

**Prolećni semestar 2019/2020**

*Sistem za biblioteke*

*IT350*

*Baze podataka*

*Projektna dokumentacija*

***Student****:*

Nikola Tasić 3698

***Asistent****:*

Veljko Grković

Sadržaj

[Apstrakt 3](#_Toc30784438)

[Opis problema 3](#_Toc30784439)

[Zahtevani upiti 3](#_Toc30784440)

[Uvod 4](#_Toc30784441)

[Cilj 4](#_Toc30784442)

[Realizacija 4](#_Toc30784443)

[Projektno rešenje 5](#_Toc30784444)

[Konceptualni model 5](#_Toc30784445)

[Fizički model 6](#_Toc30784446)

[Kreiranje baze 7](#_Toc30784447)

[Generisanje C koda 8](#_Toc30784448)

[Parsiranje DDL fajlova 8](#_Toc30784449)

[C Header fajl 9](#_Toc30784450)

[C Source fajl 10](#_Toc30784451)

# Apstrakt

## Opis problema

Opis problema: Državna uprava je odlučila da im je potreba aplikacija za potrebe svih državnih biblioteka. Aplikacija mora da čuva podatke o svim bibliotekama po opštinama u kojima se nalaze. Za svaku opštinu može se zaključiti kom regionu pripada i dobiti naziv regiona. Naziv opštine kojoj biblioteka pripada predstavlja deo kompletne adrese biblioteke. Svaka adresa ima i broj zgrade i ime ulice u kojoj se biblioteka nalazi. Postoji evidencija o zaposlenim u bibliotekama, svaki zaposleni ima ime, prezime, poziciju i biblioteku u kojoj radi. Moguće pozicije su bibliotekar, čistač i direktor. Postoji evidencija o knjigama. Svaka knjiga ima autore, ime, godinu izdanja i ISBN. Svaki autor ima ime, prezime i opis. Jedna knjiga može da ima više autora. Takođe jedan naslov knjige može da ima više primeraka. Postoje korisnici biblioteke. Svaki korisnik ima korisničko ime i šifru. Jedna knjiga može da bude iznajmljena od strane jednog korisnika ali svaki korisnik može da ima i do dve iznajmljene knjige. Potrebno je voditi evidenciju koji korisnik je iznajmio koju knjigu.

## Zahtevani upiti

1. Broj zaposlenih, kao i broj knjiga po biblioteci i gradu
2. Autore koji imaju preko 10 objavljenih knjiga od 2000. godine do trenutnog datuma
3. Adrese svih biblioteka gde se može naći pet ili više knjiga autora “Joshua Bloch” 12 e. Uraditi statistički prikaz, na dnevnom nivou koliko je knjiga u kojoj biblioteci iznajmljeno, a koliko vraćeno.
4. Za biblioteku u kojoj ima najveći broj različitih naslova (ne knjiga/primeraka) prikazati adresu, grad i ukupan broj zaposlenih. Ako postoje dve ili više biblioteke sa istim, najvećim brojem knjiga prikazati podatke za sve takve biblioteke.
5. Napisati pogled koji za svaki naslov prikazuje broj primeraka koji se nalaze u biblioteci. Prikazati samo one naslove koji imaju više od jednog autora. Statistiku uraditi samo za biblioteke koje se nalaze u Sremskom okrugu.
6. Prikazati naslov i autora knjige koji su kao nove pristigle u biblioteku. Prikazane knjige su u biblioteku došle one kalendarske godine kada je prikaz napravljen (upit izvršen). Uraditi prikaz samo za biblioteku koja ima najveći broj korisnika. Knjige sortirati po naslovu i godini izdanja.

# Uvod

## Cilj

Cilj projekta je da se dizajnira sistem koji pruža uvid u poslovanje biblioteka u jednoj državi i omogućava pregled svih knjiga i njihovih autora zajedno sa informacijama da li su te knjige trenutno izdate na čitanje nekom od korisnika.

