



Zadatci za vježbu

#11

Previous step

Submit answer

Next step

Layout ▾

Tutorial content ▾

2/2

: Okidači

Primjeri u ovom koraku tutoriala također se odnose na bazu podataka **studAdmin**. U rješenjima primjera podrazumijeva se da testiranje obavljate u bazi podataka **studAdmin** koju ste stvorili na vlastitoj instanci sustava PostgreSQL pomoću pgAdmina.

Primjer 1.
Kreirajte u bazi podataka tablicu **upisanPredmetDelLog** :

```
CREATE TABLE    upisanPredmetDelLog  (  
    JMBAG VARCHAR(10) NOT NULL  
, sifPredmet INTEGER NOT NULL  
, akGodina SMALLINT NOT NULL  
, oznGrupa VARCHAR(10) NOT NULL  
, korisnik   VARCHAR(32)   NOT   NULL  
, vrijeme   TIMESTAMP  
);
```

Svako brisanje podataka iz tablice **upisanPredmet** potrebno je zabilježiti u tablici **upisanPredmetDelLog** zajedno sa korisničkim imenom trenutno aktivnog korisnika, te točnim vremenom kada je zapis obrisani.

rješenje

Potrebno je načiniti okidač i funkciju koju taj okidač poziva nakon brisanja neke n-torke u tablici **upisanPredmet**:

```
CREATE FUNCTION upisanPredmetDelLog() RETURNS TRIGGER AS $$  
BEGIN  
    INSERT INTO upisanPredmetDelLog VALUES(OLD.JMBAG  
                                           , OLD.sifPredmet  
                                           , OLD.akGodina  
                                           , OLD.oznGrupa  
                                           , CURRENT_USER  
                                           , CURRENT_TIMESTAMP  
                                           );  
  
    RETURN NULL;  
END;  
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER upisanPredmetDel  
    AFTER DELETE ON upisanPredmet  
    FOR EACH ROW  
    EXECUTE PROCEDURE upisanPredmetDelLog();
```

Pregledajte podatke u tablici **upisanpredmet**:

```
select * from upisanpredmet;
```

Izvršite naredbu kojom ćete pobrisati jednu ili više n-torki (po izboru) iz tablice upisanpredmet. Na primjer, obrisati ćemo jednu n-torku naredbom:

```
delete from upisanpredmet where sifpredmet = 45 and jmbag = '0555000604';
```

Ako sad ispišemo sadržaj tablice **upisanPredmetDelLog**, vidjet ćemo tamo podatke iz izbrisane n-torke (odnosno izbrisanih n-torki u općem slučaju) te korisnika i vrijeme kad je brisanje obavljeno.

JMBAG	sifPredmet	akGodina	oznGrupa	korisnik	vrijeme
0555000604	45	2020	B-K	postgres	2021-05-28 11:35:05.459382

Feedback

Primjer 2.

Potrebno je spriječiti izmjenu studentove ocjene, ako je od njenog unosa prošlo više od 30 dana. Pokušaj izmjene spriječiti iznimkom koja signalizira pogrešku s tekstom 'Izmjena nije dozvoljena'.

rješenje

Potrebno je napisati okidač i funkciju:

```
CREATE FUNCTION dojavipogresku() RETURNS TRIGGER AS $$  
BEGIN  
    RAISE EXCEPTION 'Izmjena ocjene više nije dozvoljena';  
    RETURN NULL;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER ocjenaUpd  
    BEFORE UPDATE OF ocjena ON ispit  
    FOR EACH ROW  
    WHEN ( OLD.datumIspit + '30 days'::interval < CURRENT_DATE)  
        EXECUTE PROCEDURE dojavipogresku();
```

Pogledat ćemo za koje n-torke u tablici **ispit** vrijedi da je od unosa ocjene prošlo više od 30 dana:

```
select *  
from ispit  
where datumIspit + '30 days'::interval < CURRENT_DATE;
```

te ćemo pokušati promijeniti ocjenu za jednu od tih n-torki, na primjer:

```
update ispit  
set ocjena = 2  
where jmbag = '0555000490' and sifpredmet = 6 and datumrok = '2019-09-01';
```

Budući da smo kreirali okidač koji treba reagirati u ovom slučaju, dobit ćemo u PgAdminu poruku:

```
ERROR: Izmjena ocjene više nije dozvoljena  
CONTEXT: PL/pgSQL function dojavipogresku() line 3 at RAISE
```

Primjer 3.

