

# Napredne baze podataka

Kontrola toka

Kursori



# •Sadržaj predavanja

- Kontrola toka
- Kursori
- Privremene tablice

# •Kontrola toka

IF uvjetovanje

```
IF uvjet THEN naredbe  
  [ELSEIF uvjet1 THEN naredbe];  
  [ELSE naredbe];  
END IF;
```

- Moguće je ugnježđivanje IF unutar IF
- Vrijednost NULL nije ni TRUE niti FALSE

```
IF suma<0 THEN  
  DELETE FROM...;  
ELSEIF suma=0 THEN  
  SELECT FROM...;  
ELSE  
  INSERT INTO...;  
END IF;
```

# •Kontrola toka

## CASE uvjetovanje

```
CASE
```

```
  WHEN expression_1 THEN commands_1
```

```
  ...
```

```
  WHEN expression_n THEN commands_n
```

```
  ELSE commands
```

```
END CASE
```

```
CASE expr
```

```
  WHEN val_1 THEN commands_1
```

```
  ...
```

```
  WHEN val_n-1 THEN commands_n-1
```

```
  ELSE val_n
```

```
END CASE
```

# •Kontrola toka

## CASE uvjetovanje - primjer

- Napisati funkciju koja će dohvatiti trenutni datum s poslužitelja i ispisati koji je dan u tjednu.

```
DELIMITER //
```

```
CREATE FUNCTION danUTjednu() RETURNS VARCHAR(50)
```

```
DETERMINISTIC
```

```
    BEGIN
```

```
        DECLARE vrati VARCHAR(50);
```

```
        CASE DAYOFWEEK(CURDATE())
```

```
            WHEN 2 THEN SET vrati='Danas je ponedjeljak';
```

```
            WHEN 3 THEN SET vrati='Danas je utorak';
```

```
            WHEN 4 THEN SET vrati='Danas je srijeda';
```

```
            WHEN 5 THEN SET vrati='Danas je četvrtak';
```

```
            WHEN 6 THEN SET vrati='Danas je petak';
```

```
            ELSE SET vrati='Danas je vikend';
```

```
        END CASE;
```

```
        RETURN vrati;
```

```
    END; //
```

```
DELIMITER ;
```

```
SELECT danUTjednu();
```



# •Petlje

WHILE petlja

**WHILE** uvjet **DO**

**naredbe**

**END WHILE**

```
DELIMITER $$
CREATE PROCEDURE proc()
BEGIN
    DECLARE var INT;
    SET var=1;
    WHILE var<=5 DO
        SET var=var+1;
        SELECT var;
    END WHILE;
END;
$$
DELIMITER ;
CALL proc();
```

U petlji zbrajamo  $var=var+1$  sve dok var ne poprimi vrijednost 5.

Za svaku vrijednost var koju uvećamo ispišemo var (SELECT var;)

Rezultat:

2  
3  
4  
5  
6

# •Petlje

## REPEAT petlja

**REPEAT**

**naredbe;**

**UNTIL uvjet**

**END REPEAT**

```
DROP PROCEDURE proc;
DELIMITER $$
CREATE PROCEDURE proc()
    BEGIN
        DECLARE var INT;
        SET var=1;
        REPEAT
            SET var=var+1;
            SELECT var;
            UNTIL var>5
        END REPEAT;
    END;
$$
DELIMITER ;
CALL proc();
```

U petlji zbrajamo  $var=var+1$  sve dok var ne poprimi vrijednost 5.

Za svaku vrijednost var koju uvećamo, ispišemo var (SELECT var;)

Rezultat:

2  
3  
4  
5  
6

# •Petlje

## LOOP (LEAVE, ITERATE)

```
Ime_petlje: LOOP
    naredbe
    IF uvjet THEN
        LEAVE ime_petlje;
    END IF;
    IF uvjet THEN
        ITERATE ime_petlje;
    END IF;
END LOOP;
```

- LEAVE (kao break u C)
  - izlazi iz petlje ako je zadovoljen uvjet
- ITERATE (kao continue u C)
  - pokreće novu iteraciju petlje ako je zadovoljen uvjet



# •Petlje

## LOOP (LEAVE, ITERATE)

```
DROP PROCEDURE proc;  
DELIMITER $$  
CREATE PROCEDURE proc()  
    BEGIN  
        DECLARE var INT;  
        SET var=1;  
        petlja:LOOP  
            SET var=var+1;  
            SELECT var;  
            IF var>5 THEN  
                LEAVE petlja;  
            END IF;  
        END LOOP;  
    END;  
DELIMITER ;  
  
CALL proc();
```

U petlji zbrajamo  $var=var+1$  sve dok var ne poprimi vrijednost 5.

