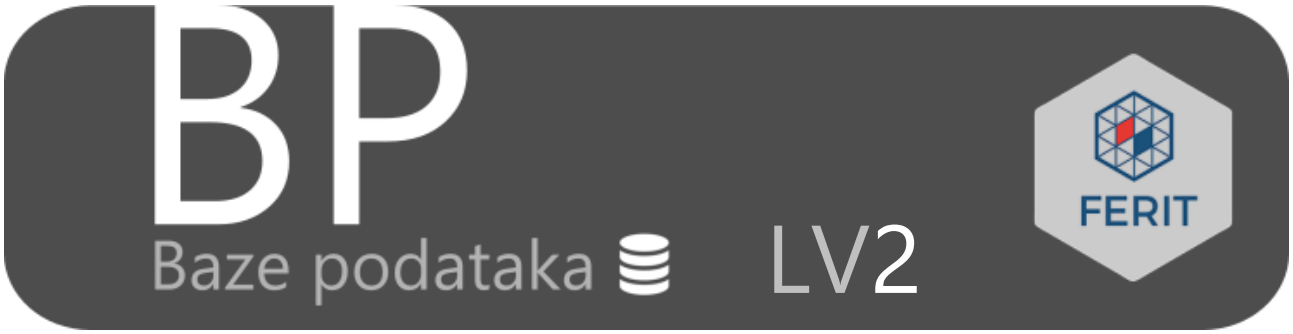

1. INSERT naredba

Naredba za upisivanje novih slogova u tablicu.

Sintaksa:

```
INSERT [ INTO]
{ table_name | view_name }
{ [ ( column_list ) ]
      { VALUES
            ( { DEFAULT | NULL | expression } [ ,...n] )
                  | derived_table
      }
}
| DEFAULT VALUES
```

column_list **-** Popis atributa odvojenih zarezom (ako ih je više). Ukoliko atribut tablice nije naveden u ovom popisu, mora postojati definirani način kako da mu se automatski pridijeli vrijednost, u suprotnom cijela operacija neće uspjeti. To je moguće ako za promatrani atribut vrijedi:

- Ima definiranu DEFAULT vrijednost
- Dozvoljava NULL vrijednost

Ukoliko lista atributa nije navedena, smatra se da će se upisivati vrijednost za svaki atribut tablice onim redom kojim su atributi navedeni prilikom kreiranja tablice

VALUES - Popis vrijednosti koji se upisuju u pojedine atribute. Redoslijed vrijednosti mora odgovarati onome iz popisa atributa

DEFAULT - Opcija koja nalaže da se kao vrijednost svakog od atributa uzme njegova DEFAULT vrijednost, odnosno ako ona nije definirana onda se upisuje NULL (ako je to dozvoljeno) ili trenutno vrijeme ako je kolona definirana kao tipa timestamp

Primjeri:

```
CREATE TABLE vozilo
  (
     reg_oznaka CHAR(10) PRIMARY KEY,
     marka      VARCHAR(20) NOT NULL,
     tip        VARCHAR(20) NOT NULL,
     km         SMALLINT,
     snaga      SMALLINT,
     proizveden DATETIME,
     ime_vl     NVARCHAR(20),
     prezime_vl NVARCHAR(20),
     prvi_vl    CHAR(2) DEFAULT 'DA',
     CONSTRAINT chk_prvi_vl CHECK (prvi_vl IN ('DA', 'NE'))
  );

INSERT INTO vozilo
  (
     reg_oznaka,
     marka,
     tip,
     prvi_vl
   )
VALUES
  (
     'OS-123 AB',
     'VW',
     'Golf',
     'NE'
   );

INSERT INTO vozilo
VALUES
  (
     'OS-123 AC',
     'FIAT',
     'UNO',
     '998',
     '33',
     '06/01/1997',
     '92000',
     'Ivana',
     'Šovagović'
   );

CREATE TABLE racunalo
  (
     inv_broj   SMALLINT PRIMARY KEY,
     tip        VARCHAR(20),
     ram        SMALLINT,
     hdd        SMALLINT,
     monitor    SMALLINT,
     mis        VARCHAR(20),
```

```
    tipkovnica VARCHAR(20),
    mreza      CHAR(2) DEFAULT 'da',
    modem      CHAR(2) DEFAULT 'da',
    grafika    VARCHAR(20),
    CONSTRAINT chk_mreza CHECK (mreza IN ('DA', 'NE')),
    CONSTRAINT chk_modem CHECK (modem IN ('DA', 'NE'))
  );

INSERT INTO racunalo
  (
    inv_broj,
    tip,
    ram,
    hdd,
    monitor,
    mis,
    tipkovnica
  )
VALUES
  (
    '123',
    'Fujitsu Siemens',
    '256',
    '20',
    '17',
    'Microsoft',
    'Microsoft'
  );

INSERT INTO racunalo
  (
    inv_broj,
    tip,
    modem,
    grafika
  )
VALUES
  (
    '321',
    'Koncar',
    'NE',
    'Nvidia'
  );
```

