Baze podataka

Predavanja

5. SQL – 2. dio

Ožujak, 2021.

Podupiti

Podupiti (*Subqueries*)

vozilo

sifVoz	nosivost
101	2500
102	2000
103	800
104	1000

teret

sifTeret	tezina
1001	1800
1002	1200
1003	1000

- Ispisati podatke o vozilima čija je nosivost veća od težine najtežeg tereta
- Pogrešan način: prvo obaviti upit kojim se određuje težina najtežeg tereta

```
SELECT MAX(tezina) FROM teret;
```

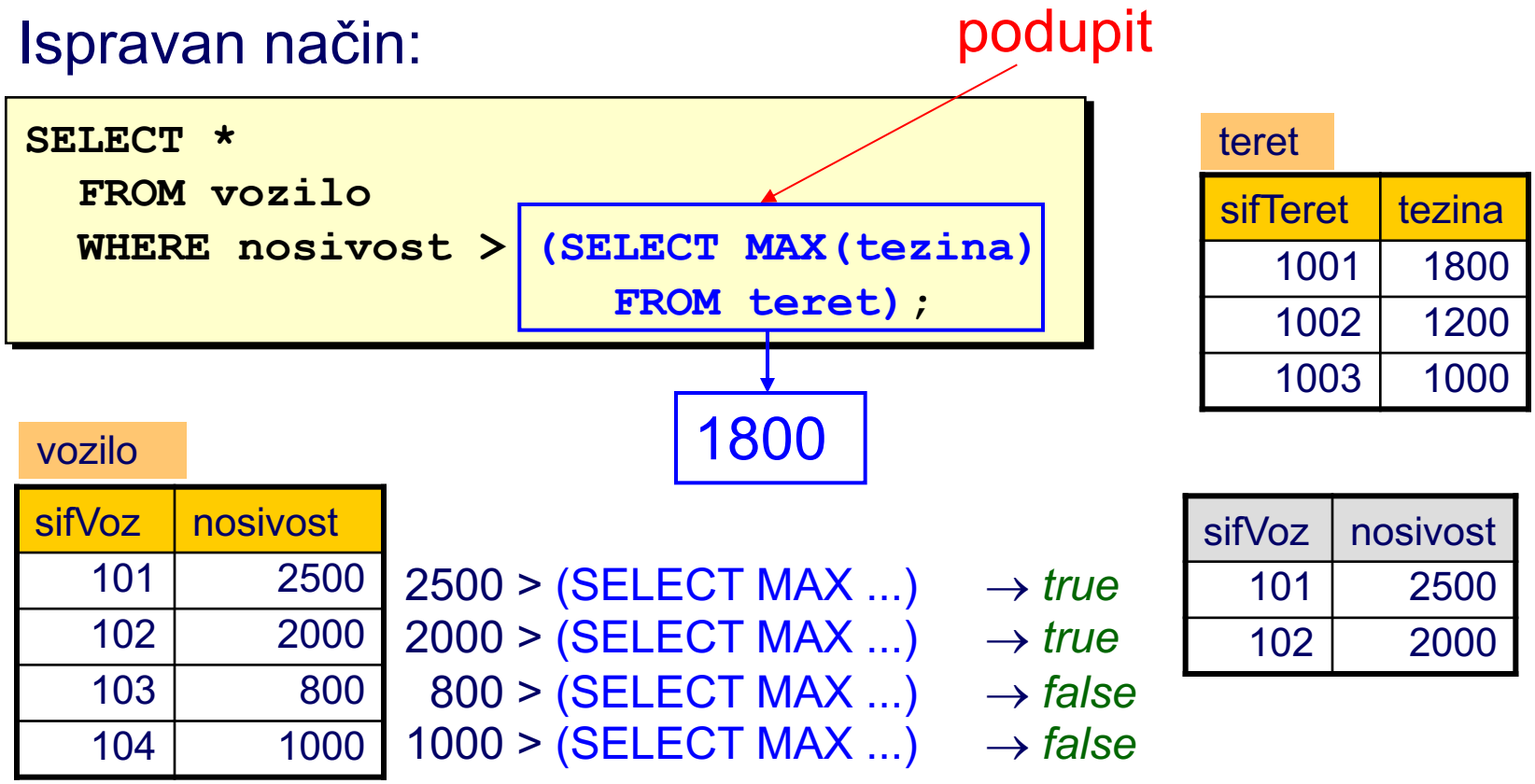
- Zapamtiti dobiveni rezultat (1800), te napisati novi upit:

```
SELECT *
  FROM vozilo
  WHERE nosivost > 1800;
```

sifVoz	nosivost
101	2500
102	2000

Podupiti (*Subqueries*)

- Ispravan način:



- U navedenom primjeru je rezultat podupita jednak za svaku n-torku iz relacije vozilo, stoga je (fizički promatrano) rezultat podupita dovoljno izračunati samo jednom tijekom obavljanja upita

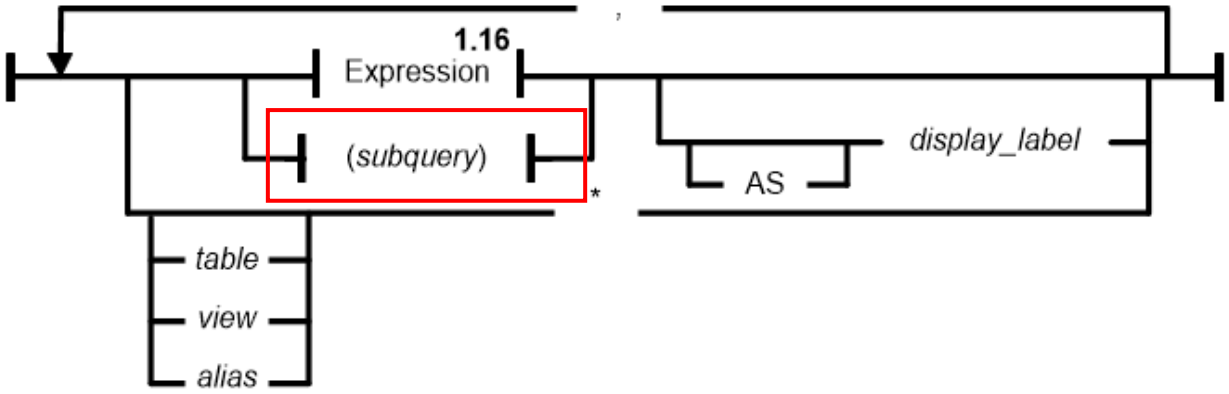
Podupiti (*Subqueries*)

- podupit je upit koji je ugrađen u neki drugi upit
  - upit u kojeg je podupit ugrađen naziva se vanjski upit (*outer query*)
  - osim izraza **podupit** (*subquery*), u literaturi se također koristi i izraz **ugniježđeni upit** (*nested query*)
- podupit se u vanjski upit može ugraditi
  - u uvjet (*Condition*) u WHERE dijelu vanjskog upita
  - u uvjet (*Condition*) u HAVING dijelu vanjskog upita
  - u listu za selekciju (*SELECT List*) vanjskog upita
- podupit može sadržavati sve do sada spomenute dijelove SELECT naredbe osim ORDER BY dijela naredbe
- u vanjski upit se može ugraditi više podupita, u svaki od podupita se može ugraditi više podupita, itd.

Skalarni podupit (*Scalar subquery*)

- za početak, razmatrat će se najjednostavniji oblik podupita: podupit čiji je rezultat **jedna jednostavna** vrijednost (skalar)
  - npr. podatak tipa: cijeli broj, niz znakova, datum, itd.
- može se reći: rezultat skalarnog podupita je "relacija" stupnja jedan i kardinalnosti jedan
  - vrijednost atributa n-torke dotične "relacije" se u vanjskom upitu koristi kao skalarna vrijednost

Podupiti u listi za selekciju



vozilo

sifVoz	nosivost
101	2500
102	2000
103	800
104	1000

teret

sifTeret	tezina
1001	1800
1002	1200
1003	1000

- Ispisati podatke o svim vozilima. Uz svako vozilo ispisati podatak o najvećoj težini tereta

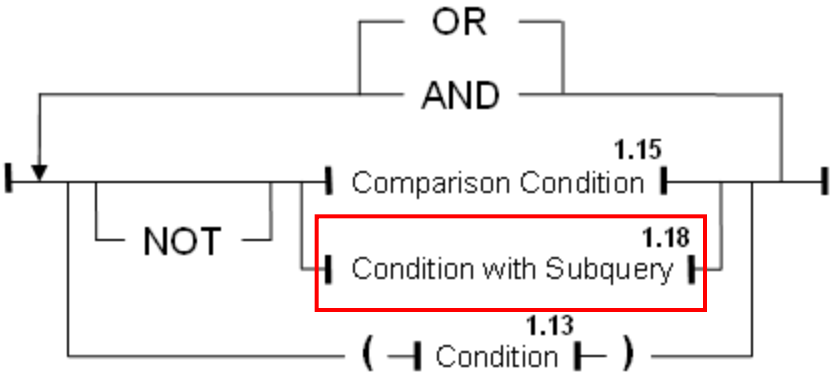
```
SELECT *
     , (SELECT MAX(tezina)
          FROM teret) AS maxTezina
  FROM vozilo;
```

