Sveučilište Jurja Dobrile u Puli
Odjel za Tehničke studije i Fakultet Informatike u Puli

MEDICINA

Dokumentacija baze podataka

tim 5

Dino Gajić	0303092395
Enrico Kokot	0009068626
Mateja Kurilić	0036496139
Ivan Tacko	0023118918

Studijski smjer: Računarstvo, Informatika

Kolegij: Baze Podataka 1

Mentor: doc. dr. sc. Goran Oreški

Pula, 2. lipnja, 2020. godine

Pregled sadržaja:

1. Uvod

Naša baza podataka služi službenicima medicinskog sustava kako bi svi podaci bili usklađeni u cjelokupnom zdravstvenom sustavu RH.

Ustanove koje su povezane u ovoj bazi su apoteke i bolnice.
Osobe koje će se služiti ovom bazom podataka su službenici ljekarne, doktori u bolnicama, osobe koje uvode nove dijagnoze, osobe koje uvode nove specijalizacije, nove liječnike, novorođenu djecu koja po rođenju ulaze u sustav i tablicu ***osoba*** sustava HZZO, nove bolnice, nove apoteke, farmaceutske tvrtke koje uvode nove terapije.

Ova baza sadrži samo osobe koje su u sustavu HZZO-a, za strane državljane bi trebali uvesti veliku količinu novih pravila koja bi značajno proširila bazu za naše ciljeve.
Pretpostavlja se da su sve osobe u sustavu HZZO-a nalaze u tablici ***osoba***.
Količina podataka u svim tablicama je smanjene od one realne za potrebe našeg projekta.

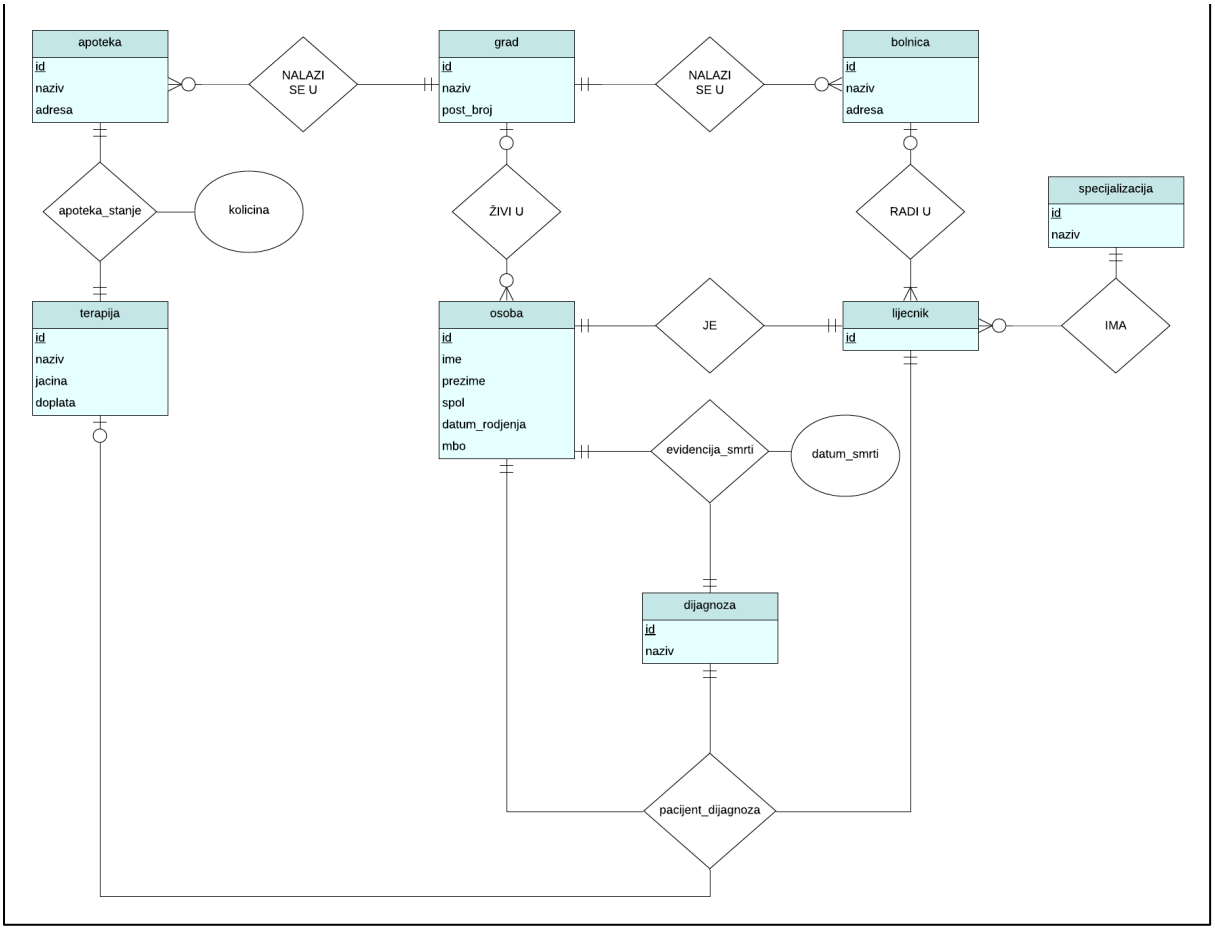
Sva imena, prezimena, mbo-ovi, datumi rođenja, id-evi, itd. su nasumično generirani na smislen način.

