**Šolski center Novo mesto**

**Srednja elektro šola in tehniška gimnazija**

**Šegova ulica 112**

**8000 Novo mesto**

Aplikacije in informacijski sistemi :

Izdelava podatkovne baze za majhen obrat

**Predmet: Računalništvo**

**Avtor: Tomaž Gril**

**Razred: T4c**

**Mentor: dr. Albert Zorko**

**Novo mesto, april 2022**

Povzetek

Podatkovna baza je zbirka podatkov, v kateri so podatki povezani, urejeni in so na voljo večjemu številu uporabnikov. V PB izvajamo opravila nad podatki kot so zbiranje, vnos v sistem, hranjenje, spreminjanje, posredovanje, varovanje pred izgubo in zaščita podatkov. Postopek gradnje podatkovne baze zavzema analizo zahtev uporabnikov, konceptualno načrtovanje, izbiro SUPB-ja, logično načrtovanje, fizično načrtovanje in implementacijo v informacijski sistem. Pri konceptualnem načrtovanju uporabljamo ER model, pri logičnem načrtovanju pa relacijski model. Normalizacija je proces učinkovite organizacije podatkov v bazi podatkov, pri kateri odstranimo odvečne podatke in zagotovimo smiselno odvisnost podatkov. SUPB je programska oprema za delo, kreiranje in vzdrževanje podatkovne baze. MySQL je sistem za upravljanje relacijske podatkovne baze. MySQL Server ponuja sistem za upravljanje baz podatkov, medtem ko je MySQL Workbench grafično orodje za delo s strežniki in bazami podatkov MySQL. Za delo s podatki v relacijski zbirkah se uporablja SQL. MySQL se od standardnega SQL-a razlikuje v nekaj razširitvah in omejitvah. Podatkovne baze lahko shranjujemo v lokalnem omrežju na fizičnih napravah, če pa je podatkovna baza na spletu oziroma drugi lokaciji, lahko z dostopom do interneta izvajamo oddaljen dostop do podatkovne baze. Za povezovanje zbirk podatkov z Javo se uporabljajo razni gonilniki, ki jih v programu potem kličemo kot razrede in vzpostavljamo povezave. Pri izdelavi aplikacij za delov s PB v Javi se med drugim uporablja knjižnica grafičnih komponent Swing, ki omogoča delo s različnimi grafičnimi komponentami.

Ključne besede:

* Java
* Knjižnica Swing
* SQL
* MySQL Sever
* MySQL Workbench
* JDBC gonilnik
* Podatkovne baze
* SUPB
* Načrtovanje podatkovnih baz
* Normalizacija
* ER model
* Relacijski model

Kazalo vsebine

[1 Uvod 5](#_Toc100741392)

[2 Izdelava podatkovne baze za majhen obrat 6](#_Toc100741393)

[2.1 Predstavitev uporabljenih teoretičnih osnov 6](#_Toc100741394)

[2.1.1 Predstavitev uporabljenih jezikov 6](#_Toc100741395)

[2.1.1.1 Java 6](#_Toc100741396)

[2.1.1.2 SQL in MySQL 7](#_Toc100741397)

[2.1.2 Predstavitev uporabljene programske opreme 7](#_Toc100741398)

[2.1.2.1 Eclipse IDE za Java razvijalce 7](#_Toc100741399)

[2.1.2.2 MySQL WorkBench in Server 8](#_Toc100741400)

[2.1.3 Podatkovne baze 10](#_Toc100741401)

[2.1.4 Grajenje podatkovnih baz 11](#_Toc100741402)

[2.1.4.1 Konceptualno načrtovanje podatkovnih baz 12](#_Toc100741403)

[2.1.4.2 Izbira SUPB 13](#_Toc100741404)

[2.1.4.3 Logično načrtovanje podatkovne baze 14](#_Toc100741405)

[2.1.4.4 Normalizacija 16](#_Toc100741406)

[2.1.4.5 Dokumentacija 17](#_Toc100741407)

[2.1.4.6 Fizično načrtovanje 17](#_Toc100741408)

[2.1.5 Povezava podatkovnih baz v omrežje 18](#_Toc100741409)

[2.1.6 Podatkovne baze in Java 20](#_Toc100741410)

[2.1.7 Izdelava uporabniških aplikacij v Javi 22](#_Toc100741411)

[2.2 Izdelava podatkovne baze in aplikacije 24](#_Toc100741412)

[2.2.1 Dokumentacije izdelave baze 24](#_Toc100741413)

[2.2.1.1 Izdelava ER in relacijskega modela 24](#_Toc100741414)

[2.2.1.2 Grajenje baze v orodju MySQL 29](#_Toc100741415)

[2.2.2 Dokumentacija izdelave programa 31](#_Toc100741416)

[2.2.2.1 Povezovanje baze z Javo 31](#_Toc100741417)

[2.2.2.2 Izdelava uporabniškega vmesnika v Javi 33](#_Toc100741418)

[3 Zaključek 48](#_Toc100741419)

[4 Zahvala 49](#_Toc100741420)

[5 Bibliografija 50](#_Toc100741421)

[6 Stvarno kazalo 53](#_Toc100741422)

[7 Priloge 54](#_Toc100741423)

Kazalo slik

[Slika 1- Vpogled v program Eclipse 8](file:///C:\Users\Tomaž%20Gril\Documents\ŠOLA\Računalništvo\Maturitetna%20seminarska\Aplikacije%20in%20informacijski%20sistemi%20-%20Izdelava%20uporabniku%20prijazne%20podatkovne%20baze%20za%20majhen%20obrat.docx#_Toc100741424)

[Slika 2 - Vpogled v program MySQL Workbench 9](file:///C:\Users\Tomaž%20Gril\Documents\ŠOLA\Računalništvo\Maturitetna%20seminarska\Aplikacije%20in%20informacijski%20sistemi%20-%20Izdelava%20uporabniku%20prijazne%20podatkovne%20baze%20za%20majhen%20obrat.docx#_Toc100741425)

[Slika 3 - Primer zapisa entite in nabora entitet 12](file:///C:\Users\Tomaž%20Gril\Documents\ŠOLA\Računalništvo\Maturitetna%20seminarska\Aplikacije%20in%20informacijski%20sistemi%20-%20Izdelava%20uporabniku%20prijazne%20podatkovne%20baze%20za%20majhen%20obrat.docx#_Toc100741426)

[Slika 4 - Primer zapisa atributa in atributa, ki je tudi ključ 12](file:///C:\Users\Tomaž%20Gril\Documents\ŠOLA\Računalništvo\Maturitetna%20seminarska\Aplikacije%20in%20informacijski%20sistemi%20-%20Izdelava%20uporabniku%20prijazne%20podatkovne%20baze%20za%20majhen%20obrat.docx#_Toc100741427)

[Slika 5 - Primer povezave entitet z kardinalnostjo ena proti več 13](#_Toc100741428)

[Slika 6 - Primer pretvorbe iz ER modela v relacijski model 14](file:///C:\Users\Tomaž%20Gril\Documents\ŠOLA\Računalništvo\Maturitetna%20seminarska\Aplikacije%20in%20informacijski%20sistemi%20-%20Izdelava%20uporabniku%20prijazne%20podatkovne%20baze%20za%20majhen%20obrat.docx#_Toc100741429)

[Slika 7 Skica ER modela za obrat Okna 25](#_Toc100741430)

[Slika 8 ER diagram za obrat Okna 28](#_Toc100741431)

[Slika 9 Prikaz tabel za obrat Okna v MySQL 29](#_Toc100741432)

[Slika 10 - Spletna stran MySQL z Connectorji 31](#_Toc100741433)

[Slika 11 - Datoteka s konektorjem 32](#_Toc100741434)

[Slika 12 - Dodajanje konektorja v knjižnico modulov 32](#_Toc100741435)

[Slika 13 - Nastavitev novega uporabnika v MySQL Workbench 35](file:///C:\Users\Tomaž%20Gril\Documents\ŠOLA\Računalništvo\Maturitetna%20seminarska\Aplikacije%20in%20informacijski%20sistemi%20-%20Izdelava%20uporabniku%20prijazne%20podatkovne%20baze%20za%20majhen%20obrat.docx#_Toc100741436)

[Slika 14 - Glavni meni aplikacije 37](#_Toc100741437)

[Slika 15 - Panela za obdelavo podatkov v bazi 38](#_Toc100741438)

[Slika 16 - Panela za iskanje podatkov v PB 38](#_Toc100741439)

[Slika 17 - Paneli za kreiranje in brisanje tabel 39](#_Toc100741440)

[Slika 18 – Primer delovanja seznama tabel za izris obrazca in prikaza izbrane tabele iz PB 45](#_Toc100741441)

[Slika 19 - Dodajanje vrstic atributov pri kreiranju tabel 46](#_Toc100741442)

[Slika 20 - Brisanje vrstic atributov pri kreiranju tabel 47](#_Toc100741443)

Kazalo odsekov kode

[Koda 1 - Povezava javanskega programa s podatkovno bazo MySQL server 21](#_Toc100741444)

[Koda 2 - Primer izvrševanja DDL in DML ukazov v Javi 21](#_Toc100741445)

[Koda 3 - Primer izvrševanja DQL ukazov v javi 21](#_Toc100741446)

[Koda 4 - Primer uporabe razreda Swing 23](#_Toc100741447)

[Koda 5 - Primer akcijskega poslušalca za gumb 23](#_Toc100741448)

[Koda 6 - Primer dodajana tabel v PB 29](#_Toc100741449)

[Koda 7 - Primer vstavljanja podatkov v tabelo 30](#_Toc100741450)

[Koda 8 – Skrajšana predstavitev glavnega razreda programa 33](#_Toc100741451)

[Koda 9 - Prestavitev glavne metode 34](#_Toc100741452)

[Koda 10 - Metode glavnega razreda za klasične SQL ukaze 36](#_Toc100741453)

[Koda 11 - Metode glavnega razreda za podatke o zgradbi tabel 36](#_Toc100741454)

[Koda 12 - Metode glavnega razreda za zagotavljanje varnosti 36](#_Toc100741455)

[Koda 13 - Uporabljen način prehoda med panelami 40](#_Toc100741456)

[Koda 14 – Primer delovanja gumba za vnos podatkov v tabelo 41](#_Toc100741457)

[Koda 15 – Primer izvršitev SQL operacije 42](#_Toc100741458)

[Koda 16 – Metoda za izris obrazca na panelo 43](#_Toc100741459)

[Koda 17 – Metoda za izris tabele na panelo 44](#_Toc100741460)

Kazalo tabel

[Tabela 1 - Primer relacijske tabele s ključem ID 15](#_Toc100741461)

[Tabela 2 - Nenormalizirana tabela 16](#_Toc100741462)

[Tabela 3 - Normalizirana tabela (razdeljena na dve tabeli) 17](#_Toc100741463)

[Tabela 4 - Vzpostavitev povezave s PB v MySQL Workbench (vnesene so nastavitve za povezavo v lokalno bazo podatkov) 19](file:///C:\Users\Tomaž%20Gril\Documents\ŠOLA\Računalništvo\Maturitetna%20seminarska\Aplikacije%20in%20informacijski%20sistemi%20-%20Izdelava%20uporabniku%20prijazne%20podatkovne%20baze%20za%20majhen%20obrat.docx#_Toc100741464)

[Tabela 5 - Tabela s entitetami in primarnimi ključi 25](#_Toc100741465)

[Tabela 6 - Tabela atributov po entitetah 26](#_Toc100741466)

[Tabela 7 - Relacijska shema PB 27](#_Toc100741467)

# Uvod

V tej seminarski nalogi sem se odločil izdelati podatkovno bazo za majhen obrat v podjetju in aplikacijo za končne uporabnike te podatkovne baze. Za model obrata sem izbral proizvodnjo izdelave oken na podlagi osebnih delovnih izkušenj v takšnem okolju.

V teoretičnem delu bom v glavnini predstavil, kaj so podatkovne baze in kakšne so potrebne faze pri načrtovanju poljubne podatkovne baze. Poleg lastnosti samih podatkovnih baz bom raziskal še kako jih lahko povežemo in implementiramo v javanskih programih. Na kratko bom opisal tudi uporabljene jezike, programsko opremo in načine izdelave grafičnih aplikacij v Javi.

V praktičnem delu bom dokumentiral načrtovanje podatkovne baze in nato implementacije te s pomočjo programske opreme MySQL Server in MySQL Workbench. Pri implementaciji bom samo opisal postopek izdelave tabel in vnosa testnih podatkov v MySQL bazo, celotna SQL koda pa bo na voljo v prilogi. V drugem delu praktičnega dela bom dokumentiral izdelavo aplikacije za delo s podatkovno bazo. Predstavil bom konceptno zgradbo programa in s izseki kode predstavil pomembnejše komponente. V dokumentacijo program bom dodal tudi opis izgleda aplikacije in kakšne funkcije ponuja uporabniku.

Naš končni cilj je pridobiti izkušnje pri izdelavi podatkovnih baz in aplikacij za uporabnike, ter ugotoviti možne izboljšave.

# Izdelava podatkovne baze za majhen obrat

## Predstavitev uporabljenih teoretičnih osnov

### Predstavitev uporabljenih jezikov

#### Java

Java je programski jezik, ki so ga prvič predstavili maja 1995. Z različico 1.2 je pridobila razvojno okolje JDK (Java Development Kit). Prednosti Jave so sposobnosti, da njeni programi delujejo na vseh operacijskih sistemih, da lahko z njo izdelamo medmrežne internetne programe in je varen jezik[[1]](#footnote-1). Vendar pa za delovanje potrebuje prevajalnik, da se prevede v izvršno kodo. (1)

»Javo kot podlago sestavljajo programski jezik, programske knjižnice in izvajalno okolje. Java je bila namenjena za potrošniško elektroniko, naprave ki bi imele majhno zmogljivost in bi si bile lahko zelo različne. Javino izvajalno orodje JVM(Java Virtual Machine) omogoča izvajanje programske kode na različnih strojnih podlagah. Danes se Java ne uporablja več toliko v obliki programčkov za spletne strani in se več uporablja kot povezovalnik člen v več nivojskih programih, v strežnikih in prenosnih napravah. Zelo pomembna so tudi njena razvojna orodja, ki omogočajo delo tudi z drugimi programskimi jeziki (Eclipse).« (2)

»JDK ponuja osnovna orodja programiranja v Javi. Poleg JDK so še druga razvojna orodja z večjo univerzalnostjo in so večkrat tudi plačljivi. JDK se lahko brezplačno prenese na Javini spletni strani. Pri namestitvi orodji je potrebno imeti v mislih da je JRE[[2]](#footnote-2) namenjen le uporabi programov v Javi in da JDK že vsebuje JRE. Velik del JDK je razvit v javi, ampak obstajajo delčki, ki so specializirani za določene operacijske sisteme.« (2)

#### SQL in MySQL

»SQL je standardni jezik za delo s podatki v relacijskih podatkovnih zbirkah, vendar pa pozna številne različice z malenkostnimi razlikami. Stavke SQL delimo v stavke za definiranje podatkovnih zbirk in stavke za upravljanje s podatki. Vsi stavki za modeliranje in večina stavkov za upravljanje sodi med akcijske poizvedbe, ki ne vračajo podatkov. Poizvedbeni stavki z začetnim ukazom SELECT pa omogočajo poljuben vpogled v podatke baze. Jezik SQL ni občutljiv na velikosti znakov.« (2)

»Od standardnega SQL-a se MySQL razlikuje v nekaj razširitvah in omejitvah. Razširitve so ukaz REPLACE in SHOW, tipi podatkov MEDIUMINT, SET in ENUM ter vsaka podatkovna zbirka je predstavljena kot mapa in vsaka tabela kot datoteka. Njegove omejitve pa so v tem, da ima onemogočene poizvedbe na več nivojih in nima podprte shranjene procedure, sprožilcev, pogledov, ukaza UNION in tujih ključev (obravnavani kot primarna ključi).« (3)

### Predstavitev uporabljene programske opreme

#### Eclipse IDE za Java razvijalce

»Eclipse je integrirano razvojno okolje (IDE) za Javo in druge programske jezike, kot so C, C++, PHP in Ruby itd. Razvojno okolje, ki ga zagotavlja Eclipse, vključuje razvojna orodja Eclipse Java (JDT) za Javo, Eclipse CDT za C/C++, in Eclipse PDT za PHP, itd. Poleg tega se lahko razvojna orodja tudi doda.« (4)

»Ob zagonu izberemo naš delovni prostor v Eclipsu, ki je fizična lokacija za shranjevanje metapodatkov in razvojnih artefaktov. Eclipse ponuja poglede in urejevalnike za krmarjenje in spreminjanje vsebine. Pogled se običajno uporablja za prikaz strukturiranih podatkov in omogoča njihovo neposredno spreminjanje. Urejevalniki se običajno uporabljajo za spreminjanje posameznega podatkovnega elementa, na primer besedilne datoteke. Projekt Eclipse vsebuje izvorne, konfiguracijske in binarne datoteke, povezane z določeno nalogo, ki jo izdelujemo. Projekti Eclipse so lahko dodeljene narave, ki opisujejo namen tega projekta. Narava Java na primer definira projekt kot projekt Java.« (5)

