomoću hash funkcije dobiva se redni broj pretinca u kojem se nalazi neki zapis s vrijednosti ključa.

Zaglavlje je obično dovoljno malo pa se za vrijeme rada s datotekom može držati u glavnoj memoriji. Hash-datoteka može se smatrati dijametralno suprotnom organizacijom od jednostavne. Nedostatak hash-datoteke je da ona ne može sačuvati sortirani redoslijed po ključu za zapise. Naime, hash-funkcija ima tendenciju „razbacivanja“ podataka na kvazi-slučajan način. Također, hash-datoteka nije pogodna ako želimo pronaći zapise gdje je vrijednost ključa u nekom intervalu. (Robert Manger, 2010.)



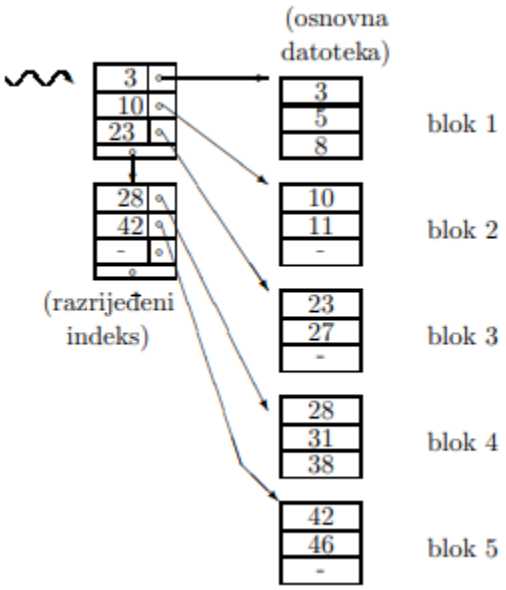
Slika 13: Građa hash datoteke (Robert Manger, 2010.)

Indeks-sekvencijalna datoteka

Indeks sekvencijalna organizacija zahtijeva da zapisi u osnovnoj datoteci budu sortirani po vrijednostima ključa (na primjer uzlazno). Blokovi ne moraju biti sasvim popunjeni. Dodajemo tzv. razrijeđeni indeks. Svaki zapis u indeksu odgovara jednom bloku osnovne datoteke i oblika je (k, a), gdje je k najmanja vrijednost ključa u dotičnom bloku, a je adresa bloka. Sam indeks je također sortiran po ključu i za sada zamišljamo da je jednostavno organiziran. (Robert Manger, 2003.)

Podaci se zapisuju jedan iza drugoga redom kojim se unose. Istovremeno se zapisuje i tablica indeksa koja se sastoji stupca s ključevima i stupca s adresama blokovima. Za pristupanje nekom bloku moramo znati samo vrijednost ključa, te bloku pristupamo direktno pomoću ključa. Tablica indeksa može sortirati blokove uzlazno ili silazno prema vrijednosti ključa.

Ako je osnovna datoteka uzlazno sortirana po ključu, tada primarni indeks može biti razrijeđen, dakle on ne mora sadržavati pokazivače na sve zapise u osnovnoj datoteci, već je dovoljno da sadrži adrese blokova i najmanju vrijednost ključa za svaki blok. Kao adresu cijele datoteke pamtimo adresu indeksa. (Robert Manger, 2010.)



Slika 14: Indeks-sekvencijalna datoteka (Robert Manger, 2003.)

Prednost je brz pristup zapisu na osnovi kjuča. Iako je nešto sporiji pristup nego hash-datoteka, čitanje datoteka sortirano po ključu izvodi se tako da slijedimo adrese blokova redoslijedom kako su upisane. Kompliciranije su operacije dodavanja, mijenjanja i brisanja podataka, zato je ona kompromis između jednostavne i hash datoteke.

7.3 Organizacija indeksa

Mnogi se upiti odnose na samo mali dio zapisa u datoteci. Indeks je struktura koja pomaže u brzom lociranju željenih zapisa relacije, bez ispitivanja svih zapisa. (Abraham Silberschatz, Henry F. Korth, S. Sudarshan, 2010.)

Neke organizacijske datoteke u sebi uključuju pomoćnu datoteku indeks. Indeks je sam za sebe posbna jedna datoteka, mora biti organizirana.

