Универзитет у Београду **02.02.2019.**
Електротехнички факултет
Катедра за рачунарску технику и информатику

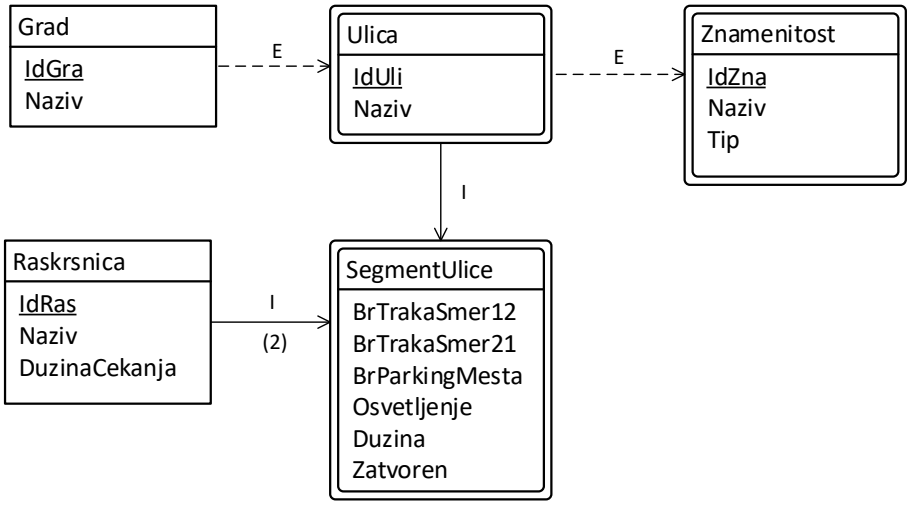
Базе података 1

- фебруарски испитни рок -

Група Б

Посматра се део базе података саобраћајне инфраструктуре. У овој бази се чувају подаци о градовима, улицама, сегментима улица, раскрсницама и знаменитостима. Свака улица се састоји из барем једног сегмента и почиње и завршава се раскрсницом. На раскрсницама могу да постоје семафори на којима се чека одређено време.

У наставку је дата релациона шема посматраног дела базе података.



Grad (IdGra, Naziv)

IdGra	- цео број, идентификује град, аутоматско додељивање наредног идентификатора
Naziv	- низ до 50 знакова, обавезно

Ulica (IdUli, Naziv, IdGra)

IdUli	- цео број, идентификује улицу, аутоматско додељивање наредног идентификатора
Naziv	- низ до 50 знакова, обавезно
IdGra	- страни кључ (табела Grad), обавезно

Raskrsnica (IdRas, Naziv, DuzinaCekanja)

IdRas	- цео број, идентификује раскрсницу, аутоматско додељивање наредног идентификатора
Naziv	- низ до 50 знакова
DuzinaCekanja	- цео број, вредност већа од 0

Напомена:

Када на раскрсници постоји семафор, у ситуацији да је на семафору упаљено црвено светло чека се DuzinaCekanja времена, у супротном одмах се пролази (нема чекања). У случају да на раскрсници не постоји семафор, DuzinaCekanja има вредност NULL.
Атрибут DuzinaCekanja је целобројног типа и дат је у минутима.

SegmentUlice (IdUli, IdRas1, IdRas2, BrTrakaSmer12, BrTrakaSmer21, BrParkingMesta, Osvetljenje, Duzina, Zatvoren)

IdUli	- страни кључ (табела Ulica), обавезно
IdRas1	- страни кључ (табела Raskrsnica), обавезно
IdRas2	- страни кључ (табела Raskrsnica), обавезно
BrTrakaSmer12	- цео број, обавезно поље, вредност већа или једнака од 0
BrTrakaSmer21	- цео број, обавезно поље, вредност већа или једнака од 0
BrParkingMesta	- цео број, вредност већа или једнака од 0
Osvetljenje	- цео број, могуће вредности у опсегу од 0 до 1
Duzina	- цео број, обавезно поље, вредност већа од 0
Zatvoren	- цео број, обавезно поље, могуће вредности у опсегу од 0 до 1

