SEMINARSKA NALOGA

PODATKOVNE BAZE 2

COMEDY PILL

|  |  |
| --- | --- |
| Datum: 28.Decembar 2020 | Avtor:  Sadmir Hasanić |

Kazalo vsebine

[PODROBEN OPIS TEME 5](#_Toc60022344)

[E-R DIAGRAM 5](#_Toc60022345)

[Funckije za dodajanje, brisanje in spreminjanje podatkov 5](#_Toc60022346)

[Entitea drzave 5](#_Toc60022347)

[Entiteta kategorije 7](#_Toc60022348)

[Entiteta knjiznica\_specijalov 8](#_Toc60022349)

[Entiteta komiki 9](#_Toc60022350)

[Entiteta najljubsi 10](#_Toc60022351)

[Entiteta ogledi 11](#_Toc60022352)

[Entiteta osebe 12](#_Toc60022353)

[Entiteta popusti 14](#_Toc60022354)

[Eniteta popusti\_specijali 15](#_Toc60022355)

[Entiteta postne\_stevilke 16](#_Toc60022356)

[Entiteta racuni\_izdelki 17](#_Toc60022357)

[Entiteta racuni 19](#_Toc60022358)

[Entiteta slike 20](#_Toc60022359)

[Entiteta specijali 22](#_Toc60022360)

[Entiteta traileri 23](#_Toc60022361)

[Entiteta uporabniki 25](#_Toc60022362)

[Entiteta vrste\_placil 26](#_Toc60022363)

[Entiteta zasluzki 27](#_Toc60022364)

[Triggers funkcije 29](#_Toc60022365)

[Trigger 1 countries 29](#_Toc60022366)

[Trigger 2 countries 29](#_Toc60022367)

[Strežniške funkcije za obdelave in izračun statistike 30](#_Toc60022368)

[Testno polnjenje, posodabljanje,brisanje in preverjanje podatkov podatkovne baze 31](#_Toc60022369)

[Entiteta countries 31](#_Toc60022370)

[Entiteta roles 31](#_Toc60022371)

[Entiteta users 31](#_Toc60022372)

[Entiteta categories 32](#_Toc60022373)

[Entiteta discounts 32](#_Toc60022374)

[Entiteta products 32](#_Toc60022375)

[Entiteta shipping\_address 33](#_Toc60022376)

[Entiteta tags 33](#_Toc60022377)

[Entiteta products\_tags 33](#_Toc60022378)

[Entiteta products\_images 33](#_Toc60022379)

[Entiteta users\_shipping\_addresses 33](#_Toc60022380)

[Entiteta ratings 33](#_Toc60022381)

[Entiteta interests 33](#_Toc60022382)

[Entiteta users\_interests 33](#_Toc60022383)

[Entiteta replies 33](#_Toc60022384)

[Entiteta comments 34](#_Toc60022385)

[Entiteta orders 34](#_Toc60022386)

[Entiteta orderitems 34](#_Toc60022387)

[Entiteta complaints 34](#_Toc60022388)

[Entiteta option\_groups 34](#_Toc60022389)

[Entiteta options 34](#_Toc60022390)

[Entiteta produts\_options 34](#_Toc60022391)

[Entiteta products physical 34](#_Toc60022392)

[Entiteta products\_physical\_products\_options 35](#_Toc60022393)

[Rezultati poizvedb statističnih funkcij 36](#_Toc60022394)

[Entiteta countries 36](#_Toc60022395)

[Entiteta roles 36](#_Toc60022396)

[Entiteta users 36](#_Toc60022397)

[Entiteta categories 36](#_Toc60022398)

[Entiteta discounts 36](#_Toc60022399)

[Entiteta products 37](#_Toc60022400)

[Entiteta shipping\_address 37](#_Toc60022401)

[Entiteta tags 37](#_Toc60022402)

[Entiteta products\_tags 37](#_Toc60022403)

[Entiteta products\_images 37](#_Toc60022404)

[Entiteta users\_shipping\_addresses 37](#_Toc60022405)

[Entiteta ratings 37](#_Toc60022406)

[Entiteta interests 37](#_Toc60022407)

[Entiteta users\_interests 37](#_Toc60022408)

[Entiteta replies 38](#_Toc60022409)

[Entiteta comments 38](#_Toc60022410)

[Entiteta orders 38](#_Toc60022411)

[Entiteta orderitems 38](#_Toc60022412)

[Entiteta complaints 38](#_Toc60022413)

[Entiteta option\_groups 38](#_Toc60022414)

[Entiteta options 38](#_Toc60022415)

[Entiteta produts\_options 38](#_Toc60022416)

[Entiteta products physical 38](#_Toc60022417)

[Entiteta products\_physical\_products\_options 39](#_Toc60022418)

[Zaključek 40](#_Toc60022419)

[Izjava o samostojnem delu 41](#_Toc60022420)

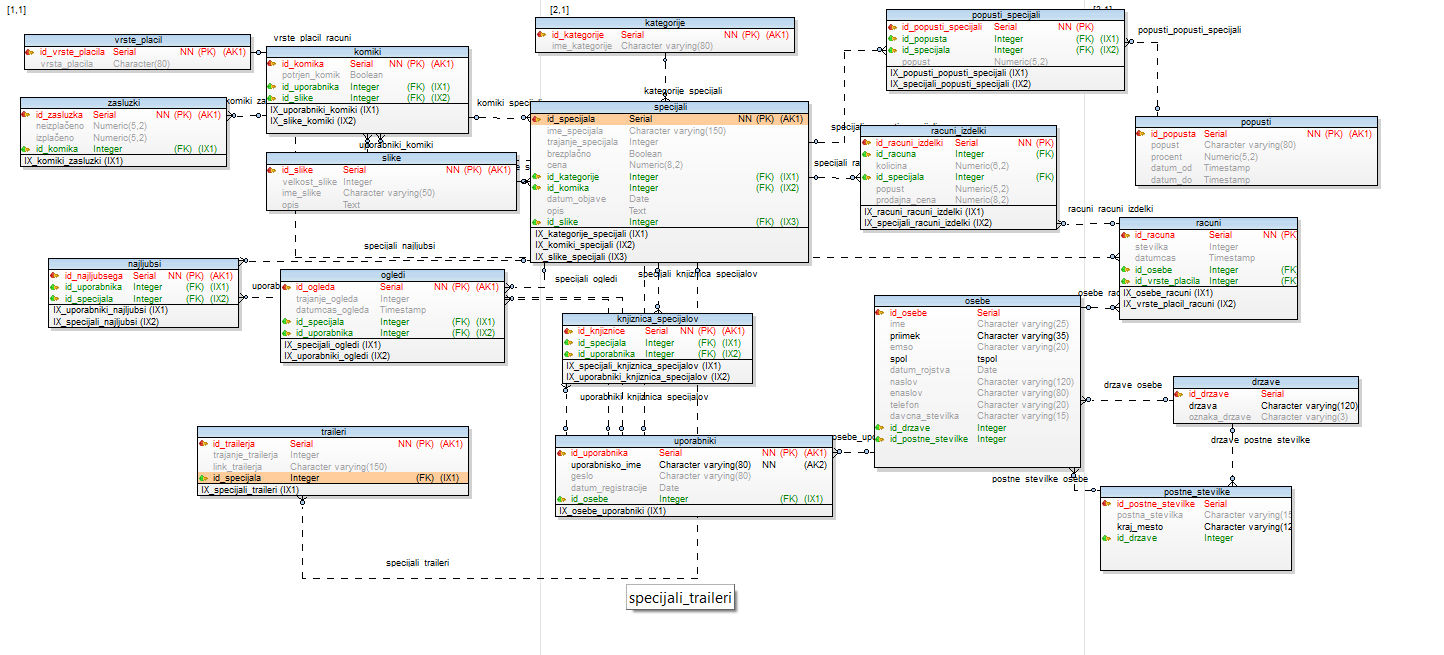
# PODROBEN OPIS TEME

Comedy Pill naj bi bila spletna stran na kateri bi stand up komiki lahko prodajali svoje stand up specijale. Sepravi vsak se lahko registrira na spletno stran oz. v podatkovno bazo ampak da bi objavljal specijale mora biti odobren od strani admina, za to obstaja tudi funckija pod nazivom 'potrjevanje\_komika'.