Za svaku vrijednost var koju uvećamo, ispišemo var (SELECT var;)

Rezultat:

2

3

4

5

6

# •Primjer

- Napisati proceduru koja će ispisati prvih n prirodnih brojeva u jednoj varijabli. Brojevi moraju biti razdvojeni zarezom. (Zanemarite ako se zarez ispisuje i nakon zadnjeg broja.)

```
DELIMITER $$
DROP PROCEDURE IF EXISTS WhileLoopProc_zarez$$
CREATE PROCEDURE WhileLoopProc_zarez n INT)
    BEGIN
        DECLARE var INT DEFAULT NULL;
        DECLARE str VARCHAR(255) DEFAULT NULL;
        SET var=1;
        SET str='';
        WHILE var<=n DO
            SET str=CONCAT(str,var,',');
            SET var=var + 1;
        END WHILE;
        SELECT str;
    END $$
DELIMITER ;
CALL WhileLoopProc_zarez);
```

# •Primjer

- Napisati proceduru koja će ispisati prvih n prirodnih brojeva u jednoj varijabli. Brojevi moraju biti razdvojeni zarezom. (Obratite pažnju da se zarez NE ispisuje i nakon zadnjeg broja.)

```
DELIMITER $$
DROP PROCEDURE IF EXISTS WhileLoopProc$$
CREATE PROCEDURE WhileLoopProc(IN n INT)
    BEGIN
        DECLARE var INT DEFAULT NULL;
        DECLARE str VARCHAR(255) DEFAULT NULL;
        SET str='1';
        SET var=2;
        WHILE var<=n DO
            SET str=CONCAT_WS(',',str,var);
            SET var=var + 1;
        END WHILE;
        SELECT str;
    END $$
DELIMITER ;
CALL WhileLoopProc(8);
```

Može li sa 'običnom'  
CONCAT funkcijom?

# •Primjer

- Napisati proceduru koja za zadani odjel broji radnike koji pripadaju tom odjelu. Ako odjel ima više od 10 radnika, procedura mora vratiti -1, a ako odjel nema radnika, procedura mora vratiti 0. U ostalim slučajevima, procedura vraća stvarni broj radnika. Napisati poziv procedure za odjel 100005. Napisati poziv procedure za odjel 5.

```
DELIMITER //
```

```
CREATE PROCEDURE odjel_rad(IN zadaniOdjel INT,OUT broj INT)
```

```
BEGIN
```

```
    SELECT COUNT(*) INTO broj FROM radnik WHERE sifOdjel=zadaniOdjel;
```

```
    IF broj>10 THEN
```

```
        SET broj=-1;
```

```
    ELSEIF broj=0 THEN
```

```
        SET broj=0;
```

```
    END IF;
```

```
END;
```

```
//
```

```
DELIMITER ;
```

```
CALL odjel_rad(100005,@a);
```

```
SELECT @a;
```

# •Primjer

- Napisati funkciju za unos novog odjela u tablicu *odjel*. Funkcija treba **provjeriti postoji li već odjel sa zadanim imenom**. Ako postoji, završiti s radom (vrati 0). Ako ne postoji, **pridijeliti mu šifru** i unijeti u tablicu (vrati 1). (Primijetiti da na šifri odjela ne postoji *autoincrement*.)

```
DELIMITER //
```

```
CREATE FUNCTION unesiOdjel(noviOdjel VARCHAR(50)) RETURNS INT
```

```
DETERMINISTIC
```

```
BEGIN
```

```
    DECLARE broj, vrati, zadnji INT DEFAULT NULL;
```

```
    SELECT COUNT(*) INTO broj FROM odjel WHERE nazivOdjel=noviOdjel;
```

```
    IF broj=0 THEN
```

```
        SELECT MAX(sifOdjel) INTO zadnji FROM odjel;
```

```
        INSERT INTO odjel (sifOdjel, nazivOdjel) VALUES ((zadnji+1),noviOdjel);
```

```
        SET vrati=1;
```

```
    ELSE
```

```
        SET vrati=0;
```

```
    END IF;
```

```
    RETURN vrati;
```

```
END;
```

```
//
```

```
DELIMITER ;
```

```
SELECT unesiOdjel('Odjel za reklamacije');
```

```
SELECT * FROM odjel;
```