Cilj ove baze podataka je primarno uspostaviti odnos između pacijenta, liječnika koji mu je postavio dijagnozu, njegove dijagnoze i ljekarne koja sadrži odgovarajuću terapiju za tog pacijenta kako bi svaki pacijent imao pristup odgovarajućoj zdravstvenoj skrbi i kako nitko ne bi mogao zlouporabiti podizanje lijeka koji mu nije namijenjen koristeći sustav.
Baza ima i druge namjene, uključujući ali ne i ograničeno na:

- praćenje radnih mjesta liječnika, njihovih specijalizacija, pacijenata i terapija
- povezivanje liječnika u sustavu sa njihovim osobnim podacima koji ih predstavljaju kao korisnike zdravstvenih usluga sustava
- povezivanje osoba, bolnica i apoteka sa gradovima u kojima borave / su sagrađene
- praćenje raznih informacija vezanih uz smrti korisnika sustava
- praćenje rizičnih skupina poput umirovljenika ili osoba zaraženih visokorizičnim bolestima
- praćenje različitih odnosa liječnika i njihovih pacijenata
- praćenje informacija vezanih uz bolnice poput potreba za više specijalističkog osoblja ili potencijalnih lokacija za njihovu izgradnju

2. Opis poslovnog procesa

Sljedeći ER dijagram pokazuje na jednostavan i pregledan način sve entitete te njihovu međusobnu povezanost. Ispod dijagrama se nalazi detaljan opis svake od veza među relacijama. Kardinalnosti su označene simbolima, N-N veze koje su zapravo tablice u bazi, prikazane su u obliku veza, gdje dvije tablice sadrže i dodatni atribut, prikazan u mjehuriću.



Slika 1 ER dijagram

ŽIVI U povezuje entitete ***grad*** i ***osoba***. U nekom gradu može se nalaziti nula ili više osoba, a svakoj osobi je pridružen jedan grad (postoji mogućnost da joj nije pridružen nijedan grad, ukoliko je grad izbrisan iz baze tada se osoba evidentira kao osoba bez mjesta stanovanja dok ju se ne ‘useli’ u neki drugi grad iz baze)

Veza **JE** na jedinstven način povezuje entitete ***osoba*** i ***lijecnik,*** jedan liječnik je točno jedna osoba, odnosno jedna osoba se ne može voditi pod više šifri liječnika

Veza **IMA** povezuje entitete ***lijecnik*** i ***specijalizacija*** na način da svaki od liječnika ima točno jednu specijalizaciju, dok neku specijalizaciju može imati nula ili više liječnika.

Veza **NALAZI SE U** spaja entitete ***apoteka*** i ***grad*** te označava da se u svakom gradu može nalaziti nula ili više apoteka, dok svaka apoteka mora biti pridružena točno jednom gradu.
Istoimena veza spaja i entitete ***bolnica*** i ***grad***. U ovom odnosu označava da svaki grad može imati 0 ili više bolnica, ali jedna bolnica može biti pridružena točno jednom gradu.

RADI U između entiteta ***lijecnik*** i ***bolnica*** povezuje liječnika s jednom ili nijednom bolnicom (može biti nezaposlen), svaka bolnica, kako bi mogla raditi, mora imati jednog ili više liječnika.

SQL tablica **apoteka_stanje** povezuje entitete ***apoteka*** i ***terapija***, na način da se u jednoj apoteci može nalaziti 0 ili više terapija, a neka terapija se može nalaziti u 0 ili više apoteka. Za pojedinu terapiju u apoteci utvrđuje se količina.

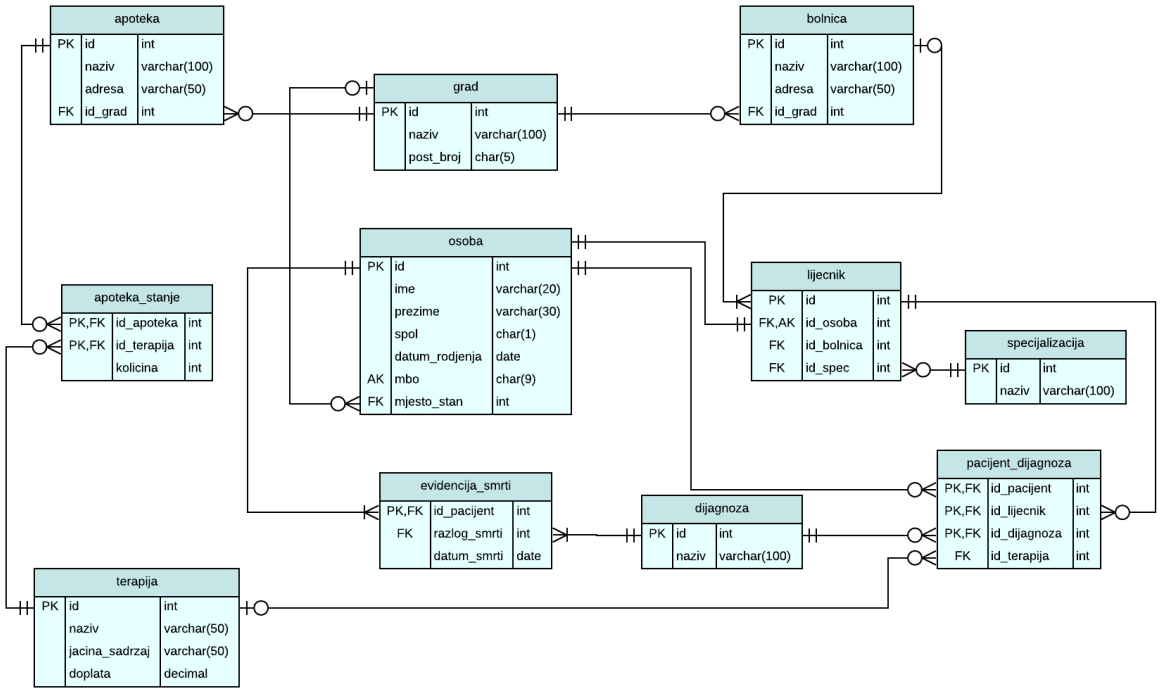
Tablica **pacijent_dijagnoza** povezuje entitete ***lijecnik,osoba, dijagnoza*** i ***terapija,*** na način da entiteti ***lijecnik, osoba*** i ***dijagnoza*** čine ternarnu vezu koja rezultira s nijednom ili jednom terapijom.

