SEMINARSKA NALOGA

PODATKOVNE BAZE 2



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

Kazalo vsebine

[PODROBEN OPIS TEME 6](#_Toc60170737)

[E-R DIAGRAM 6](#_Toc60170738)

[Funckije za dodajanje, brisanje in spreminjanje podatkov 6](#_Toc60170739)

[Entitea drzave 6](#_Toc60170740)

[Insert in Update 6](#_Toc60170741)

[Delete 8](#_Toc60170742)

[Entiteta kategorije 8](#_Toc60170743)

[Insert in Update 8](#_Toc60170744)

[Delete 9](#_Toc60170745)

[Entiteta knjiznica\_specijalov 10](#_Toc60170746)

[insert in update 10](#_Toc60170747)

[delete 11](#_Toc60170748)

[Entiteta komiki 11](#_Toc60170749)

[insert in update 11](#_Toc60170750)

[Delete 12](#_Toc60170751)

[Entiteta najljubsi 13](#_Toc60170752)

[insert in update 13](#_Toc60170753)

[Delete 14](#_Toc60170754)

[Entiteta ogledi 14](#_Toc60170755)

[insert in update 14](#_Toc60170756)

[Delete 15](#_Toc60170757)

[Entiteta osebe 16](#_Toc60170758)

[insert in update 16](#_Toc60170759)

[Delete 17](#_Toc60170760)

[Entiteta popusti 18](#_Toc60170761)

[insert in update 18](#_Toc60170762)

[Delete 19](#_Toc60170763)

[Eniteta popusti\_specijali 19](#_Toc60170764)

[insert in update 19](#_Toc60170765)

[Delete 20](#_Toc60170766)

[Entiteta postne\_stevilke 20](#_Toc60170767)

[insert in update 20](#_Toc60170768)

[Delete 21](#_Toc60170769)

[Entiteta racuni\_izdelki 22](#_Toc60170770)

[insert in update 22](#_Toc60170771)

[Delete 24](#_Toc60170772)

[Entiteta racuni 24](#_Toc60170773)

[insert in update 24](#_Toc60170774)

[Delete 26](#_Toc60170775)

[Entiteta slike 26](#_Toc60170776)

[insert in update 26](#_Toc60170777)

[Delete 27](#_Toc60170778)

[Entiteta specijali 28](#_Toc60170779)

[insert in update 28](#_Toc60170780)

[Delete 29](#_Toc60170781)

[Entiteta traileri 30](#_Toc60170782)

[insert in update 30](#_Toc60170783)

[Delete 31](#_Toc60170784)

[Entiteta uporabniki 31](#_Toc60170785)

[insert in update 31](#_Toc60170786)

[Delete 32](#_Toc60170787)

[Entiteta vrste\_placil 33](#_Toc60170788)

[insert in update 33](#_Toc60170789)

[Delete 34](#_Toc60170790)

[Entiteta zasluzki 35](#_Toc60170791)

[insert in update 35](#_Toc60170792)

[Delete 36](#_Toc60170793)

[Triggers funkcije 36](#_Toc60170794)

[Trigger (backup specijalov) 36](#_Toc60170795)

[Strežniške funkcije za obdelave in izračun statistike 37](#_Toc60170796)

[Statiscna funckija 1: zasluzki\_po\_specijalu 37](#_Toc60170797)

[Type 37](#_Toc60170798)

[Funckija 37](#_Toc60170799)

[Rezultat 38](#_Toc60170800)

[Statiscna funckija 2: procent\_oseb\_po\_drzav() 39](#_Toc60170801)

[Type 39](#_Toc60170802)

[Funckija 39](#_Toc60170803)

[Rezultat 39](#_Toc60170804)

[Statiscna funckija 3: izplacaj\_komika(); 40](#_Toc60170805)

[Funckija 40](#_Toc60170806)

[Rezultat 41](#_Toc60170807)

[Statiscna funckija 4: odgledane\_minute\_po\_specijalu 41](#_Toc60170808)

[Type 41](#_Toc60170809)

[Funkcija 41](#_Toc60170810)

[Rezultat 42](#_Toc60170811)

[Statiscna funckija 5: procent\_gledanosti\_po\_specijalu 43](#_Toc60170812)

[Type 43](#_Toc60170813)

[Funckija 43](#_Toc60170814)

[Rezultat 43](#_Toc60170815)

[Zaključek 45](#_Toc60170816)

[Izjava o samostojnem delu 46](#_Toc60170817)