# •Kursori - zašto?

- Do sada smo vidjeli sintaksu  
SELECT SUM(cijena) INTO cijena FROM....  
Ili  
SELECT naziv INTO naziv1 FROM...
- S time da moramo naglasiti da je naziv1 sadržavao samo jednu vrijednost (jednu n-torku)
- Što ako želimo u proceduri/funkciji doći do svih n-torki rezultata SQL query-a i dodatno s njima upravljati?
- Odgovor su kursori!

# •Kursori

- **Upravljanje sa setom podataka** dobivenih pomoću SELECT upita
- Uzastopno dohvaćanje n-torki iz dobivenog seta rezultata (prilikom svake iteracije petlje)
- Služi za procesiranje pojedinih n-torki koje je baza vratila kao rezultat upita
- Svojstva
  - Ne mogu se koristiti za dohvat prethodnog podatka
  - *Read-only* - kursor se ne može promijeniti
  - Koriste se isključivo za dohvat podataka
    - ne za unos niti za brisanje
    - kod UPDATEa bi moglo rezultirati neočekivanim rezultatom
  - Unutar transakcije, automatski se brišu pozivom naredbe COMMIT
- Kada ne koristiti kursor?

# •Rad sa kursorom (1)

`[I] DECLARE imeKursora CURSOR FOR select_izraz;`

- Kursor mora biti deklariran nakon deklaracije svih varijabli, ali prije bilo kojega dijela koda!

- Npr

- `DECLARE cur1 CURSOR FOR SELECT ime  
FROM osoba NATURAL JOIN mjesto;`

- `DECLARE cur1 CURSOR FOR  
SELECT ime, prezime FROM osoba;`

- ~~`DECLARE cur1 CURSOR FOR  
SELECT * FROM osoba;`~~

- Zato što kasnije neće biti moguće odrediti koji će se atribut odnositi na koju varijablu, pogotovo u slučaju promjene strukture u bazi!

# •Rad sa kursorom (2)

## [ II ] OPEN ime\_kursora

- Pokreće kursor i aktivira podatke za čitanje
- Stvara se kontejner podataka

## [ III ] FETCH ime\_kursora INTO varijabla [,varijable...]

- Dohvaća podatke iz SQL naredbe
- Broj navedenih varijabli mora odgovarati broju atributa u SELECT dijelu naredbe!
- Nakon svakoga uspješnoga dohvata automatski prelazi na sljedeću n-torku
  - U slučaju da nema sljedeće n-torke, prijavljuje grešku (Error: **1329** SQLSTATE: **02000** (ER\_SP\_FETCH\_NO\_DATA) ; Message: No data - zero rows fetched, selected, or processed )

## [ IV ] CLOSE ime\_kursora;

- Zatvara kursor i oslobađa resurse
- Poželjno upotrebljavati, makar engine garantira oslobađanje resursa pri završetku procedure/funkcije

# •Kursor - primjer

- Napisati proceduru koja će dohvatiti svaki kvar zasebno i uz njegovo ime ispisati radi li se o velikom ili malom kvaru. Kriterij je iznos atributa *satiKvar*. Kvar se smatra velikim ako je *satiKvar* veći od 3.

```
DELIMITER $$
DROP PROCEDURE IF EXISTS etiketiraj$$
CREATE PROCEDURE etiketiraj()
    BEGIN
        DECLARE trenutni_naziv VARCHAR(255);
        DECLARE trenutni_sati, trenutni_id INT;
        DECLARE kur CURSOR FOR SELECT sifKvar, nazivKvar, satiKvar FROM kvar;
        OPEN kur;
        petlja:LOOP
        FETCH kur INTO trenutni_id, trenutni_naziv, trenutni_sati;
        IF trenutni_sati>3
            THEN SELECT trenutni_naziv AS ime, 'veliki' AS velicina;
        ELSE
            SELECT trenutni_naziv AS ime, trenutni_id AS sifra, 'mali' AS velicina;
        END IF;
        END LOOP petlja;
        CLOSE kur;
        SELECT 'kraj'; /*naredba dodana za
END; $$
DELIMITER ;
18CALL etiketiraj();
```

