Универзитет у Београду **1.2.2020.**
Електротехнички факултет
Катедра за рачунарску технику и информатику

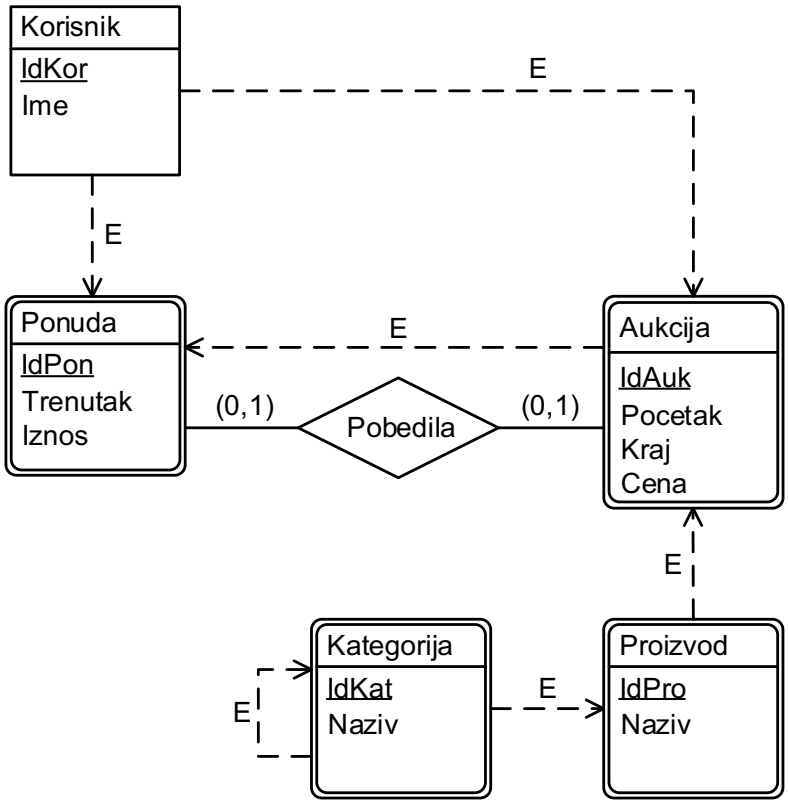
Базе података 1

(13С112БП1, 13Е113БП1)
- фебруарски испитни рок –

Група Б

Посматра се систем за аукције у коме корисници могу да креирају аукције на којима продају производе. Приликом креирања аукције, познати су производ, време почетка аукције и почетна цена, док се време краја аукције дефинише у моменту проглашења краја аукције. Сваки корисник може да даје већи број понуда док аукција траје. Износ сваке понуде мора да буде строго већа од износа претходно дате понуде за исту аукцију. У тренутку када се прогласи крај, уколико је била бар једна понуда за ту аукцију аукција се сматра успешном и проглашава се победник те аукције. У супротном, аукција се сматра неуспешном и победник не постоји. Сваки производ припада некој од категорија у хијерархији категорија које су евидентиране у систему. Сви датуми у систему су из периода од 1941. године до 2040. године.

У наставку је дата релациона шема посматраног дела базе података.



Kategorija (IdKat, Naziv, IdKatNad)

IdKat	- цео број, идентификује категорију, аутоматско додељивање наредног идентификатора
Naziv	- низ до 50 знакова, обавезно
IdKatNad	- страни кључ (табела Kategorija)

Напомена:

Атрибут IdKatNad представља категорију изнад у хијерархији.
Уколико је вредност атрибута IdKatNad једнака NULL, то значи да је то основна категорија.

Proizvod (IdPro, IdAuk, Naziv, IdKat)

IdPro	- цео број, идентификује производ, аутоматско додељивање наредног идентификатора
Naziv	- низ до 50 знакова, обавезно
IdKat	- страни кључ (табела Kategorija), обавезно

Korisnik (IdKor, Ime)

IdKor	- цео број, идентификује корисника, аутоматско додељивање наредног идентификатора
Ime	- низ до 50 знакова, обавезно

Aukcija (IdAuk, Pocetak, Kraj, Cena, IdPro, IdKor)

IdAuk	- цео број, идентификује аукцију, аутоматско додељивање наредног идентификатора
Pocetak	- цео број, обавезно, време у формату (yymmddhhmm)
Kraj	- цео број, време у формату (yymmddhhmm)
Cena	- цео број, обавезно, вредност већа од 0
IdPro	- страни кључ (табела Proizvod), обавезно
IdKor	- страни кључ (табела Korisnik), обавезно

Напомена:

Атрибут IdKor представља корисника који је иницирао аукцију.
Атрибут Cena представља почетну цену производа.
Уколико је вредност атрибута Kraj једнака NULL, то значи да је време краја аукције непознато и да је аукција активна, уколико је вредност атрибута Kraj дефинисан, аукција је завршена.
Сматрати да је вредност атрибута Kraj увек исправна (мања од тренутног времена).

