2. međuispit iz Baza podataka

13. svibnja 2009.

1. Definirajte Armstrongove aksiome pomoću relacijske sheme R(X,Y,Z). **(1,5 bodova)**

REFLEKSIVNOST
Ako je $X \subseteq Y$, tada vrijedi $Y \rightarrow X$

UVEĆANJE
Ako u shemi R vrijedi $X \rightarrow Y$, tada vrijedi i $XZ \rightarrow Y$

TRANZITIVNOST
Ako u shemi R vrijedi $X \rightarrow Y$ i $Y \rightarrow Z$, tada vrijedi i $X \rightarrow Z$

Zadaci 2.,3., 4. i 5. se odnose na relacije opisane na **slici 1.** Na slici **nisu** prikazane sve n-torke koje su sadržane u relacijama.

Slika 1.

pizza

sifPiz	*nazivPiz*	*cijena*
1	Margharita	35,00
2	Capricciossa	40,00
3	Al Tonno	40,00
4	Calzone	45,00

...

kategorija

sifKat	*nazivKat*
1	tijesto
2	povrće
3	meso

...

sastojak

sifSas	*nazivSas*	*sifKat*	*calPoKg*
1	tijesto	1	1000
2	rajčica	2	100
3	podravec	5	5000
4	šunka	4	2000
5	feferon	6	150

...

recept

sifPiz	*sifSas*	*kolicina*
1	1	0.5
1	2	0.2
1	3	0.15
2	1	0.5
2	2	0.25
2	3	0.15
2	4	0.15
3	1	0.5
3	2	0.2
3	7	0.2

...

2. Napisati po jednu SQL naredbu kojom će se obaviti sljedeće.

a) Za sve pizze čija je ukupna kalorična vrijednost veća od 1500 ispisati naziv, cijenu te ukupnu kaloričnu vrijednost. Ispis poredati po nazivu pizze abecedno. **(1,5 bodova)**

```
SELECT nazivPiz as naziv, cijena, sum(kolicina*calPoKg) AS ukupnoKalorija
  FROM pizza, sastojak, recept
 WHERE pizza.sifPiz=recept.sifPiz
   AND  sastojak.sifSas=recept.sifSas
 GROUP BY nazivPiz, cijena
HAVING sum(kolicina*calPoKg)>1500
 ORDER BY pizza.nazivPiz;
```

b) U svakom receptu količinu sastojka kategorije 'riba' smanjite za 20%. **(1.5 bodova)**

```
UPDATE recept
   SET kolicina = 0.8 * kolicina
 WHERE sifSas IN (SELECT sifSas
                    FROM sastojak, kategorija
                   WHERE sastojak.sifKat=kategorija.sifKat
                     AND  nazivKat = 'riba');
```

c) Ispisati sve podatke o sastojcima koji se koriste u svim pizzama. **(1.5 bodova)**

```
SELECT * FROM sastojak
 WHERE (SELECT COUNT(*) FROM Pizzza) =
       (SELECT COUNT(DISTINCT sifPiz) FROM   recept
         WHERE recept.sifSas = sastojak.sifSas)

SELECT * FROM sastojak
 WHERE NOT EXISTS (SELECT * FROM pizza
                    WHERE NOT EXISTS (SELECT * FROM recept
                                       WHERE recept.sifPiz  = pizza.sifPiz
                                         AND recept.sifSas = sastojak.sifSas
                                     )
                  )

SELECT * FROM sastojak
 WHERE NOT EXISTS (SELECT * FROM pizza
                    WHERE sifPiz NOT IN
                     (SELECT sifPiz FROM recept
                      WHERE recept.sifSas = sastojak.sifSas)
```

d) Za svaku kategoriju ispisati šifru i naziv, broj sastojaka koji pripadaju toj kategoriji te broj različitih pizza u kojima se koriste sastojci te kategorije. **(1.5 bodova)**

```
SELECT sifKat, nazivKat,
(SELECT COUNT(*) FROM sastojak WHERE sastojak.sifKat = kategorija.sifKat),
(SELECT COUNT (DISTINCT sifPiz)
   FROM recept JOIN sastojak ON recept.sifSas = sastojak.sifSas
  WHERE sastojak.sifKat = kategorija.sifKat)
FROM kategorija

SELECT kategorija.sifKat, kategorija.nazivKat,
       COUNT(DISTINCT sastojak.sifSas), COUNT(DISTINCT sifPiz)
  FROM kategorija LEFT JOIN sastojak ON kategorija.sifKat = sastojak.sifKat
                  LEFT JOIN recept ON sastojak.sifSas = recept.sifSas
GROUP BY sifKat, nazivKat
```