Напомене:

Значење атрибута BrTrakaSmer12: број трака у смеру IdRas1→IdRas2.
Значење атрибута BrTrakaSmer21: број трака у смеру IdRas2→IdRas1.
Претпоставити да је вредност поља IdRas1 увек мања од вредности поља IdRas2.
Значење атрибута Osvetljenje: NULL – непознато, 0 –нема осветљења, 1 – има осветљења.
Атрибут Duzina целобројног типа дат је у јединици метар.
Значење атрибута Zatvoren: 0 – отворен, 1 – затворен.
Комбинација вредности у пољима IdUli, IdRas1 и IdRas2 је увек јединствен податак на нивоу табеле SegmentUlice.

Znamenitost (IdZna, Naziv, Tip, IdUli)

IdZna	- цео број, идентификује место, аутоматско додељивање наредног идентификатора
Naziv	- низ до 50 знакова, обавезно, јединствено
Tip	- низ до 30 знакова, обавезно
IdUli	- страни кључ (табела Ulica), обавезно

Задатак 1 [3 поена]

Потребно је направити SQL упит који исписује све улице у Новом Саду које имају бар неко паркинг место. Резултат треба сортирати опадајуће по називу улице.

Резултат дати у форми: IdUlice, Naziv ulice
У Cactus-у користити таб: Zadatak 1

```
SELECT DISTINCT IdUli AS IdUlice, Ulica.Naziv AS "Naziv ulice"
FROM Ulica JOIN SegmentUlice USING(IdUli) JOIN Grad USING(IdGra)
WHERE Grad.Naziv = 'Novi Sad' AND SegmentUlice.BrParkingMesta > 1
ORDER BY Ulica.Naziv DESC
```

Задатак 2 [3 поена]

Потребно је направити SQL упит који исписује улице и број улица са којима се оне укрштају, за све улице које се укрштају са барем две улице . Резултат треба сортирати по IdUli опадајуће.

Резултат дати у форми: IdUli, Naziv, Broj Ukrstanja
У Cactus-у користити таб: Zadatak 2

```
SELECT U.IdUli, U.Naziv, COUNT (DISTINCT S2.IdUli) AS "Broj Ukrstanja"
FROM Ulica U, SegmentUlice S1, SegmentUlice S2
WHERE U.IdUli = S1.IdUli AND (S1.IdRas1 = S2.IdRas1 OR S1.IdRas1 = S2.IdRas2
        OR S1.IdRas2 = S2.IdRas1 OR S1.IdRas2 = S2.IdRas2) AND S2.IdUli != S1.IdUli
GROUP BY U.IdUli, U.Naziv
HAVING COUNT (DISTINCT S2.IdUli) >= 2
ORDER BY U.IdUli DESC
```

Задатак 3 [3 поена]

Потребно је направити SQL упит који исписује град у коме се налази најкраћа улица. У случају да постоји више таквих улица исте (најкраће) дужине, исписати све градове. Резултат треба сортирати по IdGra у опадајућем редоследу.

Резултат дати у форми: IdGra, Naziv
У Cactus-у користити таб: Zadatak 3

```
WITH DuzinaUlice (IdUli, IdGra, Duzina) AS (
        SELECT IdUli, IdGra, SUM (Duzina)
        FROM SegmentUlice NATURAL JOIN Ulica
        GROUP BY IdUli, IdGra
)
SELECT DISTINCT Grad.IdGra, Grad.Naziv
FROM Grad JOIN DuzinaUlice USING (IdGra)
WHERE Duzina = (SELECT MIN (Duzina) FROM DuzinaUlice)
ORDER BY IdGra DESC
```

Задатак 4 [3 поена]

Потребно је направити SQL упит који исписује све раскрснице које у називу имају мање од 5 речи и садрже реч трг на самом почетку назива, укупан број паркинг места које се налазе у околини (паркинг места припадају сегменту те раскрснице). Сматрати да у називу не постоје спојена два бланко знака. Резултат треба сортирати растуће по IdRas.

