PANEVROPSKI UNIVERZITET APEIRON
FAKULTET INFORMACIONIH TEHNOLOGIJA
BANJA LUKA

Seminarski rad

ONLINE REZERVACIJA KARATA

Nastavni predmet: ***Baze podataka***

Predmetni nastavnik:
Dražen Marinković

Student:
Siniša Božić
192-20/RITP

Banja Luka, 2021.

SADRŽAJ

UVOD 1
1 ONLINE REZERVACIJA KARATA..... 2
2 ER DIJAGRAM 3
3 INTEGRITETI 5
4 KORIŠTENJE FUNKCIJA..... 6
5 KORIŠTENJE POGLEDA 7
6 USKLADIŠTENE PROCEDURE..... 9
7 TRIGERI 10
8 UDF FUNKCIJE 13
9 USPOSTAVLJANJE INDEKSA 14
ZAKLJUČAK 15
LITERATURA 16

UVOD

U predmetnom seminarskom radu je obrađen problem online rezervacije avionskih karata, odnosno stvorena baza podataka sa pripadajućim tabelama i relacijama. Za DBMS je izabran MySQL, kao popularni open-source sistem upravljanja bazama podataka. Treba napomenuti se faza projektovanja i dizajna same baze ne tiče tipa softverskog rješenja, dok u procesu implementacije treba imati u vidu konkretan softver odnosno njegove specifičnosti. U tom smislu, ukoliko govorimo o relacionim bazama podataka, postoje određene razlike u MySQL u odnosu na Microsoft i Oracle baze podataka, a radi se uglavnom o sintaksnim pa i nekim semantičkim razlikama. Dizajn baze predstavlja vjerovatno polovinu posla, jer se u toj fazi planira logični izgled i povezanost tabela, određuju kolone odnosno nosioci podataka i njihovi pripadajući tipovi podataka, te kako će podaci biti povezani u različitim tabelama. Tu se primjenjuje čitav niz koncepata kao što su normalizacija podataka, integritet baze, imenovanje entiteta i kolona itd.

U konkretnom slučaju je kreirana baza podataka u MySQL Workbench 8.0 Community Edition razvojnom okruženju, sa pripadajućim tabelama, relacijama između njih, te prema zahtjevima su kreirani ER dijagram, proizvoljni korisničko-definisani pogledi i funkcije, uskladištene procedure i trigeri. Tabele su popunjene sa po 10 unosa kako bi se moglo manipulisati podacima i realizovati upite. Na tako kreiranoj bazi je moguće, pored navedenih dodatnih mogućnosti, realizovati uobičajene upite selekcije podataka, sortiranja, spajanja, ubacivanja, brisanja redova i kolona i to je jedan pojednostavljeni model sistema koji se koristi u avionskom saobraćaju. Naravno pravi sistemi su mnogo složeniji i sastoje se od više desetina i stotina tabela, pa su tamo potrebni višestruke instance entiteta kao što su avion, aerodrom, metodi plaćanja i sl. U svakom slučaju baza podataka će omogućiti automatizovano upravljanje podacima ne samo u jednoj tabeli već na nivou kompletne baze jer će radnje nad njima da budu sinhronizovane u skladu sa definisanim procedurama.

Ovako stvoren sistem je zapravo back-end odnosno pozadinski dio jednog softverskog rješenja i kada mu se doda front-end dio odnosno aplikativni interfejs kojem pristupaju korisnici različitih ovlaštenja (od krajnjih kupaca odnosno klijenata avio-kompanija do zaposlenika i administratora), dobija se kompletan softverski proizvod. Baza podataka vjerovatno čini bar 50% mogućnosti i kompleksnosti kako ovog tako i velike većine svih drugih softverskih sistema iza kojih redovno stoje baze podataka kao glavni nosioci aktivnosti manipulacije podataka.

1 ONLINE REZERVACIJA KARATA

Rezervacija avionskih karata prije pojave baza podataka je bila manuelni posao koji se sastojao od evidentiranja svih subjekata (u stručnom žargonu entiteta) kao što su letovi, putnici, aerodromi, pojedinačni avioni itd., te njihovog upisivanja u tabele, ručnog povezivanja i ažuriranja. U prošlosti su to vjerovatno radile poluautomatizovane mašine koje datiraju računarskom dobu. Pojavom sistema za upravljanje bazama podataka ovakve evidencije se mogu razviti na personalnim računarima.