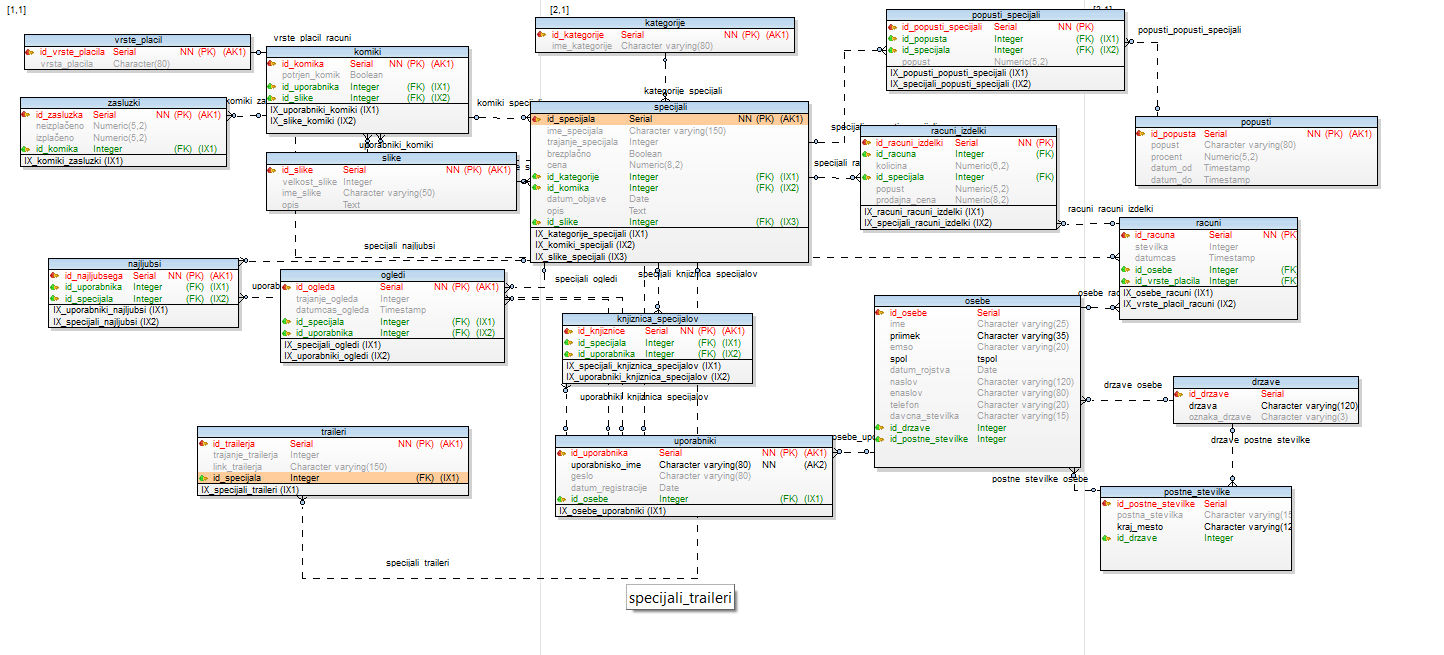
# PODROBEN OPIS TEME

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

### Insert in Update

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');

### Delete

CREATE OR REPLACE FUNCTION drzave\_d(i\_id\_drzave integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM drzave

    WHERE id\_drzave = i\_id\_drzave;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta kategorije

### Insert in Update

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');

### Delete

CREATE OR REPLACE FUNCTION kategorije\_d(i\_id\_kategorije integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM kategorije

    WHERE id\_kategorije = i\_id\_kategorije;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta knjiznica\_specijalov

### insert in update

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);

### delete

CREATE OR REPLACE FUNCTION knjiznica\_specijalov\_d(i\_id\_knjiznice integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM knjiznica\_specijalov

    WHERE id\_knjiznice = i\_id\_knjiznice;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta komiki

### insert in update

    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);

### Delete

CREATE OR REPLACE FUNCTION komiki\_d(i\_id\_komika integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM komiki

    WHERE id\_komika = i\_id\_komika;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta najljubsi

### insert in update

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);

### Delete

CREATE OR REPLACE FUNCTION najljubsi\_d(i\_id\_najljubsega integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM najljubsi

    WHERE id\_najljubsega = i\_id\_najljubsega;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta ogledi

### insert in update

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);

### Delete

CREATE OR REPLACE FUNCTION ogledi\_d(i\_id\_ogleda integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM ogledi