## Realizacija

Baza podataka će biti napravljena za **MariaDB** koja je trenutna open-source implementacija **MySQL**-a za Linux i Windows sisteme. Dizajniranje baze će biti odrađeno preko **Powerdesigner**-a.

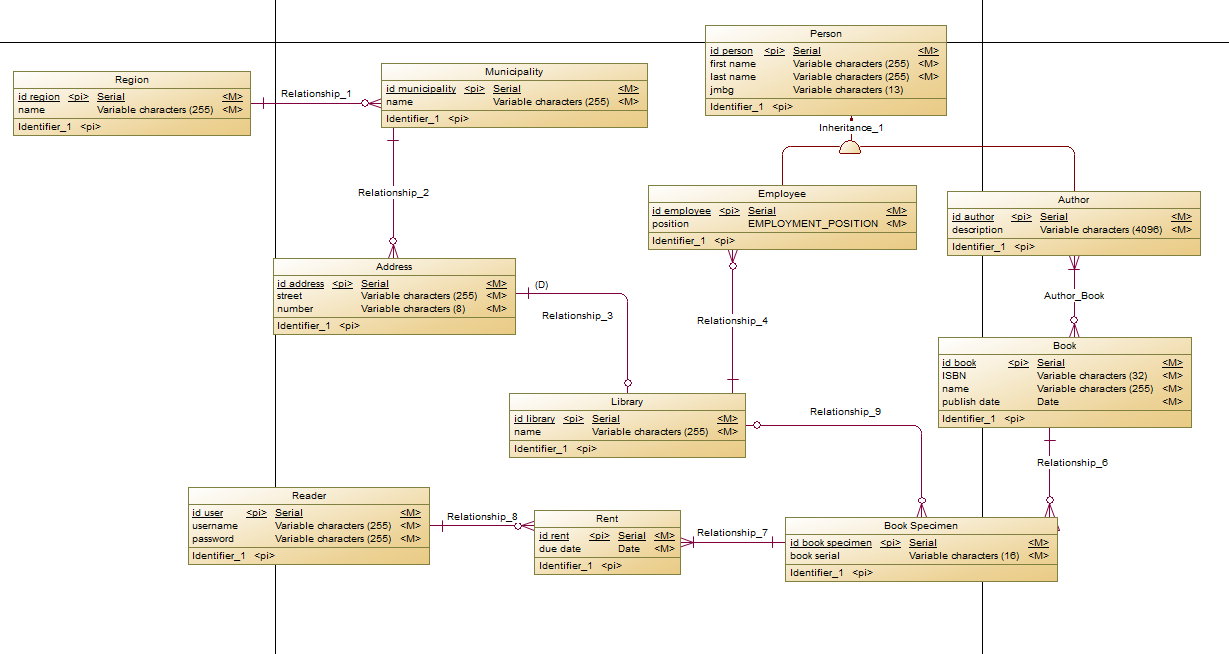
CRUD aplikacija će biti **CLI** (command-line) aplikacija izrađena u programskom jeziku C samo iz pomoć odgovarajuće biblioteke koja je u ovom slučaju **libmysql**. Lični dodatak na ovu problematiku je da će sav kod koji obradjuje logiku CRUD operacija nad bazom biti generisan automatski na osnovu DDL fajla koji predsavlja strukturu baze. Generisanje C koda će izvršiti **Python** program koji je zaduzen za struktuiranje **DDL** opisa u odgovarajuce C header i source fajlove. Među sloj između Python aplikacije i DDL-a je **NodeJS** skripta koji dati DDL čisti od komentara i parsira u **JSON** fajl koji kasnije obrađujemo. Za dizajn CLI interfejsa koristimo dobro poznatu C biblioteku **libncurses.**

# Projektno rešenje

Sledi opis projektnog rešenja od šeme baze do implementacije i primera korišćenja aplikacije.