Vsak specijal pripada doloceni kategoriji. Obstajao tudi popusti ki so povezani na dolocene specijale in se avtomatsko vrednost racuna posodobi pri nakupu specijala.

V podatkovni bazi se evidentirajo tudi ogledi, zaradi določanja relevantnosti posameznega specijala kar bi pomagalo pri prikazu na zacetni strani spletne aplikacije.

# E-R DIAGRAM



# Funckije za dodajanje, brisanje in spreminjanje podatkov

## Entitea drzave

create or replace function drzave\_iu(i\_id\_drzave integer, i\_drzava character varying, i\_oznaka\_drzave character varying(3))

RETURNS integer as

$$

    DECLARE

        kljuc integer;

BEGIN

        -- INSERT FUNCTION

        IF(i\_id\_drzave IS NULL ) THEN

            BEGIN

                kljuc = nextval('drzave\_id\_drzave\_seq');

                INSERT INTO drzave (id\_drzave,drzava, oznaka\_drzave)

                VALUES (kljuc,i\_drzava,i\_oznaka\_drzave);

                EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

            end;

        ELSE

            BEGIN

            --UPDATE function

            UPDATE drzave

            SET drzava = i\_drzava,

                oznaka\_drzave = i\_oznaka\_drzave

            WHERE id\_drzave = i\_id\_drzave;

            kljuc = i\_id\_drzave;

            --exceptions

            EXCEPTION

            WHEN integrity\_constraint\_violation THEN

                RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

            WHEN not\_null\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

            WHEN foreign\_key\_violation THEN

                RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

            WHEN unique\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

            WHEN check\_violation THEN

                RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

            WHEN others THEN

                RAISE EXCEPTION 'Napaka ...';

            END;

        END IF;

    RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM drzave\_iu(null,'Madjarska','MDR');

SELECT \* FROM drzave\_iu(11,'Madarska','MD');

## Entiteta kategorije

create or replace function kategorije\_iu(i\_id\_kategorije integer, i\_ime\_kategorije character varying)

RETURNS integer as

$$ DECLARE kljuc integer;

BEGIN

        -- INSERT FUNCTION

        IF(i\_id\_kategorije IS NULL ) THEN

            BEGIN

                kljuc = nextval('kategorije\_id\_kategorije\_seq');

                INSERT INTO kategorije (id\_kategorije, ime\_kategorije)

                VALUES (kljuc,i\_ime\_kategorije);

                EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

            end;

        ELSE

            BEGIN

            --UPDATE function

            UPDATE kategorije

            SET ime\_kategorije = i\_ime\_kategorije

            WHERE id\_kategorije = i\_id\_kategorije;

            kljuc = i\_id\_kategorije;

            --exceptions

            EXCEPTION

            WHEN integrity\_constraint\_violation THEN

                RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

            WHEN not\_null\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

            WHEN foreign\_key\_violation THEN

                RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

            WHEN unique\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

            WHEN check\_violation THEN

                RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

            WHEN others THEN

                RAISE EXCEPTION 'Napaka ...';

            END;

        END IF;

    RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM kategorije\_iu(null,'puns');

## Entiteta knjiznica\_specijalov

CREATE FUNCTION knjiznica\_specijalov\_iu(i\_id\_knjiznice\_specijalov integer,i\_id\_specijala integer,i\_id\_uporabnika integer)

RETURNS integer as

$$

DECLARE

    kljuc integer;

BEGIN

   IF (i\_id\_knjiznice\_specijalov IS NULL ) THEN

       BEGIN

          kljuc=nextval('knjiznica\_specijalov\_id\_knjiznice\_seq');

          INSERT INTO knjiznica\_specijalov (id\_knjiznice, id\_specijala, id\_uporabnika)

          VALUES (kljuc,i\_id\_specijala,i\_id\_uporabnika);

          EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

       END;

   ELSE

       BEGIN

           UPDATE knjiznica\_specijalov

           SET id\_specijala = i\_id\_specijala,

               id\_uporabnika = i\_id\_uporabnika

           WHERE id\_knjiznice = i\_id\_knjiznice\_specijalov;

           kljuc=i\_id\_knjiznice\_specijalov;

           EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

       END;

   END IF;

   RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT FROM knjiznica\_specijalov\_iu(null,12,15);

SELECT FROM knjiznica\_specijalov\_iu(1,1,15);

## Entiteta komiki

    create or replace function komiki\_iu(i\_id\_komika integer,i\_potrjen\_komik boolean,i\_id\_uporabnika integer,i\_id\_slike integer)

    RETURNS integer as

    $$

    DECLARE

        kljuc integer;

    BEGIN

            --INSERT funckija

            IF(i\_id\_komika IS NULL) THEN

                BEGIN

                    kljuc = nextval('komiki\_id\_komika\_seq');

                    INSERT INTO komiki (id\_komika, potrjen\_komik, id\_uporabnika, id\_slike)

                    VALUES (kljuc,i\_potrjen\_komik,i\_id\_uporabnika,i\_id\_slike);

                    EXCEPTION

                    WHEN integrity\_constraint\_violation THEN

                        RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                    WHEN not\_null\_violation THEN

                        RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                    WHEN foreign\_key\_violation THEN

                        RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                    WHEN unique\_violation THEN

                        RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                    WHEN check\_violation THEN

                        RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                    WHEN others THEN

                        RAISE EXCEPTION 'Napaka ...';

                end;

            --UPDATE funckija

            ELSE

                BEGIN

                    UPDATE komiki

                    SET potrjen\_komik = i\_potrjen\_komik,

                        id\_uporabnika = i\_id\_uporabnika,

                        id\_slike = i\_id\_slike

                    WHERE id\_komika = i\_id\_komika;

                    kljuc = i\_id\_komika;

                    EXCEPTION

                    WHEN integrity\_constraint\_violation THEN

                        RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                    WHEN not\_null\_violation THEN

                        RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                    WHEN foreign\_key\_violation THEN

                        RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                    WHEN unique\_violation THEN

                        RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                    WHEN check\_violation THEN

                        RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                    WHEN others THEN

                        RAISE EXCEPTION 'Napaka ...';

                end;

            end if;

    RETURN kljuc;

    END;

    $$ LANGUAGE plpgsql;

SELECT FROM komiki\_iu(null,false,16,1);

SELECT FROM komiki\_iu(11,true,1,1);

## Entiteta najljubsi

CREATE FUNCTION najljubsi\_iu(i\_id\_najljubsega integer, i\_id\_uporabnika integer,i\_id\_specijala integer)

RETURNS integer as

$$

DECLARE

    kljuc integer;

BEGIN

   IF (i\_id\_najljubsega IS NULL ) THEN

       BEGIN

          kljuc=nextval('najljubsi\_id\_najljubsega\_seq');

          INSERT INTO najljubsi (id\_najljubsega, id\_uporabnika, id\_specijala)

          VALUES (kljuc,i\_id\_uporabnika,i\_id\_specijala);

          EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

       END;

   ELSE

       BEGIN

           UPDATE najljubsi

           SET id\_uporabnika = i\_id\_uporabnika,

               id\_specijala = i\_id\_specijala

           WHERE id\_najljubsega = i\_id\_najljubsega;

           kljuc=i\_id\_najljubsega;

           EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

       END;

   END IF;

   RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT FROM najljubsi\_iu(null,15,12);