2. UPDATE naredba

Naredba za izmjenu podataka u postojećim slogovima tablice.

Sintaksa:

```
UPDATE { table_name | view_name }
SET    { column_name = { expression | DEFAULT | NULL } } [,...n]
{ { [ FROM { < table_source > [ ,...n ] ]
     [ WHERE < search_condition > ] } ] }
```

expression **-** Između ostaloga može biti i podupit koji kao rezultat vraća jednu vrijednost. Ta se vrijednost tada dodjeljuje promatranom atributu.

Primjeri:

```
UPDATE vozilo
SET    km = '100000';

UPDATE vozilo
SET    reg_oznaka = 'OS-321 AC',
       tip = 'Punto'
WHERE  reg_oznaka = 'OS-123 AB';

UPDATE vozilo
SET    km = km * 1.5;

UPDATE racunalo
SET    tip = 'IBM';
```

3. DELETE naredba

Naredba koja omogućava brisanje individualnih slogova ili više njih.

Sintaksa:

```
DELETE [ FROM ]
{ table_name | view_name }
[ WHERE { < search_condition > } ]
```

Primjeri:

```
DELETE FROM racunalo;

DELETE FROM vozilo
WHERE  tip LIKE 'P%';
```

4. SELECT naredba

SELECT je najčešće korištena SQL naredba, rabi se za dohvaćanje podataka iz baze, iz jedne ili više tablica pri čemu je rezultat ovog upita uvijek u tabličnom obliku. Dakle SELECT je naredba koja nam omogućuje:

- dohvaćanje slogova iz baze podataka prema proizvoljnim kriterijima,
- kombiniranje podataka iz različitih tablica te istovremeno vršenje projekcije i selekcije,
- objedinjavanje rezultata više upita,
- te korištenje kao podupit ili derivirana tablica unutar nekog drugog upita.

Osnovna sintaksa:

```
SELECT [ ALL | DISTINCT ] select_list
[ INTO new_table ]
[ FROM table_source ]
[ WHERE search_condition ]
[ GROUP BY group_by_expression ]
[ HAVING search_condition ]
[ ORDER BY order_expression [ ASC | DESC ] ]
```

Postoje također dodatne opcije UNION, EXCEPT i INTERSECT koje se mogu koristiti između SELECT upita za kombiniranje ili uspoređivanje njihovih rezultata.
SELECT naredba može imati više opcionalnih parametara odnosno klauzula (engl. clause):

- FROM - izbor tablice (ili tablica) koje se rabe za dohvaćanje podataka,
- WHERE - definicija uvjeta koji moraju biti zadovoljeni,
- GROUP BY - grupiranje rezultata,
- HAVING - odabir među grupama definiranim parametrom GROUP BY,
- ORDER BY - poredak rezultata.

4.1. SELECT parametar

Obavezan dio naredbe, specifikacija atributa koji se trebaju pojaviti kao rezultat upita.

Sintaksa:

```
SELECT [ ALL | DISTINCT ] < select_list >
```

ALL - dupli slogovi se mogu pojaviti u rezultatu, standardna opcija
DISTINCT - samo jedinstveni slogovi se mogu pojaviti u rezultatu

gdje je,

```
< select_list > ::=
{ *
| { table_name | view_name | table_alias }.*
```

```
| { column_name | expression }
[ [ AS ] column_alias ] } [ ,...n ]
}
```

'*' - Odabir svih atributa iz svih tablica ili pogleda navedenih u FROM stavci.
table_name* | *view_name* | *table_alias.* - Limit odabira samo na navedene objekte.
column_name - Specifikacija atributa, ako se istoimeni atributi pojavljuju u više objekata potrebno je navesti ime objekta.
expression - Izraz koji može biti atribut, konstanta, funkcija kao i kombinacija navedenog upotrebom nekog operatora, podupit.
column_alias - Pseudonim atributa za potrebe upita.

