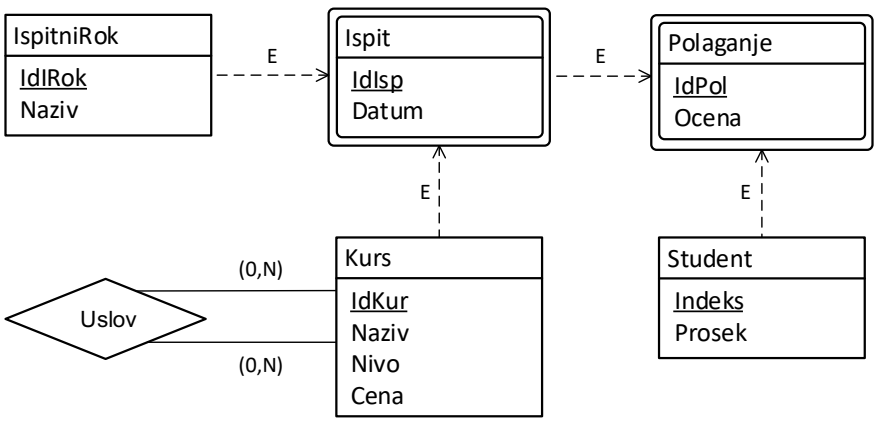
Универзитет у Београду **27.06.2019.**
Електротехнички факултет
Катедра за рачунарску технику и информатику

Базе података 1

– јулски испитни рок –

Посматра се део базе података једног факултета. У овој бази прате се студенти, курсеви, испити, испитни рокови, полагања и оцене студената. Студент поседује свој индекс који га идентификује, као и просечну оцену. Сваки курс има свој назив, ниво, као и цену коју студенти плаћају приликом изласка на испит како би имали могућност да добију оцену и да га положе. Курсеве студенти полажу у оквиру испитног рока, у оквиру којег за сваки курс постоји један испит. Дакле један курс могу полагати у више испитних рокова без обзира да ли га претходно нису положили или јесу (када желе бољу оцену). За сваки испит бележи се датум одржавања. Студент не може полагати курс уколико није претходно положио курсеве који су његов предуслов.

У наставку је дата релациона шема посматраног дела базе података.



Ispit (IdIsp, Datum, IdIRok, IdKur)

IdIsp	- цео број, идентификује испит
Datum	- цео број, обавезно, формат (ggggmmdd)
IdIRok	- страни кључ (табела IspitniRok), обавезно
IdKur	- страни кључ (табела Kurs), обавезно

IspitniRok (IdIRok, Naziv)

IdIRok	- цео број, идентификује испитни рок
Naziv	- низ од 8 знакова, обавезно, формат (mes gggg); mes је наведено малим словима (могуће вредности: jan, feb, mar, apr, maj, jun, jul, avg, sep, okt, nov, dec)

Kurs (IdKur, Naziv, Nivo, Cena)

IdKur	- цео број, идентификује курс
Naziv	- низ од 50 знакова, обавезно
Nivo	- низ од 50 знакова, обавезно
Cena	- цео број, обавезно, вредност већа од 0

Напомена: Број нивоа је променљив и није фиксан осим уколико у задатку није написано другачије.

Polaganje (IdPol, Ocena, Indeks, IdIsp)

IdPol	- цео број, идентификује полагање, аутоматско додељивање наредног идентификатора
Ocena	- цео број, обавезно, вредности у опсегу [5, 10]
Indeks	- страни кључ (табела Student), обавезно
IdIsp	- страни кључ (табела Ispit), обавезно

Напомена: Оцена 5 означава да студент није положио испит. Студент приликом сваког полагања плаћа цену курса.

Student (Indeks, Prosek)

Indeks	- низ од 9 знакова, идентификује студента, формат (gggg/bbbb)
Prosek	- разломљени број, није обавезно

Uslov (IdKurUslovljen, IdKurUslov)

IdKurUslovljen	- страни кључ (табела Kurs), обавезно, условљен курс
IdKurUslov	- страни кључ (табела Kurs), обавезно, курс који је услов за условљен

Задатак 1 [3 поена]
Потребно је направити SQL упит који исписује све испите који су одржани 2010. године у свим испитним роковима осим у фебруарском испитном року, тако што исписује идентификатор испита, датум одржавања испита и назив испитног рока. Резултат треба сортирати растуће по идентификатору испита.