Slika, ki vsebuje besede besedilo, posnetek zaslona, računalnik

Opis je samodejno ustvarjen

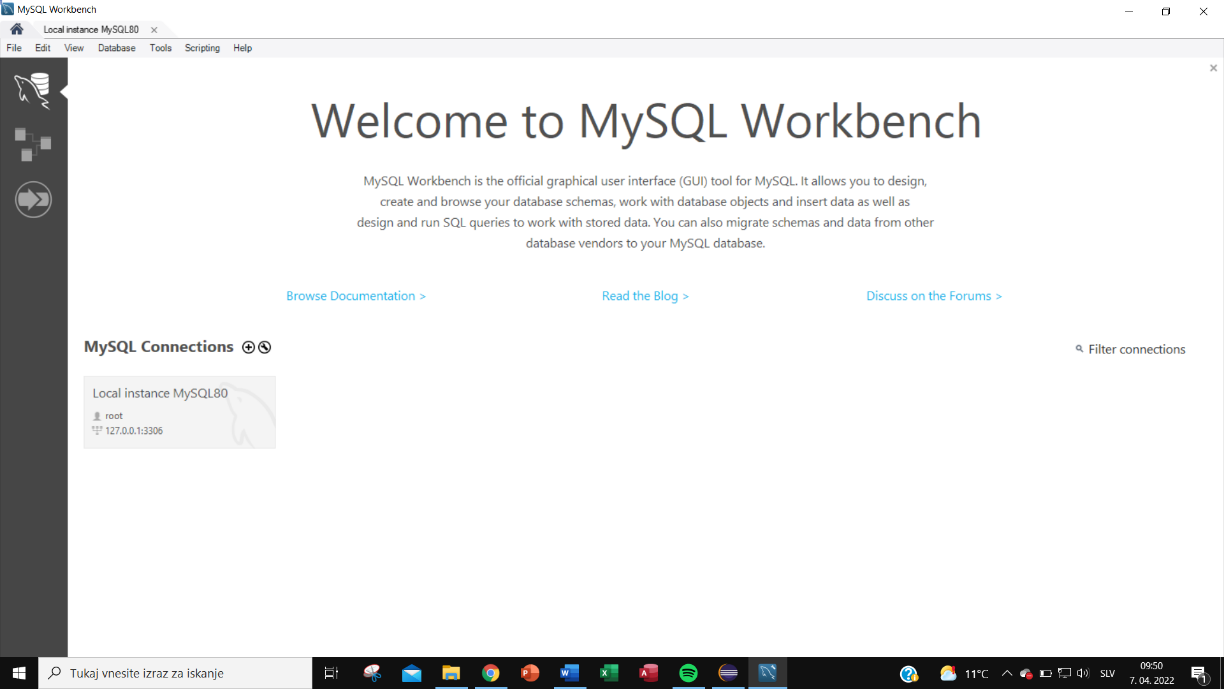
Slika - Vpogled v program Eclipse

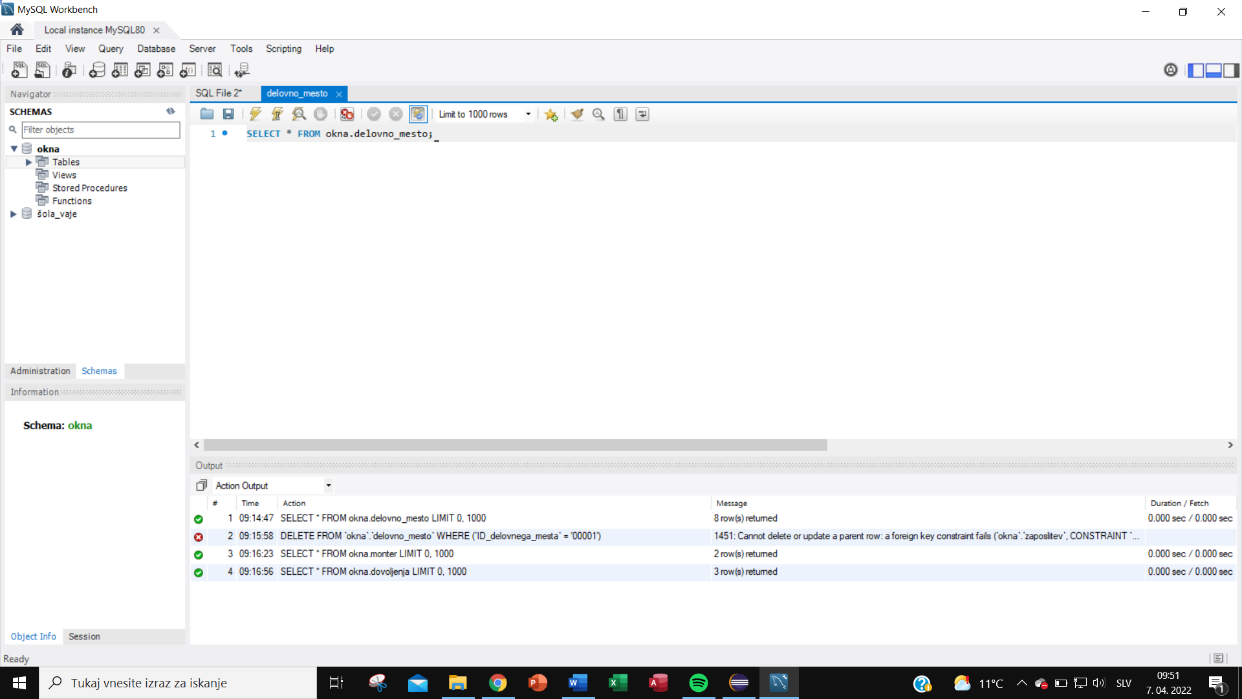
#### MySQL WorkBench in Server

»Strežnik MySQL ponuja sistem za upravljanje baz podatkov s poizvedovanjem in povezljivostjo, pa tudi z možnostjo odlične strukture podatkov in integracije z različnimi platformami. Strežnik MySQL ponuja tudi funkcije, kot so povezljivost, hitrost in varnost, zaradi česar je primeren za dostop do baz podatkov. Strežnik MySQL deluje v sistemu odjemalca in strežnika. Ta sistem vključuje večnitni strežnik SQL, ki podpira različna zaledja (backends), različne odjemalske programe in knjižnice, skrbniška orodja in številne vmesnike za programiranje aplikacij.« (6)

»MySQL Workbench je grafično orodje za delo s strežniki in bazami podatkov MySQL. V celoti podpira strežnik MySQL različice 5.6 in novejše. Funkcionalno pokriva pet glavnih tem. Prvič omogoča ustvarjanje in upravljanje povezav s strežniki baz podatkov ter zmožnost izvajanja poizvedb SQL na povezavah z urejevalnikom SQL. Drugič omogoča grafično ustvarjanje modelov sheme baze podatkov ter urejanje vseh vidikov baze podatkov s pomočjo obsežnega urejevalnika tabel. Tretjič omogoča upravljanje primerkov strežnika MySQL z upravljanjem uporabnikov, varnostnim kopiranjem in obnovitvami, pregledovanjem revizijskih podatkov itd. Četrtič omogoča selitev iz različnih podatkovnih upravljalnih sistemov (SUPB), tabel, predmetov in podatkov v MySQL. Petič podpira Enterprise izdelke, kot so MySQL Enterprise Backup, MySQL Firewall in MySQL Audit.« (7)

»MySQL Workbench je večinoma na voljo v treh izdajah. Community Edition je odprtokodna in prosto naložljiva različica. Podpira ga ogromna skupnost razvijalcev. Standard Edition je komercialna izdaja, ki zagotavlja zmožnost zagotavljanja visoko zmogljivih in razširljivih aplikacij za spletno obdelavo transakcij (OLTP). Zaradi tega je MySQL postal znan skupaj z industrijsko močjo, zmogljivostjo in zanesljivostjo. Enterprise Edition je komercialna izdaja, ki vključuje nabor naprednih funkcij, orodij za upravljanje in tehnične podpore za doseganje najvišje razširljivosti, varnosti, zanesljivosti in časa delovanja. Ta izdaja prav tako zmanjšuje tveganje, stroške, zapletenost pri razvoju, uvajanju in upravljanju aplikacij MySQL.« (8)





Slika - Vpogled v program MySQL Workbench

### Podatkovne baze

»Podatkovna baza je temelj za delovanje katerekoli sodobne organizacije, saj iz nje izhajajo podatki za izvajanje poslovnih procesov ter za pridobivanje in pretvorbo podatkov v informacije. Da je zbirka podatkov opredeljena kot podatkovna baza morajo biti njeni podatki povezani in urejeni, mora biti istočasno na voljo večjemu številu uporabnikom, mora bit shranjena v računalniku in imeti kontrolirano redundanco[[3]](#footnote-3). V PB izvajamo opravila nad podatki kot so zbiranje, vnos v sistem, hranjenje, spreminjanje, posredovanje, varovanje pred izgubo in zaščita pred nepooblaščenim dostopom oziroma zlorabo. « (9)

»Podatke v PB delimo na fizične podatke in metapodatke ki opisujejo lastnosti fizičnih podatkov (kako so shranjeni, kaj pomenijo in kako so dostopni). Oboji so shranjeni na zunanjem polnilniku, od kod jih operacijski sistem posreduje SUPB-ju. V skladu z trinivojsko arhitekturo vsebuje metapodatkovna baza (v njej so metapodatki) tri vrste opisov fizičnih podatkov, s katerimi ločimo načine fizičnega shranjevanja podatkov (notranji nivo), od načinov modeliranja okolja(konceptualni nivo) in le-te od uporabnikov (zunanji nivo). Na notranjem nivoju se zbirka fizičnih datotek preslika v zbirko logičnih zapisov (opisov podatkov). Na konceptualnem nivoju se zbirka logičnih zapisov prikaže kot imena, lastnosti in povezave entitet (prikaz s podatkovnim modelom). Na zunanjem nivoju se konceptualna podatkovna baza prikaže kot uporabnikov model okolja.« (10)

Pri podatkovnih bazah je določeno, da morajo imeti neko stopnjo neodvisnosti.

»Podatek je sam po sebi neodvisen resurs. Podatkovna neodvisnost pomeni imunost oz. odpornost višjega nivoja na spremembe, ki se zgodijo na nižjem nivoju opisov PB. Fizična podatkovna neodvisnost predstavlja neodvisnost konceptualne sheme od sprememb v notranji shemi. Programi ne smejo biti odvisni od načinov shranjevanja podatkov in obratno (strukturo podatkovne datoteke ne sme narekovati aplikacijski program). Logična podatkovna neodvisnost zagotavlja neodvisnost zunanjih shem od sprememb v konceptualni shemi. Omogoča spreminjanje konceptualne sheme, ne da bi pri tem morali nujno spremeniti tudi zunanje sheme.« (11)

»Uporabniki PB so vsi, ki uporabljajo podatke iz podatkovne baze. Delimo jih na posredne uporabnike, ki podajo svoje zahteve na neposredne, ki komunicirajo s PB. Neposredne uporabnike delimo na običajne uporabnike, ki uporabljajo zanje razvite programe in upravljajo preproste naloge. Napredni uporabniki uporabljajo povpraševalni jezik za delo z PB in upravljajo različne ne standardne analize. Razvijalci aplikacij pišejo programe za običajne uporabnike. Načrtovalci PB ali administratorji podatkov skrbijo za logični načrt PB. Skrbniki podatkovnih baz skrbijo za fizični načrt PB.« (9)

»Za izboljšanje karakteristik aplikacij PB se uvaja porazdeljeno procesiranje, kar pomeni, da se podatki v PB procesirajo na neodvisnih računalnikih, ki so povezani s strežnikom. Odjemalec je enouporabniška delovna postaja, na katerem se izvaja primeren uporabniški vmesnik. Strežnik je en ali več večuporabniških računalnikov z deljenim pomnilnikom, ki omogočajo obdelavo, povezovanje, storitve baze in primeren vmesnik. Aplikacije odjemalec-strežnik imajo deljeno procesiranje in podatke med enim ali več odjemalčevimi računalniki, ki izvajajo aplikacijo, in strežnikom, ki nudi storitve vsakemu izmed odjemalcev. Ti računalniki so povezani v omrežje.« (12)

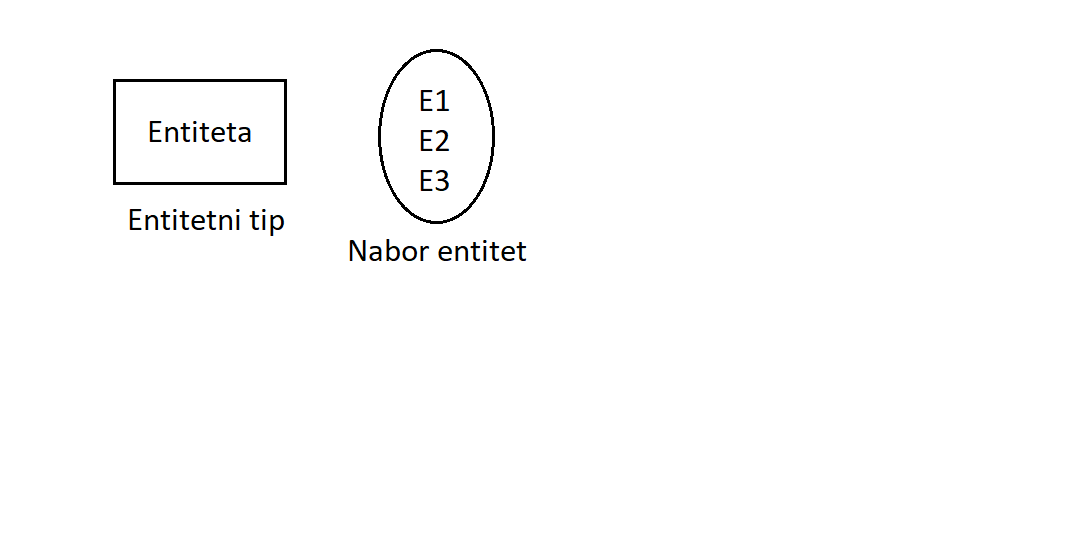
### Grajenje podatkovnih baz

»Postopek izdelave PB sestavlja šest faz. Prva je zbiranje in analiza zahtev uporabnikov, ki vključujejo tudi popise poslovnih procesov, dokumentov in podatkov. Druga faza je konceptualno načrtovanje pri kateri se izdela konceptualno shemo PB(uporaba diagramov). Tretja faza je izbira SUPB. Četrta je logično načrtovanje, pri katerem konceptualno shemo pretvorimo v podatkovni model, ki ga podpira izbrani SUPB in izvede se normalizacija. Peta faza je fizično načrtovanje v kateri se kreira podatkovna baza. Zadnja faza je implementacija PB pri kateri se PB uveljavi znotraj informacijskega sistema, zapolni se s podatki, testira in vzdržuje.« (9)

#### Konceptualno načrtovanje podatkovnih baz

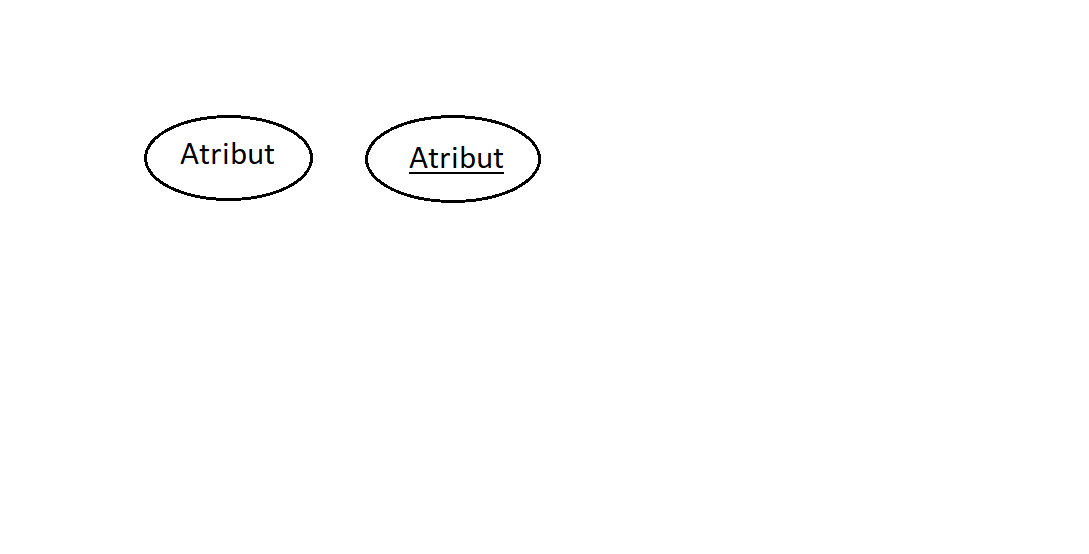
»Konceptualni podatkovni model je organiziran pogled na koncepte baze podatkov in njihova razmerja. Namen ustvarjanja konceptualnega podatkovnega modela je vzpostaviti entitete, njihove atribute in odnose. Na tej ravni modeliranja podatkov o dejanski strukturi baze podatkov ni na voljo skoraj nobenih podrobnosti. Poslovni deležniki in arhitekti podatkov običajno ustvarijo konceptualni podatkovni model.« (13)

»Z uporabo modela uporabniškega okolja oblikovalec razvije podroben konceptualni model baze podatkov. Poleg idejnega modela mora projektant razmisliti o tem, kako naj se uporablja baza podatkov. Določiti je treba vrste transakcij, količino podatkov, pogostost dostopa in druge kvantitativne podatke. Določiti je treba tudi druge omejitve, kot so proračunske omejitve in zahteve glede uspešnosti. Rezultat te faze je niz specifikacij baze podatkov.« (14)

»ER Model je kratica za Entity Relationship Model in je konceptualni diagram podatkovnega modela na visoki ravni. Model ER pomaga sistematično analizirati zahteve po podatkih za izdelavo dobro zasnovane baze podatkov. Model ER predstavlja entitete iz resničnega sveta in odnose med njimi.« (15)