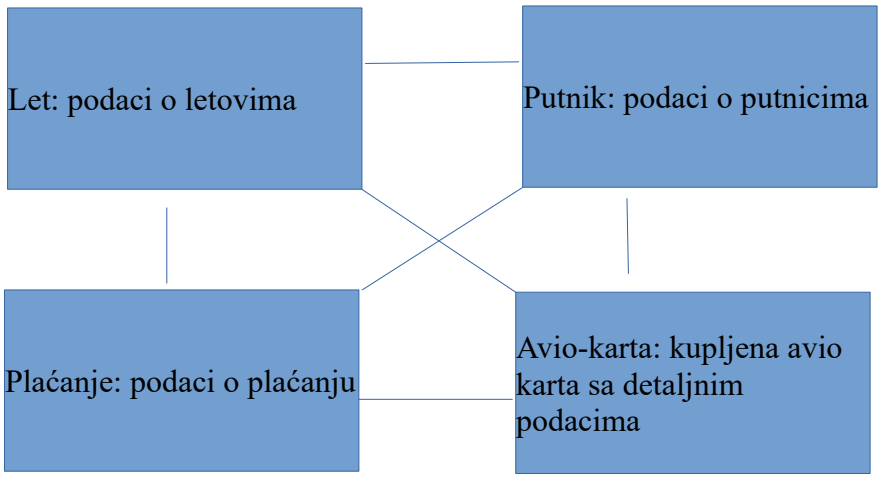
U konkretnom slučaju, predmetna baza podataka je pojednostavljen model koji se sastoji od nekoliko tabela koje su povezane određenim relacijama. Modelovani su slijedeći entiteti:

- letovi,
- avio karte,
- putnici,
- plaćanje (bankovna kartica).

Tabele su povezane relacijama između primarnih i stranih ključeva. Podaci su normalizovani u skladu sa pravilima normalizacije[1], koji obuhvataju tri najznačajnija pravila:

- 1NF – kaže da u tabelama ne može biti podataka istovjetnog tipa koji se ponavljaju,
- 2NF – podaci u tabelama trebaju da budu zavisni to jest identifikovani definisanim ključem,
- 3NF – podaci u tabelama bi trebali da ne budu redundantni – odnosno u tabelama po pravilu ne bi trebalo biti ponavljajućih kolona.

Dizajn baze podataka naravno se ne odnosi na bilo koju konkretnu tehnologiju i ne tiče se izbora određenog DBMS i to može biti, u početnoj fazi, najobičnija skica odnosno crtež:

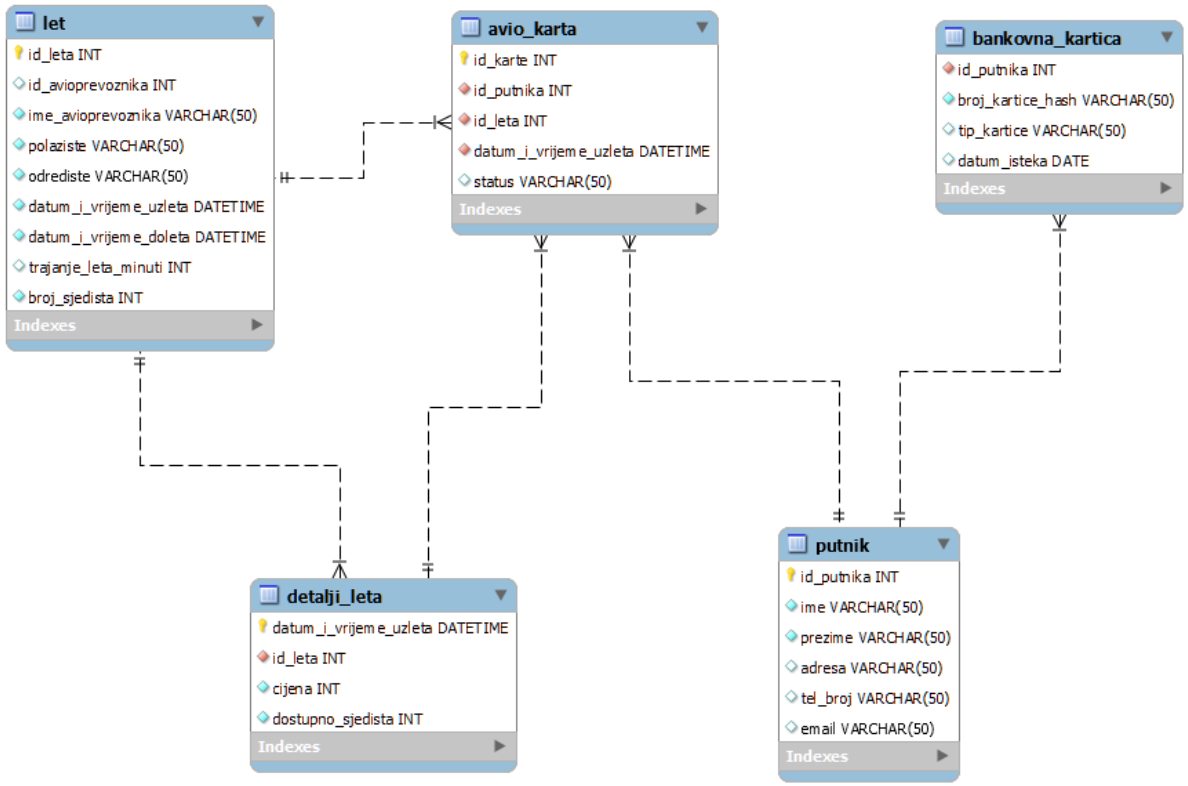


Prilog 1: Početna skica modela

1 Za detalje pogledati: https://www.studytonight.com/dbms/database-normalization.php

2 ER DIJAGRAM

ER dijagram (eng. Entity Relationship Diagram) je tip dijagrama toka koji prikazuje odnose između tzv. entiteta kao što su ljudi, objekti ili koncepti, ideje. Kod baza podataka upotrebljavaju se da bi sa logičkog nivoa prikazali datu bazu podataka, njene entitete kao i relacije između entiteta. Za razliku od obične skice koji smo prezentovali, ER dijagram sadrži detaljne podatke o strukturi objekata i o tipu veze odnosno relacija između njih. ER dijagram naše baze podataka izgleda ovako:



Prilog 2: ER dijagram baze podataka 'Online rezervacija karata'

Glavni entiteti u bazi podataka su let, putnik, avionska karta i bankovna kartica (pretpostavka je da je to jedini način plaćanja). Budući da različiti letovi mogu imati različite polaske, shodno pravilima normalizacije kreirana je nova tabela 'detalji_leta' koja sadrži podatke o cijeni i broju sjedišta za svaki pojedinačni let.

Tabele su povezane na osnovu slijedećih relacija:

- Private Key (PK) id_leta u tabeli 'let' je Foreign Key (FK) u tabelama 'avio_karta' i 'detalji_leta',
- PK datum_i_vrijeme_uzleta u tabeli 'avio_karta' je FK u tabeli 'detalji_leta',

- PK id_putnika u tabeli putnik je FK u tabelama 'avio_karta' i 'bankovna_kartica'.

Struktura entiteta je slijedeća:

Tabela let	
Naziv kolone	Tip podatka
id_leta	INT (cjelobrojni), PK
id_avioprevoznika	VARCHAR(50) (karakteri varijabilne dužine)
polaziste	VARCHAR(50)
odrediste	VARCHAR(50)
datum_i_vrijeme_uzleta	DATETIME (datum i vrijeme)
datum_i_vrijeme_doleta	DATETIME
trajanje_leta_minuti	INT
broj_sjedista	INT

Tabela 1: Prikaz tabele 'let'

Tabela avio_karta	
Naziv kolone	Tip podatka
id_karte	INT, PK
id_putnika	INT, FK
id_leta	INT, FK
datum_i_vrijeme_uzleta	DATETIME, FK
status	VARCHAR(50)

Tabela 2: Prikaz tabele 'avio_karta'

Tabela bankovna_kartica	
Naziv kolone	Tip podatka
id_putnika	INT, FK
broj_kartice_hash	VARCHAR(50)
tip_kartice	VARCHAR(50)
datum_isteka	DATE

Tabela 3: Prikaz tabele 'bankovna_kartica'

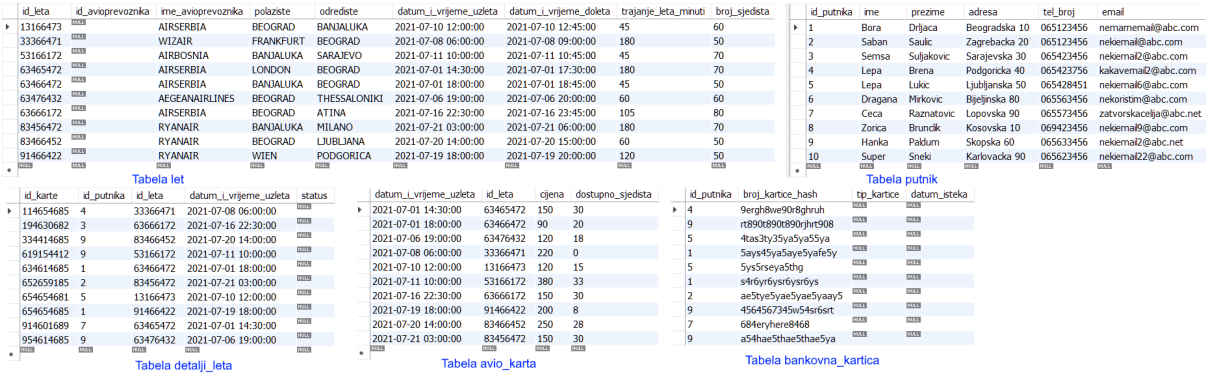
Tabela detalji_leta	
Naziv kolone	Tip podatka
datum_i_vrijeme_uzleta	DATETIME, PK
id_leta	INT, FK
cijena	INT

dostupno_sjedista	INT

Tabela 4: Prikaz tabele 'detalji_leta'

Tabela putnik	
Naziv kolone	Tip podatka
id_putnika	INT, PK
ime	VARCHAR(50)
prezime	VARCHAR(50)
adresa	VARCHAR(50)
tel_broj	VARCHAR(50)
email	VARCHAR(50)

Tabela 5: Prikaz tabele 'Putnik'



Prilog 3: Prikaz unesenih podataka

3 INTEGRITETI

Analiziraćemo integritete entiteta, domena i referencijalnosti. **Integritet entiteta** osigurava da je svaki zapis u tabeli jedinstven i da ima primarni ključ koji nije NULL. To znači da u tabeli ne postoji duplirani zapis, te da je svaki zapis jedinstveno određen atributom koji nije NULL. Na primjer, u tabeli let svaki let bi se trebao razlikovati od drugih i ne smije biti dupliranih zapia, što se postiže kolonom id_leta koja je primarni ključ koja jednoznačno određuje svaki red u ovoj tabeli.