## Konceptualni model

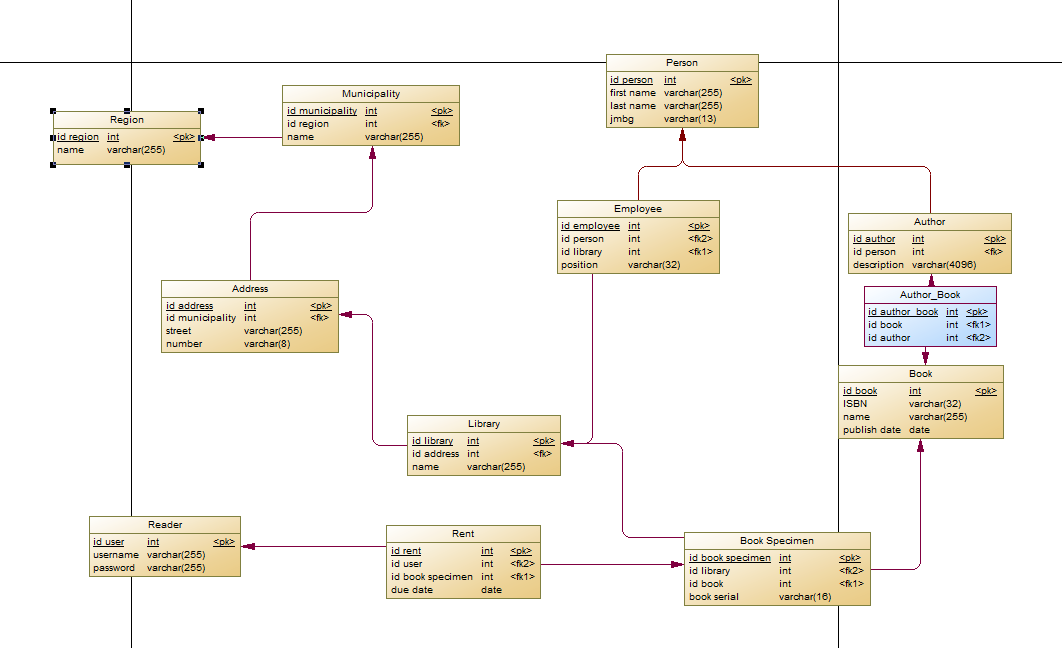
Konceptualni model predstavlja prvi nivo apstrakcije problema dobijenog iz apstrakta.



Slika 1 Konceptualni model baze podataka

## Fizički model

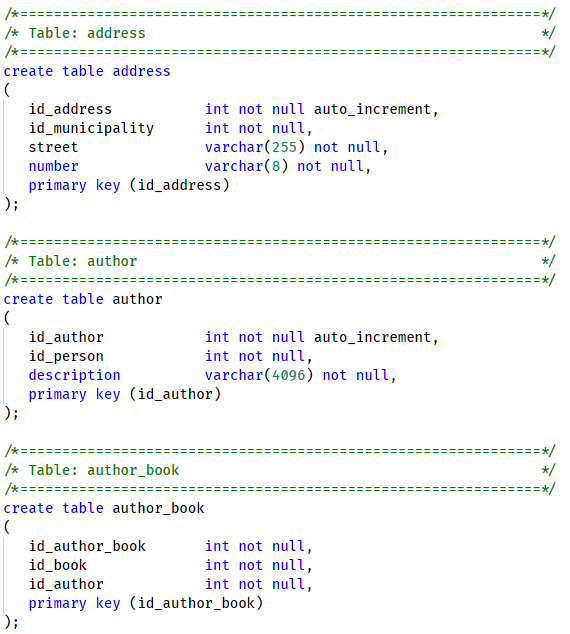
Na osnovu konceptualnog modela možemo da generišemo fizićki model koji je jedan korak bliži generisanju SQL DDL koda koji ćemo koristiti za kreairanje baze.



Slika 2 Fizički model baze podataka

## Kreiranje baze

Na kraju iz fizičkog modela možemo da generišemo SQL kod koji ćemo da koristimo za kreiranje.