»Entiteta je lahko predmet iz resničnega sveta, živ ali neživ, ki ga je mogoče zlahka prepoznati. Nabor entitet je zbirka podobnih tipov entitet.« (16)

Slika - Primer zapisa entite in nabora entitet

Entitete so predstavljene s pomočjo njihovih lastnosti, imenovanih atributi. Vsi atributi imajo vrednosti in imajo določen obseg vrednosti. Preprosti atributi so atomske[[4]](#footnote-4) vrednosti. Sestavljeni atributi so sestavljeni iz več preprostih atributov. Atributi so lahko tudi eno vrednostni ali več vrednostni. (16)

Slika - Primer zapisa atributa in atributa, ki je tudi ključ

»Ključ je atribut, ki enolično definira entiteto v nizu entitet. Super ključ je niz atributov, ki skupaj definirajo entiteto v naboru entitet. Kandidatni ključ je minimalni super ključ, kar pomeni, da ima najmanjše možno število atributov, da je še vedno super ključ. Primarni ključ je kandidatni ključ, ki ga izbere oblikovalec baze podatkov za enolično identifikacijo nabora entitet. Tuji ključ pa identificira razmerje med entitetami.« (17)

»Povezava med entitetami se imenuje razmerje. Tako kot entitete ima lahko tudi razmerje atribute. Ti atributi se imenujejo opisni atributi. Kardinalnost opredeljuje število entitet v enem nizu entitet, ki ga je mogoče povezati s številom entitet drugega niza prek niza odnosov. Odnosi so lahko v obliki ena na ena, ena na več ali več na več primerkov entitet. Za povezavo ena na ena je na entiteta iz nabora entitet A, lahko povezana z največ eno entiteto iz nabora entitet B in obratno.« (16)

Slika, ki vsebuje besede tabla

Opis je samodejno ustvarjen

Slika - Primer povezave entitet z kardinalnostjo ena proti več

#### Izbira SUPB

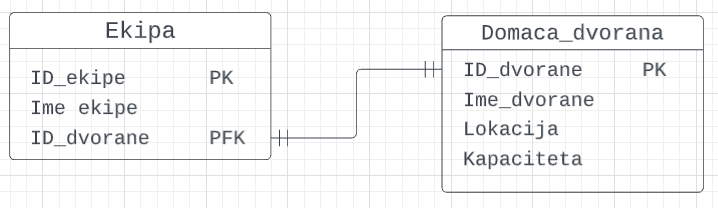
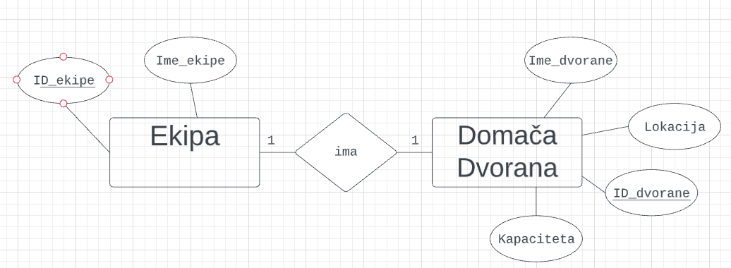
»SUPB je programska oprema, ki uporabnikom omogoča definiranje, kreiranje in vzdrževanje in zagotavlja nadzorovan dostop do PB. Sestavlja ga uporabniški vmesnik, procesor poizvedb, upravitelj transakcij(ACID[[5]](#footnote-5)) in upravitelj shranjevanja (upravitelj izravnalnika[[6]](#footnote-6) in datotek). SUPB omogoča definiranje, spreminjanje opisa ter kreiranje PB, manipuliranje s podatki, izvajanje transakcij, zaščito podatkov, sledenje podatkov, zagotavljanje integritete podatkov, vzdrževanje podatkovnega slovarja, zagotavlja podatkovno neodvisnost, nudi podporo arhiviranju in restavriranju sistema ter spremlja delovanje PB. Uporaba SUPB izboljša nadzor in delo s podatki vendar je zanj značilna kompleksnost in visoke cene.« (11)

»Oracle Database je primer komercialnega sistema za upravljanje relacijskih baz podatkov. Lahko se shrani v oblaku ali lokalno. MySQL je sistem za upravljanje relacijske baze podatkov, ki se običajno uporablja z odprtokodnimi sistemi za upravljanje vsebine in velikimi platformami, kot so Facebook, Twitter in Youtube. SQL Server, ki ga je razvil Microsoft, je sistem za upravljanje relacijskih baz podatkov, zgrajen na podlagi strukturiranega poizvedovalnega jezika (SQL), standardiziranega programskega jezika, ki skrbnikom baz podatkov omogoča upravljanje baz podatkov in poizvedovalnih podatkov.« (18)

»Podatkovna baza je lahko iz vidika upravljanja organizirana v centralizirani ali porazdeljeni obliki. Pri centralizirani organizaciji imamo samo en računalniški sistem s SUPB, ki skrbi za upravljanje celotne baze. Pri porazdeljeni centralizaciji pa je podatkovna baza nameščena na različnih lokacijah povezanih med seboj. Podatki v njej so shranjeni v posameznih računalniških sistemih pod upravljanjem usklajeno delujočih lokalnih SUPB.« (10)

#### Logično načrtovanje podatkovne baze

»Logični podatkovni model se uporablja za definiranje strukture podatkovnih elementov in za nastavitev odnosov med njimi. Logični podatkovni model dodaja dodatne informacije elementom konceptualnega podatkovnega modela. Prednost uporabe logičnega podatkovnega modela je zagotoviti osnovo za oblikovanje osnove za fizični model.« (13)

Do logične sheme podatkovne baze se pride z direktnim načrtovanjem relacijskih shem in normalizacijo ali pa se uporabi pretvorba iz konceptualnega modela. Pri relacijskih podatkovnih bazah je logični podatkovni model relacijski podatkovni model. Njegove osnove so relacija, atribut, vrednostna množica, relacijska shema in odvisnosti med atributi. (19)

Slika - Primer pretvorbe iz ER modela v relacijski model

»Relacijski model predstavlja bazo podatkov kot zbirko relacij. Relacija ni nič drugega kot tabela vrednosti. Vsaka vrstica v tabeli predstavlja zbirko povezanih vrednosti podatkov. Te vrstice v tabeli označujejo entiteto ali razmerje v resničnem svetu. Ime tabele in imena stolpcev so v pomoč pri razlagi pomena vrednosti v vsaki vrstici. Podatki so predstavljeni kot niz odnosov. V relacijskem modelu so podatki shranjeni kot tabele. Vendar pa je fizično shranjevanje podatkov neodvisno od načina, kako so podatki logično organizirani.« (15)

V relacijskem podatkovnem modelu so relacije shranjene v obliki tabel. Ta format shranjuje razmerje med entitetami. Tabela ima vrstice in stolpce, kjer vrstice predstavljajo zapise, stolpci pa atribute. Ena vrstica tabele, ki vsebuje en zapis za to relacijo, se imenuje niz. Končen niz vodnikov v sistemu relacijske baze podatkov predstavlja primer relacije. Primerki relacije nimajo podvojenih nizov. Relacijska shema opisuje ime relacije (ime tabele), atribute in njihova imena. Vsaka vrstica ima enega ali več atributov, znanih kot relacijski ključ, ki lahko enolično identificirajo vrstico v relaciji (tabela). Vsak atribut ima vnaprej določen obseg vrednosti, znan kot domena atributa. (20)

Tabela - Primer relacijske tabele s ključem ID

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Ime | Priimek | Starost |
| 1 | Mitja | Rac | 34 |
| 2 | Tanja | Globinc | 30 |
| 3 | Toni | Hitrec | 25 |

»Omejitve relacijske integritete v SUPB se nanašajo na pogoje, ki morajo biti prisotni za veljavno relacijo. Te relacijske omejitve v SUPB izhajajo iz pravil v mini svetu, ki ga predstavlja baza podatkov. V DBMS obstaja veliko vrst omejitev integritete. Omejitve sistema za upravljanje relacijske baze podatkov so večinoma razdeljene na omejitve domene, ključne omejitve in omejitve referenčne celovitosti.« (15)

#### Normalizacija

»Normalizacija je proces učinkovite organizacije podatkov v bazi podatkov. Obstajata dva cilja procesa normalizacije: odstranitev odvečnih podatkov (na primer shranjevanje istih podatkov v več kot eni tabeli) in zagotovitev, da so odvisnosti podatkov smiselne (samo shranjevanje povezanih podatkov v tabeli). Oba sta vredna cilja, saj zmanjšujeta količino prostora, ki ga zaužije baza podatkov, in zagotavljata logično shranjevanje podatkov.« (21)

»Obstaja nekaj pravil za normalizacijo baze podatkov. Vsako pravilo se imenuje "normalna oblika". Če upoštevamo prvo pravilo, naj bi bila baza podatkov v "prvi normalni obliki". Če se upoštevajo prva tri pravila, se šteje, da je zbirka podatkov v "tretji normalni obliki". Čeprav so možne tudi druge stopnje normalizacije, se tretja normalna oblika šteje za najvišjo raven, ki je potrebna za večino aplikacij.« (22)

Prva normalna oblika določa temeljna pravila za organizirano bazo podatkov. Pri tej obliki se odstrani podvojene stolpce iz iste tabele in ustvari se ločene tabele za vsako skupino povezanih podatkov in identificira vsako vrstico z edinstvenim stolpcem ali nizom stolpcev (primarni ključ). Druga normalna oblika izpolnjuje vse zahteve prve normalne oblike, odstrani pod nabore podatkov, ki veljajo za več vrstic tabele, in jih postavi v ločene tabele. Med temi novimi tabelami in njihovimi predhodnicami ustvari razmerja z uporabo tujih ključev. Tretja normalna mora izpolnjevati vse zahteve druge normalne oblike in odstrani stolpce, ki niso odvisni od primarnega ključa. (21)

Tabela - Nenormalizirana tabela

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Ime | Priimek | Razred | Razrednik | MAIL\_Razrednik |
| 1 | Tim | Kolenc | T4c | Klemen Petan | Petan.klemen@gmail.com |
| 2 | Gašper | Ravbar | T4c | Klemen Petan | Petan.klemen@gmail.com |
| 3 | Jure | Bukovec | T4c | Klemen Petan | Petan.klemen@gmail.com |

Tabela - Normalizirana tabela (razdeljena na dve tabeli)

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Ime | Priimek | Razred |
| 1 | Tim | Kolenc | T4c |
| 2 | Gašper | Ravbar | T4c |
| 3 | Jure | Bukovec | T4c |

|  |  |  |
| --- | --- | --- |
| Razred | Razrednik | MAIL\_Razrednik |
| T4c | Klemen Petan | Petan.klemen@gmail.com |

#### Dokumentacija

»Dokumentacija baze podatkov je niz opisov podatkov, ki olajšajo upravljanje, vzdrževanje in iskanje virov. Dokumentacija je lahko v različnih oblikah, vendar običajno deluje kot podatkovni katalog, ki vsebuje podrobne informacije o shemi tabele, vseh razmerjih, tipih podatkov itd. Opis naborov podatkov in posameznih podatkovnih objektov se imenuje tudi metapodatki (podatki o podatki).« (23)

»Orodja, zasnovana za podporo specifičnim tehnikam poslovnega modeliranja, se pogosto imenujejo orodja računalniško podprtega programskega inženiringa (CASE). Orodje CASE je izdelek, ki pomaga analizirati, modelirati in dokumentirati poslovne procese. Je orodje ali nabor orodij, ki podpira osnovna načela in metode analize. Nekatera orodja so zasnovana posebej za podporo določeni tehniki, medtem ko so druga orodja splošnejše narave.« (24)

#### Fizično načrtovanje

»Fizični podatkovni model opisuje izvedbo podatkovnega modela, specifično za bazo podatkov. Ponuja abstrakcijo baze podatkov in pomaga ustvariti shemo. To je posledica bogastva metapodatkov, ki jih ponuja model fizičnih podatkov. Fizični podatkovni model pomaga tudi pri vizualizaciji strukture baze podatkov s podvajanjem ključev stolpcev baze podatkov, omejitev, indeksov, sprožilcev in drugih funkcij SUPB.« (13)

»V tem koraku strokovnjaki za bazo podatkov preučujejo vidike, na primer, kateri stolpci potrebujejo indekse, ali je treba tabele razdeliti na particije in kako je treba nastaviti fizične parametre prostorov tabel. Lahko se celo odločijo za prestrukturiranje tabel za izboljšanje učinkovitosti. Rezultat fizične zasnove baze podatkov je model baze podatkov, ki prikazuje vse tabele, njihove stolpce in njihove ključe.« (25)

»Fizični podatkovni model se lahko uporabi za generiranje stavkov DDL[[7]](#footnote-7), ki jih je mogoče nato razmestiti v strežnik baz podatkov. Tipičen tok opravil za modeliranje fizičnih podatkov je: Izdelava fizičnega podatkovnega modela, po potrebi se doda pomnilniške predmete, indekse in poglede, ustvarite se skript DDL za fizični podatkovni model in se kript DDL zažene, da se ustvarijo ustrezni podatkovni objekti na strežniku.« (26)

Za implementacije strežnika se lahko tako uporablja skupina domačih računalnikov, če bomo imeli bazo omejeno na domače omrežje. Za dostop preko spleta se uporabljajo razne plačljive storitve gostovanja ali pa sami gostujemo omrežje z domačimi napravami, ki pa morajo biti dovolj zmogljive. Pri osebnem gostovanju je potrebno skrbeti za veliko nastavitev, ki jih ponudniki storitev že imajo integrirane. (27)

### Povezava podatkovnih baz v omrežje

»Izdelava in objava dinamičnih spletnih strani vključuje tudi povezavo s podatkovnimi zbirkami. Za to je potrebno imeti na strežniku podatkovno zbirko kot je MySQL in hkrati je na spletnem strežniku, ki nam ga ponuja ponudnik za dostop v internet, potrebno imeti skriptni jezik kot je PHP[[8]](#footnote-8) ter druge jezike za izdelavo spletne strani. Pri procesu poizvedbe brskalnik sproži poizvedbo in s tem kodo PHP (in posledično HTML), ki se obdela na spletnem strežniku in pošlje SQL ukaze na strežnik MySQL. Iz tega strežnika se potem vrne rezultat poizvedbe in posreduje uporabniku v obliki HTML.« (3)

Podatkovne baze se veliko uporabljajo pri spletnih straneh in zato potrebujejo tudi znanja iz oblikovanja spletnih strani in skriptnih jezikov. V tej seminarski nalogi je sicer namen izdelati podatkovno bazo za lokalno omrežje z programsko aplikacijo, vendar sem se odločil vključiti ta odstavek kot zanimivost in nakaz na zmožnost nadgradnje spletnega dostopa do moje baze podatkov. Vendar pa je za spletne podatkovne baze kot za samo oddaljen dostop do podatkovnih baz potrebno pridobiti dostop do interneta z gostovanjem strežnika preko raznih ponudnikov teh storitev.

»Za povezavo do baze z MySQL Workbench je potrebno v dodati novo povezavo. V konfiguraciji za novo povezavo poljubno poimenujemo server in za povezovalno metodo izberemo standardno (TCP/IP). Pod Hostname zapišemo IP naslov naše domene ali če uporabljamo storitev gostovanja kot je cPanel uporabimo IP, ki ga nam ponudi ponudnik storitve. Zapišemo še uporabniško ime ali geslo (v cPanel in tudi MySQL lahko nastavimo različne uporabnike in njihova geslo). Potrdimo nastavitve in shrani se nam nova povezava v namizju orodja Workbench.« (28)

Alternativno če uporabljamo podatkovno bazo shranjeno na naši napravi lahko vzpostavimo povezavo na isti način kot je opisano zgoraj, samo da se v polja vnese določene privzete nastavitve, med katerimi so mnoge že v osnovi vpisane ob odprtju novo povezave.