Резултат дати у форми: IdRas, Naziv, Ukupno parking mesta
У Cactus-у користити таб: Zadatak 4

```
SELECT IdRas, Naziv, (
        SELECT SUM (BrParkingMesta) FROM SegmentUlice
        WHERE IdRas1 = IdRas OR IdRas2 = IdRas) AS "Ukupno parking mesta"
FROM Raskrsnica
WHERE Naziv LIKE "Trg %" AND Naziv NOT LIKE "_% _% _% _% _% _%"
ORDER BY IdRas
```

Задатак 5 [4 поена]

Потребно је направити SQL скрипту која ако постоји табела ***SegmentUlice*** брише табелу ***SegmentUlice*** из шеме, а затим формира нову табелу ***SegmentUlice*** која треба да има одговарајућу структуру и ограничења. Није потребно да се реализује ограничење да је вредност поља IdRas1 увек мања од вредности поља IdRas2.

У Cactus-у користити таб Zadatak 5.

```
DROP TABLE IF EXISTS SegmentUlice;

CREATE TABLE SegmentUlice (
        IdUli INTEGER NOT NULL,
        IdRas1 INTEGER NOT NULL,
        IdRas2 INTEGER NOT NULL,
        BrTrakaSmer12 INTEGER NOT NULL CHECK (BrTrakaSmer12 >= 0),
        BrTrakaSmer21 INTEGER NOT NULL CHECK (BrTrakaSmer21 >= 0),
        BrParkingMestaa INTEGER CHECK (BrParkingMestaa >= 0),
        Osvetljenje INTEGER CHECK (Osvetljenje BETWEEN 0 AND 1),
        Duzina INTEGER NOT NULL CHECK (Duzina > 0),
        Zatvoren INTEGER NOT NULL CHECK (Zatvoren BETWEEN 0 AND 1),
        PRIMARY KEY (IdUli, IdRas1, IdRas2),
        FOREIGN KEY (IdUli) REFERENCES Ulica (IdUli)
        FOREIGN KEY (IdRas1) REFERENCES Raskrsnica (IdRas)
        FOREIGN KEY (IdRas2) REFERENCES Raskrsnica (IdRas)
);
```

Задатак 6 [4 поена]

Потребно је направити SQL упит који исписује све улице у којима се налазе сви типови знаменитости који постоје у том граду у коме постоји барем једна знаменитост, као и улице у градовима у којима не постоји ни једна знаменитост. Резултат треба сортирати по IdUli у растућем поретку.

Резултат дати у форми: IdUli, Naziv
У Cactus-у користити таб: Zadatak 6
Није дозвољено коришћење погледа.

```
SELECT DISTINCT IdUli, Naziv
FROM Ulica U
WHERE (SELECT COUNT (DISTINCT Tip) FROM Znamenitost Z WHERE IdUli=U.IdUli) = (
        SELECT COUNT (DISTINCT Tip) FROM Ulica JOIN Znamenitost USING (IdUli)
                WHERE IdGra=U.IdGra)
ORDER BY IdUli
```

Задатак 7 [4 поена]

Потребно је направити SQL упит који исписује све градове у којима се сви спортски објекти у том граду налазе у најдужој улици у том граду (*sportski objekat* је вредност типа знаменитости у табели . Градове који немају спортске објекте није потребно исписати. Резултат треба сортирати по IdGra опарајућем.

Резултат дати у форми: IdGra, Naziv

У Cactus-у користити таб: Zadatak 7
Није дозвољено коришћење погледа.

```
WITH DuzinaUlice (IdUli, IdGra, Duzina) AS (
        SELECT IdUli, IdGra, SUM (Duzina)
        FROM SegmentUlice NATURAL JOIN Ulica
        GROUP BY IdUli, IdGra
),
BrSportskihObj (IdUli, IdGra, BrSportskihObj) AS(
        SELECT IdUli, IdGra, COUNT (*)
        FROM Ulica JOIN Znamenitost USING(IdUli)
        WHERE Tip = 'sportski objekat'
        GROUP BY IdUli
)
SELECT DISTINCT Grad.IdGra, Grad.Naziv
FROM Grad NATURAL JOIN DuzinaUlice NATURAL JOIN BrSportskihObj
WHERE Duzina = (SELECT MAX (Duzina) FROM DuzinaUlice WHERE IdGra = Grad.IdGra) AND
BrSportskihObj = (SELECT SUM (BrSportskihObj) FROM BrSportskihObj WHERE IdGra = Grad.IdGra)
ORDER BY IdGra
```

Задатак 8 [4 поена]

Потребно је направити SQL скрипту којим се улица са називом *Vojvode Stepe* затвара због реконструкције, као и сви сегменти улица које се граниче са том улицом.