Резултат дати у форми: IdIsp, Datum, Ispitni rok
У Cactus-у користити таб: Zadatak 1

```
SELECT Ispit.IdIsp, Ispit.Datum, IspitniRok.Naziv AS 'Ispitni rok'
FROM Ispit JOIN IspitniRok USING (IdIRok)
WHERE IspitniRok.Naziv LIKE "% 2010" AND IspitniRok.Naziv NOT LIKE "feb %"
ORDER BY Ispit.IdIsp
```

Задатак 2 [3 поена]
Потребно је направити SQL упит који исписује који студенти су полагали које испите у испитном року са називом 'jun 2011'. Треба исписати и испите у том испитном року које ниједан студент није полагао. Прво исписати назив испитног рока, идентификатор испита, датум полагања испита и индекс студента који је полагао испит. Резултат треба сортирати опадајуће по индентификатору испита, а затим растуће по индексу студента.

Резултат дати у форми: Ispitni rok, IdIsp, Datum, Indeks
У Cactus-у користити таб: Zadatak 2

```
SELECT IspitniRok.Naziv AS 'Ispitni rok', Ispit.IdIsp, Ispit.Datum, Student.Indeks
FROM IspitniRok NATURAL JOIN Ispit LEFT OUTER JOIN Polaganje USING (IdIsp)
        LEFT OUTER JOIN Student USING (Indeks)
WHERE IspitniRok.Naziv='jun 2011'
ORDER BY Ispit.IdIsp DESC, Student.Indeks  ASC
```

Задатак 3 [3 поена]
Потребно је направити SQL упит који приказује испитне рокове који су се догодили у годинама после 2010. и у којима је полагало бар 3 студента. Резултат треба сортирати растуће по идентификатору испитног рока.

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

```
SELECT IdIRok, Naziv
FROM IspitniRok WHERE IdIRok IN(
        SELECT IdIRok
        FROM Ispit NATURAL JOIN Polaganje
        WHERE Datum/10000 > 2010
        GROUP BY IdIRok
        HAVING COUNT(DISTINCT Polaganje.Indeks) > 2
)
ORDER BY IdIRok
```

Задатак 4 [3 поена]
Потребно је направити SQL упит који за сваки ниво исписује просечну зараду по години. Зарди доприноси свако полагање курса када се сматра да је студент платио полагање. Уколико је ниво "I" треба исписати "Osnovni nivo", уколико је "II" тада "Srednji nivo" и уколико је "III" тада "Napredni nivo". Исписати прво назив нивоа, затим просечну годишњу зараду. Зараду исписати за оне нивое за које зарада постоји. Нивое треба исписати од нижих ка вишим.

Резултат дати у форми: Naziv nivoa, Prosecna godisnja zarada.
У Cactus-у користити таб: Zadatak 4

```
SELECT CASE Nivo
        WHEN "I" THEN "Osnovni nivo"
        WHEN "II" THEN "Srednji nivo"
        WHEN "III" THEN "Napredni nivo"
        END AS 'Naziv nivoa', AVG(ZaradaPoGodini) AS 'Prosecna godisnja zarada'
FROM (
        SELECT Kurs.Nivo AS Nivo, Ispit.Datum/10000 AS Godina, SUM(Kurs.Cena) AS ZaradaPoGodini
        FROM Kurs NATURAL JOIN Ispit NATURAL JOIN Polaganje
        GROUP BY Kurs.Nivo, Ispit.Datum/10000
) x
GROUP BY Nivo
ORDER BY Nivo
```

Задатак 5 [4 поена]
Потребно је направити SQL упит који исписује који студент је одређени курс полагао више од једном, као и колико пута га је полагао. Резултат сортирати прво опадајуће по броју излазака, затим растуће по индексу.

Резултат дати у форми: Indeks, IdKur, Broj.
У Cactus-у користити таб Zadatak 5.

```
SELECT Indeks, I.IdKur, COUNT(*) AS Broj
FROM Polaganje P NATURAL JOIN Ispit I
GROUP BY P.Indeks, I.IdKur
HAVING COUNT(*) > 1
ORDER BY Broj DESC, Indeks
```

Задатак 6 [4 поена]
Потребно је направити SQL скрипту која ако постоји табела ***Polaganje*** брише табелу ***Polaganje*** из шеме, а затим формира нову табелу ***Polaganje*** која треба да има одговарајућу структуру и ограничења.