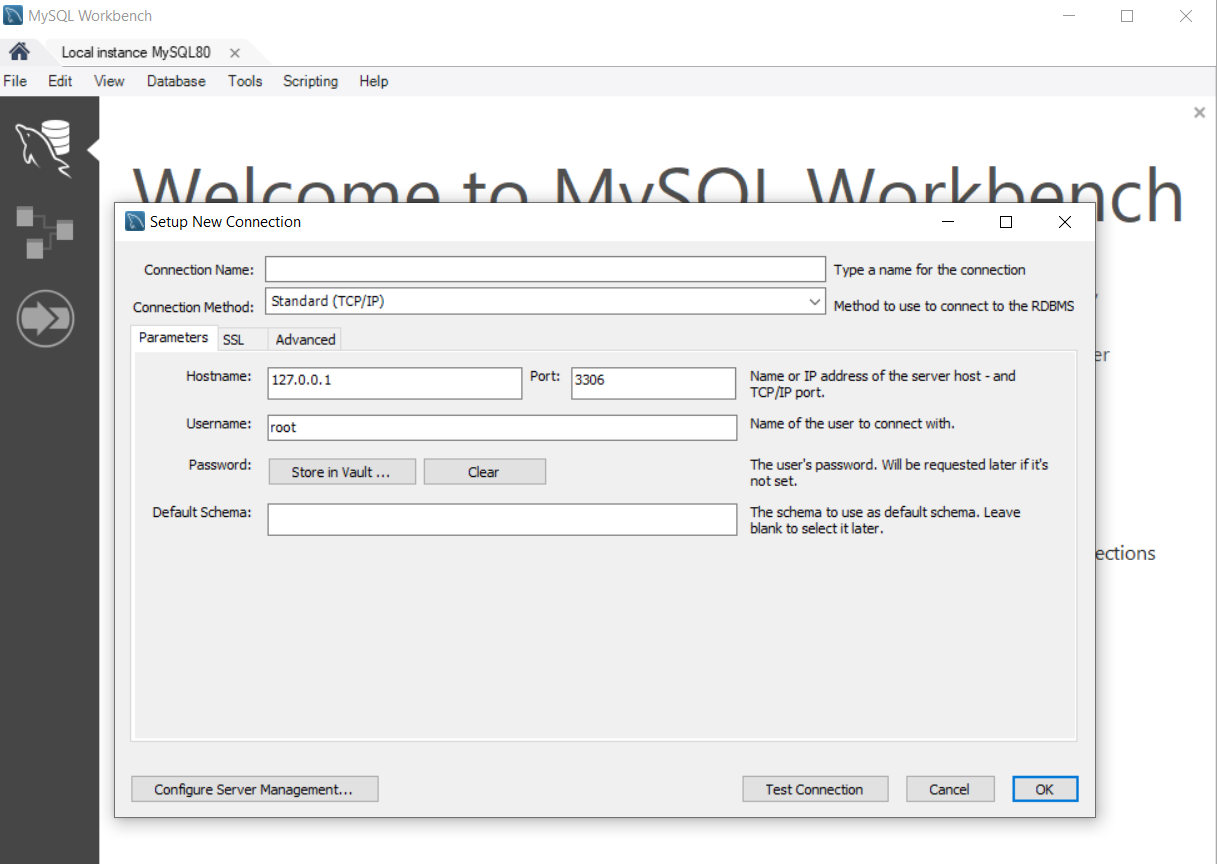


Tabela - Vzpostavitev povezave s PB v MySQL Workbench (vnesene so nastavitve za povezavo v lokalno bazo podatkov)

### Podatkovne baze in Java

»Java ima zmožnost povezovanja s podatkovnimi bazami v obliki standarda JDBC(Java DataBase Connectivity). Poglavitna prednost tehnologije poslovne Jave je neodvisnost od podlage in izvedbe podatkovne zbirke. JDBC programerjem omogoča, da s podatkovnimi viri delajo na enoten način, ne glede na izbran podatkovni strežnik. Programska koda v javi izkorišča s pomočjo upravnika gonilnikov(driver manager) na različne vrste dostopa do podatkovnega vira.« (2)

»Z rabo knjižnice JDBC je dostop do podatkovnega vir sledeč: 1. pridobitev povezave s podatkovnim virom, 2. s pomočjo povezave pridemo do podatkov. 3. Podatke obdelamo in po potrebi shranimo nazaj na strežnik. V JDK je vključena knjižnica JDBC, ki omogoča povezovanje najpreprostejših virov. Za razvoj uporabnejših programov se uporabljajo zmogljivejši podatkovni strežniki kot so MySQL, ki ponuja svoj JDBC gonilnik.« (2)

»Preden vzpostavimo povezavo z podatkovnim virom, mora izvajalni sistem prepoznati ustrezen gonilnik. Obstajajo tri možnosti navajanja gonilnikov. Nastavitev sistemske lastnosti jdbc.driver, najava razreda v ukazni vrstici in s pomočjo tehnike vpogleda. Ko se odločimo vzpostaviti povezavo moramo podatkovni vir opredeliti s posebnim naslovom URL (database URL). Povezavo se vzpostavi s klicem DriveManager.getConnection (<naslov>, <ime>, <geslo>) (glej Koda 1). Z pripravo ukaznih stavkov razreda Statement posredujemo stavke SQL podatkovnemu viru. Akcijske stavke običajno izvedemo z metodo executeUpdate() (glej Koda 2). Poizvedbene stavke pa prestrežemo s predmetom razreda ResultSet, ki ga vrne metoda executeQuery(). Pri branju rezultatov poizvedbenih stavkov uporabljamo while zanko po kateri se pomikamo z next() metodo (glej Koda 3).« (2)

// Z importom vpeljemo naš gonilnik JDBC

**import** com.mysql.cj.xdevapi.Schema.CreateCollectionOptions;

**public** **class** Razred{

// To je primer url za povezavo v lokalno PB

**static** String *url* = "jdbc:mysql://localhost:3306/okna";

// Vnesena uporabnik in geslo sta prehodno določena za povezavo v lokalno PB

**static** String *uporabnik*="root";

**static** String *geslo*="nakljucnoGESLO";

**public** **static** **void** main(String[] args) **throws** ClassNotFoundException, SQLException {

// S pomočjo razreda Connection določimo našo povezavo s PB

Connection povezava = DriverManager.*getConnection*(*url*, *username*, *password*);

…

Koda - Povezava javanskega programa s podatkovno bazo MySQL server

String ukaz="UPDATE tabela SET atribut=vrednost WHERE ključ=vrednost;

// Tak tip izvrševanja SQL ukaza v javi velja za vse SQL razen selekcije

PreparedStatement izjava=povezava.prepareStatement(ukaz\_popravi);

izjava.executeUpdate();

…

Koda - Primer izvrševanja DDL in DML ukazov v Javi

Statement izjava=povezava.createStateent();

// Za izvrševanje SQL selekcije podatkov se za izvajanje ukaza uporabi metoda

// executeQuery. Rezultat poizvedbe se shrani v ResultSet razred.

ResultSet rezultat=izjava.executeQuery(SELECT \* FROM tabela);

// Za pomikanje skozi rezultate poizvedbe se uporabi while zanke in funkcija next()

**while**(rezultat.next()) {

// Za izpis rezultata se uporablja funkcija getString v kateri opredelimo kateri

// atribut iščemo.

System.out.println(rezultat.getString(atribut));

}

Koda - Primer izvrševanja DQL ukazov v javi

### Izdelava uporabniških aplikacij v Javi

Za izdelavo grafičnih aplikacij v Javi se uporablja knjižnica Swing.

»Swing je del Java Foundation Classes(JFC), v katerem so shranjene grafične komponente (gumbi, zavihki, seznami…). Pri programiranju se uporablja še veliko drugih knjižnic s katerimi lahko izdelujemo okenske programe, kakor tudi javanske programčke.« (1)

»Paket knjižnic Swing je nastal kot zamenjava in razširitev obstoječih težkih gradnikov[[9]](#footnote-9) iz paketa AWT. Velik del gradnikov so nadgrajene različice iz paketa AWT ali pa razširjajo njegove razrede. Swing tudi omogoča izdelavo programčkov[[10]](#footnote-10) Swing, ki potrebujejo za razliko od AWT referenco na delovno površino z metodo getcontentPane(). Vendar se je podpora za Swing programčke ukinila in preusmerila čisto na samostojne programe.« (2)

»Pred programiranjem Swingowih programov se mora vpeljati potrebne knjižnice. Nato se, kot pri običajnem postopku, doda razrede in glavni blok main. Preko razreda JFrame določimo okvir programa, kateri moramo nastaviti velikost (setSize()) in vidljivost (setVisible(true)). Okvirju je potrebno dodati tudi način zapiranja (okvir.setDefaultCloseOperation()). Za dodajanje vseh elementov se večkrat uporablja JPanel ali druge ploskve(JSplitPane, JScrollPane, JTabbedPane, JToolBar..), ki ga vstavimo na okvir in potem nanj dodajamo elemente kot so gumbi, seznami, napisi, tabele, slike in tako dalje (pod pogojem se lahko tudi dodaja na gol okvir).« (1)

**import** javax.swing.\*;

**import** javax.awt.\*;

**public class** Razred\_aplikacije{

**public** **static** **void** main(String[] args){

// Definiramo okvir programa

JFrame okvir= **new** JFrame("Okvir aplikacije");

// Za okvir nastavimo ključne nastavitve.

okvir.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

okvir.setVisible(**true**);

//Določimo panelo in izberemo način postavitev (Layout) – primer Gridbaglayout

JPanel panela=**new** JPanel();

panela.setLayout(**new** GridBagLayout());

GridBagConstraints gbc =**new** GridBagConstraints();

gbc.gridx=0;

gbc.gridy=0,

// Panelo dodamo na okvir in na panelo potem še poljuben element

okvir.add(panela);

JLabel napis=new JLabel("Dober dan");

panela.add(napis, gbc);

}}

Koda - Primer uporabe razreda Swing

»LayoutManagerji se uporabljajo za razporeditev komponent po površini. Olajšajo nadzor postavitve in velikosti komponent v obrazcih GUI. LayoutManager je vmesnik, ki ga izvajajo vsi razredi upraviteljev postavitve. Obstajajo naslednji razredi, ki predstavljajo upravitelje postavitve: BorderLayout, FlowLayout, GridLayout, CardLayout, GridBagLayout,

BoxLayout, GroupLayout, ScrollPaneLayout, SpringLayout itd.« (29)

»Spreminjanje stanja predmeta je znano kot dogodek. Na primer, klik gumba, premik miške itd. Paket java.awt.event ponuja številne razrede dogodkov in vmesnike poslušalca za ravnanje z dogodki (ActionListener, MouseListener and MouseMotionListener, MouseWheelListener, KeyListener, ItemListener, TextListener, AdjustmentListener, WindowListener, ComponentListener, ContainerListener, FocusListener). Metoda actionPerformed() se uporablja kot odgovor na sprožen dogodek, ki ga zazna nek poslušalec. « (30)

gumb.addActionListener(**new** ActionListener() { **public** **void** actionPerformed(ActionEvent e) {

System.out.println("Gumb je bil pritisnjen");

}

});

…

Koda - Primer akcijskega poslušalca za gumb

## Izdelava podatkovne baze in aplikacije

### Dokumentacije izdelave baze

V tej seminarski nalogi je eden iz med ciljev izdelati podatkovno bazo na osnovi manjšega obrata v podjetju. Za model obrata sem vzel izdelavo oken, saj sem s takim obratom prišel v stik med počitniškim delom. Načrtovanje podatkovnih baz narekuje da se lahko sprva ustvari ER model in se ga nato pretvori v relacijskega. Lahko pa tudi takoj začnemo z relacijskim modelom. V seminarski nalogi sem začel s postopkom izdelave ER modela, nato pa sem v fazi izdelave diagrama ustvaril relacijski diagram podatkovne baze.

#### Izdelava ER in relacijskega modela

Na začetku grajenja baze je potrebno izdelati načrt v obliki ER diagrama. Najprej je treba določiti vse procese v okolju za katerega izdelujemo podatkovno bazo, da lahko izluščimo entitete.

Izdelava oken poteka v večjem številu faz. Sprva kot pri mnogih drugi obratih in podjetjih imamo skladišče z zalogami za ostale faze in vsemi končnimi produkti. Druga faza je žaga pri kateri se zakupljena okovja nažaga na naročene kose. Druga faza je vstavljanje železa v profile za njihovo ojačenje. Iz te faze izhajata dve fazi, ki sta si v namenu enaki. Glede na tip okovja oziroma profila okna se ali železo pritrdi s strojem ali pa ročno. Iz obeh faz profili grejo v fazo sestavljanja profilov v sama okna, ki je tudi malo porazdeljeno glede na tip okna, ampak ker je končni namen in proces tako podoben bom to fazo štel kot eno entiteto. V sestavljanje oken je vključeno tudi pritrditev nosil, kljuk, odpiralnih mehanizmov, itd. . Zadnja glavna faza je vstavljanje stekla v okna in vrata.

1. Določitev entitet

Iz opisa proizvodnje oken sledijo entitete: Skladišče, Obdelava\_podatkov, Ojačitev in Steklitev oken. Fazo sestavljanja bo poosebljala entiteta Artiklov, saj po sestavljanju nastane končni artikel. Namesto sestavljanja bom uporabil tudi entiteto Okovje, ki bo povezana z Artikli in bo imela informacija o vseh okovjih za posamezen proizvod.

Poleg faz imamo v obratu seveda zaposlene, ki delajo na osnovi naročil, ki pridejo v obrat od nekega kupca. Tako da imamo še entitete: Zaposleni, Zaposlitev zaposlenih, Delovna mesta, Naročilo in Kupec.

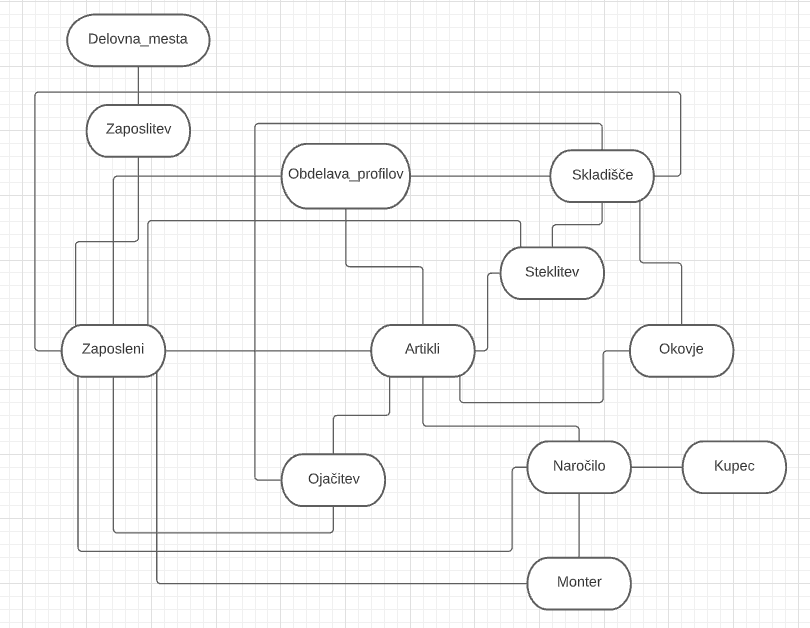
Tako imamo skupaj entitete Zaposleni, Kupec, Zaposlitev, Skladišče, Naročilo, Delovna\_mesta, Obdelava\_profilov, Artikli, Ojačitev, Monter, Steklitev in Okovje.

1. Določitev primarnih ključev entitet

Tabela - Tabela s entitetami in primarnimi ključi

|  |  |  |
| --- | --- | --- |
| Zaposleni (ID\_zaposlenega\*) | Kupec (ID\_kupca\*) | Zaposlitev(ID\_zaposlitve\*) |
| Obdelava\_profilov (ID\_profila\*) | Artikli(ID\_proizvoda\*) | Skladišče (ID\_artikla\*) |
| Naročilo (ID\_naročila\*) | Delovna\_mesta(ID\_mesta\*) | Ojačitev (ID\_železa\*) |
| Monter(ID\_skupine\*) | Steklitev (ID\_steklo\*) | Okovje(ID\_kompleta\*) |

1. Začetna skica entitetnega modela:



Slika 7 Skica ER modela za obrat Okna

Entiteta zaposleni je povezana z skoraj vsemi entitetami, saj je v vsaki fazi izdelave oken prisotna vsaj ena oseba. Vsak zaposleni ima neko zaposlitev in vsaka zaposlitev se izvaja na nekem delovnem mestu. Entitete Obdelava\_profilov, Ojačitev, Steklitev in Okovje so povezani s skladiščem, saj iz njega prejemajo material za delo. V proizvodu se združujejo vsi procesi sestavljanja. Proizvodi se izdelajo glede na naročilo, ki je povezan s kupcem, monterjem in nekim zaposlenim.