Након ажурираних података, исписати све податке из табеле ***SegmentUlice.***
У Cactus-у користити таб: Zadatak 8
Није дозвољено коришћење погледа.

```
UPDATE SegmentUlice
SET Zatvoren = 1
WHERE IdRas1 IN (
                        SELECT IdRas1
                        FROM SegmentUlice su NATURAL JOIN Ulica
                        WHERE Naziv = 'Vojvode Stepe'
                        UNION
                        SELECT IdRas2
                        FROM SegmentUlice su NATURAL JOIN Ulica
                        WHERE Naziv = 'Vojvode Stepe'
                )
OR IdRas2 IN (
                        SELECT IdRas1
                        FROM SegmentUlice su NATURAL JOIN Ulica
                        WHERE Naziv = 'Vojvode Stepe'
                        UNION
                        SELECT IdRas2
                        FROM SegmentUlice su NATURAL JOIN Ulica
                        WHERE Naziv = 'Vojvode Stepe'
                );

SELECT * FROM SegmentUlice;
```

Задатак 9 [5 поена]

Потребно је направити SQL упит који исписује улице у граду које нису кружне. Улица је кружна ако је њен почетак уједно и њен крај. У случају да не постоји таква улица, резултат треба да садржи један податак у коме пише "*Nema odgovarajucih ulica*". Резултат треба сортирати по IdUli.

Резултат дати у форми: IdUli
У Cactus-у користити таб: Zadatak 9
Није дозвољено коришћење погледа.

```
WITH NeKruzneUlice(IdUli) AS (
        SELECT S.IdUli
        FROM SegmentUlice S, Raskrsnica R
        WHERE (S.IdRas1=R.IdRas OR S.IdRas2=R.IdRas )
        GROUP BY IdUli
        HAVING COUNT (*) != 2 * COUNT (DISTINCT IdRas)
)
SELECT * FROM NeKruzneUlice
UNION
SELECT 'Nema odgovarajucih ulica'
WHERE (SELECT COUNT(*) FROM NeKruzneUlice) = 0
ORDER BY IdUli DESC
```

Задатак 10 [5 поена]
Потребно је направити SQL скрипту која проналази све улице које се међусобно секу бар два пута. Резултат треба сортирати растуће по IdUli1, а затим опадајуће по IdUli2. IdUli1 је потребно да има већу вредност од IdUli2.

Резултат дати у форми: IdUli1, Naziv1, IdUli2, Naziv2
У Cactus-у користити таб: Zadatak 10
Није дозвољено коришћење погледа

```
WITH RaskrsniceUlice (IdUli,Naziv,IdRas) AS
(
        SELECT IdUli, Naziv, IdRas1
        FROM SegmentUlice NATURAL JOIN Ulica
        UNION
        SELECT IdUli, Naziv, IdRas2
        FROM SegmentUlice  NATURAL JOIN Ulica
)
SELECT RU1.IdUli AS IdUli1, RU1.Naziv AS Naziv1, RU2.IdUli AS IdUli2, RU2.Naziv AS Naziv2
FROM RaskrsniceUlice RU1, RaskrsniceUlice RU2
WHERE RU1.IdRas = ru2.IdRas AND RU1.IdUli > RU2.IdUli
GROUP BY RU1.IdUli, RU1.Naziv, RU2.IdUli, RU2.Naziv
HAVING COUNT (*) >= 2
ORDER BY IdUli1,IdUli2
```

Задатак 11 [6 поена]

Потребно је направити SQL упит који исписује 8 најкраћих улица у Београду. RB представља редни број улице и потребно је да буде написан речима (prva, druga, treca, cetvrta, peta, sesta, sedma, osma), где је *prva* најкраћа улица, *druga* следећа по дужини итд. Обратити пажњу да две улице могу да деле исти RB ако су исте дужине. Резултат треба сортирати по RB растуће.