sifVoz	nosivost	maxTezina
101	2500	1800
102	2000	1800
103	800	1800
104	1000	1800

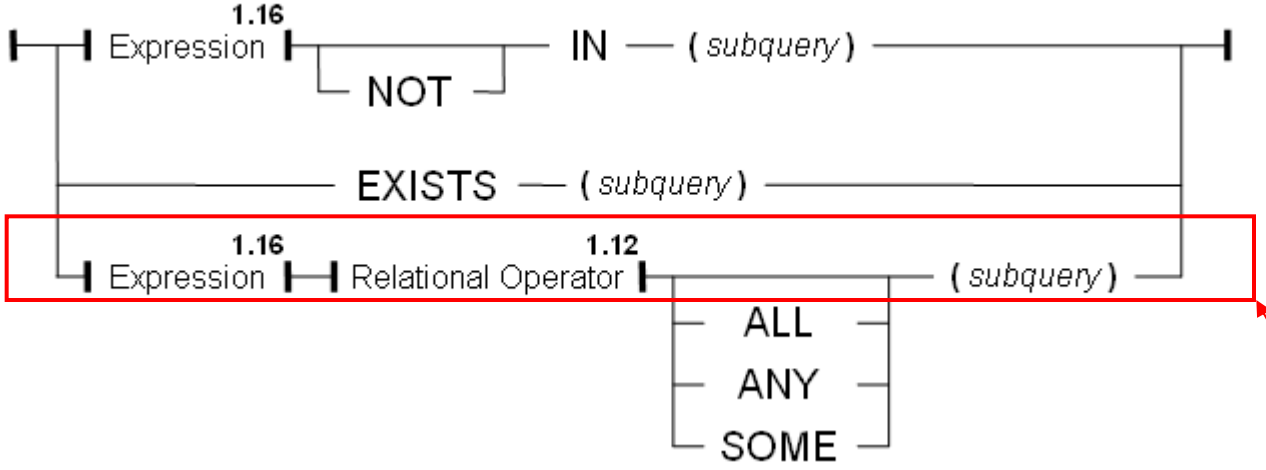
Podupiti u WHERE dijelu naredbe

1.5. WHERE Clause

1.13. Condition



1.18. Condition with Subquery



- Ako se u WHERE ili HAVING dijelu naredbe koristi ovdje označeni oblik uvjeta, dopušteno je koristiti isključivo skalarne podupite

Podupiti u WHERE dijelu naredbe

vozilo

sifVoz	nosivost
101	2500
102	2000
103	800
104	1000

teret

sifTeret	tezina
1001	1800
1002	1200
1003	1000

- Ispisati podatke o vozilima čija je nosivost veća od težine najtežeg tereta

```
SELECT *
  FROM vozilo
  WHERE nosivost > (SELECT MAX(tezina)
                    FROM teret);
```

sifVoz	nosivost
101	2500
102	2000

Podupiti u WHERE dijelu naredbe

- Ispisati podatke o studentima koji stanuju u mjestu Ludbreg

stud

mbr	prez	pbrSt
100	Horvat	42230
101	Kolar	21000
102	Novak	42230

mjesto

pbr	nazMjesto
42000	Varaždin
42230	Ludbreg
21000	Split

```
SELECT * FROM stud
  WHERE pbrSt = (SELECT pbr FROM mjesto
                   WHERE nazMjesto = 'Ludbreg');
```

→ 42230

- Često se problem može riješiti bez podupita. U konkretnom slučaju, **bolje** rješenje glasi:

```
SELECT stud.*
  FROM stud
       JOIN mjesto
         ON stud.pbrSt = mjesto.pbr
  WHERE nazMjesto = 'Ludbreg';
```

Podupiti u WHERE dijelu naredbe

stud

mbr	prez	pbrSt
100	Horvat	42230
101	Kolar	21000
102	Novak	42230

mjesto

pbr	nazMjesto
42000	Varaždin
42230	Ludbreg
21000	Split

- Ukoliko podupit čiji bi rezultat trebao biti skalar vrati više od jedne n-torke ili više nego jedan atribut, sustav će dojaviti pogrešku

```
SELECT * FROM stud
  WHERE pbrSt = (SELECT pbr FROM mjesto
                 WHERE nazMjesto LIKE '%r%');
```

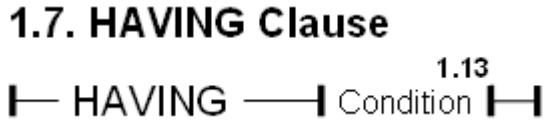
→ Pogreška

- Ukoliko podupit čiji bi rezultat trebao biti skalar ne vrati niti jednu n-torku, dobivena skalarna vrijednost će biti NULL vrijednost

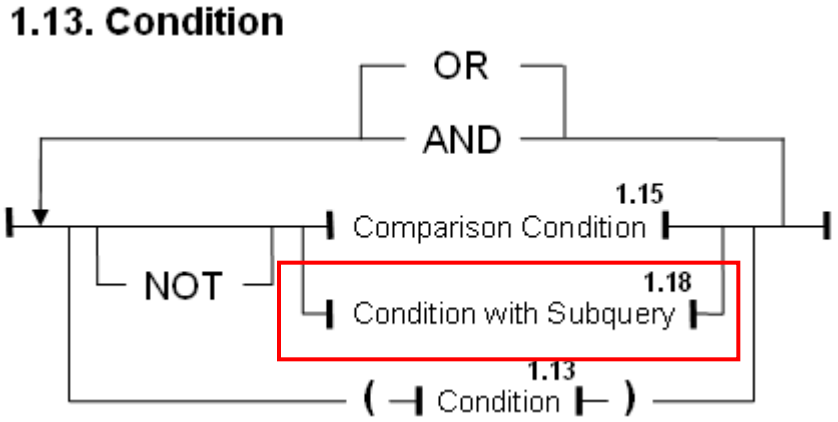
```
SELECT * FROM stud
  WHERE pbrSt = (SELECT pbr FROM mjesto
                 WHERE nazMjesto = 'Grad Split');
```

→ NULL

Podupiti u HAVING dijelu naredbe



- Podupiti u HAVING dijelu naredbe koriste se na jednak način kao u WHERE dijelu naredbe



Podupiti u HAVING dijelu naredbe

- Primjer:

dvorana

oznDv	kapacitet
D1	150
D2	200
A201	80

raspored

predmet	oznGr	brojSt
Matematika	M1	200
Matematika	M2	50
Matematika	M3	50
Fizika	F1	250
Fizika	F2	150
Elektronika	E1	50
Elektronika	E2	50
Elektronika	E3	100
Elektronika	E4	150

- Ispisati nazive predmeta za koje se u "D dvoranama" predavanja mogu održati istovremeno za sve grupe. Uz svaki takav predmet ispisati ukupni broj studenata na predmetu

```
SELECT predmet
     , SUM(brojSt) AS ukupnoStud
  FROM raspored
  GROUP BY predmet
  HAVING SUM(brojSt) <=
    (SELECT SUM(kapacitet)
       FROM dvorana
       WHERE oznDv LIKE 'D%');
```

350

predmet	ukupnoStud
Matematika	300
Elektronika	350

Korelirani podupit (*Correlated subquery*)

- Ako se u podupitu koriste atributi iz vanjskog upita, za podupit i vanjski upit se kaže da su korelirani (*correlated*)
- Za podupit koji je koreliran s vanjskim upitom koristi se naziv korelirani podupit (*correlated subquery*)
- Najčešće se korelirani podupit mora (fizički) izvršiti po jedanput za svaku n-torku iz vanjskog upita

- Sljedeći podupit **nije** korelirani podupit: u podupitu se ne koriste atributi vanjskog upita. Rezultat podupita ne ovisi o vrijednostima n-torki iz vanjskog podupita, stoga se taj podupit (fizički) treba izvršiti samo jednom tijekom jednog obavljanja vanjskog upita

```
SELECT *
  FROM vozilo
  WHERE nosivost > (SELECT MAX(tezina)
                    FROM teret);
```

→ 1800

Korelirani podupit (*Correlated subquery*)

- ispisati podatke o strojevima koji su ukupno korišteni više od dopuštenog broja radnih sati

stroj

oznStr	dopBrSati
S1	1000
S2	1500
S3	500

radStroja

oznStr	godina	brSatiRada
S1	2016	700
S1	2017	100
S1	2018	300
S2	2016	700
S2	2017	500
S3	2019	600

```
SELECT oznStr, dopBrSati
  FROM stroj
  WHERE dopBrSati <
    (SELECT SUM(brSatiRada)
       FROM radStroja
       WHERE oznStr = stroj.oznStr);
```

oznStr	dopBrSati
S1	1000
S3	500

- korelirani podupit: rezultat podupita ovisi o vrijednostima atributa vanjskog upita - za svaku n-torku vanjskog upita dobiva se drugačiji rezultat podupita

Korelirani podupit (*Correlated subquery*)

```
SELECT oznStr, dopBrSati
  FROM stroj
  WHERE dopBrSati <
    (SELECT SUM(brSatiRada)
       FROM radStroja
       WHERE radStroja.oznStr = stroj.oznStr);
```

radStroja

oznStr	godina	brSatiRada
S1	2016	700
S1	2017	100
S1	2018	300
S2	2016	700
S2	2017	500
S3	2019	600

- upit se (logički promatrano) obavlja na sljedeći način:
  - vanjski upit uzima jednu n-torku iz relacije stroj. Na temelju sadržaja te n-torke i sadržaja relacije radStroja, u podupitu se izračunava suma sati rada dotičnog stroja. Ukoliko je uvjet usporedbe zadovoljen, testirana n-torka se pojavljuje u rezultatu
  - postupak se ponavlja za svaku n-torku relacije stroj

oznStr	dopBrSati
S1	1000
S2	1500
S3	500

1000 < (SELECT SUM ... WHERE oznStr = 'S1'⇒1100) → *true*
1500 < (SELECT SUM ... WHERE oznStr = 'S2'⇒1200) → *false*
500 < (SELECT SUM ... WHERE oznStr = 'S3'⇒600) → *true*

oznStr	dopBrSati
S1	1000
S3	500

Korelirani podupit (*Correlated subquery*)

