3. DZ IZ BAZA PODATAKA / ZADACI ZA VJEŽBU

by v-v

1. Za relaciju **predmetgrupa** kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:

SELECT * FROM **predmetgrupa** WHERE sifpredmet = 1 AND akgodina = 2003;
SELECT * FROM **predmetgrupa** WHERE akgodina > 2000 AND akgodina < 2006 ORDER BY akgodina;
SELECT * FROM **predmetgrupa** WHERE sifnastavnik = 690 AND ozngrupa = 'D-A1';
SELECT * FROM **predmetgrupa** WHERE ozngrupa = 'D-A1' AND akgodina = 2003 AND sifpredmet > 20;
SELECT * FROM **predmetgrupa** ORDER BY akgodina DESC, sifpredmet;
SELECT * FROM **predmetgrupa** ORDER BY sifnastavnik;
SELECT * FROM **predmetgrupa** ORDER BY akgodina DESC, sifpredmet ASC, ozngrupa ASC;

```
CREATE INDEX in1 ON predmetgrupa (sifnastavnik, ozngrupa);
CREATE INDEX in2 ON predmetgrupa (akgodina DESC, sifpredmet ASC, ozngrupa);
```

2. Za relaciju **nastavnik** kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:

SELECT * FROM **nastavnik** WHERE prezimeNastavnik = 'Martinec';
SELECT * FROM **nastavnik** WHERE datumZaposlenOd > '1.1.2006' and datumZaposlenDo < '1.1.2008';
SELECT * FROM **nastavnik** ORDER BY koef DESC;
SELECT * FROM **nastavnik** ORDER BY prezimeNastavnik, imeNastavnik;
SELECT * FROM **nastavnik** ORDER BY prezimeNastavnik DESC;
SELECT * FROM **nastavnik** ORDER BY datumZaposlenDo DESC, datumZaposlenOd ASC;
SELECT * FROM **nastavnik** ORDER BY koef DESC, prezimeNastavnik ASC, imeNastavnik ASC;

```
CREATE INDEX in1 ON nastavnik (koef DESC, prezimeNastavnik, imeNastavnik);
CREATE INDEX in2 ON nastavnik (prezimeNastavnik, imeNastavnik);
CREATE INDEX in3 ON nastavnik (datumZaposlenDo DESC, datumZaposlenOd);
```

3. Osigurati entitetski integritet i integritet ključa u relaciji **mjesto**.
Napomena: isti naziv mjesta se može pojaviti više puta.

Osigurati da se u relaciji **nastavnik** može evidentirati samo mjesto stanovanja koje (ukoliko nije NULL) postoji u relaciji **mjesto**.

```
ALTER TABLE mjesto ADD CONSTRAINT PRIMARY KEY(pBr);
ALTER TABLE nastavnik ADD CONSTRAINT FOREIGN KEY(pbrStanNastavnik) REFERENCES
mjesto(pBr);
```

4. Za relaciju **student** kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:

SELECT * FROM **student** WHERE datumRod > '01.01.1980' ORDER BY datumRod;
SELECT * FROM **student** WHERE prezimeStudent = 'Salopek' AND imeStudent = 'Andrej';
SELECT * FROM **student** WHERE prezimeStudent = 'Salopek';
SELECT * FROM **student** ORDER BY prezimeStudent, imeStudent;
SELECT * FROM **student** WHERE datumRod > '01.01.1980';
SELECT * FROM **student** ORDER BY datumRod, jmbag;

```
CREATE INDEX in1 ON student (datumRod, jmbag);
CREATE INDEX in2 ON student (prezimeStudent, imeStudent);
```

5. Primarni ključ u relaciji **predmetGrupa** je skup atributa: {sifPredmet, akGodina, oznGrupa}.**Jednom** SQL naredbom osigurajte entitetski integritet i integritet ključa u toj relaciji.

   Primarni ključ u relaciji **predmet** je atribut: sifPredmet. **Jednom** SQL naredbom osigurajte entitetski integritet i integritet ključa u toj relaciji.

   Osigurati referencijski integritet među relacijama **predmetGrupa** i **predmet**.

```
ALTER TABLE predmetGrupa ADD CONSTRAINT PRIMARY KEY (sifPredmet,akGodina,
oznGrupa);
ALTER TABLE predmet ADD CONSTRAINT PRIMARY KEY(sifPredmet);
ALTER TABLE predmetGrupa ADD CONSTRAINT FOREIGN KEY(sifPredmet) REFERENCES
predmet(sifPredmet);
```

6. Primarni ključ u relaciji **upisanPredmet** je skup atributa: {jmbag, sifPredmet, akGodina, oznGrupa}.**J ednom** SQL naredbom osigurajte entitetski integritet i integritet ključa u toj relaciji.