Резултат дати у форми: RB, IdUli, Naziv
У Cactus-у користити таб: Zadatak 11

```
WITH DuzinaUlice (IdUli, Naziv, Duzina) AS (
	SELECT IdUli, Ulica.Naziv, SUM (Duzina)
	FROM SegmentUlice NATURAL JOIN Ulica JOIN  Grad USING (IdGra)
	WHERE Grad.Naziv = 'Beograd'
	GROUP BY Ulica.IdUli
),
RBUlice (IdUli, Naziv, RBr) AS (
	SELECT IdUli, Naziv, (
		SELECT COUNT (*) + 1
		FROM DuzinaUlice D2
		WHERE D2.Duzina < D1.Duzina) AS RB
	FROM DuzinaUlice D1
)
SELECT CASE RBr
		WHEN 1 THEN 'prva'
		WHEN 2 THEN 'druga'
		WHEN 3 THEN 'treca'
		WHEN 4 THEN 'cetvrta'
		WHEN 5 THEN 'peta'
		WHEN 6 THEN 'sesta'
		WHEN 7 THEN 'sedma'
	END AS RB, IdUli, Naziv
FROM RBUlice D1
WHERE RBr <= 7
ORDER BY RBr
```

Задатак 12 [6 поена]

Колегиница Мина Шекуларац жури да стигне да пожели срећу свом друштву пред испит који почиње за 100 минута. Потребно је направити SQL упит који проналази колико најдуже времена може да остане са друштвом док не почне испит. Мина полази са раскрснице под називом *Minnie raskrsnica,* и треба да дође до раскрснице под називом *ETF raskrsnica*. Њена брзина кретања је увек иста и износи . Марфијев закон каже да ће светло на првом семафору на који наиђе бити црвено и да ће бити црвено светло на сваком другом семафору. Потребно је водити рачуна да је кретање могуће само по сегментима улица које су отворене и у коректном су смеру. Обратити пажњу да семафор не мора да се налази на свакој раскрсници, као и на целобројно/реално дељење. У случају да Мина не стиже на време, резултат треба да садржи један податак у коме пише "*Ne stize*".
На почетној раскрсници (*Minnie raskrsnica*) се не чека семафор, ако га има, док на последњој раскрсници (*ETF raskrsnica*) се чека семафор, ако га има.

Резултат дати у форми: Vreme do ispita
У Cactus-у користити таб: Zadatak 12

```
WITH RECURSIVE Putanja (IdRas, Vreme, Status) AS (
	SELECT MAX (IdRas), 100, 1 FROM Raskrsnica WHERE Naziv = 'Minnie raskrsnica'
	UNION
	SELECT R.IdRas, P.Vreme - S.Duzina / 10.0- CASE P.Status
			WHEN 1 THEN COALESCE (R.DuzinaCekanja, 0)
			ELSE 0
		END, CASE
			WHEN DuzinaCekanja IS NULL THEN Status
			ELSE (Status + 1) % 2
		END
	FROM Putanja P, SegmentUlice S, Raskrsnica R
	WHERE ((P.IdRas = S.IdRas1 AND R.IdRas = S.IdRas2 AND S.BrTrakaSmer12 > 0)
			OR (P.IdRas = S.IdRas2 AND R.IdRas = S.IdRas1 AND S.BrTrakaSmer21 > 0))
		AND S.Zatvoren = 0 AND P.Vreme - S.Duzina / 10.0 >=0
)
SELECT COALESCE (MAX (Vreme), 'Ne stize') AS "Vreme do ispita"
FROM Putanja
WHERE IdRas = (SELECT MAX (IdRas) FROM Raskrsnica WHERE Naziv = 'ETF raskrsnica')
```
