Sveučilište Jurja Dobrile u Puli

Fakultet informatike u Puli

Baze podataka II

Sustav za upravljanje skladištem

Projekt

Tim 2

David Belovari

Boris Besek

Mateo Legović

Randi Mohorović

Tedi Pačić

Mentor: doc. dr. sc. Goran Oreški

Pula, 2022.

Uvod

U ovom projektu iz kolegija Baze podataka II bavili smo se izradom baze podataka i aplikacije. Ideja nam je bila kreirati bazu podataka koja bi se koristila većinu skladišta i izraditi aplikaciju koja bi se također mogla koristit u nekom od skladišta. Osim same izrade projekta cilj nam je bio poboljšati naše znanje vezano uz baze podataka II, te isto tako da naučimo nešto novo i da sami otkrijemo ono što nam nudi program MySQL. Kao tim imali smo puno ideja, ali zbog ograničenog vremena i manjka znanja morali smo se odrediti samo za neke naše ideje. Uz zajedničko pomaganje, istraživanjem i radom uspijeli smo steći nova znanja koja su nam pomoga u usavršavanju našeg projekta.

O projektu

Kao tim za temu projekta odabrali smo Sustav za upravljane skladištem. Rad se zasniva na izradi baze podataka u kojoj se evidentiraju i prikupljaju razni podatci koji su ključni za ciljeve poslovanja odnosno za upravljanje skladistem. Za dizajniranje i implementiranje baze podataka koristili smo MySQL Workbench.Baza podataka sastoji se od 10 povezanih relacija po kojima je modeliran jednostavan proces poslovanja upravljanjem skladista. Prilikom modeliranja baze odnosno ER (entiteti veze atributi) modela koristio se program Lucid.app. Uz navedeni program za potrebe vizualizacije i modeliranja tablica koristio se program Microsoft Excel 2010. U samoj bazi napisali smo 5 složenih upita, 5 pogleda, 5 procedura, 4 okidaća i 1 transakcija. Aplikacija je napravljena u c++ te sluzi za laksu primjenu baze.

Izradom baze podataka i odgovarajuće aplikacije, cijeli je tim stekao nova znanja koje nisu nužna za prolaz kolegija ali su svakako zanimljiva i korisna za daljnji razvoj iz SQL-a. Što se tiće samog izbora teme nije bilo problema ali što smo više razmišljali o načinu kreiranja i implementiranja određenih komponenata u bazi podataka dolazilo je do komplikacija zbog našeg prijašnje slabog znaja. Svaki problem na koji smo naišli sa zadovoljstvom smo ga otklonili i stekli nova znanja.

Baza podataka u MySQL-u

Baza podataka je organizirani mehanizam koji ima sposobnost pohranjivanja informacije, kroz koji korisnik može dohvatiti pohranjene informacije na učinkovit način. Jednostavnije rečeno baza podataka je kolekcija trajno pohranjenih podataka. Naša baza podataka izrađena je u aplikaciji MySQL Workbench-u. Unutar aplikacije pomoću raznih naredbi kreirali su relacije, poglede, funkcije, pohranjene procedure, okidače, transakcije…

Relacije (tablice)

Relacija ili tablica predstavlja podatke, njihove domene i veze između podataka. Napravili smo deset smislenih relacija koje su ključne za ispravan rad projekta. O uvom dijelu opisat ćemo detaljno jednu tablicu.

Relacije u našoj bazi podataka:

- Firma
- Posao
- Skladiste
- Kupac
- Zaposlenici
- Proizvod
- Vozilo
- Dostavljać
- Narudžba
- Dobavljač

Za svaku smo relaciju odredili atribute, a za svaki atribut odgovarajuću domenu. Domene koje smo koristili za našu bazu podataka su slijedeće: integer, char, varchar, date, bigint. Osim domena koristili ograničenja: check, unique. Kako bi mogli razlikovati pojedine n-torke unutar relacije koristili smo ključeve (primarni ključ i strani ključ). Za različite ID-eve koristili smo auto_increment koji nam je omogućio brži unos

različitih brojeva za svaku relaciju. Pomoću insert-a dodali smo u našu bazu za svaku tablicu određeni broj redova, odnosno unesli smo određene podatke.