- postoje liječnici koji su postavili nula dijagnoza, osobe koje ne boluju, te dijagnoze koje nisu postavljene i terapije koje nisu dodijeljene
- svaki liječnik može postaviti više dijagnoza većem broju pacijenata, jednom pacijentu može više liječnika postaviti više dijagnoza
- svaka kombinacija ***lijecnik, osoba*** i ***dijagnoza*** gleda se kao ključ koji može, ali i ne mora odrediti točno jednu terapiju (terapija se ne određuje u slučaju da pacijent odbija liječenje ili ne postoji potreba za liječenjem/ ne postoji terapija)

Tablica **evidencija_smrti** spaja entitete ***osoba*** i ***dijagnoza.*** Više osoba može umrijeti pod istom dijagnozom, no razlog smrti jedne osobe može biti točno jedan (točno jedna dijagnoza). Entitet ***osoba***, odnosno njen **id** tvori primarni ključ koji jednoznačno određuje raznog i datum smrti.

3. Tablice, atributi i domene

U ovom poglavlju ćemo detaljno opisati svaku od tablica, svaki od njenih atributa i domenu svakog atributa.
Slika prikazuje relacijski model naše baze podataka.



Slika 2 Relacijski model baze

3.1. grad

```
CREATE TABLE grad (
      id          INTEGER PRIMARY KEY,
      naziv       VARCHAR(100) NOT NULL,
      post_broj   CHAR(5) UNIQUE NOT NULL,
CHECK (length(post_broj) = 5)
);
```

Tablica ***grad*** sadrži:

- primarni ključ (**id** - INTEGER) koji identificira grad,
- naziv grada (VARCHAR(100)) te
- njegov poštanski broj (CHAR(5)) koji je također i univerzalan za svaki grad te se može koristiti kao prirodni ključ.

Pošto smo u Hrvatskoj, poštanski broj mora sadržavat točno 5 brojčanih charactera.
Sve vrijednosti moraju biti ispunjene, ne smiju biti NULL.

3.2. bolnica

```
CREATE TABLE bolnica (
      id          INTEGER PRIMARY KEY,
      naziv       VARCHAR(100) NOT NULL,
      adresa      VARCHAR(50) NOT NULL,
      id_grad     INTEGER NOT NULL,
FOREIGN KEY (id_grad) REFERENCES grad(id) ON DELETE CASCADE
);
```

Tablica ***bolnica*** sadrži primarni ključ (**id** - INTEGER) koji identificira bolnicu pa mora biti jedinstven.
Definirana je:

- nazivom bolnice (VARCHAR(100)),
- adresom (VARCHAR(50)) te
- gradom, tj njegovim id-om (INTEGER) koji je u ovoj tablici strani ključ.

Sve vrijednosti u tablici moraju biti različite od NULL.
Bolnica se briše ukoliko ukoliko se grad u kojem nalazi izbriše iz baze.

3.3. apoteka

```
CREATE TABLE apoteka (
      id          INTEGER PRIMARY KEY,
      naziv       VARCHAR(100) NOT NULL,
      adresa      VARCHAR(50) NOT NULL,
      id_grad     INTEGER NOT NULL,
FOREIGN KEY (id_grad) REFERENCES grad(id) ON DELETE CASCADE
);
```

Tablica ***apoteka*** sadrži primarni ključ (**id** - INTEGER) koji identificira apoteku.
Svaka apoteka je određena:

- nazivom (VARCHAR (100))
- adresom(VARCHAR(50))
- nalazi se u gradu čiji je id (INTEGER) strani ključ ove tablice.

Kao i u slučaju bolnice, apoteka se briše iz baze ukoliko se ukloni grad u kojem se ona nalazi.
Podaci u tablici ne smiju poprimiti vrijednost NULL.

3.4. specijalizacija

```
CREATE TABLE specijalizacija (
      id          INTEGER PRIMARY KEY,
      naziv       VARCHAR(100) UNIQUE NOT NULL
);
```

Tablica ***specijalizacija*** sadrži:

- primarni ključ (**id** - INTEGER) koji identificira specijalizaciju
- svaka specijalizacija ima svoj naziv koji je jedinstven te se također može koristiti kao ključ u dohvaćanju id-a

Naziv ne smije poprimiti vrijednost NULL.

3.5. osoba

```
CREATE TABLE osoba (
      id                INTEGER PRIMARY KEY,
      ime               VARCHAR(20) NOT NULL,
      prezime           VARCHAR(30) NOT NULL,
      spol              CHAR(1) NOT NULL,
      datum_rodjenja    DATE NOT NULL,
      mbo               CHAR(9) UNIQUE NOT NULL,
      mjesto_stan       INTEGER,
FOREIGN KEY (mjesto_stan) REFERENCES grad(id) ON DELETE SET NULL,
CHECK (length(mbo) = 9),
CHECK (spol IN ('M', 'Ž'))
);
```

Tablica ***osoba*** sadrži:

- primarni ključ (**id** - INTEGER) koji identificira osobu pa on mora biti jedinstven,
- ime (VARCHAR(20)),
- prezime (VARCHAR(30)),
- spol (CHAR(1)),
- datum rođenja (DATE),
- matični broj osiguranika (CHAR(9)) koji je jedinstven te se može koristiti kao primarni ključ,
- strani ključ koji određuje grad, a označava mjesto stanovanja (mjesto_stan - INTEGER).

Pri brisanju grada iz baze, osoba gubi status građanina, dok se ne useli u jedan od postojećih gradova.
Vrijednosti koje spol može poprimiti su ‘M’ i ‘Ž’, a duljina matičnog broja osiguranika mora biti 9.

3.6. lijecnik

```
CREATE TABLE lijecnik (
      id          INTEGER PRIMARY KEY,
      id_osoba    INTEGER UNIQUE NOT NULL,
      id_bolnica  INTEGER,
      id_spec     INTEGER NOT NULL,
FOREIGN KEY (id_osoba) REFERENCES osoba(id) ON DELETE CASCADE,
FOREIGN KEY (id_bolnica) REFERENCES bolnica(id) ON DELETE SET NULL,
FOREIGN KEY (id_spec) REFERENCES specijalizacija(id) ON UPDATE CASCADE
);
```

Tablica ***lijecnik*** sadrži primarni ključ (**id** - INTEGER) koji identificira liječnika.
Svaki podatak u ovoj tablici određen je primarnim ključem i tri strana ključa:

- id_osoba (INTEGER) - jedinstvena oznaka osobe, dakle liječniku se ovime omogućuje da i on bude pacijent nekog liječnika te da mu se dodijeli dijagnoza i potrebna terapija
- id_bolnica (INTEGER) - oznaka bolnice u kojoj ovaj liječnik radi, može poprimiti NULL vrijednost ukoliko liječnik dobije otkaz
- id_spec (INTEGER) - oznaka specijalizacije za koju je liječnik školovao

3.7. dijagnoza

```
CREATE TABLE dijagnoza (
      id          INTEGER PRIMARY KEY,
      naziv       VARCHAR(100) UNIQUE NOT NULL
);
```

Tablica ***dijagnoza*** sadrži:

- primarni ključ (**id** - INTEGER) koji identificira dijagnozu
- svaka je dijagnoza opisana nazivom (VARCHAR(100) koji je jedinstven te se može koristiti kao ključ za dohvaćanje id-a)

3.8. terapija

```
CREATE TABLE terapija (
      id                INTEGER PRIMARY KEY,
      naziv             VARCHAR(50) NOT NULL,
      jacina_sadrzaj    VARCHAR(50) NOT NULL,
      doplata           NUMERIC(5,2) NOT NULL,
CHECK (doplata >= 0)
);
```

Tablica ***terapija*** sadrži primarni ključ (**id** - INTEGER) te je opisana:

- svojim nazivom (VARCHAR(50))
- jačinom i sadržajem paketa (VARCHAR(50))
- doplatom u HRK (NUMERIC(5,2)) koja ne može biti negativnog iznosa.

3.9. apoteka_stanje

```
CREATE TABLE apoteka_stanje (
      id_apoteka        INTEGER NOT NULL,
      id_terapija       INTEGER NOT NULL,
      kolicina          INTEGER NOT NULL,
 FOREIGN KEY (id_apoteka) REFERENCES apoteka(id) ON DELETE CASCADE,
 FOREIGN KEY (id_terapija) REFERENCES terapija(id) ON DELETE CASCADE,
 PRIMARY KEY (id_apoteka, id_terapija),
 CHECK (kolicina >= 0)
);
```

Primarni ključ tablice ***apoteka_stanje*** se sastoji od dva strana ključa (**id_apoteka, id_terapija**): primarni ključevi tablica ***apoteka*** i ***terapija*** (INTEGER).
Cilj ove tablice je spojiti terapiju s apotekom u kojoj je ona dostupna te prikazati dostupnu količinu (INTEGER).

3.10. pacijent_dijagnoza

```
CREATE TABLE pacijent_dijagnoza (
      id_pacijent        INTEGER NOT NULL,
      id_lijecnik        INTEGER NOT NULL,
      id_dijagnoza       INTEGER NOT NULL,
      id_terapija        INTEGER,
FOREIGN KEY (id_pacijent) REFERENCES osoba(id) ON DELETE CASCADE,
FOREIGN KEY (id_lijecnik) REFERENCES lijecnik(id) ON DELETE CASCADE,
FOREIGN KEY (id_dijagnoza) REFERENCES dijagnoza(id) ON DELETE CASCADE,
FOREIGN KEY (id_terapija) REFERENCES terapija(id) ON DELETE SET NULL,
PRIMARY KEY (id_pacijent, id_lijecnik, id_dijagnoza)
);
```

Primarni ključ tablice ***pacijent_dijagnoza*** (**id_pacijent, id_lijecnik, id_dijagnoza**) čine strani ključevi koji su primarni u tablicama ***pacijent***, ***lijecnik*** i ***dijagnoza*** (INTEGER).
Ovaj kombinirani ključ određuje terapiju (**id_terapija** - INTEGER) koju je liječnik odredio pacijentu s određenom dijagnozom.
Terapija ne mora biti i dodijeljena (može imati vrijednost NULL), a može biti poništena zbog odluke liječnika ili pacijenta, ali i zbog uklanjanja te terapije iz prodaje.
Redak tablice izbrisat će se ukoliko se iz baze uklone pacijent, liječnik ili dijagnoza.

3.11. evidencija_smrti

```
CREATE TABLE evidencija_smrti (
      id_pacijent       INTEGER NOT NULL,
      razlog_smrti      INTEGER NOT NULL,
      datum_smrti       DATE NOT NULL,
FOREIGN KEY (id_pacijent) REFERENCES osoba(id) ON DELETE CASCADE,
FOREIGN KEY (razlog_smrti) REFERENCES dijagnoza(id),
PRIMARY KEY (id_pacijent, razlog_smrti),
UNIQUE (id_pacijent, datum_smrti)
);
```

Tablica ***evidencija_smrti*** sadrži podatke o osobama iz baze koje su umrle.
Kao primarni ključ koristi se **id_pacijent** (INTEGER), pošto pacijent može umrijeti samo jedan put.
Ključ određuje razlog (dijagnozu - INTEGER) i datum smrti (DATE).

4. Poslovna pravila

4.1. TRIGGER

- liječnik **ne smije** postaviti dijagnozu sebi
- potrebne su dvije tablice za ovo ograničenje te je korišten **TRIGGER** koji će, nakon pokušaja umetanja u tablicu (ukoliko je vrijednost atributa **id_osoba** iz tablice ***lijecnik*** jednaka vrijednosti **id_pacijent** koji želimo unijeti), poništiti upis te ispisati odgovarajuću poruku

```
CREATE TRIGGER istaOsoba BEFORE INSERT ON pacijent_dijagnoza
FOR EACH ROW
BEGIN
      IF (SELECT id_osoba FROM lijecnik
            WHERE id = new.id_lijecnik) = new.id_pacijent
                 THEN SIGNAL SQLSTATE '45000'
                 SET MESSAGE_TEXT = 'Lijecnik ne moze sam sebi
postaviti dijagnozu!';
      END IF;
END;
```

4.2. Strani ključevi i kombinirani primarni

4.2.1. bolnica

- ukoliko se grad izbriše iz baze, bolnica se također **briše**

```
FOREIGN KEY (id_grad) REFERENCES grad(id) ON DELETE CASCADE
```

4.2.2. apoteka

- ukoliko se grad izbriše iz baze, apoteka se također **briše**

```
FOREIGN KEY (id_grad) REFERENCES grad(id) ON DELETE CASCADE
```

