DDL - Ponavljanje

Evidencija račun

Potrebno je evidentirati račune izdane u trgovini. Za svaki račun potrebno je pratiti *broj_računa*, *datum_izdavanja* i *zaposlenika* koji je račun izdao, te kupca koji je platio račun. Za zaposlenika se prati: *ime*, *prezime*, *oib*, *datum_zaposlenja*, dok se za kupca prati *ime* i *prezime*. Na svakom računu se nalazi stavke koje u sebi sadrže *artikl* i *količinu*. Artikl se sastoji od *naziva* i *cijene*.

```
kupac(id, ime, prezime)
zaposlenik(id, ime, prezime, oib, datum_zaposlenja)
artikl(id, naziv, cijena)
racun(id, id_zaposlenik, id_kupac, broj, datum_izdavanja)
stavka_racun(id, id_racun, id_artikl, kolicina)
```

Baza podataka & tablice

```
CREATE DATABASE trgovina;
USE trgovina;

CREATE TABLE kupac (
  id INTEGER NOT NULL,
  ime VARCHAR(10) NOT NULL,
  prezime VARCHAR(15) NOT NULL
);

CREATE TABLE zaposlenik (
  id INTEGER NOT NULL,
  ime VARCHAR(10) NOT NULL,
  prezime VARCHAR(15) NOT NULL,
  oib CHAR(10) NOT NULL,
  datum_zaposlenja DATETIME NOT NULL
);

CREATE TABLE artikl (
  id INTEGER NOT NULL,
  naziv VARCHAR(20) NOT NULL,
  cijena NUMERIC(10,2) NOT NULL
);

CREATE TABLE racun (
```

```
  id INTEGER NOT NULL,
  id_zaposlenik INTEGER NOT NULL,
  id_kupac INTEGER NOT NULL,
  broj VARCHAR(100) NOT NULL,
  datum_izdavanja DATETIME NOT NULL
);

CREATE TABLE stavka_racun (
  id INTEGER NOT NULL,
  id_racun INTEGER NOT NULL,
  id_artikl INTEGER NOT NULL,
  kolicina INTEGER NOT NULL
);
```

Unos podataka

```
INSERT INTO kupac VALUES (1, 'Lea', 'Fabris'),
                         (2, 'David', 'Sirotić'),
                         (3, 'Tea', 'Bibić');

INSERT INTO zaposlenik VALUES
  (11, 'Marko', 'Marić', '123451', STR_TO_DATE('01.10.2020.', '%d.%m.%Y.')),
  (12, 'Toni', 'Milovan', '123452', STR_TO_DATE('02.10.2020.', '%d.%m.%Y.')),
  (13, 'Tea', 'Marić', '123453', STR_TO_DATE('02.10.2020.', '%d.%m.%Y.'));

INSERT INTO artikl VALUES (21, 'Puding', 5.99),
                          (22, 'Milka čokolada', 30.00),
                          (23, 'Čips', 9);

INSERT INTO racun VALUES
  (31, 11, 1, '00001', STR_TO_DATE('05.10.2020.', '%d.%m.%Y.')),
  (32, 12, 2, '00002', STR_TO_DATE('06.10.2020.', '%d.%m.%Y.')),
  (33, 12, 1, '00003', STR_TO_DATE('06.10.2020.', '%d.%m.%Y.'));

INSERT INTO stavka_racun VALUES (41, 31, 21, 2),
                                (42, 31, 22, 5),
                                (43, 32, 22, 1),
                                (44, 32, 23, 1);
```

Schemas: Napravi novu bazu podataka sa istom skriptom sa nazivom *'trgovina_2'* i prikaži sve korisnike iz baza podataka *'trgovina'* i *'trgovina_2'*

```
CREATE DATABASE trgovina_2; -- plus pokrenuti naredbe za kreiranje tablica i
unos podataka
```

```
# Upit
USE trgovina;

SELECT *
  FROM kupac -- zbog naredbe USE se podrazumijeva da se tablica nalazi u bazi 'trgovina'
UNION ALL
SELECT *
  FROM trgovina_2.kupac;


# ili: upit radi u oba slučaja (može se koristiti naredba USE, ali i ne treba)
SELECT *
  FROM trgovina.kupac
UNION ALL
SELECT *
  FROM trgovina_2.kupac;
```

