BAZE PODATAKA – ZAVRŠNI ISPITI

2007 - 2013

Završni ispit iz Baza podataka

21. lipnja 2013.

Zadaci **1-7** odnose se na bazu podataka **legoDB** u kojoj se pohranjuju podaci o kompletima kockica i elementima (tj. kockicama) koje ti kompleti sadrže. Kako je moguća prodaja i kompleta i pojedinačnih elemenata, svi kompleti i elementi navedeni su u tablici **artikl**, a pojedini artikl može biti vrste K ili E (komplet ili element). Jedinstveni identifikator svakog artikla je šifra artikla, a osim vrste artikla, za artikl se navodi naziv i trenutna cijena. Kompleti su dodatno opisani u relaciji **komplet**, a elementi u relaciji **element**. Atributi **proizvOd** i **proizvDo** predstavljaju datume početka, odnosno završetka proizvodnje kompleta. U relaciji **sadrzaj** naveden je sadržaj kompleta (atribut **sifArtiklK**), tj. koje elemente (atribut **sifArtiklE**) u kojoj boji i količini (atributi **boja** i **brElem)** taj komplet sadrži.

Na slici **nisu** prikazane sve n-torke sadržane u relacijama.

artikl

sifArtikl	*nazArtikl*	*vrArtikl*	*cijena*
1	Off-Road Power	K	219.99
2	T-Rex	K	59.99
3	Brick 1X2	E	0.80
4	Plate 2X3	E	0.60
5	Brick 2X2	E	0.80
6	Creator Box	K	99.49
7	Wheel Technic 30x 14	E	5,49

komplet

sifArtikl	*proizvOd*	*proizvDo*
1	01.03.2010	30.05.2013
2	01.06.2012	NULL
6	01.01.2004	01.01.2008

element

sifArtikl	*vrElement*
3	brick
4	plate
5	brick
7	wheel

sadrzaj

sifArtiklK	*sifArtiklE*	*boja*	*brElem*
1	3	black	12
1	4	blue	7
1	4	red	4
1	4	black	4
2	3	red	2
2	5	red	8

1. Napisati SQL naredbe za kreiranje relacija u bazi **legoDB**, kojima će biti definirana sva integritetska ograničenja navedena u uvodnom opisu modela (koja je moguće kreirati CREATE TABLE naredbom), kao i sljedeća ograničenja:
   - atribut **sifArtiklK** u relaciji **sadrzaj** može poprimiti vrijednost šifre artikla navedene u tablici **komplet**, a atribut **sifArtiklE** vrijednost šifre artikla navedene u tablici **element**;
   - u tablici **sadrzaj** smije postojati samo jedan zapis kojim se evidentira da neki komplet sadrži neki element u određenoj boji (npr. smije postojati samo jedan zapis o tome da komplet 1 sadrži element 3 u crnoj boji);
   - NULL vrijednost smije poprimiti samo atribut **proizvDo** (za komplete koji se još uvijek proizvode);
   - ne može postojati više kompleta istog naziva niti više elemenata istog naziva.

   Smisleno odabrati tipove atributa. **(4 boda)**

2. Napisati SQL naredbe kojima će biti spriječena promjena vrijednosti atributa **vrArtikl** u tablici **artikl**, ako za artikl postoji odgovarajući zapis u tablici **komplet** ili **element**, ovisno o vrsti artikla. Prilikom pokušaja obavljanja takve izmjene, dojaviti jednu od poruka: "Artikl *sifra* - komplet je već opisan!" ili " Artikl *sifra* - element je već opisan!". Napomena: *sifra* u tekstu poruke mora odgovarati stvarnoj vrijednosti šifre artikla. (**6 bodova)**

3. Ispisati naziv i cijenu elemenata čiji naziv sadrži znakovni niz '*2x2'* ili '*1x2'*, nisu vrste *brick*, a ima ih barem 6 komada u crnoj (*black*) boji u barem jednom kompletu. Bliže početku liste ispisati skuplje elemente, a one iste cijene poredati po nazivu elementa. Podatke o elementu ispisati samo jednom, čak i ako se traženi element pojavljuje u više kompleta **Zadatak riješiti jednim SQL upitom, bez korištenja podupita. (3 boda)**

4. Za elemente koji se ne pojavljuju niti u jednom kompletu čija je proizvodnja započela od početka prošle godine, ispisati naziv elementa, te ukupan broj kompleta u kojima se element pojavljuje (bez obzira na boju elementa i vrijeme proizvodnje kompleta). Rezultat mora sadržavati i elemente koji se nikada nisu pojavili niti u jednom kompletu - za takve je elemente za ukupan broj kompleta potrebno ispisati vrijednost 0. **(4 boda)**

5. Napisati (jednostavan) SQL upit koji predstavlja primjer za **agregaciju** (operaciju relacijske algebre) te napisati rezultat obavljanja napisanog upita. U ovom zadatku pretpostaviti da su na slici prikazani svi podaci pohranjeni u bazi podataka. **(2 boda)**

6. Pretpostavka je da je u bazi podataka kreiran samo jedan indeks, naredbom:

```
CREATE INDEX i1 ON sadrzaj (sifArtiklE, boja)
```

Izvodi se sljedeći upit:

```
SELECT * FROM artikl, sadrzaj, element
 WHERE artikl.sifArtikl = sadrzaj.sifArtiklE
   AND artikl.sifArtikl = element.sifArtikl
   AND cijena < 30.00
   AND boja = 'red'
```

Optimizator raspolaže sljedećim informacijama:

```
N(artikl) = 12 000
N(sadrzaj) = 300 000
N(element) = 9 000
V(cijena, artikl) = 8000
V(boja, sadrzaj) = 30
```

Nacrtati stablo upita nakon provedene heurističke optimizacije. Redoslijed spajanja relacija određen je redoslijedom kojim su relacije navede u FROM dijelu SELECT naredbe. Navesti sve izraze prema kojima je obavljena procjena broja n-torki u međurezultatima. U planu naznačiti očekivani broj zapisa za međurezultate, te korištene metode pristupa. Dovoljno je nacrtati samo konačno stablo upita. **(5,5 bodova)**

7. Korisnik **k1** je vlasnik baze **legoDB** i svih objekata u njoj. Pristup bazi, bez mogućnosti kreiranja novih objekata, omogućio je samo korisnicima **k2**, **k3** i **k4**, a niti jedan korisnik nema nikakvu dozvolu nad objektima u bazi. Obavljanje prikazanih naredbi pokreće se navedenim redoslijedom, pri čemu će neke naredbe biti obavljene uspješno, a neke će rezultirati pogreškom. Korisnik pokreće naredbu bez obzira je li prethodna naredba uspješno obavljena. Za svaku naredbu navesti hoće li biti uspješno obavljena, te obavezno obrazložiti razlog uspjeha/neuspjeha njezina obavljanja.

```
1) k1: GRANT SELECT ON artikl TO k2 WITH GRANT OPTION;
2) k2: GRANT SELECT ON artikl TO k3;
3) k3: GRANT SELECT ON artikl TO k4;
4) k1: REVOKE SELECT ON artikl FROM k2 RESTRICT;
```

Koji sve korisnici imaju SELECT dozvolu nad relacijom **artikl** i je li ona prenosiva, nakon pokretanja prikazanih naredbi? **(2,5 boda)**

8. U bazi podataka se evidentiraju podaci o kandidatima, političkim strankama i mjestima, potrebni za provođenje izbora za gradonačelnika. O kandidatima za gradonačelnika se bilježi OIB, ime i prezime, datum rođenja te prebivalište. Prate se promjene prebivališta kandidata. Za prebivalište se evidentira ulica i kućni broj prebivanja kao i datum početka prebivanja na konkretnoj adresi. Osoba u jednom trenutku može prebivati samo na jednoj adresi. Za ulicu se bilježi šifra (identifikator) i naziv te mjesto u kojem se ulica nalazi, a za mjesto poštanski broj (identifikator) i naziv.

   Za političke stranke se bilježi šifra (identifikator) i naziv stranke te godina osnivanja.

   Za svako mjesto za koje se bira gradonačelnik, evidentiraju se kandidati političkih stranaka za gradonačelnika. Jedna politička stranka u jednom mjestu podržava samo jednog kandidata. Pojedinog kandidata u mjestu može podržavati više političkih stranaka.

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

9. Objasniti ulogu i opisati faze protokola dvofaznog zaključavanja (2PL). Navesti i ukratko opisati svojstvo transakcije koje taj protokol osigurava. **(4 boda)**

10. Administrator IBM Informix sustava za upravljanje bazama podataka obavlja arhiviranje na sljedeći način:
    - razina 0 svakog prvog ponedjeljka u mjesecu (npr. za lipanj 2013: 03. lipanj),
    - razina 1 svakih 7 dana (za lipanj 2013: 10. lipanj, 17. lipanj,…),
    - razina 2 svakodnevno u 03:00 sata.

    Zbog kvara na fizičkom mediju na kojem su pohranjeni podaci iz baza podataka, do kojeg je došlo 12. lipnja 2013 u 11:00 sati administrator mora obaviti obnovu. Sve arhive su ispravne.

    - Navesti što i kojim redoslijedom se koristi prilikom obnove.
    - Mogu li u postupku obnove biti izgubljeni efekti transakcija koje su potvrđene neposredno prije kvara, te transakcija koje su bile u tijeku u trenutku kvara. Obavezno obrazložiti odgovor. **(4 boda)**

Rješenja:

1. **(4 boda)**

```
   CREATE TABLE artikl (sifArtikl INTEGER PRIMARY KEY
                      , nazArtikl CHAR(250) NOT NULL
                      , vrArtikl  CHAR(1) NOT NULL CHECK (vrArtikl IN ('K','E'))
                      , cijena    DECIMAL(8,2) NOT NULL
                      , UNIQUE (nazArtikl, vrArtikl));

   CREATE TABLE komplet (sifArtikl INTEGER PRIMARY KEY REFERENCES artikl(sifArtikl)
                       , proizvOd  DATE NOT NULL
                       , proizvDo  DATE);

   CREATE TABLE element (sifArtikl   INTEGER PRIMARY KEY REFERENCES artikl(sifArtikl)
                      , vrElement   CHAR(20) NOT NULL);

   CREATE TABLE sadrzaj (sifArtiklK INTEGER REFERENCES komplet(sifArtikl)
                       , sifArtiklE INTEGER REFERENCES element(sifArtikl)
                       , boja               CHAR(30)
                       , brElem              SMALLINT NOT NULL
                       , PRIMARY KEY (sifArtiklK, sifArtiklE, boja));
```

2. **(6 bodova)**

```
CREATE PROCEDURE provjeriIzmjenu (p_sifArtikl LIKE artikl.sifArtikl,
                                  p_vrArtikl  LIKE artikl.vrArtikl)
  IF p_vrArtikl = 'K' THEN
     IF EXISTS (SELECT * FROM komplet WHERE sifArtikl = p_sifArtikl) THEN
        RAISE EXCEPTION -746, 0, 'Artikl: ' || p_sifArtikl || ' - komplet je već opisan!';
     END IF
  ELSE
     IF EXISTS (SELECT * FROM element WHERE sifArtikl = sifArtikl) THEN
        RAISE EXCEPTION -746, 0, 'Artikl: ' || p_sifArtikl || ' - element je već opisan!';
     END IF
  END IF
END PROCEDURE;

CREATE TRIGGER artiklUpd
   UPDATE OF vrArtikl ON artikl
   REFERENCING NEW AS noviArtikl OLD AS stariArtikl
   FOR EACH ROW
      WHEN (noviArtikl.vrArtikl<> stariArtikl.vrArtikl)
            (EXECUTE PROCEDURE provjeriIzmjenu (stariArtikl.sifArtikl, stariArtikl.vrArtikl));
```

3. **(3 boda)**

```
SELECT DISTINCT artikl.nazArtikl, artikl.cijena
     FROM artikl JOIN sadrzaj ON sadrzaj.sifArtiklE = artikl.sifArtikl
                 JOIN element ON artikl.sifArtikl = element.sifArtikl
    WHERE brElem >= 6
      AND boja = 'black'
      AND vrElement <> 'brick'
      AND (nazArtikl LIKE '%2x2%' OR nazArtikl LIKE '%1x2%')
   ORDER BY artikl.cijena DESC, artikl.nazArtikl
```

4. **(4 boda)**

```
  SELECT nazArtikl, (SELECT COUNT (DISTINCT sifArtiklK) FROM sadrzaj
                     WHERE sifArtiklE = artikl.sifArtikl)
    FROM artikl
   WHERE vrArtikl = 'E'
     AND sifArtikl NOT IN (SELECT DISTINCT sifArtiklE
                              FROM sadrzaj JOIN komplet
                                           ON sadrzaj.sifArtiklK = komplet.sifArtikl
                            WHERE YEAR(proizvOd) >= YEAR(TODAY)-1);
ili
SELECT nazArtikl, COUNT (DISTINCT sifArtiklK)
  FROM artikl LEFT JOIN sadrzaj ON sifArtiklE = artikl.sifArtikl
 WHERE vrArtikl = 'E'
   AND sifArtikl NOT IN (SELECT DISTINCT sifArtiklE
                           FROM sadrzaj JOIN komplet ON sadrzaj.sifArtiklK = komplet.sifArtikl
                          WHERE YEAR(proizvOd) >= YEAR(TODAY)-1)
 GROUP BY nazArtikl;
```

5. **(2 boda)**

agregacija - primjenom agregatne funkcije dobije se relacija stupnja 1 i kardinalnosti 1, npr.

```
SELECT COUNT(*) from artikl
```

6. **(5,5 bodova)**

- procjena broja n-torki u međurezultatima dobivenom selekcijama:

$sadrzaj_1 = \sigma_{boja='red'}(sadrzaj)$

$N(sadrzaj_1) = N(sadrzaj) / V(boja, sadrzaj) = 300\,000 / 30 = 10\,000$

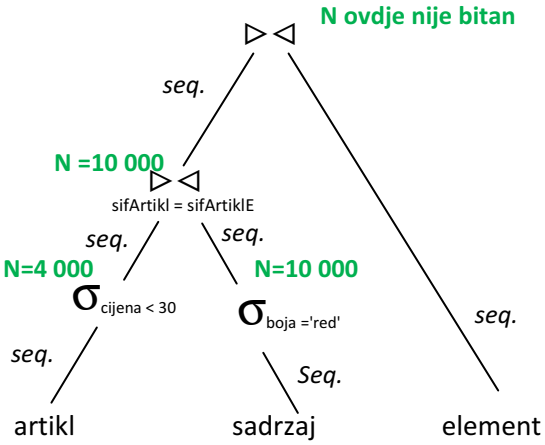
$artikl_1 = \sigma_{cijena<30}(artikl)$

$N(artikl_1) = N(artikl) / 3 = 12\,000 / 3 = 4\,000$

- procjena broja n-torki u međurezultatu dobivenom spajanjem:

$N(artikl_1 \underset{sifArtikl = sifArtiklE}{\triangleright\!\triangleleft} sadrzaj_1) = N(sadrzaj_1) = 10\,000$

($artikl_1 \cap sadrzaj_1$ je ključ relacije artikl)

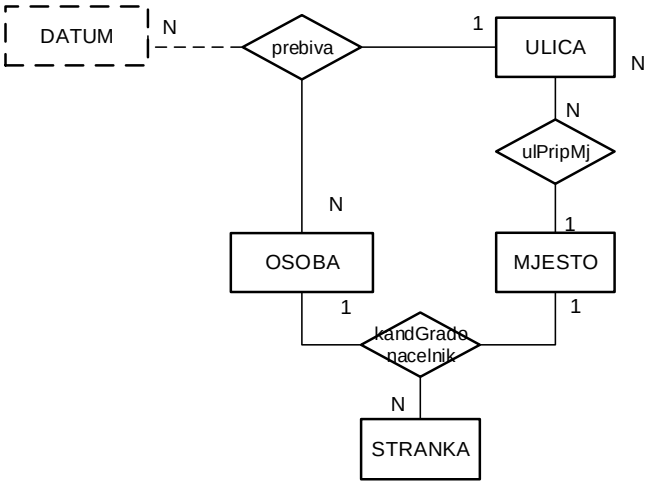


7. **(2,5 boda)**

1) k1-vlasnik baze dodjeljuje prenosivu dozvolu korisniku k2 - ok, jer je vlasnik objekta
2) k2-dobio je prenosivu dozvolu, pa daje dozvolu pregleda korisniku k3 -ok;
3) k3-nema prenosivu dozvolu pa ne može dati dozvolu pregleda korisniku k4 -pogreška;
4) k1 zbog opcije RESTRICT neće moći povući dozvolu s k2, jer k2 dodijelio dozvolu korisniku k3

Na kraju bi dozvolu imali k1, k2 i k3. Od toga k1 i k2 prenosivu.

