VISOKA TEHNIČKA ŠKOLA
STRUKOVNIH STUDIJA

E-biblioteka

Predmet: Administriranje baza podataka

Profesor:
dr Dušan Stefanović

Studenti:
Marijana Stanisavljević REr 10/17
Filip Stojanović REr 56/17
Goran Đukić REr 38/17
Kristina Manić SEr 52/17

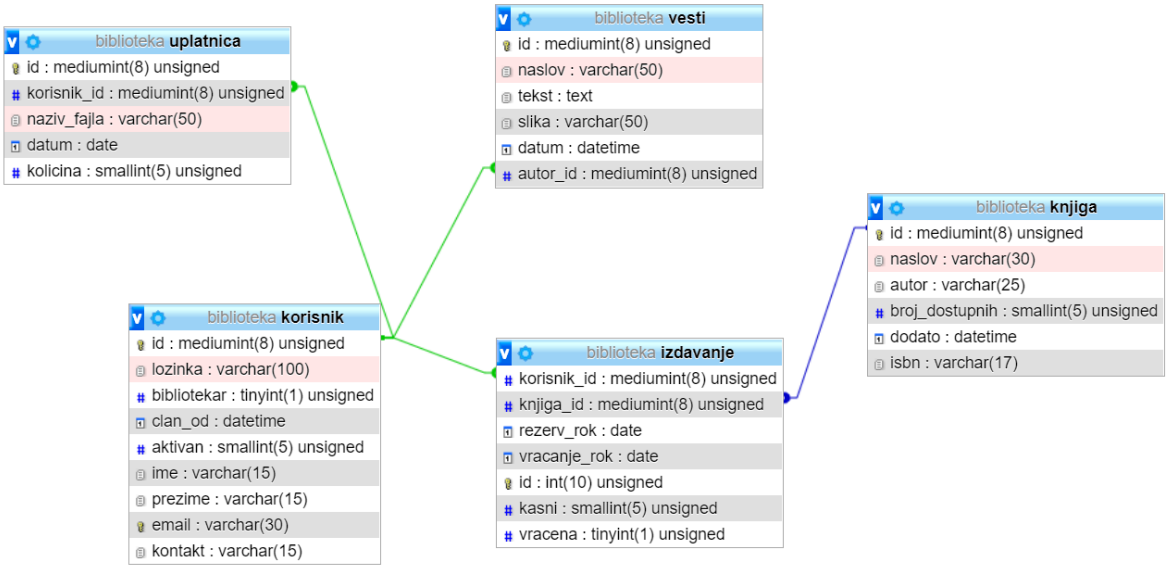
12.6.2020.

SADRŽAJ

Uvod

Link ka projektu: https://github.com/fistmedia/E-biblioteka

Projekat E-biblioteka je web aplikacija čiji je cilj da olakša rad biblioteke.
Na klijentskom delu su korišćeni HTML5, Bootstrap 4, CSS3, jQuery i Jinja templejtski jezik, dok su na backendu korišćeni Flask i MySQL baza podataka.
Postoje dve role: administrator odnosno bibliotekar i običan korisnik biblioteke.
Bibliotekar ima pristup sledećim podacima: listi knjiga, listi korisnika, podaci o korisnicima kao i knjigama koje su zadužili, podaci o uplatama i može zabraniti ili odobriti pristup korisnika određenim delovima aplikacije, može dodavati knjige, može dodavati, brisati, ažurirati i kreirati vesti, zadužiti i razdužiti knjige, pretraživati knjige i korisnike.
Korisnik se prvo registruje i dostavlja dokaz o uplati preko web stranice da bi mu administrator odobrio pristup korisničkim funkcijama aplikacije.
Kada mu je pristup odobren korisnik može pretražiti dostupne knjige i može videti knjige koje je zadužio kao i rok za povratak. Takođe može poslati još dokaza o uplati da bi produžio svoju članarinu.



Baza podataka se sastoji iz pet tabela. Korisnik tabela kao primarni ključ ima id koji je auto increment, a povezana je sa tri druge tabele iz baze stranim ključem: izdavanje, vesti i uplatnica.
Tabela Izdavanje je povezana sa druge dve tabele: sa korisnikom preko id tabele korisnik i sa knjiga tabelom preko id kolone iz tabele knjiga. Ne koristmo kompozitni primarni ključ, jer isti korisnik može iznajmiti istu knjigu više puta pa koristimo autoincrement ključ id.

Sposobnosti baze

Indeksi

Indeksi omogućavaju brzo pretraživanje podataka po određenoj tabeli ali usporavaju pisanje u bazu.
Tehnički svaki primarni i strani ključ je podrazumevano indeksiran, kolone koje smo mi indeksirali su:
-korisnik.email, postavljen je kao UNIQUE key, i koristi se u pretrazi korisnika
-knjiga.naslov, koristi se u pretrazi knjiga po naslovu
-knijga.autor, koristi se u pretrazi knjiga po imenu autora
-izdavanje.vracena, koristi se za sortiranje izdatih knjiga korisnika

Triggeri

Triggeri su SQL iskazi koji se izvršavaju pre ili posle INSERT, UPDATE i DELETE upita
Omogućavaju rad sa podacima nakon upisa, proveru sa prethodnim podacima, ...