3. Napišite SELECT naredbu kojom biste mogli ispitati da li u relaciji *SASTOJAK* eventualno vrijedi funkcijska zavisnost: **nazivSas → sifKat calPoKg**
Objasnite što se iz rezultata izvođenja SELECT naredbe može zaključiti. **(1,5 bodova)**

```
SELECT *
        FROM sastojak s1, sastojak s2
        WHERE s1.nazivSas=s2. nazivSas
        AND (s1.sifKat <> s2. sifKat
        OR  s1.calPoKg <> s2.calPoKg);
```

Objašnjenje:
SELECT naredba ispisuje (ili broji) sve one n-torke koje ne zadovoljavaju funkcijsku zavisnost. Ako takve postoje → FZ sigurno ne vrijedi. Ako takve n-torke ne postoje → FZ bi mogla vrijediti (ali to sa sigurnošću ne možemo tvrditi).

4. Napisati izraz relacijske algebre kojim će se dobiti relacija ISPIS. Relacija ISPIS sadrži šifru, naziv i broj sastojaka za sve pizze koje koštaju manje od 50kn. **(2 boda)**

$$\rho_{ISPIS(sifra,\ naziv,\ brojSas)}\ (_{sifPiz,\ nazivPiz}\mathcal{G}_{\ COUNT(sifSas)\ })(\sigma_{cijena<50}\ (pizza \triangleright\!\triangleleft recept)))$$

5. Napisati SQL naredbe koje će kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje (pomoću B+ stabla) svih dolje navedenih upita. **(2 boda)**

```
SELECT * FROM sastojak ORDER BY nazivSas DESC, sifSas DESC, CalPoKg DESC;
SELECT * FROM sastojak ORDER BY nazivSas , sifSas
SELECT * FROM sastojak ORDER BY nazivSas DESC;
SELECT * FROM sastojak ORDER BY sifKat DESC;
SELECT * FROM sastojak ORDER BY nazivSas DESC, CalPoKg DESC;
SELECT * FROM sastojak WHERE CalPoKg >20 AND CalPoKg <2000;
SELECT * FROM sastojak WHERE sifKat = 2;
```

Napišite barem jednu SELECT naredbu nad relacijom **sastojak** koji se ne može efikasno obaviti kreiranim indeksima. Naredba mora sadržavati barem jedan uvjet u WHERE dijelu.

```
CREATE INDEX i1 ON sastojak (nazivSas, sifSas,  CalPoKg);
CREATE INDEX i2 ON sastojak (nazivSas, CalPoKg);
CREATE INDEX i3 ON sastojak (sifKat);
CREATE INDEX i4 ON sastojak (CalPoKg);
```

```
SELECT * FROM sastojak
        WHERE sifSas = 3
        ORDER BY sifKat, CalPoKg
```

6. Uz pretpostavku da na relacijskoj shemi **R = ABCDEFG** vrijede funkcijske zavisnosti iz skupa:
F = {E→B, D→C, B→DA, GC→DE}, ispitajte vrijedi li funkcijska zavisnost **ADG→B**.
U svakom koraku obavezno navedite koji aksiom ili pravilo koristite. **(1,5 boda)**

Rješenje:
moguće je više rješenja, najjednostavnije je pomoću akumulacije:

- refleksivnost **ADG→ADG**
- akumulacija **ADG→ADG ∧ D→C ⇒ ADG→ADGC**
- akumulacija **ADG→ADGC ∧ GC→DE ⇒ ADG→ADGCE**
- akumulacija **ADG→ADGCE ∧ E→B ⇒ ADG→ADGCEB**
- dekompozicija **ADG→ADGCEB ⇒ ADG→B**

7. Zadano je B+ stablo reda *n* (*n* je paran broj). Ako stablo uz minimalnu popunjenost ima 20 000 kazaljki i ako je u tom slučaju za dohvat zapisa prema vrijednosti ključa potrebno točno 6 U/I operacija (uključujući operaciju za dohvat bloka podataka), odredite kojeg je reda stablo. Napomena: zbog činjenice da je *n* paran broj vrijedi $\lceil (n-1)/2 \rceil = \lceil n/2 \rceil$. **(2 boda)**