8. **(5 bodova)**



OSOBA
OIB
imeOsoba
prezOsoba
datumRod
K = {OIB}

kandGradonacelnik
sifPolStranka
OIBKand
pbr
K1 = {sifPolStranka, OIBKandidat}
K2 = {sifPolStranka, pbr}

ULICA
sifUlica
nazivUlica
K = {sifUlica}

prebiva
sifOsoba
datOd
sifUlica
kBr
K = {sifOsoba, datOd}

MJESTO
pbr
nazMjesto
K = {pbr}

ulPripMj
sifUlica
pbr
K = {sifUlica}

STRANKA
sifStranka
nazStranka
K = {sifStranka}

9. **(4 boda)**
Protokol dvofaznog zaključavanja osigurava serijalizabilan redoslijed izvršavanja transakcija.
Sastoji se od 2 faze:
1. prije obavljanja operacije nad objektom (npr. n-torkom iz baze), transakcija mora za taj objekt zatražiti ključ
2. nakon otpuštanja ključa transakcija ne smije više zatražiti nikakav ključ

Transakcije koje poštuju 2PL protokol imaju 2 faze - fazu pribavljanja ključeva (faza rasta - ***growing phase***) i fazu otpuštanja ključeva (fazu sužavanja - ***shrinking phase***)

2PL protokol osigura svojstvo izolacije transakcije - kada se paralelno obavljaju dvije ili više transakcija, njihov učinak mora biti jednak kao da su se obavljale jedna iza druge

10. **(4 boda)**

Obnovu je moguće obaviti na sljedeći način:

1. Arhiva razine 0 od 3.lipnja
2. Arhiva razine 1 od 10.lipnja
3. Arhiva razine 2 od 12.lipnja
4. Logički dnevnik započet 12.lipnja nakon izrade arhive razine 2

Efekti transakcija koje su potvrđene prije kvara su očuvani, a efekti transakcija koje su bile u tijeku u trenutku kvara su izgubljeni.

Sve promjene koje se obavljaju nad bazom podataka se prvo bilježe u dnevniku.Sadržaj spremnika dnevnika mora biti zapisan u dnevnik prije nego što završi procedura potvrđivanja transakcije (write-ahead log rule). Dakle, u trenutku potvrđivanja transakcije sigurno postoje sve informacije o transakciji u dnevniku. Stoga se efekti transakcija koje su potvrđene prije kvara mogu ponoviti na temelju dnevnika pa su ti efekti očuvani. Efekti transakcija koje nisu završene prije kvara se poništavaju na temelju zapisa iz dnevnika (čak i u slučaju da su bili u bazu zapisani u trenutku kvara).

Završni ispit iz Baza podataka

15. lipnja 2012.

Zadaci **1-6** odnose se na bazu podataka **filmoPedia** u kojoj su kreirane relacije prikazane na slici. U bazu podataka se pohranjuju podaci o filmovima te osobama koje su u njima glumile. Prihodi su iskazani u milijunima dolara.

NULL vrijednost može poprimiti samo atribut **krajSnim** relacije **film**. Na slici **nisu** prikazane sve n-torke sadržane u relacijama.

film

sifFilm	*naslov*	*pocSnim*	*krajSnim*	*prihodFilm*
1	Prljavi Harry	01.03.1971	23.12.1971	28.1
2	Nepomirljivi	10.01.1992	07.08.1992	101.2
3	Mostovi okruga Madison	30.11.1994	02.06.1995	176.5
4	Hobbit	02.02.2012	NULL	0.0

osoba

sifOsoba	*ime*	*prezime*
1	Clint	Eastwood
2	Morgan	Freeman
3	Meryl	Streep

glumi

sifFilm	*sifOsoba*	*prihodGlumac*
2	1	5.5
2	2	4.0
3	1	5.5
3	3	10.0

U zadacima 1-3 napisati po jedan SQL upit.

1. Za sve filmove čije je snimanje započelo i završilo iste godine ispisati naslov, prihod i rashod filma. Rashod filma je prihod svih glumaca u tom filmu. Ispisati samo filmove s više od stotinu glumaca. **Zadatak riješiti bez korištenja podupita.** **(2,5 boda)**

2. Za **svaku** osobu ispisati šifru osobe, ime i prezime te broj filmova u kojima je zaradila više od 5 milijuna dolara. Rezultate poredati silazno prema broju filmova, a unutar toga abecedno po prezimenu. **Zadatak riješiti bez korištenja podupita.** **(3 boda)**

3. Za svaki film koji je zaradio više od 100 milijuna dolara, ispisati naziv filma i ime glumca (ili više njih) koji je zaradio najviše novca na tom filmu. **(2 boda)**

4. Administrator baze podataka *admin* kreirao je bazu podataka **filmoPedia** i sve prikazane relacije.

   Napisati niz naredbi koje će korisniku *admin* omogućiti da samo jednom GRANT naredbom može nekom korisniku baze podataka **filmoPedia** dodijeliti sljedeće dozvole:
   - pregled podataka relacije *glumi*
   - unos, izmjena i brisanje zapisa relacije *glumi* koji se odnose na filmove kojima snimanje nije završilo,

   a zatim napisati naredbu kojom će korisnik *admin* korisniku *brad* dodijeliti navedene dozvole. Napomena: korisniku *brad* je već omogućeno uspostavljanje SQL sjednice za rad s bazom podataka **filmoPedia**.
   (3,5 boda)

5. Napisati pohranjenu proceduru *povecajPrihodGlumac*. Ulazni argumenti procedure su šifra filma i postotak povećanja prihoda svih glumaca u tom filmu (cijeli broj, npr. 15 za traženo povećanje prihoda glumaca od 15%). Procedura obavlja traženo povećanje prihoda glumaca samo ako tim povećanjem ukupan rashod filma ne premašuje njegov prihod. U suprotnom procedura ne obavlja izmjenu prihoda glumaca, niti dojavljuje poruku da povećanje nije obavljeno. **(3 boda)**

6. Napisati SQL naredbe kojima će se osigurati da se prilikom izmjene prihoda filma obavi sljedeće: ako se prihod filma barem udvostručio, uvećati i prihode svih glumaca u tom filmu za 20%. Za povećanje prihoda glumaca koristiti pohranjenu proceduru *povecajPrihodGlumac* iz 5. zadatka. **(3 boda)**

7. Navesti razine apstrakcije na kojima se opisuje shema (struktura) baze podataka. Svaku razinu opisati jednom rečenicom. **(3 boda)**

8. Objasniti (a) čemu služi kontrolna točka i (b) što sadrži zapis kontrolne točke? **(4 boda)**

9. U bazi podataka se evidentiraju podaci o liječenju stabala u šumi. Za svako pojedino stablo evidentirana je šifra stabla, vrsta stabla (hrast, bukva, itd) i visina stabla. Stablo se tretira različitim vrstama lijekova, ponekad i s više različitih lijekova istog dana, ali nikad se isti lijek ne primjenjuje na istom stablu istog dana. Kod svake primjene lijeka evidentira se šumar koji ga je primijenio. Stablo tijekom istog dana mogu tretirati različiti šumari.

   Relacijska shema LIJECENJE sastoji se od sljedećih atributa:

   - sifStablo - jedinstvena šifra stabla
   - datPrimjene - datum primjene lijeka
   - sifVrStablo - jedinstvena šifra vrste stabla
   - nazVrStablo - naziv vrste stabla
   - sifLijek - jedinstvena šifra lijeka
   - nazLijek - naziv lijeka
   - visina - visina stabla
   - sifSumar - jedinstvena šifra šumara
   - prezime - prezime šumara

   Odrediti ključ relacijske sheme LIJECENJE tako da ona bude u 1NF, a zatim **postupno** normalizirati relacijsku shemu na 2NF i 3NF. **(4 boda)**

10. Vodi se evidencija dobavljača lijekova u ljekarnama. Za dobavljače se evidentira šifra i naziv. Za ljekarnu se evidentira matični broj koji ju jedinstveno identificira (može sadržavati slova i znamenke), te radi li se o ljekarni u privatnom vlasništvu (D/N). Za lijek se evidentira šifra, naziv i proizvođač (proizvođači imaju šifru i naziv). Pretpostavka je da jedan lijek proizvodi jedan proizvođač. Ako je lijek generički, evidentira se koji lijek je njegov originalni lijek. Za neki originalni lijek može postojati više generičkih lijekova.

    Svaka ljekarna potpisuje ugovor s dobavljačem za dobavljanje određenog lijeka, pri čemu je obavezno evidentirati ugovorenu cijenu lijeka i datum potpisivanja ugovora. Evidentirani su samo trenutno važeći ugovori. Za dobavljanje nekog lijeka ljekarna koristi samo jednog dobavljača. Dobavljač može isti lijek dobavljati za više ljekarna, a za jednu ljekarnu može dobavljati više različitih lijekova. **(8 bodova)**

    - Nacrtati ER model i opisati entitete i veze. Entitete, osim slabih entiteta, opisati isključivo vlastitim atributima. Sve sheme moraju zadovoljavati 3NF.
    - Za ER model napisati ekvivalentni relacijski model u obliku SQL naredbi za kreiranje relacija s ugrađenim integritetskim ograničenjima koja proizlaze iz modela. Tipove atributa odredite sami**.**

11. Relacija *film* kreirana je prikazanom naredbom i napunjena filmovima sa šiframa 1 do 1000:

```
CREATE TABLE film (
   sifFilm INTEGER PRIMARY KEY
 , nazFilm CHAR(100))
LOCK MODE ROW;
```

    Nad bazom podataka aktivna su samo dva korisnika: A i B. Korisnici, svaki u svojoj sjednici, izvode naredbe pomoću interaktivnog alata za izvođenje SQL naredbi. Naredbe izvode redom prema brojevima navedenim ispred naredbe: korisnik A izvede naredbu {1} do kraja, zatim korisnik B izvede naredbu {2} do kraja, itd. Korisnik kojem je zbog zaključavanja dojavljena pogreška prestaje obavljati daljnje naredbe ali ne prekida transakciju, a drugi korisnik nastavlja s radom dok ne izvede sve svoje naredbe ili dok se i njemu ne dogodi pogreška.

    Za svaku naredbu **kojom mogu biti postavljeni ključevi** napisati koja vrsta ključa se postavlja na koji objekt te kada će postavljeni ključ biti otpušten, a ako se ključ ne postavlja obrazložiti zbog čega. Ako prilikom izvođenja naredbe dođe do pogreške, obrazložiti uzrok pogreške. U zadatku se podrazumijeva korištenje IBM Informix SUBP-a. **(4 boda)**

Korisnik A	**Korisnik B**
{1} BEGIN WORK;	{2} BEGIN WORK;
{3} SET TRANSACTION ISOLATION LEVEL **SERIALIZABLE;**	{4} SET TRANSACTION ISOLATION LEVEL **READ UNCOMMITTED;**
{5} UPDATE film SET nazFilm = 'F10' WHERE sifFilm = 10;	{6} SELECT * FROM film WHERE sifFilm = 5;
{7} SELECT * FROM film WHERE sifFilm <= 10;	{8} UPDATE film SET nazFilm = 'F5' WHERE sifFilm = 5;
{9} COMMIT WORK;	{10} COMMIT WORK;

Rješenja:

1.
```
SELECT naslov, prihodFilm, SUM(prihodGlumac) AS rashod
    FROM film JOIN glumi ON film.sifFilm = glumi.sifFilm
   WHERE YEAR(pocSnim) = YEAR(krajSnim)
   GROUP BY film.sifFilm, naslov, prihodFilm
   HAVING COUNT(*) > 100;
```

2.
```
SELECT osoba.sifOsoba, ime, prezime, COUNT (glumi.sifOsoba)
  FROM osoba LEFT JOIN glumi ON osoba.sifOsoba = glumi.sifOsoba AND
                                glumi.prihodGlumac > 5
  GROUP BY osoba.sifOsoba, ime, prezime
  ORDER BY COUNT(glumi.sifOsoba) DESC, prezime ASC;
```

3.
```
SELECT naslov, ime, prezime
  FROM film JOIN glumi ON film.sifFilm = glumi.sifFilm
            JOIN osoba ON glumi.sifOsoba = osoba.sifOsoba
  WHERE prihodFilm > 100
    AND prihodGlumac = (SELECT MAX(prihodGlumac)
                          FROM glumi
                         WHERE glumi.sifFilm = film.sifFilm)
```

4.
```
CREATE ROLE uloga;
   GRANT SELECT ON glumi TO uloga;

CREATE VIEW glumi2 AS
SELECT * FROM glumi
 WHERE sifFilm IN (SELECT sifFilm FROM film
                   WHERE krajSnim IS NULL)
   WITH CHECK OPTION;
GRANT INSERT, UPDATE, DELETE ON glumi2 TO uloga;
GRANT uloga TO brad;
```

5.
```
CREATE PROCEDURE povecajPrihodGlumac (p_sifFilm INTEGER, postotak SMALLINT)
   DEFINE p_ukRashod, p_prihodFilma DECIMAL (10,2);
   SELECT SUM (prihodGlumac) INTO p_ukRashod FROM glumi
          WHERE sifFilm = p_sifFilm;
   SELECT prihodFilm INTO p_prihodFilma FROM film
      WHERE sifFilm = p_sifFilm;
   IF (p_prihodFilma >= p_ukRashod*(100+ postotak)/100) THEN
      UPDATE glumi SET prihodGlumac = prihodGlumac * (100+ postotak)/100
       WHERE sifFilm = p_sifFilm;
   END IF
  END PROCEDURE
```

6.
```
CREATE TRIGGER filmUpd
   UPDATE OF prihodFilm ON film
   REFERENCING NEW AS noviFilm OLD AS stariFilm
   FOR EACH ROW
      WHEN (noviFilm.prihodFilm >= stariFilm.prihodFilm*2)
           (EXECUTE PROCEDURE povecajPrihodGlumac (noviFilm.sifFilm, 20));
```

7. Konceptualna, unutarnja i vanjska (pred1, slide 56 do 61)

8. Korištenjem kontrolne točke ubrzava se obnova baze podataka u slučaju razrušenja, time što se u trenutku obnove logički dnevnik ne treba pretraživati od početka.
Zapis kontrolne točke sadrži listu transakcija koje su bile aktivne u trenutku kontrolne točke, te za svaku transakciju adresu najnovijeg zapisa u dnevniku.

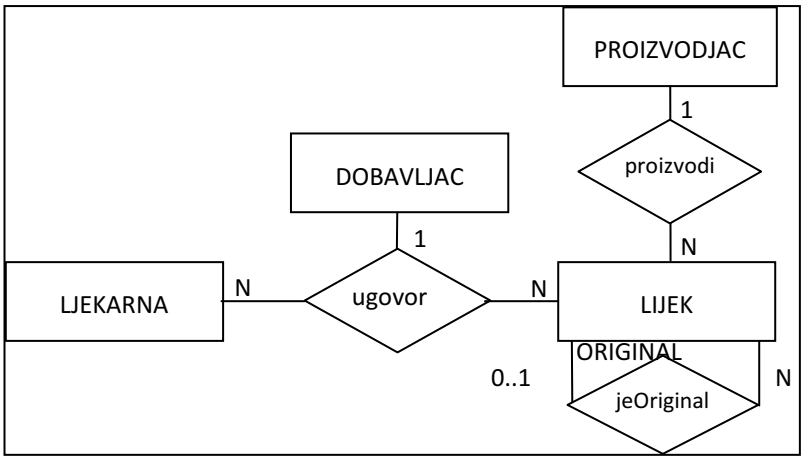
9.

1NF: LIJECENJE = { **sifStablo**, **datPrimjene**, sifVrStablo, nazVrStablo, **sifLijek**, nazLijek, visina, sifSumar, prezIme, }

2NF: STABLO = { **sifStablo**, sifVrStablo, nazVrStablo, visina }
LIJEK = { **sifLijek**, nazLijek },
LIJECENJE1 = { **sifStablo**, **datPrimjene**, **sifLijek**, sifSumar, prezIme }

3NF SUMAR = { **sifSumar**, prezIme }, K_{SUMAR} = { sifSumar }
LIJEK = { **sifLijek**, nazLijek }, K_{LIJEK} = { sifLijek }
LIJECENJE2 = { **sifStablo, datPrimjene, sifLijek**, sifSumar}
STABLO = { **sifStablo**, sifVrStablo, visina}
VRSTABLO = { **sifVrStablo**, nazVrStablo }, $K_{KATEGORIJA}$ = {sifKat}

10.




```
PROIZVODJAC = sifPr, nazPr
PK = {sifPr}

LIJEK = sifLijek, nazLijek
PK = {sifLijek}

LJEKARNA = mbr,oznPrivatna
PK = {mbr}

DOBAVLJAC = sifDobavljac,
nazDobavljac
PK = {sifDobavljac}
```

```
proizvodi = sifLijek, sifPr                                  K = {sifLijek}
jeOriginal = sifLijek, sigLijekOrig                          K = {sifLijek}
ugovor = mbr, sifLijek, sifDobavljac, datum, cijena          K = {mbr, sifLijek}
```

Relacijski model:

```
CREATE TABLE proizvodjac (sifPr INTEGER PRIMARY KEY,
                          nazPr CHAR(100);

CREATE TABLE lijek (sifLijek INTEGER PRIMARY KEY,
                    nazLijek CHAR(100),
                    sifPr INTEGER REFERENCES proizvodjac(sifPr)
                    sifLijekOrig INTEGER REFERENCES lijek (sifLijek));

CREATE TABLE dobavljac (sifDobavljac INTEGER PRIMARY KEY,
                        nazDobavljac CHAR(100);

CREATE TABLE ljekarna (mbr CHAR(15) PRIMARY KEY,
                       oznPrivatna CHAR(1) NOT NULL CHECK (oznPrivatna IN ('D', 'N'));

CREATE TABLE ugovor (mbr CHAR(15) REFERENCES ljekarna(mbr),
                     sifLijek INTEGER REFERENCES lijek(sifLijek),
                     sifDobavljac INTEGER NOT NULL REFERENCES dobavljac (sifDobavljac),
                     datum DATE NOT NULL,
                     cijena DECIMAL (8,2) NOT NULL
                     PRIMARY KEY (mbr, sifLijek))
```

11.

KORISNIK A	**KORISNIK B**
{5} postavlja ključ za pisanje nad n-torkom sa šifrom 10 relacije pred. Ključ za pisanje traje do kraja transakcije (do obavljanja naredbe 10)	{6} podaci se čitaju bez zaključavanja i bez provjere da li su možda zaključani
{7} postavlja ključ za čitanje na n-torke sa šifrom < 10 relacije pred. Postavljeni ključevi traju do kraja transakcije (do obavljanja naredbe {10}).	{8} pokušava postaviti ključ za pisanje nad n-torkom sa šifrom 5 relacije pred, ali ne uspijeva jer je nad tom n-torkom u sjednici A naredbom {5} postavljen ključ za čitanje - dojavljuje pogrešku. Prestaje obavljati daljnje naredbe i ostaje u transakciji.
{9} otpušta sve postavljene ključeve	

Završni ispit iz Baza podataka

28. lipnja 2010.