Kada je u pitanju **integritet domena**, radi se o tome da se u kolone koje su označene određenim tipom podataka unose predviđeni tipovi podataka, odnosno u INT kolone se ne unose karakteri već numerički podaci, u kolone DATE(TIME) se unose podaci sa vremenskom odrednicom odnosno datum (i vrijeme), a u CHAR označene kolone (odnosno podvarijante VARCHAR) se unose karakteri sa određenom maksimalnom dužinom. Ako npr.

ne koristimo DATETIME tip podatka već umjesto toga INT, to ne bi bio odgovarajući tip podatka za takve unose i unosom novih redova bi sigurno nastale kolizije jer se podaci ili ne bi unosili jednoobrazno ili bi bili u smislu vremenske odrednice neupotrebljivi.

Referentni integritet se odnosi na povezanost podataka u tabelama i kaže da svaki strani ključ (FK) u jednoj tabeli treba da bude primarni ključ (PK) u drugoj tabeli[2]. Ukoliko to nije slučaj, onda se podaci ne mogu referencirati preko više tabela i baza podataka gubi svrhu jer nije povezana a to onda ima za posljedicu da se upiti ne mogu vršiti na više tabela i tako dalje. U predmetnoj bazi referentni integritet je postignut tako što su primarni ključevi u respektivnim tabelama istovremeno strani ključevi u povezanim tabelama tako da je moguće referencirati podatke u više tabela, shodno određenim kriterijumima. Npr. PK id_leta u tabeli 'let' je FK u tabelama 'avio_karta' i 'detalji_leta', što omogućava jednoznačno identifikovanje podataka u ovako povezanim tabelama (npr. komandom JOIN je moguće po osnovu ovih kriterijuma združiti podatke). Tabela 'avio_karta' ima tri FK, odnosno referencirana je tabelama 'let', 'detalji_leta' i 'putnik'. Više FK može biti referisano ka jednom PK, odnosno jedna tabela može sadržati jedan ili više FK, ili FK u više tabela mogu upućivati ka jednom PK, ali unutar tabele može postojati samo jedan PK.

4 KORIŠTENJE FUNKCIJA

Funkcije predstavljaju predefinisane mogućnosti koje rješavaju neki problem. U svakom DBMS postoje desetine pa i stotine ugrađenih funkcija a u najkorištenije vjerovatno spadaju MIN, MAX, COUNT, AVG i SUM pa ćemo u nastavku dati njihove primjere na datoj bazi podataka.

Sintaksa ovih funkcija glasi:

```
SELECT MIN(column_name)
FROM table_name
WHERE condition;

SELECT MAX(column_name)
FROM table_name
WHERE condition;

SELECT COUNT(column_name)
FROM table_name
WHERE condition;

SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

2 Za detalje pogledati: https://www.databasejournal.com/features/mysql/article.php/2248101/Referential-Integrity-in-MySQL.htm

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

Na primjeru u datoj bazi primjeri ovih funkcija mogu da budu:

```
select min(trajanje_leta_minuti) from let
where broj_sjedista between 80 and 100;
```

(ispisuje minimum iz kolone trajanje_leta_minuti iz tabele let, uz dodatni uslov da je vrijednost iz kolone broj_sjedista između 80 i 100),

```
select max(cijena) from detalji_leta
where datum_i_vrijeme_uzleta > '2021-07-15 00:00:00';
```

(ispisuje maksimum iz kolone cijena iz tabele let, uz dodatni uslov da je vrijednost iz kolone datum_i_vrijeme_uzleta veće od 15.07.2021),

```
select count(id_karte) from avio_karta
where id_putnika=9;
```

(ispisuje broj unosa iz kolone id_karte iz tabele avio_karta gdje je vrijednost kolone id_putnika jednaka broju 9),

```
select avg (cijena) from detalji_leta;
```

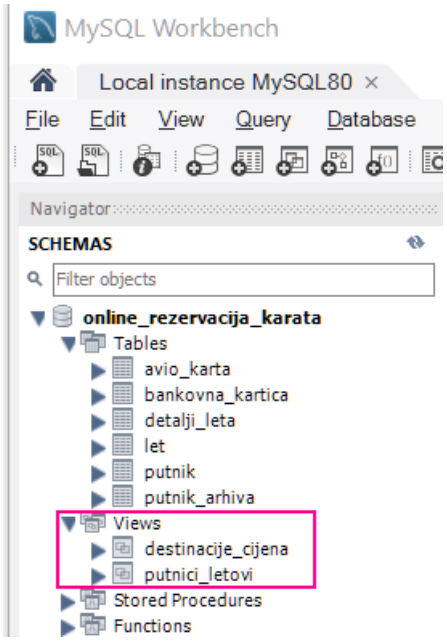
(ispisuje prosjek vrijednosti iz kolone cijena iz tabele detalji_leta),