Indeksi bilo koje vrste obično se fizički prikazuju kao B-stabla. B-stablo je reda m m-narno stablo sa sljedećim svojstvima:

• korijen je ili list ili ima bar dvoje djece;

• svaki čvor izuzev korijena i listova ima između m/2 i m djece;

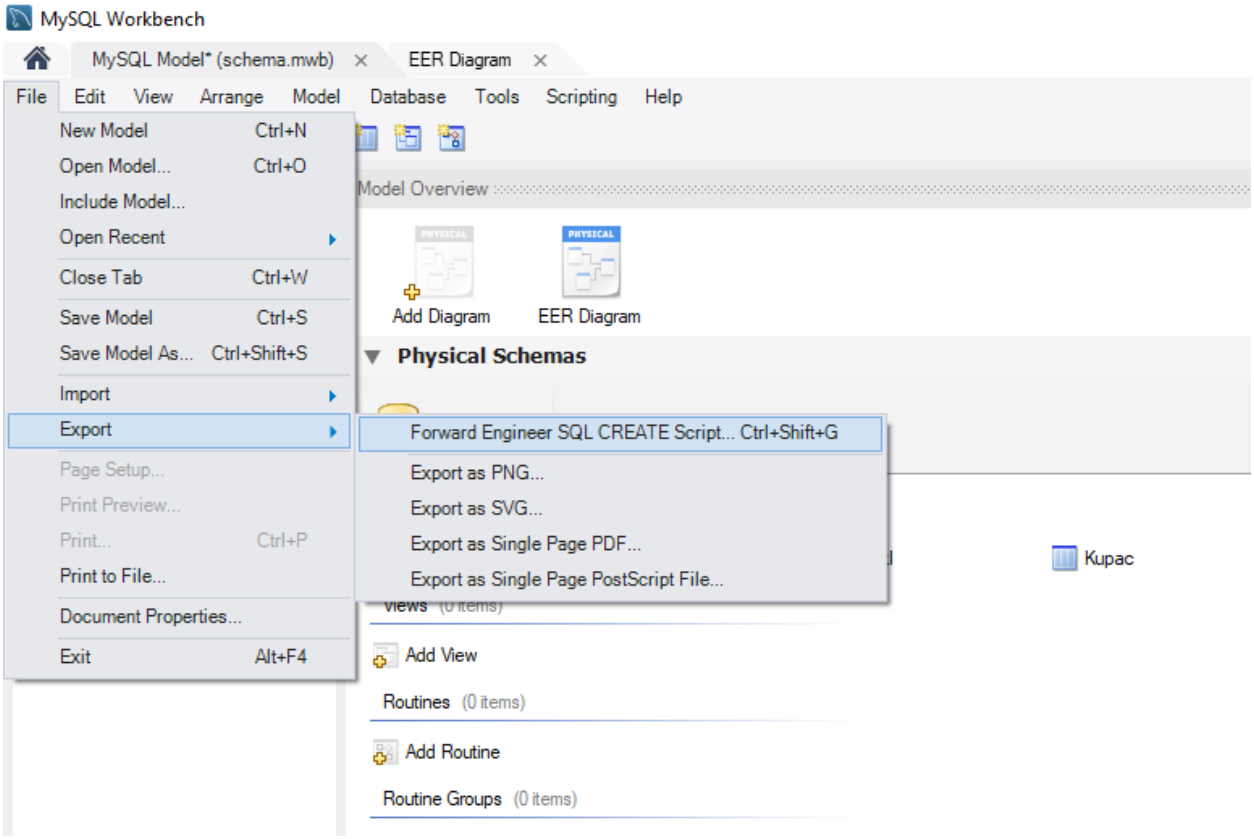
• svi putovi od korijena do lista imaju istu duljinu.

Čvorovi B-stabla zapravo su blokovi vanjske memorije. Cijelo B-stablo može se promatrati kao hijerarhija (jednostavno organiziranih) indeksa. Listovi čine "najniži" indeks s pointerima u osnovnu datoteku. Unutrašnji čvorovi neposredno iznad listova čine razrijedeni indeks najnižeg indeksa. Unutrašnji čvorovi na idućem nivou čine indeks od indeksa od indeksa, .. . , i tako dalje, . . ., korijen predstavlja "najviši" indeks. Cijela hijerarhija može se smatrati i jednim (hijerarhijski gradenim) indeksom. (Robert Manger, 2008.)

8. Primjer web aplikacije s MySQL bazom podataka

Izraditi ću jednostavnu web aplikaciju koja koristi MySQL bazu podataka. Kroz nju ću prikazati primjere korištenja baze podataka, te izvršavanje operacija nad bazom preko web aplikacije. Koristiti ću MySQL, HTML jezik za izradu web stranice, PHP skriptni jezik za povezivanje sa bazom podataka i izvršavanje operacija.