1. Opredelitev atributov in njihova razporeditev po entitetah:

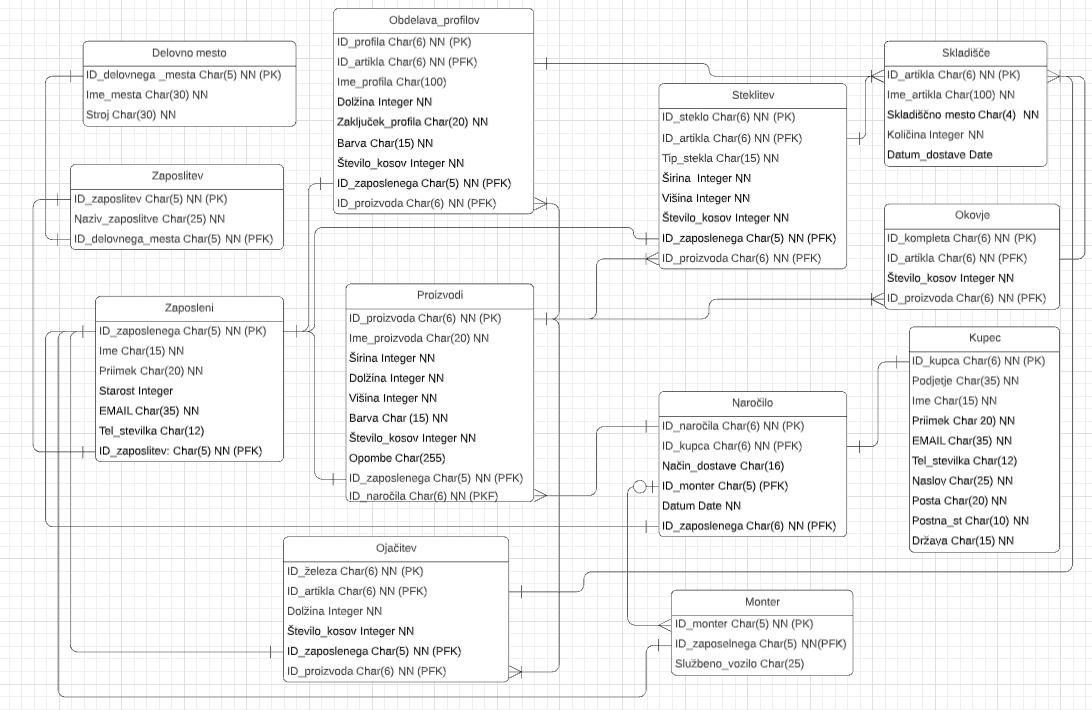
Tabela - Tabela atributov po entitetah

|  |
| --- |
| Zaposleni (ID\_zaposlenega\*, Ime, Priimek, Starost, EMAIL, Tel\_stevilka, ID\_zaposlitev) |
| Zaposlitev(ID\_zaposlitev\*, Naziv\_zaposlitve, ID\_delovnega\_mesta) |
| Delovno\_mesto(ID\_delovnega\_mesta\*, Ime\_mesta, Stroj) |
| Kupec (ID\_kupca\*, Podjetje, Ime, Priimek, EMAIL, Tel\_stevilka, Naslov, Posta, Postna\_st, Država) |
| Naročilo (ID\_naročila\*, ID\_kupca, Način\_dostave, ID\_monter, Datum, ID\_zaposlenega) |
| Proizvodi (ID\_proizvoda\*, Ime\_proizvoda, Širina, Dolžina, Višina, Barva, Število\_kosov, Opombe, ID\_zaposlenega, ID\_naročila) |
| Monter (ID\_monter\*, ID\_zaposlenega, Službeno\_vozilo) |
| Okovje (ID\_kompleta\*, ID\_artikla, Število\_kosov, ID\_proizvoda) |
| Skladišče (ID\_artikla\*, Ime\_artikla , Skladiščno\_mesto, Količina, Datum\_dostave) |
| Obdelava\_profilov (ID\_profila\*, ID\_artikla, Ime\_profila, Dolžina, Zaključek\_profila, Barva, Število\_kosov, ID\_zaposlenega, ID\_proizvoda) |
| Ojačitev (ID\_železa\*, ID\_artikla, Dolžina, Število\_kosov, ID\_zaposlenega, ID\_proizvoda) |
| Steklitev (ID\_steklo\*, ID\_artikla, Tip\_Stekla, Širina, Dolžina, Število\_kosov, ID\_zaposlenega, ID\_proizvoda) |

1. Relacijske sheme:

Tabela - Relacijska shema PB

|  |
| --- |
| Zaposleni (ID\_zaposlenega: A(5), Ime: A(15), Priimek: A(20), Starost: N, EMAIL: A(35), Tel\_stevilka: A(12) , ID\_zaposlitev: A(5) →Zaposlitev)) |
| Zaposlitev (ID\_zaposlitev: A(5), Naziv\_zaposlitve: A(25), ID\_delovnega\_mesta: A(5) → Delovno\_mesto) |
| Delovno\_mesto (ID\_delovnega\_mesta: A(5), Ime\_mesta: A(30), Stroj A(30)) |
| Kupec (ID\_kupca: A(6), Podjetje: A(35), Ime: A(15), Priimek: A(20), EMAIL: A(35), Tel\_stevilka: A(12), Naslov: A(25), Posta: A(20), Postna\_st: A(10), Država: A(15)) |
| Naročilo (ID\_naročila: A(6), ID\_kupca: A(6) →Kupec, Način\_dostave; A(16), ID\_monter: A(5) →Monter, Datum: Date, ID\_zaposlenega: A(5) →Zaposleni) |
| Proizvodi(ID\_proizvoda: A(6), Ime\_proizvoda: A(20), Širina: N, Dolžina: N, Višina: N, Barva: A(15), Število\_kosov: N, Opombe: A(255), ID\_zaposlenega: A(5) →Zaposleni, ID\_naročila: A(6) →Naročila) |
| Monter(ID\_monter: A(5), ID\_zaposlenega: A(5) →Zaposleni, Službeno\_vozilo: A(25)) |
| Okovje(ID\_kompleta: A(6), ID\_artikla: A(6)→Skladišče, Število\_kosov; N, ID\_proizvoda: A(6)→Proizvodi) |
| Skladišče (ID\_artikla: A(6), Ime\_artikla: A(100) , Skladiščno\_mesto: A(4) NN, Količina: N, Datum\_dostave: Date) |
| Obdelava\_profilov (ID\_profila\*, ID\_artikla: A(6)→Skladišče, Ime\_profila: A(100), Dolžina: N, Zaključek\_profila: A(20), Barva: A(15), Število\_kosov: N, ID\_zaposlenega: A(5) →Zaposleni, ID\_proizvoda: A(6)→ Proizvodi) |
| Ojačitev (ID\_železa: A(6), ID\_artikla: A(6)→Skladišče, Dolžina: N, Število\_kosov: N, ID\_zaposlenega: A(5) →Zaposleni, ID\_proizvoda: A(6)→ Proizvodi) |
| Steklitev (ID\_steklo: A(6), ID\_artikla: A(6)→Skladišče, Tip\_Stekla: A(15), Širina: N, Dolžina: N, Število\_kosov: N, ID\_zaposlenega: A(5) →Zaposleni, ID\_proizvoda: A(6)→ Proizvodi) |

1. ER diagram:

Slika 8 ER diagram za obrat Okna

#### Grajenje baze v orodju MySQL

V programskem orodju MySQL WorkBench najprej ustvarimo povezavo do naše podatkovne baze. V mojem primeru se bo ta nahajalo na domači napravi, zato nastavimo povezavo do lokalnega gostitelja (glej 2.1.5). Znotraj nape podatkovne baze nato ustvarimo shemo, ki bo vsebovala vse naše tabele. To se naredi s ukazom CREATE SCHEMA Okna. Nato v bazo dodamo vse načrtovane tabele. Primer tega je naslednji odsek kode (Priloga 1).

create table IF NOT EXISTS Zaposleni(

ID\_zaposlenega Char(5) not null,

Ime Char(15) not null,

Priimek Char(20) not null,

Starost Integer,

EMAIL Char(25) not null,

Tel\_stevilka Char(12),

Zaposlitev Char(15) not null,

Primary Key (ID\_zaposlenega)

);

Koda 6 - Primer dodajana tabel v PB

Po končani izvedbi celotne kode dobimo naslednji rezultat.

Slika, ki vsebuje besede besedilo

Opis je samodejno ustvarjen

Slika 9 Prikaz tabel za obrat Okna v MySQL

Preden se lotimo izdelave programa, ki bo delal s to bazo je potrebno v vsako tabelo vnesti nekaj poskusnih podatkov (Priloga 2). Ti podatki so izključno poskusne narave. Sledijo dani tematiki izdelave oken, vendar niso točni.

INSERT INTO obdelava\_profilov (ID\_profila, ID\_artikla, Ime\_profila, Dolžina, Zaključek\_profila, Barva, Število\_kosov, ID\_zaposlenega, ID\_proizvoda)

VALUES('000001', '000002', 'Okvir', 1050, '/', 'Bela',2, '00005', '000001' ),

('000002', '000003', 'Pokončnik', 1050, 'Spojnik za pokončnik', 'Bela',2, '00005', '000001' ),

…

Koda 7 - Primer vstavljanja podatkov v tabelo

Če želimo postaviti to podatkovno bazo na novi napravi v domačem omrežju, samo sledimo zgornjim navodilom. Če bi pa želeli imeti PB povezano z internetom, bi potrebovali drugačen postopek in tudi drugače zasnovano aplikacijo v nadaljevanju.

### Dokumentacija izdelave programa

Za izdelavo aplikacije sem si zamislil tip aplikacije, ki bi bil namenjen naivnim uporabnikom in bi omogočal grafično obdelavo podatkov in v skrajni sili tudi sposobnost urejanja samih tabel. Aplikacija ne bi omogočala kompleksnejših poizvedb ali prirejanja lastnosti že obstoječih tabel, vendar bi se v prihodnosti dalo to tudi naknadno dodati. Pri tem pa je vendarle treba upoštevati, da je SUPB MySQL-a v vsakem primeru bolj primeren za kompleksnejše operacije in je tako aplikacija namenjena za neizkušene uporabnike.

V mojem primeru bo moj računalnik igral vlogo delovne postaje in strežnika. V praksi je bolje, da so te funkcije na ločenih napravah zaradi boljših karakteristik.

V mojih seminarski nalogi sem si zastavil, da se ne bom posluževal gostovanja ali oddaljenega dostopa vendar sem postopek nastavitve opisal v teoretičnem delu.

#### Povezovanje baze z Javo

Za povezovanje podatkovnih baz s raznimi programerskimi okolji se uporabijo povezovalniki (connectors), ki jih ponujajo ustvarjalci SUPB-jev.

MySQL ponuja MySQL JDBC driver na svoji spletni strani za povezovanje z razvijalnim okoljem Eclipse. Prenesemo ga in shranimo na napravo.

Slika, ki vsebuje besede besedilo

Opis je samodejno ustvarjen

Slika 10 - Spletna stran MySQL z Connectorji

Slika, ki vsebuje besede miza

Opis je samodejno ustvarjen

Slika 11 - Datoteka s konektorjem

V Javi v našem Java projektu (Glejte spodnjo sliko) pod lastnosti najdemo knjižnice in dodamo shranjen konektor oziroma gonilnik, kot enega izmed dodatnih modulov. To nam omogoči povezovanje z MySQL serverjem v tem Java projektu.

Slika, ki vsebuje besede besedilo

Opis je samodejno ustvarjen

Slika 12 - Dodajanje konektorja v knjižnico modulov

#### Izdelava uporabniškega vmesnika v Javi

Program, ki ustvari uporabniški vmesnik za naivne oziroma končne uporabnike je sestavljen iz dveh glavnih razredov. Prvi razred MySQL\_Okna je javni razred in nosilec glavne metode. V sami glavni metodi je inicializirana in deklarirana povezava s podatkovno bazo preko MySQL JDBC gonilnika in inicializacija razreda za izdelavo samega grafičnega vmesnika. Za vzpostavitev povezave sem v glavni razred dodal konstantne spremenljivke za URL podatkovne baze in elemente razreda Swing za izdelavo prijavnega okna. V razredu je tudi konstruktor za nastavitev grafičnih komponent.

//Uvoz knjižnic za razne uporabljene metode

**import** java.sql.\*;

**import** java.time.LocalDateTime;

**…**

//Uvoz gonilnika za povezavo s PB

**import** com.mysql.cj.xdevapi.Schema.CreateCollectionOptions;

**public** **class** MySQL\_Okna {

//Konstanta s naslovom podatkovne baze

**static** String *url* = "jdbc:mysql://localhost:3306/okna";

//Ta konstanta se uporalja v beleženju zgodovine uporabe aplikacije

**public** **static** String *pravi\_uporabnik*="";

//Deklaracija in inicializacija grafičnih Swingovih komponent

**static** JFrame *prijava*=**new** JFrame("Prijava v sistem");

**static** JPanel *vsebina\_prijave*=**new** JPanel();

**…**

//Konstruktor glavnega razreda s katerim nastavimo lastnosti grafičnih komponent

MySQL\_Okna(){

*prijava*.add(*vsebina\_prijave*);

*prijava*.setDefaultCloseOperation(JFrame.***DISPOSE\_ON\_CLOSE***);

*prijava*.setSize(300,210);

*prijava*.setLocationRelativeTo(**null**);

*vsebina\_prijave*.setLayout(**null**);

*…}*

Koda 8 – Skrajšana predstavitev glavnega razreda programa

V glavni metodi kliče konstruktor glavnega razreda in vsebuje akcijski poslušalec. S klicem konstruktorja nastavimo naše prijavno okno in z akcijskim poslušalcem ob pritisku gumba prijave preverimo vneseno uporabniško ime in geslo. Od akcijskega poslušalca je odvisno ali bomo lahko vstopili v aplikacijo. Akcijski poslušalec sprožimo s pritiskom na gumb na prijavnem oknu in ta izvede poskus vzpostavitve povezave.

**public** **static** **void** main(String[] args) **throws** ClassNotFoundException, SQLException {

//Klic konstruktorja razreda za nastavitev prijavnega okna

**new** MySQL\_Okna();

//Ukaz za prikaz prijavnega okna

*prijava*.setVisible(**true**);

*prijavi\_se*.addActionListener(**new** ActionListener(){

@Override

**public** **void** actionPerformed(ActionEvent e) {

//Zajetje vnesenega uporabniškega imena in gesla

String uporabnik=*prijava\_uporabnik\_vnos*.getText();

String geslo="";

**char**[] geslo\_char=*prijava\_geslo\_vnos*.getPassword();

**for**(**int** z=0;z<geslo\_char.length;z++) {

geslo+=geslo\_char[z];

}

//Če vneseno uporabniško ime in geslo obstajata v PB kličemo konstruktor //razreda za uporabniško aplikacije. V nasprotnem primeru nam je dostop //zavrnjen.

**try** {

Connection povezava=DriverManager.*getConnection*(*url*, uporabnik, geslo); *prijava*.setVisible(**false**);

**new** Generator\_obrazca(povezava, uporabnik);

}

**catch** (Exception g) {

*nepravilno\_u\_ali\_g*.setBounds(20,120,250,25);

*vsebina\_prijave*.add(*nepravilno\_u\_ali\_g*);

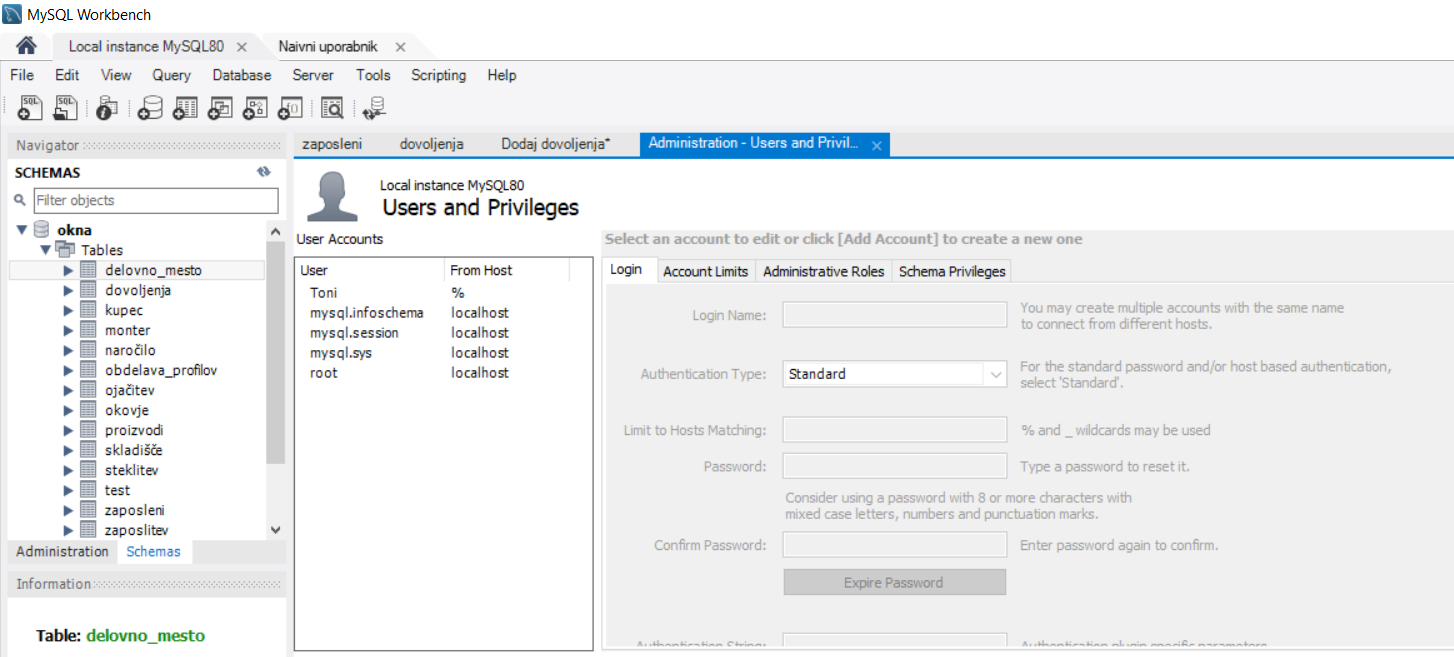
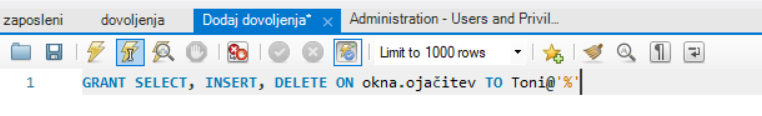
*prijava*.repaint();

}

} });

}

Koda - Prestavitev glavne metode

V MySQL server lahko v podatkovno bazo dodamo več uporabnikov in jim dodeljujemo razna dovoljena gleda na njihovo vlogo pri delu s PB. Ob klicu povezave s podatkovno bazo v parameter metode vstavimo URL naslov, uporabnika in geslo. Povezava se izpostavi, če je bil vnesen pravilen naslov baze podatkov in da ta vsebuje vnesenega uporabnika z geslom.

Slika - Nastavitev novega uporabnika v MySQL Workbench

Za mojo podatkovno bazo imamo vnesena dva uporabnika. Za glavnega administratorja imamo uporabniško ime root in geslo !preCURSOR4de117. Za primer končnega uporabnika imamo uporabnika z imenom Toni in geslom shramba.

V glavni razred so vključen tudi vse metode, ki so neposredno povezane z delom s podatkovno bazo. Te metode kličejo razni dogodki iz razreda z grafičnim vmesnikom in jih lahko razporedimo na tri skupine.

Prva skupina metod izvajala običajne SQL ukaze kot so upravljanje s tabelami in vstavljanje podatkov.