Zadaci **1** - **4** odnose se na bazu podataka ***tvprogram*** opisanu na **slici 1**. Na slici **nisu** prikazane sve n-torke koje su sadržane u relacijama.

Slika 1.

program

sifProgram	*nazProgram*
1	HTV 1
2	HTV 2
3	RTL
4	Nova TV
…	…

emisija

sifEmisija	*nazEmisija*	*sifVrstaProgram*
1	Dnevnik	2
2	TV Kalendar	2
3	Nogometno SP: Brazil - Sjeverna Koreja	1
4	Dnevnik Nove TV	2
5	Koledžicom po svijetu	4
…	…	…

raspored

sifProgram	*sifEmisija*	*datPoc*	*minPoc*	*trajanje*
1	1	15.06.2010	1170	30
4	4	15.06.2010	1155	45
2	3	15.06.2010	1220	120
3	5	15.06.2010	1200	60
1	2	15.06.2010	1140	15
…	…	…	…	…

vrstaProgram

sifVrstaProgram	*nazVrstaProgram*
1	Sport
2	Informativni program
3	Dokumentarni program
4	Filmski program
5	Zabavni program
…	…

Relacija **raspored** sadrži podatke o rasporedu emitiranja **emisija**. Emisije se emitiraju na različitim **programima**. U rasporedu se definira datum i minuta početka emitiranja (broj minuta proteklih od ponoći, npr. ako emisija počinje u 19:30, tada će minuta početka biti **19***60+**30**=1170) te trajanje emisije. Svaka emisija pripada određenoj vrsti programa. Iste emisije se mogu emitirati na različitim programima (npr. serija Seinfeld se prikazivala na 3 različita programa).

1. Napisati po jednu SQL naredbu kojom će se obaviti sljedeće:

a) Za TV programe ispisati naziv te ukupno i prosječno trajanje emisija koje u nazivu sadrže znakovni niz "gomet".
Zadatak riješiti bez podupita. **(1 bod)**

b) Za sve vrste programa ispisati broj emisija koje počinju u tekućem mjesecu (bez obzira je li ih bilo).
Zadatak riješiti bez podupita. **(2 boda)**

c) Ispisati podatke o emisijama koje se emitiraju u trenutku izvođenja upita.
Pretpostavite da postoji pohranjena funkcija *getCurrMin()* koja vraća koliko je minuta prošlo od ponoći.
Potrebno je ispisati šifru i naziv programa, naziv emisije, te opisno koliko je proteklo od početka emisije:
- „prva polovica“ ako je proteklo manje od 50% trajanja emisije
- „druga polovica“ ako je proteklo 50% ili više emisije

Zapise poredati uzlazno po šifri programa.
Zadatak riješiti bez podupita. **(2 boda)**

d) Ispisati šifru različitih emisija koje su u rasporedu za emitiranje na više od jednog programa. **(2 boda)**

e) Za svaki program ispisati šifru **posljednje** emisije koja se emitira tekućeg dana te vrijeme početka u formatu sat:minuta (npr. 23:5). **(2 boda)**

2. Napisati izraz relacijske algebre kojim će se dobiti sve emisije čiji je početak emitiranja uvijek nakon 20 sati (na bilo kojem programu). **(2 boda)**.

3. Napisati pohranjenu funkciju *provjeriEmitiranje.* Ulazni argumenti funkcije su šifra emisije i datum, a funkcija u pozivajući program vraća:
- pogrešku (*exception*) s tekstom poruke *"neispravan argument datum"*, ako se funkciji kao ulazni argument zada datum koji je manji od tekućeg datuma
- 1 ako za zadani datum postoji emitiranje emisije sa zadanom šifrom emisije
- 0 ako na zadani datum nema emitiranja emisije sa zadanom šifrom emisije **(4 boda)**

4. Administrator baze podataka **admin** kreirao je bazu podataka ***tvprogram***. Napisati niz naredbi koje će korisniku **admin** omogućiti da samo jednom GRANT naredbom može nekom korisniku dodijeliti sljedeće dozvole:
- pregled podataka relacije *emisija* i *vrstaProgram*
- unos, izmjena i brisanje zapisa relacije *raspored* koji se odnose na emisije informativnog programa s datumom početka jednakim ili većim od današnjeg

Zatim, napisati jednu naredbu kojom će korisnik **admin** korisniku **horvat** dodijeliti sve navedene dozvole.
Zadatak riješiti bez uporabe pohranjenih procedura ili funkcija. **(4 boda)**

5. Zadane su relacije r, s, t. Napišite rezultate obavljanja sljedećih operacija (SQL upite nije potrebno pisati):

r(A	B)
1	1
2	2
3	null
4	2

s(B	C)
1	1
2	2
null	2
3	3
3	null

t(C	D)
2	2
3	4
4	5
null	6

a) $r \triangleright\triangleleft^* (s \triangleright\triangleleft t)$

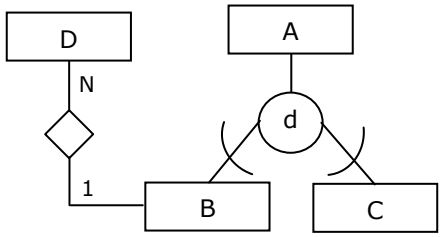
b) $\pi_C (t) \setminus \pi_C (s)$ **(2 boda)**

6. Učenici srednjih škola polažu ispite državne mature iz odabranih predmeta. Učenik za svaki predmet odabire razinu težine ispita kojeg želi polagati (npr. niža razina, viša razina). Učenik može različite predmete polagati na različitim razinama (npr. matematiku može pisati iz niže razine, a fiziku iz više razine). Za predmet se evidentira šifra i naziv, a za razinu također šifra i naziv.
Učenik ispit iz istog predmeta može polagati više puta (jer ga npr. prvi put nije položio) i pri tom ne nužno iz iste razine (npr. prvi put polaže matematiku iz više razine, a drugi put iz niže razine).
U bazi podataka evidentira se datum polaganja, broj osvojenih bodova i rang kojeg je učenik osvojio polaganjem ispita određene razine iz određenog predmeta.
Za učenike se evidentira OIB, ime i prezime te srednja škola koju pohađa. Učenik pohađa samo jednu srednju školu. Također se evidentiraju i prijave učenika za upis na studije. Učenik se može prijaviti na više studija, a za svaki odabrani studij navodi prioritet (npr. učenik koji prvenstveno želi upisati studij kineziologije, a alternativno studij medicine, navest će uz studij Kineziologije prioritet 1, a uz studij medicine prioritet 2).
Za studije se evidentira šifra i naziv, trajanje u semestrima i fakultet koji ga izvodi. Isti fakultet može izvoditi različite studije. Određeni studij izvodi samo jedan fakultet.
Fakulteti i srednje škole su obrazovne ustanove za koje se evidentira šifra i naziv te vrsta obrazovne ustanove. Za vrstu obrazovne ustanove se evidentira šifra i naziv vrste (npr. Srednja škola, Visoka škola, Fakultet,…).

Nacrtati ER model i opisati entitete i veze. Entitete, osim slabih entiteta, opisati isključivo vlastitim atributima. Sve sheme moraju zadovoljavati 3NF. **(6 bodova)**

7. Za zadani E-R model napišite ekvivalentni relacijski model u obliku SQL naredbi za kreiranje relacija s ugrađenim opisima **primarnih, alternativnih ključeva**, **pravila referencijskog integriteta i ograničenja NULL vrijednosti koja proizlaze iz modela**. Za sve atribute odaberite tip podatka INTEGER. **(3 boda)**

	vlastiti atributi entiteta	**ključevi**
entitet A	a_1 a_2 a_3	K = { a_1 }
entitet B	b_1 b_2	K = { b_1 }
entitet C	c_1 c_2	K = { c_1 }
entitet D	d_1 d_2 d_3	K = { d_1, d_2 }



Rješenja:

1.

a)

```
SELECT nazProgram, SUM(trajanje) AS ukTrajanje
                 , AVG(trajanje) AS prosjTrajanje
  FROM program JOIN raspored
                 ON program.sifProgram = raspored.sifProgram
               JOIN emisija
                 ON emisija.sifEmisija = raspored.sifEmisija
 WHERE nazEmisija LIKE '%gomet%'
 GROUP BY program.sifProgram, nazProgram
```

b)

```
SELECT nazVrstaProgram, COUNT(emisija.sifEmisija)
  FROM emisija JOIN raspored
                 ON emisija.sifEmisija = raspored.sifEmisija
                AND MONTH(datPoc)  = MONTH (TODAY)
                AND YEAR(datPoc)   = YEAR(TODAY)
         RIGHT JOIN vrstaProgram
                 ON emisija.sifVrstaProgram = vrstaProgram.sifVrstaProgram
 GROUP BY vrstaProgram.sifVrstaProgram, nazVrstaProgram
-- ili:
SELECT nazVrstaProgram, COUNT(emisija.sifEmisija)
  FROM vrstaProgram
       LEFT JOIN emisija
              ON vrstaProgram.sifVrstaProgram = emisija.sifVrstaProgram
       LEFT JOIN raspored
              ON raspored.sifEmisija = emisija.sifEmisija
             AND MONTH(datPoc) = MONTH(TODAY)
             AND YEAR(datPoc)   = YEAR(TODAY)
GROUP BY vrstaProgram.sifVrstaProgram, nazVrstaProgram
```

c)

```
SELECT  program.sifProgram
      , nazProgram
      , nazEmisija
      , CASE WHEN ((getCurrMin() - minPoc) < 0.5 * trajanje) THEN 'prva polovica'
             ELSE 'druga polovica'
        END as proteklo
  FROM raspored JOIN emisija
                  ON emisija.sifEmisija = raspored.sifEmisija
                JOIN program
                  ON raspored.sifProgram = program.sifProgram
 WHERE datPoc = TODAY
   AND getCurrMin() BETWEEN minPoc AND (minPoc + trajanje)
 ORDER BY sifProgram
```

d)

```
  SELECT DISTINCT sifEmisija
    FROM raspored
   WHERE EXISTS (SELECT *
                   FROM raspored r
                  WHERE r.sifProgram <> raspored.sifProgram
                    AND r.sifEmisija = raspored.sifEmisija
                )
```

e)

```
SELECT program.sifProgram,
       nazProgram,
       sifEmisija,
       round((minPoc - mod(minPoc,60))/60, 0) || ':' || mod(minPoc, 60) as termin
  FROM raspored JOIN program
                  ON raspored.sifProgram = program.sifProgram
 WHERE minPoc = (SELECT max(minPoc)
                   FROM raspored
                  WHERE datPoc = TODAY
                    AND sifProgram = program.sifProgram )
   AND datPoc = TODAY
```

2.

$(\pi_{\text{sifEmisija}}(\text{raspored}) \setminus \pi_{\text{sifEmisija}}(\sigma_{\text{minPoc} <= 20*60}(\text{raspored}))) \triangleright\triangleleft \text{emisija}$

3.

```
CREATE FUNCTION provjeriEmitiranje (p_datum DATE, p_sifEmisija SMALLINT)
      RETURNING SMALLINT

      IF (p_datum < TODAY) THEN
            RAISE EXCEPTION -746, 0, 'Neispravan unos datuma'
      ELIF (SELECT COUNT(*)
              FROM raspored
             WHERE datum = p_datum AND sifEmisija = p_sifEmisija) = 0 THEN
            RETURN 0;
      ELSE
            RETURN 1;
      END IF

END FUNCTION
```

4.

```
CREATE ROLE uloga;
GRANT SELECT ON emisija TO uloga;
GRANT SELECT ON vrstaProgram TO uloga;

CREATE VIEW rasporedinfo
SELECT raspored.*
  FROM raspored
 WHERE datPoc >= TODAY
   AND sifEmisija IN (SELECT *
                        FROM emisija, vrstaProgram
                       WHERE emisija.sifvrstaprogram = vrstaprogram.sifvrstaprogram
                         AND vrstaprogram.nazvrstaprogram = 'Informativni program')
WITH CHECK OPTION;

GRANT INSERT, UPDATE, DELETE ON rasporedinfo TO uloga;

GRANT uloga TO horvat;
```

5.

a)

$s \triangleright\triangleleft t$

B	C	D
2	2	2
null	2	2
3	3	4

$r \triangleright\triangleleft^* (s \triangleright\triangleleft t)$

A	B	C	D
2	2	2	2
4	2	2	2
null	null	2	2
null	3	3	4

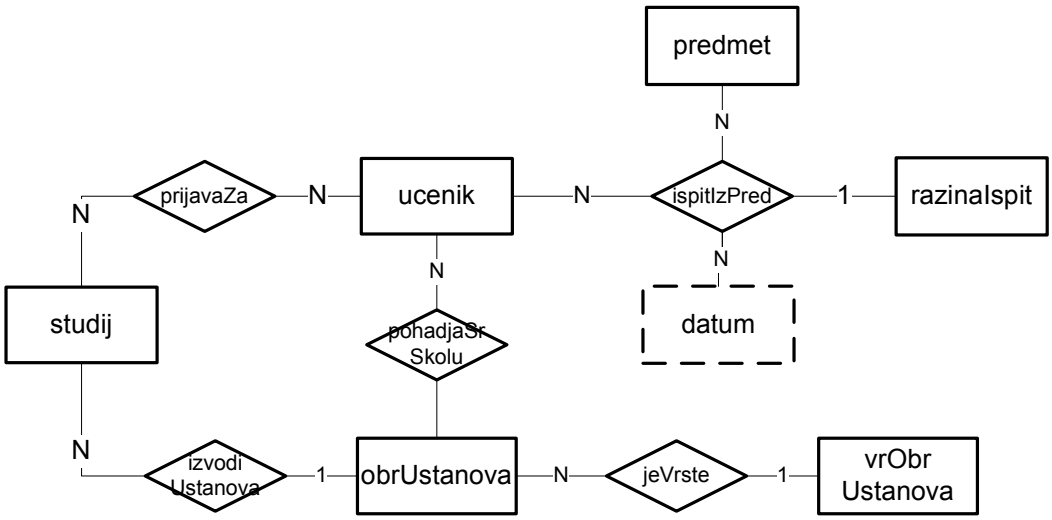
b)

$\pi_C(t) \setminus \pi_C(s)$

C
4

- Sve ili ništa (priznavati i ako su ostavili duplu n-torku u t)

6.




```
vrObrUstanova:(sifVrObrUstanova, nazivVrObrUstanova)
obrUstanova:(sifObrUstanova, nazivObrUstanova)
studij:(sifStudij, nazivStudij, trajanje)
predmet:(sifPredmet, nazivPredmet)
razinaIspit:(sifRazinaIspit, nazivRazinaIspit)
ucenik:(OIBUcenik, ime, prezime)

jeVrste: (sifObrUstanova, sifVrObrUstanova)
izvodiUstanova: (sifStudij,sifObrUstanova)
pohadjaSrSkolu: (OIBUcenik, sifObrUstanova)
prijavaZa: (OIBUcenik, sifStudij, prioritet)
ispitIzPred: (OIBUcenik, sifPredmet,datIspit, sifRazinaIspit, osvojenoBodova, rang)
```

7.

```
CREATE TABLE A (
a1 INTEGER,
a2 INTEGER,
a3 INTEGER,
PRIMARY KEY (a1))

CREATE TABLE B (
b1 INTEGER,
b2 INTEGER,
a1 INTEGER,
PRIMARY KEY (b1),
UNIQUE (a1),
FOREIGN KEY a1 REFERENCES A(a1))

CREATE TABLE C (
c1 INTEGER,
c2 INTEGER,
a1 INTEGER,
PRIMARY KEY (c1),
UNIQUE (a1),
FOREIGN KEY a1 REFERENCES A(a1))

CREATE TABLE D (
d1 INTEGER,
d2 INTEGER,
d3 INTEGER,
b1 INTEGER NOT NULL,
PRIMARY KEY (d1, d2)
FOREIGN KEY (b1) REFERENCES B(b1))
```

Završni ispit iz Baza podataka

27. lipnja 2011.

Zadaci **1-7** odnose se na bazu podataka **znanstvenaPublikacija** u kojoj su kreirane relacije prikazane na slici. Na slici **nisu** prikazane sve n-torke sadržane u relacijama.

autor

sif Autor	*ime Autor*	*prez Autor*	*datumRod*	*zadGod*
1	Onur	Hosten	15.07.1957	NULL
2	Mary	Adler	01.08.1979	NULL
3	John	Foley	01.03.1951	NULL
4	Roel	Andriga	13.02.1963	NULL
5	Berg	Eric	21.12.1982	NULL

casopis

sif Casopis	*naz Casopis*	*brIzd God*
1	Nature	12
2	Science	12
3	Astrophysical Journal	6

citiranost

sif Clanak	*godina*	*brCit*
1	2008	4439
1	2009	4830
2	2008	4092
2	2009	4446

clanak

sif Clanak	*naslovClanak*	*sif Casopis*	*datum Objava*
1	Quantum physics: How to catch a wave	1	08.06.2009
2	How the Fruit Fly Got His Spots	2	27.04.2010
3	Newtonian gravity and the Bargmann algebra	3	11.04.2011

autorClanak

sifClanak	*rbrAutor*	*sifAutor*
1	1	1
2	1	2
2	2	3
3	1	4
3	2	5
3	3	2

U bazu podataka se pohranjuju podaci o autorima članaka koji se objavljuju u časopisima. Atribut **autor.zadGod** predstavlja zadnju godinu u kojoj je autor objavio barem jedan članak. Članak može imati više autora (relacija **autorClanak)** i objavljuje se samo jednom. U relaciji **citiranost** se za članak evidentira broj citiranja članka u pojedinoj godini. Za članak koji u nekoj godini nije niti jednom citiran, za tu godinu ne postoji zapis u relaciji **citiranost**. Atribut **casopis.brIzdGod** predstavlja broj izdanja časopisa u godini. Osigurano je da nijedan atribut, osim **autor.zadGod,** ne može poprimiti NULL vrijednost.

