Ponovljeni završni ispit iz Baza podataka

4. srpnja 2008.

Zadaci 1. – 4. odnose se na bazu podataka **prodaja** koja sadrži podatke o prodaji računalne opreme u nekoj tvrtci. Bilježe se podaci o djelatnicima, kupcima, proizvodima te o prodajama. Relacija djelatnik sadrži podatke o djelatnicima tvrtke. Za svakog djelatnika evidentira se i djelatnik koji mu je nadređen. Relacije racun i stavkaRacun sadrže podatke o prodanim proizvodima. Jedan racun predstavlja jednu „prodaju“ kojom je kupcu prodan jedan ili više proizvoda. Podaci o proizvodima kupljenim jednim računom nalaze se u relaciji stavkaRacun.
Napomena: Podvučeni atributi čine ključ relacije.

proizvod

<u>sifProizv</u>	INTEGER
naziv	CHAR(255)
cijena	DECIMAL(8,2)

racun

<u>sifRac</u>	INTEGER	šifra računa
sifDjel	INTEGER	šifra djelatnika koji je račun izdao
sifKupac	INTEGER	šifra kupca
datRac	DATE	datum računa
iznos	DECIMAL(8,2)	ukupni iznos računa

kupac

<u>sifKupac</u>	INTEGER
imeKupac	CHAR(30)
prezKupac	CHAR(40)

djelatnik

<u>sifDjel</u>	INTEGER	šifra djelatnika
sifNadDjel	INTEGER	šifra nadređenog djelatnika
imeDjel	CHAR(30)	ime djelatnika
prezDjel	CHAR(40)	prezime djelatnika
login	CHAR(8)	korisničko ime djelatnika

stavkaRacun

<u>sifRac</u>	INTEGER	šifra računa
<u>sifProizv</u>	INTEGER	šifra proizvoda
kolicina	SMALLINT	broj kupljenih proizvoda

1. Za svaki izraz relacijske algebre napišite jednu odgovarajuću SQL naredbu:

a) $(\pi_{\text{sifProizv}}(\sigma_{\text{kolicina}>100}(\textbf{stavkaRacun})) \setminus \pi_{\text{sifProizv}}(\sigma_{\text{iznos}<100}(\textbf{stavkaRacun} \triangleright\!\triangleleft \textbf{racun}))$ **(2 boda)**

```
SELECT DISTINCT sifProizv
  FROM stavkaRacun
 WHERE kolicina > 100
   AND sifProizv NOT IN (SELECT DISTINCT sifProizv FROM stavkaRacun, racun
                           WHERE stavkaRacun.sifRacun = racun.sifRacun
                             AND iznos < 100);
```

b) $_{\textbf{sifKupac, imeKupac, prezKupac}}\mathcal{G}_{\textbf{SUM(iznos)}}(\textbf{kupac}^{*} \triangleright\!\triangleleft \textbf{racun})$ **(2 boda)**

```
SELECT kupac.sifKupac, imeKupac, prezKupac, SUM(iznos)
  FROM kupac LEFT JOIN racun
    ON kupac.sifKupac = racun.sifKupac
GROUP BY sifKupac, imeKupac, prezKupac;
```

2. Napisati SQL upite kojim će se obaviti sljedeće (gdje god je moguće, zadatak riješite jednim upitom):

a) Svim proizvodima koji su skuplji od prosjeka smanjiti cijenu za 20%. **(2 boda)**

```
SELECT AVG(cijena) AS avgC FROM PROIZVOD INTO TEMP avgCijena;

UPDATE proizvod SET cijena = 0.8*cijena
WHERE cijena > (SELECT avgC FROM avgCijena);
```

b) Za svakog kupca ispisati njegovu šifru, ime i prezime, te koliko je različitih proizvoda kupio tijekom **prošle** godine. U listi se moraju nalaziti i kupci koji tijekom **prošle** godine nisu kupili niti jedan proizvod. **Zadatak riješiti bez upotrebe podupita**. **(2 boda)**

```
SELECT kupac.sifKupac, imeKupac, prezKupac, COUNT(DISTINCT sifProizv)
  FROM kupac LEFT JOIN racun ON kupac.sifKupac = racun.sifKupac
                             AND YEAR(TODAY) = YEAR(datRac)+1
  LEFT JOIN stavkaRacun ON racun.sifRac = stavkaRacun.sifRac
GROUP BY kupac.sifKupac, imeKupac, prezKupac
```