```
select sum(broj_sjedista) from let
where ime_avioprevoznika = 'AIRSERBIA' and polaziste = 'BEOGRAD'
```

(ispisuje sumu vrijednosti kolone broj_sjedista iz tabele let, uz pripadajući dodatni uslov).

5 KORIŠTENJE POGLEDA

Pogledi su alternativni načini prikaza tabela koje korisnik može da predefiniše, shodno unesenim upitima. Na primjer, u predmetnoj bazi smo definisali dva pogleda:



Prilog 4: Definisani pogledi

Pogled destinacije_cijena omogućava pregled baze podataka putem memorisanog upita, kao da je u pitanju zasebna tabela:

polaziste	odrediste	cijena	dostupno_sjedista
BANJALUKA	BEOGRAD	90	20
BEOGRAD	THESSALONIKI	120	18
BEOGRAD	BANJALUKA	120	15
LONDON	BEOGRAD	150	30
BEOGRAD	ATINA	150	30
BANJALUKA	MILANO	150	30
WIEN	PODGORICA	200	8
FRANKFURT	BEOGRAD	220	0
BEOGRAD	LJUBLJANA	250	28
BANJALUKA	SARAJEVO	380	33

Prilog 5: Izgled pogleda 'destinacije_cijena'

Sintaksa za ovaj upit je slijedeća:

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `destinacije_cijena` AS
    SELECT
        `l`.`polaziste` AS `polaziste`,
        `l`.`odrediste` AS `odrediste`,
        `d`.`cijena` AS `cijena`,
        `d`.`dostupno_sjedista` AS `dostupno_sjedista`
    FROM
        (`let` `l`
        JOIN `detalji_leta` `d` ON ((`l`.`id_leta` = `d`.`id_leta`)))
    ORDER BY `d`.`cijena`
```

Pogled putnici_letovi omogućava pregled putnika po letovima, kao što slijedi:

ime	prezime	datum_i_vrijeme_doleta	polaziste	odrediste
Lepa	Brena	2021-07-08 09:00:00	FRANKFURT	BEOGRAD
Semsa	Suljakovic	2021-07-16 23:45:00	BEOGRAD	ATINA
Hanka	Paldum	2021-07-20 15:00:00	BEOGRAD	LJUBLJANA
Hanka	Paldum	2021-07-11 10:45:00	BANJALUKA	SARAJEVO
Bora	Drljaca	2021-07-01 18:45:00	BANJALUKA	BEOGRAD
Saban	Saulic	2021-07-21 06:00:00	BANJALUKA	MILANO
Lepa	Lukic	2021-07-10 12:45:00	BEOGRAD	BANJALUKA
Bora	Drljaca	2021-07-19 20:00:00	WIEN	PODGORICA
Ceca	Raznatovic	2021-07-01 17:30:00	LONDON	BEOGRAD
Hanka	Paldum	2021-07-06 20:00:00	BEOGRAD	THESSALONIKI

Prilog 6: Izgled pogleda 'putnici_letovi'

Sintaksa ovog pogleda je:

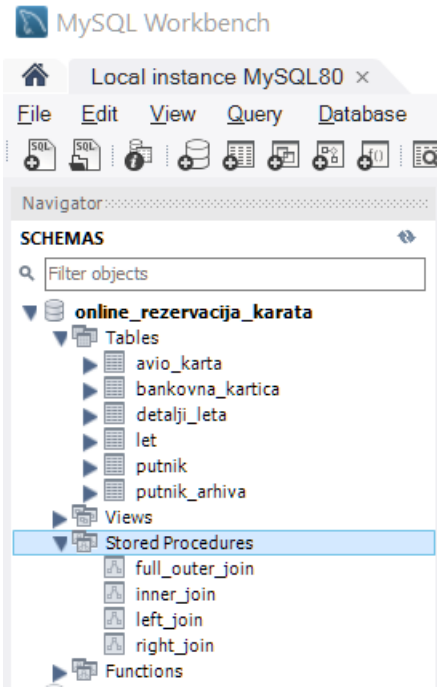
```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `putnici_letovi` AS
    SELECT
        `p`.`ime` AS `ime`,
        `p`.`prezime` AS `prezime`,
        `l`.`datum_i_vrijeme_doleta` AS `datum_i_vrijeme_doleta`,
        `l`.`polaziste` AS `polaziste`,
        `l`.`odrediste` AS `odrediste`
    FROM
        ((`putnik` `p`
        JOIN `avio_karta` `a` ON ((`p`.`id_putnika` = `a`.`id_putnika`)))
        JOIN `let` `l` ON ((`a`.`id_leta` = `l`.`id_leta`)))
```

6 USKLADIŠTENE PROCEDURE

DBMS nudi mogućnost skladištenja upita koji su predefinisani. Postoje sličnosti i različitosti između pogleda i uskladištenih procedura koje ćemo ovdje opisati.