Slika 3 Primer generisanog DDL koda

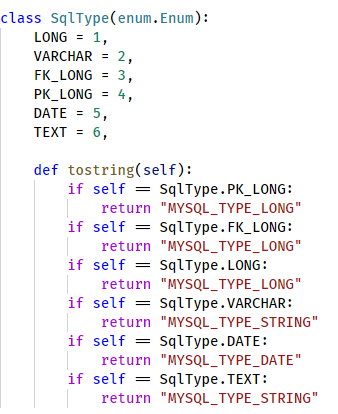
Iz ovako generisanog SQL fajla mozemo veoma lako kreirati bazu i sve odgovarajuce tabele **source** komandom.

## Generisanje C koda

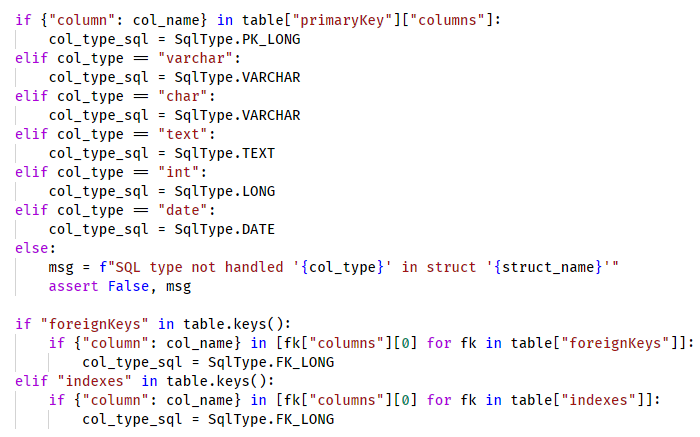
Kao sto je već pomenuto sav kog koji predstavlja logiku vršenja upita nad bazom podataka biće dinamički generisan. Ovakav pristup simulira neku vrstu ORM frejmvorka a i takođe olaksava sam proces jer je kod potreban za izvršavanje upita veoma repetativan.

### Parsiranje DDL fajlova

Na osnovu DDL-a i libmysql dokumentacije mozemo prilično lako da definemo konverziju tipova ova dva veoma različita sistema.



Slika 4 Enum reprezentacija DDL tipova

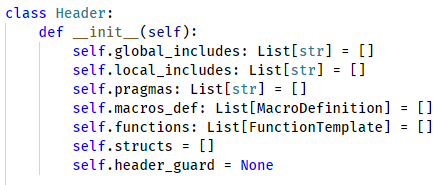


Slika 5 Parsiranje JSON-a u enum

U slici broj 5 vidimo da naravno nisu svi tipovi podrzani ali mozemo lako da dodajemo nove. U tome nam pomaze **assert** koji se okida ako naiđemo na neki tip koji nije podržan. Kada se kod izvrši bez ijednog asserta onda znači da smo implementirali svaku potrebnu metodu. Ovakav pristup je verovatno lakse implementirati preko neke vrste interfejsa ali rezultati parsiranja JSON-a i kod koji je potreban za izvršavanje svih upita ne moze baš tako lako da se generalizuje. Ovakav pristup pročešljavanja grešaka je bio najlakši za implementaciju.

### C Header fajl

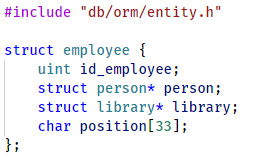
Iz enuma je lako generisati C header fajl.



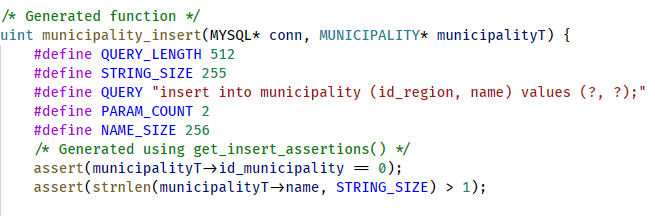
Slika 6 Python reprezentacija C header fajla

Ovakva struktura nam omogucuje da dinamički dodajemo sve makroe koji su nam potrebni da tipa definisemo header guardove i importove (#include) ostalih header fajlova koji su nam potrebni za komunikaciju. Takodje suštinski deo svakog header fajla su funkcije koje isto sa lakoćom mozemo dodati preko klase **FunctionTemplate**.