Prvo ER dijagram eksportiramo u fizičku bazu podataka. Pomoću alata MySQL Workspace dijagram se exportira u bazu podataka, te nastaje jedna datoteka sa .sql ekstenzijom. Datoteka je fizička shema baze koja sadrži sve funckije, operacije, veze, entitete i drugo za kreiranje baze podata. Datoteka se izradi putem file>Export>Forward engineer SQL CREATE script… prikazano na slici 15.



Slika 15: Kreiranje fizičke baze podataka

Sada kada imamo .sql datoteku baze podataka nju možemo koristiti za udomljavanje baze. Učitavajući .sql datoteku tablice s atributima automatski se kreiraju.

Početna stranica aplikacije sastoji se od 3 veze koje vode na crud operacije. Prva veza vodi na create gdje se unose podaci u bazu pomoću forme. Druga veza vodi na read gdje su prikazane 3 tablice iz baze s podacima sortirano u tabularnom prikazu. Treća veza vodi na update i delete operacije gdje se pomoću formi mogu mijenjati ili brisati odabrane tablice, tj. njihovi podaci. Crud operacije detaljno su prikazane i opisane u poglavlju 8.2.

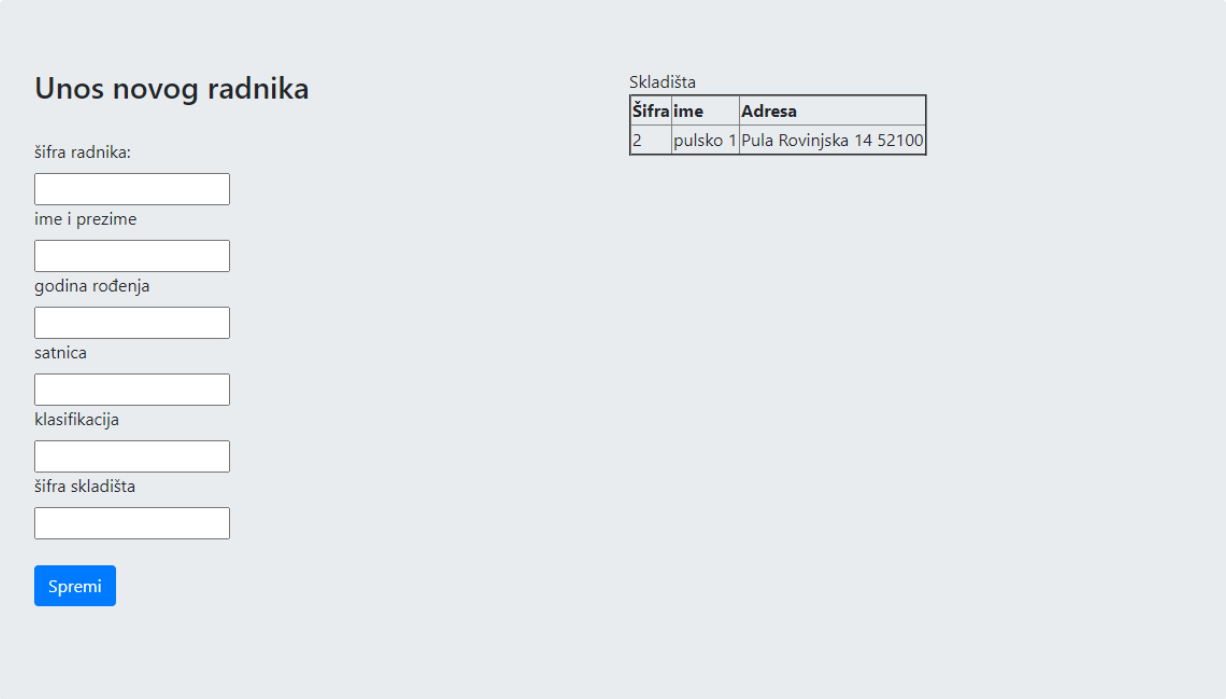
8.1 PHP i MySQL

PHP je jedan od najpopularnijih server-side jezika za razvoj web aplikacija. Besplatan je, open source i server-side jezik što znači da se kod izvršava na server. Pomoću njega kreiramo HTML stranicu na server te je zatim šaljemo klijentu s dinamičkim sadržajem. Tako klijent ne može vidjeti izvorni kod koji je generirao sadržaj nego samo HTML kod. Jedna od najvećih prednosti PHP-a kao serverskog skriptnog jezika je ta što je moguće na vrlo jednostavan način koristiti velik broj bazi podataka. MySQL je najpopularnija baza podataka koja se koristi u kombinaciji s PHP-om jer su oboje besplatni i open-source. Danas, više od 80% svih web stranica koristi PHP čiji programski jezik na strani poslužitelja. Neke od najpopularnijih web destinacija koje koriste PHP u kombinaciji s MySQL su Facebook, Wordpress, Yahoo, Flickr, Wikipedia. Kombinacija PHP i MySQL daje bezbroj mogučnosti za stvaranje raznih web aplikacija. Aplikacije koje su izgrađeni na ovim sustavima nose karakteristike dinamičnosti, skalabilnosti, visoke sigurnosti, isplativost, te popularnost. PHP i MySQL nudi web izvedbe s visokim performansama i skalabilnošću uz veliku isplativost jer su oboje otvorenog koda.