   Primarni ključ u relaciji **predmetGrupa** je skup atributa: {sifPredmet, akGodina, oznGrupa}. **Jednom** SQL naredbom osigurajte entitetski integritet i integritet ključa u toj relaciji.

   Osigurati referencijski integritet među relacijama **upisanPredmet** i **predmetGrupa**.

```
ALTER TABLE upisanPredmet
ADD CONSTRAINT PRIMARY KEY (jmbag, sifPredmet, akGodina, oznGrupa);

ALTER TABLE predmetGrupa
ADD CONSTRAINT PRIMARY KEY(sifPredmet, akGodina, oznGrupa);

ALTER TABLE upisanPredmet
ADD CONSTRAINT FOREIGN KEY (sifPredmet, akGodina, oznGrupa)
REFERENCES predmetGrupa (sifPredmet, akGodina, oznGrupa);
```

7. Primarni ključ u relaciji **predmetGrupa** je skup atributa: {sifPredmet, akGodina, oznGrupa}.**Jednom** SQL naredbom osigurajte entitetski integritet i integritet ključa u toj relaciji.

   Osigurati entitetski integritet i integritet ključa u relaciji **dvorana**.

   Osigurati referencijski integritet među relacijama **predmetGrupa** i **dvorana**.

```
ALTER TABLE predmetGrupa ADD CONSTRAINT PRIMARY KEY (sifPredmet, akGodina,
oznGrupa);

ALTER TABLE dvorana ADD CONSTRAINT PRIMARY KEY(oznDvorana);

ALTER TABLE predmetGrupa ADD CONSTRAINT FOREIGN KEY (oznDvorana) REFERENCES
dvorana(oznDvorana);
```

8. Za relaciju **upisanpredmet** kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:
   SELECT * FROM **upisanpredmet** WHERE akgodina = 2005 AND ozngrupa = 'C-B4';
   SELECT * FROM **upisanpredmet** WHERE sifpredmet = 10 AND ozngrupa = 'C-B4' AND akgodina = 2005 ;
   SELECT * FROM **upisanpredmet** ORDER BY akgodina, ozngrupa DESC, sifpredmet DESC;
   SELECT * FROM **upisanpredmet** ORDER BY akgodina DESC;
   SELECT * FROM **upisanpredmet** WHERE datumocjena = '01.01.2006' AND sifpredmet = 10;
   SELECT * FROM **upisanpredmet** ORDER BY sifpredmet ASC, datumocjena DESC, ocjena DESC;

```
CREATE INDEX in1 ON upisanpredmet (sifpredmet, datumocjena DESC, ocjena
DESC);
CREATE INDEX in2 ON upisanpredmet (akgodina, ozngrupa DESC, sifpredmet DESC);
```

9. Primarni ključ u relaciji **grupa** je skup atributa: {akGodina, oznGrupa}.**Jednom** SQL naredbom osigurajte entitetski integritet i integritet ključa u toj relaciji.

   Osigurati referencijski integritet među relacijama **predmetGrupa** i **grupa**.

```
ALTER TABLE grupa ADD CONSTRAINT PRIMARY KEY(akGodina, oznGrupa);
ALTER TABLE predmetGrupa ADD CONSTRAINT FOREIGN KEY(akGodina, oznGrupa)
REFERENCES grupa(akGodina, oznGrupa);
```

10. Napraviti **2 virtualne relacije**:

    1. Napraviti virtualnu relaciju stanBr sa shemom relacije STANBR = pbrstan, broj koja će omogućiti pregled broja studenata koji su položili barem jedan predmet u 2008/2009akademskoj godini po poštanskom broju stanovanja.

    Primjer rezultata:

    SELECT * FROM stanBr; **upisanPredmet**

pbrstan	broj
99999	1
(null)	5
11111	3
...	...

Primijetite mogućnost postojanja n-torke koja predstavlja studente čije mjesto stanovanja nije poznato.