Rezulat jedne ovakve klase je na primier ovakav C header fajl. Svaka od FK veza iz DDL-a je predstavljena kao pointer do nove strukture dok su ostala polja parsirana u odgovarajuće C tipove. Obraćamo paznju na to da svako tekstualno polje ima +1 veću dužinu od definisanog. To je potrebno da bi smo uključili u obzir *'0' (null)* karakter za terminaciju stringa.

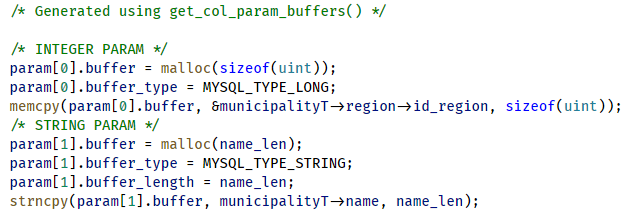


### C Source fajl

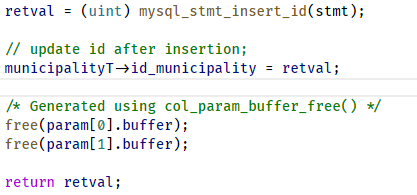


Slika 7 Primer C source fajla i metode za INSERT

Naravno u header fajl dolazi C izvorni fajl koji sadrži sve osnovne metode za CRUD operacije.



Slika 8 Primer pripreme buffera koji izvršava upit



Slika 9 Oslobadjanje memorije buffera i azuriranje ID-a

# Rešenja zahtevanih upita

## Upit *a*

*Broj zaposlenih, kao i broj knjiga po biblioteci i gradu*

select library.name, municipality.name as 'Municipality', count(book\_specimen.id\_book\_specimen) as 'Books'

from book\_specimen

         join library on library.id\_library = book\_specimen.id\_library

         join address on address.id\_address = library.id\_address

         join municipality on address.id\_municipality = municipality.id\_municipality

group by library.id\_library;

select library.name, municipality.name as 'Municipality', count(employee.id\_employee) as 'Employees'

from employee

         join library on library.id\_library = employee.id\_library

         join address on address.id\_address = library.id\_address

         join municipality on address.id\_municipality = municipality.id\_municipality

group by library.id\_library;

## Upit *b*

*Autore koji imaju preko 10 objavljenih knjiga od 2000. godine do trenutnog datuma*

select book\_authors.id\_author, person.first\_name, person.last\_name, `Books Published`

from (select author.id\_author, author.id\_person, count(books.id\_book) as `Books Published`

      from (select id\_author, author\_book.id\_book, publish\_date

            from author\_book

                     join book b on author\_book.id\_book = b.id\_book

            where (b.publish\_date between cast(2000-01-01' as date) and curdate())) as `books`

               join author on books.id\_author = author.id\_author

      group by author.id\_author) as book\_authors

         join person on person.id\_person = book\_authors.id\_person

where `Books Published` > 10;

## Upit *c*

*Adrese svih biblioteka gde se može naći pet ili više knjiga autora “Joshua Bloch”*