4.2.3. osoba

- ukoliko se grad izbriše iz baze, osoba gubi mjesto stanovanja, atribut poprima vrijednost **NULL**

```
FOREIGN KEY (mjesto_stan) REFERENCES grad(id) ON DELETE SET NULL
```

4.2.4. lijecnik

- pri brisanju osobe (**id_osoba**) iz baze, **briše se** i podatak o liječniku koji je ta osoba
- pri brisanju bolnice, liječnik postaje nezaposlen (**NULL**)
- ukoliko se promijeni **specijalizacija.id**, mijenja se i **id_spec** u tablici ***lijecnik***

```
FOREIGN KEY (id_osoba) REFERENCES osoba(id) ON DELETE CASCADE,
FOREIGN KEY (id_bolnica) REFERENCES bolnica(id) ON DELETE SET NULL,
FOREIGN KEY (id_spec) REFERENCES specijalizacija(id) ON UPDATE CASCADE
```

4.2.5. apoteka_stanje

- ukoliko dođe do brisanja apoteke ili terapije, **brišu** se podaci koji su sadržavali njihov id
- primarni ključ čine dva strana ključa te se time omogućuje **N-N način povezivanja** tablica apoteka i terapija

```
FOREIGN KEY (id_apoteka) REFERENCES apoteka(id) ON DELETE CASCADE,
FOREIGN KEY (id_terapija) REFERENCES terapija(id) ON DELETE CASCADE,
PRIMARY KEY (id_apoteka, id_terapija)
```

4.2.6. *dijagnoza_pacijent*

- ukoliko dođe do brisanja pacijenta, liječnika ili dijagnoze, svi podaci koji su ih sadržavali se **brišu**
- ukoliko se terapija izbriše iz sustava, svi pacijenti koji su ju koristili ostaju bez zadane terapije (**NULL**)
- primarni ključ sastavljen od tri strana ključa čini **ternarnu vezu** čiji je rezultat terapija

```
FOREIGN KEY (id_pacijent) REFERENCES osoba(id) ON DELETE CASCADE,
FOREIGN KEY (id_lijecnik) REFERENCES lijecnik(id) ON DELETE CASCADE,
FOREIGN KEY (id_dijagnoza) REFERENCES dijagnoza(id) ON DELETE CASCADE,
FOREIGN KEY (id_terapija) REFERENCES terapija(id) ON DELETE SET NULL,
PRIMARY KEY (id_pacijent, id_lijecnik, id_dijagnoza)
```

4.2.7. *evidencija_smrti*

- ukoliko se osoba, čiji je id strani ključ tablice, izbriše iz baze, tada se **briše** i podatak o smrti
- razlog smrti **mora** biti postojeća dijagnoza
- strani ključ **id_pacijent** je također i primarni, jedan pacijent ne može umrijeti više od jedanput

```
FOREIGN KEY (id_pacijent) REFERENCES osoba(id) ON DELETE CASCADE,
FOREIGN KEY (razlog_smrti) REFERENCES dijagnoza(id),
PRIMARY KEY (id_pacijent)
```

4.3. CHECK

4.3.1. *grad*

- poštanski broj **mora** imati točno 5 znamenki

```
CHECK (length(post_broj) = 5)
```

4.3.2. *osoba*

- mbo **mora** imati točno 9 znamenki
- dozvoljene vrijednosti za spol su **‘M’** i **‘Ž’**

```
CHECK (length(mbo) = 9),
CHECK (spol IN ('M', 'Ž'))
```

4.3.3. *terapija*

- iznos doplate je minimalno 0,00kn

```
CHECK (doplata >= 0)
```

4.3.4. *apoteka_stanje*

- količina neke terapije u apoteci ne može biti negativna

```
CHECK (kolicina >= 0)
```

4.4. UNIQUE

4.4.1. *grad*

- poštanski broj je prirodni ključ pa se ne smije ponavljati

```
post_broj   CHAR(5) UNIQUE NOT NULL
```

4.4.2. *osoba*

- mbo je prirodni ključ pa se ne smije ponavljati

```
mbo         CHAR(9) UNIQUE NOT NULL
```

4.4.3. *lijecnik*

- jedan liječnik može biti samo jedna osoba

```
id_osoba    INTEGER UNIQUE NOT NULL
```

5. Upiti

5.1. SELECT

5.1.1. Tablica *dijagnoza*

- Ispisuje kompletan sadržaj tablice ***dijagnoza***.

```
SELECT * FROM dijagnoza;
```

5.1.2. Liječnici

- Ispisuje sve liječnike i sve informacije o njima sadržane u tablicama ***osoba*** i ***lijecnik***.

```
SELECT *
      FROM osoba
      JOIN lijecnik ON osoba.id = lijecnik.id_osoba;
```

5.1.3. Osobe u dobi od 60 i više godina

- Ispisuje sve osobne podatke osoba iz tablice ***osoba*** koje su navršile 60 godina života.

```
SELECT *
    FROM osoba
    WHERE DATEDIFF(CURDATE(), datum_rodjenja) > 60*365;
```

5.1.4. Dobi preminulih

- Ispisuje sve preminule osobe i njihove navršene godine života.

```
SELECT *, CAST(DATEDIFF(datum_smrti, datum_rodjenja)/365 AS UNSIGNED) dob
      FROM evidencija_smrti AS evi_smrt
      JOIN osoba ON osoba.id = evi_smrt.id_pacijent;
```

5.1.5. Specijalističko osoblje bolnice

- Ispisuje sve podatke iz tablice Osoba liječnika koji su zaposleni u traženoj zdravstvenoj ustanovi (u ovom slučaju “Klinički bolnički centar Zagreb”).

```
SELECT osoba.*
    FROM osoba
    JOIN lijecnik AS dok ON osoba.id = dok.id_osoba
    JOIN bolnica ON bolnica.id = dok.id_bolnica
    WHERE bolnica.naziv = 'Klinički bolnički centar Zagreb';
```