Ponuda (IdPon, Trenutak, Iznos, IdKor, IdAuk)

IdPon	- цео број, идентификује понуду, аутоматско додељивање наредног идентификатора
Trenutak	- цео број, време у формату (yymmddhhmm)
Iznos	- цео број, обавезно, вредност већа од 0
IdKor	- страни кључ (табела Korisnik), обавезно
IdAuk	- страни кључ (табела Aukcija), обавезно

Напомена:

Атрибут IdKor представља корисника који даје понуду.
Гарантује се да ће Trenutak понуде бити исправан и да ће припадати интервалу у коме је аукција активна. Гарантује се да ће износ сваке касније понуде за исту аукцију бити строго већи.

Pobedila (IdAuk, IdPon)

IdAuk	- страни кључ (табела Aukcija), примарни кључ, обавезно
IdPon	- страни кључ (табела Ponuda), обавезно

Задатак 1 [4 поена]

Потребно је направити SQL упит који исписује кориснике које су након свих аукција у којима су учествовали остварили строго више новца него што су имали пре аукција (разлика између укупне цене по којој су производе продавали и укупне цене по којој су производе куповали је позитивна). Активне аукције се не посматрају. Сортирати по IdKor растуће.

Резултат дати у форми: IdKor, Ime
У Cactus-у користити таб: Zadatak 1

```
SELECT IdKor, Ime
FROM Korisnik
WHERE COALESCE((SELECT SUM(Iznos)
        FROM Aukcija JOIN Pobedila USING (IdAuk) JOIN Ponuda USING (IdPon)
        WHERE Aukcija.IdKor=Korisnik.IdKor),0)
>COALESCE((SELECT SUM(Iznos)
        FROM Aukcija JOIN Pobedila USING (IdAuk) JOIN Ponuda USING (IdPon)
        WHERE Ponuda.IdKor=Korisnik.IdKor),0)
ORDER BY IdKor
```

Задатак 2 [4 поена]

Потребно је направити SQL скрипту која ако постоји табела ***Aukcija*** избацује табелу ***Aukcija*** из шеме, а затим формира нову табелу ***Aukcija*** која треба да има одговарајућу структуру и ограничења. Није потребно да се реализује ограничење да је вредност поља Pocetak увек мања од вредности поља Kraj, као ни ограничење да је датум у исправном формату.

У Cactus-у користити таб: Zadatak 2

```
DROP TABLE IF EXISTS Aukcija;

CREATE TABLE Aukcija
(
        IdAuk INTEGER PRIMARY KEY,
        Pocetak INTEGER NOT NULL,
        Kraj INTEGER,
        Cena INTEGER NOT NULL CHECK (Cena>0),
        IdKor INTEGER NOT NULL REFERENCES Korisnik(IdKor),
        IdPro INTEGER NOT NULL REFERENCES Proizvod(IdPro)
);
```

Задатак 3 [4 поена]

Потребно је направити SQL упит који исписује статистику везану за аукције. Потребно је приказати колико има аукција излицитираних за цену мању или једнаку од дупло, затим за цену већу дупло и мању или једнаку тродупло, па за цену већу од тродупло и мању или једнаку четвородупло и за цену већу од четвородупло у односу на почетну. Потребно је исписати и статистике за групе у којима нема ни једне аукције. Резултат треба да буде формиран у четири реда – групе. У првом реду треба да буде група "<=200%", у другом реду ">200% i <=300%", у трећем реду ">300% i <=400%", а у четвртом реду ">400%".