1. Ispisati šifru, ime i prezime autora kojima je do početka godine u kojoj su navršili 30 godina starosti objavljen članak na kojem su bili prvi autor. Osigurati da se autor u popisu pojavljuje samo jednom. **Zadatak riješiti bez pod upita.** (**1.5 bod**)
2. Za godine u kojima je broj citiranja članka bio veći od prosječnog broja citiranja svih ostalih članaka, ispisati godinu citiranja, naziv članka i broj autora članka. U bazi podataka može postojati više članaka istog naziva. **(2.5 boda)**
3. Za svaki članak izdan u časopisu u kojem ove godine nije objavljen niti jedan članak, ispisati šifru članka, naziv članka, godinu u kojoj je citiran te pripadni broj citiranja. Ako za članak nije evidentirana citiranost, za godinu i broj citiranja ispisati vrijednosti 0. **(2.5 boda)**
4. Svim autorima ažurirati vrijednost atributa **zadGod.** **(1.5 bod)**

5. Napisati izraz **relacijske algebre** kojim će se za članke koji su objavljeni nakon 1.1.2011 i imaju više od tri autora, ispisati šifra, naslov i broj autora. **(2 boda)**

6. Napisati niz SQL naredbi kojima će se u relaciji **clanak** zabraniti izmjena šifre časopisa u kojem je članak objavljen. Pri pokušaju izmjene časopisa u kojem je članak objavljen potrebno je dojaviti poruku `"Promjena časopisa u kojem je članak objavljen nije dozvoljena!".`
   Uz pretpostavku da su prikazane n-torke sadržane u relacijama baze **znanstvenaPublikacija** napisati jednu naredbu čije će izvođenje pokušati narušiti pravilo o zabrani izmjene šifre časopisa u relaciji **clanak**. (**3 boda**)

7. U bazi podataka je dodatno kreirana relacija **korisnik**, koja povezuje korisnike sustava s relacijom **autor**. Vrijednost atributa **login** jednaka je korisničkom imenu (USER) s kojim korisnik uspostavlja SQL sjednicu. Kreirana je i uloga (*role*) **autorR** i svim korisnicima-autorima (*foley*, *adler*, *hosten*, ...) dodijeljena je dozvola pristupa bazi podataka te uloga **autorR**.

korisnik

login	*sifAutor*
adler	*2*
foley	*3*
horvat	*NULL*
hosten	*1*

Napisati niz SQL naredbi kojima će se svim korisnicima-autorima pomoću uloge **autorR** omogućiti pregled podataka iz relacija **casopis** i **clanak**, te izmjenu naslova članaka objavljenih u tekućoj godini, čiji su oni autori. **(4 boda)**

8. Agencija za prodaju nekretnina evidentira podatke o stanovima koji su preko te agencije ponuđeni na tržištu.

   Stanovi su šifrirani i za njih se evidentira trenutna cijena, kvadratura, mjesto i adresa. Mjesto ima jedinstveni poštanski broj te naziv. Stan može biti u zajedničkom vlasništvu više osoba (šifra, oib, prezime, ime, mjesto i adresa stanovanja), a ista osoba može biti vlasnik više stanova.

   Zaposlenici agencije i potencijalni kupci su osobe za koje se evidentiraju dodatni podaci: za zaposlenika datum zapošljavanja u agenciji i ukupna kunska vrijednost svih stanova u čijoj je prodaji posredovao, a za kupca maksimalni kunski iznos kojeg je spreman izdvojiti za kupnju stana. Kupci mogu biti i zaposlenici agencije. Sve osobe mogu biti vlasnici stana.

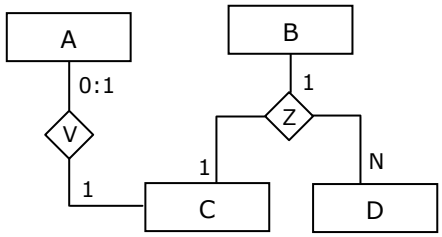
   Kupac neki stan obilazi samo jednom i to sa samo jednim zaposlenikom. Za svaki obavljeni obilazak evidentira se datum obilaska.

   Nacrtati ER model i opisati entitete i veze. Entitete, osim slabih entiteta, opisati isključivo vlastitim atributima. Sve sheme moraju zadovoljavati 3NF. (**6 bodova**)

9. Za zadani E-R model napisati ekvivalentni relacijski model u obliku SQL naredbi za kreiranje relacija s ugrađenim opisima **primarnih, alternativnih ključeva**, **pravila referencijskog integriteta i ograničenja NULL vrijednosti koja proizlaze iz modela**. Svi atributi su tipa INTEGER. **(3 boda)**

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

	vlastiti atributi veza
veza v	v_1 v_2



10. Što se zapisuje u logički dnevnik izmjena (*logical log*, *journal*) i koje operacije dnevnici omogućuju. (**2 boda**)

11. Definirati transakciju i navesti njezina četiri svojstva (nije ih potrebno objašnjavati). (**1 bod)**

12. U pravokutnik napisati SQL naredbu čije će izvođenje uzrokovati anomaliju neponovljivog čitanja. (**1 bod**)

Neponovljivo čitanje	

Transakcija A
SELECT datumObjave
  FROM clanak
  WHERE sifclanak = 2;
.
-- Ne mijenja datumObjave
-- za članak sa šifrom 2
.
.
SELECT datumObjave
  FROM clanak
  WHERE sifclanak = 2;
	Transakcija B

Rješenja:

1. (1.5 bod)

```
SELECT DISTINCT autor.sifAutor, imeAutor, prezAutor
  FROM autor JOIN autorClanak ON autor.sifAutor = autorClanak.sifAutor
             JOIN clanak ON clanak.sifClanak = autorClanak.sifClanak
 WHERE autorClanak.rbrAutor = 1
   AND YEAR(autor.datumRod)+30 > YEAR(clanak.datumObjava)
```

2. (2.5 boda)

```
SELECT godina, naslovClanak, COUNT(*)
  FROM clanak JOIN citiranost ON clanak.sifClanak = citiranost.sifclanak
              JOIN autorClanak ON autorClanak.sifClanak = clanak.sifClanak
 WHERE brCit > (SELECT AVG(c.brCit)
                  FROM citiranost c
                 WHERE c.sifClanak <> citiranost.sifClanak)
 GROUP BY godina, naslovClanak, clanak.sifClanak
```

3. (2.5 boda)

```
SELECT clanak.sifclanak, naslovClanak
     , CASE WHEN citiranost.godina IS NOT NULL THEN godina ELSE 0 END
     , CASE WHEN citiranost.brCit  IS NOT NULL THEN brCit  ELSE 0 END
  FROM clanak LEFT JOIN citiranost
              ON clanak.sifClanak = citiranost.sifClanak
  WHERE NOT EXISTS (SELECT *
                      FROM clanak clanak1
                     WHERE clanak1.sifCasopis = clanak.sifCasopis
                       AND YEAR(TODAY) = YEAR(clanak1.datumObjava))
```

ILI

```
SELECT clanak.sifclanak, naslovClanak
     , CASE WHEN godina IS NOT NULL THEN godina ELSE 0 END
     , CASE WHEN brCit  IS NOT NULL THEN brCit  ELSE 0 END
  FROM clanak LEFT JOIN citiranost
              ON clanak.sifClanak = citiranost.sifClanak
  WHERE clanak.sifcASOPIS NOT IN (SELECT DISTINCT sifCasopis
                                    FROM clanak
                                   WHERE YEAR(TODAY) = YEAR(datumObjava))
```

4. (1.5 bod)

```
UPDATE autor
   SET zadGod = (SELECT MAX(YEAR(datumObjava))
                   FROM clanak, autorClanak
                  WHERE clanak.sifClanak = autorClanak.sifClanak
                    AND autor.sifAutor = autorClanak.sifAutor);
```

5. (2 boda)

$$\sigma_{brojAutora>3}(\rho_{brAutCl(sifClanak,naslovClanak,brojAutora)}({}_{sifClanak,naslovClanak}\mathcal{G}_{COUNT(sifAutor)}(\sigma_{datumObjave>1.1.2011}(clanak \triangleright\!\triangleleft autorClanak))))$$

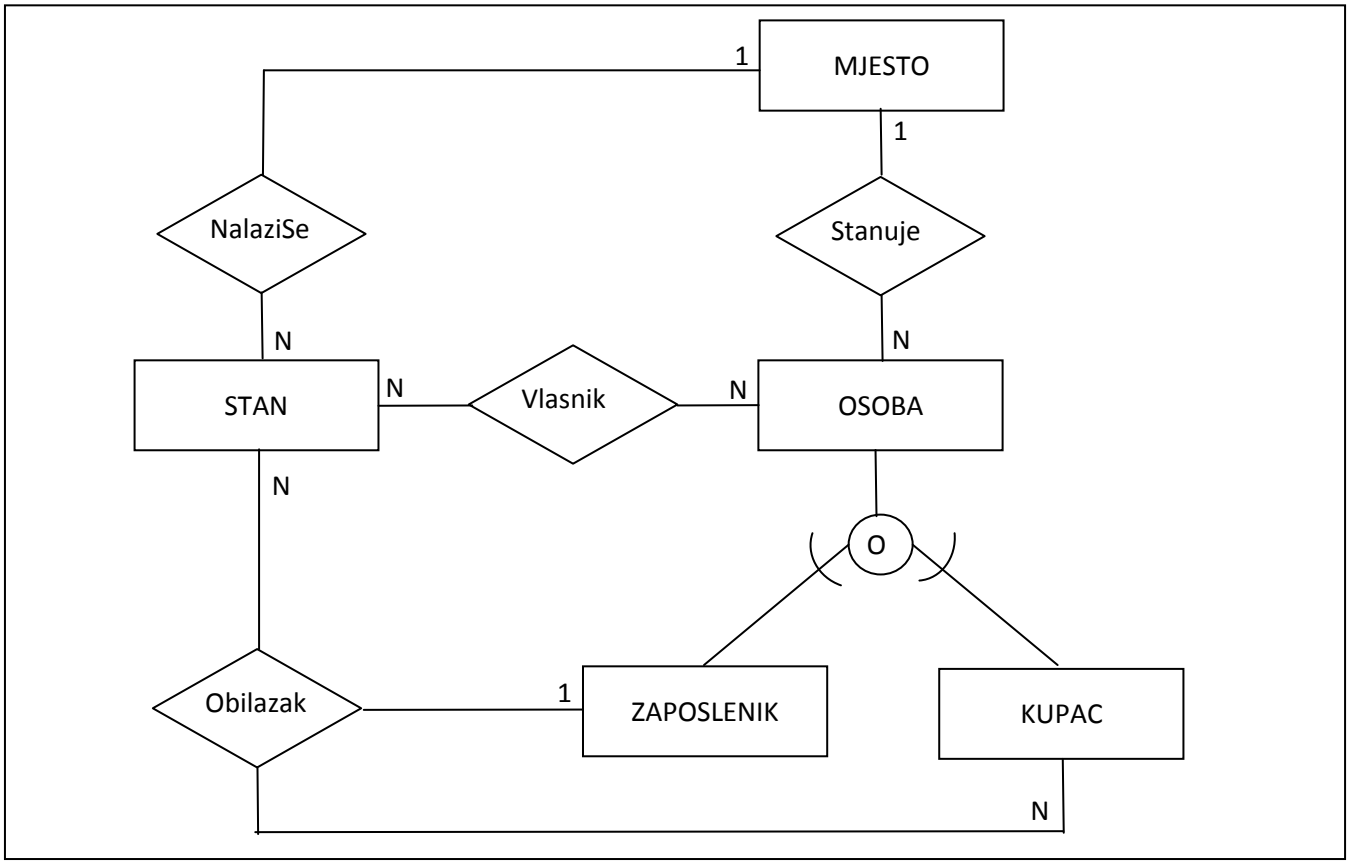
6. (3 boda)

```
CREATE PROCEDURE dojaviPogresku ()
   RAISE EXCEPTION -746, 0,
              'Promjena časopisa u kojem je  članak objavljen nije dozvoljena!';
END PROCEDURE;


CREATE TRIGGER clanakUpd
   UPDATE OF sifCasopis ON clanak
   REFERENCING NEW AS noviClanak OLD as stariClanak
   FOR EACH ROW
   WHEN (stariClanak.sifCasopis <> noviClanak.sifCasopis)
     (EXECUTE PROCEDURE dojaviPogresku ())

UPDATE clanak SET sifcasopis = 3 WHERE sifCasopis = 1;
```

7. (4 boda)

```
GRANT SELECT ON casopis TO autorR;
GRANT SELECT ON clanak TO autorR;
CREATE VIEW clanciAutorOveGodine AS
   SELECT clanak.*
     FROM clanak
    WHERE YEAR(daumObjave) = YEAR(TODAY)
      AND sifClanak IN (SELECT sifClanak
                          FROM autorClanak JOIN korisnik
                            ON autorClanak.sifAutor = korisnik.sifAutor
                         WHERE korisnik.login = USER
                        )
WITH CHECK OPTION;

GRANT UPDATE(naslovClanka) ON clanciAutorOveGodine TO autor;
```

8. (6 bodova)




```
MJESTO = pbr, nazMjesto                    PK = {pbr}
OSOBA = sifOsoba, oib, prezime, ime        K1 = {sifOsoba}   K2 = {oib } PK = K1
ZAPOSLENIK = sifOsoba, datZap, prodaoKN    PK = {sifOsoba}
KUPAC = sifOsoba, maxIznos                 PK = {sifOsoba}
STAN = sifStan, cijenaKN, kvadratura       K = {sifStan}

Stanuje = sifOsoba, pbr, adresa            K = {sifOsoba}
NalaziSe = sifStan, pbr, adresa            K = {sifStan}
Vlasnik = sifStan, sifOsoba                K = {sifStan, sifOsoba}
Obilazak = sifOsobaKupac, sifStan, sifOsobaZaposlenik, datum
                                           K = { sifOsobaKupac, sifStan }
```

9. (3 boda)

$V = \{ a_1, c_1, v_1, v_2 \}$ $K = \{ a_1 \}$

$Z = \{ b_1, c_1, d_1, d_2 \}$ $K1 = \{ b_1, d_1, d_2 \}$ $K2 = \{ c_1, d_1, d_2 \}$

Relacijski:

```
CREATE TABLE A (a1 INTEGER PRIMARY KEY,
                a2 INTEGER,
                c1 INTEGER NOT NULL REFERENCES C(c1),
                v1 INTEGER,
                v2 INTEGER,
                UNIQUE (c1));

CREATE TABLE B (b1 INTEGER PRIMARY KEY,      CREATE TABLE C (c1 INTEGER PRIMARY KEY,
                b2 INTEGER)                                  c2 INTEGER)

CREATE TABLE D (d1 INTEGER,
                d2 INTEGER,
                d3 INTEGER,
                PRIMARY KEY (d1, d2))

CREATE TABLE Z (b1 INTEGER REFERENCES B(b1),
                c1 INTEGER REFERENCES C(c1),
                d1 INTEGER,
                d2 INTEGER,
                PRIMARY KEY (b1, d1, d2),
                UNIQUE (c1, d1, d2),
                FOREIGN KEY (d1, d2) REFERENCES D(d1, d2))
```

10. (2 boda)

Svaka izmjena u bazi podataka evidentira se u logičkom dnevniku izmjena (*logical log*, *journal*)

- stara vrijednost zapisa, nova vrijednost zapisa
- korisnik, vrijeme, …
- izmjena se prvo zapisuje u dnevnik, a tek se onda provodi!

Dnevnici izmjena omogućuju

- poništavanje transakcija (važno radi svojstva nedjeljivosti)
- ponovno obavljanje transakcija (važno radi svojstva izdržljivosti)

11. (1 bod)

- jedinica rada nad bazom podataka, sastoji se od niza logički povezanih izmjena
- svojstva:
  - *Atomicity* – nedjeljivost transakcije (atomarnost)
  - *Consistency* – konzistentnost
  - *Isolation* – izolacija
  - *Durability* – izdržljivos

12. (1 bod)

Neponovljivo čitanje: bilo koja UPDATE naredba koja mijenja datumObjave za članak sa šifrom 2 – npr.

```
UPDATE clanak
   SET datumObjave = datumObjave+1
 WHERE sifClanak = 2;
```

Završni ispit iz Baza podataka

23. lipnja 2009.