    WHERE id\_ogleda = i\_id\_ogleda;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta osebe

### insert in update

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);

### Delete

CREATE OR REPLACE FUNCTION osebe\_d(i\_id\_osebe integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM osebe

    WHERE id\_osebe = i\_id\_osebe;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta popusti

### insert in update

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))

### Delete

CREATE OR REPLACE FUNCTION popusti\_d(i\_id\_popusta integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM popusti

    WHERE id\_popusta = i\_id\_popusta;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Eniteta popusti\_specijali

### insert in update

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);

### Delete

CREATE OR REPLACE FUNCTION popusti\_specijali\_d(i\_id\_ps integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM popusti\_specijali

    WHERE id\_popusti\_specijali = i\_id\_ps;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta postne\_stevilke

### insert in update

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);

### Delete

CREATE OR REPLACE FUNCTION postne\_stevilke\_d(i\_id\_postne\_stevilke integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM postne\_stevilke

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

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta racuni\_izdelki

### insert in update

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);

### Delete

CREATE OR REPLACE FUNCTION racuni\_izdelki\_d(i\_id\_racuna\_izdelka integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM racuni\_izdelki

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

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta racuni

### insert in update

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);

### Delete

CREATE OR REPLACE FUNCTION racuni\_d(i\_id\_racuna integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM racuni

    WHERE id\_racuna = i\_id\_racuna;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta slike

### insert in update

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');

### Delete

CREATE OR REPLACE FUNCTION slike\_d(i\_id\_slike integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM slike WHERE id\_slike = i\_id\_slike;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta specijali

### insert in update

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);

### Delete

CREATE OR REPLACE FUNCTION speciali\_d(i\_id\_specijala integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM specijali

    WHERE id\_specijala = i\_id\_specijala;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta traileri

### insert in update

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);

### Delete

CREATE OR REPLACE FUNCTION traileri\_d(i\_id\_trailera integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM traileri

    WHERE id\_trailerja = i\_id\_trailera;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta uporabniki

### insert in update

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);

### Delete

CREATE OR REPLACE FUNCTION uporabniki\_d(i\_id\_uporabnika integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM uporabniki

    WHERE id\_uporabnika = i\_id\_uporabnika;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta vrste\_placil

### insert in update

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');

### Delete

CREATE OR REPLACE FUNCTION vrste\_placil\_d(i\_id\_vrste\_placila integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM vrste\_placil

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

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

## Entiteta zasluzki

### insert in update

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);

### Delete

CREATE OR REPLACE FUNCTION zaskuzki\_d(i\_id\_zasluzka integer) RETURNS INTEGER AS

$$

DECLARE

    izbrisane\_vrstice integer;

BEGIN

    DELETE FROM zasluzki

    WHERE id\_zasluzka = i\_id\_zasluzka;

    GET DIAGNOSTICS izbrisane\_vrstice = ROW\_COUNT;

    RETURN izbrisane\_vrstice;

END;

$$ LANGUAGE plpgsql;

# Triggers funkcije

## Trigger (backup specijalov)

Trigger , ki se aktivira ob INSERT,UPDATE ali DELETE stavko na tabelo **specijali**

--TABELA

CREATE TABLE backup\_specijalov

(

    id\_specijala\_backup serial not null constraint pk\_backup\_specijalov primary key,

    ime\_specijala character varying,

    cena numeric (8,2),

    opis character varying

);

--FUNCKIJA

CREATE OR REPLACE FUNCTION tg\_backup\_specials ()

RETURNS TRIGGER AS

$$

BEGIN

   -- check operation

    IF (upper(tg\_op) = 'INSERT') THEN -- INSERT

        INSERT INTO backup\_specijalov (id\_specijala\_backup,ime\_specijala, cena, opis)

            VALUES (new.id\_specijala,new.ime\_specijala,new.cena,new.opis);

    ELSIF (upper(tg\_op) = 'UPDATE') THEN -- UPDATE

        UPDATE backup\_specijalov

        SET ime\_specijala = new.ime\_specijala,

            cena = new.cena,

            opis = new.opis

        WHERE id\_specijala\_backup = old.id\_specijala;

    ELSE -- DELETE

        DELETE FROM backup\_specijalov WHERE id\_specijala\_backup = old.id\_specijala;

    END IF;

   RETURN NULL;

END;

$$ LANGUAGE plpgsql;

--TRIGGER

CREATE TRIGGER tg\_specijali

    AFTER INSERT OR UPDATE OR DELETE

    ON specijali FOR EACH ROW

    EXECUTE PROCEDURE tg\_backup\_specials();

--KLIC

SELECT FROM specijali\_iu(null,'Kaligula',20,false,1,3,1,'20.02.200','kaligaugdfs',16);

# Strežniške funkcije za obdelave in izračun statistike

## Statiscna funckija 1: zasluzki\_po\_specijalu

Ta funckija vrne celotne zasluzke rasporejene po specijalih.