2. Napraviti virtualnu relaciju mjestoBr sa shemom relacije MJESTOBR = nazmjesto, broj koja će po SVIM mjestima omogućiti pregled broja studenata koji su položili predmet u 2008/2009 akademskoj godini koji stanuju u tom mjestu. Uključiti i studente čije mjesto stanovanja nije poznato.
Virtualnu relaciju mjestoBr obavezno napraviti pomoću relacije stanBr.

Primjer rezultata:

SELECT * FROM mjestoBr;

nazmjesto	broj
Zadar	325
(null)	267
Zagreb	355
Hum	(null)
Makarska	(null)
...	...

Primijetite mogućnost postojanja n-torke koja predstavlja studente čije mjesto stanovanja nije poznato i mogućnost postojanja n-torki koje predstavljaju mjesta za koja nema položenih predmeta u zadanoj akademskoj godini.

```
CREATE VIEW stanBr (pbrstan, broj) AS
SELECT pbrStanStudent, COUNT(DISTINCT upisanPredmet.jmbag) FROM
upisanPredmet RIGHT OUTER JOIN student
ON upisanPredmet.jmbag = student.jmbag
WHERE akGodina = 2008 AND ocjena > 1
GROUP BY pbrStanStudent;

CREATE VIEW mjestoBr (nazmjesto, broj) AS
SELECT nazMjesto, broj FROM
stanBr FULL OUTER JOIN mjesto
ON mjesto.pbr = stanBr.pbrStan;
```

11. Za relaciju **student** kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:

SELECT * FROM **student** WHERE datumrod > '01.01.1980' ORDER BY datumrod;
SELECT * FROM **student** WHERE prezimeStudent = 'Salopek' AND imeStudent = 'Andrej';
SELECT * FROM **student** WHERE prezimeStudent = 'Salopek';
SELECT * FROM **student** ORDER BY datumrod DESC;
SELECT * FROM **student** WHERE jmbg = '0101980554687';
SELECT * FROM **student** WHERE datumrod > '01.01.1980';
SELECT * FROM **student** ORDER BY datumrod, jmbag;

```
CREATE INDEX in1 ON student (datumrod,jmbag);
CREATE INDEX in2 ON STUDENT (prezimeStudent, imeStudent);
CREATE INDEX in3 ON STUDENT (jmbg);
```

12. Napraviti **2 virtualne relacije**:

1. Napraviti virtualnu relaciju **stanPO** sa shemom relacije **STANPO = pbrstan, prosjekocjena** koja će omogućiti pregled prosjeka ocjena **položenih** predmeta u **2007/2008**akademskoj godini po **poštanskom broju stanovanja.** Prosjek ocjena nije potrebno zaokruživati.

Primjer rezultata:
SELECT * FROM stanPO;

pbrstan	prosjekocjena
99999	3.25
(null)	2.66666666666666666666666666667
11111	3.67346938775510204081632653061 2
...	...

Primijetite mogućnost postojanja n-torke koja predstavlja studente čije mjesto stanovanja nije poznato.

2. Napraviti virtualnu relaciju **mjestoPO** sa shemom relacije **MJESTOPO = nazmjesto, prosjekocjena** koja će po **SVIM mjestima** omogućiti pregled prosjeka ocjena **položenih**predmeta u **2007/2008** akademskoj godini studenata koji **stanuju** u tom mjestu**.** Uključiti i prosjek ocjena za studente čije mjesto stanovanja nije poznato. Prosjek ocjena nije potrebno zaokruživati.
Virtualnu relaciju mjestoPO <u>obavezno</u> napraviti pomoću relacije stanPO.
Primjer rezultata:
SELECT * FROM mjestoPO;

nazmjesto	prosjekocjena
Zadar	3.25
(null)	2.66666666666666666666666666667
Zagreb	3.67346938775510204081632653061 2
Hum	(null)
Makarska	(null)
...	...

Primijetite mogućnost postojanja n-torke koja predstavlja studente čije mjesto stanovanja nije poznato i mogućnost postojanja n-torki koje predstavljaju mjesta za koja prosjek ocjena nije poznat.

```
CREATE VIEW StanPO(pbrstan,prosjekocjena) AS
SELECT pbrStanStudent, AVG(ocjena) FROM
upisanpredmet RIGHT OUTER JOIN student
ON upisanpredmet.jmbag = student.jmbag
WHERE akGodina = 2007 AND ocjena > 1
GROUP BY pbrStanStudent;

CREATE VIEW mjestoPO(nazmjesto,prosjekocjena) AS
SELECT nazMjesto, prosjekocjena FROM
StanPO FULL OUTER JOIN mjesto
ON StanPO.pbrStan = mjesto.pbr;
```