SELECT FROM najljubsi\_iu(13,15,10);

## Entiteta ogledi

CREATE FUNCTION ogledi\_iu(i\_id\_ogleda integer, i\_trajanje\_ogleda integer, i\_datumcas\_ogleda timestamp,

i\_id\_specijala integer, i\_id\_uporabnika integer)

RETURNS integer as

$$

DECLARE

    kljuc integer;

BEGIN

   IF (i\_id\_ogleda IS NULL ) THEN

       BEGIN

          kljuc=nextval('ogledi\_id\_ogleda\_seq');

          INSERT INTO ogledi (id\_ogleda, trajanje\_ogleda, datumcas\_ogleda, id\_specijala, id\_uporabnika)

          VALUES (kljuc,i\_trajanje\_ogleda,i\_datumcas\_ogleda,i\_id\_specijala,i\_id\_uporabnika);

          EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

       END;

   ELSE

       BEGIN

           UPDATE ogledi

           SET trajanje\_ogleda = i\_trajanje\_ogleda,

               datumcas\_ogleda = i\_datumcas\_ogleda,

               id\_specijala = i\_id\_specijala,

               id\_uporabnika = i\_id\_uporabnika

           WHERE id\_ogleda = i\_id\_ogleda;

           kljuc=i\_id\_ogleda;

           EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

       END;

   END IF;

   RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT FROM ogledi\_iu(null,2,cast(current\_timestamp as timestamp),12,15);

SELECT FROM ogledi\_iu(1,20,cast(current\_timestamp as timestamp),12,15);

## Entiteta osebe

create or replace function osebe\_iu(i\_id\_osebe integer,i\_ime character varying,i\_priimek character varying, i\_emso character varying (20),

                                    i\_spol character varying(1),i\_datum\_rojstva date, i\_naslov character varying (120),

                                    i\_enaslov character varying,i\_telefon character varying(20),

                                    i\_davcna\_stevilka character varying (15), i\_id\_drzave integer,

                                    i\_id\_postne\_stevilke integer)

RETURNS integer as

$$

    DECLARE

        kljuc integer;

BEGIN

        -- INSERT FUNCTION

        IF(i\_id\_osebe IS NULL ) THEN

            BEGIN

                kljuc = nextval('osebe\_id\_osebe\_seq');

                INSERT INTO osebe (id\_osebe, ime, priimek, emso, spol, datum\_rojstva, naslov, enaslov, telefon, davcna\_stevilka, id\_drzave, id\_postne\_stevilke)

                VALUES (kljuc,i\_ime,i\_priimek,i\_emso,i\_spol,i\_datum\_rojstva,i\_naslov,i\_enaslov,i\_telefon,i\_davcna\_stevilka,i\_id\_drzave,i\_id\_drzave);

                EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

            end;

        ELSE

            BEGIN

            --UPDATE function

            UPDATE osebe

            SET ime = i\_ime,

                priimek = i\_priimek,

                emso = i\_emso,

                spol = i\_spol,

                datum\_rojstva = i\_datum\_rojstva,

                naslov = i\_naslov,

                enaslov = i\_enaslov,

                telefon = i\_telefon,

                davcna\_stevilka = i\_davcna\_stevilka,

                id\_drzave = i\_id\_drzave,

                id\_postne\_stevilke = i\_id\_postne\_stevilke

            WHERE id\_osebe = i\_id\_osebe;

            kljuc = i\_id\_osebe;

            --exceptions

            EXCEPTION

            WHEN integrity\_constraint\_violation THEN

                RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

            WHEN not\_null\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

            WHEN foreign\_key\_violation THEN

                RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

            WHEN unique\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

            WHEN check\_violation THEN

                RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

            WHEN others THEN

                RAISE EXCEPTION 'Napaka ...';

            END;

        END IF;

    RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM osebe\_iu (null,'Andrej','Skraba','12345677','M','20.02.200','Jenkova cesta 25','andrej@gmail.com','12345','1234567',2,2);

SELECT \* FROM osebe\_iu (26,'Andrej','Skrabar','1234567','M','12.12.2000','Jenkova cesta 25','andrej@gmail.com','12345','1234567',2,2);

## Entiteta popusti

create or replace function popusti\_iu(i\_id\_popusta integer,i\_popust character varying, i\_procent numeric(5,2),i\_datum\_od timestamp, i\_datum\_do timestamp)

RETURNS integer as

$$

    DECLARE

        kljuc integer;

BEGIN

        -- INSERT FUNCTION

        IF(i\_id\_popusta IS NULL ) THEN

            BEGIN

                kljuc = nextval('popusti\_id\_popusta\_seq');

                INSERT INTO popusti (id\_popusta,popust, procent, datum\_od, datum\_do)

                VALUES (kljuc,i\_popust,i\_procent,i\_datum\_od,i\_datum\_do);

                EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

            end;

        ELSE

            BEGIN

            --UPDATE function

            UPDATE popusti

            SET popust = i\_popust,

                procent = i\_procent,

                datum\_od = i\_datum\_od,

                datum\_do = i\_datum\_do

            WHERE id\_popusta = i\_id\_popusta;

            kljuc = i\_id\_popusta;

            --exceptions

            EXCEPTION

            WHEN integrity\_constraint\_violation THEN

                RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

            WHEN not\_null\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

            WHEN foreign\_key\_violation THEN

                RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

            WHEN unique\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

            WHEN check\_violation THEN

                RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

            WHEN others THEN

                RAISE EXCEPTION 'Napaka ...';

            END;

        END IF;

    RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM popusti\_iu(10,'testni',0.5,cast(current\_timestamp as timestamp),cast(current\_timestamp as timestamp))

## Eniteta popusti\_specijali

create or replace function popusti\_specijali\_ui(i\_id\_popusta integer, i\_id\_specijala integer)

RETURNS boolean as $$

BEGIN

    --insert into popusti\_specijali

    INSERT INTO popusti\_specijali (id\_popusta, id\_specijala, popust)

    VALUES (i\_id\_popusta,i\_id\_specijala,(SELECT procent FROM popusti WHERE popusti.id\_popusta=i\_id\_popusta));

    return true;

END

$$ LANGUAGE plpgsql;

--klic funkcije

SELECT \* FROM popusti\_specijali\_ui(5,5);

## Entiteta postne\_stevilke

create or replace function postne\_stevilke\_iu(i\_id\_postne\_stevilke integer, i\_postna\_stevilka character varying, i\_kraj\_mesto character varying, i\_id\_drzave integer)

RETURNS integer as

$$

    DECLARE

        kljuc integer;

BEGIN

        -- INSERT FUNCTION

        IF(i\_id\_postne\_stevilke IS NULL ) THEN

            BEGIN

                kljuc = nextval('postne\_stevilke\_id\_postne\_stevilke\_seq');

                INSERT INTO postne\_stevilke (id\_postne\_stevilke, postna\_stevilka, kraj\_mesto, id\_drzave)

                VALUES (kljuc,i\_postna\_stevilka,i\_kraj\_mesto,i\_id\_drzave);

                EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

            end;

        ELSE

            BEGIN

            --UPDATE function

            UPDATE postne\_stevilke

            SET postna\_stevilka = i\_postna\_stevilka,

                kraj\_mesto = i\_kraj\_mesto,

                id\_drzave = i\_id\_drzave

            WHERE id\_postne\_stevilke = i\_id\_postne\_stevilke;

            kljuc = i\_id\_postne\_stevilke;

            --exceptions

            EXCEPTION

            WHEN integrity\_constraint\_violation THEN

                RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

            WHEN not\_null\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

            WHEN foreign\_key\_violation THEN

                RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

            WHEN unique\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

            WHEN check\_violation THEN

                RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

            WHEN others THEN

                RAISE EXCEPTION 'Napaka ...';

            END;

        END IF;

    RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM postne\_stevilke\_iu(null,'12345','Matuzici',2);

SELECT \* FROM postne\_stevilke\_iu(11,'4321','donji Matuzici',2);

## Entiteta racuni\_izdelki

create or replace function racuni\_izdelki (i\_id\_racuni\_izdelki integer,i\_id\_racuna integer,i\_id\_specijala integer,

i\_kolicina integer)

RETURNS integer as

$$

DECLARE kljuc integer;

        cenaSpecijala numeric(8,2);

        id\_popusta\_help integer;

        vrednost\_popusta numeric(5,2);

        prodajnaCenaRacuna numeric(8,2);

        preveriPopust integer;

    --postoji mogucnost da cena vrijednost null

BEGIN

    --get data about special

    SELECT cena

    INTO cenaSpecijala

    FROM specijali

    WHERE id\_specijala = i\_id\_specijala;

    --check if INSERT or UPDATE

    IF (i\_id\_racuni\_izdelki IS NULL) THEN

        --INSERT

        BEGIN

                --check if sepcial is free

                IF (cenaSpecijala IS NULL) THEN

                    BEGIN

                        prodajnaCenaRacuna = 0;

                    end;

                ELSE

                    begin

                        --preveri ce obstaja popust za ta specijal

                        SELECT COUNT (id\_popusti\_specijali)

                        INTO preveriPopust

                        FROM popusti\_specijali

                        WHERE id\_specijala = i\_id\_specijala;

                        IF(preveriPopust > 0) THEN

                            begin

                                --get popust value with id\_popusta

                                SELECT procent

                                INTO vrednost\_popusta

                                FROM popusti

                                WHERE id\_popusta = id\_popusta\_help;

                                prodajnaCenaRacuna = cenaSpecijala \* i\_kolicina;

                                prodajnaCenaRacuna = prodajnaCenaRacuna - (prodajnaCenaRacuna \* vrednost\_popusta);

                            end;

                        else

                            begin

                                vrednost\_popusta = 0;

                                prodajnaCenaRacuna = cenaSpecijala \* i\_kolicina;

                            end;

                        end if;

                    end;

                end if;

            kljuc = nextval('racuni\_izdelki\_id\_racuni\_izdelki\_seq');

            INSERT INTO racuni\_izdelki(id\_racuni\_izdelki,id\_racuna, id\_specijala, popust, prodajna\_cena, kolicina)

            VALUES (kljuc,i\_id\_racuna,i\_id\_specijala,vrednost\_popusta,prodajnaCenaRacuna,i\_kolicina);

            return kljuc;

            --catach exceptions

            EXCEPTION

            WHEN integrity\_constraint\_violation THEN

                RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

            WHEN not\_null\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

            WHEN foreign\_key\_violation THEN

                RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

            WHEN unique\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

            WHEN check\_violation THEN

                RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

            WHEN others THEN

                RAISE EXCEPTION 'Napaka ...';

        end;

    ELSE

        BEGIN

            kljuc = i\_id\_racuni\_izdelki;

            --execute UPDATE

            --edina vrednost ki je logicna da se lahko spreminja je id\_racuna

            UPDATE racuni\_izdelki

            SET id\_racuna = i\_id\_racuna

            WHERE id\_racuni\_izdelki = i\_id\_racuni\_izdelki;

            --catach exceptions

            EXCEPTION

            WHEN integrity\_constraint\_violation THEN

                RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

            WHEN not\_null\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

            WHEN foreign\_key\_violation THEN

                RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

            WHEN unique\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

            WHEN check\_violation THEN

                RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

            WHEN others THEN

                RAISE EXCEPTION 'Napaka ...';

        end;

    END IF;

return kljuc;

END

$$ LANGUAGE plpgsql;

--insert

SELECT \* FROM racuni\_izdelki(null,1,10,2);

--update

SELECT \* FROM racuni\_izdelki (2,1,5,2);

## Entiteta racuni

create or replace function racuni\_ui (i\_id\_racuna integer,i\_stevilka integer,i\_datumcas timestamp,

i\_id\_osebe integer,i\_id\_vrste\_placila integer)

RETURNS integer as

$$

DECLARE

    kljuc integer;

BEGIN

    --check if id\_racuna is passed

    IF (i\_id\_racuna IS NULL) THEN

        BEGIN

            kljuc = nextval('racuni\_id\_racuna\_seq');

            INSERT INTO racuni (id\_racuna, stevilka, datumcas, id\_osebe, id\_vrste\_placila)

                        VALUES (kljuc,i\_stevilka,i\_datumcas,i\_id\_osebe,i\_id\_vrste\_placila);

                                   EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

        END;

    ELSE

        BEGIN

            UPDATE racuni

            SET stevilka = i\_stevilka,

                datumcas = i\_datumcas,

                id\_osebe = i\_id\_osebe,

                id\_vrste\_placila = i\_id\_vrste\_placila

            WHERE id\_racuna = i\_id\_racuna;

                kljuc = i\_id\_racuna;

                EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

        END;

    END IF;

return kljuc;

END; --end of function

$$ LANGUAGE plpgsql;

--insert

SELECT \* FROM racuni\_ui(null,12343,cast(current\_timestamp as timestamp),25,2);

SELECT \* FROM racuni\_ui(null,54321,cast(current\_timestamp as timestamp),25,10);

--update

SELECT \* FROM racuni\_ui(1,54321,cast(current\_timestamp as timestamp),23,2);

## Entiteta slike

create or replace function slike\_iu(i\_id\_slike integer, i\_velikost\_slike integer,i\_ime\_slike character varying,i\_opis character varying)

RETURNS integer as

$$

    DECLARE

        kljuc integer;

BEGIN

        -- INSERT FUNCTION

        IF(i\_id\_slike IS NULL ) THEN

            BEGIN

                kljuc = nextval('slike\_id\_slike\_seq');

                INSERT INTO slike (id\_slike, velkost\_slike, ime\_slike, opis)

                VALUES (kljuc,i\_velikost\_slike,i\_ime\_slike,i\_opis);

                EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

            end;

        ELSE

            BEGIN

            --UPDATE function

            UPDATE slike

            SET velkost\_slike = i\_velikost\_slike,

                ime\_slike = i\_ime\_slike,

                opis = i\_opis

            WHERE id\_slike = i\_id\_slike;

            kljuc = i\_id\_slike;

            --exceptions

            EXCEPTION

            WHEN integrity\_constraint\_violation THEN

                RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

            WHEN not\_null\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

            WHEN foreign\_key\_violation THEN

                RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

            WHEN unique\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

            WHEN check\_violation THEN

                RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

            WHEN others THEN

                RAISE EXCEPTION 'Napaka ...';

            END;

        END IF;

    RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM slike\_iu(null,1024,'unforgiven.jpg','Unforgiven stand up special cover from Dave  Chappelle');

SELECT \* FROM slike\_iu(17,1024,'Unforgiven.jpg','2020 - Unforgiven stand up special cover from Dave  Chappelle');

## Entiteta specijali

CREATE FUNCTION specijali\_iu(i\_id\_specijala integer,i\_ime\_specijala character varying (150),i\_trajanje\_specijala integer,i\_brezplacno boolean, i\_cena numeric(8,2),i\_id\_kategorije integer,i\_id\_komika integer,i\_datum\_objave date,i\_opis character varying (150),i\_id\_slike integer)

RETURNS integer as

$$

DECLARE

    kljuc integer;

BEGIN

   IF (i\_id\_specijala IS NULL ) THEN

       BEGIN

          kljuc=nextval('specijali\_id\_specijala\_seq');

          INSERT INTO specijali (id\_specijala, ime\_specijala, trajanje\_specijala, brezplačno, cena, id\_kategorije, id\_komika, datum\_objave, opis, id\_slike)