**static** **void** Posodobi\_podatke(Connection povezava, String tabela, String podatki, String ključ) {…}

**static** **void** Izbriši\_podatke(Connection povezava, String tabela, String podatki) {…}

**static** **void** Vstavi\_podatke(Connection povezava, String tabela, String podatki) {…}

**static** **boolean** Naredi\_tabelo(Connection povezava, String izraz) {…}

**static** **boolean** Zbriši\_tabelo(Connection povezava, String tabela) {…}

**static** ArrayList<ArrayList<String>> Naredi\_selekcijo (Connection povezava,

String poizvedba1){…}

Koda 10 - Metode glavnega razreda za klasične SQL ukaze

Druga skupina metod tudi izvajala SQL ukaze vendar so to samo ukazi selekcije, s katerimi želimo pridobiti podatke o samih tabelah podatkovne baze, kot so njeni atributi, primarni ključi, velikost in tako dalje.

**static** ArrayList<String> Pridobi\_atribute(Connection povezava, String tabela) {…}

**static** **int** DobiVelikost(Connection povezava, String tabela, **int** id) {…}

**boolean** JeTujiKljuč(Connection povezava,String tabela, **int** atribut) {…}

**static** String DobiPRimarni(Connection povezava,String tabela) {…}

**static** String[] Poizvedba\_moznosti(Connection povezava,String tabela, String atribut) {…}

Koda 11 - Metode glavnega razreda za podatke o zgradbi tabel

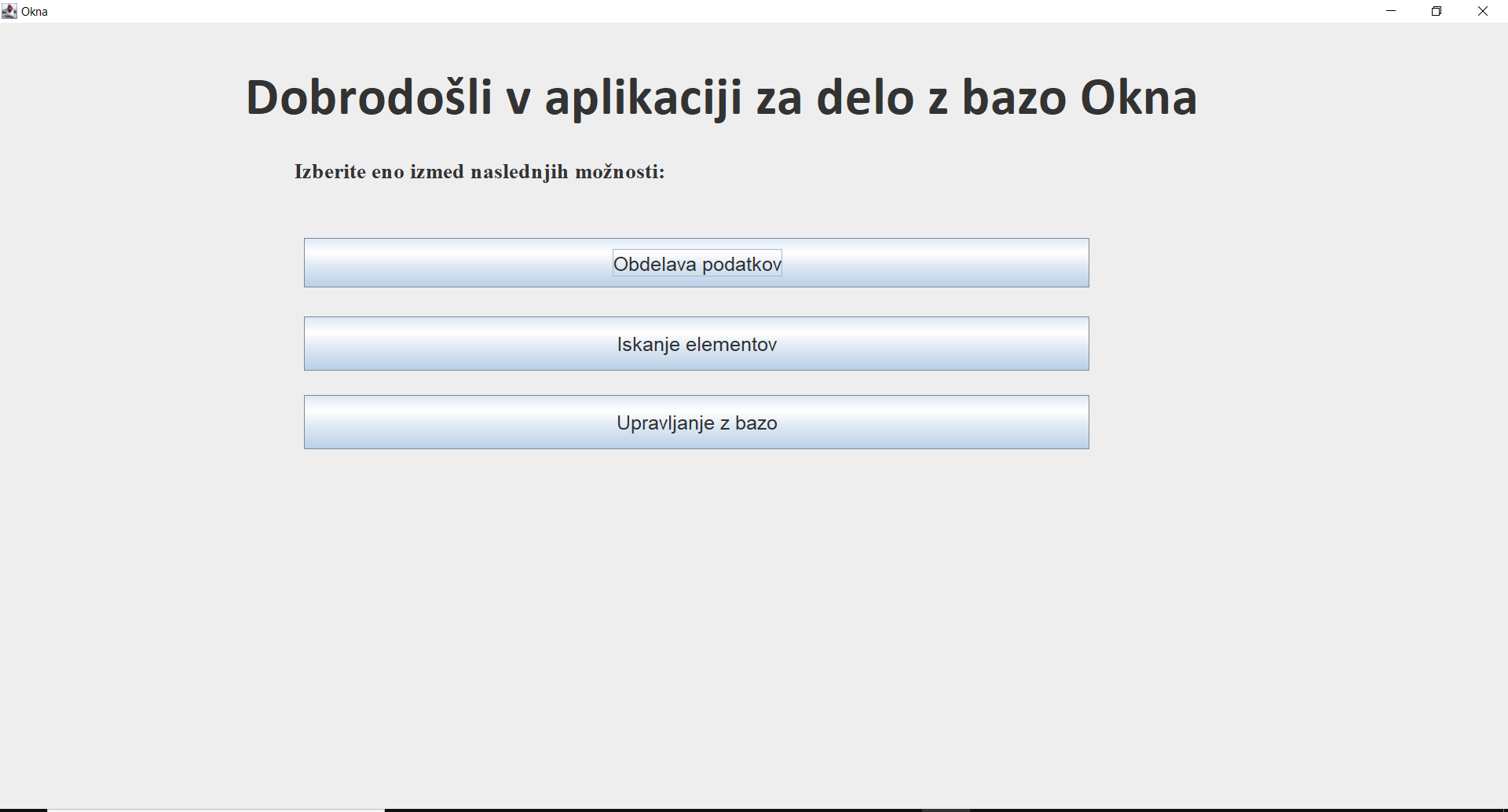
V tretjo skupino spada metoda za zapis vseh storjenih SQL operacij v datoteko, kot sistem varnosti.

**public** **static** **void** Zapisi\_v\_log(String izraz) {...}

Koda 12 - Metode glavnega razreda za zagotavljanje varnosti

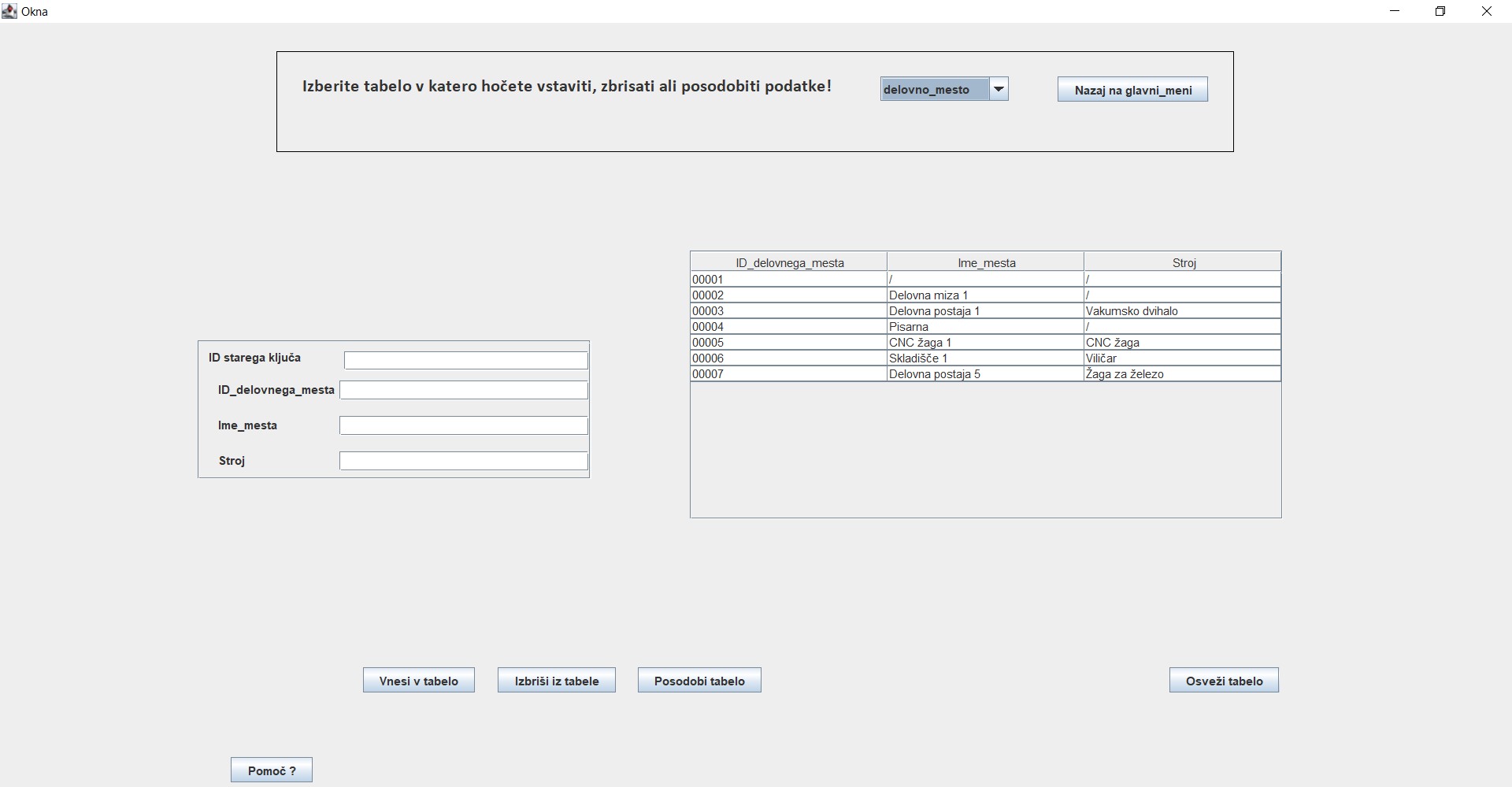
V razredu Generator\_obrazca poteka izgradnja grafičnega vmesnika. Vsebuje številne deklaracije in inicializacije Swingovskih objektov, ki so podane kot spremenljivke razreda. V konstruktorju se tem objektom določijo lastnosti in glede na različne dogodke, kot so pritisk gumba, določi reakcijo. V konstruktorju se nahajava veliko akcijskih poslušalcev, ki v večini omogočajo premike med različnimi okni aplikacije, pripravljajo spremembe vsebine prikazanega okna in sestavljajo različne SQL ukaze za mnoge metode. Izven konstruktorja so metode za spreminjanje vsebine oken in metode, ki pridobivajo potrebne informacije za metode spreminjanja oken.

Sama aplikacija je sestavljena iz petih glavnih panel. Prva panela predstavlja glavni meni za izbiro željene akcije.



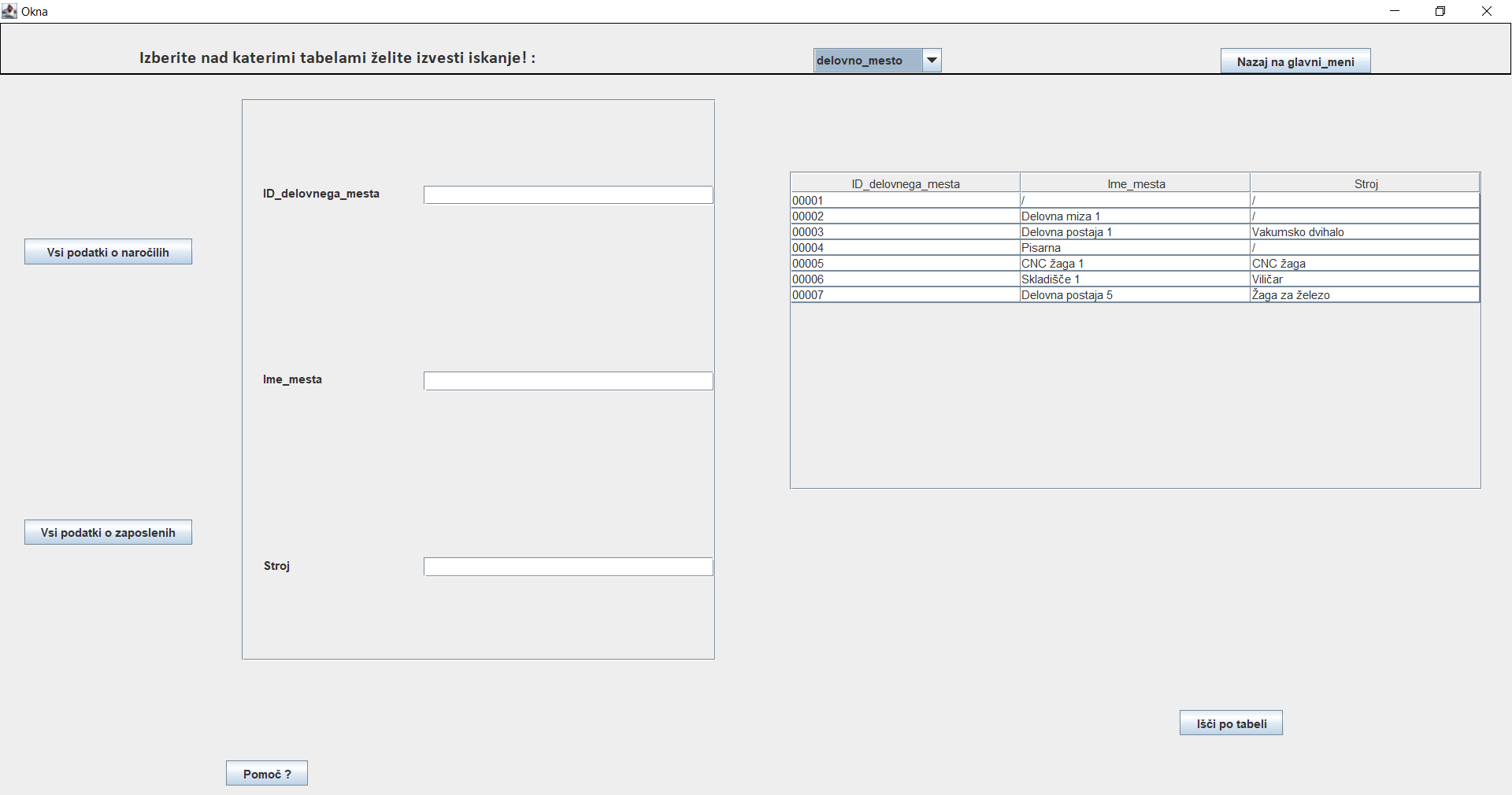
Slika 14 - Glavni meni aplikacije

Prva izbira je obdelava podatkov, v kateri se združujejo funkcije vnosa, zbrisa in posodabljanja podatkov v tabela podatkovne baze. Panela vsebuje seznam za izbiro tabele, okno s prikazom tabele, gumbe za izbiro akcije, pomoč in vrnitev na meni in obrazec za vnos podatkov oziroma parametrov. Glede na izbrano tabelo iz seznama se samodejno spremeni obrazec in prikaz tabele. V obrazec lahko vnašamo nove podatke (če so tuji ključi nam ponudi tudi opcije) ali pa hkrati vnašamo parametre za izbris oziroma posodobitev podatkov. Pri posodobitvi je včasih potrebno vnesti še dodaten parameter starega primarnega ključa, če hočemo zamenjati kakšen primarni ključ.



Slika 15 - Panela za obdelavo podatkov v bazi

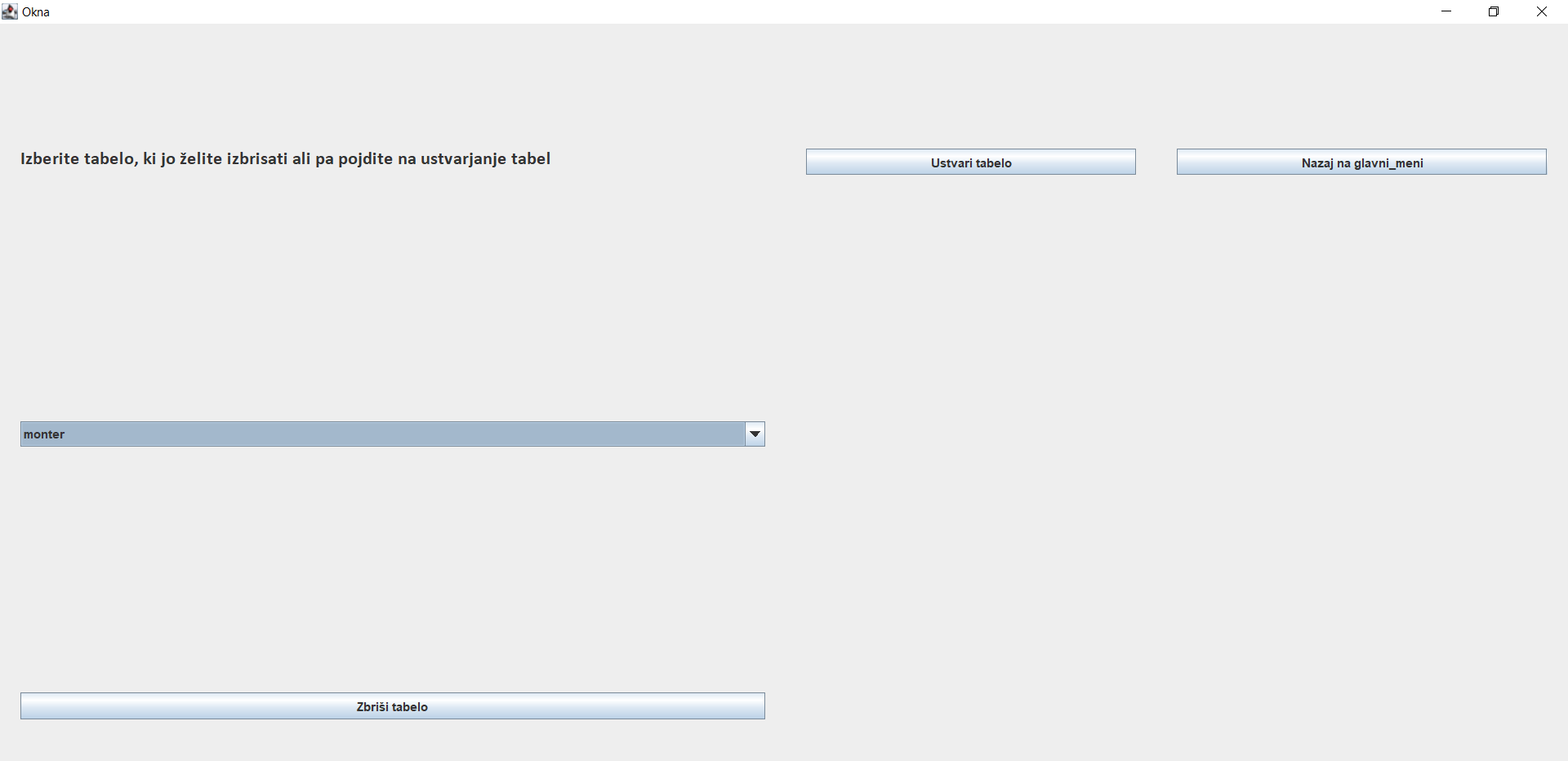
Druga izbira na meniju je iskanje podatkov. Kot pri obdelavi podatkov imamo seznam za izbiro tabele ter prikaz izbrane tabele in obrazec, ki se samodejno posodabljata. V obrazec vnašamo parametre iskanja in z gumbom sprožimo iskanje. Poleg tega imamo tudi nekatere že vnaprej pripravljene poizvedbe, ki pa niso več neodvisni od uporabljene podatkovne baze.