select `Count`, library.name, street, number

from (select book\_specimen.id\_book\_specimen,

             book\_specimen.id\_library,

             one\_author.id\_book,

             one\_author.id\_author,

             count(book\_specimen.id\_book\_specimen) as `Count`

      from (select book.id\_book, ab.id\_author

            from book

                     join author\_book ab on book.id\_book = ab.id\_book

                     join author a on ab.id\_author = a.id\_author) as one\_author

               join book\_specimen on one\_author.id\_book = book\_specimen.id\_book

      group by one\_author.id\_book) as libraries

         join library on libraries.id\_library = library.id\_library

         join address a2 on library.id\_address = a2.id\_address

         join author on author.id\_author = libraries.id\_author

         join person on person.id\_person = author.id\_person

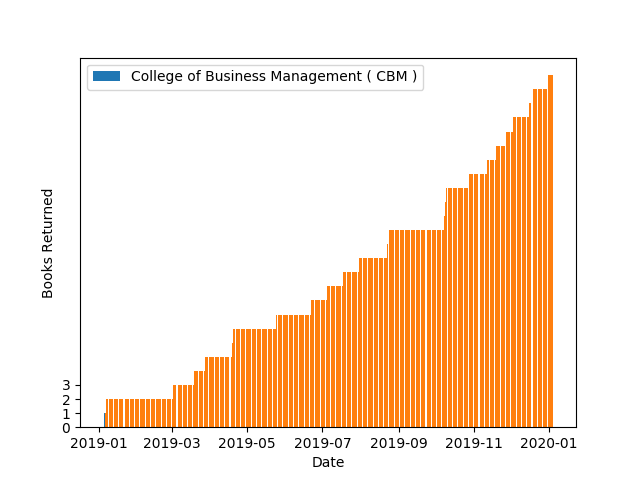
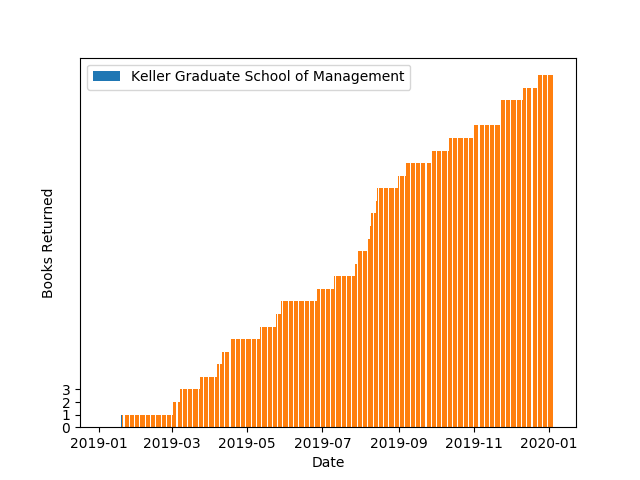
where person.first\_name = 'Joshua'

  and person.last\_name = 'Bloch'

  and `Count` >= 5;

## Upit *d*

*Uraditi statistički prikaz, na dnevnom nivou koliko je knjiga u kojoj biblioteci iznajmljeno, a koliko vraćeno.*



Obzirom na to da se trazio statisticki prikaz rešenje je odrađeno uz pomoć Python biblioteke **matplotlib.**

## Upit *e*

*Za biblioteku u kojoj ima najveći broj različitih naslova (ne knjiga/primeraka) prikazati adresu, grad i ukupan broj zaposlenih. Ako postoje dve ili više biblioteke sa istim, najvećim brojem knjiga prikazati podatke za sve takve biblioteke.*

select top\_libraries.id\_library,

       name,

       street,

       number,

       count(top\_libraries.id\_library) as `No. Employees`,

       `Title Count`

from (select id\_library, name, street, number, max(`Title Count`) as `Title Count`

      from (select \*, count(id\_library) as `Title Count`

            from (select library.id\_library, library.name

                  from library

                           join book\_specimen bs on library.id\_library = bs.id\_library

                           join book b on bs.id\_book = b.id\_book

                  group by isbn, library.id\_library

                  order by library.id\_library) as unique\_books

            group by name

            order by `Title Count` desc) as libraries

               join address on address.id\_address = libraries.id\_library) as top\_libraries

         join employee on top\_libraries.id\_library = employee.id\_library

group by employee.id\_library;