### Type

CREATE TYPE s\_zasluzki as (

    ime\_specijala character varying,

    zasluzek numeric (8,2)

);

### Funckija

CREATE OR REPLACE FUNCTION zasluzki\_po\_specijalu()

RETURNS SETOF s\_zasluzki AS

$$

DECLARE

    speijali\_table specijali%ROWTYPE;

    t\_row s\_zasluzki;

    stevilo\_prodanih integer;

BEGIN

    FOR speijali\_table IN SELECT \* FROM specijali

    LOOP

        t\_row.ime\_specijala = speijali\_table.ime\_specijala;

        --get vrednost

        SELECT COUNT(\*) INTO stevilo\_prodanih

        FROM racuni\_izdelki

        WHERE id\_specijala = speijali\_table.id\_specijala;

        t\_row.zasluzek = stevilo\_prodanih \* speijali\_table.cena;

        RETURN NEXT t\_row;

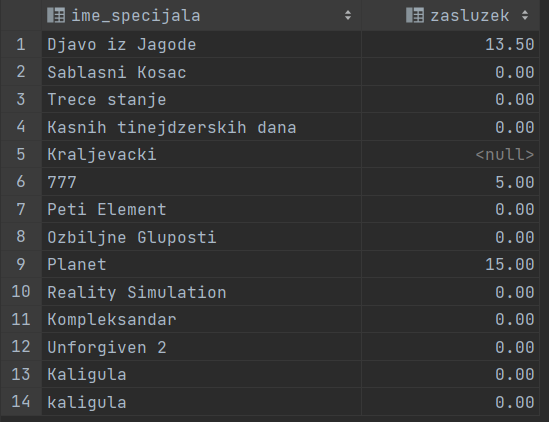
    END LOOP;

END;

$$ LANGUAGE plpgsql;

### Rezultat

SELECT \* from zasluzki\_po\_specijalu();



## Statiscna funckija 2: procent\_oseb\_po\_drzav()

Ta funckija vrne procentualno rasporejenost oseb po drzavah.

### Type

create type t\_drzave as(

    ime\_drzave character varying,

    procent\_oseb numeric (5,2)

);

### Funckija

create or replace function procent\_oseb\_po\_drzav()

RETURNS SETOF t\_drzave AS

$$

DECLARE

    t\_row t\_drzave;

    drzave\_table drzave%ROWTYPE;

    vse\_osebe numeric(5,2);

    drzava\_osebe numeric(5,2);

BEGIN

    SELECT COUNT(\*) INTO vse\_osebe

    FROM osebe;

    FOR drzave\_table IN SELECT \* FROM drzave

    LOOP

        t\_row.ime\_drzave = drzave\_table.drzava;

        SELECT COUNT(\*) INTO drzava\_osebe

        FROM osebe

        WHERE id\_drzave = drzave\_table.id\_drzave;

        t\_row.procent\_oseb = drzava\_osebe / vse\_osebe \*100;

        RETURN NEXT t\_row;

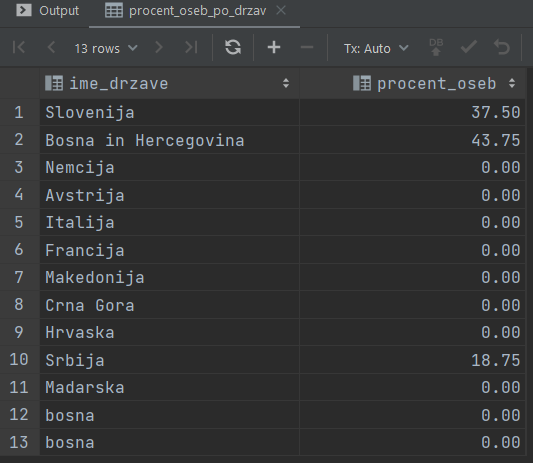
    end loop;

end;

$$ LANGUAGE plpgsql;

### Rezultat

SELECT \* FROM procent\_oseb\_po\_drzav();



## Statiscna funckija 3: izplacaj\_komika();

Ta funckija vzame dva vhodna parametra ( id\_komika z pomocjo katerega najde komika za katerega bi se naj izplacio izvedlo in vrednost zahtevanega izplacila).