Ne dozvoliti izmjenu vrijednosti atributa ECTSod za predmete koje je upisao jedan ili više studenata. Pokušaj izmjene spriječiti iznimkom koja signalizira pogrešku s tekstom 'Izmjena nije dozvoljena'.

rješenje

```
CREATE OR REPLACE FUNCTION dojavipogresku()
  RETURNS TRIGGER AS $$
BEGIN
  IF ( NOT EXISTS ( SELECT *
                    FROM upisanPredmet
                    WHERE upisanPredmet.sifPredmet = OLD.sifPredmet)
    )
  THEN
    RETURN NEW;
  ELSE
    RAISE EXCEPTION 'Izmjena za predmet % nije dozvoljena', OLD.nazPredmet;
  END IF;
  RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER predUpd
  BEFORE UPDATE OF ECTSbod ON predmet
  FOR EACH ROW
  WHEN (OLD.ECTSbod <> NEW.ECTSbod)
    EXECUTE PROCEDURE dojavipogresku ();
```

ili:

```
CREATE OR REPLACE FUNCTION dojavipogresku()
  RETURNS TRIGGER AS $$
DECLARE
  cnt SMALLINT;
BEGIN
  SELECT COUNT(*) INTO cnt
  FROM upisanPredmet
  WHERE upisanPredmet.sifPredmet = OLD.sifPredmet;

  IF cnt > 0 THEN
    RAISE EXCEPTION 'Izmjena za predmet % nije dozvoljena', OLD.nazPredmet;
    RETURN NULL;
  END IF;

  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER predUpd
  BEFORE UPDATE OF ECTSbod ON predmet
  FOR EACH ROW
  WHEN (OLD.ECTSbod <> NEW.ECTSbod)
    EXECUTE PROCEDURE dojavipogresku ();
```

Pogledajmo koje predmete je upisao jedan ili više studenata:

```
select *
from predmet
where EXISTS (SELECT *
              FROM upisanPredmet
              WHERE upisanPredmet.sifPredmet = predmet.sifPredmet)
```

Odaberimo jedan od tih predmeta i pokušajmo mu promijeniti broj ECTS bodova. Na primjer:

```
update predmet
set ectsbod = 5.0
where sifpredmet = 26
```

Dobit ćemo poruku:

Feedback

```
ERROR: Izmjena za predmet Logička algebra nije dozvoljena
CONTEXT: PL/pgSQL function dojavipogresku() line 10 at RAISE
```

Primjer 4.

Baza podataka **prodaja** sadrži podatke o prodaji računalne opreme. Kreirajte tablice **proizvod**, **kupac**, **racun** i **stavkaRacuna**:

```
CREATE TABLE proizvod
(
    sifProiz INTEGER,
    nazivProiz VARCHAR(100) NOT NULL,
    cijena NUMERIC(8,2) NOT NULL,
    PRIMARY KEY (sifProiz)
);

CREATE TABLE kupac
(
    sifKupac INTEGER,
    imeKupac VARCHAR(30) NOT NULL,
    prezKupac VARCHAR(40) NOT NULL,
    PRIMARY KEY (sifKupac)
);

CREATE TABLE racun
(
    sifRacun INTEGER,
    sifKupac INTEGER,
    datumRacun DATE NOT NULL,
    iznos NUMERIC(8,2) NOT NULL,
    PRIMARY KEY (sifRacun),
    FOREIGN KEY (sifKupac) REFERENCES kupac(sifKupac)
);

CREATE TABLE stavkaRacuna
(
    sifRacun INTEGER,
    sifProiz INTEGER,
    kolicina SMALLINT NOT NULL,
    PRIMARY KEY (sifRacun, sifProiz),
    FOREIGN KEY (sifRacun) REFERENCES racun (sifRacun),
    FOREIGN KEY (sifProiz) REFERENCES proizvod (sifProiz)
);
```

Napomena: sami odlučite hoćete li raditi novu bazu podataka ili ćete tablice **proizvod**, **kupac**, **racun** i **stavkaRacuna** samo dodati u bazu **StudAdmin**.