          VALUES (kljuc,i\_ime\_specijala, i\_trajanje\_specijala, i\_brezplacno, i\_cena, i\_id\_kategorije, i\_id\_komika, i\_datum\_objave, i\_opis, i\_id\_slike);

          EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

       END;

   ELSE

       BEGIN

           UPDATE specijali

           SET ime\_specijala = i\_ime\_specijala,

               trajanje\_specijala = i\_trajanje\_specijala,

               brezplačno = i\_brezplacno,

               cena= i\_cena,

               id\_kategorije= i\_id\_kategorije,

               id\_komika = i\_id\_komika,

               datum\_objave = i\_datum\_objave,

               opis = i\_opis,

               id\_slike = i\_id\_slike

           WHERE id\_specijala=i\_id\_specijala;

           kljuc=i\_id\_specijala;

           EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

       END;

   END IF;

   RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT FROM specijali\_iu(null,'Unforgiven',55,false,10,1,1,'24.12.2020','Dugo ocekivani specijal od DaveChappella koji ce vas smijehom sigurno obiriti sa nogu',17);

SELECT FROM specijali\_iu(12,'Unforgiven 2',55,false,15,1,1,'24.12.2020','Dugo ocekivani specijal od DaveChappella koji ce vas smijehom sigurno obiriti sa nogu',17);

## Entiteta traileri

CREATE FUNCTION traileri\_iu(i\_id\_trailerja integer,i\_trajanje\_trailerja integer,i\_link\_trailerja character varying,i\_id\_specijala integer)

RETURNS integer as

$$

DECLARE

    kljuc integer;

BEGIN

   IF (i\_id\_trailerja IS NULL ) THEN

       BEGIN

          kljuc=nextval('traileri\_id\_trailerja\_seq');

          INSERT INTO traileri (id\_trailerja, trajanje\_trailerja, link\_trailerja, id\_specijala)

          VALUES (kljuc,i\_trajanje\_trailerja,i\_link\_trailerja,i\_id\_specijala);

          EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

       END;

   ELSE

       BEGIN

           UPDATE traileri

           SET trajanje\_trailerja = i\_trajanje\_trailerja,

               link\_trailerja = i\_link\_trailerja,

               id\_specijala = i\_id\_specijala

           WHERE id\_trailerja= i\_id\_trailerja;

           kljuc=i\_id\_trailerja;

           EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

       END;

   END IF;

   RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT FROM traileri\_iu(null,2,'https://www.youtube.com/watch?v=ZqTVSpJEicg',12);

SELECT FROM traileri\_iu(12,2,'https://www.youtube.com/watch?v=GSGzW4MYcXU',12);

## Entiteta uporabniki

create or replace function uporabniki\_iu(i\_id\_uporabnika integer,i\_uporabnisko\_ime character varying, i\_geslo character varying,i\_datum\_registracije date, i\_id\_osebe integer)

RETURNS integer as

$$

    DECLARE

        kljuc integer;

BEGIN

        -- INSERT FUNCTION

        IF(i\_id\_uporabnika IS NULL ) THEN

            BEGIN

                kljuc = nextval('uporabniki\_id\_uporabnika\_seq');

                INSERT INTO uporabniki (id\_uporabnika, uporabnisko\_ime, geslo, datum\_registracije, id\_osebe)

                VALUES (kljuc,i\_uporabnisko\_ime,i\_geslo,i\_datum\_registracije,i\_id\_osebe);

                EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

            end;

        ELSE

            BEGIN

            --UPDATE function

            UPDATE uporabniki

            SET uporabnisko\_ime = i\_uporabnisko\_ime,

                geslo = i\_geslo,

                datum\_registracije = i\_datum\_registracije

            WHERE i\_id\_uporabnika = i\_id\_uporabnika;

            kljuc = i\_id\_uporabnika;

            --exceptions

            EXCEPTION

            WHEN integrity\_constraint\_violation THEN

                RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

            WHEN not\_null\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

            WHEN foreign\_key\_violation THEN

                RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

            WHEN unique\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

            WHEN check\_violation THEN

                RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

            WHEN others THEN

                RAISE EXCEPTION 'Napaka ...';

            END;

        END IF;

    RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM uporabniki\_iu(null,'skrabaKiller2','skraba123','01.01.2020',26);

SELECT \* FROM uporabniki\_iu(13,'skraba Komik','fultezjegeslo','01.01.2020',26);

## Entiteta vrste\_placil

create or replace function vrste\_placil\_iu(i\_id\_vrste\_placila integer,i\_vrsta\_placila character varying)

RETURNS integer as

$$

    DECLARE

        kljuc integer;

BEGIN

        -- INSERT FUNCTION

        IF(i\_id\_vrste\_placila IS NULL ) THEN

            BEGIN

                kljuc = nextval('vrste\_placil\_id\_vrste\_placila\_seq');

                INSERT INTO vrste\_placil (id\_vrste\_placila, vrsta\_placila)

                VALUES (kljuc,i\_vrsta\_placila);

                EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

            end;

        ELSE

            BEGIN

            --UPDATE function

            UPDATE vrste\_placil

            SET vrsta\_placila = i\_vrsta\_placila

            WHERE id\_vrste\_placila = i\_id\_vrste\_placila;

            kljuc = i\_id\_vrste\_placila;

            --exceptions

            EXCEPTION

            WHEN integrity\_constraint\_violation THEN

                RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

            WHEN not\_null\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

            WHEN foreign\_key\_violation THEN

                RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

            WHEN unique\_violation THEN

                RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

            WHEN check\_violation THEN

                RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

            WHEN others THEN

                RAISE EXCEPTION 'Napaka ...';

            END;

        END IF;

    RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT \* FROM vrste\_placil\_iu(null,'mastercard');

SELECT \* FROM vrste\_placil\_iu(16,'visa');

## Entiteta zasluzki

CREATE FUNCTION zasluzki\_iu(i\_id\_zasluzka integer, i\_neizplaceno integer, i\_izplaceno integer, i\_id\_komika integer)

RETURNS integer as

$$

DECLARE

    kljuc integer;

BEGIN

   IF (i\_id\_zasluzka IS NULL ) THEN

       BEGIN

          kljuc=nextval('zasluzki\_id\_zasluzka\_seq');

          INSERT INTO zasluzki (id\_zasluzka, neizplačeno, izplačeno, id\_komika)

          VALUES (kljuc,i\_neizplaceno,i\_izplaceno,i\_id\_komika);

          EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

       END;

   ELSE

       BEGIN

           UPDATE zasluzki

           SET neizplačeno = i\_neizplaceno,

               izplačeno =i\_izplaceno,

               id\_komika = i\_id\_komika

           WHERE id\_zasluzka=i\_id\_zasluzka;

           kljuc=i\_id\_zasluzka;

           EXCEPTION

                WHEN integrity\_constraint\_violation THEN

                    RAISE EXCEPTION 'Napaka ... referenčna integriteta.';

                WHEN not\_null\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni zahtevane vrednosti polja.';

                WHEN foreign\_key\_violation THEN

                    RAISE EXCEPTION 'Napaka ... neustrezna vrednost tujega ključa.';

                WHEN unique\_violation THEN

                    RAISE EXCEPTION 'Napaka ... ni enolične vrednosti polja.';

                WHEN check\_violation THEN

                    RAISE EXCEPTION 'Napaka ... validacijsko pravilo.';

                WHEN others THEN

                    RAISE EXCEPTION 'Napaka ...';

       END;

   END IF;

   RETURN kljuc;

END;

$$ LANGUAGE plpgsql;

SELECT FROM zasluzki\_iu(null,0,0,1);

SELECT FROM zasluzki\_iu(6,10,100,1);

# Triggers funkcije

## Trigger 1 countries

Sprožilec, ki se sproži ob INSERT,UPDATE ali DELETE stavko na tabelo **countries**.