8.2 CRUD operacije

CREATE

Za prikaz primjera create izrađena je stranica za unos novog radnika u bazu. Sastoji se od forme u koju se unesi podaci te nakon slanja forme ti podaci se upišu u bazu podataka.



Slika 16: Unos podataka u bazu preko php aplikacije

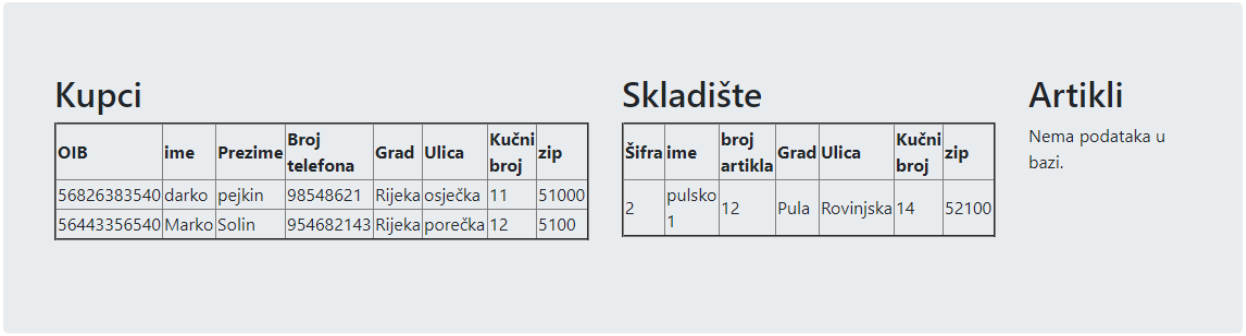
Backend aplikacije funkcionira tako da se prvo povežemo s bazom, a to činimo pozivanjem datoteke testmysql.php. Html forma zaprima podatke i klikom na gumb spremi podaci se šalju preko php koda u bazu podataka. Sa desne strane stoji ispis skladišta iz baze. U formi je potrebno navesti šifru skladišta u kojoj radnik radi. To polje je atribut koji je strani ključ naziva šifra_skladišta u tablici radnik.

```php
<?php

if(isset($_POST['save']))
{
    $sql = "INSERT INTO radnik (šifra_radnika, ime_i_prezime, god_rođenja, satnica, klasifikacija, skladište_šifra_skladišta)
    VALUES ('".$_POST["šifra_radnika"]."','".$_POST["ime_i_prezime"]."','".$_POST["god_rođenja"]."','".$_POST["satnica"]."',
    '".$_POST["klasifikacija"]."','".$_POST["skladište_šifra_skladišta"]."')";

    $result = mysqli_query($conn,$sql);

}


?>
```

Slika 17: PHP kod za spremanje podataka u bazu

READ

Za primjer read funkcije izrađena je stranica koja u 3 redka prikazuje tablice kupci, skladište i artikli.



Kupci

OIB	ime	Prezime	Broj telefona	Grad	Ulica	Kučni broj	zip
56826383540	darko	pejkin	98548621	Rijeka	osječka	11	51000
56443356540	Marko	Solin	954682143	Rijeka	porečka	12	5100

Skladište

Šifra	ime	broj artikla	Grad	Ulica	Kučni broj	zip
2	pulsko 1	12	Pula	Rovinjska	14	52100

Artikli

Nema podataka u bazi.

Slika 18: Prikaz podataka iz baze

Kod za dohvat podataka sadrži select upit prema bazi, te generira tablicu s vračenim podacima iz baze. Kako tablice kupci i skladište imaju relaciju adresa kao zasebnu tablicu, kako bi se prikazala adresa uz svakog prikapadajućeg kupca, tj skladište, korišten je inner join.