Pogled ne prihvata parametre, može se koristiti kao sub-upit, može sadržati samo jednu komandu select, ne može da vrši modifikacije u tabelama, dok uskladištena procedura prihvata parametre, ne može se koristiti u većim upitima kao sub-upit, može sadržati više operanada i struktura kao što su petlje, IF ELSE, može da modifikuje jednu ili više tabela, i ne može da bude odredište INSERT, UPDATE ili DELETE komandi[3].

3 Za detalje pogledati: https://stackoverflow.com/questions/5194995/what-is-the-difference-between-a-stored-procedure-and-a-view



Prilog 7: Definisane procedure u bazi

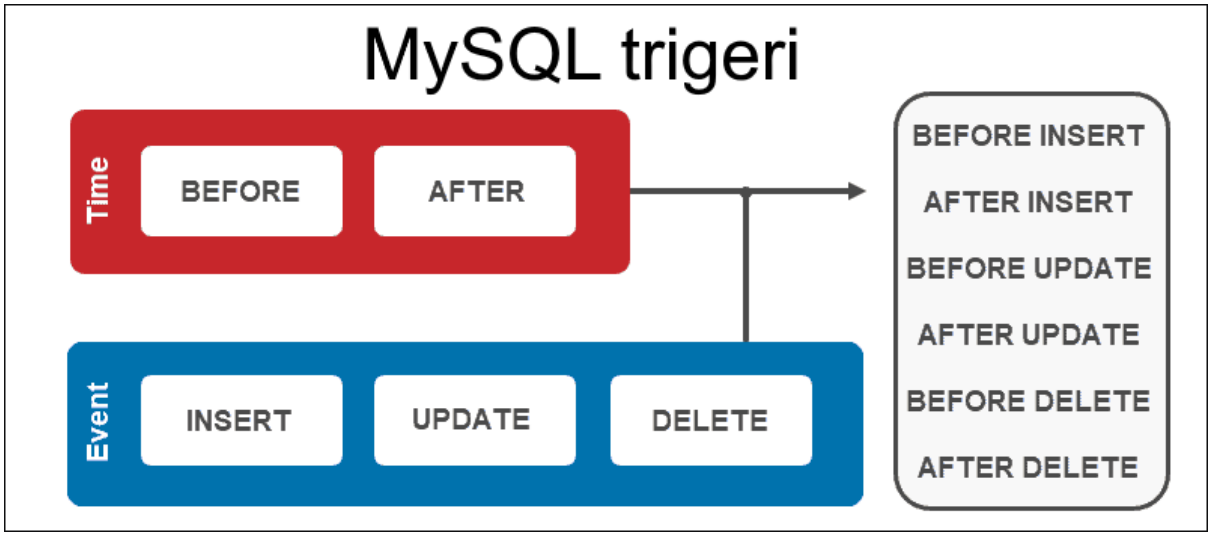
U predmetnoj bazi su definisane 4 procedure, u pitanju su varijante JOIN komande. Svrha ovih upita jeste njihovo brzo izvršavanje i pozivanje, jednom kada se definišu.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `left_join`()
BEGIN
select ime, prezime, datum_i_vrijeme_uzleta from putnik p
left join avio_karta a
on p.id_putnika=a.id_putnika;
END

CREATE DEFINER=`root`@`localhost` PROCEDURE `right_join`()
BEGIN
select ime, prezime, broj_kartice_hash from bankovna_kartica b
right join putnik p
on b.id_putnika=p.id_putnika
order by prezime;
END

CREATE DEFINER=`root`@`localhost` PROCEDURE `inner_join`()
BEGIN
select ime, prezime, odrediste from putnik p
join avio_karta a
on p.id_putnika=a.id_putnika
join let l
on a.id_leta=l.id_leta;

END

CREATE DEFINER=`root`@`localhost` PROCEDURE `full_outer_join`()
```

```
BEGIN
select ime, prezime, id_leta from putnik p
left join avio_karta a
on p.id_putnika=a.id_putnika
union
select ime, prezime, id_leta from putnik p
right join avio_karta a
on p.id_putnika=a.id_putnika;
END
```

7 TRIGERI

Kod baza podataka, trigeri su objekti koji se aktiviraju kada se određeni događaj desi u tabeli. Tačnije rečeno, trigeri su posebna vrsta uskladištenih procedura unutar tabele[4].



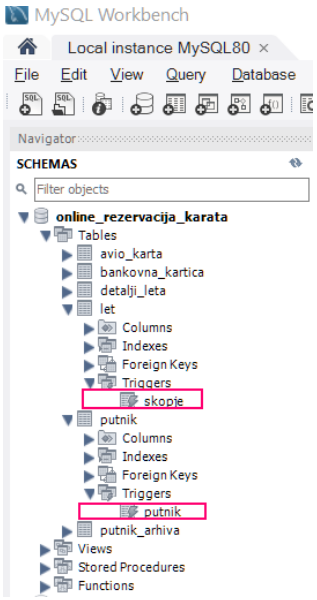
Prilog 8: Tipovi trigera u MySQL

Kao što se vidi, trigeri se mogu definisati kao događaji koji reaguju na 6 slučajeva u jednoj tabeli. Sintaksa za kreiranje trigera je slijedeća:

```
CREATE TRIGGER <trigger name> <trigger time > <trigger event>
ON <table name>
FOR EACH ROW
<trigger body>;
```

U predmetnoj bazi su kreirana dva trigera, u tabeli let je kreiran triger 'skopje', a u tabeli 'putnik' triger 'putnik':

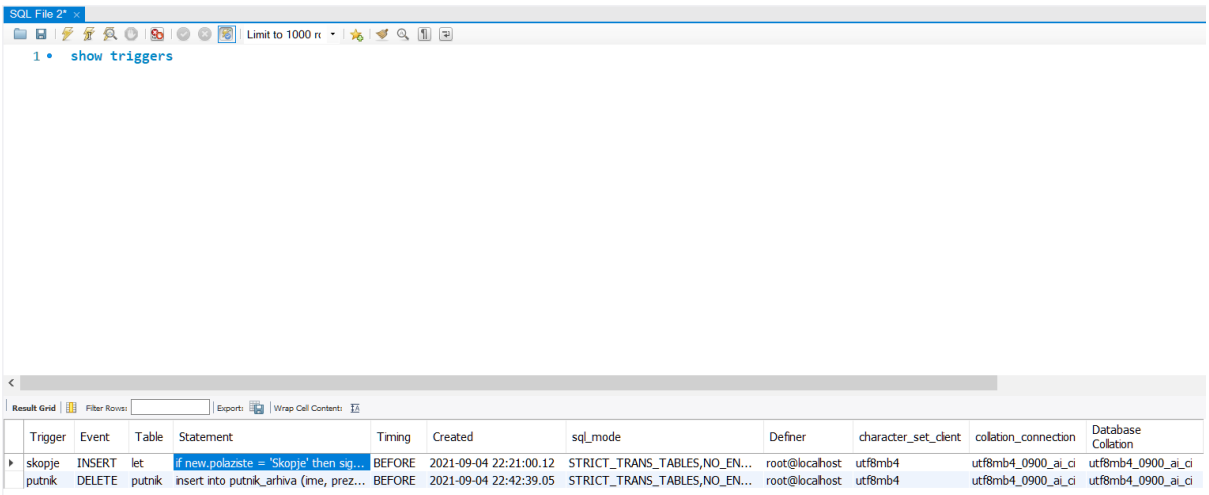
4 Za detalje pogledati: https://phoenixnap.com/kb/mysql-trigger



Prilog 9: Trigeri u bazi

Alternativno se trigeri mogu prikazati slijedećom komandom:

```
show triggers
```



Trigger	Event	Table	Statement	Timing	Created	sql_mode	Definer	character_set_client	collation_connection	Database Collation
skopje	INSERT	let	if new.polaziste = 'Skopje' then sig...	BEFORE	2021-09-04 22:21:00.12	STRICT_TRANS_TABLES,NO_EN...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci
putnik	DELETE	putnik	insert into putnik_arhiva (ime, prez...	BEFORE	2021-09-04 22:42:39.05	STRICT_TRANS_TABLES,NO_EN...	root@localhost	utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

Prilog 10: Detaljan prikaz trigera

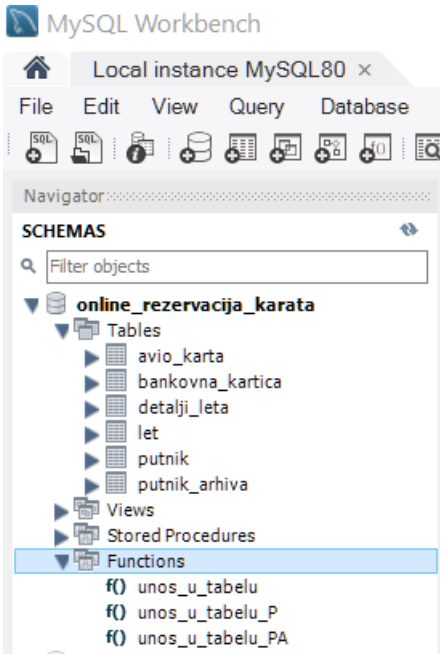
Kod trigera skopje, koji je definisan u tabeli let, logika je slijedeća: ukoliko se ubaci novi red u tu tabelu gdje je vrijednost kolone polaziste 'Skopje', ispisaće se predefinisana poruka 'Trenutno ne letimo iz Skopja'. To je primjer trigera koji reaguje na insert komandu.

Kod trigera 'putnik', koji je definisan u tabeli 'putnik', ukoliko se tabela 'putnik' izbriše, ovaj triger će napraviti kopije podataka iz tabele 'putnik' i smjestiti ih u ranije definisanu tabelu 'putnik_arhiva', odnosno, ovaj triger reaguje na delete komandu.

8 UDF FUNKCIJE

UDF funkcije, odnosno korisnički definisane funkcije predstavljaju funkcionalnost DMBS gdje se mogu definisati funkcije koje korisnik želi da koristi. Potrebno je napraviti razliku između korisničkih funkcija i predefinisanih procedura, budući da se radi o sličnim konceptima, sa ipak značajnim razlikama. Procedura ne vraća vrijednost, već se poziva sa CALL komandom i obično joj je svrha modifikacija tabele ili obrada podataka unutar nje. Funkcija se poziva unutar upita i može da vraća neku vrijednost koja se može dalje koristiti, te, funkcija se ne poziva sa komandom CALL[5].

U predmetnoj bazi kreirane su tri funkcije, koje u različite tabele unose različite unose i skladište ih u vidu redova. Na ovaj način je olakšan unos u tabele obzirom da korisnik treba samo da unese vrijednost polja a ne i da kuca komandu insert into..sa svim pripadajućim parametrima:



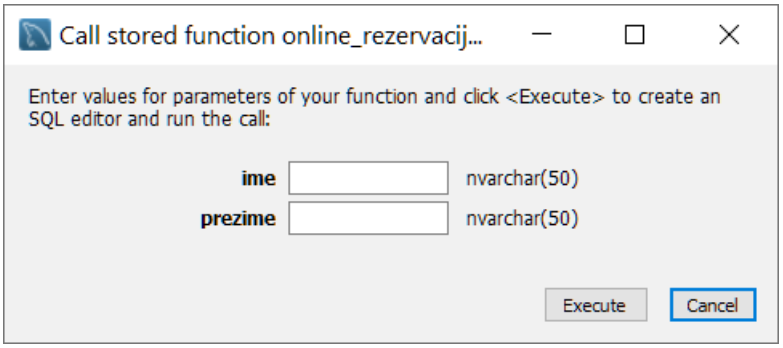
Prilog 11: Funkcije u bazi

Sintaksa je slijedeća:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `unos_u_tabelu_P`(ime
nvarchar(50), prezime nvarchar(50)) RETURNS varchar(50) CHARSET utf8mb3
    DETERMINISTIC
BEGIN
```