- Primjer: korelirani podupit u listi za selekciju
- uz svaki stroj koji je korišten više od dopuštenog broja sati, ispisati broj sati korištenja stroja

```
SELECT oznStr
     , dopBrSati
     , (SELECT SUM(brSatiRada)
          FROM radStroja
          WHERE radStroja.oznStr = stroj.oznStr) AS koristenSati
  FROM stroj
  WHERE dopBrSati <
    (SELECT SUM(brSatiRada)
       FROM radStroja
       WHERE radStroja.oznStr = stroj.oznStr);
```

oznStr	dopBrSati	koristenSati
S1	1000	1100
S3	500	600

Korelirani podupit (*Correlated subquery*)

- Rješenje istog problema bez korištenja podupita:

```
SELECT stroj.oznStr
     , dopBrSati
     , SUM(brSatiRada) AS koristenSati
  FROM stroj
       JOIN radStroja
         ON stroj.oznStr = radStroja.oznStr
  GROUP BY stroj.oznStr, dopBrSati
  HAVING dopBrSati < SUM(brSatiRada);
```

oznStr	dopBrSati	koristenSati
S1	1000	1100
S3	500	600

Korelirani podupit (*Correlated subquery*)

- Primjer: korelirani podupit u HAVING dijelu naredbe

ispit

mbr	predmet	akGod	ocj
100	Matematika	2018	2
101	Matematika	2018	3
102	Matematika	2018	4
100	Fizika	2018	2
101	Fizika	2018	5
100	Elektronika	2018	3
101	Elektronika	2018	3
110	Matematika	2019	3
111	Matematika	2019	5
110	Fizika	2019	3
111	Fizika	2019	3
112	Fizika	2019	3
113	Fizika	2019	2
111	Elektronika	2019	5
112	Elektronika	2019	4

- ispisati predmete čija je prosječna ocjena za 2019. godinu veća od prosječne ocjene tog istog predmeta za 2018. godinu

```
SELECT predmet
  FROM ispit AS ispit2019
  WHERE akGod = 2019
  GROUP BY predmet
  HAVING AVG(ocj) >
    (SELECT AVG(ocj)
       FROM ispit
       WHERE predmet = ispit2019.predmet
         AND ispit.akGod = 2018);
```

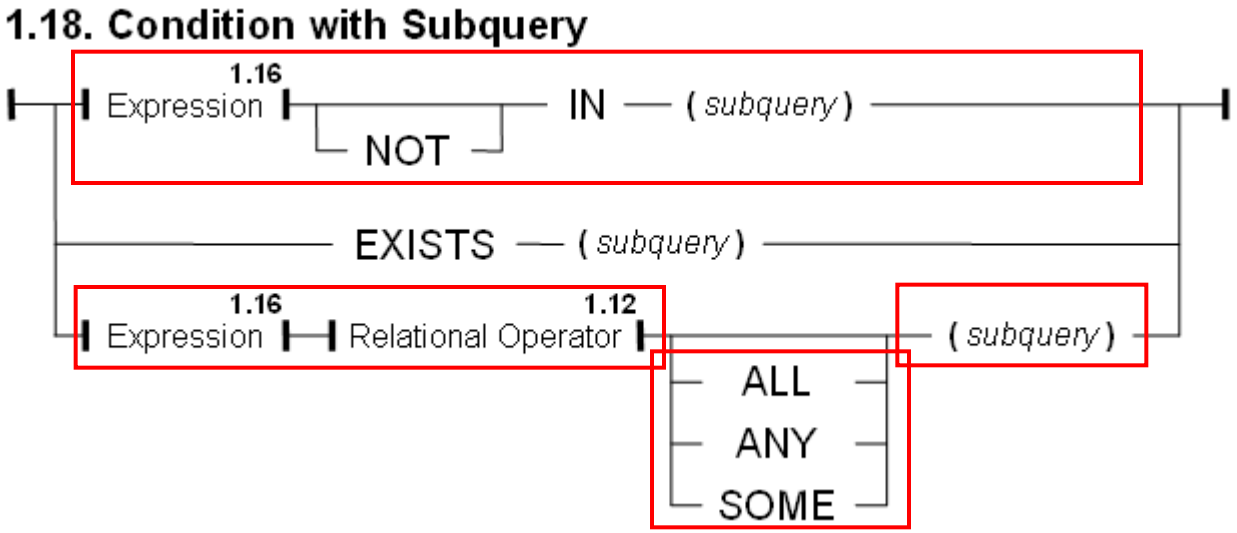
predmet
Matematika
Elektronika

Korištenje atributa vanjskog upita u podupitu

- u podupitu se mogu koristiti atributi iz vanjskog upita (obratno ne vrijedi)
- ukoliko se imena atributa (relacija) vanjskog upita podudaraju s imenima atributa (relacija) podupita:
  - ime atributa (relacije) navedeno u podupitu se odnosi na ime atributa (relacije) iz podupita
  - ime atributa (relacije) navedeno u vanjskom upitu se odnosi na ime atributa (relacije) vanjskog upita
- ukoliko je potrebno razriješiti dvosmislenost (npr. ista relacija se koristi u FROM dijelu vanjskog upita i FROM dijelu podupita, a u podupitu se koriste atributi relacije iz vanjskog upita), dovoljno je preimenovati relaciju u vanjskom upitu ili u podupitu
  - prethodni primjer ilustrira takav slučaj

Jednostupčani podupit (*Single-column subquery*)

- Rezultat jednostupčanog podupita je relacija stupnja jedan, s (moguće) više n-torki
  - također je dopušteno da jednostupčani podupit vrati jednu ili niti jednu n-torku
- jednostupčani podupiti se koriste u WHERE dijelu ili HAVING dijelu vanjskog upita
- jednostupčani podupiti se ne koriste u listi za selekciju



Jednostupčani podupit (*Single-column subquery*)

- Ispisati podatke o studentima čije je prezime različito od svih prezimena nastavnika

stud

mbr	ime	prezime
100	Ivan	Horvat
101	Ana	Kolar
102	Marko	Novak

nastavnik

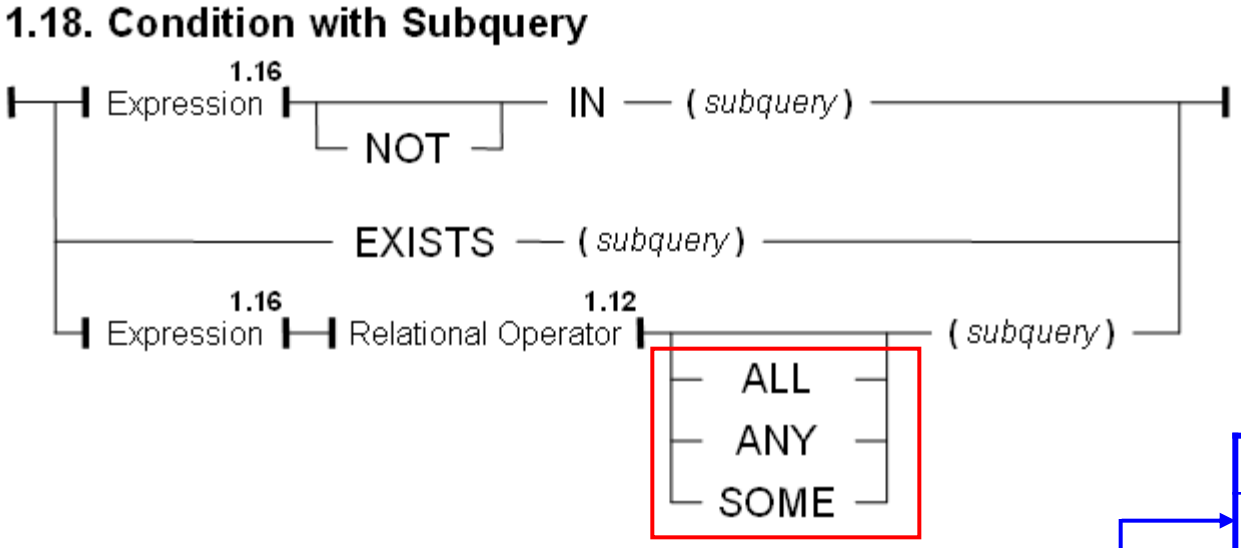
jmbg	prezime
12345	Kolar
23456	Ban
34567	Pernar

```
SELECT *
  FROM stud
  WHERE prez <> (SELECT prez
                   FROM nastavnik);
```

Kolar
Ban
Pernar

- Ovako napisan podupit nije ispravan - sustav će dojaviti pogrešku jer pomoću relacijskog operatora <> pokušavamo usporediti skalarnu vrijednost i rezultat podupita koji sadrži tri vrijednosti
- Potrebno je koristiti oblike usporedbe s IN, ALL, ANY, SOME

Jednostupčani podupit (*Single-column subquery*)




```
SELECT *
  FROM stud
  WHERE prez <> ALL (SELECT prez
                     FROM nastavnik);
```

Kolar
Ban
Pernar

mbr	ime	prez
100	Ivan	Horvat
102	Marko	Novak

- uvjet selekcije će biti zadovoljen za one n-torke iz relacije stud čija je vrijednost atributa prez različita od vrijednosti svih članova (multi)skupa dobivenog obavljanjem podupita