Rješenje:
Najgori slučaj:
6 operacija= 6-1=5 razina u stablu
S obzirom da je n parni broj $\lceil (n-1)/2 \rceil = \lceil n/2 \rceil$ nakon zaokruživanja pa vrijedi:

2*n/2*n/2*n/2*n/2=20000
N*n*n*n=160 000
N=20

Zadaci 8. i 9. se odnose na relacijske sheme
VIZA = {oznDrzava, brPutovnica, datumIzdavanje, datumIstek} i
DRZAVA = {oznDrzava, nazivDrzava, clanEU}.
Sheme opisuju proces izdavanja viza za boravak u Hrvatskoj stranim državljanima. Atribut **clanEU** poprima vrijednost 1 za države koje su članovi EU, a za sve ostale države vrijednost 0. Putovnici sa brojem **brPutovnica** iz države **oznDrzava** izdana je viza na dan **datumIzdavanje**. Viza istječe na dan **datumIstek**.

8. Napisati SQL naredbe koje će kreirati relacije ***viza*** i ***drzava*** prema relacijskim shemama VIZA i DRZAVA. Smisleno odaberite tipove podataka. Prilikom kreiranja relacija kombinaciju atributa **oznDrzava**, **brPutovnica** i **datumIzdavanje** postaviti kao primarni ključ u relaciji ***viza*** i osigurati da:

- datum isteka vize ne može poprimiti nul-vrijednosti
- dvije države ne mogu imati isti naziv
- naziv države i atribut članstva u EU ne mogu poprimiti nul-vrijednosti
- vrijednost atributa članstva u EU može biti samo 0 ili 1
- oznaka države u relaciji ***viza*** poprima samo vrijednosti atributa **oznDrzava** u relaciji ***drzava***
- viza vrijedi najmanje 30, a najviše 90 dana
- prilikom brisanja zapisa iz relacije ***drzava*** budu obrisani i svi zapisi u relaciji ***viza*** koji se odnose na obrisanu državu **(3 boda)**

```
CREATE TABLE drzava(
  oznDrzava nchar(3)
, nazivDrzava nchar(70) NOT NULL
, clanEU smallint NOT NULL
, PRIMARY KEY(oznDrzava)
, UNIQUE(nazivDrzava)
, CHECK(clanEU IN (1, 0))
);

CREATE TABLE viza(
  oznDrzava nchar(3)
, brPutovnica nchar(15)
, datumIzdavanje date
, datumIstek date NOT NULL
, PRIMARY KEY(oznDrzava, brPutovnica, datumIzdavanje)
, FOREIGN KEY(oznDrzava) REFERENCES drzava(oznDrzava) ON DELETE CASCADE
, CHECK(datumIstek-datumIzdavanje>=30 AND datumIstek-datumIzdavanje<=90)
);
```

Moguća je i nešto drugačija sintaksa

```
CREATE TABLE drzava(
  oznDrzava nchar(3) PRIMARY KEY
, nazivDrzava nchar(70) NOT NULL UNIQUE
, clanEU smallint NOT NULL CHECK(clanEU IN (1, 0))
);

CREATE TABLE viza(
  oznDrzava nchar(3) REFERENCES drzava(oznDrzava) ON DELETE CASCADE
, brPutovnica nchar(15)
, datumIzdavanje date
, datumIstek date NOT NULL
, PRIMARY KEY(oznDrzava, brPutovnica, datumIzdavanje)
, CHECK(datumIstek-datumIzdavanje>=30 AND datumIstek-datumIzdavanje<=90)
);
```

9. Napisati SQL naredbu za kreiranje virtualne relacije ***unija*** kojom će se korisnicima omogućiti pregled, unos, izmjenu i brisanje svih podataka iz relacije **viza** za osobe koje žive u Europskoj Uniji. **(1.5 bodova)**

```
CREATE VIEW unija (oznDrzava, brPutovnica, datumIzdavanje, datumIstek) AS
        SELECT oznDrzava, brPutovnica, datumIzdavanje, datumIstek
        FROM viza
        WHERE oznDrzava IN
                (
                SELECT oznDrzava FROM drzava WHERE clanEU=1
                )
WITH CHECK OPTION;
```