|  |
| --- |
| TABELA |
| CREATE TABLE copy\_countries  (  id serial not null constraint pk\_copy\_countries primary key,  country\_name varchar,  country\_code varchar(8)  ); |

|  |
| --- |
| FUNKCIJA |
| CREATE OR REPLACE FUNCTION tg\_copy\_countries ()  RETURNS TRIGGER AS  $$  BEGIN  -- preverimo operacijo (INSERT,UPDATE,DELETE)  IF (upper(tg\_op) = 'INSERT') THEN -- INSERT  INSERT INTO copy\_countries VALUES (new.id,new.country\_name,new.country\_code);  ELSIF (upper(tg\_op) = 'UPDATE') THEN -- UPDATE  UPDATE copy\_countries  SET id = new.id,  country\_name = new.country\_name,  country\_code = new.country\_code  WHERE id = old.id;  ELSE -- DELETE  DELETE FROM copy\_countries WHERE id=old.id;  END IF;  RETURN NULL;  END;  $$ LANGUAGE plpgsql; |

|  |
| --- |
| TRIGGER |
| CREATE TRIGGER tg\_countries  AFTER INSERT OR UPDATE OR DELETE  ON countries FOR EACH ROW  EXECUTE PROCEDURE tg\_copy\_countries(); |

|  |
| --- |
| Stavki |
| INSERT INTO countries (country\_name, country\_code) VALUES ('XY', 'XY');  SELECT \* FROM copy\_countries WHERE country\_code='XY';  DELETE FROM countries WHERE country\_code='XY'; |

## Trigger 2 countries

Omogočeno samodejno oštevilčenje ob insert stavko. Sproži se enako pri spremembah na tabelo **countries**.

Primer izpisa:

Sprememba: **UPDATE**; Uporabnik: **Nejc**

|  |
| --- |
| TABELA |
| CREATE TABLE log\_countries  (  id serial not null constraint pk\_log\_countries primary key,  --date\_time timestamp without time zone NOT NULL DEFAULT (current\_timestamp at time zone 'UTC'),  date\_time timestamp,  log\_user varchar(120) not null,  change\_desc text not null  ); |

|  |
| --- |
| FUNKCIJA |
| CREATE OR REPLACE FUNCTION tg\_log\_countries()  RETURNS TRIGGER AS  $$  DECLARE  kljuc integer;  tg\_change\_desc text;  BEGIN  kljuc = nextval('log\_countries\_id\_seq');  IF (upper(tg\_op) = 'INSERT') THEN -- INSERT  tg\_change\_desc = 'Sprememba: INSERT; Uporabnik: ' || 'Nejc';  INSERT INTO log\_countries VALUES (kljuc,cast(current\_timestamp as timestamp),'Nejc',tg\_change\_desc);  ELSIF (upper(tg\_op) = 'UPDATE') THEN -- UPDATE  tg\_change\_desc = 'Sprememba: UPDATE; Uporabnik: ' || 'Nejc';  INSERT INTO log\_countries VALUES (kljuc,cast(current\_timestamp as timestamp),'Nejc',tg\_change\_desc);  ELSE -- DELETE  tg\_change\_desc = 'Sprememba: DELETE; Uporabnik: ' || 'Nejc';  INSERT INTO log\_countries VALUES (kljuc,cast(current\_timestamp as timestamp),'Nejc',tg\_change\_desc);  END IF;  RETURN NULL;  END;  $$ LANGUAGE plpgsql; |

|  |
| --- |
| TRIGGER |
| CREATE TRIGGER tg\_log\_countries  AFTER INSERT OR UPDATE OR DELETE  ON countries FOR EACH ROW  EXECUTE PROCEDURE tg\_log\_countries(); |

|  |
| --- |
| Stavki |
| INSERT INTO countries (country\_name, country\_code) VALUES ('XYtest', 'XYtest');  UPDATE countries SET country\_name='XYtestpopravek' WHERE country\_code='XYtest';  DELETE FROM countries WHERE country\_code='XYtest';  SELECT \* FROM log\_countries; |

# Strežniške funkcije za obdelave in izračun statistike

Discounts

Ta funkcija vrača število izdelkov glede na isti discount tip (tedenski, mesečni, super,normal,...)

Primer:

Normal 2

Super 0

Tedenski 2

Mesečni 5

....

|  |
| --- |
| TYPE |
| CREATE TYPE tdiscounts AS (  discount\_name varchar(80),  sum integer  ); |

|  |
| --- |
| FUNKCIJA |
| CREATE OR REPLACE FUNCTION products\_discounts\_s()  RETURNS SETOF tdiscounts AS  $$  DECLARE  t\_numof\_products integer;  -- variable for result row  t\_row tdiscounts;  t\_discount\_types discounts%ROWTYPE;  t\_numofsame\_type integer;  BEGIN  SELECT count(\*) INTO t\_numof\_products FROM products;  -- loop through discount types  FOR t\_discount\_types IN SELECT \* FROM discounts  LOOP  -- processing the row of discount types  t\_row.discount\_name = t\_discount\_types.type;  -- current numof products with current discount type  SELECT count(\*) INTO t\_numofsame\_type FROM products  WHERE products.discount\_id = t\_discount\_types.id;  t\_row.sum = t\_numofsame\_type;  -- return result  RETURN NEXT t\_row;  END LOOP;  END;  $$ LANGUAGE plpgsql; |

|  |
| --- |
| Stavki |
| SELECT \* FROM products\_discounts\_s();  SELECT \* FROM products; |

# Testno polnjenje, posodabljanje,brisanje in preverjanje podatkov podatkovne baze

## Entiteta countries

|  |
| --- |
| Stavki |
| SELECT FROM countries\_iu(null,'drzava1','dr1' );  SELECT FROM countries WHERE id = 100;  SELECT FROM countries WHERE id 246;  SELECT FROM countries\_iu(248,'udrzava2','udr2');  DROP FUNCTION countries\_iu(iu\_id integer, iu\_country\_name character varying, iu\_country\_code character varying); |

## Entiteta roles

|  |
| --- |
| Stavki |
| SELECT FROM roles\_iu(null,'Normal');  SELECT FROM roles\_iu(null,'Admin');  SELECT FROM roles\_iu(null,'Normall');  SELECT FROM roles\_iu(3,'Superuser');  SELECT FROM roles; |

## Entiteta users

|  |
| --- |
| Stavki |
| SELECT FROM users\_iu(NULL, (SELECT id FROM roles WHERE upper(name) = 'NORMAL'),'Nejc','Rogelšek','nejc@gmail.com',NULL,'nejc123',true,false,true,cast(current\_timestamp as timestamp),NULL);  SELECT FROM users\_iu(NULL, (SELECT id FROM roles WHERE upper(name) = 'NORMAL'),'Anže','Konečnik','anze@gmail.com',NULL,'anze123',true,false,true,cast(current\_timestamp as timestamp),NULL);  SELECT FROM users\_iu(NULL, (SELECT id FROM roles WHERE upper(name) = 'SUPERUSER'),'Sadmir','Hasanic','sadmir@gmail.com',NULL,'sadmir123',true,false,true,cast(current\_timestamp as timestamp),NULL);  SELECT FROM users\_iu(2, (SELECT id FROM roles WHERE upper(name) = 'ADMIN'),'Nejc','Rogelšek','nejc@gmail.com',NULL,'nejc12345',true,false,true,NULL, cast(current\_timestamp as timestamp));  SELECT FROM users;  DELETE FROM users WHERE id = 2;  DROP FUNCTION users\_iu(iu\_id integer, iu\_role character varying, iu\_first\_name character varying, iu\_last\_name character varying, iu\_email character varying, iu\_phone character varying, iu\_password character varying, iu\_terms\_and\_conditions boolean, iu\_subscribed boolean, iu\_activated boolean, iu\_created\_at timestamp, iu\_updated\_at timestamp); |