Zadaci 1. – 4. odnose se na bazu podataka **upisNaFaks** koja sadrži podatke o upisnim kvotama fakulteta za pojedine studije. Podaci o fakultetima i ustanovama koje su im nadređene evidentiraju se u relaciji orgjed. Podaci o studijima koje fakulteti izvode evidentiraju se u relaciji studij. Broj kandidata koje fakulteti mogu upisati u akademskoj godini na studijima u mjestima u kojima ih izvode evidentira se u relaciji upisnaKvota. Npr. Ekonomski fakultet u Zagrebu studij *Poslovne ekonomije* izvodi u Zagrebu i u Koprivnici, pa se upisne kvote navode za isti studij u dva različita mjesta. Baza sadrži podatke o upisnim kvotama za više akademskih godina.
Studij pripada određenoj razini (relacija razinaStudij), a moguće razine su npr. "Preddiplomski studij", "Diplomski studij",itd. Za razinu studija je propisano maksimalno dozvoljeno trajanje studija u semestrima (maksDozvTrajanje). Za studij se, između ostalog, evidentira akademska godina u kojoj se studij počeo izvoditi (akGodinaOd) i akademska godina u kojoj je studij zadnji put izveden (akGodinaDo). Studiji koji se još uvijek izvode imaju nepoznatu (NULL) vrijednost atributa akGodinaDo. **Napomena**: Podvučeni atributi čine ključ relacija.

orgJed

sifOrgJed	INTEGER
nazOrgJed	CHAR(255)
sifNadOrgJed	INTEGER

razinaStudij

sifRazinaSt	SMALLINT
nazRazinaSt	CHAR(50)
maksDozvTrajanje	SMALLINT

mjesto

pBrMjesto	INTEGER
nazMjesto	CHAR(50)

upisnaKvota

sifStudij	SMALLINT
pBrMjesto	INTEGER
akGodina	SMALLINT
brojMjesta	SMALLINT

studij

sifStudij	INTEGER	
nazStudij	CHAR(255)	
sifOrgJed	INTEGER	fakultet na kojem se izvodi
sifRazinaSt	SMALLINT	razina kojoj studij pripada
trajanje	SMALLINT	u semestrima
akGodinaOd	SMALLINT	u kojoj se studij počeo izvoditi
akGodinaDo	SMALLINT	u kojoj je studij zadnji put izveden ili NULL ako će se izvoditi i u budućnosti

1. Napisati izraz relacijske algebre kojim će se ispisati šifra i naziv svih studija za koje upisna kvota nije definirana niti u akademskoj godini 2008 niti u akademskoj godini 2009. **(2 boda)**

$\pi_{\text{sifStudij, nazStudij}}(\text{studij}) \setminus \pi_{\text{sifStudij, nazStudij}}(\sigma_{\text{akGodina} = 2008 \vee \text{akGodina} = 2009}(\text{upisnaKvota} \triangleright \triangleleft \text{studij}))$

2. Napisati po jedan SQL upit kojim će se:

a) Ispisati šifru fakulteta, akademsku godinu i ukupnu upisnu kvotu studija razine "Preddiplomski studij" koje su definirane u toj akademskoj godini na tom fakultetu, ako je ukupna upisna kvota na zadanoj razini barem 1000. Zapise poredati prema akademskoj godini uzlazno, a zatim prema ukupnoj upisnoj kvoti silazno. **(1.5 bodova)**

```
SELECT studij.sifOrgJed, upisnaKvota.akGodina, SUM(brojMjesta) as ukupnaKvota
  FROM studij
         INNER JOIN razinaStudij
         ON studij.sifRazinaSt = razinaStudij.sifRazinaSt
            INNER JOIN upisnaKvota
            ON studij.sifStudij = upisnaKvota.sifStudij
   WHERE razinaStudij.nazRazinaStudij = 'Preddiplomski studij'
    GROUP BY studij.sifOrgJed, upisnaKvota.akGodina
    HAVING SUM(brojMjesta) > 1000
    ORDER BY upisnaKvota.akGodina, ukupnaKvota DESC
```

b) Za svaki studij koji u nazivu sadrži niz znakova "rač" ili niz znakova "ele" i pri tom traje 6 semestara ispisati naziv studija, naziv fakulteta na kojem se izvodi i naziv ustanove nadređene fakultetu. Ispisati podatke i za studije koji se izvode na fakultetima koji nemaju nadređenu ustanovu. **(1.5 bodova)**

```
SELECT studij.nazStudij, orgJed.nazOrgJed, nadOrgJed.nazOrgJed
  FROM studij INNER JOIN orgJed
              ON studij.sifOrgJed = orgJed.sifOrgJed
              LEFT OUTER JOIN orgJed nadOrgJed
              ON orgJed.sifNadOrgJed = nadOrgJed.sifOrgJed
  WHERE (studij.nazStudij LIKE '%rač%' OR
         studij.nazStudij LIKE '%ele%')
    AND studij.trajanje = 6
```

c) Ispisati šifru i naziv fakulteta te broj studija koji su se izvodili na tom fakultetu u akademskoj godini 2007, a za koje nije bila definirana upisna kvota u toj akademskoj godini. **(1.5 bodova)**

```
SELECT orgJed.sifOrgJed, orgJed.nazOrgJed, COUNT(studij.sifStudij)
  FROM orgJed, studij
 WHERE orgJed.sifOrgJed = studij.sifOrgJed
   AND studij.akGodinaOd <= 2007
   AND (studij.akGodinaDo IS NULL OR studij.akGodinaDo >= 2007)
   AND NOT EXISTS (SELECT * FROM upisnaKvota
                   WHERE upisnaKvota.sifStudij = studij.sifStudij
                     AND upisnaKvota.akGodina = 2007)
GROUP BY orgJed.sifOrgJed, orgJed.nazOrgJed

ILI

SELECT orgJed.sifOrgJed, orgJed.nazOrgJed, COUNT(studij.sifStudij)
  FROM orgJed, studij
 WHERE orgJed.sifOrgJed = studij.sifOrgJed
   AND studij.akGodinaOd <= 2007
   AND (studij.akGodinaDo IS NULL OR studij.akGodinaDo >= 2007)
   AND sifStudij NOT IN(SELECT sifStudij FROM upisnaKvota
                        WHERE upisnaKvota.akGodina = 2007)
GROUP BY orgJed.sifOrgJed, orgJed.nazOrgJed
```

d) Ispisati akademsku godinu, šifru i naziv studija za koje su u toj akademskoj godini definirane upisne kvote u barem dva mjesta izvođenja. Osigurati da se podaci za isti studij u ispisu pojave samo jedan put. **(1.5 bodova)**

```
SELECT upisnaKvota.akGodina , studij.sifStudij, studij.nazStudij
  FROM studij, upisnaKvota
 WHERE studij.sifStudij = upisnaKvota.sifStudij
GROUP BY upisnaKvota.akGodina, studij.sifStudij, studij.nazStudij,
HAVING COUNT(*) > 1

Ili

SELECT upisnaKvota1.akGodina studij.sifStudij, studij.nazStudij
  FROM studij, upisnaKvota  upisnaKvota1
 WHERE studij.sifStudij = upisnaKvota.sifStudij
   AND (SELECT COUNT(upisnaKvota2.postOzn)
          FROM upisnaKvota upisnaKvota2
         WHERE upisnaKvota2.akGodina = upisnaKvota1.akGodina
           AND upisnaKvota2.sifStudij = upisnaKvota1.sifStudij) > 1

Ili

SELECT DISTINCT upisnaKvota1.akGodina studij.sifStudij, studij.nazStudij
  FROM studij, upisnaKvota upisnaKvota1, upisnaKvota upisnaKvota2
 WHERE studij.sifStudij = upisnaKvota1.sifStudij
   AND studij.sifStudij = upisnaKvota2.sifStudij
   AND upisnaKvota1.akGodina = upisnaKvota2.akGodina
   AND upisnaKvota1.postOzn <> upisnaKvota2.postOzn
```

3. Pretpostavite da su u bazi podataka **upisNaFaks** kreirane sve relacije te da su definirani samo primarni ključevi. Osigurano je da izuzev atributa sifNadorgJed u relaciji orgJed i akGodinaDo u relaciji studij nijedan drugi atribut ne može poprimiti NULL vrijednost. Napisati naredbe kojima će se osigurati sljedeće:

a) Vrijednost atributa sifStudij u relaciji upisnaKvota mora biti jedna od postojećih vrijednosti atributa sifStudij u relaciji studij. U slučaju brisanja neke n-torke iz relacije studij, automatski treba obrisati i sve odgovarajuće n-torke iz relacije upisnaKvota. **(1 bod)**

```
ALTER TABLE upisnaKvota ADD CONSTRAINT
      FOREIGN KEY (sifStudij) REFERENCES studij (sifStudij)
      ON DELETE CASCADE CONSTRAINT cAkGodinaOdDo;
```

b) Akademska godina u kojoj se studij prestao izvoditi mora biti ili nepoznata (NULL) ili veća od akademske godine u kojoj se studij počeo izvoditi. **(1 bod)**

```
ALTER TABLE studij ADD CONSTRAINT CHECK (akGodinaDo IS NULL OR akGodinaDo >
akGodinaOd) CONSTRAINT cAkGodinaDo;
```

c) Ne smiju postojati dvije razine studija jednakog naziva. **(1 bod)**

```
ALTER TABLE razinaStudij ADD CONSTRAINT UNIQUE (nazRazinaSt)
               CONSTRAINT uiNazRazSt;

ili

CREATE UNIQUE INDEX uiNazRazSt ON razinaStudij(nazRazinaSt);
```

d) Studij ne smije trajati dulje od maksimalnog dozvoljenog trajanja propisanog za razinu studija kojoj pripada. Treba spriječiti narušavanje navedenog pravila operacijom izmjene n-torke u relaciji studij. Pri tom prijaviti poruku: "Prekoračeno dozvoljeno trajanje studija!“. Zanemariti činjenicu da je navedeno pravilo moguće narušiti i pri unosu n-torke u relaciju studij. Također, podrazumijevati da se vrijednost atributa maksDozvTrajanje u relaciji razinaStudij ne mijenja. (**3 boda)**

```
CREATE PROCEDURE chkStudij (p_sifRazinaSt    LIKE studij.sifRazinaSt
                          , p_Trajanje       LIKE studij.trajanje)
   IF (SELECT maksDozvTrajanje
         FROM razinaStudij
        WHERE sifRazinaSt = p_sifRazinaSt) < p_trajanje THEN
      RAISE EXCEPTION -746, 0, 'Prekoračeno maksimalno dozvoljeno trajanje studija!'
   END IF
END PROCEDURE;


CREATE TRIGGER StudijUpd
  UPDATE OF trajanje, sifRazinaSt ON studij    (ILI UPDATE ON studij)
  REFERENCING OLD AS stariStudij
              NEW AS noviStudij
  FOR EACH ROW (
    WHEN (noviStudij.trajanje <> stariStudij.trajanje OR
          noviStudij.sifRazinaSt <> stariStudij.sifRazinaSt)
    EXECUTE PROCEDURE chkStudij(noviStudij.sifRazinaSt
                              , noviStudij.trajanje);
  )
```

Podsjetnik: Postojećim tablicama ograničenja integriteta dodaju se naredbom ALTER TABLE koja ima sintaksu:

```
ALTER TABLE imeTablice ADD CONSTRAINT ograničenjeIntegriteta;
npr.  ALTER TABLE mjesto ADD CONSTRAINT PRIMARY KEY (pbrMjesto);
```

4. Vlasnik baze podataka **upisNaFaks** i svih relacija u bazi je korisnik *admin*. Osim korisnika *admin* nitko nema dozvolu pristupa bazi podataka niti objektima baze podataka. Korisnik *admin* je obavio sljedeće SQL naredbe:

```
GRANT CONNECT TO anic;
GRANT RESOURCE TO baric;
GRANT SELECT ON razinaStudij TO anic WITH GRANT OPTION;
GRANT SELECT, INSERT, UPDATE, DELETE ON mjesto TO kovac;
```

a) U nastavku je naveden niz SQL naredbi koje korisnici izvode navedenim redoslijedom. Za svaku naredbu napišite hoće li ju navedeni korisnik moći uspješno obaviti i razlog zbog kojeg naredbu može/ne može obaviti. Uzeti u obzir da se naredbe izvode slijedno i da ovise o rezultatima prethodno obavljenih naredbi. **(2 boda)**

```
anic:   GRANT SELECT ON razinaStudij TO kovac;
baric:  CREATE TABLE drzava (oznDrzava CHAR(2), nazivDrzava NCHAR(100);
baric:  GRANT CONNECT TO kovac;
kovac:  INSERT INTO mjesto VALUES (20000, 'Dubrovnik');

može jer ima SELECT ON razinaStudij WITH GRANT OPTION
može jer ima RESOURCE dozvolu
ne može jer nema DBA dozvolu
ne može jer nema CONNECT dozvolu
```

b) Kreirana je relacija korisnikDozvola pomoću koje se definiraju dozvole korisnika na fakultetima. Vrijednost atributa login jednaka je korisničkom imenu s kojim korisnik uspostavlja SQL sjednicu.

```
CREATE TABLE korisnikDozvola (
   login       CHAR(10)
 , sifOrgJed   INTEGER REFERENCES orgJed(sifOrgJed)
 , PRIMARY KEY (login, sifOrgJed) CONSTRAINT pkKorisnikDozvola
 );
```

Napisati SQL naredbu za kreiranje virtualne relacije *virtUpisnaKvota* kojom će se svim trenutnim i budućim korisnicima omogućiti unos, izmjena i brisanje podataka o upisnim kvotama svih studija koji se izvode na fakultetima za koje korisnici imaju dozvolu. **(2 boda)**

```
CREATE VIEW virtUpisnaKvota AS
SELECT upisnaKvota.*
FROM upisnaKvota
WHERE sifStudij IN (SELECT studij.sifStudij
                      FROM studij, korisnikDozvola
                     WHERE studij.sifOrgJed = korisnikDozvola.sifOrgJed
                       AND korisnikDozvola.login = USER)
                       WITH CHECK OPTION;
```

5. Na primjeru objasniti anomalije izmjene i brisanja kod loše koncipirane sheme baze podataka**.** **(2 boda)**

Izmjena: Ako neka prodavaonica promijeni adresu, promjenu adrese potrebno je obaviti na više mjesta da bi se zadržala konzistentnost podataka, npr. prodavaonica Konzum-7 se preseli na adresu "Ilica 22".

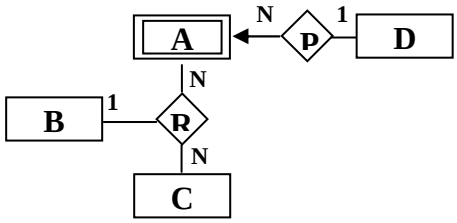
nazProd	pbr	nazMjesto	adresa	brNar	datNar	sifArtikl	nazArtikl	kolicina
Konzum-7	10000	Zagreb	Ilica 22	13/25	1.5.2006	129	Napolitanke	1200
Konzum-7	10000	Zagreb	Ilica 22	13/25	1.5.2006	139	Albert keks	2000
Diona-28	21000	Split	Bolska 7	43-21	7.2.2006	129	Napolitanke	1200
Diona-28	21000	Split	Bolska 7	43-21	7.2.2006	221	Domaćica	1800
Konzum-7	10000	Zagreb	Ilica 22	41/56	4.2.2007	129	Napolitanke	1100

Brisanje: Brisanjem svih narudžbi za neki artikl gube se podaci o artiklu, npr. ako se obriše posljednja n-torka o narudžbama artikla Domaćica, podatke o tom artiklu više nećemo imati u bazi podataka

nazProd	pbr	nazMjesto	adresa	brNar	datNar	sifArtikl	nazArtikl	kolicina
Konzum-7	10000	Zagreb	Ilica 20	13/25	1.5.2006	129	Napolitanke	1200
Konzum-7	10000	Zagreb	Ilica 20	13/25	1.5.2006	139	Albert keks	2000
Diona-28	21000	Split	Bolska 7	43-21	7.2.2006	129	Napolitanke	1200
Diona-28	21000	Split	Bolska 7	43-21	7.2.2006	221	Domaćica	1800
Konzum-7	10000	Zagreb	Ilica 20	41/56	4.2.2007	129	Napolitanke	1100

6.