Резултат дати у форми: Naziv grupe, Broj aukcija
У Cactus-у користити таб: Zadatak 3

```
SELECT '<=200%' AS "Naziv grupe", COUNT(*) AS "Broj aukcija"
FROM Aukcija JOIN Pobedila USING (IdAuk) JOIN Ponuda USING (IdPon)
WHERE Iznos<=2*Cena
UNION ALL
SELECT '>200% i <=300%', COUNT(*)
FROM Aukcija JOIN Pobedila USING (IdAuk) JOIN Ponuda USING (IdPon)
WHERE Iznos>2*Cena AND Iznos<=3*Cena
UNION ALL
SELECT '>300% i <=400%', COUNT(*)
FROM Aukcija JOIN Pobedila USING (IdAuk) JOIN Ponuda USING (IdPon)
WHERE Iznos>3*Cena AND Iznos<=4*Cena
UNION ALL
SELECT '>400%', COUNT(*)
FROM Aukcija JOIN Pobedila USING (IdAuk) JOIN Ponuda USING (IdPon)
WHERE Iznos>4*Cena
```

Задатак 4 [4 поена]

Потребно је написати SQL упит који исписује све аукције које су успешне, а које су почеле и завршиле се истог дана и које су трајале мање од 6 сати. Потребно је исписати и ко је победник аукције и по којој цени је производ купљен. Атрибут Pobednik представља IdKor корисника који је победио на аукцији. Сортирати по IdAuk опадајуће.

Резултат дати у форми: IdAuk, Pocetak, Kraj, Pocetna cena, Krajnja cena, Pobednik
У Cactus-у користити таб: Zadatak 4

```
SELECT Aukcija.IdAuk, Pocetak, Kraj, Cena AS "Pocetna cena",
Iznos AS "Krajnja cena", Ponuda.IdKor AS Pobednik
FROM Aukcija JOIN Pobedila USING (IdAuk) JOIN Ponuda USING (IdPon)
WHERE kraj IS NOT NULL AND pocetak/10000=kraj/10000
AND kraj%10000-pocetak%10000<600
ORDER BY Aukcija.IdAuk DESC
```

Задатак 5 [5 поена]

Потребно је направити SQL упит који ажурира стања одговарајућих табела на начин да проглашава све активне аукције завршеним за које постоји бар један корисник који је дао понуду за ту аукцију. И том приликом корисника са највишом понудом проглашава за победника те аукције. Сматрати да је тренутно време 2002012005. Након тога приказати информације о свим аукцијама и њиховим победницима. Потребно је исписати и аукције које и даље немају победника. Атрибут Period се исписује у формату *Pocetak - Kraj*. Атрибут Pobednik представља IdKor корисника који је победио на аукцији. Резултат сортирати по IdAuk растуће.

Резултат дати у форми: IdAuk, Period, Pobednik
У Cactus-у користити таб: Zadatak 5
Није дозвољено коришћење погледа.

```
INSERT INTO Pobedila (IdAuk, IdPon)
SELECT IdAuk, IdPon FROM Ponuda JOIN Aukcija USING(IdAuk)
WHERE Kraj IS NULL AND
Iznos=(SELECT MAX (Iznos) FROM Ponuda WHERE IdAuk=Aukcija.IdAuk);

UPDATE Aukcija SET Kraj=2002012005
WHERE Kraj IS NULL AND EXISTS ( SELECT * FROM Ponuda WHERE IdAuk=Aukcija.IdAuk );

SELECT Aukcija.IdAuk, Pocetak||' - '||Kraj AS  Period, Ponuda.IdKor AS Pobednik
FROM Aukcija LEFT JOIN Pobedila USING (IdAuk) LEFT JOIN Ponuda USING (IdPon)
ORDER BY Aukcija.IdAuk
```

Задатак 6 [5 поена]

Потребно је направити SQL упит који исписује све парове корисника код којих је била узајамна купопродаја на аукцијама (први корисник је купио неки производ који је други корисник ставио на аукцију и други корисник је купио неки производ који је први корисник ставио на аукцију). Потребно је исписати и укупан број аукција у којима су један од другог нешто откупили. Резултат треба сортирати растуће по IdKor1, а затим растуће по IdKor2. IdKor1 је потребно да има мању вредност од IdKor2.

Резултат дати у форми: IdKor1, Ime1, IdKor2, Ime2, BrojAukcija
У Cactus-у користити таб: Zadatak 6

```
WITH BrojProdaja( IdProdavac, IdKupac, Broj) AS (
        SELECT Aukcija.Idkor, Ponuda.IdKor, COUNT(*)
        FROM Aukcija JOIN Pobedila USING (IdAuk) JOIN Ponuda USING (IdPon)
        GROUP BY Aukcija.Idkor, Ponuda.IdKor
)
SELECT K1.IdKor AS IdKor1, K1.Ime AS Ime1, K2.IdKor AS IdKor2, K2.Ime AS Ime2,
        BP1.Broj + BP2.Broj AS BrojAukcija
FROM Korisnik K1, Korisnik K2, BrojProdaja BP1, BrojProdaja BP2
WHERE  K1.IdKor<K2.IdKor AND  BP1.IdProdavac=K1.IdKor AND BP1.IdKupac=K2.IdKor
AND BP2.IdProdavac=K2.IdKor AND BP2.IdKupac=K1.IdKor
ORDER BY K1.IdKor, K2.IdKor
```