13. Za relaciju **upisanpredmet** kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:

SELECT * FROM **upisanpredmet** WHERE akgodina = 2005 AND jmbag = '055000491';
SELECT * FROM **upisanpredmet** WHERE akgodina = 2005 AND ozngrupa = 'C-B4';
SELECT * FROM **upisanpredmet** WHERE sifpredmet = 10 AND ozngrupa = 'C-B4' AND akgodina = 2005 ;
SELECT * FROM **upisanpredmet** ORDER BY akgodina, ozngrupa DESC, sifpredmet DESC;
SELECT * FROM **upisanpredmet** ORDER BY akgodina DESC;
SELECT * FROM **upisanpredmet** ORDER BY akgodina ASC, jmbag ASC, sifpredmet ASC, ocjena DESC;

```
CREATE INDEX in1 ON upisanpredmet (akgodina ASC, jmbag ASC, sifpredmet ASC,
ocjena DESC);
CREATE INDEX in2 ON upisanpredmet (akgodina ASC, ozngrupa DESC, sifpredmet
DESC);
```

14. Primarni ključ u relaciji **grupa** je skup atributa: {akGodina, oznGrupa}.**Jednom** SQL naredbom osigurajte entitetski integritet i integritet ključa u toj relaciji.

Osigurati referencijski integritet među relacijama **predmetGrupa** i **grupa**.

```
ALTER TABLE grupa ADD CONSTRAINT PRIMARY KEY (akGodina, oznGrupa);

ALTER TABLE predmetGrupa ADD CONSTRAINT FOREIGN KEY (akGodina, oznGrupa)
REFERENCES grupa(akGodina, oznGrupa);
```

15. Svim nastavnicima čiji je koeficijent jednak maksimalnom koeficijentu postaviti koeficijent na minimalni koeficijent.
Npr.

Prije:

sifNastavnik	koef
1	3.0
2	7.9
3	3.0

Poslije:

sifNastavnik	koef
1	3.0
2	7.9
3	8.5

Napomena: radi preglednosti iz primjera su izbačeni ostali atributi relacije nastavnik.

```
CREATE TEMP TABLE koeficijenti (sifNast INTEGER, koef DECIMAL(3,2) );
INSERT INTO koeficijenti SELECT sifNastavnik, koef FROM nastavnik;

UPDATE nastavnik SET koef = (SELECT MIN(koef) FROM koeficijenti)
WHERE koef = (SELECT MAX(koef) FROM koeficijenti);
```

16. Svim predmetima koji imaju maksimalan broj sati tjedno povećati broj ECTS bodova za 20%. Nove ECTS bodove zaokružiti na cijeli broj.
Npr.
Prije:

sifPredmet	nazPredmet	ectsBod	ukBrSatiTjedno
1	Matematika	6.0	4.0
2	Fizika	7.0	5.0
3	Informatika	7.0	5.0

Poslije:

sifPredmet	nazPredmet	ectsBod	ukBrSatiTjedno
1	Matematika	6.0	4.0
2	Fizika	6.0	5.0
3	Informatika	6.0	5.0

```
CREATE TEMP TABLE maxSati (sifPredmet INTEGER);
INSERT INTO maxSati
SELECT sifpredmet FROM predmet GROUP BY sifPredmet, ukBrSatiTjedno HAVING
ukBrSatiTjedno = MAX(ukBrSatiTjedno);

UPDATE predmet SET ectsBod = ROUND(ectsBod*1.2,0)
```

17. Za relaciju **predmet** kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:

SELECT DISTINCT ECTSBod, ukBrSatiTjedno FROM **predmet**
WHERE ECTSBod > 2 AND ECTSBod < 5 AND ukBrSatiTjedno = 2;
SELECT * FROM **predmet** WHERE nazPredmet = 'Baze podataka';
SELECT nazPredmet FROM **predmet** ORDER BY ECTSBod DESC, ukBrSatiTjedno DESC, nazPredmet;
SELECT nazPredmet FROM **predmet** ORDER BY nazPredmet, ukBrSatiTjedno DESC;

```
CREATE INDEX in1 ON predmet (ECTSBod DESC, ukBrSatiTjedno DESC, nazPredmet
ASC);
CREATE INDEX in2 ON predmet (nazPredmet ASC, ukBrSatiTjedno DESC);
```