5.1.6. Liječnici u gradu

- Ispisuje sve osobne podatke liječnika koji su zaposleni u traženom gradu (dani primjer je “Zagreb”) te naziv bolnice u kojoj rade.

```
SELECT *
    FROM osoba
    JOIN lijecnik ON osoba.id = lijecnik.id_osoba
    JOIN bolnica ON bolnica.id = lijecnik.id_bolnica
    JOIN grad ON grad.id = bolnica.id_grad
    WHERE grad.naziv = 'Zagreb';
```

5.1.7. Liječnici koji trebaju odobrenje nadređenog za nastavak rada

- Ispisuje informacije o liječnicima koji su navršili 65 godina života.

```
SELECT bol.naziv, COUNT(*) AS broj_specijalista
    FROM bolnica AS bol
    JOIN lijecnik ON bol.id = lijecnik.id_bolnica
    GROUP BY bol.id
    ORDER BY broj_specijalista ASC;
```

5.1.8. Opskrbljenost bolnica

- Ispisuje opskrbljenost svih bolnica koje imaju osoblje u rastućem slijedu.

```
SELECT bol.naziv, COUNT(*) AS broj_specijalista
    FROM bolnica AS bol
    JOIN lijecnik ON bol.id = lijecnik.id_bolnica
    GROUP BY bol.id
    ORDER BY broj_specijalista ASC;
```

5.1.9. Nedostajuće terapije

- Ispisuje sve informacije iz tablice ***terapija*** te njihovu ukupnu količinu onih terapija kojih u svim apotekama u sustavu sveukupno ima manje od 50 komada.

```
SELECT ter.*, COALESCE(SUM(kolicina), 0) AS ukupna_kolicina
      FROM terapija AS ter
      LEFT JOIN apoteka_stanje ON apoteka_stanje.id_terapija = ter.id
      GROUP BY terapija.id
      HAVING ukupna_kolicina < 50
      ORDER BY ukupna_kolicina ASC;
```

5.1.10. Tretiranje dijagnoza pacijenata

- Vraća dva stupca: broj osoba koje za svoju dijagnozu primaju terapiju i broj osoba koje ne primaju terapiju za svoju dijagnozu

```
SELECT * FROM (
            SELECT COUNT(*) AS pacijenti_bez_terapije
                FROM pacijent_dijagnoza
                WHERE id_terapija IS NULL
      ) AS broj_pacijenata_bez_terapije, (
            SELECT COUNT(*) AS pacijenti_sa_terapijom
                FROM pacijent_dijagnoza
                WHERE id_terapija IS NOT NULL
      ) AS broj_pacijenata_sa_terapijom;
```

5.1.11. Liječnici i terapije

- Pronalazi liječnike koji dobivaju određenu terapiju (u ovom slučaju “Aerius”).

```
SELECT osoba_dok.*, dok.*, ter.id, ter.naziv, ter.jacina_sadrzaj
    FROM osoba AS osoba_dok
    JOIN lijecnik AS dok ON osoba_dok.id = dok.id_osoba
    JOIN pacijent_dijagnoza AS pd ON pd.id_pacijent = dok.id_osoba
    JOIN terapija AS ter ON pd.id_terapija = ter.id
    WHERE ter.naziv = 'Aerius';
```

5.1.12. Kritične dijagnoze

- Ispisuje 3 bolesti koje su dijagnosticirane kod najvećeg broja ljudi koji su preminuli u tekućem mjesecu.

```
SELECT dijagnoza.*, COUNT(*) AS broj_preminulih
    FROM dijagnoza AS di
    JOIN evidencija_smrti AS ev_smrt ON di.id = ev_smrt.razlog_smrti
    WHERE MONTH(ev_smrt.datum_smrti) = MONTH(CURDATE())
    GROUP BY di.id
    ORDER BY broj_preminulih DESC
    LIMIT 3;
```

5.1.13. Deficitarne specijalizacije

- Ispisuje specijalizacije koje imaju manje od 5 specijalista.

```
SELECT spec.*, COALESCE(COUNT(dok.id), 0) AS broj_spec
      FROM specijalizacija AS spec
      LEFT JOIN lijecnik AS dok ON dok.id_spec = spec.id
      GROUP BY spec.id
      HAVING broj_spec < 5
      ORDER BY broj_spec;
```

5.1.14. Dostupnost terapije

- Ispisuje apoteke u kojima je dostupna tražena terapija (u ovom slučaju “Ofev”).

```
SELECT apo.*, ap_st.kolicina
    FROM apoteka AS apo
    JOIN apoteka_stanje AS ap_st ON apo.id = ap_st.id_apoteka
    JOIN terapija AS ter ON ap_st.id_terapija = ter.id
    WHERE ter.naziv = 'Ofev'
    ORDER BY kolicina DESC;
```

5.1.15. Povijest pacijenata liječnika

- Ispisuje sve osobe kojima je traženi liječnik (u ovom slučaju “Karlo Mikulić”) barem jednom postavio dijagnozu, neovisno o tome jesu li ti pacijenti preminuli.

```
SELECT DISTINCT dok.id, pac.*
      FROM osoba AS osoba_dok
      JOIN lijecnik AS dok ON osoba_dok.id = dok.id_osoba
      JOIN pacijent_dijagnoza AS pd ON dok.id = pd.id_lijecnik
      JOIN osoba AS pac ON pac.id = pd.id_pacijent
      WHERE osoba_dok.ime = 'Karlo' AND osoba_dok.prezime = 'Mikulić';
```

5.1.16. Gradovi kandidati za gradnju nove bolnice

- Tablica 10 gradova s najviše stanovnika koji nemaju bolnicu.

```
SELECT grad.*, COALESCE(COUNT(osoba.id), 0) AS broj_stanovnika
      FROM grad
      LEFT JOIN osoba ON grad.id = osoba.mjesto_stan
      GROUP BY grad.id
      HAVING grad.id NOT IN (SELECT id_grad FROM bolnica)
      ORDER BY broj_stanovnika DESC
      LIMIT 10;
```

5.1.17. Kritični liječnici

- Pronalazi liječnike kojima je dijagnosticirana određena bolest, zbog diskretnosti koristimo Id dijagnoze (u ovom slučaju “567”).

```
SELECT grad.*, COALESCE(COUNT(osoba.id), 0) AS broj_stanovnika
      FROM grad
      LEFT JOIN osoba ON grad.id = osoba.mjesto_stan
      GROUP BY grad.id
      HAVING grad.id NOT IN (SELECT id_grad FROM bolnica)
      ORDER BY broj_stanovnika DESC
      LIMIT 10;
```