Slika 16 - Panela za iskanje podatkov v PB

Tretja izbira je upravljanje z bazo. Ta del aplikacije ni več namenjen naivnim uporabnikom ampak služi bolj kot skrajna alternativa za kreiranje novih tabel v PB. Vsebuje zmožnost brisanja in kreiranja tabel, vendar mu manjka sposobnost popravljanja zgradbe tabel (primer dodajanje atributov). Ta del aplikacije je bolj eksperimentalne narave, pri katerem se ugotavljal kako bi bilo možno v aplikacijo vpeljati še funkcije ne značilne za končne uporabnike. Ta izbira je na meniju zaščitena še z enim dodatnim dostopnim geslom in je sestavljena iz dveh panel (prva panela za izbris tabel in druga za kreiranje tabel).

Slika, ki vsebuje besede miza

Opis je samodejno ustvarjen

Slika 17 - Paneli za kreiranje in brisanje tabel

Za premikanje iz menija do vseh drugih panel in nazaj sem uporabil CardLayout razred. V razredu se deklarira spremenljivko in se kot način postavitve doda poljubni paneli. Nato lahko na to panelo dodajamo še druge panele skupaj z oznako. Za premik med paneli samo kličemo panele po njihovi določeni oznaki.

//Deklarirano v razredu

JPanel vsebina= **new** JPanel();

CardLayout cl = **new** CardLayout();

…

//V konstruktorju nastavimo poljubno panelo s CardLayout postavitvijo in tej paneli //dodamo druge panele skupaj z oznako.

vsebina.setLayout(cl);

vsebina.add(glavni\_meni,"1"); vsebina.add(obdelujteelemente, "2"); vsebina.add(iščiteelemente, "3"); vsebina.add(upravljajzbazo, "4"); vsebina.add(ustvarjanje\_tabel,"5");

//Za prvo prikazano stran nastavimo glavni meni

cl.show(vsebina, "1");

…

//Primer prehoda med panelami ob stisku gumba

gumb\_v\_obdelavo.addActionListener(**new** ActionListener(){

@Override

**public** **void** actionPerformed(ActionEvent e) {

cl.show(vsebina,"2");}

});

Koda 13 - Uporabljen način prehoda med panelami

Vse panele, ki predstavljajo določene zgoraj predstavljene operacije imajo komponente deklarirane v razredu, medtem pa imajo nastavitve teh komponent znotraj konstruktorja. Glavni akterji na vseh panelih so gumbi, ki ob pritisku vzamejo zapisane podatke v obrazcu in izvedejo določeno operacijo ali pa pripravijo SQL stavek in pokličejo metodo iz glavnega razreda, da ga izvrši.

V primeru stiska gumba za vnos podatkov, se iz obrazca preberejo podatki in se v metodo Vstavi\_podatke pošljejo podatki za vnos v izbrano tabelo. Pri tem postopku imamo tri glavne pogoje. Če imamo v tabeli spremenljivko, ki je tuji ključ, se v metodi izrisa obrazca namesto vnosnega polja naredi selekcijsko polje. Če zanka naleti na tako polje poteče zajem podatkov po malo drugačni metodi. Drugi pogoj sprašuje, če je polje, ki ga beremo, prvo najdeno z vneseno vrednostjo, ker jo to pomemben podatek za sintakso SQL. Znotraj prvih dveh pogojih se zaradi sintakse SQL preverja ali mora biti vneseni podatek število ali nabor znakov.

gumbvnesi\_v\_tabelo.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

//V podatke bode zapisane vse vrednosti, ki jih bomo vstavili v tabelo

String Podatki="";

Boolean prvi\_element=**true**;

**for**(**int** i=0; i < ar\_vnosi.size(); i++) { //Zanka gre skozi vse obrazce vnosa

**try** {

//Spodnji pogoj preverja, če imamo v tabeli spremenljivko, ki je tuji ključ.

**if**(tujikljuc.contains(i)==**true**) {

// Spodnji pogoj preverja, če je imamo polje s prvo najdeno z vneseno vrednostjo

**if**(i!=(ar\_vnosi\_podani.size()-1) || prvi\_element==**true**) {

//Preverjanje ali je vneseno vrednost število

**if**(!preveri\_če\_je\_število(i, povezava, *izbrana\_tabela*))

//Pridobi ID je posebna metoda za zajem podatkov iz seznama tujih

//ključev (glej prvi pogoj).

Podatki+="'"+PridobiID(ar\_vnosi\_podani.get(i))+"',";

**else**

**if**(preveri\_če\_je\_AI(i, povezava, *izbrana\_tabela*)==**true** && PridobiID(ar\_vnosi\_podani.get(i)).isBlank())

Podatki+="null,"; **else** Podatki+=PridobiID(ar\_vnosi\_podani.get(i))+",";

}

**else** { **if**(!preveri\_če\_je\_število(i, povezava, *izbrana\_tabela*)) Podatki+="'"+PridobiID(ar\_vnosi\_podani.get(i))+"'";

…

}**else** {

//Ko nimamo tujega ključa za atribut, namesto klica metode PridobiID, samo //uporabimo get(i).getText().

… }

}**catch**(Exception f) {System.***out***.println(f); }

//Po vnosu prvega podatka iz obrazca spremenimo pogoj prvega elementa

prvi\_element=**false**;

}

//Ko imamo vse podatke za vnos v tabelo kličemo metodo za vnos v tabelo in podamo še //parametre za izbrano tabelo in povezavo v PB.

*Vstavi\_podatke*(povezava, *izbrana\_tabela*, Podatki); *seznam\_tabel1*.actionPerformed(e);

} });

Koda 14 – Primer delovanja gumba za vnos podatkov v tabelo

Vsi gumbi tega tipa na podoben način pripravijo kriterije za SQL ukaze ali pa sami sestavijo in podajo SQL metodam glavnega razreda. Metoda za vnos v tabelo prejme le parametre za SQL ukaz in izvede operacijo vstavljanja podatkov v bazo podatkov.

**static** **void** Vstavi\_podatke(Connection povezava, String tabela, String podatki) {

**try** {

//Priprava ukaza iz prejetih parametrov

String ukaz\_vstavi="INSERT INTO "+tabela+" VALUES("+podatki+");";

//Beleženje akcije je prisoten v vseh metodah glavnega razreda

*Zapisi\_v\_log*(ukaz\_vstavi);

//Priprava in izvršitev ukaza v podatkovni bazi

PreparedStatement izjava=povezava.prepareStatement(ukaz\_vstavi);

izjava.executeUpdate();

}**catch**(SQLException e) {

System.***out***.println("Napaka pri vstavljanju - "+e);

}

}

Koda 15 – Primer izvršitev SQL operacije

Za risanje samega obrazca, v katerega nato vpisujemo podatke ali pogoje imamo svojo metodo. Metoda za risanje obrazca pridobi podatke o vseh atributih tabele, statične grafične objekte definirane v razredu obrazca(tako lahko inicializiramo več primerkov istega tipa grafičnega objekta s pomočjo tabel), panelo na katero rišemo obrazec, nekaj pogojnih vrednosti in povezavo do podatkovne baze. Ta metoda je neodvisna od vsebine podatkovne baze.

**protected** **void** Nariši\_obrazec (ArrayList<JLabel> arnapisov, …) {

//Pred risanjem novih elementov zbriše predhodne in počisti tabele grafičnih objektov

**for**(**int** m=0; m < arnapisov.size(); m++) panela.remove(arnapisov.get(m));

**for**(**int** l=0; l < ar.size(); l++)

panela.remove(ar.get(l));

**for**(**int** k=0;k<ar\_podani.size();k++)

panela.remove(ar\_podani.get(k));

arnapisov.clear(); ar.clear(); ar\_podani.clear(); tujikljuc.clear();

panela.revalidate(); panela.repaint();

//Zanka gre skozi vse atribute v tabeli in dodaja grafične elemente v tabele

**for**(**int** i=0; i < ar\_stolpci.size(); i++) {

//Inicializiramo napis atributa poleg polja za vnos in ga dodamo v tabelo.

atribut= **new** JLabel(ar\_stolpci.get(i));

arnapisov.add(atribut);

//Inicializiramo vnosno polje

obrazec=**new** JTextField(25);

//Ta metoda, deklarirana v razredu aplikacije, se uporablja za izoblikovanje //besedil. V našem primeru služi kot skrbnik, da ne moremo v obrazec pisati več //znakov kot jih ima spremenljivka določena.

AbstractDocument d = (AbstractDocument) obrazec.getDocument();

d.setDocumentFilter(**new** Omeji\_velikost(*DobiVelikost*(povezava,tabela,i)));

ar.add(obrazec); //Polje za vnos dodamo v tabelo

String [] podane\_moznosti; //Ta tabela služi za shranjevanje možnih tujih ključev

//Če je atribut tuji ključ se namesto vnosnega polja doda seznam

**if**(JeTujiKljuč(povezava,tabela,i) && iskanje==**false**) {

//Iz PB pridobimo možne tuje ključe podane\_moznosti=*Poizvedba\_moznosti*(povezava,tabela,ar\_stolpci.get(i));

//Naredimo primerek seznama, ga dodamo v tabelo seznamov in zabeležimo mesto seznama //v obrazcu (tujiključ.add(i);)).

obrazec\_podan=**new** JComboBox<Object>(podane\_moznosti);

ar\_podani.add(obrazec\_podan);

tujikljuc.add(i);

}**else** {

//V nasprotnem primeru v tabelo seznamov in mest seznamov vstavimo »prazne« podatke.

String[] nul= {"n"};

obrazec\_podan=**new** JComboBox<Object>(nul);

ar\_podani.add(obrazec\_podan);

tujikljuc.add(-1);

}panela.repaint();}

//Zanka gre skozi zgoraj dodane grafične elemente v tabelah in jih zapiše na panelo.

**for**(**int** i=0; i < arnapisov.size(); i++) {

gbc.fill = GridBagConstraints.***HORIZONTAL***; gbc.insets = **new** Insets(10,20,10,0);

gbc.gridwidth=1; gbc.gridheight=1; gbc.gridx =x\_kord; gbc.gridy =y\_kord+i;

gbc.weighty=4;gbc.weightx=5;

panela.add(arnapisov.get(i),gbc);

gbc.insets = **new** Insets(5,5,0,0);gbc.gridx=x\_kord+1;

**if**(tujikljuc.contains(i))

panela.add(ar\_podani.get(i),gbc);

**else**

panela.add(ar.get(i),gbc);

}

panela.revalidate(); panela.repaint();

}

Koda 16 – Metoda za izris obrazca na panelo

Na panelo se nariše tudi tabela za vpogled v podatke izbrane tabele. Ta metoda je tudi neodvisna od vsebine podatkovne baze. Za izdelavo grafične tabele se pokličejo metode za zajem podatkov in atributov iz tabele v bazi podatkov.

**protected** **void** Nariši\_tabelo(Connection povezava, String izbrana\_tabela, JPanel panela, JTable tabela) {

//Pripravi SQL ukaz glede na izbrano tabelo

String poizvedba=("SELECT \* FROM "+izbrana\_tabela);

//Kliče metode iz glavnega razreda da pridobi podatke o tabelah PB

ArrayList<ArrayList<String>> ar\_vrstice=*Naredi\_selekcijo*(povezava,poizvedba);

ArrayList <String> ar\_stolpci=*Pridobi\_atribute*(povezava, izbrana\_tabela);

//V prvo tabelo objektov vstavimo podatke o atributih

Object [] stolpci=**new** Object[ar\_stolpci.size()];

**for**(**int** i=0; i < ar\_stolpci.size();i++) {stolpci[i]=ar\_stolpci.get(i);}

**try** {

Object [][]podatki=**new** Object[ar\_vrstice.size()][ar\_vrstice.get(0).size()];

//V drugo tabelo objektov vstavimo same podatke iz tabele PB

**int** z=0;

**for**(**int** i=0; i < ar\_vrstice.size();i++) {

**for**(**int** j=0; j < ar\_vrstice.get(i).size();j++) {

podatki[z][j]=ar\_vrstice.get(i).get(j); } z++;

}

//V statično tabelo deklarirano v razredu aplikacije vstavimo podatke in stolpce.

modeltabele.setDataVector(podatki, stolpci);

//Inicializiramo novo tabelo in nastavimo, da je ni mogoče ročno spreminjati

tabela = **new** JTable(modeltabele) {

@Override

**public** **boolean** isCellEditable(**int** row, **int** column) {

**return** **false**;

}

};

panela.repaint();

}**catch** (Exception e) {System.***out***.println(e);}

}

Koda 17 – Metoda za izris tabele na panelo

Poleg gumbov za premike med paneli, gumbi za sprožanje raznih operacij, obrazcev tabel, napisov in okna za pomoč imajo panele lahko tudi seznam za izbiro tabel. Ti seznami se nahajajo na paneli za obdelavo podatkov, iskanje podatkov in izbris tabele. S pomočjo tega seznama izberemo, katero tabelo želimo obravnavati in se ob izbiri na panelo izriše novi obrazec in tabela z vpogledom v podatke tabele. Ti seznami se napolnijo ob zagonu aplikacije in posodobijo ob brisu in kreiranju tabel.

ArrayList<String> ar=**new** ArrayList<String>();

ar = poišči\_vse\_tabele(ar,povezava); //Metoda v glavnem razredu poišče vse tabele

Object[] tabela\_tabel=ar.toArray();

//Deklarirani grafični seznam v razredu aplikacije inicializiramo s tabelo tabel PB

*seznam\_tabel1*=**new** JComboBox<Object>(tabela\_tabel);

…

//Ob izbiri predmeta iz seznama se aktivira spodnji dogodek

*seznam\_tabel1*.addActionListener(**new** ActionListener(){ @Override

**public** **void** actionPerformed(ActionEvent e) {

**if**(e.getSource()==*seznam\_tabel1*) {

//To so posebni elementi na paneli za obdelavo podatkov

obdelujteelemente.remove(id\_ključa); obdelujteelemente.remove(id\_ključa\_vnos);

//Glede na izbrano tabelo iz seznama kliče metodo za izris obrazca in tabele na //panelo.