У Cactus-у користити таб Zadatak 6.

```
DROP TABLE IF EXISTS Polaganje;

CREATE TABLE Polaganje(
        IdPol INTEGER PRIMARY KEY AUTOINCREMENT,
        Ocena INT NOT NULL CHECK (Ocena >=5 AND Ocena <= 10),
        Indeks VARCHAR(9) NOT NULL REFERENCES Student(Indeks),
        IdIsp INTEGER NOT NULL REFERENCES Ispit(IdIsp)
);
```

Задатак 7 [4 поена]
Потребно је направити SQL упит који исписује нивое курсева у којима ни један курс тог нивоа не условљава неки други курс. У овом задатку сматрати да у бази може бити произвољно много нивоа, дакле тај број није фиксан ни предефинисан. Сортирати нивое у опадајућем поретку.

Резултат дати у форми: Nivo
У Cactus-у користити таб: Zadatak 7

```
SELECT Kurs.Nivo
FROM Kurs LEFT JOIN Uslov ON (Kurs.IdKur = Uslov.IdKurUslov)
GROUP BY Kurs.Nivo
HAVING (COUNT(Uslov.IdKurUslov) = 0)
ORDER BY Kurs.Nivo DESC
```

Задатак 8 [4 поена]
Потребно је направити SQL упит који исписује све курсеве који се морају положити како би положио курс са називом „OOP“. Резултат треба сортирати растуће по нивоу, затим растуће по идентификатору курса.

Резултат дати у форми: IdKur, Naziv, Nivo.
У Cactus-у користити таб: Zadatak 8
Није дозвољено коришћење погледа.

```
WITH RECURSIVE Stablo(IdKur) AS (
        SELECT IdKur FROM Kurs WHERE Naziv = 'OOP'
        UNION
        SELECT u.IdKurUslov FROM Stablo s, Uslov u WHERE u.IdKurUslovljen = s.IdKur
)
SELECT k.IdKur, k.Naziv, k.Nivo
FROM Stablo s JOIN Kurs k USING(IdKur)
ORDER BY Nivo, IdKur
```

Задатак 9 [5 поена]

Потребно је направити SQL скрипту која мења просечну оцену за студента, затим приказује све податке из табеле **Student** сортиране растуће по индексу студента. За студенте који немају ниједан положен испит исписати 0 за просек.

Резултат дати у форми: Indeks, Prosek

У Cactus-у користити таб: Zadatak 9
Није дозвољено коришћење погледа.

```
UPDATE Student
SET Prosek =
(
        SELECT COALESCE(POcena,0) AS Prosek
        FROM  Student S LEFT OUTER JOIN
                (SELECT Indeks,AVG(Ocena) AS POcena
                 FROM Polaganje P NATURAL JOIN Ispit I
                 WHERE Ocena>5 AND NOT EXISTS ( SELECT *
                         FROM Polaganje P1 NATURAL JOIN Ispit I1
                         WHERE P.Indeks=P1.Indeks AND I.IdKur=I1.IdKur AND I.Datum<I1.Datum)
                 GROUP BY Indeks
                ) USING (Indeks)
        WHERE Student.Indeks=S.Indeks
);

SELECT * FROM Student
ORDER BY Indeks
```

Задатак 10 [5 поена]

Потребно је направити SQL исказ који поставља нову цену за курсеве који припадају нивоу чија је просечна цена мања од просечне цене свих нивоа и уколико се у оквиру тог нивоа налази курс са најмањом пролазношћу. Курс са најмањом пролазношћу је онај који има највише оцена 5. Нова цена за курсеве је за 10% већа од просечне цене свих нивоа.

Након ажурирања података, исписати све податке из табеле ***Kurs*** сортиране растуће по идентификатору курса**.**

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

```
WITH ProsekNivoi (Prosek) AS (
SELECT AVG(Nivo.Cena)
	FROM
		(SELECT K.Nivo, AVG(K.Cena) AS Cena
		FROM Kurs K
		GROUP BY K.Nivo
		) AS Nivo
)
UPDATE Kurs
SET Cena = (SELECT Prosek FROM ProsekNivoi) * 1.1
WHERE (
	(SELECT AVG(K.Cena)
	FROM Kurs K
	WHERE K.Nivo = Kurs.Nivo)
	<
	(SELECT Prosek FROM ProsekNivoi)
)
AND
Kurs.Nivo = (
	SELECT Kurs.Nivo
	FROM Kurs WHERE Kurs.IdKur IN(
		SELECT Ispit.IdKur
		FROM Polaganje NATURAL JOIN Ispit
		WHERE Polaganje.Ocena = 5
		GROUP BY Ispit.IdKur
		ORDER BY COUNT(*) DESC
		LIMIT 1
	)
);

SELECT * FROM Kurs
ORDER BY IdKur
```