Jednostupčani podupit (*Single-column subquery*)

- izraz { **<** | **<=** | **=** | **<>** | **>** | **>=** } **ALL** (podupit)
  - *true* ako je izraz { **<** | **<=** | **=** | **<>** | **>** | **>=** } od svih vrijednosti dobivenih podupitom

- izraz { **<** | **<=** | **=** | **<>** | **>** | **>=** } **SOME** (podupit)
  - *true* ako je izraz { **<** | **<=** | **=** | **<>** | **>** | **>=** } od barem jedne vrijednosti dobivene podupitom

- **ANY** je sinonim za **SOME**

Jednostupčani podupit (*Single-column subquery*)

- Ispisati podatke o dvoranama čiji je kapacitet veći od broja studenata u barem jednoj od grupa

dvorana

oznDv	kapacitet
D1	150
D2	120
A201	80

grupa

oznGr	brojSt
M1	100
F1	170
E4	150

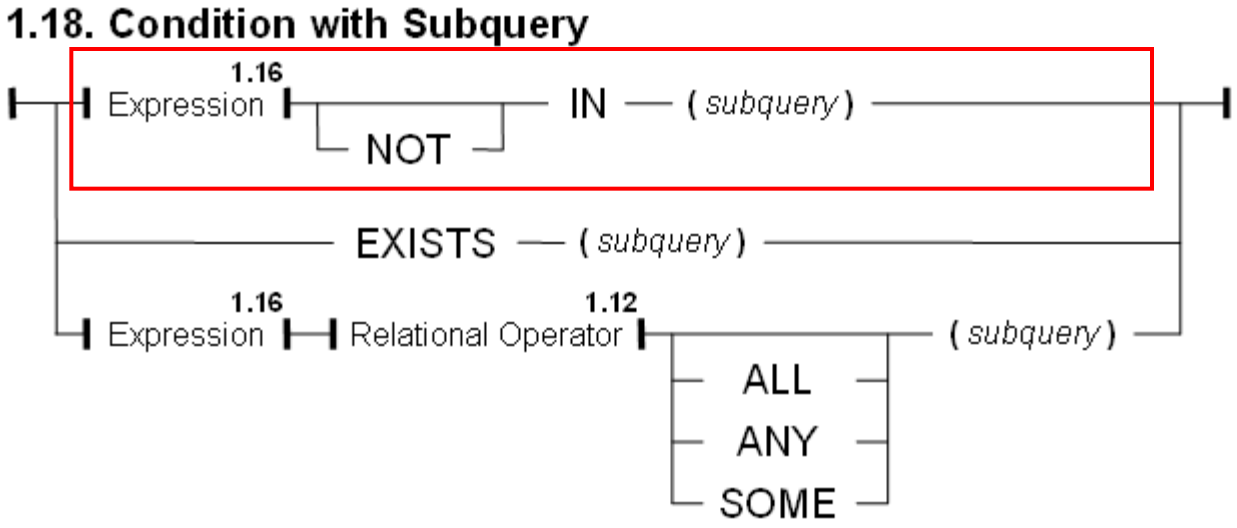
```
SELECT *
  FROM dvorana
  WHERE kapacitet > SOME (SELECT brojSt
                          FROM grupa);
```

100
170
150

oznDv	kapacitet
D1	150
D2	120

- za vježbu riješiti bez podupita (spajanje, selekcija, projekcija)

Jednostupčani podupit (*Single-column subquery*)



- izraz **IN** (podupit)
  - *true* ako se u (multi)skupu vrijednosti dobivenih podupitom nalazi barem jedan element jednak vrijednosti izraza
  - ekvivalentno sa: izraz = SOME (podupit)
- izraz **NOT IN** (podupit)
  - *true* ako se u (multi)skupu vrijednosti dobivenih podupitom **ne** nalazi niti jedan element jednak vrijednosti izraza
  - ekvivalentno sa: izraz <> ALL (podupit)

Jednostupčani podupit (*Single-column subquery*)

- Ispisati podatke o studentima koji su bilo koji predmet položili tijekom akademske godine 2018.

stud

mbr	prez
100	Horvat
101	Kolar
102	Novak
103	Ban

ispit

mbr	predmet	akGod	ocjena
100	Matematika	2019	1
100	Elektronika	2018	2
100	Fizika	2018	3
102	Elektronika	2018	2
102	Fizika	2019	5
103	Elektronika	2018	NULL
103	Matematika	2020	4

```
SELECT *
  FROM stud
  WHERE mbr IN
     (SELECT mbr
        FROM ispit
        WHERE akGod = 2018
          AND ocjena > 1);
```

100
100
102

mbr	prez
100	Horvat
102	Novak

- za vježbu riješiti bez podupita (spajanje, selekcija, projekcija)

Jednostupčani podupit (*Single-column subquery*)

- Ispisati podatke o studentima koji nisu položili niti jedan predmet tijekom ak. godine 2018.

stud

mbr	prez
100	Horvat
101	Kolar
102	Novak
103	Ban

ispit

mbr	predmet	akGod	ocjena
100	Matematika	2019	1
100	Elektronika	2018	2
100	Fizika	2018	3
102	Elektronika	2018	2
102	Fizika	2019	5
103	Elektronika	2018	NULL
103	Matematika	2020	4

```
SELECT *
  FROM stud
  WHERE mbr NOT IN
     (SELECT mbr
        FROM ispit
        WHERE akGod = 2018
          AND ocjena > 1);
```

100
100
102

mbr	prez
101	Horvat
103	Ban

- može li se riješiti bez podupita?

NULL vrijednosti i jednostupčani podupiti

- izraz *relOp* **ALL** (podupit)
  - ako je podupitom dobiven skup vrijednosti { x_1, x_2, ..., x_n }, efektivno se uvjet izračunava na sljedeći način:
    - izraz *relOp* x_1 AND izraz *relOp* x_2 AND ... AND izraz *relOp* x_n
- izraz *relOp* **SOME** (podupit)
  - ako je podupitom dobiven skup vrijednosti { x_1, x_2, ..., x_n }, efektivno se uvjet izračunava na sljedeći način:
    - izraz *relOp* x_1 OR izraz *relOp* x_2 OR ... OR izraz *relOp* x_n
- izraz **IN** (podupit)
  - ekvivalentno sa: izraz = **SOME** (podupit)
- izraz **NOT IN** (podupit)
  - ekvivalentno sa: izraz <> **ALL** (podupit)

Primjeri: podupiti i NULL vrijednosti

- naročitu pažnju pri korištenju podupita čiji rezultat može sadržavati NULL vrijednosti treba obratiti na uvjete selekcije oblika:

```
WHERE expression relationalOperator ALL (subquery)
```

```
WHERE expression NOT IN (subquery)
```

ukoliko se u rezultatu ovakvih podupita nalazi makar jedna NULL vrijednost, rezultat izračunavanja uvjeta selekcije nikad neće biti *true*

Primjeri: podupiti i NULL vrijednosti

uplata

rbrUpl	iznosUpl
1	50
2	NULL
3	200
4	120

isplata

rbrIspl	iznosIspl
1	80
2	100
3	NULL
4	150

```
SELECT * FROM uplata
   WHERE iznosUpl >
   SOME (SELECT iznosIspl FROM isplata);
```

80
100
NULL
150

rbrUpl	iznosUpl
1	50
2	NULL
3	200
4	120

50 > 80 OR 50 > 100 OR 50 > NULL OR 50 > 150 → *unknown*

NULL > 80 OR NULL > 100 OR NULL > NULL OR NULL > 150 → *unknown*

200 > 80 OR 200 > 100 OR 200 > NULL OR 200 > 150 → *true*

120 > 80 OR 120 > 100 OR 120 > NULL OR 120 > 150 → *true*

rbrUpl	iznosUpl
3	200
4	120

Primjeri: podupiti i NULL vrijednosti

uplata

rbrUpl	iznosUpl
1	50
2	NULL
3	200
4	120

isplata

rbrIspl	iznosIspl
1	80
2	100
3	NULL
4	150

```
SELECT * FROM uplata
   WHERE iznosUpl >
   ALL (SELECT iznosIspl FROM isplata);
```

80
100
NULL
150

rbrUpl	iznosUpl
1	50
2	NULL
3	200
4	120

50 > 80 AND 50 > 100 AND 50 > NULL AND 50 > 150 → *false*

NULL > 80 AND NULL > 100 AND NULL > NULL AND NULL > 150 → *unknown*

200 > 80 AND 200 > 100 AND 200 > NULL AND 200 > 150 → *unknown*

120 > 80 AND 120 > 100 AND 120 > NULL AND 120 > 150 → *false*

rbrUpl	iznosUpl

Primjeri: podupiti i NULL vrijednosti

uplata

rbrUpl	iznosUpl
1	50
2	NULL
3	100
4	80

isplata

rbrIspl	iznosIspl
1	80
2	100
3	NULL
4	150

```
SELECT * FROM uplata
   WHERE iznosUpl IN
      (SELECT iznosIspl FROM isplata);
```

80
100
NULL
150

rbrUpl	iznosUpl
1	50
2	NULL
3	100
4	80

50 = 80 OR 50 = 100 OR 50 = NULL OR 50 = 150 → *unknown*

NULL = 80 OR NULL = 100 OR NULL = NULL OR NULL = 150 → *unknown*

100 = 80 OR 100 = 100 OR 100 = NULL OR 100 = 150 → *true*

80 = 80 OR 80 = 100 OR 80 = NULL OR 80 = 150 → *true*

rbrUpl	iznosUpl
3	100
4	80

Primjeri: podupiti i NULL vrijednosti

uplata

rbrUpl	iznosUpl
1	50
2	NULL
3	100
4	80

isplata

rbrIspl	iznosIspl
1	80
2	100
3	NULL
4	150

```
SELECT * FROM uplata
   WHERE iznosUpl NOT IN
      (SELECT iznosIspl FROM isplata);
```