Primjeri:

Najjednostavniji oblik, uključuje sve atribute:
```
SELECT *
```

Popis pojedinačnih atributa:
```
SELECT broj,
       datum,
       kupac
```

Korištenje aliasa (pseudonima):
```
SELECT broj  AS 'broj racuna',
       datum AS 'datum racuna'
```

Korištenje pune reference atributa:
```
SELECT racuni.broj,
       datum,
       stavke.broj
```

4.2. FROM parametar

Specifikacija tablica i pogleda nad kojima se vrši upit. Moguće je korištenje pseudonima za tablice i poglede pomoću 'AS' operatora.

Sintaksa:

```
FROM   {<table_source>} [,...n ]
```

gdje je,

< table_source > ::=
table_name [[AS] *table_alias*] [,...n])]
| *view_name* [[AS] *table_alias*] [,...n])]
| *derived_table* [AS] *table_alias* [(*column_alias* [,...n])]| *< joined_table >*

Primjeri:

Popis tablica / pogleda:
```
FROM   racuni,
       stavke
```

Korištenje pseudonima za tablicu:
```
FROM   mobitel AS m,
       pozivi AS p
```

Spajanje tablica korištenjem operatora JOIN:
```
FROM   titles
       RIGHT OUTER JOIN publishers
                    ON titles.pub_id = publishers.pub_id
```

Korištenje derivirane tablice (dobivene drugom SELECT naredbom):
```
FROM   authors a,
       (SELECT title_id,
               au_id
        FROM   titleauthor) AS d1
```

Primjeri: (baza podataka 'stuslu')

```
SELECT ALL spol
FROM   student;

SELECT DISTINCT spol
FROM   student;

SELECT ALL sif_pred,
           dat_ispit
FROM   ispit;

SELECT DISTINCT sif_pred,
                dat_ispit
FROM   ispit;
```

4.3. WHERE parametar

Opcionalni dio SELECT naredbe, specificira uvjete prema kojem se vrši restrikcija prilikom odabiranja podataka iz baze.

Sintaksa:
```
     WHERE      < search_condition >
```
gdje je,

< *search_condition* > ::=
{ [NOT] < *predicate* > | (< *search_condition* >) }
[{ AND | OR } [NOT] { < *predicate* > | (< *search_condition* >) }]
} [,...n]

Nakon WHERE riječi slijedi logički izraz. Rezultat logičkog izraza je logička vrijednost – 'true', 'false' ili 'unknown'. Primjer osnovnog logičkog izraza je usporedba: boja = 'crvena', ovaj izraz vraća:

- true – ako atribut boja sadrži vrijednost 'crvena',
- false – ako atribut boja ne sadrži vrijednost 'crvena',
- unknown – ako atribut boja sadrži null vrijednost.

Operatore koje se mogu koristiti u izrazima WHERE parametra mogu se vidjeti u sljedećoj tablici:

Operator	**Opis**
=	Jednako
<>	Nije jednako. U nekim verzijama SQL-a, operator se koristi kao !=
>	Veće od
<	Manje od
>=	Veće ili jednako
<=	Manje ili jednako
BETWEEN	Između (zadanih) graničnih vrijednosti
LIKE	Pretraga za uzorkom (djelom znakovnog zapisa), koriste se '_' i '%' kao zamjenski znakovi
IN	Skup definiranih vrijednosti

Kombiniranje više uvjeta moguće je korištenjem logičkih operatora: AND, OR i NOT. Logički operatori koriste više logičkih izraza i vraćaju jedan logički rezultat ('true', 'false', 'unknown'). Zagrade se koriste za grupiranje izraza sa logičkim operatorima.