Задатак 7 [5 поена]

Потребно је направити SQL упит који дохвата све кориснике којима се исплатила продаја претходно купљеног производа. Продаја претходно купљеног производа се исплати уколико му износ по којој је производ продат у процентима није пала за више од броја година колико је производ био код њега. Број година се посматра као цео број и заокружује се на више. Проценат за коју је вредност пала се рачуна као однос разлике износа и претходног износа. Производ може да буде максимално једном купљен и максимално једном продат од стране једног истог корисника. Резултат сортирати по IdKor растуће, а затим растуће по IdPro.

Резултат дати у форми: IdKor, Ime, IdPro, Naziv
У Cactus-у користити таб: Zadatak 7
Није дозвољено коришћење погледа.

```
WITH Ishod (IdKor1, IdKor2, Datum, IdPro, Iznos) AS (
        SELECT Aukcija.IdKor, Ponuda.IdKor,
        (CASE WHEN Kraj<4100000000 THEN Kraj+10000000000 ELSE Kraj END),
        IdPro, Iznos
        FROM Aukcija JOIN Pobedila USING (IdAuk) JOIN Ponuda USING (IdPon)
)
SELECT I1.IdKor2 AS IdKor, K.Ime, I1.idPro, P.Naziv
FROM Ishod I1, Ishod I2, Korisnik K, Proizvod P
WHERE I1.IdPro=I2.idPro AND I1.idPro=P.idPro
AND I1.IdKor2=I2.IdKor1 AND I1.IdKor2=K.IdKor
AND I1.Datum<I2.Datum
AND (I2.Datum-I1.Datum+99999999)/100000000 >(I1.Iznos-I2.Iznos)*100.0/I1.Iznos
ORDER BY K.IdKor, P.IdPro
```

Задатак 8 [6 поена]

Потребно је направити SQL упит који исписује колико има производа у свакој од основних категорија, као и колико њих је тренутно на аукцији, колико њих је било на аукцији, просечан износ производа по којој су последњи пут продавани (производи који су тренутно на аукцију и даље немају коначну цену па њих није потребно рачунати, већ цену по којој су продати на претходној аукцији уколико је она постојала). За категорије које немају ни један производ, потребно је исписати -2 за просечну цену. За категорије које немају ни један производ који је купљен на аукцији, потребно је исписати -1 за просечну цену. Категорија је основна уколико нема надкатегорију. Сортирати по Bilo na aukciji опадајуће, а затим по Trenutno na aukciji растуће и на крају по IdKat опадајуће.

Резултат дати у форми: IdKat, Naziv, Broj proizvoda, Bilo na aukciji, Trenutno na aukciji, Prosecan iznos
У Cactus-у користити таб: Zadatak 8
Није дозвољено коришћење погледа.

```
WITH RECURSIVE OsnovnaKategorija(IdKat, IdOsn, Naziv) AS (
	SELECT IdKat, IdKat, Naziv FROM Kategorija WHERE IdKatnad IS NULL
	UNION ALL
	SELECT K.IdKat, IdOsn, OK.Naziv
	FROM OsnovnaKategorija OK JOIN Kategorija K ON (OK.IdKat=K.IdKatNad)
),
BrojAukcija( IdPro, BrPrZav, BrPrAKt) AS (
	SELECT  IdPro, COALESCE(COUNT(Kraj),0),
		COALESCE(COUNT(IdAuk),0)-COALESCE(COUNT(Kraj),0)
	FROM Aukcija
	GROUP BY IdPro
),
CenaPoslednje( IdPro, PoslednjiIznos) AS (
	SELECT  IdPro, Iznos
	FROM Aukcija A1 JOIN Pobedila USING (IdAuk) JOIN Ponuda USING (IdPon)
	WHERE (CASE WHEN Kraj<4100000000 THEN Kraj+10000000000 ELSE Kraj END)=(
		SELECT MAX (CASE WHEN Kraj<4100000000 THEN Kraj+10000000000 ELSE Kraj
END)
		FROM Aukcija A2 WHERE A2.IdPro=A1.IdPro)
)
SELECT IdOsn AS IdKat, OK.Naziv,
COUNT(P.IdPro) AS "Broj proizvoda",
COALESCE(SUM(BrPrZav),0) AS "Bilo na aukciji",
COALESCE(SUM(BrPrAKt),0) AS "Trenutno na aukciji",
CASE WHEN COALESCE(COUNT(P.IdPro),0)=0 THEN -2
	WHEN COALESCE(SUM(BrPrZav),0)=0 THEN -1
	ELSE AVG(PoslednjiIznos) END AS "Prosecan iznos"
FROM OsnovnaKategorija OK LEFT JOIN Proizvod P USING (idKat)
	LEFT JOIN BrojAukcija BA USING (IdPro) LEFT JOIN CenaPoslednje USING (IdPro)
GROUP BY IdOsn
ORDER BY 4 DESC, 5, 1 DESC
```