80
100
NULL
150

rbrUpl	iznosUpl
1	50
2	NULL
3	100
4	80

50 ≠ 80 AND 50 ≠ 100 AND 50 ≠ NULL AND 50 ≠ 150 → *unknown*

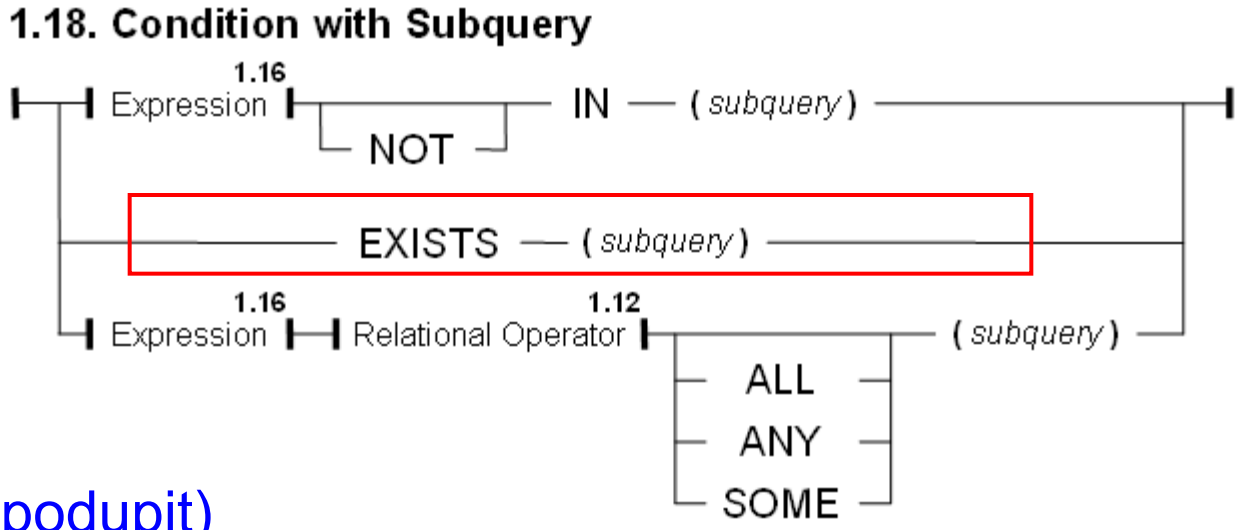
NULL ≠ 80 AND NULL ≠ 100 AND NULL ≠ NULL AND NULL ≠ 150 → *unknown*

100 ≠ 80 AND 100 ≠ 100 AND 100 ≠ NULL AND 100 ≠ 150 → *false*

80 ≠ 80 AND 80 ≠ 100 AND 80 ≠ NULL AND 80 ≠ 150 → *false*

rbrUpl	iznosUpl

Operator EXISTS



- **EXISTS** (podupit)
  - *true* ako rezultat podupita sadrži barem jednu n-torku (bilo kakvu). Pri tome nije važno koliko u dobivenoj n-torci ili n-torkama ima atributa (podupit ne mora biti jednostupčan) niti koje su vrijednosti njihovih atributa
- **NOT EXISTS** (podupit)
  - *true* ako rezultat podupita ne sadrži niti jednu n-torku
- na rezultat vanjskog upita ne utječe eventualna pojava NULL vrijednosti u rezultatu podupita

Operator EXISTS

- Ispisati podatke o studentima koji u akademskoj godini u kojoj su upisali studij nisu položili niti jedan ispit

stud

mbr	prez	akGodUpis
100	Horvat	2018
101	Kolar	2018
102	Novak	2019
103	Ban	2018

ispit

mbr	predmet	akGod	ocjena
100	Matematika	2018	1
100	Fizika	2019	2
100	Elektronika	2020	4
100	Matematika	2019	3
101	Matematika	2018	2
101	Fizika	2018	5
102	Matematika	2019	4

```
SELECT *
  FROM stud
  WHERE NOT EXISTS
    (SELECT * FROM ispit
       WHERE ispit.mbr = stud.mbr
         AND akGod = akGodUpis
         AND ocjena > 1 );
```

ovdje se npr. moglo napisati samo mbr: dobio bi se jednak rezultat

mbr	prez	akGodUpis
100	Horvat	2018
103	Ban	2018

Podupiti u HAVING dijelu naredbe

- Svi prikazani oblici podupita mogu se također koristiti i u HAVING dijelu naredbe
- Primjer: ispisati naziv(e) predmeta s najvećim prosjekom

ispit

mbr	predmet	ocjena
100	Matematika	4
101	Matematika	5
102	Matematika	3
100	Fizika	3
101	Fizika	4
101	Elektronika	5
102	Elektronika	3

```
SELECT predmet
  FROM ispit
  GROUP BY predmet
  HAVING AVG(ocjena) >= ALL
    (SELECT AVG(ocjena)
       FROM ispit
       GROUP BY predmet);
```

4.0
3.5
4.0

predmet
Matematika
Elektronika

Nepotrebno korištenje podupita

- Ispisati podatke o studentima i nazivima mjesta u kojima stanuju. U rezultatu trebaju biti i studenti čije je mjesto stanovanja nepoznato
- **Vrlo loše** rješenje:

```
SELECT student.*
     , (SELECT nazMjesto FROM mjesto
          WHERE pbr = pbrStan) AS nazMjesto
  FROM student;
```

- **Ispravno** rješenje:

```
SELECT student.*, nazMjesto
  FROM student
       LEFT OUTER JOIN mjesto
          ON pbrStan = pbr;
```

- **Zašto LEFT OUTER JOIN?**

Kada moramo koristiti podupite u SELECT listi

- U situacijama kada u SELECT listu treba uključiti izraz čiji su uvjeti ili način grupiranja različiti od uvjeta u vanjskom upitu.
- Primjer (baza studAdmin): za svaki predmet ispisati šifru, ukupan broj studenata koji su ga do sada položili te ukupan broj upisa predmeta:

```
SELECT sifPredmet,
       (SELECT COUNT(*)
          FROM upisanpredmet up
            WHERE up.sifPredmet = upisanpredmet.sifPredmet
            AND ocjena > 1) položili,
        COUNT(*) upisan
  FROM upisanpredmet
GROUP BY sifPredmet;
```

SQL naredbe za izmjenu sadržaja relacije

INSERT

DELETE

UPDATE

INSERT

```
CREATE TABLE mjesto (
   pbr            INTEGER NOT NULL
 , nazMjesto      VARCHAR(30) NOT NULL
 , sifZup         SMALLINT
);
```

mjesto

pbr	nazMjesto	sifZup

```
INSERT INTO mjesto
   VALUES (42000, 'Varaždin', 7);
```

```
INSERT INTO mjesto
    (pbr, sifZup, nazMjesto)
   VALUES (52100, 4, 'Pula');
```

```
INSERT INTO mjesto
          (pbr, nazMjesto)
   VALUES (42230, 'Ludbreg');
```

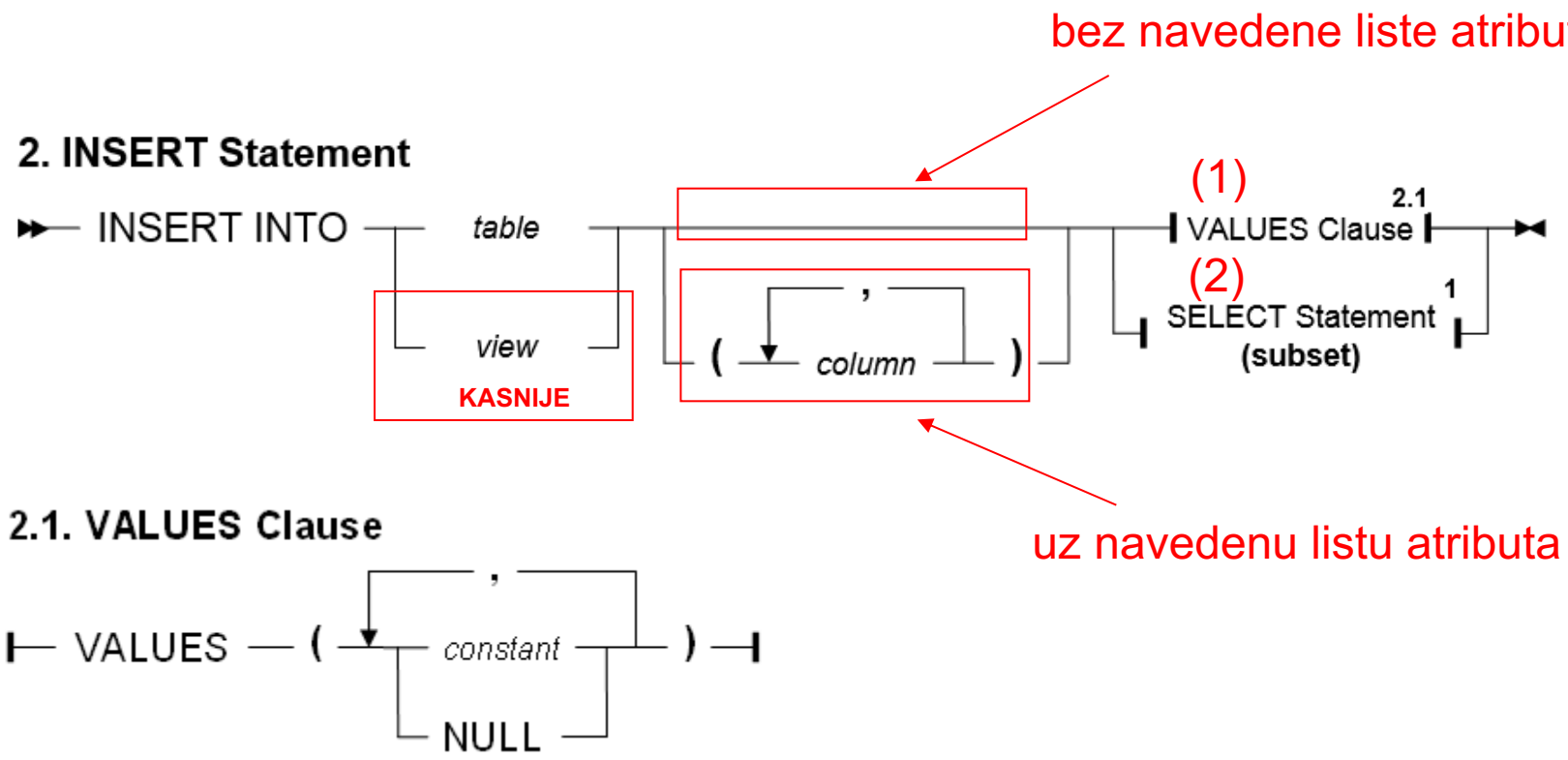
```
INSERT INTO mjesto
   VALUES (10000, 'Zagreb', NULL);
```