Задатак 11 [6 поена]

Потребно је направити SQL упит који исписује колико нивоа је завршио сваки од студената. Студент је завршио одређени ниво уколико је положио све курсеве тог нивоа. Резултат сортирати растуће по индексу студента.

Резултат дати у форми: Indeks, Broj zavrsenih nivoa.
У Cactus-у користити таб: Zadatak 11
Није дозвољено коришћење погледа.

```
WITH NivoKurs (Nivo, BrKurseva) AS
(
	SELECT Nivo, COUNT(*)
	FROM Kurs
	GROUP BY Nivo
),
PolozeniIspitiStudenta (Indeks, Nivo, BrPolozenihPoNivou) AS
(
	SELECT Indeks, Nivo, COUNT(*)
	FROM Polaganje P1 JOIN Ispit I1 USING(IdIsp) JOIN Kurs USING(IdKur)
	WHERE P1.Ocena>5 AND NOT EXISTS (
		SELECT *
		FROM Ispit I2 NATURAL JOIN Polaganje P2
		WHERE  I2.IdKur=I1.IdKur AND I2.Datum >I1.Datum AND P2.Indeks=P1.Indeks
	)
	GROUP BY Indeks, Nivo
)
SELECT Indeks, (
		SELECT COUNT(*)
		FROM NivoKurs K NATURAL JOIN PolozeniIspitiStudenta
		WHERE PolozeniIspitiStudenta.Indeks=S.Indeks AND BrKurseva=BrPolozenihPoNivou
		) AS "Broj polozenih nivoa"
FROM Student S
ORDER BY Indeks
```

Задатак 12 [6 поена]

Потребно је направити SQL упит који за сваког студента исписује колико најмање испита мора положити и колико најмање новца мора дати тако да заврши последњи ниво, односно положи све испите на последњем нивоу. Последњи ниво је ниво који нема ни један курс који условљава други курс. Претпоставити да постоји само један највиши ниво. Студент не мора положити све испите са претходних нивоа. Није потребно исписивати студенте који су положили све испите последњег нивоа. Резултат сортирати опадајуће по броју курсева, затим опадајуће по новцу, затим по индексу студента растуће.

Резултат дати у форми: Indeks, BrojKurseva, Novca
У Cactus-у користити таб: Zadatak 12
Није дозвољено коришћење погледа.

```
WITH RECURSIVE NajvisiNivo(Nivo) AS
(
	SELECT Nivo
	FROM Kurs LEFT JOIN Uslov ON (Kurs.IdKur = Uslov.IdKurUslov)
	GROUP BY Nivo
	HAVING (COUNT(IdKurUslov) = 0)
),
Stablo(IdKur) AS (
	SELECT IdKur FROM Kurs NATURAL JOIN NajvisiNivo
	UNION
	SELECT IdKurUslov
	FROM Stablo s JOIN Uslov u ON (s.idKur = u.idKurUslovljen)
),
DoKraja(Indeks, BrojKurseva, Novca) AS
(
	SELECT Indeks, COUNT(Kurs.IdKur), SUM(Kurs.Cena)
	FROM(
		SELECT Student.Indeks AS Indeks, s.IdKur
		FROM Stablo s, Student
		EXCEPT
		SELECT p.Indeks, i.IdKur
		FROM Polaganje p NATURAL JOIN Ispit i
		WHERE p.Ocena > 5 AND i.Datum = (
			SELECT MAX(i2.Datum)
			FROM Polaganje p2 JOIN Ispit i2 USING(IdIsp)
			WHERE p2.Indeks = p.Indeks AND i2.IdKur = i.IdKur)
	) AS KurseviDoKraja NATURAL JOIN Kurs
	GROUP BY Indeks
)
SELECT * FROM DoKraja
ORDER BY BrojKurseva DESC, Novca DESC, Indeks ASC
```