Relecija koju ćemo Vam detaljno opisati je relacija zaposlenici koja se sastoji od sljedećih atributa i domena:

- ID INTEGER NOT NULL,
- ime VARCHAR(50) NOT NULL,
- prezime VARCHAR(50) NOT NULL,
- OIB BIGINT NOT NULL,
- primanja INTEGER NOT NULL,
- datum_zaposlenja DATE NOT NULL,
- ID_posao INTEGER NOT NULL,
- ID_skladiste INTEGER NOT NULL,
- PRIMARY KEY (ID)

Opis korištenih domena u relaciji zaposlenici:

- INTEGER - ovu domenu smo koristili za ID, primanja, tj. plaću zaposlenika
- VARCHAR - varchar je domena kojoj moramo odredit broj znakova, u oba naša slučaja u ovoj relaciji ograničili smo varchar na 50 znakova.
- BIGINT - koristili smo bigint za oib jer običan in može spremiti do broja 2147483 što nam nije bilo dovoljno za oib
- DATE - datum je korišten na atributu datum_zaposlenja kako bi mogli evindetirati datume zaposlenja radnika

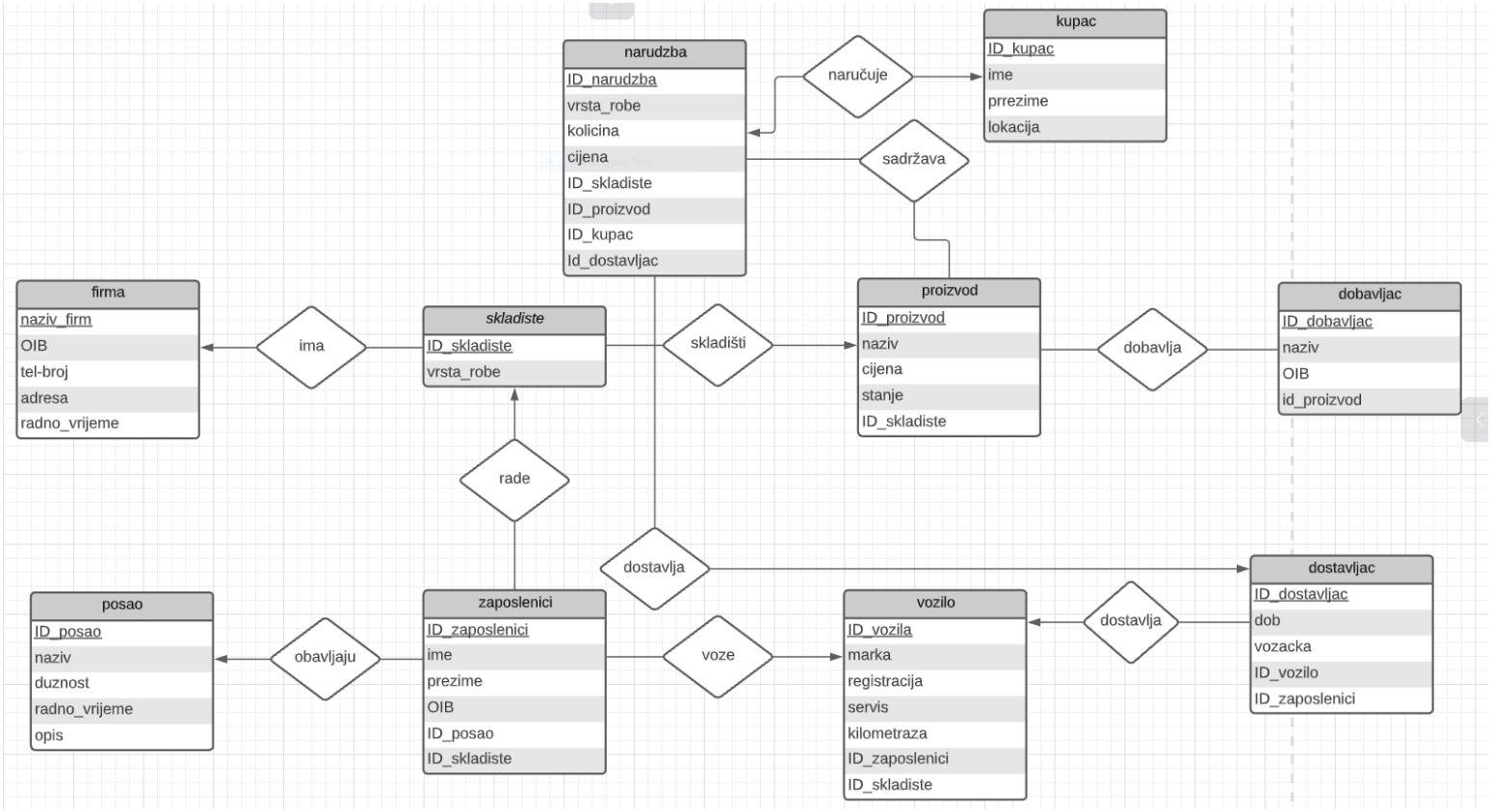
Nakon dodavanja domene atributima, dodali smo neka ograničenja. Ograničenje *not null* koristili smo primarno za ID kako bi spriječili da ID bude nula. *Not null* smo postavili i na ostale atribute. *Primary key*, odredili smo da svaki zaposlenik ima svoj ID koji se neće duplirati s nekim drugim ID-em.