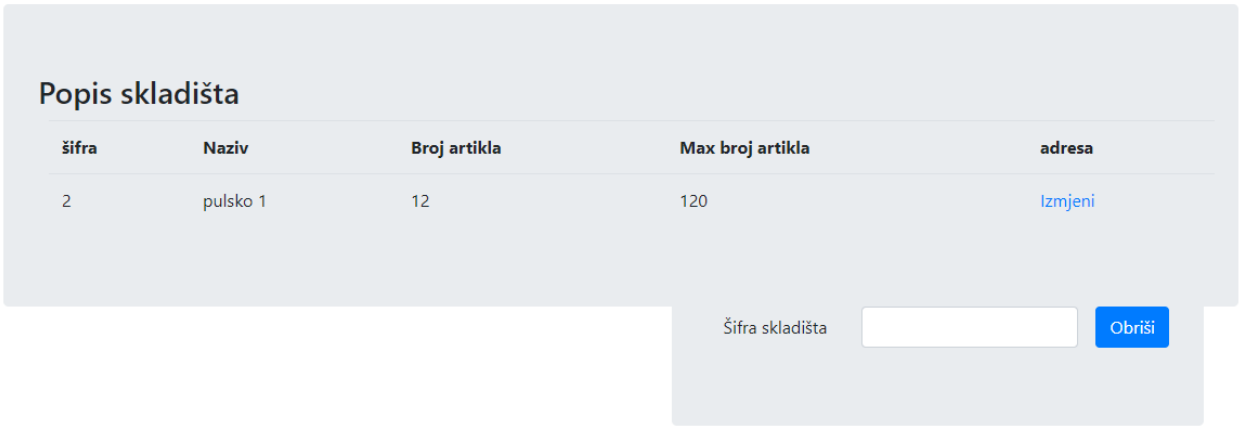
```php
$sql = "SELECT * FROM kupac INNER JOIN adresa ON adresa.šifra_adrese=kupac.adresa_šifra_adrese";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
  // output data of each row
  echo "<table border='2'>
<tr>
<th>OIB</th>
<th>ime</th>
<th>Prezime</th>
<th>Broj telefona</th>
<th>Grad</th>
<th>Ulica</th>
<th>Kučni broj</th>
<th>zip</th>
</tr>";
  while($row = mysqli_fetch_assoc($result)) {
    echo "<tr>";
echo "<td>" . $row['oib'] . "</td>";
echo "<td>" . $row['ime'] . "</td>";
echo "<td>" . $row['prezime'] . "</td>";
echo "<td>" . $row['broj_telefona'] . "</td>";
echo "<td>" . $row['grad'] . "</td>";
echo "<td>" . $row['ulica'] . "</td>";
echo "<td>" . $row['kučni_broj'] . "</td>";
echo "<td>" . $row['zip'] . "</td>";
echo "</tr>";
  }
  echo "</table>";
} else {
  echo "Nema podataka u bazi.";
}
```