5.1.18. Praćenje godišnjih smrti

- Vraća kalendarsku godinu, broj umrlih u toj godini te prosječnu navršenu starosnu dob umrlih.

```
SELECT YEAR(datum_smrti) AS godina_smrti,
COUNT(*) AS broj_umrlih,
CAST(prosjecna_dob_umrlih AS UNSIGNED) prosjecna_starost_umrlih
    FROM evidencija_smrti
    JOIN (SELECT YEAR(datum_smrti) AS god_smrt,
AVG(DATEDIFF(datum_smrti, datum_rodjenja) / 365) AS prosjecna_dob_umrlih
        FROM evidencija_smrti
        JOIN osoba ON osoba.id = evidencija_smrti.id_pacijent
        GROUP BY YEAR(datum_smrti))
AS pros_dob_umrli ON pros_dob_umrli.god_smrt = YEAR(datum_smrti)
    GROUP BY godina_smrti;
```

5.1.19. Deset najzauzetijih liječnika

- Lista 10 liječnika koji su dijagnosticirali najviše osoba u svojoj karijeri.

```
SELECT osoba_dok.*, COUNT(pac.id) AS broj_pacijenata
    FROM osoba AS osoba_dok
    JOIN lijecnik AS dok ON osoba_dok.id = dok.id_osoba
    JOIN pacijent_dijagnoza AS pd ON dok.id = pd.id_lijecnik
    JOIN osoba AS pac ON pac.id = pd.id_pacijent
    GROUP BY dok.id
    ORDER BY broj_pacijenata DESC
```

5.1.20. Kritična smrtnost pacijenata

- Pronalazi liječnike kojima je u posljednjih godinu dana preminulo više od 20 pacijenata.

```
SELECT osoba_dok.*, COUNT(osoba_pac.id) AS broj_preminulih_pacijenata
    FROM lijecnik AS dok
    JOIN osoba AS osoba_dok ON osoba_dok.id = dok.id_osoba
    JOIN pacijent_dijagnoza AS pd ON pd.id_lijecnik = dok.id
    JOIN osoba AS osoba_pac ON osoba_pac.id = pd.id_pacijent
    JOIN evidencija_smrti AS evid_smrt ON osoba_pac.id =
evid_smrt.id_pacijent
    WHERE DATEDIFF(CURDATE(), datum_smrti) < 365
    GROUP BY dok.id
    HAVING broj_preminulih_pacijenata > 20;
```

5.1.21. Povijest bolesti

- Povijest pacijenta, dijagnoza, liječnik, terapija

```
SELECT
    oso.ime,
    oso.prezime,
    dij.naziv,
    ter.naziv,
    ter.jacina_sadrzaj
FROM
    osoba oso
        LEFT JOIN
    pacijent_dijagnoza p_dij ON p_dij.id_pacijent = oso.id
        LEFT JOIN
    terapija ter ON ter.id = p_dij.id_terapija
        LEFT JOIN
    dijagnoza dij ON dij.id = p_dij.id_dijagnoza
WHERE
    p_dij.id_dijagnoza IS NOT NULL;
```

5.1.22. Broj preminulih u posljednjih mjesec dana

```
SELECT
    COUNT(*) br_umrlih
FROM
    evidencija_smrti
WHERE
    datum_smrti > DATE_SUB(NOW(), INTERVAL 1 MONTH);
```

5.1.23. Prosječni broj umrlih

- Koliko ljudi prosječno umre svaki mjesec, u zadnjih godinu dana

```
SELECT
    ROUND(AVG(po_mj))
FROM
    (SELECT
        COUNT(*) po_mj
    FROM
        evidencija_smrti
    GROUP BY MONTH(datum_smrti)) mj;
```

5.1.24. Mjesec s najviše preminulih

```
SELECT
    MONTH(datum_smrti) mjesec, COUNT(*) br_preminulih
FROM
    evidencija_smrti
GROUP BY MONTH(datum_smrti)
ORDER BY br_preminulih DESC
LIMIT 1;
```

5.1.25. Postavljenost dijagnoze

- Je li osobi postavljena dijagnoza?

```
SELECT
    o.ime,
    o.prezime,
    CASE
        WHEN id_dijagnoza IS NULL THEN 'Nije'
        ELSE 'Je'
    END dijagnoza_postavljena
FROM
    osoba o
        LEFT JOIN
    pacijent_dijagnoza pd ON pd.id_pacijent = o.id;
```

5.1.26. Liječnik i pacijent koji žive u istom gradu

```
SELECT
    o.ime ime_lijecnik,
    o.prezime prezime_lijecnik,
    pac.ime ime_pacijent,
    pac.prezime prezime_pacijent,
    grad.naziv
FROM
    pacijent_dijagnoza p
        LEFT JOIN
    lijecnik l ON l.id = p.id_lijecnik
        LEFT JOIN
    osoba o ON l.id_osoba = o.id
        LEFT JOIN
    grad ON grad.id = o.mjesto_stan
        LEFT JOIN
    osoba pac USING (mjesto_stan);
```

5.1.27. Drugi po redu najgori liječnik

- Drugi po redu liječnik s najviše umrlih pacijenata

```
SELECT
    lij.ime ime_lijecnik, lij.prezime prezime_lijecnik
FROM
    evidencija_smrti es
        LEFT JOIN
    osoba o ON o.id = es.id_pacijent
        LEFT JOIN
    pacijent_dijagnoza pd ON pd.id_pacijent = o.id
        LEFT JOIN
    lijecnik l ON l.id = pd.id_lijecnik
        LEFT JOIN
    osoba lij ON l.id_osoba = lij.id
GROUP BY lij.id
ORDER BY COUNT(*) DESC
LIMIT 1 , 1
```