Veze među relacijama

Relacije su međusobno povezane pomoću stranih ključeva. Njihova međusobna povezanost je slijedeća:

- relacija zaposlenici povezana je s relacijama: posao, skladiste
- relacija narudzba povezana je s relacijama: skladiste, proizvod, kupac, dostavljac
- relacija proizvod povezana je s relacijom skladiste
- relacija dobavljac povezana je s relacijom proizvod
- relacija dostavljac povezana je s relacijama: vozilo, zaposlenici
- relacija vozilo povezana je s relacijama: zaposlenici, skladište

ER dijagram



Pošto smo ER dijagram detaljno opisali u projektu iz „*Baza podataka I*“, kao tim zajedničkim dogovorom odlučili smo ne pretjerano opisitavi ER dijagram kako bi se mogli posvetiti novom gradivu.

ER dijagram prikazuje relacije između entiteta u bazi podataka. Zbog svoje je jednostavnosti razumljiv ljudima različitih struka te stoga predstavlja odličan alat za komunikaciju dizajnera baze podataka s korisnicima i programerima u najranijoj fazi razvoja baze podataka. ER dijagram prikazuje entitete (tablice) pomoću pravokutnika, dok su veze među njima prikazane pomoću rombova koji se s entitetima povezuju bridovima. U dijagramu se također nalaze i imena entiteta i veza, kao i funkcionalnost veza. Relacije ili veze između entiteta mogu biti unarne, binarne ili n-arne. Ako je veza unarna, onda je broj entiteta koji sudjeluju u toj vezi jedan. U binarnoj vezi sudjeluju dva, a u n-arnoj više od dva entiteta.

Funkcije, pohranjene procedure, okidači, transakcije, pogledi...

Upiti

Upit koji ispisuje proizvode iz narudzbe gdje je u narudzbi vise od 8 proizvoda

```
select p.*
from proizvod as p
where p.id in (select id_proizvod
from narudzba
group by id_proizvod
HAVING SUM(kolicina) > 8);
```

Upit koji prikazuje sva vozila marke mercedes koja su bila na servisu u zadnjih 8 mjeseci

```
select * from vozilo
where marka = 'Mercedes' and servis > now() - interval 8 month;
```

Upit koji prikazuje najskupljiji proizvod

```
SELECT *
FROM proizvod
ORDER BY cijena_proizvod DESC
LIMIT 1;
```

Upit koji prikazuje 3 volila najblize servisu

```
SELECT marka, servis, kilometara
FROM vozilo
ORDER BY servis ASC
LIMIT 3 ;
```

Upit koji prikazuje najjeftiniji alkohol

```
SELECT *
FROM proizvod
WHERE ID_skladiste = 2
ORDER BY cijena_proizvod ASC
LIMIT 1;
```

Funkcije

Funkcije pohranjene u MySQL koriste se za kapsuliranje izračuna ili operacija s zapisima i poljima podataka koji su uzeti iz SQL upita i uobičajeni su zadaci ili poslovna pravila. Velika prednost je što su oni za višekratnu upotrebu i programski jezik u kojem su razvijene funkcije je kroz SQL izjave i uvjetne ili ponavljajuće strukture. Za razliku od pohranjene procedure, možete koristiti funkciju pohranjenu u SQL izrazima gdje se koristi izraz koji vam omogućuje stvaranje uvjetnih pravila.