Slika 19: Kod za dohvat tablice kupac

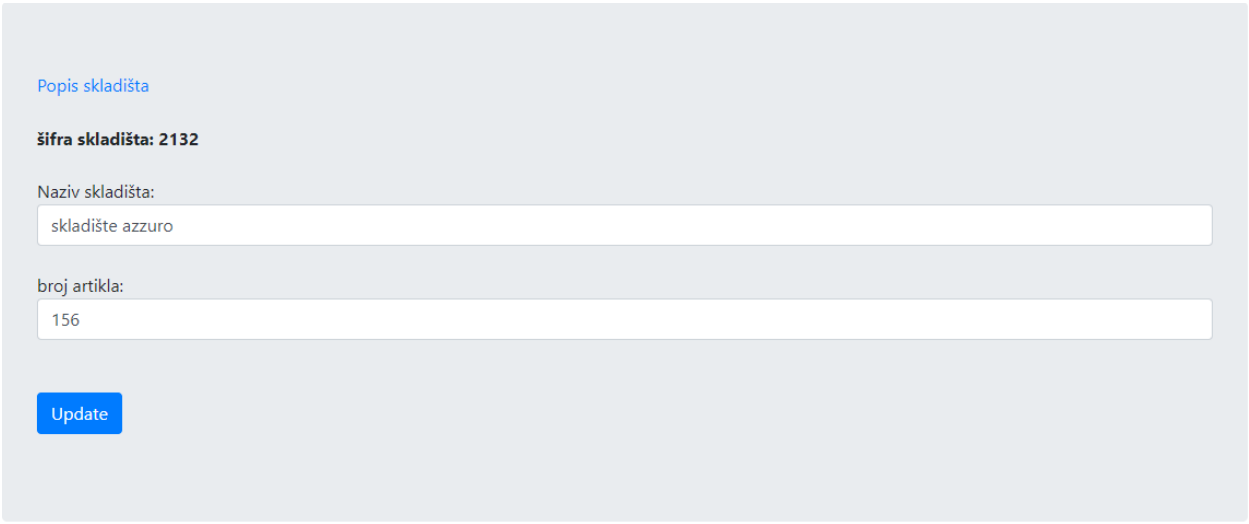
UPDATE

Za primjer update baze izrađena je stranica gdje se prvo koristi read za ispis tablice skladišta prikazano kaon a slici 16. Klikom na gumb izmjeni odabrana je tablica čiji se podaci žele izmjeniti te se prebacuje na stranicu s formom za izmjenu podataka prikazano kao na slici 17.

Kod za izmjenu podataka je ustvari samo sql update gdje uzima podatke iz forme na slici 17 i određene vrijednosti u tablici zamjenjuje sa njima. Kod je prikazan na slici 18.



Slika 20: Stranica za update i brisanje podataka



Slika 21: Izmjena podataka, update nad podacima u bazi

```php
<?php
include_once 'testmysql.php';
if(count($_POST)>0) {
mysqli_query($conn,"UPDATE skladište set naziv_skladišta='" . $_POST['naziv_skladišta'] . "',
broj_artikla='" . $_POST['broj_artikla'] . "' WHERE šifra_skladišta='" . $_GET['šifra_skladišta'] . "'");
$message = "Baza updateana !";
}
$result = mysqli_query($conn,"SELECT * FROM skladište WHERE šifra_skladišta='" . $_GET['šifra_skladišta'] . "'");
$row= mysqli_fetch_array($result);
?>
```

Slika 22: Kod za update podataka

DELETE

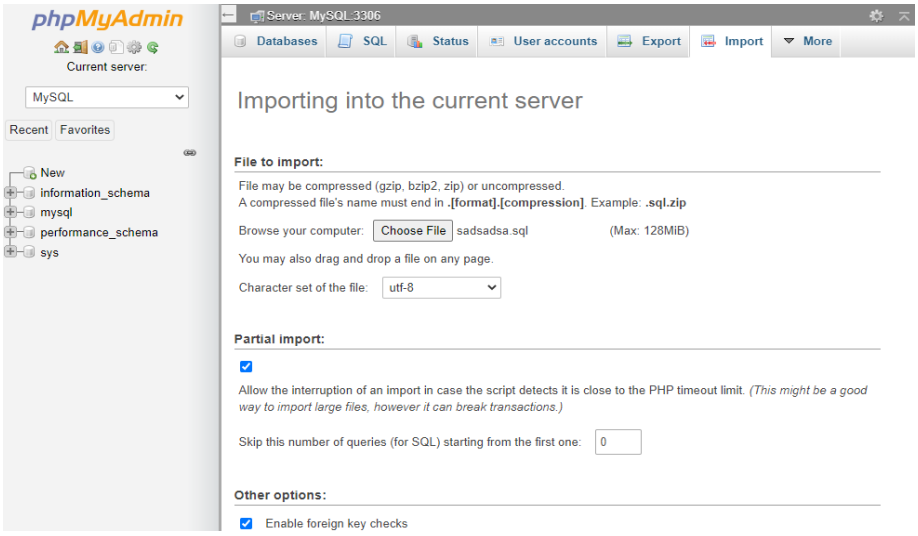
Brisanje tablica u bazi vrši se preko iste stranice kao za update podataka. Pogledajmo ponovno sliku 16. Pri dnu stranice nalazi se polje za unos šifre skladišta, koju možemo vidjeti na istoj stranici, te gumb za brisanje. Nakon pritiska na gumb, cijela tablica s tom šifrom skladišta biti će potpuno obrisana. Kod za brisanje tablice prikazan je na slici 19 ispod. Pritiskom na gumb obriši pokreće se sql naredba za brisanje tablice sa šifrom skladišta koja je unesena u polje.

```
if(count($_POST)>0) {
mysqli_query($conn,"DELETE FROM skladište WHERE
šifra_skladišta='" . $_POST['šifra_skladišta'] . "'");
}
```

Slika 23: Kod za brisanje podataka

8.3 Udomljavanje aplikacije

Za udomljavanje aplikacije koristio sam program wamp server na platformi windows. Po osobnom mišljenju jednostavniji je za korištenje i udomljavanje MySQL baze podataka i PHP programskih aplikacija. U paketu dolazi s instaliranim phpMyAdmin koji je besplatni alat za upravljanje MySQL i MariaDB. Omogučava izravno povezivanje s bazom, importanje baze, sve CRUD operacije dostupne su u par klika mišom. Kako bi baza podataka bila dostupna za povezivanje s aplikacija, tj. udomljena, potrebno je otvoriti phpMyAdmin te importirati .sql datoteku baze. Tada je baza učitana i sve tablice postavljene i spremne za unos podataka.