Primjeri:

Kombinacija dva uvjeta (barem jedan mora biti zadovoljen):
```
WHERE  stavke.broj_racuna = racun.broj_racuna
        OR datum_racuna > '06/21/2018'
```

Usporedba znakovnog zapisa:
```
WHERE  kupac LIKE 'a%'
```

Usporedba s NULL vrijednošću:
```
WHERE  cijena IS NOT NULL
       AND kolicina IS NULL
```

Usporedba s graničnim vrijednostima:

```
WHERE  datum BETWEEN '01/01/2000' AND '12/31/2000'
```

Korištenje podupita:

```
WHERE  pub_id IN (SELECT pub_id
                  FROM   titles
                  WHERE  type = 'business')
```

Usporedba prema zadanom popisu:

```
WHERE  broj_racuna IN ( 1001, 1002, 1003, 1005, 1100 )
```

Kod rada sa znakovnim zapisima, npr. prilikom usporedbe pomoću naredbe LIKE moguće je koristiti zamjenske znakove (tzv. wildcard characters). Standardno se koriste slijedeći znakovi:

- '_' vrijedi za jedan karakter
- '%' vrijedi za jedan ili više karaktera

Ukoliko želimo tražiti baš jedan od zamjenskih karaktera, tada treba definirati tzv. ESCAPE karakter koji se u tom slučaju navodi ispred traženog karaktera, npr:

```
WHERE  description LIKE 'gs_' ESCAPE 's'
```

Osnovni princip rada SELECT naredbe možemo vidjeti na primjeru u sljedećoj tablici.

Tablica "T"	Primjeri upita	Rezultat
C1 C2 1 a 2 b	SELECT * FROM T;	**C1 C2** 1 a 2 b
C1 C2 1 a 2 b	SELECT C1 FROM T;	**C1** 1 2
C1 C2 1 a 2 b	SELECT * FROM T WHERE C1 = 1;	**C1 C2** 1 a
C1 C2 1 a 2 b	SELECT * FROM T ORDER BY C1 DESC;	**C1 C2** 2 b 1 a

Detaljniji opis obrade jednog upita možemo vidjeti na sljedećem primjeru:

- Imamo tablicu *probna* koja izgleda:

rb	ime	grad
S1	Pierre	Paris
S2	John	London
S3	Mario	Rim

- Postavljamo upit:

```
SELECT ime
FROM   probna
WHERE  grad = 'Rim'
```

- Upit prvo pristupa slogovima tablice *probna*.

- Nakon toga filtrira slogove u kojima je vrijednost atributa *grad* jednaka "Rim".

rb	ime	grad
S3	Mario	Rim

- Na kraju, upit vraća vrijednost tablicu s atributom *ime* za svaki isfiltirani slog (u ovom slučaju samo jedna vrijednost):

name
Mario

Primjeri: (baza podataka 'stuslu')

```
SELECT *
FROM   ispit
WHERE  dat_ispit BETWEEN '01/01/1997' AND '11/09/1998';
ORDER BY dat_ispit DESC;

SELECT *
FROM   mjesto
WHERE  naz_mjesto LIKE 'z%'
        OR naz_mjesto LIKE '_sije_';

SELECT *
FROM   student
WHERE  pbr_stan IN ( 54000, 41000 );

SELECT *
FROM   student
WHERE  pbr_stan NOT IN (SELECT pbr
                        FROM   mjesto
```

```
                        WHERE  naz_mjesto = 'zagreb'
                               OR naz_mjesto = 'osijek')
ORDER BY ime_stud;
```

4.4. Spajanje tablica

FROM parametar dozvoljava dohvaćanje podataka iz više od jedne tablice, ali samo nabrajanje tablica će vrlo rijetko vratiti očekivane rezultate. Slogovi jedne tablice moraju se povezati sa slogovima druge tablice. Taj se postupak zove spajanje.
Slijedeći primjer prikazuje mehanizam spajanja.
Dvije tablice:

tablica1

rb	sif_boje
1	c
2	p

tablica2

sif_boje	boja
c	crvena
p	plava
z	zelena

prikazuju se slijedećom naredbom:

```
SELECT *
FROM   tablica1,
       tablica2
```

Pri čemu se dobiva sljedeći rezultat:

rb	sif_boje	sif_boje	boja
1	c	c	crvena
1	c	p	plava
1	c	z	zelena
2	p	c	crvena
2	p	p	plava
2	p	z	zelena

Dakle, svaki slog *tablice1* je kombiniran sa svakim slogom *tablice2*, što rezultira sa 6 slogova u dobivenoj tablici. Takav rezultat se naziva i ***kartezijev produkt***.
Korisniji upit se dobije kada se slogovi tablice1 povežu sa slogovima tablice2, povezivanje se radi preko zajedničkog atributa (koji je najčešće strani ključ između te dvije tablice, ali nije obavezno):

```
SELECT *
FROM   tablica1,
       tablica2
WHERE  tablica1.sif_boje = tablica2.sif_boje
```

Rezultat sada izgleda:

rb	sif_boje	sif_boje	boja

1	c	c	crvena
2	p	p	plava

4.5. GROUP BY parametar

Predstavlja uvjet za grupiranje rezultirajućih slogova. Koriste se kod agregatnih funkcija (LV 3).