Ili

```
SELECT kupac.sifKupac, imeKupac, prezKupac, COUNT(DISTINCT sifProizv)
  FROM racun JOIN stavkaRacun ON racun.sifRac = stavkaRacun.sifRac
       RIGHT JOIN kupac ON kupac.sifKupac = racun.sifKupac
                       AND YEAR(TODAY) = YEAR(datRac)+1;
GROUP BY kupac.sifKupac, imeKupac, prezKupac
```

c) Ispisati šifru, ime i prezime onog djelatnika koji ima najviše **izravno** podređenih djelatnika (ako ima više takvih, ispisati sve takve). **(2 boda)**

```
SELECT nad.sifDjel, nad.imeDjel, nad.prezDjel
  FROM djelatnik nad, djelatnik pod
 WHERE nad.sifDjel = pod.sifNadDjel
GROUP BY nad.sifDjel, nad.imeDjel, nad.prezDjel
HAVING COUNT(*)
>= ALL (SELECT COUNT(*)
          FROM djelatnik nad, djelatnik pod
         WHERE nad.sifDjel = pod.sifNadDjel
         GROUP BY nad.sifDjel);
```

3. Pretpostavite da su u bazi podataka **prodaja** kreirane sve relacije te da su definirani svi primarni i strani ključevi. Napisati naredbe kojima će se osigurati da za svaki račun može postojati najviše 100 stavki. Ako se pokuša unijeti 101. stavka, operaciju onemogućiti uz poruku 'Prekoračen dopušten broj stavki!'. **(3 boda)**

```
CREATE PROCEDURE pogreska()
  RAISE EXCEPTION -746, 0, 'Prekoračen dopušten broj stavki!';
END PROCEDURE;

CREATE TRIGGER stavkaIns
INSERT ON stavkaRacun
REFERENCING NEW AS stavkaNew
FOR EACH ROW
   WHEN ((SELECT COUNT(*) FROM stavkaRacun
           WHERE sifRac = stavkaNew,sifRac) > 100)
   (EXECUTE PROCEDURE pogreska());
```

4. Vlasnik baze podataka **prodaja** i svih relacija u bazi je korisnik *admin*. Korisnicima *horvat*, *novak* i *kolar* već je dodijeljena dozvola za uspostavljanje SQL sjednice. Korisnici nemaju drugih dozvola.

a) Napisati SQL naredbe kojima će korisnik *admin* korisniku *horvat* omogućiti pregled svih podataka u relaciji kupac, te za svakog kupca i ukupan iznos koji je taj kupac potrošio. Korisnik *horvat* smije vidjeti samo sumarne podatke, ali ne i podatke o pojedinim računima. **(2 boda)**

```
CREATE VIEW kupacPlus(sif, ime, prez, iznos) AS
SELECT kupac.sifKupac, imeKupa, prezKupac, SUM(iznos)
  FROM kupac LEFT JOIN racun ON kupac.sifKupac = racun.sifKupac
GROUP BY kupac.sifKupac, imeKupa, prezKupac;

GRANT SELECT ON kupacPlus TO horvat;
```

b) Napisati niz SQL naredbi koje će dovesti do stanja u kojem korisnik *kolar* može uspješno obaviti naredbu:

```
REVOKE UPDATE ON stavkaRacuna FROM novak;
```

ali ne može uspješno obaviti naredbu:

```
REVOKE UPDATE ON stavkaRacuna FROM novak RESTRICT;
```

Uz svaku naredbu navedite koji ju korisnik obavlja te pazite na redoslijed naredbi. **(2 boda)**

```
admin: GRANT UPDATE ON stavkaRacun TO kolar WITH GRANT OPTION;
kolar: GRANT UPDATE ON stavkaRacun TO novak WITH GRANT OPTION;
novak: GRANT UPDATE ON stavkaRacun TO horvat WITH GRANT OPTION;
```