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



Za zadani E-R model:

a) Opišite sheme veza **P** i **R** (njihove atribute i ključeve). **(2 boda)**

```
P = {a1, d1}, K= {a1, d1}
R = {a1, d1, b1, b2, c1}, K={a1, d1, c1}
```

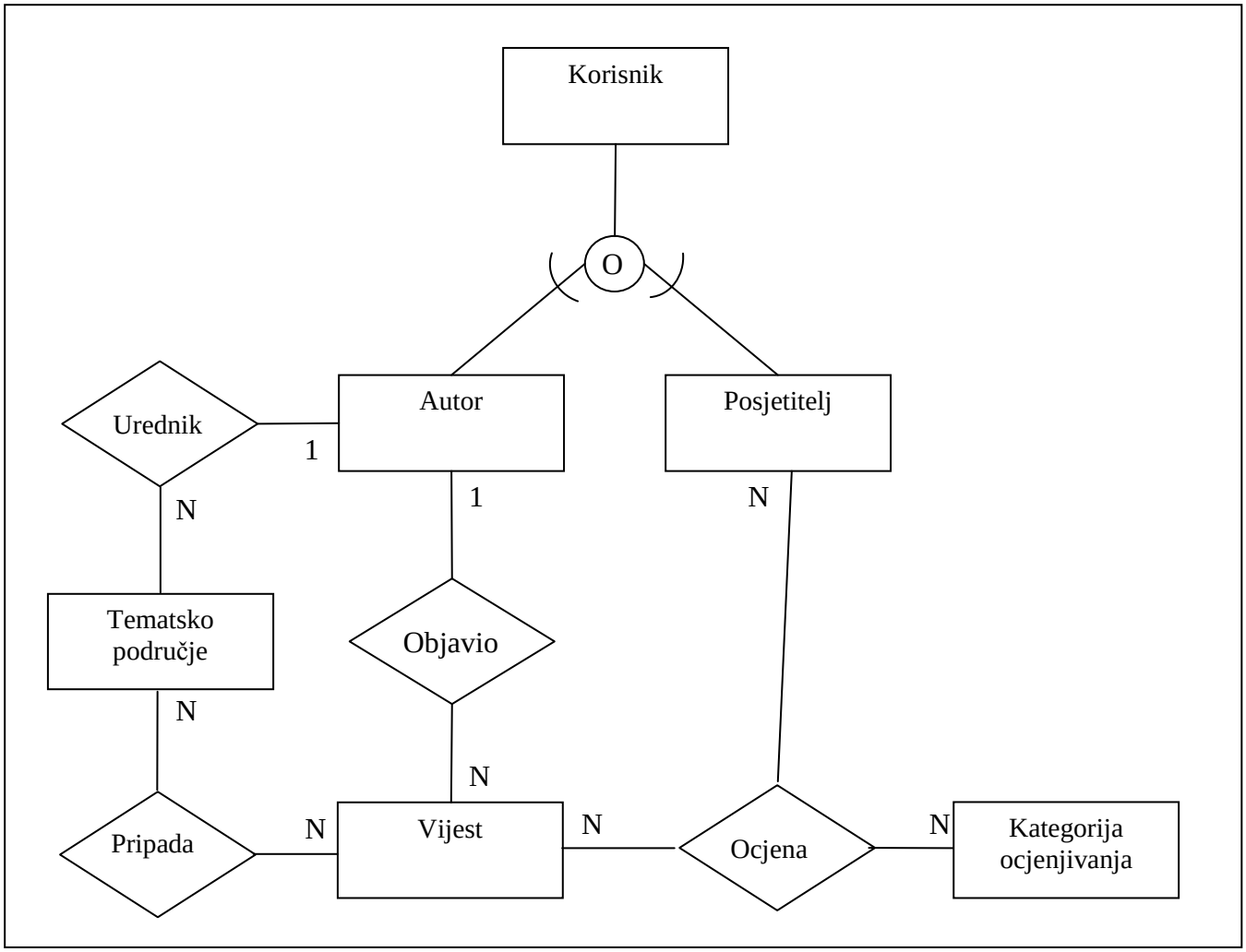
b) napišite ekvivalentni relacijski model u obliku SQL naredbi za kreiranje relacija s ugrađenim opisima **primarnih, alternativnih ključeva** i **pravila referencijskog integriteta**. Tipove podataka u naredbama ne treba navoditi. **(3 boda)**

```
CREATE TABLE D(
        d1, d2
      , PRIMARY KEY (d1)
)

CREATE TABLE A(
        a1, d1, a2
      , PRIMARY KEY (a1, d1)
      , FOREIGN KEY (d1) REFERENCES D(d1)
)

CREATE TABLE B(
        b1, b2
      , PRIMARY KEY (b1, b2)
)

CREATE TABLE C(
        c1, c2, c3
      , PRIMARY KEY (c1)
)

CREATE TABLE R (
        a1, d1, b1, b2, c1
      , PRIMARY KEY (a1, d1, c1)
      , FOREIGN KEY (a1, d1) REFERENCES A(a1, d1)
      , FOREIGN KEY (b1, b2) REFERENCES B(b1, b2)
      , FOREIGN KEY (c1) REFERENCES C(c1)
)
```

7. Na Internet portalu objavljuju se vijesti. Osobe koje koriste portal mogu bili autori vijesti ili posjetitelji (ili oboje). Za svaku osobu bilježi se šifra, korisničko ime, lozinka, ime i prezime te e-mail adresa. Za autore se dodatno bilježi broj računa na koji primaju honorare, dok se za posjetitelje bilježi oznaka jesu li pretplaćeni na mjesečne vijesti koje se automatski šalju na e-mail adresu.
Za svaku vijest bilježi se šifra, datum objavljivanja, naslov i tekst vijesti. Vijest ima samo jednog autora, i može pripadati u više tematskih područja (sport, glazba, politika ...). Tematsko područje ima šifru i naziv.
Svako tematsko područje uređuje jedan autor. Isti autor može biti urednik više tematskih područja, neovisno o tematskim područjima vijesti koje taj autor inače objavljuje.

Posjetitelji Internet portala imaju mogućnost brojčanog ocjenjivanja vijesti. Ocjene se daju za određenu vijest u različitim kategorijama ocjenjivanja (pismenost, stil, aktualnost, zanimljivost). Istu vijest u istoj kategoriji korisnik može ocijeniti samo jednom.
Za kategorije ocjenjivanja bilježi se šifra i naziv.

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




```
Korisnik:(sifKorisnik, login, psswd, ime, prezime, email)
Autor: (sifKorisnik, racun)
Posjetitelj: (sifKorisnik, pretplata)
TematskoPodrucje: (sifTemPod, nazTemPod)
Vijest: (sifVijest, datum, naslov, tekst)
KategorijaOcjenjivanja: (sifKatOcj, nazKatOcj)

Urednik: (sifTemPod, sifKorisnik)
Pripada: (sifVijest, sifTemPod)
Objavio: (sifVijest, login)
Ocjena: (sifKorisnik, sifVijest, sifKatOcj, ocjena)
```

Završni ispit iz Baza podataka

27. lipnja 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

sifProizv	INTEGER
naziv	CHAR(255)
cijena	DECIMAL(8,2)

racun

sifRac	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

sifKupac	INTEGER
imeKupac	CHAR(30)
prezKupac	CHAR(40)

djelatnik

sifDjel	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

sifRac	INTEGER	šifra računa
sifProizv	INTEGER	šifra proizvoda
kolicina	SMALLINT	broj kupljenih proizvoda

1. Napisati po jedan izraz relacijske algebre (**NE SQL UPIT**) kojim će se dobiti:
a) Relacija koja sadrži podatke o djelatnicima i kupcima kojima su djelatnici izdali račun. Relacija sadrži šifru, ime i prezime djelatnika, te šifru, ime i prezime kupca. Za djelatnike koji nisu prodali ništa, relacija sadrži samo jedan zapis u kojem su podaci o kupcu NULL.

$$\pi_{\text{sifDjel,imeDjel,prezDjel, sifKupac, imeKupac, prezKupac}} (\text{djelatnik} *\bowtie (\text{racun} \bowtie \text{kupac}))$$

b) Relacija koja sadrži šifre svih djelatnika koji nikada **nisu** prodali niti jedan proizvod s nazivom 'DVD player'.

$$\pi_{\text{sifDjel}}(\text{djelatnik}) \setminus \pi_{\text{sifDjel}}(\sigma_{\text{naziv = 'DVD player'}} (\text{proizvod} \bowtie \text{stavkaRacun} \bowtie \text{racun}))$$

2. Napisati po jedan SQL upit kojim će se:

a) Za svakog djelatnika koji je nekome nadređen ispisati šifru, ime i prezime te ukupan broj proizvoda koje su prodali njegovi **izravno** podređeni djelatnici. Ispisati samo zapise za koje je ukupan broj prodanih proizvoda veći od 10,000.

```
--1.
SELECT nad.sifDjel, nad.imeDjel, nad.prezDjel, SUM(kolicina)
  FROM djelatnik nad, djelatnik pod, racun, stavkaRacun
 WHERE pod.sifNadDjel = nad.sifDjel
   AND pod.sifDjel    = racun.sifDjel
   AND racun.sifRac = stavkaRacun.sifRac
GROUP BY nad.sifDjel, nad.imeDjel, nad.prezDjel
HAVING SUM(kolicina)>10000
```

b) Ispisati proizvode koji **nisu** prodavani u **tekućoj** godini.

```
-- 2.
```

```
SELECT *
  FROM proizvod
 WHERE sifProizv NOT IN (SELECT sifProizv
                           FROM stavkaRacun, racun
                          WHERE stavkaRacun.sifRac = racun.sifRac
                            AND YEAR(TODAY) = YEAR(datRac))
```

c) Ispisati šifre svih proizvoda koji se nalaze na barem dva različita računa.

```
-- 3.
SELECT DISTINCT sifProizv
FROM stavkaRacun sr1
WHERE EXISTS (
            SELECT *
              FROM stavkaRacun sr2
             WHERE sr1.sifRac <> sr2.sifRac
               AND sr1.sifProizv = sr2.sifProizv
      )

SELECT sifProizv
  FROM stavkaRacun
GROUP BY sifProizv
HAVING COUNT(sifRac) >= 2;
```

d) Za svaki proizvod ispisati šifru i naziv te ukupan broj prodanih primjeraka tog proizvoda. Zapise poredati silazno po broju prodanih primjeraka. U listi se moraju nalaziti i proizvodi koji nikada nisu prodani.

```
-- 4.
SELECT sifProizv, naziv, SUM(kolicina) AS brProd
  FROM proizvod LEFT JOIN stavkaRacun
                       ON proizvod.sifProizv = stavkaRacun.sifProizv
 GROUP BY sifProizv, naziv
 ORDER BY brProd DESC
```

3. Pretpostavite da su u bazi podataka **prodaja** kreirane sve relacije te da su definirani samo primarni ključevi. Napisati naredbe kojima će se osigurati sljedeće:

a) Vrijednost atributa sifRac u relaciji stavkaRacun mora biti jedna od postojećih vrijednosti atributa sifRac u relaciji racun. U slučaju brisanja neke n-torke iz relacije racun, automatski treba obrisati i sve odgovarajuće n-torke iz relacije stavkaRacun.

```
ALTER TABLE stavkaRacun ADD CONSTRAINT FOREIGN KEY (sifRac) REFERENCES racun(sifRac) ON DELETE CASCADE;
```

b) Atribut iznos u relaciji racun smije poprimiti samo vrijednosti veće od 0.

```
ALTER TABLE racun ADD CONSTRAINT CHECK iznos > 0;
```

c) Ne smiju postojati dva djelatnika s istim korisničkim imenom.

```
ALTER TABLE djelatnik ADD CONSTRAINT UNIQUE(login);
```

d) Kad god se u relaciju stavkaRacuna **unese** jedna ili više-ntorki, treba ažurirati i odgovarajući iznos u relaciji racun. Pri tome treba osigurati da ukupni iznos na nekom računu ne prijeđe iznos od 10000kn. Ukoliko se to dogodi, treba spriječiti operaciju unosa u relaciju stavkeRacuna i prijaviti poruku „Prevelik iznos računa!“.

```
CREATE PROCEDURE chkInsStavka(p_sifRac LIKE stavkaRacun.sifRac
                            , p_sifProizv LIKE stavkaRacun.sifProizv
                            , p_kolicina LIKE stavkaRacun.kolicina)

  DEFINE p_cijena LIKE proizvod.cijena;
  DEFINE p_iznos LIKE racun.iznos;
```

```
  SELECT cijena INTO p_cijena
    FROM proizvod
   WHERE sifProizv = p_sifProizv;

  SELECT iznos INTO p_iznos
    FROM racun
   WHERE sifRac = p_sifRac;

  IF p_iznos + p_cijena*p_kolicina > 10000 THEN
    RAISE EXCEPTION -746, 0, 'Prevelik iznos računa!'
  ELSE
    UPDATE racun SET iznos = iznos + p_cijena*p_kolicina
     WHERE sifRac = p_sifRac;
  END IF
END PROCEDURE;

CREATE TRIGGER stavkaIns
  INSERT ON stavkaRacun
  REFERENCING NEW AS novaStavka
  FOR EACH ROW (
    EXECUTE PROCEDURE chkInsStavka(novaStavka.sifRac
                                 , novaStavka.sifProizv
                                 , novaStavka,kolicina);
  )
```

4. Vlasnik baze podataka **prodaja** i svih relacija u bazi je korisnik *admin*. Osim korisnika *admin* nitko nema dozvolu pristupa bazi podataka niti objektima baze podataka. Napisati niz SQL naredbi koje treba obaviti korisnik *admin*, a koje će osigurati sljedeće:

a) Korisniku horvat omogućiti pregled svih podataka u relaciji djelatnik, osim korisničkog imena.

```
GRANT CONNECT TO horvat;
GRANT SELECT(sifDjel, sifNadDjel, imeDjel, prezDjel) ON djelatnik TO horvat;
```

b) Korisniku kolar omogućiti pregled i izmjenu svih podataka o kupcima s mogućnošću dodjeljivanja tih dozvola drugim korisnicima.

```
GRANT CONNECT TO kolar;
GRANT SELECT, UPDATE ON kupac TO kolar WITH GRANT OPTION;
```

c) Svim trenutnim i budućim djelatnicima omogućiti pregled, unos, izmjenu i brisanje podataka samo o onim računima koje su sami oni izdali.

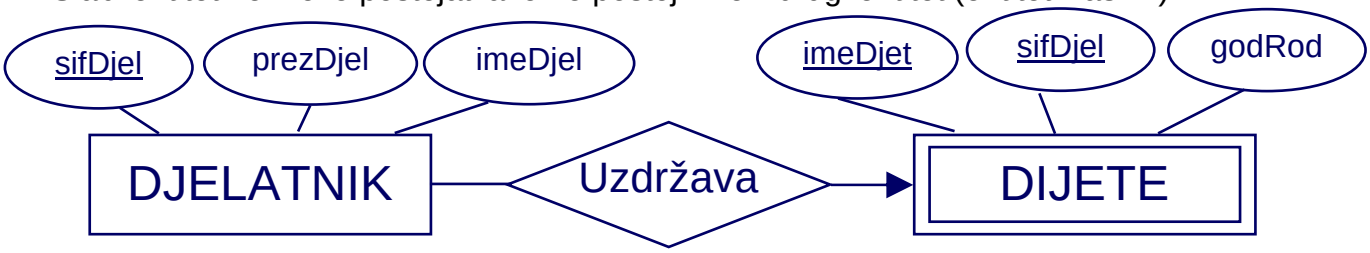
```
CREATE VIEW racunDjel AS
  SELECT * FROM racun
   WHERE sifDjel IN (SELECT sifDjel FROM djelatnik WHERE login = USER)
WITH CHECK OPTION;
GRANT CONNECT TO PUBLIC;
GRANT SELECT, INSERT, UPDATE, DELETE ON racunDjel TO PUBLIC;
```

5. Jedno od važnih svojstava transakcija osigurava da se zadatak kojeg transakcija obavlja mora ili obaviti u cijelosti i samo jednom ili se uopće ne smije obaviti. Navedite naziv tog svojstva transakcije te na vlastitom primjeru objasnite važnost tog svojstva (što bi se moglo dogoditi kad se to svojstvo ne bi poštovalo).

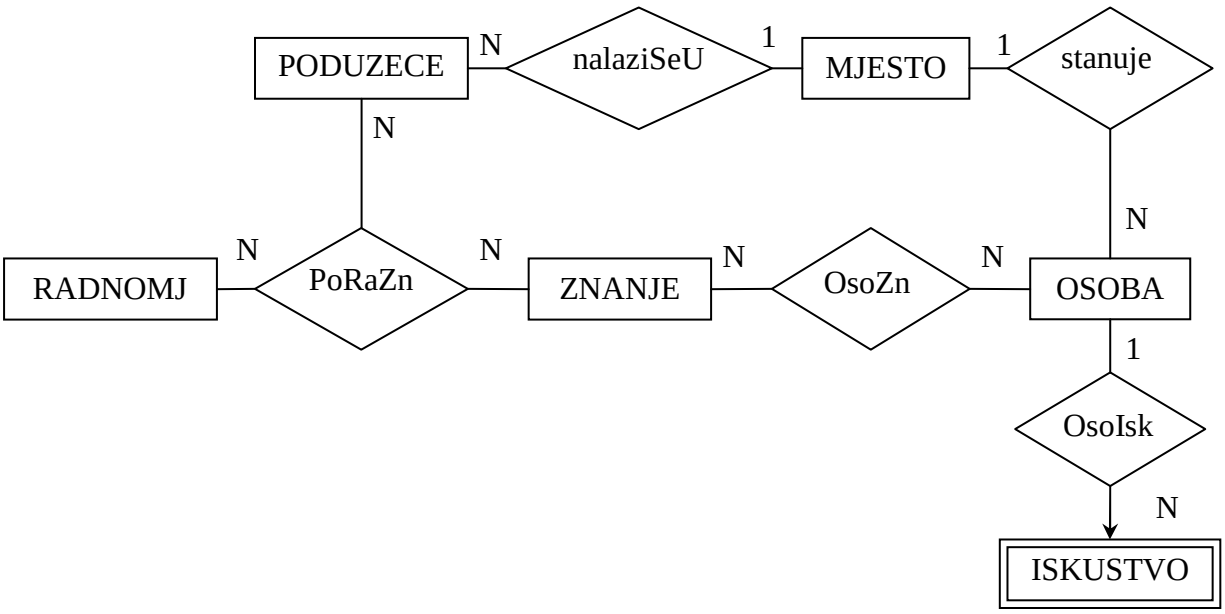
Svojstvo je atomarnost.
Primjer: Prebacivanje novca sa jednog računa na drugi. Prvo se stanje na jednom računu umanji, a zatim se uveća na drugom računu. Kada se ne bi osiguravalo svojstvo atomarnosti, tada bi se moglo dogoditi da se, zbog prekida rada sustava, samo smanji stanje na prvom računu.

6. Što su *slabi entiteti*? Navedite primjer **identifikacijski** slabog entiteta (Objasnite zašto je taj entitet identifikacijski slab).

Slabi entitet ne može postojati ako ne postoji i neki drugi entitet (entitet vlasnik).



7. Agencija za zapošljavanje prikuplja podatke o poduzećima (šifra poduzeća, naziv poduzeća, adresa poduzeća, poštanski broj, naziv mjesta), radnim mjestima (šifra radnog mjesta, naziv radnog mjesta) i potrebnim znanjima (šifra znanja, tekst koji opisuje znanje).
Svako poduzeće za svako radno mjesto za koje traži zaposlenika evidentira jedno ili više potrebnih znanja. Različita poduzeća za isto radno mjesto mogu zahtijevati različita potrebna znanja. Za potrebno znanje za neko radno mjesto u nekom poduzeću evidentira se potrebna razina znanja kao cijeli broj iz intervala [1,5].
Za osobu koja traži posao evidentiraju se osnovni podaci (JMBG, prezime, ime, adresa osobe, poštanski broj, naziv mjesta), znanja koja osoba posjeduje, uz oznaku razine [1-5] te podaci o prethodnim radnim iskustvima (redni broj radnog iskustva koji za svaku osobu počinje od 1, opis radnog iskustva).
Nacrtati ER model i opisati entitete i veze. Entitete, osim slabih entiteta, opisati isključivo vlastitim atributima. Sve sheme moraju zadovoljavati 3NF.