## Entiteta categories

|  |
| --- |
| Stavki |
| SELECT FROM categories;  SELECT FROM categories\_iu(null, null, 'Oblačila');  SELECT FROM categories\_iu(null, null, 'Šport');  SELECT FROM categories\_iu(null, (SELECT id FROM categories WHERE lower(title)='oblačila'), 'Majice');  SELECT FROM categories\_iu(null, (SELECT id FROM categories WHERE lower(title)='oblačila'), 'Hlače');  SELECT FROM categories\_iu(null, (SELECT id FROM categories WHERE lower(title)='šport'), 'Žoge');  SELECT FROM categories\_iu(null, (SELECT id FROM categories WHERE lower(title)='šport'), 'Kopački');  DROP FUNCTION categories\_iu(iu\_id integer, iu\_parent\_id integer, iu\_title character varying);  SELECT FROM categories\_iu((SELECT id FROM categories WHERE lower(title)='žoge'), (SELECT id FROM categories WHERE lower(title)='šport'), 'Žoge');  SELECT FROM categories\_iu((SELECT id FROM categories WHERE lower(title)='kopački'), (SELECT id FROM categories WHERE lower(title)='šport'), 'Kopački'); |

## Entiteta discounts

|  |
| --- |
| Stavki |
| SELECT FROM discounts\_iu(null,'Normal', 30,cast(current\_timestamp as timestamp),cast((select now() + interval '3' day) as timestamp));  SELECT FROM discounts\_iu(null,'Super', 80,cast(current\_timestamp as timestamp),cast((select now() + interval '1' day) as timestamp));  SELECT FROM discounts\_iu(null,'Tedenski', 50,cast(current\_timestamp as timestamp),cast((select now() + interval '7' day) as timestamp));  SELECT FROM discounts\_iu(3,'Tedenski', 90,cast(now() as timestamp),cast((select now() + interval '7' day) as timestamp));  SELECT FROM discounts;  DROP FUNCTION discounts\_iu(iu\_id integer, iu\_type character varying, iu\_value integer, iu\_startAt timestamp, iu\_endAt timestamp);  select now() + interval '7' day ; -- vrne datum  SELECT now() + 1; |

## Entiteta products

|  |
| --- |
| Stavki |
| SELECT FROM products\_iu(null, (SELECT id FROM categories WHERE lower(title)='majice'), 'Majica N26 Nike', 'N26 Nike', 'long description', 'short description', 0.88, '75849274829', 50, 22, 61, (SELECT id FROM discounts WHERE type='Normal'), false, 7, false, true, 4, cast(current\_timestamp as timestamp), null);  SELECT FROM products\_iu(null, (SELECT id FROM categories WHERE lower(title)='hlače'), 'Hlače Adidas Super', 'Adidas Super', 'long description', 'short description', 0.5, '75849274829', 50, 22, 61, (SELECT id FROM discounts WHERE type='Tedenski'), false, 7, false, true, 3, cast(current\_timestamp as timestamp), null);  SELECT FROM products\_iu(null, (SELECT id FROM categories WHERE lower(title)='kopački'), 'Kopački Adidas R56', 'R56 Kopački', 'long description', 'short description', 0.6, '75849274829', 50, 22, 61, (SELECT id FROM discounts WHERE type='Normal'), false, 7, false, true, 2, cast(current\_timestamp as timestamp), null);  SELECT FROM products\_iu(null, (SELECT id FROM categories WHERE lower(title)='žoge'), 'Žoga Sonico', 'Sonico', 'long description', 'short description', 0.7, '75849274829', 50, 22, 61, (SELECT id FROM discounts WHERE type='Tedenski'), false, 7, false, true, 5, cast(current\_timestamp as timestamp), null);  SELECT FROM products\_iu(1, (SELECT id FROM categories WHERE lower(title)='oblačila'), 'Majica N26 Nike', 'N26 Nike', 'long description', 'short description', 0.8, '75849274829', 50, 22, 61, (SELECT id FROM discounts WHERE type='Normal'), false, 16, false, true, 4, null, cast(current\_timestamp as timestamp));  SELECT FROM products;  DELETE FROM products WHERE id =1;  DROP FUNCTION products\_iu(iu\_id integer, iu\_category\_id integer, iu\_name character varying, iu\_shortname character varying, iu\_longdesc text, iu\_shortdesc character varying, iu\_weight numeric, iu\_SKU character varying, iu\_original\_price numeric, iu\_vat numeric, iu\_vat\_price numeric, iu\_discount\_id integer, iu\_digital boolean, iu\_stock integer, iu\_unlimited\_stock boolean, iu\_limited\_edition boolean, iu\_rating integer, iu\_created\_at timestamp, iu\_updated\_at timestamp); |

## Entiteta shipping\_address

|  |
| --- |
| Stavki |
| SELECT FROM countries;  SELECT FROM post\_numbers;  SELECT FROM shipping\_addresses;  SELECT FROM shipping\_addresses\_iu(null, 'Tomšičeva ulica 30','Slovenj Gradec','Koroška', '2380',(SELECT id FROM countries WHERE upper(country\_code)='SI'),(SELECT id FROM post\_numbers WHERE lower(city)='slovenj gradec'));  DROP FUNCTION shipping\_addresses\_iu(iu\_id integer, iu\_address character varying, iu\_city character varying, iu\_state character varying, iu\_zipcode character varying, iu\_country\_id integer, iu\_post\_number\_id integer); |

## Entiteta tags

|  |
| --- |
| Stavki |
| SELECT FROM tags\_iu(null,'Skate',cast(current\_timestamp as timestamp));  SELECT FROM tags\_iu(null,'Football',cast(current\_timestamp as timestamp));  SELECT FROM tags\_iu(null,'Wedding',cast(current\_timestamp as timestamp));  SELECT FROM tags;  DROP FUNCTION tags\_iu(iu\_id integer, iu\_name character varying, iu\_created\_at timestamp); |

## Entiteta products\_tags

|  |
| --- |
| Stavki |
| SELECT FROM products\_tags\_iu(null,(SELECT id FROM products WHERE lower(name)='kopački adidas r56' ),(SELECT id FROM tags WHERE lower(name)='football'));  SELECT FROM products\_tags;  DELETE FROM products\_tags WHERE id=2;  DROP FUNCTION products\_tags\_iu(iu\_id integer, iu\_product\_id integer, iu\_tag\_id integer); |

## Entiteta products\_images

|  |
| --- |
| Stavki |
| SELECT FROM products\_images\_iu(null,'httpsurldoslike.png', 'httpsurldothumbail.png', (SELECT id FROM products WHERE lower(name)='kopački adidas r56')); |

## Entiteta users\_shipping\_addresses

|  |
| --- |
| Stavki |
| SELECT FROM users;  SELECT FROM shipping\_addresses;  SELECT FROM users\_shipping\_addresses\_iu(null, (SELECT id FROM users WHERE lower(email)='sadmir@gmail.com'), (SELECT id FROM shipping\_addresses WHERE lower(city)='slovenj gradec')); |

## Entiteta ratings

|  |
| --- |
| Stavki |
| SELECT FROM ratings;  SELECT FROM ratings\_iu(null,3,(SELECT id FROM users WHERE lower(email)='sadmir@gmail.com'),(SELECT id FROM products WHERE lower(name)='kopački adidas r56')); |

## Entiteta interests

|  |
| --- |
| Stavki |
| SELECT FROM interests;  SELECT FROM interests\_iu(null,(SELECT id FROM categories WHERE lower(title)='majice'),cast(current\_timestamp as timestamp)); |

## Entiteta users\_interests

|  |
| --- |
| Stavki |
| SELECT FROM users\_interests;  SELECT FROM users\_interests\_iu(null,(SELECT id FROM users WHERE lower(email)='sadmir@gmail.com'),(SELECT id FROM interests WHERE category\_id=8)); |