## Upit *f*

*Napisati pogled koji za svaki naslov prikazuje broj primeraka koji se nalaze u biblioteci. Prikazati samo one naslove koji imaju više od jednog autora. Statistiku uraditi samo za biblioteke koje se nalaze u Sremskom okrugu.*

create view `Books Per Library` as

select id\_library, books3.name as `Book Name`, isbn, `Library Name`, id\_book, `Specimen Count`, id\_address

from (select \*, count(isbn) as `Specimen Count`

      from (select \*

            from (select library.id\_library,

                         library.id\_address,

                         library.name as `Library Name`,

                         id\_book\_specimen,

                         bs.id\_book   as idbook

                  from library

                           join book\_specimen bs on library.id\_library = bs.id\_library

                           join book b on bs.id\_book = b.id\_book) as books) as books2

               join book b on b.id\_book = books2.idbook

      group by isbn) as books3;

select `Book Name`, isbn, `Library Name`, `Specimen Count`, `Authors`, m.name as `Municipality Name`

from (select id\_library, `Book Name`, isbn, `Library Name`, `Specimen Count`, count(id\_author) as `Authors`, id\_address

      from `Books Per Library`

               join author\_book ab on `Books Per Library`.id\_book = ab.id\_book

      group by `Book Name`) as `Books and Authors`

         join address on address.id\_address = `Books and Authors`.id\_address

         join municipality m on address.id\_municipality = m.id\_municipality

where `Authors` > 1

  and m.name = 'Sremski okrug';

## Upit *g*

*Prikazati naslov i autora knjige koji su kao nove pristigle u biblioteku. Prikazane knjige su u biblioteku došle one kalendarske godine kada je prikaz napravljen (upit izvršen). Uraditi prikaz samo za biblioteku koja ima najveći broj korisnika. Knjige sortirati po naslovu i godini izdanja.*

select id\_library, `Library Name`, `Readers`

from (select reader.id\_reader, library.id\_library, library.name as `Library Name`, count(reader.id\_reader) as 'Readers'

      from reader

               join rent on reader.id\_reader = rent.id\_reader

               join book\_specimen bs on bs.id\_book\_specimen = rent.id\_book\_specimen

               join library on library.id\_library = bs.id\_library

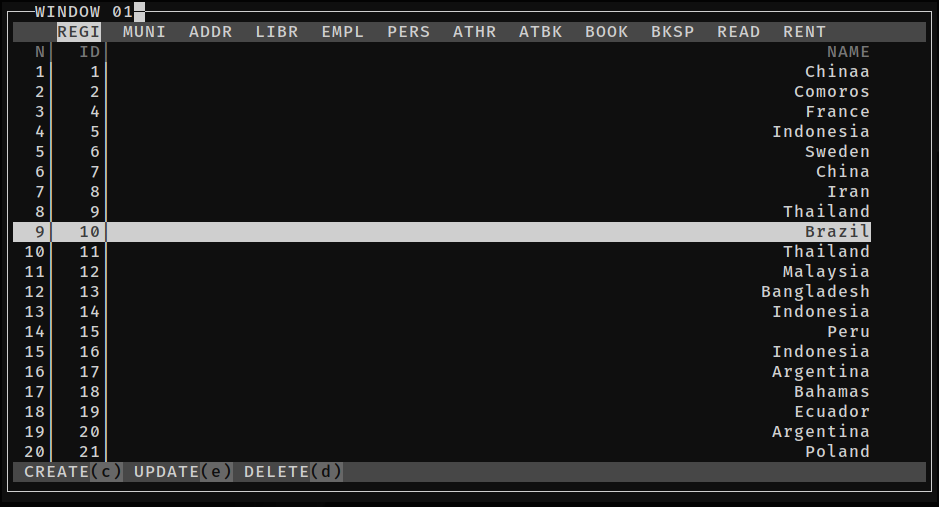
      group by library.id\_library) as `Readers per Library`

group by id\_library order by `Readers` desc;

# Aplikacija

U ovom delu cemo napravili osnovni pregled izgelda i funkcionalnosti CRUD aplikacije. Aplikaciju pokrecemo iy komandne linije pomocu *./it350\_pz\_app*

## List view (READ)



Slika 10 Izgled osnovnog list view-a

U ovom view-u imamo pregled svih tabela iz baze do kojih mozemo da navigiramo pomoću strelica levo i desno dok selekciju vršimo strelicama gore i dole. Na dnu prozora se nalazi informacije koje komande korsitimo za CRUD operacije. Pritiskom na dugme 'L' otvaramo novi identični List View koji može da nam posluži ako zelimo da napravimo brzu promenu u nekoj drugoj tabeli.