Funkcija koja vraća broj proizvoda

```
DELIMITER //
CREATE FUNCTION broj_proizvoda() RETURNS INTEGER
DETERMINISTIC
BEGIN
DECLARE  rez INTEGER DEFAULT 0;

SELECT COUNT(*) INTO rez
FROM proizvod;

RETURN rez;
END//
DELIMITER;
SELECT broj_proizvoda() from DUAL;
```

Pohranjene procedure

Pohranjeni postupci su mali programi razvijeni u SQL kodu. Pohranjeni postupak je skup SQL naredbi koje se pohranjuju zajedno s bazom podataka. Prednost pohranjenog postupka je u tome što ga možemo stvoriti u bilo kojem uređivaču teksta, pa čak i na poslužitelju, izvodi ga baza podataka i nije dostupan korisniku, već samo administratoru. Spremljena procedura šalje svoje rezultate aplikaciji tako da ih prikazuje na zaslonu.

Procedura koja ispisuje minimalnu i maksimalnu cijenu proizvoda

```
DELIMITER //
CREATE PROCEDURE cijena_proizvoda(OUT min_cijena DECIMAL(10,2), OUT max_cijena DECIMAL(10, 2))
BEGIN
 DECLARE cur CURSOR FOR
 SELECT MIN(cijena_proizvod), MAX(cijena_proizvod) FROM proizvod;

 OPEN CUR;
 FETCH cur INTO min_cijena, max_cijena;
 CLOSE cur;

END //
DELIMITER ;
CALL cijena_proizvoda(@min_cijena, @max_cijena);
SELECT @min_cijena, @max_cijena FROM DUAL;
```

Procedura koja unosi kupce

```
DELIMITER //
CREATE PROCEDURE unos_kupca (IN p_ID INTEGER, IN p_ime VARCHAR(20), IN p_prezime VARCHAR(20), IN p_lokacija VARCHAR(50))
BEGIN

   INSERT INTO kupac VALUES (p_ID, p_ime, p_prezime, p_lokacija);

END //
DELIMITER ;
CALL unos_kupca(31, 'Ana', 'Anić', 'Poreč');
CALL unos_kupca(32, 'Leana', 'Buršić', 'Zagreb');
CALL unos_kupca(33, 'Lina', 'Burić', 'Poreč');
CALL unos_kupca (34, 'Zdenko', 'Tadić', 'Zagreb');
CALL unos_kupca (35, 'Matijas', 'Houdek', 'Pula');
CALL unos_kupca (36, 'Karlo', 'Burić', 'Pula');
CALL unos_kupca (37, 'Stjepan', 'Živko', 'Rijeka');
CALL unos_kupca (38, 'Tomislav', 'Martinović', 'Zagreb');
CALL unos_kupca (39, 'Josip', 'Ciklić', 'Rijeka');
CALL unos_kupca (40, 'Bruno', 'Antunović','Pula');
```

Procedura koja unosi proizvode

```
DELIMITER //
CREATE PROCEDURE unos_proizvoda (IN p_ID INTEGER,IN p_naziv VARCHAR(50),
IN p_cijena_proizvod NUMERIC(10,2), IN p_stanje CHAR(2),IN p_ID_skladiste INTEGER)
BEGIN
   INSERT INTO proizvod VALUES (p_ID, p_naziv, p_cijena_proizvod, p_stanje, p_ID_skladiste);
END //
DELIMITER ;
CALL unos_proizvoda(31, 'Iso-sport', 9.00, 'DA', 1);
CALL unos_proizvoda(32, 'Malibu', 115.00, 'DA', 2);
CALL unos_proizvoda(33, 'Pringles', 15.00, 'DA', 3);
CALL unos_proizvoda(34, 'Šljivovica', 75.00, 'DA', 2);
CALL unos_proizvoda(35, 'Viljamovka', 99.00, 'DA', 2);
```