mjesto

pbr	nazMjesto	sifZup
42000	Varaždin	7
52100	Pula	4
42230	Ludbreg	NULL
10000	Zagreb	NULL

INSERT

- INSERT naredba se koristi za unos jedne n-torke (1) ili skupa n-torki (2) u relaciju *table*



INSERT (1), bez navedene liste atributa – antiprimjer!

- U relaciju se upisuje jedna n-torka pri čemu **vrijednosti svih** atributa n-torke **moraju biti navedene redoslijedom** kojim su atributi navedeni u CREATE TABLE naredbi kojom je relacija definirana

```
CREATE TABLE stud (
  mbr      INTEGER
, ime      VARCHAR(30)
, prez     VARCHAR(50)
, stipend  VARCHAR(1) DEFAULT 'N'
, pbrStan  INTEGER DEFAULT 10000
 );
```

stud

mbr	ime	prez	stipend	pbrStan

Znači: ako se prilikom unosa nove n-torke ne navede vrijednost za atribut stipend, sustav će kao vrijednost tog atributa postaviti vrijednost N

```
INSERT INTO stud VALUES (
   100
 , 'Ivan'
 , 'Horvat'
 , 'N'
 , NULL
 );
```

stud

mbr	ime	prez	stipend	pbrStan
100	Ivan	Horvat	N	NULL

INSERT (1), bez navedene liste atributa

- Opisani oblik INSERT naredbe ima neke nedostatke:
  - u slučaju kada se u relaciju upisuje n-torka čiji relativno veliki broj atributa treba postaviti na NULL vrijednost ili pretpostavljenu vrijednost (*default value*)
    - ipak se moraju navesti vrijednosti **svih** atributa
  - u slučaju kada se relacijska shema promijeni (npr. promijeni se "redoslijed" atributa)
    - budući da vrijednosti atributa moraju biti navedene redoslijedom atributa u CREATE TABLE naredbi, INSERT naredbe koje su napisane prije promjene relacijske sheme više neće biti ispravne

- zbog navedenih nedostataka, preporuča se korištenje oblika INSERT naredbe s navedenom listom atributa

INSERT (1), uz navedenu listu atributa

- U prvom dijelu naredbe navode se imena atributa (i njihov redoslijed) čije će vrijednosti (u odgovarajućem redoslijedu) biti navedene u drugom dijelu naredbe
  - atributi čije vrijednosti nisu navedene u INSERT naredbi, postavljaju se na pretpostavljenu (*default*) vrijednost (ukoliko je takva definirana u CREATE TABLE naredbi) ili na NULL vrijednost

```
INSERT INTO stud (
   prez
 , mbr
 , pbrStan)
  VALUES (
     'Kolar'
   , 101
   , 10000);
```

stud

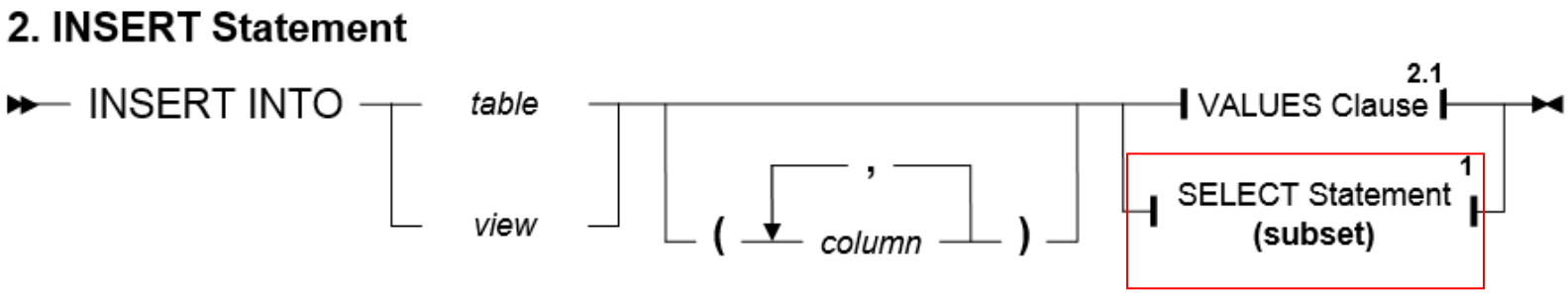
mbr	ime	prez	stipend	pbrStan
100	Ivan	Horvat	N	NULL

stud

mbr	ime	prez	stipend	pbrStan
100	Ivan	Horvat	N	NULL
101	NULL	Kolar	N	10000

ime? stipend?

INSERT (2)



- u relaciju *table* upisuju se n-torke dobivene dijelom INSERT naredbe koji je sličan SELECT naredbi - u sintaksnom dijagramu taj je dio naredbe označen sa *SELECT Statement* ***(subset)***

INSERT (2), bez navedene liste atributa

- u relaciju polozioFiz upisati podatke o studentima koji su položili predmet Fizika

stud

mbr	ime	prez	pbrSt
102	Ana	Novak	10000
105	Rudi	Kolar	21000
107	Jura	Horvat	41000
109	Tea	Ban	51000

ispit

mbr	predmet	ocjena
102	Elektronika	1
102	Matematika	3
105	Fizika	3
105	Matematika	4
107	Fizika	1
109	Fizika	5

polozioFiz

mbr	imeSt	prezSt

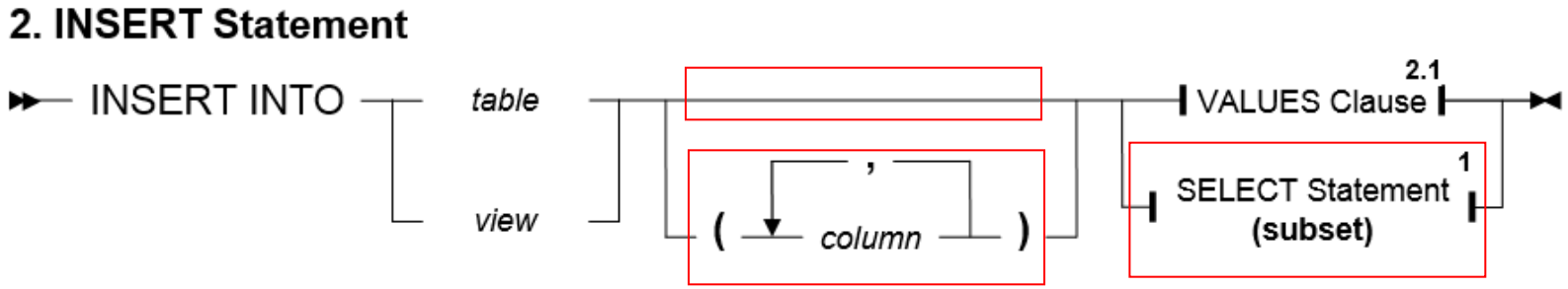
```
INSERT INTO polozioFiz
  SELECT stud.mbr, ime, prez
    FROM stud, ispit
    WHERE stud.mbr = ispit.mbr
      AND predmet = 'Fizika'
      AND ocjena > 1;
```

imena atributa u SELECT listi ne moraju odgovarati imenima atributa u relaciji polozioFiz

polozioFiz

mbr	imeSt	prezSt
105	Rudi	Kolar
109	Tea	Ban

INSERT (2)



- jednako kao kod oblika INSERT naredbe za unos jedne n-torke u relaciju
  - bez navedene liste imena atributa, broj i redoslijed atributa u n-torkama dobivenim obavljanjem SELECT dijela naredbe mora odgovarati broju i redoslijedu atributa u CREATE TABLE naredbi (prikazano u prethodnom primjeru)
  - uz navedenu listu imena atributa, atributi čije vrijednosti nisu navedene u INSERT naredbi, postavljaju se na pretpostavljenu (*default*) vrijednost (ukoliko je takva definirana u CREATE TABLE naredbi) ili na NULL vrijednost