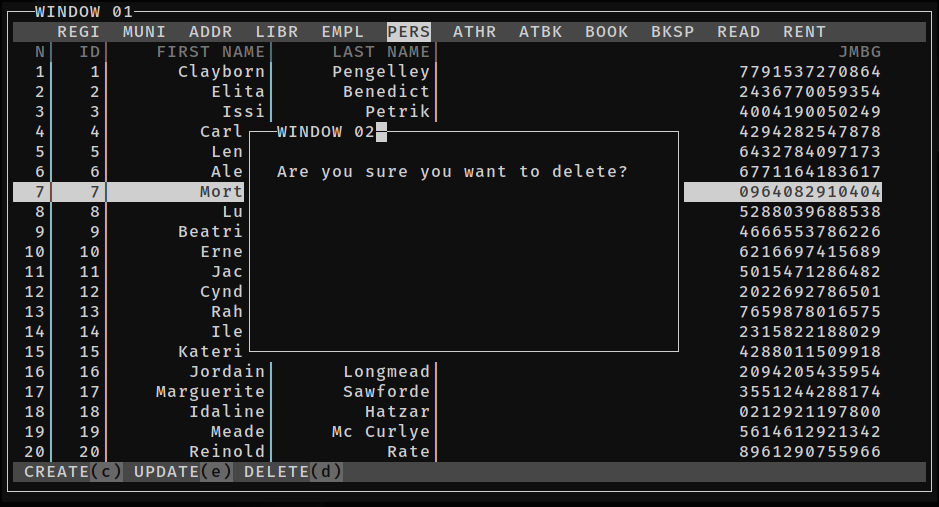
## Form view (UPDATE)



Slika 11 Primer UPDATE interfejsa

Posle popunjavanja podataka koristimo CTRL+X da sačuvamo podatke (po uzoru na *nano* editor) ili CTRL+D (po uzoru na logout prečicu u linux terminalima) da bi se vratili na početni list view.

## Popup view (DELETE)



Slika 12 Primer popup-a za brisanje podatka

Pritiskom na Y ili N odgovaramo na prosto pitanje.

## Form view (CREATE)



Slika 13 Slčan primer kao kod UPDATE forme

# Zakljucak

Projekat realizuje osnovne CRUD operacije nad definisanom bazom podataka. Akcenat je naravno bačen na automatsko generisanje koda i parsiranje DDL-a što uvodi malo kompleksnosti u inače trivijalan i monoton posao dizajniranje CRUD aplikacije. Dizajn baze je bio takav da se što je više moguće smanje redundantnost podatka što je i postignutno. Sama aplikacija naravno je veoma bazična ali daje dobar proof-of-concept za gore pomenutu automatsko generisanje i parsiranje.

# Bibliografija

Duarte, A. F. (n.d.). *SQL DDL to JSON Schema*. Retrieved from npm: https://www.npmjs.com/package/sql-ddl-to-json-schema

Guittet, T. (n.d.). *fields\_magic.c*. Retrieved from GitHub: https://gist.github.com/alan-mushi/c8a6f34d1df18574f643

Hall, J. (n.d.). *Getting started ncurses*. Retrieved from LinuxJournal: https://www.linuxjournal.com/content/getting-started-ncurses

*MYSQL C API Reference*. (n.d.). Retrieved from MYSQL: https://dev.mysql.com/doc/refman/8.0/en/c-api-function-overview.html

Padala, P. (n.d.). *NCURSES programming HOWTO*. Retrieved from tldp: http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/