Primarni ključ: Na svim tablicama postavi ograničenje primarnog ključa

```
# na svim tablicama je potrebno definirati PK za stupac 'id', npr:

-- 1. način
CREATE TABLE kupac (
  id INTEGER NOT NULL PRIMARY KEY,
  ...
);

-- ili
CREATE TABLE kupac (
  id INTEGER NOT NULL,
  ...,
  PRIMARY KEY(id)
);
```

Strani ključ: Na potrebnim tablicama postavi ograničenje stranog ključa

```
CREATE TABLE racun (
  ...,
  id_zaposlenik INTEGER NOT NULL,
  id_kupac INTEGER NOT NULL,
  ...,
  FOREIGN KEY (id_zaposlenik) REFERENCES zaposlenik (id),
  FOREIGN KEY (id_kupac) REFERENCES kupac (id)
);
```

```
CREATE TABLE stavka_racun (
  ...,
  id_racun INTEGER NOT NULL,
  id_artikl INTEGER NOT NULL,
  ...,
  FOREIGN KEY (id_racun) REFERENCES racun (id),
  FOREIGN KEY (id_artikl) REFERENCES artikl (id)
);
```

Unique: Dodaj ograničenje tako da smije postojati samo jedan kupac sa određenim imenom i prezimenom (npr. ime i prezime 'Tea Ivić' se ne smije dva puta ponoviti)

```
CREATE TABLE kupac (
  ...,
  ime VARCHAR(10) NOT NULL,
  prezime VARCHAR(15) NOT NULL,
  UNIQUE (ime, prezime)
);
```

Unique: Dodaj ograničenje tako da se određeni artikl može samo jednom dodati na račun

```
CREATE TABLE stavka_racun (
  ...,
  id_racun INTEGER NOT NULL,
  id_artikl INTEGER NOT NULL,
  ...
  UNIQUE (id_racun, id_artikl)
);
```

Check: Dodaj ograničenje da cijena artikla mora biti pozitivna

```
CREATE TABLE artikl (
  ...,
  cijena NUMERIC(10,2) NOT NULL CHECK (cijena > 0)
);

-- ili
CREATE TABLE artikl (
  ...,
  cijena NUMERIC(10,2) NOT NULL,
  CHECK (cijena > 0)
);
```

Check: Dodaj ograničenje da prezime kupca mora biti duže nego ime (prezime mora imati veći broj slova od imena)

```
CREATE TABLE kupac (
  ...,
  ime VARCHAR(10) NOT NULL,
  prezime VARCHAR(15) NOT NULL,
  CHECK (LENGTH(prezime) > LENGTH(ime))
);
```

Cascade: Stavke računa nemaju smisla postojati bez računa. Dodaj ograničenje koje će prilikom brisanja računa, automatski brisati i stavke računa

```
CREATE TABLE stavka_racun (
  ...,
  id_racun INTEGER NOT NULL,
  ...,
  FOREIGN KEY (id_racun) REFERENCES racun (id) ON DELETE CASCADE
);
```

Zadaća

1. Dodaj ograničenje koje će osigurati da je naziv artikla jedinstven
2. Dodaj ograničenje koje će zabranit unos zaposlenika kojemu je ime isto kao i prezime (npr. unos zaposlenika sa imenom i prezimenom *'Teo Teo'* će rezultirati greškom)
3. Doradi skriptu tako da se prilikom brisanja računa, na sve povezane stavke kao referentna vrijednost stranog ključa postavi vrijednost *NULL*
4. Napravi pogled (View) koji prikazuje sve kupce koji imaju ime jednako prezimenu, pritom je omogućen unos kupaca kroz pogled samo ukoliko je navedeni uvjet zadovoljen
5. Napravi pogled (View) koji prikazuje sve račune i njihov ukupan iznos. Nakon toga napiši upit

koji koristi prethodno napravljeni pogled kako bi se pronašao račun sa najvećim iznosom