Sintaksa:

[GROUP BY [ALL] *group_by_expression* [,...n]]

ALL - U rezultat ulaze svi slogovi, čak i oni koji ne zadovoljavaju uvjet iz WHERE stavke.

group_by_expression - Izraz prema kojemu se vrši grupiranje. Taj izraz mora sadržavati barem sve one atribute, navedene u SELECT stavci koji nisu unutar agregatnih funkcija ili mora biti identičan SELECT stavci.

4.6. HAVING parametar

Specifikacija uvjeta pretraživanja za grupu. Koristi se zajedno sa GROUP BY stavkom da bi se dodatno filtrirali rezultati nakon grupiranja.

Sintaksa:

[HAVING < search_condition >]

4.7. ORDER BY parametar

Određuje redoslijed prikaza podataka u rezultirajućoj tablici (sortiranje).

Sintaksa:

[ORDER BY { *order_by_expression* [ASC | DESC] } [,...n]]

order_by_expression - Specificira atribut(e) prema kojemu se vrši sortiranje rezultata. Atribut(i) se može navesti svojim imenom ili pseudonimom

ASC - Uzlazni redoslijed sortiranja

DESC - Silazni redoslijed sortiranja

Primjeri SELECT naredbe s dodatnim parametrima: (baza podataka 'stuslu')

```
SELECT mbr_stud,
       ime_stud,
       prez_stud
FROM   student
WHERE  spol = 'Ž'
ORDER  BY prez_stud;

SELECT i.sif_pred,
       Avg(ocjena) AS "prosjek"
INTO   prosijeci
FROM   ispit AS i,
       predmet AS p
WHERE  i.sif_pred = p.sif_pred
       AND i.ocjena > 1
GROUP  BY i.sif_pred
HAVING Avg(ocjena) > 2.5
ORDER  BY Avg(ocjena);
```

5. Zadaci

Zadatak 1. (**baza podataka 'student'**)

Kreirati tablicu *osobe* koja će imati atribute: ime, prezime, jmbg, datum rođenja, spol, visina, slika i broj cipela. Definirajte sva potrebna ograničenja da bi tablica bila funkcionalna. Unijeti 5 osoba u tablicu koristeći INSERT naredbu.

Zadatak 2. (**baza podataka 'student'**)

Zamijeniti ime i prezime druge osobe iz tablice zadatka 1. Povećati broj cipela svim osobama u tablici (koristeći jednu UPDATE naredbu). Obrisati treću osobu iz tablice zadatka 1.

Zadatak 3. (baza podataka 'stuslu')

Ispisati sve podatke o studentima prema silaznom redoslijedu datuma upisa na fakultet (iz tablice *Student*).
Ispisati ime, prezime i datum rođenja svih studenata (poredano abecedno po prezimenu) koji su rođeni prije 01.01.1978. godine. Za atribute koristiti opisne nazive (AS).

Zadatak 4. (baza podataka 'stuslu')

Ispisati ime i prezime studenta i naziv predmeta za sve one ispite koji su položeni s ocjenom 4 ili većom. Za rješavanje zadatka potrebno je koristiti podatke iz 3 tablice (*Student*, *Predmet* i *Ispit*).

6. Assignments

Assignment 1. ('student' database)
Create a table persons which has attributes: first_name, last_name, JMBG, date_of_birth, gender, height, image, shoe_size. Also define all necessary constraints to make table 'student' functional.
Then insert 5 persons in the table by using INSERT command.

Assignment 2. ('student' database)
Switch first name and last name for a second person in the table ‘persons’. Increase the shoe size for all the persons in the table ‘person’ (by using a single UPDATE clause). Delete the third person from the table.

Assignment 3. (‘stuslu’ database)
Get all the data about students by the descending order of their year of college enrollment (from the table ‘student’). Get the first name, last name and the date of birth of all the students born before 01.01.1978. (by the ascending order of their last name). Use aliases for the columns (AS).

Assignment 4. (‘stuslu’ database)
Get the first name, last name and the course name for all the exams passed by the grade 4 or more. Use data from three tables (‘student’, ‘predmet’, ‘ispit’).