PODUZECE: sifP nazivP	**MJESTO**: pbr nazivMj	**RADNOMJ**: sifRM nazivRM	**ZNANJE**: sifZn opisZn	**OSOBA**: JMBG ime prezime	**ISKUSTVO**: JMBG rbr OpisIsk
PoRaZn: sifP sifRM sifZn razina	**OsoZn**: JMBG sifZn razina	**nalaziSeU**: sifP pbr adresaP	**stanuje**: JMBG pbr adresaO	**OsoIsk**: JMBG rbr	

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"

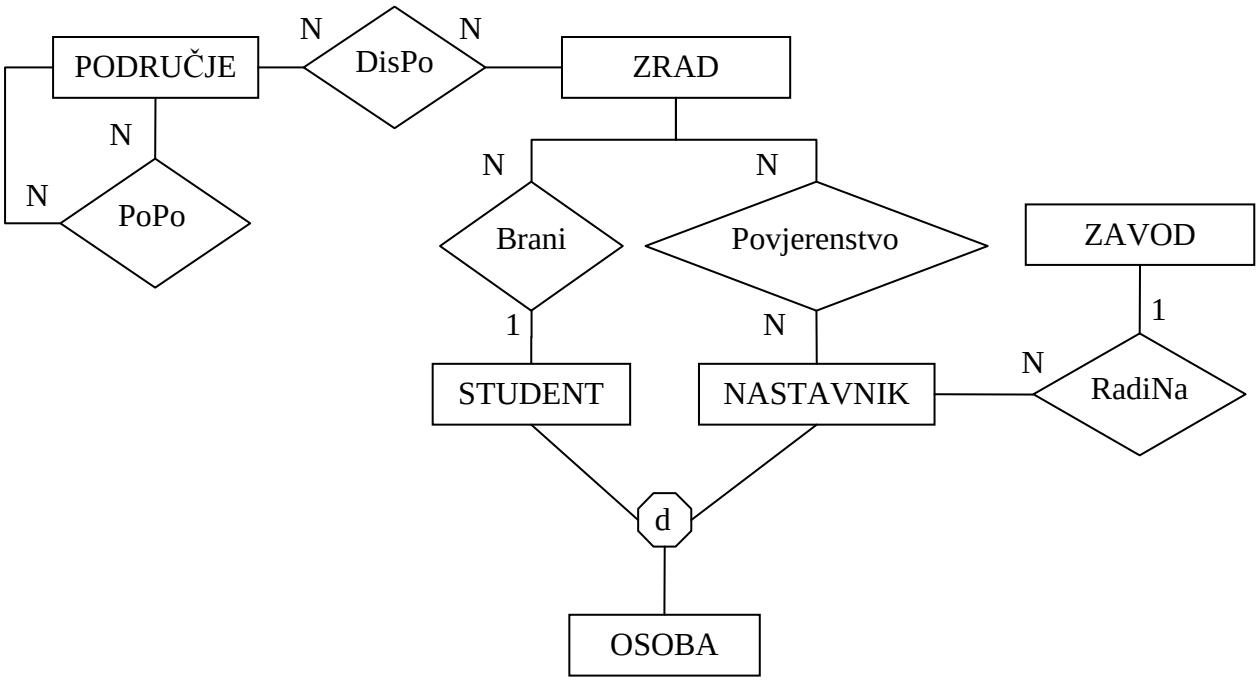
7. Oblikovati bazu podataka o završnim radovima studenata FER-a. Evidentirati podatke o studentima,završnim radovima, područjima i članovima povjerenstva za obranu.

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.

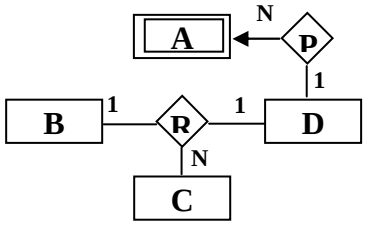
Nacrtati ER model i opisati entitete i veze. Entitete, osim slabih entiteta, opisati isključivo vlastitim atributima. Sve sheme moraju zadovoljavati 3NF. **(5 bodova)**



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)
);
```

Završni ispit iz Baza podataka

28. lipnja 2007.

Zadaci 1 – 4 se odnose na bazu podataka **provjera** koja sadrži podatke o studentima (relacija **stud**), predmetima (relacija **pred**), upisanim predmetima studenata (relacija **upisanPred**) i rezultatima koje su studenti postigli na provjerama znanja iz predmeta (relacija **rezProvjera**). Podvučeni atributi čine primarni ključ relacije.

stud

JMBAG	CHAR(10)	JMBAG studenta
ime	NCHAR(40)	ime studenta
prezime	NCHAR(40)	prezime studenta

pred

sifPred	SMALLINT	šifra predmeta
naziv	NCHAR(120)	naziv predmeta
ECTSBod	DECIMAL(3,1)	ECTS bodovi predmeta

upisanPred

JMBAG	CHAR(10)	
sifPred	SMALLINT	
ocjena	SMALLINT	ocjena koju je student dobio iz predmeta (NULL, 1, 2, 3, 4 ili 5).

rezProvjera

JMBAG	CHAR(10)	
sifPred	SMALLINT	
rbrProvjera	SMALLINT	
brBod	DECIMAL(3,1)	broj bodova koje je student osvojio na toj provjeri iz predmeta

1. Napisati SQL upite koji će:

   a) ispisati šifre, nazive predmeta i broj dobivenih ocjena izvrstan (5) za one predmete na kojima je više od 100 studenata dobilo ocjenu izvrstan. **Zadatak riješiti bez upotrebe podupita**. ***(2 boda)***

   b) **za svakog** studenta (uključujući i one koji nisu upisali niti jedan predmet) ispisati JMBAG, ime, prezime, broj upisanih predmeta i prosječnu ocjenu upisanih predmeta. ***(2 boda)***

   c) ispisati JMBAG i naziv predmeta za svaki studentov **upisani** predmet iz kojeg **nije obavio niti jednu provjeru**. ***(2 boda)***

   d) svaku ocjenu nedovoljan (1) prepraviti u ocjenu dovoljan (2) ukoliko je student na provjerama iz dotičnog predmeta skupio više od 30 bodova. ***(2 boda)***

2. Napisati izraze relacijske algebre (ne SQL upit) kojim će se dobiti:

   a) relacija koja sadrži **različite** JMBAG-ove studenata koji su upisali barem jedan predmet s više od 6 ECTS bodova (***1 bod)***

   b) relacija s atributima jmbag, ime, prezime koja sadrži podatke o studentima koji su položili predmet sa šifrom 1001, ali nisu položili predmet sa šifrom 1002. ***(3 boda)***

3. Napisati naredbe kojima će se osigurati sljedeće:

   a) atribut brBod u relaciji rezProvjera mora biti veći ili jednak 0.0 ***(1 bod)***

   b) u relaciji **pred** ne smiju postojati dva predmeta istog naziva s istim brojem ECTS bodova. ***(1 bod)***

   c) nije dozvoljena izmjena vrijednosti atributa ECTSBod za predmete koje je upisao jedan ili više studenata. Pokušaj izmjene spriječiti iznimkom koja signalizira pogrešku s tekstom 'Izmjena nije dozvoljena'. ***(3 boda)***

4. Vlasnik baze podataka **provjera** i svih relacija u bazi podataka je korisnik *admin*. Niti jedan drugi korisnik nema nikakvu dozvolu. Napisati niz SQL naredbi koje treba obaviti korisnik *admin*, a koje će osigurati da korisnici *anic*, *babic* i *caric* imaju SVE potrebne dozvole za obavljanje sljedećih operacija:

korisnik:	**operacije**
anic	pregled svih podataka u relaciji stud s mogućnošću dodjeljivanja tih dozvola drugim korisnicima
babic	pregled i izmjenu podataka o predmetima (relacija **pred**) koje je upisao barem jedan student
caric	pregled JMBAG-a i prosjeka ocjena položenih predmeta za svakog studenta (korisnik caric smije vidjeti samo sumarne podatke, ali ne i podatke o pojedinim ocjenama iz predmeta)

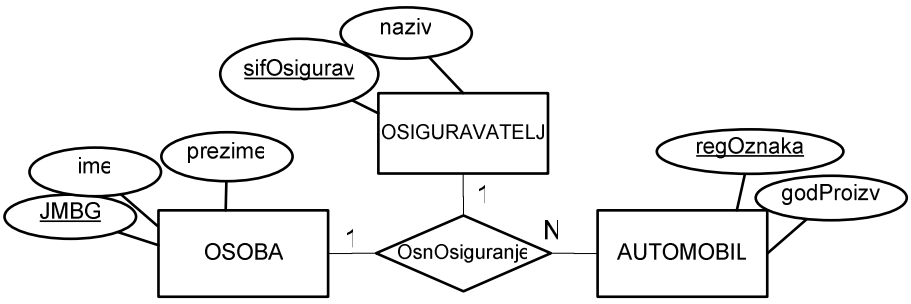
(4 boda)

5. Tijekom ljeta na Jadranu se održavaju jednodnevne jedriličarske regate. Istog dana se može održati nekoliko različitih regata. Svaka regata ima samo jednog sponzora za kojeg se evidentira šifra i naziv sponzora. Jedan sponzor može sponzorirati nekoliko regata, ali regate istog sponzora sigurno počinju različitih datuma. Za svaku se regatu, pored sponzora, evidentira datum održavanja regate, trajanje regate izraženo u satima, grad iz kojeg regata isplovljava i grad u kojeg regata uplovljava. Za gradove se evidentiraju poštanski brojevi i nazivi gradova.

   Za svaku pojedinu jedrilicu koja sudjeluje u regati evidentira se registarski broj koji jedinstveno identificira jedrilicu, ime jedrilice i najveći dopušteni broj članova posade. Za jedriličara se evidentira jmbg, prezime i ime, te u kojoj regati plovi na kojoj jedrilici. Jedriličar ne mora na svakoj regati ploviti na istoj jedrilici, a očito je da tijekom iste regate ne može ploviti na dvije jedrilice.

   Nacrtati ER model i opisati entitete i veze. Entitete, osim slabih entiteta, opisati isključivo vlastitim atributima. Sve sheme moraju zadovoljavati 3NF. ***(6 bodova)***

6. Napišite SQL naredbe za kreiranje relacija s ugrađenim integritetskim ograničenjima (integritet ključa, entitetski integritet, referencijski integritet) kojima će se prikazani ER model preslikati u ekvivalentan **relacijski model**. Ključevi entiteta su podvučeni. Tipove podataka pojedinih atributa odaberite po nahođenju.



(3 boda)

RJEŠENJA:

1.

a)

```
SELECT pred.sifPred, naziv, COUNT(*) as br5
  FROM  pred, upisanpred
 WHERE pred.sifPred  = upisanPred.sifPred
   AND ocjena = 5
 GROUP BY pred.sifPred, naziv
HAVING COUNT(*) > 100;
```

b)

```
SELECT stud.jmbag, ime, prezime, COUNT(sifPred) AS brUpPred, AVG(ocjena) AS
prOcj
  FROM stud LEFT JOIN upisanPred
                  ON stud.jmbag = upisanPred.jmbag

GROUP BY stud.jmbag, ime, prezime;
```

ILI

```
SELECT jmbag, ime, prezime,
(SELECT COUNT(sifPred) FROM upisanPred
  WHERE stud.jmbag = upisanPred.jmbag),
(SELECT AVG(ocjena) FROM upisanPred
  WHERE stud.jmbag = upisanPred.jmbag
    AND upisanPred.ocjena >1)
  FROM stud
```

c)

```
SELECT jmbag, naziv
  FROM upisanPred, pred
 WHERE upisanPred.sifPred = pred.sifPred
      AND NOT EXISTS (
         SELECT *
           FROM rezProvjera
          WHERE rezProvjera.sifPred = upisanPred.sifPred
            AND rezProvjera.jmbag   = upisanPred.jmbag);
```

d)

```
UPDATE upisanPred
   SET ocjena = 2
 WHERE ocjena = 1
   AND (SELECT SUM(brBod)
          FROM rezProvjera
         WHERE rezProvjera.sifPred = upisanPred.sifPred
           AND rezProvjera.jmbag = upisanPred.jmbag) > 30;
```

2.

a)

$$\pi_{\text{jmbag}}\left(\sigma_{\text{ECTSbod} > 6.0}(\text{pred}) \bowtie \text{upisanPred}\right)$$

b)

$$\text{stud} \bowtie \left(\pi_{\text{jmbag}}\left(\sigma_{\text{sifPred}=1001 \wedge \text{ocjena}>1}(\text{upisanPred})\right) \setminus \pi_{\text{jmbag}}\left(\sigma_{\text{sifPred}=1002 \wedge \text{ocjena}>1}(\text{upisanPred})\right)\right)$$

3.

a)

```
ALTER TABLE rezProvjera ADD CONSTRAINT CHECK (brBod >= 0.0))
      CONSTRAINT cBrBod;
```

b)

```
ALTER TABLE pred ADD CONSTRAINT UNIQUE (naziv, ECTSBod) CONSTRAINT uiPred;
```

ILI

```
CREATE UNIQUE INDEX uiPred ON pred (naziv, ECTSBod);
```

c)

```
CREATE PROCEDURE dojaviPogresku (p_tekstPogreske CHAR(74))
   RAISE EXCEPTION -746, 0, p_tekstPogreske;
END PROCEDURE;

CREATE TRIGGER predUpd
   UPDATE OF ECTSBod ON pred
   REFERENCING OLD AS oldPred NEW AS newPred
   FOR EACH ROW
   WHEN (oldPred.ECTSBod <> newPred.ECTSBod
      AND EXISTS (SELECT *
                    FROM upisanPred
                   WHERE upisanPred.sifPred = oldPred.sifPred))
      (EXECUTE PROCEDURE dojaviPogresku ('Izmjena nije dozvoljena'));
```

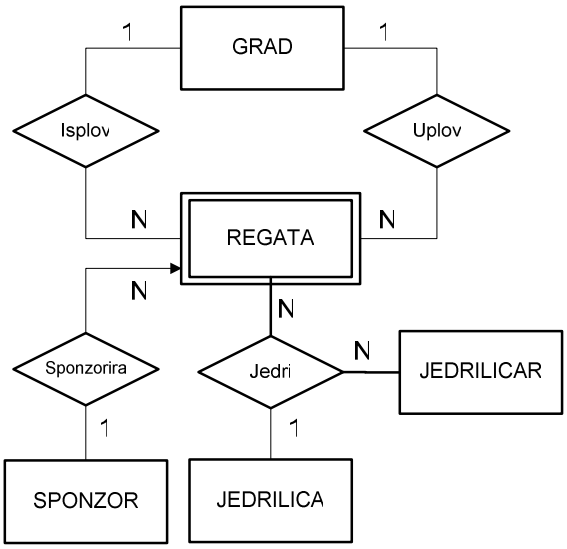
ILI

```
CREATE PROCEDURE chkUpdPred (p_sifPred LIKE pred.sifPred)
   DEFINE cnt SMALLINT;

   SELECT COUNT(*)
     INTO cnt
     FROM upisanPred
    WHERE upisanPred.sifPred = p_sifPred;
   IF cnt > 0 THEN
      RAISE EXCEPTION -746, 0, 'Izmjena nije dozvoljena';
   END IF
END PROCEDURE;

CREATE TRIGGER predUpd
   UPDATE OF ECTSBod ON pred
   REFERENCING OLD AS oldPred NEW AS newPred
   FOR EACH ROW
   WHEN (oldPred.ECTSBod <> newPred.ECTSBod)
      (EXECUTE PROCEDURE chkUpdPred (oldPred.sifPred));
```

4.

```
GRANT CONNECT TO anic;
GRANT CONNECT TO babic;
GRANT CONNECT TO caric;

GRANT SELECT ON stud TO anic WITH GRANT OPTION;

CREATE VIEW pred1 AS
SELECT * FROM pred
 WHERE EXISTS (SELECT * FROM upisanPred WHERE pred.sifPred =
upisanPred.sifPred)
  WITH CHECK OPTION;
GRANT SELECT, UPDATE ON pred1 TO babic;


CREATE VIEW pred2 (JMBAG, prosjek) AS
SELECT JMBAG, AVG(ocjena)
  FROM upisanPred
   WHERE ocjena > 1
GROUP BY JMBAG;
GRANT SELECT ON pred2 TO caric;
```

5.



GRAD
pbr
nazGrad
K={pbr}

SPONZOR
sifSponzor
nazSponzor
K={sifSponzor}

JEDRILICA
regBrJedr
imeJedr
maxClanova
K={regBrJedr}

JEDRILICAR
JMBG
prezime
ime
K={JMBG}

REGATA
sifSponzor
datRegata
trajanje
K={sifSponzor, datRegata}

ISPLOV
sifSponzor
datRegata
pbrIsplov
K={sifSponzor, datRegata}

UPLOV
sifSponzor
datRegata
pbrUplov
K={sifSponzor, datRegata}

SPONZORIRA
sifSponzor
datRegata
K={sifSponzor, datRegata}

JEDRI
sifSponzor
datRegata
JMBG
regBrJedr
K={ sifSponzor, datRegata, JMBG}

6.