*izbrana\_tabela*=(String) *seznam\_tabel1*.getSelectedItem();

ArrayList <String> ar\_stolpci=*Pridobi\_atribute*(povezava, *izbrana\_tabela*);

Nariši\_obrazec(ar\_napisi,ar\_vnosi,ar\_vnosi\_podani,*obrazec\_podan*,*obrazec*,*atribut*,obdelujteelemente\_levo\_vsebina, ar\_stolpci, gbc, 0, 2,**true**,*izbrana\_tabela*,**false**,povezava);

Nariši\_tabelo(povezava, *izbrana\_tabela*, obdelujteelemente\_desno, *tabela\_obdelava*);

gbc.insets = **new** Insets(0,10,0,0); gbc.weighty=0.1; gbc.weightx=0.1;

gbc.gridx =0; gbc.gridy =1; obdelujteelemente\_levo\_vsebina.add(id\_ključa,gbc);

gbc.insets = **new** Insets(10,10,0,0); gbc.gridx =1; gbc.gridy=1;

obdelujteelemente\_levo\_vsebina.add(id\_ključa\_vnos,gbc);

} }

});

Slika 18 – Primer delovanja seznama tabel za izris obrazca in prikaza izbrane tabele iz PB

Večja posebnost v aplikaciji se nahaja na paneli za kreiranje tabele. Na tej paneli imamo opcijo dodajanja in odstranjevanja poljubnega števila atributov (maksimalno 15), ki jih želimo imeti v novi tabeli. Tem atributom lahko določimo ime, podatkovni tip, velikost podatkovnega tipa in lastnosti kot so primarni ključ, tuji ključ, avto inkrementacija in neničelno vrednost.

dodaj\_vrstico.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) { **if**(stevilo\_vrstic<15) {

//Kličemo metodo, ki inicializira grafične objekte iz razreda in jih posamično doda v //tabelo podobnih si gradnikov(tudi deklarirane v razredu aplikacije). Zapolni\_array\_atributov(tabela\_tabel, oznakaatributa,ime\_at, tip\_at, vel\_at, AI\_at, PK\_at, NN\_at, PFK\_at, reference\_tabela);

//Grafične objekte iz tabel riše na podlagi oznake vrstice na kateri se nahajamo

gbc.fill = GridBagConstraints.***NONE***;

gbc.gridy=1+stevilo\_vrstic;

gbc.gridx=0; ustvarjanje\_tabele\_spodaj\_vsebina.add(ar\_oznaka\_at.get(stevilo\_vrstic),gbc);

gbc.gridx=1;

ustvarjanje\_tabele\_spodaj\_vsebina.add(ar\_ime\_at.get(stevilo\_vrstic),gbc);gbc.gridx=2;

ustvarjanje\_tabele\_spodaj\_vsebina.add(ar\_tip\_at.get(stevilo\_vrstic),gbc);gbc.gridx=3;ustvarjanje\_tabele\_spodaj\_vsebina.add(ar\_vel\_at.get(stevilo\_vrstic),gbc);gbc.gridx=4;ustvarjanje\_tabele\_spodaj\_vsebina.add(ar\_AI\_at.get(stevilo\_vrstic),gbc); gbc.gridx=5;

ustvarjanje\_tabele\_spodaj\_vsebina.add(ar\_PK\_at.get(stevilo\_vrstic),gbc); gbc.gridx=6;

ustvarjanje\_tabele\_spodaj\_vsebina.add(ar\_NN\_at.get(stevilo\_vrstic),gbc); gbc.gridx=7

ustvarjanje\_tabele\_spodaj\_vsebina.add(ar\_PFK\_at.get(stevilo\_vrstic),gbc);gbc.gridx=8;ustvarjanje\_tabele\_spodaj\_vsebina.add(ar\_reference\_tabela.get(stevilo\_vrstic),gbc);

ustvarjanje\_tabele\_spodaj\_vsebina.revalidate(); ustvarjanje\_tabele\_spodaj\_vsebina.repaint();

//Ob zapisu vrstice povečamo številko vrstice na kateri smo končali

stevilo\_vrstic++;

}

}

});

Slika 19 - Dodajanje vrstic atributov pri kreiranju tabel

zbriši\_vrstico.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) { **if**(stevilo\_vrstic>0) { **try** {

//Pri brisanju vrstic atributov uporabljamo številko vrstice, da izbrišemo nazadnje //dodane vrstice.

ustvarjanje\_tabele\_spodaj\_vsebina.remove(ar\_oznaka\_at.get(stevilo\_vrstic-1));

ustvarjanje\_tabele\_spodaj\_vsebina.remove(ar\_ime\_at.get(stevilo\_vrstic-1));

ustvarjanje\_tabele\_spodaj\_vsebina.remove(ar\_tip\_at.get(stevilo\_vrstic-1));

ustvarjanje\_tabele\_spodaj\_vsebina.remove(ar\_vel\_at.get(stevilo\_vrstic-1));

ustvarjanje\_tabele\_spodaj\_vsebina.remove(ar\_AI\_at.get(stevilo\_vrstic-1));

ustvarjanje\_tabele\_spodaj\_vsebina.remove(ar\_PK\_at.get(stevilo\_vrstic-1));

ustvarjanje\_tabele\_spodaj\_vsebina.remove(ar\_NN\_at.get(stevilo\_vrstic-1));

ustvarjanje\_tabele\_spodaj\_vsebina.remove(ar\_PFK\_at.get(stevilo\_vrstic-1));

ustvarjanje\_tabele\_spodaj\_vsebina.remove(ar\_reference\_tabela.get(stevilo\_vrstic-1));

//Izbris vrstice pomeni tudi izbris objektov, ki so sestavljali to vrstico, iz tabel

ar\_oznaka\_at.remove(stevilo\_vrstic-1);

ar\_ime\_at.remove(stevilo\_vrstic-1);

ar\_tip\_at.remove(stevilo\_vrstic-1);

ar\_vel\_at.remove(stevilo\_vrstic-1);

ar\_AI\_at.remove(stevilo\_vrstic-1);

ar\_PK\_at.remove(stevilo\_vrstic-1);

ar\_NN\_at.remove(stevilo\_vrstic-1);

ar\_PFK\_at.remove(stevilo\_vrstic-1);

ar\_reference\_tabela.remove(stevilo\_vrstic-1);

ustvarjanje\_tabele\_spodaj\_vsebina.revalidate();

ustvarjanje\_tabele\_spodaj\_vsebina.repaint();

stevilo\_vrstic--;

}**catch** (Exception g) {System.***out***.println(g);}

}

}

});

Slika 20 - Brisanje vrstic atributov pri kreiranju tabel

Sam postopek izdelave tabele je zelo podoben zgoraj vpisanemu opisanemu postopku vnašanju podatkov v tabele. Ob stisku gumba se zapisane vrednosti v obrazcu preberejo in izoblikujejo v celoten SQL ukaz za izdelavo tabele. Ukaz se prenese v metodo v glavnem razredu, ki izvrši ukaz v podatkovni bazi.

Za vse preostale operacije in grafične konstrukte veljajo podobna pravila oziroma principu delovanja, kot so opisani zgoraj. V razredu Generator\_obrazca prevladujejo izrazi za deklaracijo vseh komponent, v konstruktorju se komponentam nastavijo lastnosti in izven konstruktorja so še metode za obdelavo panel v aplikaciji.

Celotna koda se nahaja na povezavi pod prilogo 3.

# Zaključek

Izdelava podatkovne baze zajema šest faz. Najprej je potrebno analizirati zahteve bodočih uporabnikov podatkovne baze in določiti kakšna bo funkcija te podatkovne baze. V drugi fazi naredimo konceptualni model podatkovne baze, pri katerem z diagrami zastavimo načrt podatkovne baze. Naslednji postopek je izbira SUPB. V moji seminarski nalogi sem predhodno izbral MySQL za izdelavo podatkovne baze. V četrti fazi z logičnim načrtovanjem ustvarimo relacijski model, ki ima za razliko od ER modela iz konceptualnega načrtovanja namesto diagramov obliko tabel z definiranimi relacijami. V peti fazi nastopi fizično načrtovanje v katerem izdelamo podatkovno bazo. V zadnji fazi sledi implementacija podatkovne baze. Po načelu teh faz sem izdelal svojo podatkovno bazo v MySQL za uporabo v lokalnem sistemu nepovezanem z internetom.

Pri izdelavi aplikacije sem imel več težav pri oblikovanju aplikacij, kot s samimi SQL operacijami, ki jih po povezavi pošiljamo v bazo podatkov. Ugotovil sem, da je oblikovanje aplikacij z ročnim programiranjem zamudno in bi v prihodnje raje preizkusil dodatne module za grafično oblikovanje aplikacij v javi ali pa bi aplikacij poskusil izdelati v drugem programskem jeziku. Aplikacijo sem skušal zasnovati, tako da bi bila prijazna za uporabnika, vendar bi bilo potrebno narediti izboljšave v izgledu aplikacije in dodatni pomoči. Sem pa z aplikacijo dosegel zadani cilj saj lahko z njo dodajamo, brišemo, spreminjamo in iščemo podatke v PB in celo v sili dodajamo in brišemo tabele iz PB.

V tej seminarski nalogi smo dosegli cilj izdelave podatkovne baze in aplikacije, za nadaljnje izboljšave bi predlagal možnost povezave podatkovne baze z internetom za oddaljen dostop, izboljšanje izgleda aplikacije in vpeljava možnosti izdelave poročil v aplikaciji.

# Zahvala

Zahvalil bi se rad vsem profesorjem računalništva na Šolskem centru Novo mesto, ki so mi s svojim delom v preteklih štirih gimnazijskih letih omogočili priti do te točke, kjer sem zdaj. Za pridobitev znanja potrebnega za izdelavo te seminarske naloge se zahvaljujem profesorju računalništva dr. Albertu Zorku, profesorju za računalniške sisteme in omrežja univ. dipl. inž. Gregorju Medetu, profesorju za računalništvo univ. dipl. inž. Tomažu Ferbežarju in profesorju računalništva Simonu Vovku. Zahvalil bi se rad tudi vsem sošolcem, ki so kakor koli doprinesli k izdelavi te seminarske naloge in staršem za njihovo podporo.

# Bibliografija

1. **Mrhar, Peter.** *Java 2 prvi koraki.* Nova Gorica : Flamingo, 2002. 961-6176-66-8.

2. **Fabjan, Borut in Mesojedec, Uroš.** *Java 2.* Ljubljana : Pasadena, 2004. 961-6361-30-9.

3. **Bilke, Petra.** *Spoznajmo PHP in MySQL.* Nova Gorica : Flamingo, 2002. 961-6176-67-6.

4. **Tutorialspoint.** Eclipse. *tutorialspoint.com.* [Elektronski] Tutorialspoint, 21. september 2012. [Navedeno: 23. marec 2022.] https://www.tutorialspoint.com/eclipse/index.htm.

5. **Vogel, Lars.** Using the Eclipse IDE for Java programming - Tutorial. *vogella.com.* [Elektronski] vogella, 25. oktober 2021. [Navedeno: 24. marec 2022.] https://www.vogella.com/tutorials/Eclipse/article.html.

6. **IBM.** MySQL server overview. *ibm.com.* [Elektronski] IBM Corporation, 8. marec 2021. [Navedeno: 24. marec 2022.] https://www.ibm.com/docs/en/ztpf/1.1.0.15?topic=concepts-mysql-server-overview.

7. **MySQL.** MySQL Workbench Manual / General Information. *dev.mysql.com.* [Elektronski] Oracle Corporation, 11. april 2011. [Navedeno: 24. marec 2022.] https://dev.mysql.com/doc/workbench/en/wb-intro.html.

8. **Javapoint.** MySQL Workbench. *javatpoint.com.* [Elektronski] Javapoint, 6. februar 2020. [Navedeno: 24. marec 2022.] https://www.javatpoint.com/mysql-workbench.

9. **šet, Andreja.** *Načrtovanje in postavitev podatkovnih baz.* Ljubljana : i2, 2017. 978-961-6348-98-0.

10. **Mohorič, Tomaž.** *Podatkovne baze.* Ljubljana : BI-TIM, 2002. 961-6046-12-8.

11. **colos.fri.uni-lj.** Sistem za upravljanje podatkovne baze. *colos.fri.uni-lj.si.* [Elektronski] Fri.uni-lj, 14. april 2007. [Navedeno: 26. marec 2022.] http://colos.fri.uni-lj.si/eri/RACUNALNISTVO/PODATKOVNE\_BAZE/opredelitev\_pojma\_supb.html.

12. **Lončarić, Tea, in drugi.** APLIKACIJE ODJEMALEC - STREŽNIK. *gradiva.txt.si.* [Elektronski] E-računalništvo, 19. oktober 2010. [Navedeno: 2022. marec 17.] https://gradiva.txt.si/racunalnistvo/podatkovne-baze/a\_podatkovne\_baze/08\_aplikacije/.

13. **Taylor, David.** Data Modelling: Conceptual, Logical, Physical Data Model Types. *guru999.com.* [Elektronski] Guru99, 19. februar 2022. [Navedeno: 3. april 2022.] https://www.guru99.com/data-modelling-conceptual-logical.html.

14. **Jones & Bartlett learning.** Database Planning and Database Architecture. *http://samples.jbpub.com/.* [Elektronski] 18. april 2011. [Navedeno: 27. marec 2022.] http://samples.jbpub.com/9781449606008/06008\_ch02\_ricardosec.pdf.

15. **Peterson, Richard.** guru99.com. *ER Diagram: Entity Relationship Diagram Model | DBMS Example.* [Elektronski] 7. oktober 2021. https://www.guru99.com/er-diagram-tutorial-dbms.html.

16. **Tutorialspoint.** ER Model - Basic Concepts. *tutorialspoint.com.* [Elektronski] tutorialspoint, 13. september 2016. [Navedeno: 27. marec 2022.] https://www.tutorialspoint.com/dbms/er\_model\_basic\_concepts.htm.

17. **Glen, Stephanie.** datasciencecentral.com. *Key Attributes in ER Diagrams.* [Elektronski] Data Science Central, 31. julij 2021. [Navedeno: 27. november 2021.] https://www.datasciencecentral.com/profiles/blogs/key-attributes-in-er-diagrams.

18. **Appdynamics.** What is Database Management Systems (DBMS)? *appdynamics.com.* [Elektronski] CISCO, 31. oktober 2020. [Navedeno: 3. april 2022.] https://www.appdynamics.com/topics/database-management-systems.

19. **Mohorič, Tomaž.** *Načrtovanje relacijskih podatkovnih baz.* Ljubljana : BI-TIM, 1997. 961-6046-05-5.

20. **Tutorialspoint.** Relation Data Model. *https://www.tutorialspoint.com/.* [Elektronski] Tutorialspoint, 11. september 2016. [Navedeno: 3. april 2022.] https://www.tutorialspoint.com/dbms/relational\_data\_model.htm.

21. **Chapple, Mike.** The Basics of Database Normalization. *https://www.lifewire.com/.* [Elektronski] Lifewire, 22. februar 2022. [Navedeno: 3. april 2022.] https://www.lifewire.com/database-normalization-basics-1019735.

22. **Microsoft.** Description of the database normalization basics. *docs.microsoft.com.* [Elektronski] Microsoft, 31. marec 2022. [Navedeno: 4. april 2022.] https://docs.microsoft.com/en-us/office/troubleshoot/access/database-normalization-descriptionDescription of the database normalization basics.

23. **DBMS Tools.** Database documentation tools. *https://dbmstools.com/.* [Elektronski] DBMS Tools, 26. april 2017. [Navedeno: 3. april 2022.] https://dbmstools.com/categories/database-documentation-tools.

24. **Greef, Gerhard in Ghoshal, Ranjan.** *Practical E-Manufacturing and Supply Chain Management.* s.l. : Newnes, 2004. 978-0-7506-6272-7.

25. **F. van der Lans, Rick.** *Data Virtualization, Information Management, and Data Governance.* s.l. : Elsevier Inc., 2012. 978-0-12-394425-2.

26. **IBM.** Physical data models. *ibm.com.* [Elektronski] IBM, 8. marec 2021. [Navedeno: 3. april 2022.] https://www.ibm.com/docs/en/ida/9.1.1?topic=modeling-physical-data-models.

27. **Ellingwood, Justin.** 5 ways to host MySQL databases. *https://www.prisma.io/dataguide/mysql/5-ways-to-host-mysql.* [Elektronski] Prisma's Data Guide, 2. maj 2020. [Navedeno: 17. marec 2022.] https://www.prisma.io/dataguide/mysql/5-ways-to-host-mysql.

28. **Richardson, James.** How to Connect to a Database with MySQL Workbench. *inmotionhosting.com.* [Elektronski] 4. november 2021. [Navedeno: 5. april 2022.] https://www.inmotionhosting.com/support/website/connect-database-remotely-mysql-workbench/.

29. **Javatpoint.** BorderLayout (LayoutManagers). *javatpoint.com.* [Elektronski] Javatpoint, 31. maj 2012. [Navedeno: 3. april 2022.] https://www.javatpoint.com/java-layout-manager.

30. —. Event and Listener (Java Event Handling). *javatpoint.com.* [Elektronski] Javatpoint, 31. maj 2012. [Navedeno: 3. april 2022.] https://www.javatpoint.com/event-handling-in-java.

# Stvarno kazalo

D

Dokumentacija, 13, 27

E

ER Model, 8

F

Fizični podatkovni model, 13, 14

J

JDBC(Java DataBase Connectivity), 16

JDK, 2, 16

K

Konceptualni podatkovni model, 8

L

Logični podatkovni model, 10

M

MySQL, 2, 1, 3, 4, 5, 10, 14, 15, 16, 17, 25, 27, 28, 29, 30, 31, 44

MySQL Workbench, 2, 1, 4, 5, 15

N

Normalizacija, 2, 12

P

Podatkovna baza, 2, 6, 10

R

Relacijski model, 2, 11

S

SQL, 3

SUPB, 2, 4, 7, 9, 10, 11, 13, 27, 44

Swing, 2, 18, 19, 29

U

uporabniški vmesnik, 7, 9, 29

# Priloge

Priloga 1

<https://github.com/TomazGril/Seminarska-SQL-ukazi-ustvari.git>

Priloga 2

<https://github.com/TomazGril/Seminarska-SQl-ukazi-zapolni.git>

Priloga 3

<https://github.com/TomazGril/Seminarska-naloga-java-koda.git>

Priloga 4

<https://github.com/TomazGril/Seminarska-naloga.git>

1. Prepovedano brisanje in zapisovanje podatkov na uporabniškem sistemu. [↑](#footnote-ref-1)
2. Java runtime environment – okolje za izvajanje Java programov [↑](#footnote-ref-2)
3. Redundanca podatkov pomeni, da so isti podatki shranjeni večkrat. [↑](#footnote-ref-3)
4. Ni jih mogoče deliti na manjše. [↑](#footnote-ref-4)
5. Atomarnost, Konsistentnost, Izolacija, Trajnost [↑](#footnote-ref-5)
6. Ravnanje s prostorom v glavnem polnilniku in bloki. [↑](#footnote-ref-6)
7. DDL – Data Definition Language [↑](#footnote-ref-7)
8. PHP je skriptni jezik, ki je vključen v kodo HTML [↑](#footnote-ref-8)
9. Za izrisovanje je zadolžen operacijski sistem [↑](#footnote-ref-9)
10. Izvajajo se v spletnem brskalniku in imajo namesto metode main uporabljajo metodo init() [↑](#footnote-ref-10)