5 Za detalje pogledati: https://bestinterviewquestion.medium.com/difference-between-stored-procedure-and-function-in-mysql-52f845d70b05

```
insert into putnik values (ime, prezime);

RETURN 1;
END
```



Prilog 12: Izvršavanje funkcije 'unos_u_tabelu'

9 USPOSTAVLJANJE INDEKSA

Indeks u bazi podataka je takva struktura koja ubrzava operacije. Indeksi se mogu kreirati koristeći jednu ili više kolona. Suštinski, indeksi su takođe tabele koje sadrže vrijednosti primarnih ključeva ili indeksiranih polja koja upućuju na svaki pojedinačni zapis u tabeli. Korisnici ne mogu vidjeti indekse odnosno njihovo funkcionisanje.

Sintaksa je slijedeća:

```
CREATE INDEX index_name
ON table_name (column1, column2, ...);

CREATE UNIQUE INDEX index_name
ON table_name (column1, column2, ...);
```

(jedinstveni indeks, ne dozvoljava duplirane odnosno istovjetne unose)

```
ALTER TABLE table_name
DROP INDEX index_name;
```

(brisanje indeksa)

ZAKLJUČAK

U predmetnom radu pokazan je elementarni proces dizajniranja baze podataka u skladu sa zahtjevima odnosno definisanom problemu, te njeno kreiranje u MySQL. MySQL Workbench kao korišteni alat nudi mnoštvo mogućnosti od kojih bih izdvojio modeliranje prije i poslije skiciranja, odnosno alat nudi mogućnost reverse-engineering u kojem stvara ER dijagram iz već kreiranih tabela i međuzavisnosti, odnosno i obrnutog - forward-engineeringa koji obuhvata mogućnost prvo vizuelnog kreiranja tabela a onda njihovo stvarno kreiranje iz navedenog nacrta. Baza je populisana sa podacima kako bi se omogućilo izvršavanje upita, kreiranje pogleda, funkcija i predefinisanih procedura, te kreirale međuzavisnosti između tabela. Konačno, baza je eksportovana kako bi se mogla naseliti kod drugih korisnika odnosno predmetnog profesora. Kako su predavanja iz datog predmeta bila uz korištenje Microsoft SQL Servera, a da je ovdje korišten MySQL, postoji niz sitnijih razlika koji je autor rada morao da prevaziđe kako bi uspostavljanje baze i izvršavanje upita teklo bez problema. Kako se autor nije ranije susretao sa bazama podataka, zaključak je da su to veoma moćni alati za manipulaciju digitalnim podacima sa mnoštvom mogućnosti i ovaj rad je samo obrisao prašinu sa ovog vrlo važnog i širokog polja u informatičkoj nauci, ali istovremeno bio vrlo koristan i interesantan jer je usmjerio autora ka daljem izučavanju ove oblasti.

LITERATURA

[1] Murach's MySQL, 2nd Edition, Joel Murach, Mike Murach&Associates, Inc. 2015

[2] MySQL 8.0 Reference Manual, https://dev.mysql.com/doc/refman/8.0/en/, pristupano 05.09.2021

[3] PHP, MySQL&JavaScript, Richard Blum, John Wiley&Sons, 2018

[4] Uvod u baze podataka, Peto izdanje, Prof. dr Mladen Veinović, Doc. dr Goran Šimić, Univerzitet Singidunum, 2010