Unesite zatim u kreirane tablice sljedeće podatke:

```
INSERT INTO proizvod VALUES (10, 'Memorija USB 32 GB', 149.90);
INSERT INTO proizvod VALUES (11, 'Hard disk eksterni 1 TB', 534.90);
INSERT INTO proizvod VALUES (12, 'Memorija USB 8 GB', 54.90);
INSERT INTO proizvod VALUES (13, 'Kabel HDMI 2m', 64.90);

INSERT INTO kupac VALUES (1, 'Mia', 'Novak');
INSERT INTO kupac VALUES (2, 'Luka', 'Ban');
INSERT INTO kupac VALUES (3, 'Lea', 'Kolar');

INSERT INTO racun VALUES (32, 2, '23.05.2021', 599.80);
INSERT INTO racun VALUES (33, 1, '24.05.2021', 299.80);

INSERT INTO stavkaRacuna VALUES (32, 11, 1);
INSERT INTO stavkaRacuna VALUES (32, 13, 1);
INSERT INTO stavkaRacuna VALUES (33, 10, 2);
```

Kad god se u relaciju **stavkaRacuna** unese jedna ili više n-torki, treba ažurirati i ukupni iznos na računu (atribut **iznos** u tablici **racun**). Pri tome treba osigurati da ukupni iznos na nekom računu ne prijeđe 5000kn. Ako se to dogodi, treba spriječiti operaciju unosa u relaciju stavkaRacuna i dojaviti poruku „Prevelik iznos računa!“.

rješenje

```
CREATE OR REPLACE FUNCTION chkInsStavka() RETURNS TRIGGER AS $$  
DECLARE  
    p_cijena NUMERIC(8,2);  
    p_iznos NUMERIC(8,2);  
BEGIN  
    SELECT cijena INTO p_cijena  
    FROM proizvod  
    WHERE sifProiz = NEW.sifProiz;  
  
    SELECT iznos INTO p_iznos  
    FROM racun  
    WHERE sifRacun = NEW.sifRacun;  
  
    IF (p_iznos + p_cijena * NEW.kolicina > 5000)  
        THEN RAISE EXCEPTION 'Prevelik iznos računa!';  
    ELSE  
        UPDATE racun  
        SET iznos = iznos + p_cijena * NEW.kolicina  
        WHERE sifRacun = NEW.sifRacun;  
    END IF;  
  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER stavkaIns  
AFTER INSERT ON stavkaRacuna  
FOR EACH ROW  
EXECUTE PROCEDURE chkInsStavka();
```

Pregledajte sadržaj tablica **racun** i **proizvod** te dodajte u tablicu **stavkaRacuna** n-torku:

```
insert into stavkaRacuna values (33, 12, 1);
```

U tablici **racun** bi se sada ukupni iznos računa sa šifrom 33 trebao povećati za 54.90 kn (cijena proizvoda sa šifrom 12).

[Question](#)[Playground](#)

Koja od navedenih tvrdnji vezanih za okidače **nije** istinita?

a

Moguće je koristiti jednu funkciju za više okidača.

b

Sistemska varijabla OLD sadrži staru vrijednost n-torke na koju je djelovala operacija koja je aktivirala okidač.

c

Okidače treba koristiti onda kada integritetska ograničenja nije moguće opisati na drugi način.

Feedback

d

Sistemska varijabla NEW ima vrijednost NULL ako se okidač aktivira izvođenjem naredbe INSERT.

e

Moguće je specificirati obavljaju li se akcije navedene u okidaču FOR EACH ROW ili FOR EACH STATEMENT.