18. Za relaciju **nastavnik** kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:

SELECT * FROM **nastavnik** WHERE imeNastavnik = 'Ivana' AND prezimeNastavnik = 'Martinec';
SELECT * FROM **nastavnik** WHERE datumZaposlenOd > '1.1.2006';
SELECT * FROM **nastavnik** ORDER BY prezimeNastavnik DESC, imeNastavnik;
SELECT * FROM **nastavnik** ORDER BY prezimeNastavnik;
SELECT * FROM **nastavnik** ORDER BY datumZaposlenDo, datumZaposlenOd;

```
CREATE INDEX in1 ON nastavnik (prezimeNastavnik DESC, imeNastavnik ASC);
CREATE INDEX in2 ON nastavnik (datumZaposlenDo, datumZaposlenOd);
CREATE INDEX in3 ON nastavnik (datumZaposlenOd);
```

19. Za relaciju **predmetgrupa** kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:

SELECT * FROM **predmetgrupa** WHERE sifpredmet = 1 AND akgodina = 2003;
SELECT * FROM **predmetgrupa** WHERE akgodina > 2000 AND akgodina < 2005;
SELECT * FROM **predmetgrupa** ORDER BY ozngrupa ASC, sifnastavnik DESC;
SELECT * FROM **predmetgrupa** ORDER BY akgodina DESC, sifpredmet ASC;

```
CREATE INDEX in1 ON predmetgrupa (ozngrupa ASC, sifnastavnik DESC);
CREATE INDEX in2 ON predmetgrupa (akgodina DESC, sifpredmet ASC);
```

20. Primarni ključ u relaciji **upisanPredmet** je skup atributa: {jmbag, sifPredmet, akGodina, oznGrupa}.**Jednom** SQL naredbom osigurajte entitetski integritet i integritet ključa u toj relaciji.

Primarni ključ u relaciji **predmet** je atribut: sifPredmet. **Jednom** SQL naredbom osigurajte entitetski integritet i integritet ključa u toj relaciji.

Osigurati referencijski integritet među relacijama **upisanPredmet** i **predmet** .

```
ALTER TABLE upisanPredmet
ADD CONSTRAINT PRIMARY KEY (jmbag, sifPredmet,akGodina, oznGrupa);

ALTER TABLE predmet ADD CONSTRAINT PRIMARY KEY(sifPredmet);

ALTER TABLE upisanPredmet
ADD CONSTRAINT FOREIGN KEY(sifPredmet) REFERENCES predmet(sifPredmet);
```

21. Svim predmetima čiji je broj sati tjedno manji od prosjeka smanjiti broj ECTS bodova za 20%. Nove ECTS bodove zaokružiti na cijeli broj.
Npr.

Prije:

sifPredmet	nazPredmet	ectsBod	ukBrSatiTjedno
1	Matematika	5.0	4.0
2	Fizika	5.0	4.0
3	Informatika	6.0	5.0

Poslije:

sifPredmet	nazPredmet	ectsBod	ukBrSatiTjedno
1	Matematika	6.0	4.0
2	Fizika	6.0	4.0
3	Informatika	6.0	5.0

```
CREATE TEMP TABLE ispodprosjecni (sifPredmet INTEGER);

INSERT INTO ispodprosjecni
SELECT sifpredmet FROM predmet GROUP BY sifPredmet, ukBrSatiTjedno HAVING
ukBrSatiTjedno <= AVG(ukBrSatiTjedno);

UPDATE predmet SET ectsBod = ROUND(ectsBod*0.8,0)
WHERE sifPredmet IN (SELECT sifPredmet FROM ispodprosjecni);
```

22. Svim predmetima čiji je broj sati tjedno veći od prosjeka povećati broj ECTS bodova za 20%. Nove ECTS bodove zaokružiti na cijeli broj.
Npr.