5. Uz pretpostavku da na relacijskoj shemi R = XYZUVWQ vrijede funkcijske zavisnosti iz skupa:
F = { YZW → Q, XQ → YQ, XY → WY, ZU → Q } korištenjem isključivo pravila o akumulaciji (uz refleksivnost u prvom koraku i dekompoziciju u zadnjem) ispitajte **vrijedi li** ili **ne vrijedi** funkcijska zavisnost XYZ → QU. **(2 boda)**

XYZ → XYZ (refleksivnost)
XYZ → XYZ + XY → WY ⇒ XYZ → XYZW (akumulacija)
XYZ → XYZW + YZW → Q ⇒ XYZ → XYZWQ (akumulacija)

Desna strana funkcijske zavisnosti više se ne može proširiti dodatnim atributima, a dekompozicijom ne možemo dobiti traženu funkcijsku zavisnost. Stoga zaključujemo da funkcijska zavisnost XYZ → QU ne vrijedi.

6. Objasnite razliku između pojmova sigurnost i integritet baze podataka. **(2 boda)**

- Integritet baze podataka (*database integrity*) - operacije nad podacima koje korisnici obavljaju **su ispravne** (tj. uvijek rezultiraju konzistentnim stanjem baze podataka)
    - "podaci se štite od ovlaštenih korisnika"
- Sigurnost baze podataka (*database security*) - korisnici koji obavljaju operacije nad podacima **su ovlašteni** za obavljanje tih operacija
    - "podaci se štite od neovlaštenih korisnika"

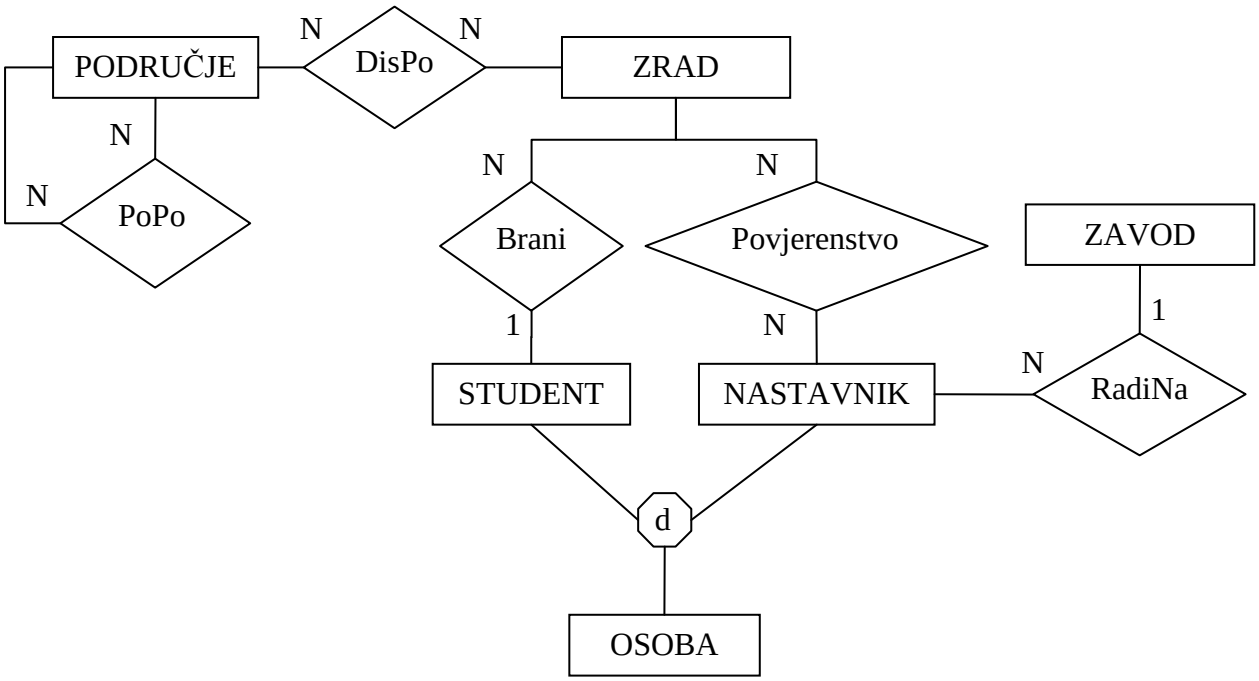
7. Oblikovati bazu podataka o završnim radovima studenata FER-a. Evidentirati podatke o studentima,završnim radovima, područjima i članovima povjerenstva za obranu.