### Funckija

CREATE OR REPLACE FUNCTION izplacaj\_komika(i\_id\_komika integer, vrednost\_izplacila numeric(5,2))

RETURNS numeric(5,2) AS

$$

DECLARE

    trenutno\_stanje numeric(5,2);

BEGIN

    SELECT neizplačeno INTO trenutno\_stanje

    FROM zasluzki

    WHERE id\_komika = i\_id\_komika;

    --preveriti ce komik ima toliko neizplacenih sredstv

    IF(vrednost\_izplacila>trenutno\_stanje) then

        begin

            return 0;

        end;

    else

        begin

            UPDATE zasluzki

            SET neizplačeno = neizplačeno - vrednost\_izplacila,

                izplačeno = izplačeno + vrednost\_izplacila

            WHERE id\_komika = i\_id\_komika;

            return vrednost\_izplacila;

        end;

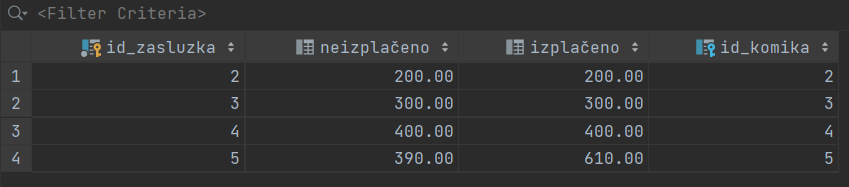
    end if;

END;

$$ LANGUAGE plpgsql;

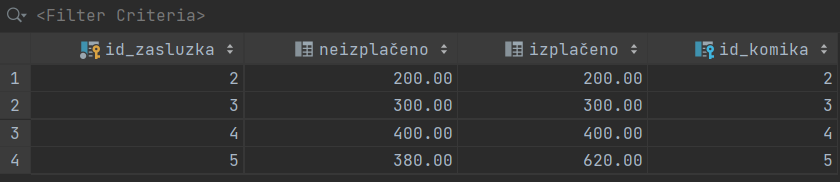
### Rezultat

Pred klicem funckije



SELECT \* FROM izplacaj\_komika(5,10);

Po klicu funckije



## Statiscna funckija 4: odgledane\_minute\_po\_specijalu

Ta funckija vrne vsoto vseh odgledanih minut po specijalu

### Type

CREATE TYPE t\_ogledi as (

    ime\_specijala character varying,

    stevilo\_odgledanih\_minut integer

);

### Funkcija

CREATE OR REPLACE FUNCTION odgledane\_minute\_po\_specijalu()

RETURNS SETOF t\_ogledi AS

$$

DECLARE

    speijali\_table specijali%ROWTYPE;

    ogledi\_table ogledi%ROWTYPE;

    t\_row t\_ogledi;

    stevilo\_minut integer;

BEGIN

    FOR speijali\_table IN SELECT \* FROM specijali

    LOOP

        stevilo\_minut = 0;

        t\_row.ime\_specijala = speijali\_table.ime\_specijala;

        FOR ogledi\_table IN SELECT \* FROM ogledi

        LOOP

            IF(ogledi\_table.id\_specijala = speijali\_table.id\_specijala) then

                stevilo\_minut = stevilo\_minut + ogledi\_table.trajanje\_ogleda;

            end if;

        end loop;

        t\_row.stevilo\_odgledanih\_minut = stevilo\_minut;

        RETURN NEXT t\_row;

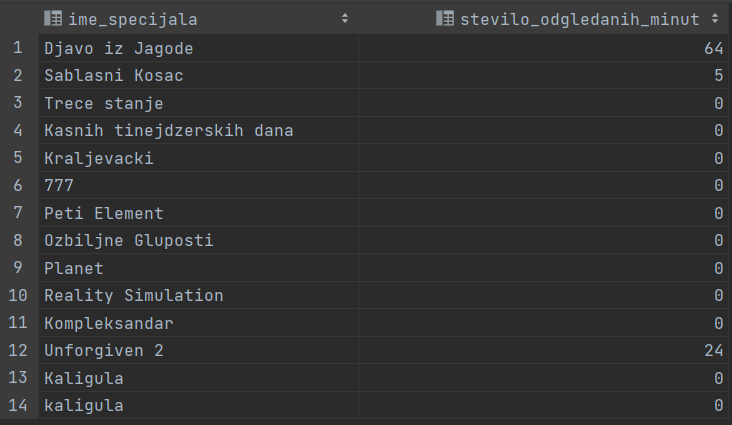
    END LOOP;

END;

$$ LANGUAGE plpgsql;

### Rezultat

SELECT \* FROM odgledane\_minute\_po\_specijalu();



## Statiscna funckija 5: procent\_gledanosti\_po\_specijalu

Ta funckija vrne povprecni procent gledanosti po specijalu.