Mi smo ih postavli u tabelama izdavanje i uplatnica:

U tabeli izdavanje:

izdavanje

```
CREATE TRIGGER izdavanje
AFTER INSERT
ON izdavanje FOR EACH ROW
BEGIN
    UPDATE knjiga
    SET broj_dostupnih = broj_dostupnih - 1
    WHERE id = NEW.knjiga_id;
END
```

Ovaj trigger nakon INSERT upita smanjuje broj dostupnih primerka knjige tipa koje je izdata

vracanje

```
CREATE TRIGGER vracanje
BEFORE UPDATE
ON izdavanje FOR EACH ROW
BEGIN
    IF (NEW.vracena <> OLD.vracena AND NEW.vracena = 1) THEN
        UPDATE knjiga
        SET broj_dostupnih = broj_dostupnih + 1
        WHERE id = NEW.knjiga_id;
    END IF;
END
```

Ovaj trigger nakon UPDATE upita koji stavlja da je knjiga vraćena povećava broj dostupnih primerka vraćene knjige

U tabeli uplatnice:

-dodaj_clanarinu

```
CREATE TRIGGER `dodaj_clanarinu` BEFORE UPDATE ON uplatnica FOR EACH ROW
    BEGIN
        IF (NEW.kolicina <=> OLD.kolicina) THEN
            UPDATE korisnik
            SET aktivan = aktivan + 365 * NEW.kolicina / 500
            WHERE id = NEW.korisnik_id;
        END IF;
    END
```

Ovaj trigger nakon UPDATE upita tj nakon što bibliotekar odobri uplatu za članarinu i unese koliko je član uplatio dodaje godišnju članarinu članu

Eventovi

Eventovi su događaji u bazi nakon kojih se izvršava neki iskaz.
Kod nas oni se dešavaju jednom dnevno u 1 časova ujutru i to su:

-clanarina

```
CREATE EVENT clanarina
ON SCHEDULE
  EVERY 1 DAY_HOUR
  COMMENT 'Uzni clanarinu svakog dana'
  DO
    UPDATE korisnik
    SET aktivan = aktivan - 1
    WHERE aktivan > 0;

CREATE EVENT izdavanje
ON SCHEDULE
```

Ovaj događaj dnevno oduzima 1 dan članarine svakom članu, sve dok ne budu imali 0 dana članarine preostalih

- izdavanje

```
CREATE EVENT izdavanje
ON SCHEDULE
  EVERY 1 DAY_HOUR
  COMMENT 'Broj dana koje nisu knijge vracane posle roka'
  DO
    UPDATE izdavanje
    SET kasni = kasni + 1
    WHERE vracena = 0 AND CURDATE() > vracanje_rok;
```

Ovaj događaj svakog dana povećava broj dana kašnjena izdate knjige ukoliko ona nije vraćana u roku

Procedure

Najveći CRUD upiti bi trebalo da budu procedure u bazi koje se pozivaju zarad povećanja performansi baze podataka jer se onda ne šalje ceo upit i pamti se optimizacija upita,
Kod nas to su:

- korisnik_knjige

```
DELIMITER $$
CREATE PROCEDURE korisnik_knjige (IN korisnik mediumint(8) unsigned)
BEGIN
    SELECT autor, naslov, isbn, vracanje_rok
    FROM izdavanje LEFT JOIN knjiga
    ON knjiga.id = izdavanje.knjiga_id
    WHERE korisnik_id = korisnik AND vracena = 0;
END $$
DELIMITER ;
```

Ova procedura nalazi informacije o knjigama koje je određeni korisnik iznajmio ali nije vratio

- jedan_korisnik

```
DELIMITER $$
CREATE PROCEDURE jedan_korisnik (IN korisnik mediumint(8) unsigned)
BEGIN
    SELECT izdavanje.id, autor, naslov, isbn, vracanje_rok, vracena
    FROM izdavanje LEFT JOIN knjiga
    ON knjiga.id = izdavanje.knjiga_id
    WHERE korisnik_id = korisnik
    ORDER BY vracena;
END $$
DELIMITER ;
```

Ova procedura nalazi informacije o svim knjigama koje je određeni korisnik iznajmio i sortira ih po tome da li su vraćene da bi bibliotekar imao listu.

- admin_uplata

```
DELIMITER $$
CREATE PROCEDURE admin_uplata()
BEGIN
    SELECT ime, email, uplatnica.id, naziv_fajla
    FROM uplatnica LEFT JOIN korisnik
    ON korisnik.id = uplatnica.korisnik_id
    WHERE kolicina IS NULL;
END $$
DELIMITER ;
```

Ova procedura nalazi sve uplate koje bibliotekar još uvek nije odobrio