Za svaki završni rad, evidentira se šifra, naslov i datum obrane rada. Svaki rad može pripadati u više različitih područja. Područja su opisana šifrom i nazivom te su međusobno povezana – jedno područje može imati više podređenih i više nadređenih područja.

Za svakog studenta koji brani završni rad, evidentira se JMBG, JMBAG, ime i prezime te godina koje je student upisao studij. Moguće je da je student branio više završnih radova. Završni rad brani se pred povjerenstvom od nekoliko članova. Za svakog nastavnika koji može biti član povjerenstva evidentira se JMBG, ime, prezime, nastavno zvanje te pripadnost jednom od zavoda. Za svaki zavod evidentira se šifra i naziv. Svaki član povjerenstva završni rad ocjenjuje ocjenom od 1 do 5. Jedan nastavnik može biti u povjerenstvu za obranu više završnih radova.

Uočite da student i nastavnik imaju zajedničke atribute.

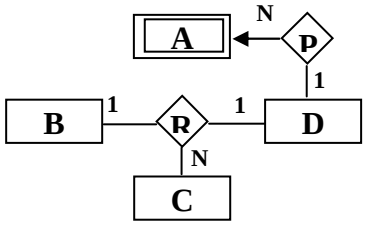
Nacrtati ER model i opisati entitete i veze. Entitete, osim slabih entiteta, opisati isključivo vlastitim atributima. Sve sheme moraju zadovoljavati 3NF. **(5 bodova)**



PODRUČJE: sifPo naziv	**ZRAD**: sifRad naslov datObr	**ZAVOD**: sifZ naziv	**OSOBA**: JMBG ime prezime	**NASTAVNIK**: JMBG zvanje	**STUDENT**: JMBG JMBAG godUpis
DisPo: sifPo sifD	**PoPo**: sifPoNad sifPoPod	**Brani**: sifRad JMBG	**Povjerenstvo**: sifRad JMBG ocjena	**RadiNa**: JMBG sifU	

8.

	atributi	**ključevi**
entitet A	a_1 d_1 a_2	K = { a_1, d_1}
entitet B	b_1 b_2 b_3	K = { b_1, b_2 }
entitet C	c_1 c_2 c_3	K = { c_1 }
entitet D	d_1 d_2 d_3	K = { d_1 }



Za zadani E-R model napišite ekvivalentni relacijski model u obliku SQL naredbi za kreiranje relacija s ugrađenim opisima **primarnih, stranih i alternativnih ključeva**. Tipove podataka u naredbama ne treba navoditi. **(4 boda)**

```
CREATE TABLE A (
  a1, d1, a2
, PRIMARY KEY (a1, d1)
, FOREIGN KEY (d1) REFERENCES D(d1)
);
CREATE TABLE C (
  c1, c2, c3
, PRIMARY KEY (c1)
);
CREATE TABLE D (
  d1, d2, d3
, PRIMARY KEY (d1)
);
```

```
CREATE TABLE B (
  b1, b2, b3
, PRIMARY KEY (b1, b2)
);

CREATE TABLE R (
  b1, b2, c1, d1
, PRIMARY KEY (c1, d1)
, UNIQUE (c1, b1, b2)
, FOREIGN KEY (b1, b2)
    REFERENCES B(b1, b2)
, FOREIGN KEY (c1) REFERENCES C(c1)
, FOREIGN KEY (d1) REFERENCES D(d1)
);
```