INSERT i SERIAL

```
CREATE TABLE predmet (
   sifPred  SERIAL
 , nazPred  VARCHAR(100)
 , ECTSBod  NUMERIC(4,1));
```

predmet

sifPred	nazPred	ECTSBod

```
INSERT INTO predmet (nazPred, ECTSBod)
            VALUES ('Baze podataka', 6.0);
```

sifPred	nazPred	ECTSBod
1	Baze podataka	6.0

```
INSERT INTO predmet (nazPred, ECTSBod)
            VALUES ('Programiranje', 7.0);
INSERT INTO predmet (nazPred, ECTSBod)
            VALUES ('Badminton', 1.5);
```

sifPred	nazPred	ECTSBod
1	Baze podataka	6.0
2	Programiranje	7.0
3	Badminton	1.5

- SERIAL je autoinkrementalni INTEGER
  - raspon je: [1, 2147483647]
- uvijek se generira prvi sljedeći broj (interni generator o tome vodi računa)

DELETE

- brisanje n-torki iz relacije

```
DELETE FROM mjesto
   WHERE sifZup = 7;
```

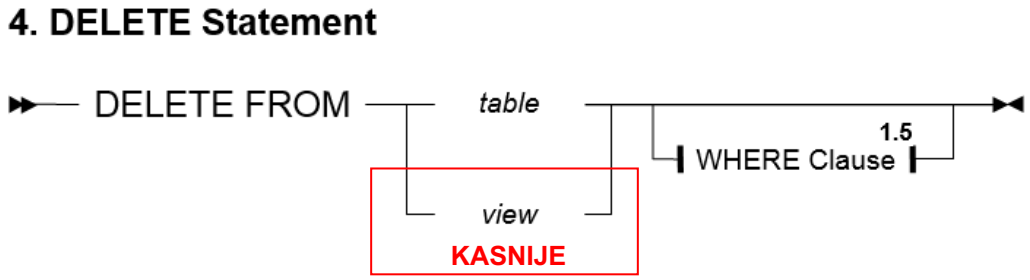
mjesto

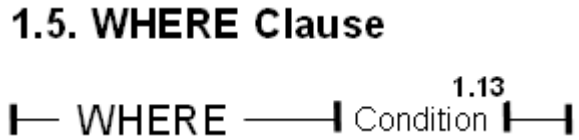
pbr	nazMjesto	sifZup
42230	Ludbreg	7
42000	VARAŽDIN	7
52100	Pula	4

mjesto

pbr	nazMjesto	sifZup
52100	Pula	4

DELETE





- DELETE naredba briše one n-torke relacije *table* za koje se uvjet naveden u WHERE dijelu naredbe izračuna kao *true*
  - ako se WHERE dio naredbe ne navede, iz relacije *table* se brišu sve n-torke
- u WHERE dijelu naredbe mogu se koristiti svi oblici uvjeta (*Condition*) koji se koriste u WHERE dijelu SELECT naredbe
- neki sustavi (npr. IBM Informix) ne dopuštaju korištenje relacije *table* u koreliranom podupitu, a neki (npr. PostgreSQL, SQL Server, Oracle) dopuštaju.

```
DELETE FROM mjesto
  WHERE pbr IN (SELECT pbr FROM mjesto m1
                WHERE m1.nazMjesto = mjesto.nazMjesto);
```

DELETE

- iz relacije polozioFiz obrisati n-torke onih studenata koji (sudeći prema podacima iz relacije ispit) nisu položili predmet Fizika

stud

mbr	ime	prez	pbrSt
102	Ana	Novak	10000
105	Rudi	Kolar	21000
107	Jura	Horvat	41000
109	Tea	Ban	51000

ispit

mbr	predmet	ocjena
102	Elektronika	1
102	Matematika	3
105	Fizika	3
105	Matematika	4
107	Fizika	1
109	Fizika	5

polozioFiz

mbr	imeSt	prezSt
102	Ana	Novak
105	Rudi	Kolar
107	Jura	Horvat
109	Tea	Ban
111	Ivan	Polak

```
DELETE FROM polozioFiz
  WHERE mbr NOT IN
    (SELECT mbr
       FROM ispit
       WHERE predmet = 'Fizika'
       AND ocjena > 1);
```

polozioFiz

mbr	imeSt	prezSt
105	Rudi	Kolar
109	Tea	Ban

INSERT, DELETE i SERIAL

```
CREATE TABLE predmet (
   sifPred  SERIAL
 , nazPred  VARCHAR(100)
 , ECTSBod  NUMERIC(4,1));
```

predmet

sifPred	nazPred	ECTSBod
1	Baze podataka	6.0
2	Programiranje	7.0
3	Badminton	1.5

```
DELETE FROM predmet
WHERE sifPred = 3;
INSERT INTO predmet (nazPred, ECTSBod)
              VALUES ('Badminton', 1.5);
```

predmet

sifPred	nazPred	ECTSBod
1	Baze podataka	6.0
2	Programiranje	7.0
4	Badminton	1.5

- Eventualni slobodni brojevi (identifikatori, šifre) nastali brisanjem se ne koriste ponovno

UPDATE

- Svim studentima, čija je ocjena manja od 5, ocjenu uvećati za 1, a broj bodova postaviti na NULL

ispit

mbrSt	brBod	ocjena
1001	200	5
1002	150	4
1003	130	3
1004	110	2

```
UPDATE ispit
  SET ocjena = ocjena + 1
    , brBod = NULL
  WHERE ocjena < 5;
```

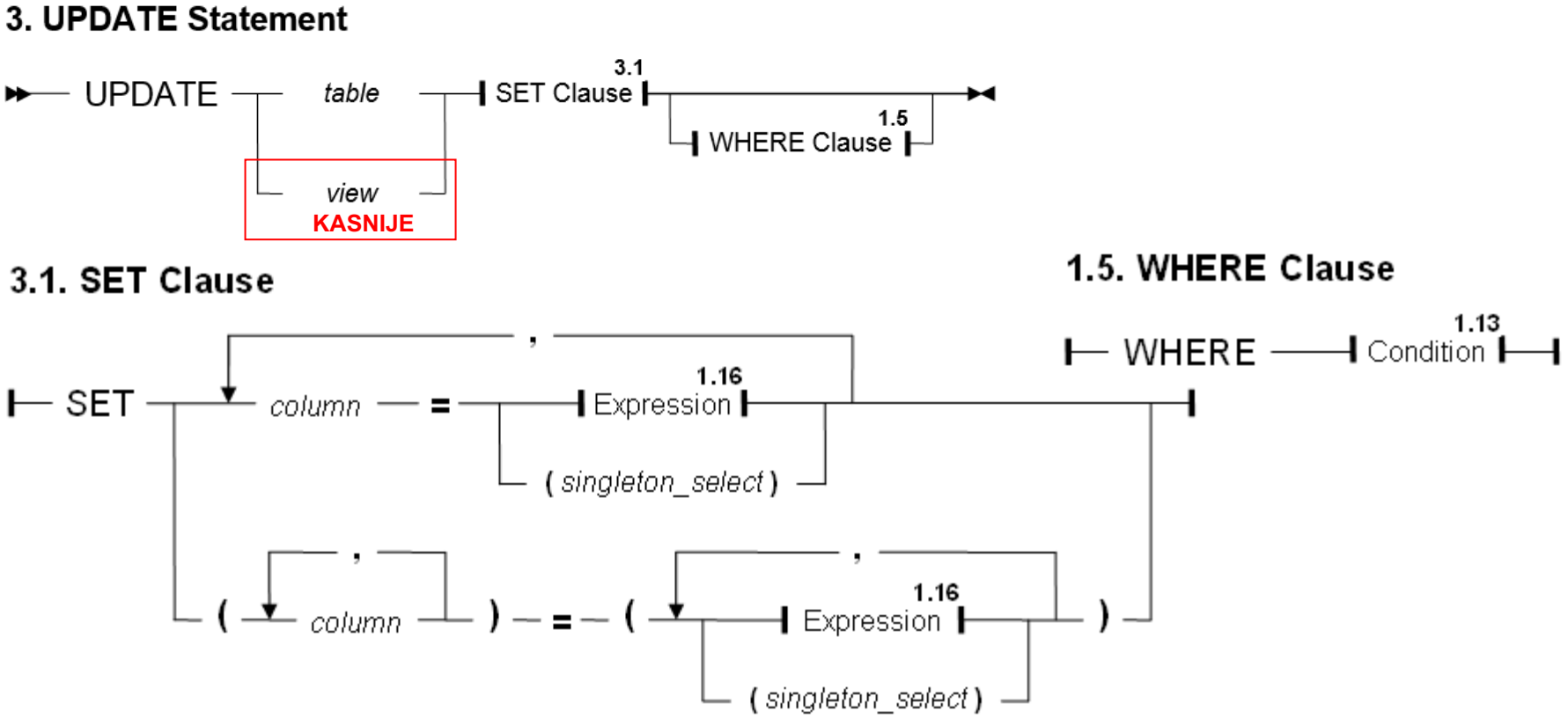
ili

```
UPDATE ispit
  SET (ocjena, brBod) = (ocjena + 1, NULL)
  WHERE ocjena < 5;
```