Ili bez liste atributa:

```
CREATE VIEW unija AS
        SELECT oznDrzava, brPutovnica, datumIzdavanje, datumIstek
        FROM viza
        WHERE oznDrzava IN
                (
                SELECT oznDrzava FROM drzava WHERE clanEU=1
                )
WITH CHECK OPTION;
```

Ili samo sa *:

```
CREATE VIEW unija AS
        SELECT *
        FROM viza
        WHERE oznDrzava IN
                (
                SELECT oznDrzava FROM drzava WHERE clanEU=1
                )
WITH CHECK OPTION;
```

10. U bazi podataka RAZMJENA spremaju se podaci o razmjeni studenata FER-a u sklopu ERASMUS projekta. Relacijska shema baze sastoji se od sljedećih atributa:

JMBAG – matični broj studenta koji se prijavljuje za razmjenu
sifSveuciliste – šifra sveučilišta na koje se student prijavljuje
nazSveuciliste – naziv sveučilišta
sifDržava – šifra države u kojoj se nalazi sveučilište
nazDrzava – naziv države
pbrGrad – poštanski broj grada u kojem se nalazi sveučilište
nazGrad – naziv grada
akGodina – akademske godina za koju se student prijavljuje
semestar – semestar za koji se student prijavljuje (zimski ili ljetni)
odobreno – oznaka je li prijava za razmjenu odobrena ili nije

Vrijede sljedeća pravila:

- Student se za istu godinu i semestar može prijaviti na više sveučilišta
- Student se na isto sveučilište može prijaviti više puta, ali svaki puta u različitoj akademskoj godini i semestru
- poštanski broj je jedinstven za gradove unutar jedne države ali dva grada u različitim državama mogu imati jednak poštanski broj

Odaberite ključ relacijske sheme RAZMJENA tako da ona bude u 1NF, a zatim postupno normalizirajte relacijsku shemu na 2NF i 3NF.

Rješenje:

1NF:

K = {JMBAG, sifSveuciliste, akGodina, semestar}

2NF:

PRIJAVA = {JMBAG, sifSveuciliste, akGodina, semestar, odobreno}
K = {JMBAG, sifSveuciliste, akGodina, semestar }
SVEUCILISTE = {sifSveuciliste, nazSveuciliste, pbrGrad, nazGrad, sifDrzava, nazDrzava}
K = {sifSveuciliste}

3NF:

PRIJAVA = {JMBAG, sifSveuciliste, akGodina, semestar, odobreno}
K = {JMBAG, sifSveuciliste, akGodina, semestar }
SVEUCILISTE = {sifSveuciliste, nazSveuciliste, pbrGrad, sifDrzava}
K = {sifSveuciliste}
DRZAVA = {sifDrzava, nazDrzava}, K = {sifDrzava}
GRAD = {sifDrzava, pbrGrad, nazGrad}, K = {sifDrzava, pbrGrad}

Rješenje ako se pretpostavi da je šifra sveučilišta jedinstvena unutar jedne države:

1NF:

K = {JMBAG, sifSveuciliste, sifDrzava, akGodina, semestar}

2NF:

PRIJAVA = {JMBAG, sifSveuciliste, sifDrzava, akGodina, semestar, odobreno}
K = {JMBAG, sifSveuciliste, sifDrzava, akGodina, semestar }
SVEUCILISTE = {sifSveuciliste, nazSveuciliste, pbrGrad, nazGrad, sifDrzava }
K = {sifSveuciliste, sifDrzava}
DRZAVA = {sifDrzava, nazDrzava}, K = {sifDrzava}

3NF:

PRIJAVA = {JMBAG, sifSveuciliste, sifDrzava, akGodina, semestar, odobreno}
K = {JMBAG, sifSveuciliste, sifDrzava, akGodina, semestar }
SVEUCILISTE = {sifSveuciliste, nazSveuciliste, pbrGrad, sifDrzava}
K = {sifSveuciliste, sifDrzava }
DRZAVA = {sifDrzava, nazDrzava}, K = {sifDrzava}

GRAD = {sifDrzava, pbrGrad, nazGrad}, K = {sifDrzava, pbrGrad}