## Entiteta replies

|  |
| --- |
| Stavki |
| SELECT FROM replies; |

## Entiteta comments

|  |
| --- |
| Stavki |
| SELECT FROM comments;  SELECT FROM comments\_iu(null,(SELECT id FROM users WHERE lower(email)='sadmir@gmail.com'),(SELECT id FROM products WHERE lower(name)='kopački adidas r56'), 'Bolaniiiiiii kopački!!!', 10, cast(current\_timestamp as timestamp), null); |

## Entiteta orders

|  |
| --- |
| Stavki |
| SELECT FROM orders;  SELECT FROM users;  DROP FUNCTION orders\_iu(iu\_id integer, iu\_transaction\_id integer, iu\_user\_id integer, iu\_complete boolean, iu\_created\_at timestamp, iu\_users\_shipping\_address\_id integer);  SELECT FROM orders\_iu(null,57654453,(SELECT id FROM users WHERE role\_id=2),false, cast(current\_timestamp as timestamp),(SELECT id FROM users\_shipping\_address WHERE user\_id = 3)); |

## Entiteta orderitems

|  |
| --- |
| Stavki |
| -- order items seq  CREATE SEQUENCE orderitems\_id\_seq;  -- alter  ALTER TABLE orderitems  ALTER COLUMN id  SET DEFAULT nextval('orderitems\_id\_seq');  SELECT FROM orderitems\_iu(null,(SELECT id FROM products WHERE lower(name)='kopački adidas r56'),(SELECT id FROM orders WHERE transaction\_id=573), 50,10,2);  SELECT FROM orderitems;  DROP FUNCTION orderitems\_iu(iu\_id integer, iu\_product\_id integer, iu\_order\_id integer, iu\_price numeric, iu\_discount integer, iu\_quantity integer); |

## Entiteta complaints

|  |
| --- |
| Stavki |
| SELECT FROM complaints;  DROP FUNCTION complaints\_iu(iu\_id integer, iu\_order\_id integer, iu\_title character varying, iu\_description character varying, iu\_created\_at timestamp, iu\_updated\_at timestamp, iu\_reviewed boolean);  SELECT FROM complaints\_iu(null,(SELECT id FROM orders WHERE transaction\_id=573),'Complaint1 title', 'complaint 1 description', cast(current\_timestamp as timestamp), cast(current\_timestamp as timestamp), false); |

## Entiteta option\_groups

|  |
| --- |
| Stavki |
| SELECT FROM option\_groups;  SELECT FROM option\_groups\_iu(null, 'Barve');  SELECT FROM option\_groups\_iu(null, 'Velikost');  DROP FUNCTION option\_groups\_iu(iu\_id integer, iu\_name character varying); |

## Entiteta options

|  |
| --- |
| Stavki |
| SELECT FROM options;  DROP FUNCTION options\_iu(iu\_id integer, iu\_name character varying, iu\_option\_group\_id integer);  SELECT FROM options\_iu(null, 'Zelena', (SELECT id FROM option\_groups WHERE lower(name) = 'barve'));  SELECT FROM options\_iu(null, 'Rdeča', (SELECT id FROM option\_groups WHERE lower(name) = 'barve'));  SELECT FROM options\_iu(null, '42', (SELECT id FROM option\_groups WHERE lower(name) = 'velikost')); |

## Entiteta produts\_options

|  |
| --- |
| Stavki |
| SELECT FROM products\_options;  DROP FUNCTION products\_options\_iu(iu\_id integer, iu\_products\_id integer, iu\_option\_group\_id integer, iu\_required boolean);  SELECT FROM products\_options\_iu(null,(SELECT id FROM products WHERE lower(name)='kopački adidas r56'),(SELECT id FROM option\_groups WHERE lower(name) = 'velikost'),true); |

## Entiteta products physical

|  |
| --- |
| Stavki |
| DROP FUNCTION products\_physical\_iu(iu\_id integer, iu\_product\_id integer, iu\_stock integer);  SELECT FROM products\_physical;  SELECT FROM products\_physical\_iu(null, (SELECT id FROM products WHERE lower(name)='kopački adidas r56'), 10); |

## Entiteta products\_physical\_products\_options

|  |
| --- |
| Stavki |
| SELECT FROM products\_options\_products\_phisical;  SELECT FROM products\_options\_products\_phisical\_iu(null, (SELECT id FROM products\_physical WHERE products\_id=4 AND stock=10),(SELECT id FROM options WHERE lower(name)='rdeča'));  DROP FUNCTION products\_options\_products\_phisical\_iu(iu\_id integer, iu\_products\_physical\_id integer, iu\_option\_id integer); |

# Rezultati poizvedb statističnih funkcij

## Entiteta countries

|  |
| --- |
| SELECT \* FROM countries; |
|  |

## Entiteta roles

|  |
| --- |
| SELECT \* FROM roles; |
|  |

## Entiteta users

|  |
| --- |
| SELECT \* FROM users; |
|  |

## Entiteta categories

|  |
| --- |
| SELECT \* FROM categories; |
|  |

## Entiteta discounts

|  |
| --- |
| SELECT \* FROM discounts; |
|  |

## Entiteta products

|  |
| --- |
| SELECT \* FROM products; |
|  |

## Entiteta shipping\_address

|  |
| --- |
| **SELECT \* FROM shipping\_addresses;** |
|  |

## Entiteta tags

|  |
| --- |
| SELECT \* FROM tags; |
|  |

## Entiteta products\_tags

|  |
| --- |
| SELECT \* FROM products\_tags; |
|  |

## Entiteta products\_images

|  |
| --- |
| SELECT \* FROM products\_images; |
|  |

## Entiteta users\_shipping\_addresses

|  |
| --- |
| SELECT \* FROM users; |
|  |

## Entiteta ratings

|  |
| --- |
| SELECT \* FROM ratings; |
|  |

## Entiteta interests

|  |
| --- |
| SELECT \* FROM interests; |
|  |

## Entiteta users\_interests

|  |
| --- |
| SELECT \* FROM users\_interests; |
|  |

## Entiteta replies

|  |
| --- |
| SELECT \* FROM replies; |
|  |

## Entiteta comments

|  |
| --- |
| SELECT \* FROM comments; |
|  |

## Entiteta orders

|  |
| --- |
| SELECT \* FROM orders; |
|  |

## Entiteta orderitems

|  |
| --- |
| SELECT \* FROM orderitems; |
|  |

## Entiteta complaints

|  |
| --- |
| SELECT \* FROM complaints; |
|  |

## Entiteta option\_groups

|  |
| --- |
| SELECT \* FROM option\_groups; |
|  |

## Entiteta options

|  |
| --- |
| SELECT \* FROM options; |
|  |

## Entiteta produts\_options

|  |
| --- |
| SELECT \* FROM products\_options; |
|  |

## Entiteta products physical

|  |
| --- |
| SELECT \* FROM products\_physical; |
|  |

## Entiteta products\_physical\_products\_options

|  |
| --- |
| SELECT \* FROM products\_options\_products\_phisical; |
|  |

# Zaključek

Evidenca nepremičnin je prikaz računalniške obdelave nepremičnin, ki jih upravlja eno podjetje. Je osnova za evidentiranje nepremičnin, vendar za resno uporabo potrebuje še nekaj sprememb. Najbolj bi bilo potrebno dodati še kakšno funkcijo oz. dopolniti osnovne, da bi lahko ločili med različnimi enotami podjetja. Prav tako bi bilo potrebno zgraditi aplikacijo, ki bi uporabljala to bazo.

# Izjava o samostojnem delu

Nejc Rogelšek, rojen 6.3.2000 izjavljam, da je seminarska naloga z naslovom "SPLETNA TRGOVINA" plod mojega lastnega dela in znanja.

Slovenj Gradec, 9.12 december 2020