```
CREATE TABLE osoba (
  JMBG      CHAR(13)    PRIMARY KEY CONSTRAINT pkOsoba
, ime       NCHAR(40)
, prezime   NCHAR(40)
);

CREATE TABLE osiguravatelj(
     sifOsigurav  SMALLINT  PRIMARY KEY CONSTRAINT pkOsiguravatelj
   , naziv        NCHAR(120));

CREATE TABLE automobil(
     regOznaka      CHAR(20) PRIMARY KEY CONSTRAINT  pkAutomobil
   , godProizv      SMALLINT
   );

CREATE TABLE osnOsiguranje(
     JMBG         CHAR(13)
   , sifOsigurav  SMALLINT
   , regOznaka    CHAR(20)

   , PRIMARY KEY (regOznaka, JMBG) CONSTRAINT pkOsnOsiguranje
   , UNIQUE      (regOznaka, sifOsigurav) CONSTRAINT uiOsnOsiguranje
   , FOREIGN KEY (JMBG)        REFERENCES osoba (JMBG)                CONSTRAINT fk1
   , FOREIGN KEY (sifOsigurav) REFERENCES osiguravatelj(sifOsigurav) CONSTRAINT fk2
   , FOREIGN KEY (regOznaka)   REFERENCES automobil(regOznaka)       CONSTRAINT fk3
);
```

Ponovljeni završni ispit iz Baza podataka

5. srpnja 2007.

knjiga

sifKnjiga	INTEGER	šifra knjige
naslov	NCHAR(250)	naslov knjige

primjerak

invBroj	CHAR(10)	inventarski broj primjerka
sifKnjiga	INTEGER	
cijena	DECIMAL(5,2)	cijena po kojoj je primjerak kupljen

stud

JMBAG	CHAR(10)	JMBAG studenta
ime	NCHAR(40)	ime studenta
prezime	NCHAR(40)	prezime studenta
oznRedovit	CHAR(1)	oznaka da li je student redovit ('D' ili 'N')

posudba

JMBAG	CHAR(10)	
invBroj	CHAR(10)	
datumPos	DATE	datum posudbe
datumDo	DATE	datum do kada student mora vratiti posuđeni primjerak
datumVra	DATE	datum kada je student vratio primjerak, NULL ako knjiga nije vraćena

1. Napisati SQL upite koji će:

a) ispisati ukupnu vrijednost svih **primjeraka** knjiga koji su **uvijek bili vraćeni na vrijeme**.
U obzir uzeti samo one zapise kojima **je poznato** vrijeme povrata knjige. (Jedan primjerak knjige u ukupnoj sumi sudjeluje samo jednom.) **Zadatak riješiti bez upotrebe podupita**. ***(2 boda)***

```
SELECT SUM(cijena)
FROM primjerak
WHERE invBroj NOT IN (SELECT DISTINCT invBroj FROM posudba WHERE datumDo <= datumVra)
```

NAPOMENA: U ovom zadatku je pogrešno pisalo da se treba riješiti bez upotrebe podupita. To će se uzimati u obzir prilikom ispravljanja.

b) **za svaku** knjigu ispisati šifru, naslov, broj primjeraka i ukupan broj posudbi (primjeraka) te knjige. **Zadatak riješiti bez upotrebe podupita**. ***(2 boda)***

```
SELECT knjiga.sifKnjiga, naslov, COUNT(DISTINCT primjerak.invBroj), COUNT(datumPos)
  FROM knjiga JOIN primjerak
                ON knjiga.sifKnjiga = primjerak.sifKnjiga
  LEFT  OUTER JOIN posudba
                ON primjerak.invBroj = posudba.invBroj
 GROUP BY knjiga.sifKnjiga, naslov;
```

c) **za svakog** studenta ispisati JMBAG, inicijale (atribut nazvati inicijali), broj primjeraka knjiga koje **je posudio i vratio** (atribut nazvati brVra) i broj primjeraka knjiga koje je **posudio a nije vratio** (atribut nazvati brDug). ***(2 boda)***

Npr.

JMBAG	inicijali	brVra	brDug
1234567890	I.K.	10	1
...	...	...	...

student s JMBAG-om 1234567890 je vratio 10 primjeraka a nije vratio 1 primjerak.

```
SELECT stud.JMBAG
     , substring(ime from 1 for 1) || '.' ||
       substring(prezime from 1 for 1) || '.' as inicijali
     , (SELECT COUNT(*) FROM posudba
         WHERE jmbag = stud.jmbag
           AND datumVra IS NOT NULL) as brVra
     , (SELECT COUNT(*) FROM posudba
         WHERE jmbag = stud.jmbag
           AND datumVra IS NULL) as brDug
  FROM stud;
```

d) svim **redovitim** studentima koji nisu vratili knjige čiji primjerci koštaju manje od 50 kn postaviti datum povrata knjige na današnji dan. ***(2 boda)***

```
UPDATE posudba
   SET datumVra = TODAY
 WHERE invBroj  IN (SELECT invBroj FROM primjerak WHERE cijena < 50)
   AND jmbag    IN (SELECT jmbag FROM stud WHERE oznRedovit = 'D')
   AND datumVra IS NULL;
```

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

a) $(\pi_{\text{JMBAG, invBroj}}(\sigma_{\text{datumPos='1.1.2006'}}(\textbf{posudba})) \setminus \pi_{\text{JMBAG, invBroj}}(\sigma_{\text{datumPos='1.1.2007'}}(\textbf{posudba}))$ ***(2 boda)***

```
SELECT DISTINCT JMBAG, invBroj
  FROM posudba
 WHERE datumPos = '1.1.2006'
   AND NOT EXISTS (
      SELECT *
        FROM posudba p2
       WHERE posudba.JMBAG   = p2.JMBAG
         AND posudba.invBroj = p2.invBroj
         AND datumPos = '1.1.2007'
   );
```

b) JMBAG,ime,prezime$\mathscr{G}_{\textbf{COUNT(invBroj)}}(\textbf{stud} * \triangleright\triangleleft \textbf{posudba} \triangleright\triangleleft \textbf{primjerak})$ ***(2 boda)***

```
SELECT stud.jmbag, ime, prezime, COUNT(posudba.invBroj)
  FROM primjerak JOIN posudba
                   ON primjerak.invBroj = posudba.invBroj
            RIGHT JOIN stud
                   ON posudba.jmbag = stud.jmbag
GROUP BY stud.jmbag, ime, prezime;
```

3. Nadopunite započeti tekst definicije prirodnog spajanja: ***(2 boda)***

Zadane su relacije r(R) i s(S). Neka je $R \cap S = \{ A_1, A_2, ..., A_n \}$.
Obavljanjem operacije $r \bowtie s$ dobiva se ...

.... relacija sa shemom
$R \cup S$ koja sadrži n-torke nastale spajanjem n-torki $t_r \in r$, $t_s \in s$, za koje vrijedi $t_r(A_1) = t_s(A_1) \wedge t_r(A_2) = t_s(A_2) \wedge$ **...** $t_r(A_n) = t_s(A_n)$.

4. a) Navedite karakteristike loše koncipirane sheme baze podataka. Na primjeru prikazane relacije **upisPredmeta** ilustrirajte anomalije koje se pojavljuju pri radu s relacijom definiranom na loše koncipiranom relacijskom shemom. Atributi koji čine primarni ključ relacije su podvučeni. ***(3 boda)***

JMBAG	***sifPred***	**nazivPred**	**akGodina**
0036344899	295	Matematika 1	2005
0036432987	295	Matematika 1	2006
0036344899	320	Baze podataka	2006

Karakteristike loše koncipirane sheme baze podataka:

redundancija

neracionalno korištenje prostora za pohranu

anomalija unosa – ne mogu se unijeti podaci o predmetima koje nije nitko upisao; svaki put kad se unosi podatak o upisanom predmetu studenta mora se ponovno upisati i naziv predmeta

anomalija izmjene – ako predmet promijeni naziv tu je izmjenu potrebno obaviti na više mjesta da vi se zadržala konzistentnost podataka

anomalija brisanja brisanjem podataka o svim upisima određenog predmeta gube se podaci o tom predmetu

pojava lažnih n-torki

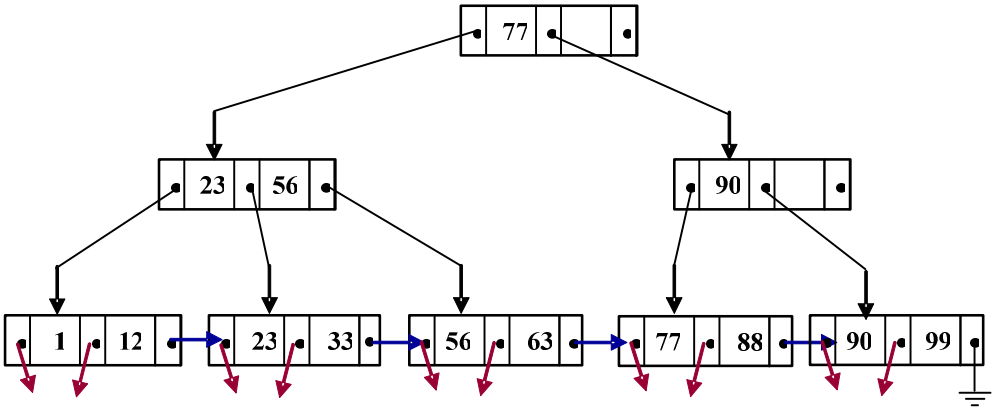
5. Zadana je relacijska shema R = {A, B, C, D, E, G, H, J} i na njoj skup funkcijskih zavisnosti F = { ABC→DEG, A→E, GH→D, AC→DGHJ, G→J }.
Domene atributa sadrže samo jednostavne vrijednosti, vrijednost svakog atributa je samo jedna vrijednost iz domene tog atributa.
Odrediti primarni ključ relacijske sheme (tako da bude zadovoljen uvjet 1NF prema kojem neključni atributi funkcijski ovise o ključu), te shemu postupno normalizirati na 2NF i 3NF. ***(4 boda)***

1NF: R= {A, B, C, D, E, G, H, J } K_R= {A, B, C}

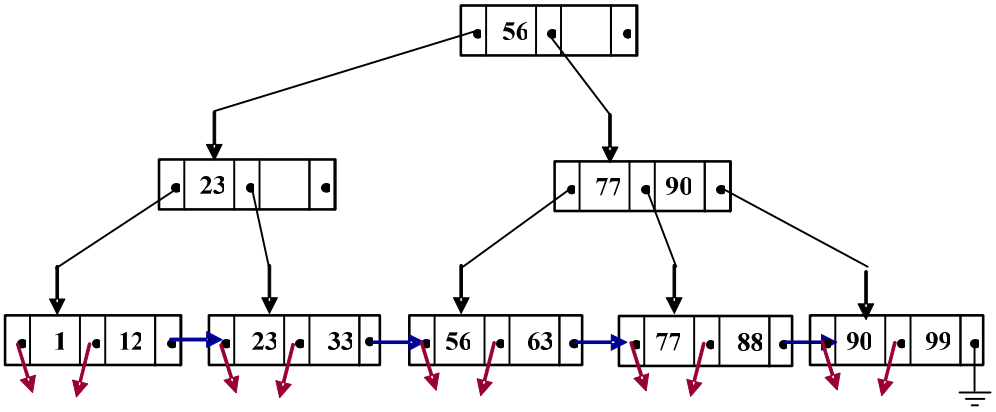
2NF: R1= {A, B, C} K_{R1}= {A, B, C}
R2= {A, E} K_{R2}= {A }
R3= {A, C, D, G, H, J} K_{R3}= {A, C}

3NF: R1= {A, B, C} K_{R1}= {A, B, C}
R2= {A, E} K_{R21}= {A}
R31= {A, C, G, H} K_{R31}= {A, C}
R32= {G, H, D} K_{R32}= {G, H}
R33= {G, J} K_{R33}= {G }

6. Nacrtati B+ stablo reda 3, čiji su čvorovi maksimalno popunjeni. Stablo sadrži kazaljke na zapise čiji su ključevi cijeli brojevi iz skupa { 1, 12, 23, 33, 56, 63, 77, 88, 90, 99}. ***(2 boda)***



Ili



7. Vlasnik baze podataka **knjiznica** je korisnik ***admin***. Korisnicima ***anic***, ***babic*** i ***caric*** već je dodijeljena dozvola za uspostavljanje SQL sjednice. Navedite niz SQL naredbi koje će dovesti do stanja u kojem korisnik ***anic*** može uspješno obaviti naredbu

```
REVOKE DELETE ON primjerak FROM babic;
```

ali ne može uspješno obaviti naredbu

```
REVOKE DELETE ON primjerak FROM babic RESTRICT;
```

(2 boda)

Uz svaku naredbu navedite koji ju korisnik obavlja te pazite na redoslijed naredbi.

admin: `GRANT DELETE ON primjerak TO anic WITH GRANT OPTION;`

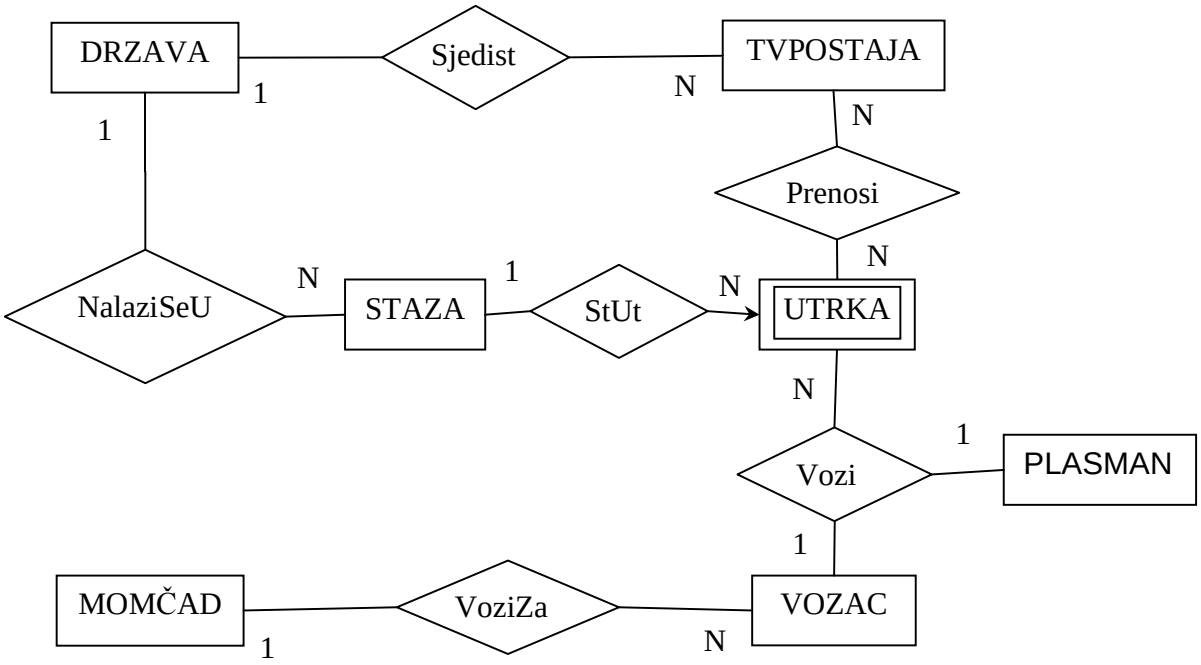
anic: `GRANT DELETE ON primjerak TO babic WITH GRANT OPTION;`

babic: `GRANT DELETE ON primjerak TO caric;`

8. Tijekom jedne sezone vozači se natječu u automobilskim utrkama. Za svakog vozača evidentira se njegov matični broj, te ime i prezime. Za svaku stazu na kojoj se održavaju utrke evidentira se šifra i naziv staze te država u kojoj se staza nalazi. Na jednoj stazi može se održati više utrka, ali najviše jedna utrka u jednom danu. Za svaku utrku se evidentira datum i ukupni nagradni fond utrke. U jednoj državi može biti nekoliko staza. Za svaku državu se bilježi oznaka koja ju jednoznačno određuje, te naziv.

Za svakog se vozača bilježi koji je plasman te koliko je bodova osvojio na kojoj utrci. Prvo mjesto uvijek donosi 10 bodova, drugo mjesto 8 bodova i td. Određeni plasman na utrci može osvojiti samo jedan vozač (ne može se dogoditi da npr. dva vozača dijele 4. mjesto). Svaki vozač je član jedne momčadi. Za jednu momčad može voziti više vozača. Za momčad se bilježi šifra i naziv.

Također se vodi evidencija o tome koje TV postaje prenose koju utrku. Jednu utrku može prenositi nekoliko TV postaja. Za svaku TV postaju bilježi se kratica koja ju jednoznačno određuje, naziv te država u kojoj se nalazi sjedište TV postaje. Nacrtati ER model i opisati entitete i veze. Entitete, osim slabih entiteta, opisati isključivo vlastitim atributima. Sve sheme moraju zadovoljavati 3NF. ***(6 bodova)***



DRZAVA: <u>oznDrzava</u>, NazivDrzava
TVPOSTAJA: <u>kratTV</u>, nazivTV
STAZA: <u>sifStaza</u>, nazivStaza
UTRKA: <u>datum, sifStaza</u>, nagradniFond
PLASMAN: <u>rbrMjesto</u>, bodovi
VOZAC: <u>matBr</u>, ime, prezime
MOMCAD: <u>sifra</u>, nazivMomcad

Sjedist: kratTV, oznDrzava
Prenosi: kratTV, datum, sifStaza
NalziSeU: sifStaza, oznDrzava
StUt: datum, sifStaza
Vozi: datum, sifStaza, matBr, rbrMjesto (alternativni ključ je datum, sifStaza, rbrMjesto)
VoziZa: matBr, sifra