Задатак 9 [6 поена]

Потребно је направити SQL упит за испис три најбоље рангирана корисника на листи тактичара. Тактизирање се мери на следећи начин: Израчуна се колико је у просеку за сваку завршену аукцију на којој је корисник победио дао више пара у односу на износ претходно предложене понуде за ту аукцију. Ако је корисник био једини понуђач аукције (па самим тим и победник), онда се гледа колико је дао више пара у односу на почетну цену на аукцији. Што је добијена вредност мања, то је корисник боље рангиран на листи тактичара. Ранг представља редни број на листи и потребно је да буде написан речима ("prvi", "drugi", "treci"). Обратити пажњу да две особе могу имати исти ранг (деле исти ранг). Резултат треба сортирати по Rang растуће (по редном броју), па по идентификатору корисника растуће.

Резултат дати у форми: Rang, Ime, Poeni
У Cactus-у користити таб: Zadatak 9
Није дозвољено коришћење погледа.

```
WITH Taktika(IdKor, Poeni) AS (
        SELECT P1.IdKor, AVG(Iznos-COALESCE(
                (SELECT MAX(Iznos)
                FROM Ponuda P2
                WHERE P2.IdAuk=A.IdAuk AND P1.IdPon!=P2.IdPon),
                Cena))
        FROM Aukcija A JOIN Pobedila USING (IdAuk) JOIN Ponuda P1 USING (IdPon)
        GROUP BY P1.IdKor
),
RBKorisnika(IdKor, Poeni, RB) AS (
SELECT IdKor, Poeni, 1+
COALESCE((SELECT COUNT(*) FROM Taktika T2 WHERE T1.Poeni >T2.Poeni) ,0)
FROM Taktika T1
)
SELECT CASE RB
        WHEN 1 THEN 'prvi'
        WHEN 2 THEN 'drugi'
        WHEN 3 THEN 'treci'
        END AS Rang, Ime, Poeni
FROM RBKorisnika JOIN Korisnik USING (IdKor)
WHERE RB<=3
ORDER BY RB
```

Задатак 10 [7 поена]

Потребно је направити SQL упит који дохвата све категорије за које важи следеће:

- Та категорија има бар једну поткатегорију
- Сви производи који спадају у ту категорију налазе у некој од поткатегорија које нису даље дељиве на поткатегорије
- Не постоји поткатегорија те категорије, која није даље дељива на поткатегорије, таква да се у њој не налази ни један производ.

Сортирати по IdKat растуће.

Резултат дати у форми: IdKat, Naziv
У Cactus-у користити таб: Zadatak 10
Није дозвољено коришћење погледа.

```
WITH RECURSIVE Stat(IdKat, IdPoc, Broj, List) AS (
        SELECT IdKat, IdKat,
        COALESCE((SELECT COUNT(*) FROM Proizvod WHERE IdKat=K.IdKat),0),
        CASE WHEN
                EXISTS (SELECT * FROM Kategorija K2 WHERE K2.IdKatNad=K.IdKat) THEN 0
        ELSE 1 END
        FROM Kategorija K
        UNION ALL
        SELECT K.IdKatNad, S.IdPoc, Broj, List
        FROM Stat S JOIN Kategorija K USING (IdKat)
        WHERE K.IdKatNad IS NOT NULL
),
UslovZadovoljava AS (
        SELECT K.IdKat, K.IdKatNad, Naziv
        FROM Kategorija K JOIN Stat S USING (IdKat)
        GROUP BY K.IdKat
        HAVING SUM(Broj)=SUM(CASE WHEN List=1 THEN Broj ELSE 0 END)
        AND SUM(Broj) >0 AND K.IdKat NOT IN
                ( SELECT IdKat FROM Stat WHERE List=1 AND Broj=0 )
)
SELECT IdKat, Naziv
FROM UslovZadovoljava
WHERE EXISTS (SELECT IdKat FROM Kategorija WHERE IdKatnad=UslovZadovoljava.IdKat)
ORDER BY IDKat
```