ispit

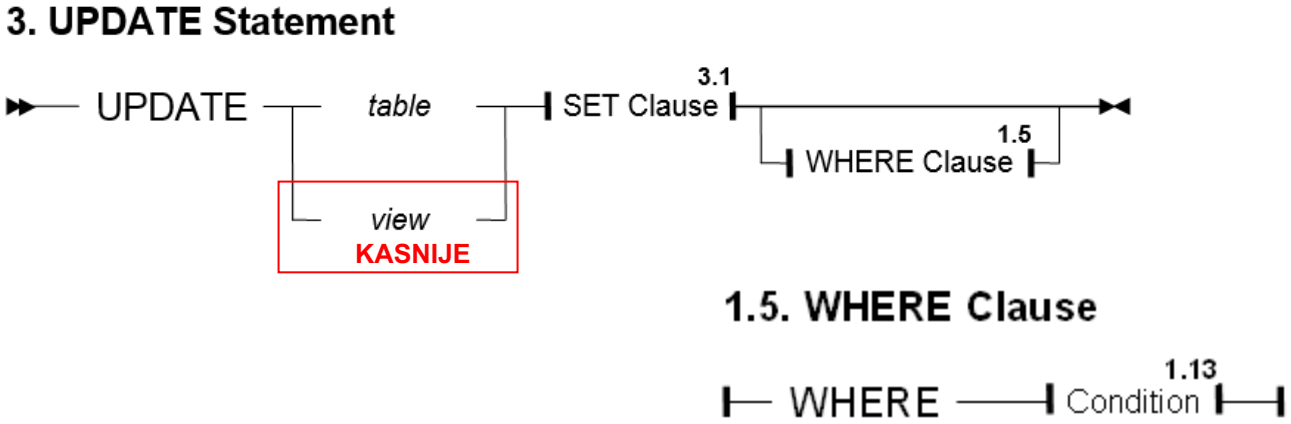
mbrSt	brBod	ocjena
1001	200	5
1002	NULL	5
1003	NULL	4
1004	NULL	3

UPDATE



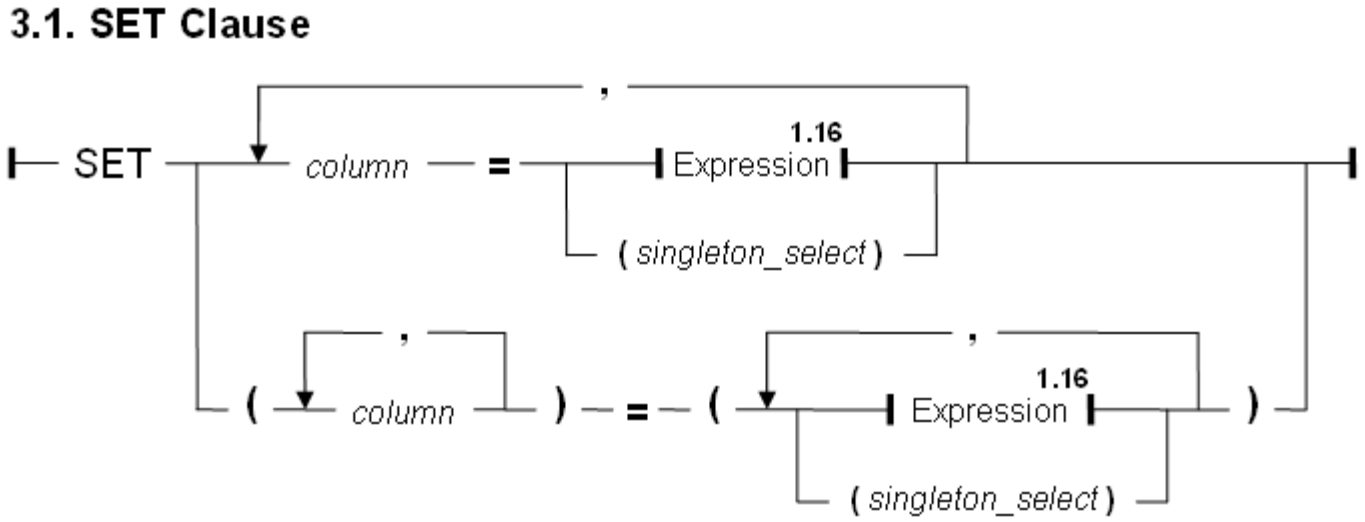
- UPDATE naredba mijenja vrijednosti atributa postojećih n-torki relacije *table*. U WHERE dijelu naredbe opisuje se koje n-torke će biti promijenjene; u SET dijelu naredbe opisuje se koji će atributi biti postavljeni na koje vrijednosti

UPDATE



- UPDATE naredba mijenja vrijednosti atributa onih n-torki relacije *table* za koje se uvjet naveden u WHERE dijelu naredbe izračuna kao *true*
  - ako se WHERE dio naredbe ne navede, mijenjaju se vrijednosti atributa u svim n-torkama relacije *table*
  - u WHERE dijelu naredbe mogu se koristiti svi oblici uvjeta (*Condition*) koji se koriste u WHERE dijelu SELECT naredbe,
  - neki sustavi (npr. IBM Informix) ne dopuštaju korištenje relacije *table* u koreliranom podupitu, a neki (npr. SQL Server, Oracle) dopuštaju.

UPDATE



- SET dio naredbe određuje nove vrijednosti atributa. Nova vrijednost atributa može biti definirana kao:
  - *Expression* - konstante, NULL, atributi iz relacije *table*, binarni i unarni operatori, funkcije, uvjetni izraz, zagrade
  - *singleton_select* - jednako kao skalarni podupit (korelirani ili nekorelirani)
    - relaciju *table* nije dopušteno koristiti u FROM dijelu

UPDATE

- Bodove na međuispitu uvećati za 10 bodova onim studentima koji time ne bi stekli ukupno (bodLab+bodMI) više od 100 bodova

bodovi

mbr	prez	bodLab	bodMI
101	Novak	30	55
102	Polak	NULL	20
103	Kolar	20	10
104	Ban	10	80
105	Horvat	50	49
106	Seljan	10	NULL

```
UPDATE bodovi
   SET bodMI = bodMI + 10
   WHERE bodLab + bodMI + 10 <= 100;
```

bodovi

mbr	prez	bodLab	bodMI
101	Novak	30	65
102	Polak	NULL	20
103	Kolar	20	20
104	Ban	10	90
105	Horvat	50	49
106	Seljan	10	NULL

UPDATE

- Bodove na međuispitu uvećati za 2 boda studentima koji time ne bi stekli više od 50 bodova za međuispit, bodove za laboratorij povećati za 3 boda onim studentima koji time ne bi stekli ukupno više od 40 bodova za laboratorij

```
UPDATE bodovi SET
  bodMI = CASE
            WHEN bodMI + 2 <= 50
              THEN bodMI + 2
            ELSE bodMI
          END
, bodLab = CASE
             WHEN bodLab + 3 <= 40
               THEN bodLab + 3
             ELSE bodLab
           END;
```

bodovi

mbr	prez	bodLab	bodMI
101	Novak	30	55
102	Polak	NULL	20
103	Kolar	20	10
104	Ban	10	80
105	Horvat	50	49
106	Seljan	10	NULL

bodovi

mbr	prez	bodLab	bodMI
101	Novak	33	55
102	Polak	NULL	22
103	Kolar	23	12
104	Ban	13	80
105	Horvat	50	49
106	Seljan	13	NULL

UPDATE

- U sintaksnom dijagramu za *SET Clause* može se vidjeti da postoje dva slična, jednako vrijedna oblika:
  - SET atribut1=vrijednost1, atribut2=vrijednost2, ...
  - SET (atribut1, atribut2, ...)=(vrijednost1, vrijednost2, ...)
- Prethodna UPDATE naredba se mogla napisati na sljedeći način:

```
UPDATE bodovi SET
   (bodMI, bodLab) =
   (CASE
      WHEN bodMI + 2 <= 50
        THEN bodMI + 2
      ELSE bodMI
    END
  , CASE
      WHEN bodLab + 3 <= 40
        THEN bodLab + 3
      ELSE bodLab
    END
   );
```

UPDATE

- U relaciji mjesto zamijeniti stare poštanske brojeve i nazive (samo onim mjestima za koje postoje opisani novi poštanski brojevi i nazivi)

mjesto

pbr	nazMjesto
41000	ZAGREB
51400	PAZIN
52000	PULA
54000	OSIJEK

konverzija

stariPbr	noviPbr	noviNaziv
51400	52000	Pazin
52000	52100	Pula
54000	31000	Osijek

```
UPDATE mjesto
  SET (pbr, nazMjesto) = (
     (SELECT noviPbr
       FROM konverzija
       WHERE konverzija.stariPbr = mjesto.pbr
     )
    , (SELECT noviNaziv
       FROM konverzija
       WHERE konverzija.stariPbr = mjesto.pbr
     )
  )
  WHERE pbr IN (SELECT stariPbr
                FROM konverzija);
```

mjesto

pbr	nazMjesto
41000	ZAGREB
52000	Pazin
52100	Pula
31000	Osijek

UPDATE

- Nastavnicima koji su na ispitima iz BP podijelili (prosječno) najveće ocjene, povećati plaću za 20000

nast

sifNast	prez	placa
101	Novak	52000
102	Kolar	55000
103	Horvat	48000
104	Ban	57000

ispitBP

mbrSt	ocjena	sifNast
1001	5	101
1002	4	101
1003	3	101
1004	2	102
1005	4	102
1006	5	103
1007	5	103
1008	2	103
1009	3	104

```
UPDATE nast SET placa = placa + 20000
  WHERE sifNast IN (
    SELECT sifNast
      FROM ispitBP
      GROUP BY sifNast
      HAVING AVG(ocjena) >= ALL
         (SELECT AVG(ocjena)
            FROM ispitBP
            GROUP BY sifNast)
    );
```

nast

sifNast	prez	placa
101	Novak	72000
102	Kolar	55000
103	Horvat	68000
104	Ban	57000