Prije:

sifPredmet	nazPredmet	ectsBod	ukBrSatiTjedno
1	Matematika	6.0	4.0
2	Fizika	6.0	4.0
3	Informatika	7.0	5.0

Poslije:

sifPredmet	nazPredmet	ectsBod	ukBrSatiTjedno
1	Matematika	6.0	4.0
2	Fizika	6.0	4.0
3	Informatika	6.0	5.0

```
CREATE TEMP TABLE ispodprosjecni (sifPredmet INTEGER);

INSERT INTO ispodprosjecni
SELECT sifpredmet FROM predmet GROUP BY sifPredmet, ukBrSatiTjedno HAVING
ukBrSatiTjedno >= AVG(ukBrSatiTjedno);

UPDATE predmet SET ectsBod = ROUND(ectsBod*1.2,0)
WHERE sifPredmet IN (SELECT sifPredmet FROM ispodprosjecni);
```

23. Osigurati entitetski integritet i integritet ključa u relaciji **mjesto**.
Napomena: isti naziv mjesta se može pojaviti više puta.
Osigurati sva ograničenja referencijskog integriteta među relacijama **student** i **mjesto.**

```
ALTER TABLE mjesto ADD CONSTRAINT PRIMARY KEY(pBr);
ALTER TABLE student ADD CONSTRAINT FOREIGN KEY(pbrStanStudent) REFERENCES
mjesto(pBr);
ALTER TABLE student ADD CONSTRAINT FOREIGN KEY(pbrRodStudent) REFERENCES
mjesto(pbr);
```

24. **Napraviti 2 virtualne relacije:**

1. Napraviti virtualnu relaciju **polozeno** sa shemom relacije **POLOZENO = sifraPredmet, brojUspjesnihPolaganja** koja će omogućiti pregled šifri **svih** predmeta i broja uspješnih polaganja u školskoj godini **2006/2007**.

Primjer rezultata:
SELECT * FROM polozeno;

sifraPredmet	brojUspjesnihPolaganja
2	13
29	29
62	0
...	...

Primijetite mogućnost postojanja n-torke koja predstavlja predmete koje nijedan student nije uspješno položio navedene godine.

2. Napraviti virtualnu relaciju **polozenoNaziv** sa shemom relacije **POLOZENONAZIV= nazivPredmet, brojUspjesnihPolaganja** koja će omogućiti pregled naziva **svih** predmeta i broja uspješnih polaganja u školskoj godini **2006/2007**.
Virtualnu relaciju polozenoNaziv obavezno napraviti pomoću relacije polozeno.
Primjer rezultata:
SELECT * FROM polozenoNaziv;

nazivPredmet	brojUspjesnihPolaganja
Matematika 2	0
Logička Algebra	21
Fizika 1	0
Operacijski sustavi	16
Baze podataka	0
...	...

```
CREATE VIEW polozeno (sifraPredmet, brojUspjesnihPolaganja) AS
SELECT predmet.sifPredmet, (SELECT COUNT(jmbag)
       FROM upisanPredmet
       WHERE akGodina = 2006 AND ocjena > 1
       AND upisanPredmet.sifPredmet = predmet.sifPredmet)
FROM predmet;

CREATE VIEW polozenoNaziv (nazivPredmet, brojUspjesnihPolaganja) AS
SELECT predmet.nazPredmet, polozeno.brojUspjesnihPolaganja
FROM predmet JOIN polozeno
ON predmet.sifPredmet = polozeno.sifraPredmet;
```

25. Za relaciju **predmet** kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:

SELECT * FROM **predmet** WHERE nazPredmet = 'Baze podataka';
SELECT * FROM **predmet** WHERE ECTSBod > 2 and ECTSBod < 7;
SELECT * FROM **predmet** ORDER BY ECTSBod DESC*, ukBrSatiTjedno
SELECT * FROM **predmet** ORDER BY nazPredmet, ECTSBod DESC;

```
CREATE INDEX in1 ON predmet (nazPredmet ASC, ECTSBod DESC);
CREATE INDEX in2 ON predmet (ECTSBod ASC, ukBrSatiTjedno ASC);
```

*krivo zadano (nebi trebalo pisati DESC)

26. Svim grupama iz akademske godine 2009/2010 čiji kapacitet odgovara maksimalnom kapacitetu svih grupa na oznaku dodati sufiks "-velika" (npr. grupa "A-A" će postati "A-A-velika").