Slika 24: phpMyAdmin učitavanje baze podataka

Ostali dio aplikacije čini php koji je udomljen putem apache poslužitelja. Kako bi aplikacija funkcionirala i mogla čitatii pisati podatke u bazu potrebno ih je spojiti. Kreira se posebna datoteka naziva tipa connect.php koja se kasnije poziva gdje je potrebno slanje upita prema bazi. Datoteka s kodom izgleda kao na slici ispod, a u sebi sadrži podatke za spajanje poput korisničkog imena, lozinke, ime baze i adresu poslužitelja. Kako su svi poslužitelji pokrenuti lokalno, adresa za spajanje na bazu je localhost, port ostaje defaultni, korisničko ime je odmah zadano na root bez lozinke.

testmysql.php

```php
<?php

$user = 'root'; // Korisničko ime za konekciju, defaultno uvijek root
$password = ''; // Lozinka nije postavljena
$database = 'mydb'; // Ime baze, u ovom slučaju mydb
$port = NULL; // Postavljamo NULL kako bi se koristio defaultni port
$mysqli = new mysqli('localhost', $user, $password, $database, $port); // Povezivanje na lokalnu adresu

if ($mysqli->connect_error) {
    die('Connect Error (' . $mysqli->connect_errno . ') '
            . $mysqli->connect_error);
}
echo '<p>Konekcija OK <br>'. $mysqli->host_info.'</p>';
echo '<p>Server '.$mysqli->server_info.'</p>';
echo '<p>charset: '.$mysqli->character_set_name().'</p>';

$mysqli->close();
?>
```

Slika 25 testmysql.php datoteka za povezivanje s bazom

MySQL baza podataka i aplikacija ne moraju biti udomljeni na istoj adresi. Baza može biti udomljena na nekom drugom mjestu, ali s podacima za pristup i poznavajući adresu mjesta gdje je udomljena moguće je pristupiti bazu i izvršavati operacije.

9. Zaključak

MySQL je sustav za upravljanje relacijskim bazama podataka baziranim na sql koji postoji već 25 godina. Karakterizira ga skalabilnost, sigurnost i brzina. Zbog otvorenog koda, jednostavnosti i dostupnosti i dalje je najpopularniji sustav. Najčešće korišten za spremanje podataka kod web aplikacija, poput Drupal i Wordpress. Dostupne su aplikacije za modeliranje i upravljanje bazom, koje su besplatne i jednostavne za korištenje, sa grafičkim sučeljem poput službenog alata MySQL workspace, phpmyadmin, dbedit i ostali. Za izradu ovog rada koristio sam MySQL workspace za izradu relacijske scheme te izradu fizičke bazu. Relacijska shema predstavlja upravljanjem skladištem koji zaprima i otprema narudžbe. Sustav se može proširiti i nadograditi. MySQL je dobar izbor jer se lako kombinira sa php jezikom. Podaci su uvijek dostupni i zaštićeni. Za sam pristup bazi potrebno je znati ime baze, korisničko ime, lozinku i adresu poslužitelja gdje je baza udomljena. Također su stavke poput lozinka koje se čuvaju u bazi zaštićene posebnim algoritmom. MySQL verzija 8 donosi puno novih značajki ali i znatno brže izvođenje operacijama nad bazom podataka. Što se više operacija izvodi nad bazom, to verzija 8 pokazuje bolje rezultate od starije verzije 5.7. Kako se nalazi pod vodstvom Oracle kompanije sustav je otvorenog koda pod GPL licencom. Korištenje MySQL-a preporučase za sustave gdje su sigurnost i visoke performanse od velike važnosti.

Za izradu primjera aplikacije koja koristi MySQL bazu podataka koristio sam wamp paket servisa. Wamp dolazi s MySQL, phpMyAdmin i apache servisima te mnogim drugima koje nisam koristio. Namijenjen je windows platformi. Mnogi linux operacijski sustavi dolaze s već predinstaliranom verzijom MySQL. Aplikacija je pokrenuta lokalno, apache je koristio za pokretanje php skripti, MySQL za čuvanje podataka u bazi, te phpMyAdmin za administriranje i brzi pregled podataka u bazi.