Procedura koja unosi narudžbe

```
DELIMITER //
CREATE PROCEDURE unos_narudzbe (IN p_ID INTEGER, IN p_vrsta_robe VARCHAR(50), IN p_kolicina INTEGER,
IN p_cijena NUMERIC(10,2), IN p_ID_skladiste INTEGER, IN p_ID_proizvod INTEGER, IN p_ID_kupac INTEGER, IN p_ID_dostavljac INTEGER)
BEGIN
   INSERT INTO narudzba VALUES (p_ID, p_vrsta_robe, p_kolicina, p_cijena, p_ID_skladiste, p_ID_proizvod, p_ID_kupac, p_ID_dostavljac);
END //
DELIMITER ;
CALL unos_narudzbe(31,'Gazirano piće', 2,18.00,1,31,1,1);
CALL unos_narudzbe(32,'Alkohol',1,115.00,2,32,2,1);
CALL unos_narudzbe(33,'Grickalice',3,45.00,3,33,3,2);
CALL unos_narudzbe(34,'Alkohol',1,75.00,2,34,4,2);
CALL unos_narudzbe(35,'Alkohol',1,99.00,2,35,5,3);
```

Procedura koja određuje akcije

```
DELIMITER //
CREATE PROCEDURE akcija (OUT akcijska_cijena DECIMAL(10,2))
BEGIN
    SELECT * ,10/cijena_proizvod AS akcijska_cijena
    FROM proizvod;
END//
DELIMITER ;
CALL akcija(@akcijska_cijena);
SELECT @akcijska_cijena FROM DUAL;
```

Okidači

Okidači su vrsta objekta koji se nalazi u bazama podataka. Kad se određeni događaj ili operacija izvodi u navedenoj bazi podataka, naš okidač automatski aktivira radnju za koju smo je prethodno programirali.

Okidači se automatski aktiviraju kada se izvode INSERT, DELETE ili UPDATE operacije, a to su operacije umetanja, brisanja ili ažuriranja. Prilikom izvršavanja bilo koje od ovih radnji naši okidači izvode upute ili blok uputa za koje su prethodno programirani.

Okidač koji omogućuje da se naziv firme ne može mijenjati

```
DELIMITER //
CREATE TRIGGER bu_firma
 BEFORE UPDATE ON firma
 FOR EACH ROW
    BEGIN
     SET new.naziv_firme = old.naziv_firme;
    END//
DELIMITER ;
```

Okidač koji omogućuje da kod unosa novog proizvoda cijena ne moze biti veca od max cijene

```
DELIMITER //
CREATE TRIGGER zr_cijena
 BEFORE INSERT ON proizvod
 FOR EACH ROW
    BEGIN
     DECLARE max_cijena DECIMAL(10, 2);
     SELECT MAX(cijena_proizvod) INTO max_cijena
     FROM proizvod;

     IF new.cijena_proizvod > max_cijena THEN
     SIGNAL SQLSTATE '40000'
     SET MESSAGE_TEXT = 'Cijena novog proizvoda ne može biti veća od najskupljega!';
        ELSE
            SET new.cijena_proizvod = new.cijena_proizvod;
     END IF;
    END//
DELIMITER ;
INSERT INTO proizvod VALUES (55, 'Brandy', 277.00,'DA',2);
INSERT INTO proizvod VALUES (57, 'Vutra', 747.00,'DA',2);
SELECT * FROM proizvod;
```

Okidač koji omogućuje da kad je cijena manja od 0, stavlja se u min cijenu

```
DELIMITER //
CREATE TRIGGER tr_cijena
 BEFORE INSERT ON proizvod
 FOR EACH ROW
    BEGIN
    DECLARE min_cijena DECIMAL(10, 2);

     SELECT MIN(cijena_proizvod) INTO min_cijena
     FROM proizvod;

     IF new.cijena_proizvod < 0 THEN
     SET new.cijena_proizvod = min_cijena;
     END IF;
    END//
DELIMITER ;
INSERT INTO proizvod VALUES (111, 'Southern Comfort', -6.59,'DA',2);
SELECT * FROM proizvod;
```

Okidač koji onemogućava negativnu količinu

```
DELIMITER //
CREATE TRIGGER bi_proizvod
 BEFORE INSERT ON proizvod
 FOR EACH ROW
BEGIN
 IF new.cijena_proizvod < 0 THEN
    SET new.cijena_proizvod = 1;
 END IF;
END//
DELIMITER ;
INSERT INTO proizvod VALUES(1110, 'Hidra', -14.00, 'DA', 1);
INSERT INTO proizvod VALUES(1000, 'Kokice', -3.50, 'DA', 3);
```

Transakcije

Transakcija u bazi podataka je na primjer umetanje zapisa ili ako to vidimo kao poslovni model Dodajte novog kupca, Izmijenite proizvod. Transakcije uvijek proizvode promjenu, umetanje, izmjenu, brisanje, upit nije transakcija jer ne proizvodi promjene.