### Type

CREATE type v\_procent as (

    ime\_specijala character varying,

    procent\_gledanosti numeric (8,2)

);

### Funckija

CREATE OR REPLACE FUNCTION procent\_gledanosti\_po\_specijalu()

RETURNS SETOF v\_procent AS

$$

DECLARE

    speijali\_table specijali%ROWTYPE;

    ogledi\_table ogledi%ROWTYPE;

    t\_row v\_procent;

    stevilo\_ogledov integer;

    stevilo\_odgledanih\_minut numeric(8,2);

    procent\_gledanosti numeric(8,2);

BEGIN

    FOR speijali\_table IN SELECT \* FROM specijali

    LOOP

        --get number of views

        SELECT COUNT(\*) INTO stevilo\_ogledov

        FROM ogledi

        WHERE id\_specijala = speijali\_table.id\_specijala;

        --get all minutes watched

        SELECT SUM(trajanje\_ogleda) INTO stevilo\_odgledanih\_minut

        FROM ogledi

        WHERE id\_specijala = speijali\_table.id\_specijala;

        t\_row.ime\_specijala = speijali\_table.ime\_specijala;

        procent\_gledanosti = stevilo\_odgledanih\_minut / (stevilo\_ogledov \* speijali\_table.trajanje\_specijala) \*100;

        t\_row.procent\_gledanosti =  procent\_gledanosti;

        RETURN NEXT t\_row;

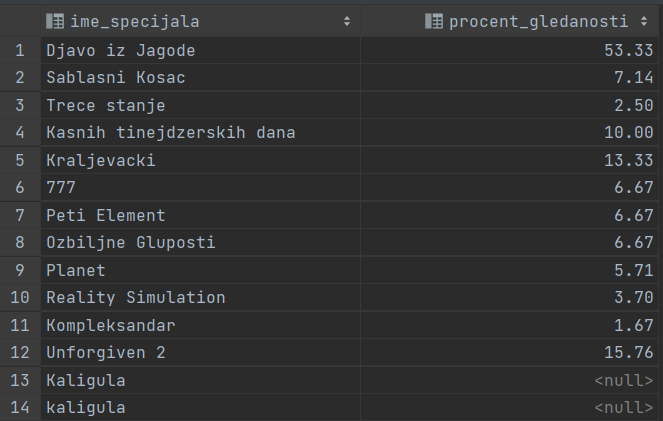
    END LOOP;

END;

$$ LANGUAGE plpgsql;

### Rezultat

SELECT \* FROM procent\_gledanosti\_po\_specijalu();



# Zaključek

Spletna aplikacija za prodajo stand up specijalov ni tako enostava zadeva kot sem jaz pricakoval. Ta podatkovna baza je 'alpha' verzija podatkovne baze za spletno aplikacijo pod imenom 'comedy pill'.

Bolj napredno verzijo te podatkovne baze zaenkrat ni smisleno delati se ker celoten projekt ni še definitivno urejen in veliko stvari ni odlocano do konca.

Iz te perspektive že vidim veliko pomanjkljivosti, recimo tabela za cuvanje podatkov o kreditnih karticah, paypalih in tako naprej še ne obstaja.

Za to podatkovno bazo bi po tem tudi mogla biti narejena spletna aplikacija prav tako tudi in mobilna aplikacija zaradi uporabniskih navad oseb in obcutka bolj personalizirane izkušnje.

V to podatkovno bazo bi mogoc tudi dodal nekaj tabel ki nebi ble povezane za prodajo splecijalov ampak za lazje organiziranje in objavljanje vsebine na socijalnih omrezij za ta projekt.

Upam da ta seminarska naloga ne bo ostala samo na temu da je seminarska naloga ampak zacetek enega velikega in uspešnega projekta.

# Izjava o samostojnem delu

Sadmir Hasanic, rojen 20.02.2000 izjavljam, da je seminarska naloga z naslovom "comedy pill" plod mojega lastnega dela in znanja.

Velenje, 29.12 december 2020