(NAPOMENA: Vodite računa o mogućem postojanju praznih mjesta u originalnim nazivima grupa!)

```
CREATE TEMP TABLE najvece (oznGrupa CHAR(10));
INSERT INTO najvece SELECT oznGrupa FROM grupa
WHERE akGodina = 2009
GROUP BY oznGrupa, kapacitet
HAVING kapacitet = MAX(kapacitet);

UPDATE grupa SET oznGrupa = TRIM(oznGrupa)||'-velika'
WHERE oznGrupa IN (SELECT oznGrupa FROM najvece);
```

27. Za relaciju **predmet** kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:

SELECT DISTINCT ECTSBod, ukBrSatiTjedno FROM **predmet** WHERE ECTSBod > 2 AND ECTSBod < 5 AND ukBrSatiTjedno = 2;
SELECT * FROM **predmet** WHERE nazPredmet = 'Baze podataka';
SELECT nazPredmet FROM **predmet** ORDER BY ECTSBod DESC, ukBrSatiTjedno DESC, nazPredmet;
SELECT nazPredmet FROM **predmet** ORDER BY nazPredmet, ukBrSatiTjedno DESC;

```
CREATE INDEX in1 ON predmet (nazPredmet ASC, ukBrSatiTjedno DESC);
CREATE INDEX in2 ON predmet (ECTSBod DESC, ukBrSatiTjedno DESC, nazPredmet);
```

28. Osigurati entitetski integritet i integritet ključa u relaciji **mjesto**.
Napomena: isti naziv mjesta se može pojaviti više puta.
Osigurati da se u relaciji **nastavnik** može evidentirati samo mjesto stanovanja koje (ukoliko nije NULL) postoji u relaciji mjesto..

```
ALTER TABLE mjesto ADD CONSTRAINT PRIMARY KEY(pBr);

ALTER TABLE nastavnik ADD CONSTRAINT FOREIGN KEY(pBrStanNastavnik) REFERENCES
mjesto(pBr);
```

29. Za relaciju **student** kreirati najmanji mogući broj indeksa koji će omogućiti efikasno obavljanje svih navedenih upita:
SELECT * FROM **student** WHERE datumrod > '01.01.1980' ORDER BY datumrod;
SELECT * FROM **student** WHERE prezimeStudent = 'Salopek' AND imeStudent = 'Andrej';
SELECT * FROM **student** WHERE prezimeStudent = 'Salopek';
SELECT * FROM **student** ORDER BY datumrod DESC;
SELECT * FROM **student** WHERE jmbg = '0101980554687';
SELECT * FROM **student** WHERE datumrod > '01.01.1980';
SELECT * FROM **student** ORDER BY datumrod, jmbag;

```
CREATE INDEX in1 ON student (datumrod, jmbag);
CREATE INDEX in2 ON student (prezimeStudent, imeStudent);
CREATE INDEX in3 ON student (jmbg);
```

30. Napraviti **2 virtualne relacije**:

1. Napraviti virtualnu relaciju stanBr5 sa shemom relacije STANBR5 = pbrstan, broj5 koja će omogućiti pregled broja studenata koji su položili predmet s ocjenom 5 u 2007/2008 akademskoj godini po poštanskom broju stanovanja.

Primjer rezultata:

SELECT * FROM stanBr5; **upisanPredmet**

pbrstan	broj5
99999	1
(null)	5
11111	3
...	...

Primijetite mogućnost postojanja n-torke koja predstavlja studente čije mjesto stanovanja nije poznato.

2. Napraviti virtualnu relaciju mjestoBr sa shemom relacije MJESTOBR5 = nazmjesto, broj5 koja će po SVIM mjestima omogućiti pregled broja studenata koji su položili predmet s ocjenom 5 u 2007/2008 akademskoj godini koji stanuju u tom mjestu. Uključiti i studente čije mjesto stanovanja nije poznato.
Virtualnu relaciju mjestoBr obavezno napraviti pomoću relacije stanBr5.

Primjer rezultata:

SELECT * FROM mjestoBr5;

nazmjesto	broj5
Zadar	325
(null)	267
Zagreb	355
Hum	(null)
Makarska	(null)
...	...

Primijetite mogućnost postojanja n-torke koja predstavlja studente čije mjesto stanovanja nije poznato i mogućnost postojanja n-torki koje predstavljaju mjesta za koja nema položenih predmeta u zadanoj akademskoj godini.

```
CREATE VIEW stanBr5 (pbrstan, broj5) AS
SELECT pbrstanstudent, COUNT(student.jmbag) FROM student
WHERE student.jmbag IN (SELECT jmbag FROM upisanPredmet WHERE
upisanPredmet.akgodina = 2007 and upisanPredmet.ocjena = 5)
GROUP BY pbrstanstudent;

CREATE VIEW mjestoBr5 (nazmjesto, broj5) AS
SELECT nazMjesto, broj5
FROM stanBr5 FULL OUTER JOIN mjesto
ON mjesto.pbr = stanBr5.pbrstan;
```