5.1.28. Broj terapija koje primaju osobe

- Koliko terapija primaju osobe koje primaju vise od jedne terapije?

```
SELECT
    o.ime ime_pacijent,
    o.prezime prezime_pacijent,
    CASE COUNT(*)
        WHEN
            (SELECT
                    COUNT(*) max
                FROM
                    pacijent_dijagnoza
                GROUP BY id_pacijent
                HAVING COUNT(*) >= 2
                ORDER BY COUNT(*) DESC
                LIMIT 1)
        THEN
            'Najvise'
        WHEN
            (SELECT
                    COUNT(*) min
                FROM
                    pacijent_dijagnoza
                GROUP BY id_pacijent
                HAVING COUNT(*) >= 2
                ORDER BY COUNT(*) ASC
                LIMIT 1)
        THEN
            'Najmanje'
        ELSE 'Sredina'
    END AS test
FROM
    pacijent_dijagnoza pd
        LEFT JOIN
    osoba o ON o.id = pd.id_pacijent
GROUP BY id_pacijent
HAVING COUNT(*) >= 2;
```

5.2. INSERT INTO

5.2.1. Novi specijalist

- Omogućuje nam unos novog liječnika u bazu.

```
INSERT INTO lijecnik VALUES (10099, 50033, 22005, 143);
```

5.2.2. Unos novog specijalista, uz prethodnu provjeru

```
SELECT
    *
FROM
    lijecnik
WHERE
    id_osoba IN (SELECT
            id
        FROM
            osoba
        WHERE
            ime = 'Lukas' AND prezime = 'Rusac');
            -- nije, sad insert

INSERT INTO lijecnik VALUES   (
            (select max(id)+1 from lijecnik l1),
            select id from osoba where ime='Lukas' and prezime='Rusac'),
            null,
            (select id from specijalizacija where naziv='Neurologija')
);
```

5.3. UPDATE

5.3.1. Promjena cijene terapije

- Omogućuje nam promjenu cijene određene terapije ili određenog skupa terapija (u ovom slučaju smo povećali cijenu svih terapija koje se naplaćuju za 5%)

```
UPDATE terapija
      SET doplata = doplata + (0.05 * doplata)
      WHERE id IN (SELECT *
                      FROM (SELECT id
                              FROM terapija
                              WHERE doplata > 0) AS terapija_doplata);
```

5.3.2. Promjena imena

- Dodaj 'R.I.P.' pored imena umrlih

```
UPDATE osoba
SET
    ime = CONCAT(ime, ' R.I.P.')
WHERE
    EXISTS( SELECT
            id_pacijent
        FROM
            evidencija_smrti es
        WHERE
            es.id_pacijent = osoba.id);
```

5.4. DELETE

5.4.1. Otpuštanje liječnika neprikladnih za nastavak rada

- Omogućuje nam brisanje iz sustava liječnika koji ne zadovoljavaju neki kriterij (u ovom slučaju su to liječnici koji su navršili 80 godina života).

```
DELETE FROM lijecnik
    WHERE id_osoba
    IN (SELECT id
          FROM osoba
          WHERE DATEDIFF(CURDATE(), datum_rodjenja) > 80*365);
```

5.4.2. Brisanje određenog liječnika

- Brisanje liječnika Lukas Rusac

```
DELETE FROM lijecnik
WHERE
    id_osoba IN (SELECT
        id
    FROM
        osoba
    WHERE
        ime = 'Lukas' AND prezime = 'Rusac');
```

5.5. CREATE VIEW

5.5.1. Bolnice koje bismo trebali zatvoriti

- Pogled koji nam prikazuje sve bolnice koje nemaju osoblja.

```
CREATE VIEW bolnice_bez_osoblja AS
SELECT bolnica.*, COALESCE(COUNT(lijecnik.id), 0) AS broj_specijalista
      FROM bolnica
      LEFT JOIN lijecnik ON bolnica.id = lijecnik.id_bolnica
      GROUP BY bolnica.id
      HAVING broj_specijalista = 0
      ORDER BY broj_specijalista ASC;
```

5.5.2. Osobe koje nisu liječnici

- Pregled svih osoba koje nisu liječnici, provjera i unos podataka u view

```
CREATE VIEW not_lijecnik AS
    (SELECT
        *
    FROM
        osoba
    WHERE
        id NOT IN (SELECT
                id_osoba
            FROM
                lijecnik)) WITH CHECK OPTION;
            -- Provjera
SELECT
    *
FROM
    not_lijecnik;

INSERT INTO not_lijecnik VALUES
(50200,'ime11','prezime22','M',str_to_date('11.3.1994','%d.%m.%Y'),'123456
789',49000);
```

6. Zaključak

Najzahtjevniji dio projekta je bilo usuglašavanje svih sudionika u timu, obzirom na okolnosti socijalnog distanciranja koje su nas snašle. Mišljenja smo da je ovaj prisilni rad na daljinu bio dobra vježba za našu buduću karijeru.
Ipak, zaključili smo da oba načina rada (uživo i na daljinu) na zajedničkom projektu imaju svoje prednosti.

Smatramo da smo stvorili bazu koja funkcionira u svojoj sadašnjoj formi ali ima i još mnogo mjesta za napredak i uvođenje novih funkcionalnosti što je primarno uvjetovano našim ograničenim poznanstvom sa izgradnjom ovakve baze.

Trebalo bi dodati neke kompliciranije provjere kako bi ovakva baza bila od koristi stvarnom sustavu medicine te velik broj novih relacija koje bi omogućile kreiranje novih ograničenja.

Primjerice, trebalo bi u našu bazu dodati ograničenje za kombinacije dijagnoze i terapije, odnosno, da se neke terapije ne smiju davati pacijentu s određenom dijagnozom pošto je time moguće proizvesti kontra-efekt.

Također bi se trebalo onemogućiti da pedijatar postavlja dijagnozu osobama koje su punoljetne.

Pokrili smo velik broj ograničenja i pravila, no uvijek postoji mjesta za napredak.

Od tehnologija smo koristili:

- za komunikaciju: Slack i Discord
- za Cloud Storage: Slack i Google Disk
- za organizaciju tablica: Google Sheets
- za pisanje koda: MySQL Workbench
- za crtanje dijagrama: Lucidchart
- za pisanje dokumentacije: Microsoft Word i Google Docs