10. Literatura

1. Brad Bulger, Jay Greenspan, David Wall, 2004.. *MySQL/PHP Database Applications [2nd ed]*, Wiley Pub

2. Charles Bell, 2018.. *Introducing the MySQL 8 Document Store*, Apress

3. Adrian W. West, Steve Prettyman, 2018.. *Practical PHP 7, MySQL 8, and MariaDB Website Databases*, Apress

4. Abraham Silberschatz, Henry F. Korth, S. Sudarshan, 2010.. *Database System Concepts, 6th Edition*, McGraw-Hill

5. Shabbir Challawala, Jaydip Lakhatariya, Chintan Mehta, Kandarp Patel, 2017.. *MySQL 8 for Big Data*, Packt Publishing

6. Eric Vanier, Birju Shah, Tejaswi Malepati, 2019.. *Advanced MySQL 8: Discover the full potential of MySQL and ensure high performance of your database*, Packt Publishing

7. Robert Manger, 2010.. *Osnove projektiranja baza podataka* [Mrežno]
   Link: https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/d310_polaznik.pdf
   [Pokušaj pristupa 2 Rujan 2020.].

8. Koraljka B. Damir M., n.d., *UTJECAJ DENORMALIZACIJE BAZE PODATAKA NA WEB APLIKACIJE* [Mrežno]
   Link: https://bib.irb.hr/datoteka/500408.CASE_10_-_Utjecaj_denormalizacije_baza.pdf
   [Pokušaj pristupa 2 Rujan 2020.].

9. Robert Manger, 2008.. *BAZE PODATAKA Dodaci uz skripta* [Mrežno]
   Link: http://web.studenti.math.pmf.unizg.hr/~manger/bp/dodaci.pdf
   [Pokušaj pristupa 2 Rujan 2020.].

10. Robert Manger, 2003.. *BAZE PODATAKA skripta* [Mrežno]
    Link: http://jadran.izor.hr/~dadic/EKO/baze-podataka.pdf
    [Pokušaj pristupa 2 Rujan 2020.].

11. Dries Buytaert, 2010.. *The history of MySQL AB* [Mrežno]
    Link: https://dri.es/the-history-of-mysql-ab
    [Pokušaj pristupa 2 Rujan 2020.].
12. Anon., 2020.. *MySQL 8.0 reference Manual* [Mrežno]
    Link: https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html
    [Pokušaj pristupa 2 Rujan 2020.].

13. Leah Beatty, 2013.. *The Pros and Cons of MySQL* [Mrežno]
Link: https://www.smartfile.com/blog/the-pros-and-cons-of-mysql/
[Pokušaj pristupa 2 Rujan 2020.].

14. Edward S., 2019.. *What to Use – SQLite or MySQL* [Mrežno]
Link: https://www.hostinger.com/tutorials/sqlite-vs-mysql-whats-the-difference/
[Pokušaj pristupa 2 Rujan 2020.].

15. Krasimir Hristozov, 2019.. *MySQL vs PostgreSQL – Choose the right database for your project* [Mrežno]
Link: https://developer.okta.com/blog/2019/07/19/mysql-vs-postgres
[Pokušaj pristupa 2 Rujan 2020.].

16. Ostezer, Mark Drake, 2019, *SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems* [Mrežno]
Link: https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-systems
[Pokušaj pristupa 2 Rujan 2020.].

17. Geir Hoydalsvik, 2018.. *What's New in MySQL 8.0* [Mrežno]
Link: https://www.mysql.com/why-mysql/white-papers/whats-new-mysql-8-0/
[Pokušaj pristupa 7 Rujan 2020.].

18. Anon., 2013.. *Relational Model* [Mrežno]
Link: http://wiki.c2.com/?RelationalModel
[Pokušaj pristupa 2 Rujan 2020.].

19. Anon., n.d. *Relational Data Model in DBMS: Concepts, Constraints, Example* [Mrežno]
Link: https://www.guru99.com/relational-data-model-dbms.html
[Pokušaj pristupa 2 Rujan 2020.].

20. Paul Namuag, 2019.. *MySQL Performance Benchmarking: MySQL 5.7 vs MySQL 8.0* [Mrežno]
Link: https://severalnines.com/database-blog/mysql-performance-benchmarking-mysql-57-vs-mysql-80
[Pokušaj pristupa 6 Rujan 2020.].

21. Emmanuelle Rieuf, 2016.. *History of MySQL* [Mrežno]
Link: https://www.datasciencecentral.com/profiles/blogs/history-of-mysql
[Pokušaj pristupa 6 Rujan 2020.].