Transakcija koja kod unosa novog zaposlenika oib nesmije biti identičan

```
DELIMITER //
CREATE PROCEDURE unos_novog_zaposlenika(p_id INTEGER, p_ime VARCHAR(20), p_prezime VARCHAR(20),p_oib CHAR(9),p_id_posao INTEGER, p_id_skladiste INTEGER )
BEGIN
 DECLARE EXIT HANDLER FOR 1062
 BEGIN
 ROLLBACK;
 SELECT CONCAT('Nemoguć unos zaposlenika, već postoji');
 END;

 SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;
 START TRANSACTION;

 INSERT INTO zaposlenici VALUES (p_ID, p_ime, p_prezime, p_OIB, p_ID_posao , p_ID_skladiste  );
 COMMIT;
END //
DELIMITER ;
SELECT ID,OIB FROM zaposlenici;
CALL unos_novog_zaposlenika(31, 'Noel', 'Hafizović', '123456795', 1, 1);
```

Pogledi

Pogledi su MySQL funkcionalnosti koje nam omogućuju definiranje prezentacije jedne ili više tablica. Moguće je izvršiti SELECT upite nad pogledima, te ovisno o definiciji pogleda, promijeniti naredbe INSERT, UPDATE i DELITE.

Pogled koji prikazuje sve kupce poredani po abecednom redu

```
CREATE VIEW pogled_kupac AS
    SELECT * FROM kupac
    WHERE ID IN (SELECT ID_kupac FROM narudzba)
    ORDER BY ime;
SELECT * FROM pogled_kupac;
```

Pogled koji prikazuje proizvode

```
CREATE VIEW pogled_proizvoda AS
  SELECT naziv, cijena_proizvod
    FROM proizvod;
    SELECT * FROM pogled_proizvoda;
```

Pogled koji prikazuje akcije

```
CREATE VIEW pogled_akcija AS
 SELECT naziv, 10/cijena_proizvod AS akcijska_cijena
    FROM proizvod;
    SELECT * FROM pogled_akcija;
```

Pogled koji prikazuje registracije

```
CREATE VIEW datum_registracije AS
    SELECT registracija FROM vozilo
    ORDER BY registracija ASC;
SELECT * FROM datum_registracije;
```

Pogled koji prikazuje dob dostavljaća

```
CREATE VIEW pogled_dobi_dostavljaca AS
SELECT dob from dostavljac ORDER BY dob DESC;
SELECT * FROM pogled_dobi_dostavljaca;
```

Programsko rješenje

Zaključak

Iako smo nailazili na mnogo prepreka i problema, bazu podataka smo uspiješno napravili isto kao i dokumentaciju i programsko rješenje. Stekli smo mnogo novih znanja koje će nam poslužiti za daljnje obrazovanje vezano uz baze podataka u MySql-u. Prilikom izrade pojekta nije došlo do međuljudskih svađa ili problema te smo se međusobno poštovali i zajedničkim snagama uspiješno napravili zadani projekt.

Svaki član tima snosio je određenu odgovornost za ono što je njegov zadatak. Kada je bilo najteže i kada smo mislili da nećemo moći, uvijek smo jedni druge gurali naprijed, davali si nove ideje, nove zadatke i nove prepreke koje treba proći. Nije jednostavno napraviti projekt u par mjeseci sa novostećenim znanjem i novim vještinama koje smo stekli tijekom predavanja i vježba. Osim stečenog znanja usavršili smo međusobnu komunikaciju jer je to zapravo najbitnije kabo bi se ostavrili određeni ciljevi.

Zadovoljni smo svojim zajedničkim radom i možemo reći da bi vrlo rado ovaj projekt usavršavali još dugo i poboljšavali ono što je dosad napisano jer bi svi rado učili dalje što nam baza podataka može pružiti, usavršavali bi i dalje svoje znanje i nadograđivali ga novim znanjem. Kao tim nadamo se da ćemo ubrzo nam se svima pružiti prilika da ponovno izrađujemo nekakvu novu bazu podataka i da radimo neki novi projekt zajedno te da ponovno pokažemo naše novo.