**Primijetiti: posljednji redak vraća obavijest o pogreški**  
**“No data - zero rows fetched, selected, or processed”**  
**Rješenje: prebrojati retke (COUNT) + REPEAT/WHILE**  
**umjesto LOOP PETELJE**  
**Ili**  
**CONDITIONS & HANDLERS**



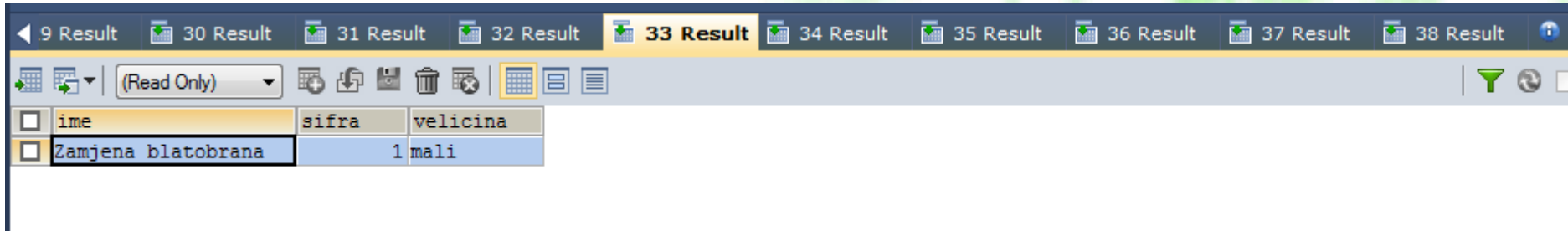
# •Kursor – primjer (izbjegavanje NOT FOUND)

- Napisati proceduru koja će dohvatiti svaki kvar zasebno i uz njegovo ime ispisati radi li se o velikom ili malom kvaru. Kriterij je iznos atributa *satiKvar*. Kvar se smatra velikim ako je *satiKvar* veći od 3.

```
DELIMITER $$
DROP PROCEDURE IF EXISTS etiketiraj$$
CREATE PROCEDURE etiketiraj()
    BEGIN
        DECLARE trenutni_naziv VARCHAR(255) DEFAULT NULL;
        DECLARE trenutni_sati, trenutni_id, dohvaceno INT DEFAULT NULL;
        DECLARE i INT DEFAULT 0;
        DECLARE kur CURSOR FOR SELECT sifKvar, nazivKvar, satiKvar FROM kvar;
        OPEN kur;
        SELECT FOUND_ROWS() INTO dohvaceno;
        WHILE i<dohvaceno DO
            FETCH kur INTO trenutni_id, trenutni_naziv, trenutni_sati;
            IF trenutni_sati>5
                THEN SELECT trenutni_naziv AS ime, 'veliki' AS velicina;
            ELSE
                SELECT trenutni_naziv AS ime, trenutni_id AS sifra, 'mali' AS
velicina;
            END IF;
            SET i=i+1;
        END WHILE;
        CLOSE kur;
        SELECT 'kraj'; /*naredba dodana za provjeru izvođenja procedure*/
    END; $$
DELIMITER ;
CALL etiketiraj();
```

# • Primjer (nastavak) – privremena tablica

- Navedeni program vratit će onoliko setova rezultata koliko je puta pozvan SELECT
  - Svaki SELECT zaseban je set rezultata
  - Takav ispis je nepregledan



ime	sifra	velicina
Zamjena blatobrana	1	mali

- Pozor:
  - Nemamo mogućnost vratiti više n-torki u glavni program
  - Rješenje: snimit ćemo ih u **privremenu tablicu**!
  - Modificirat ćemo program

# • Primjer (nastavak) – privremena tablica

- Napisati proceduru koja će dohvatiti svaki kvar zasebno i uz njegovo ime ispisati radi li se o velikom ili malom kvaru. Kriterij je iznos atributa *satiKvar*. Kvar se smatra velikim ako je *satiKvar* veći od 3.

```
DELIMITER $$  
DROP PROCEDURE IF EXISTS etiketiraj_2$$  
CREATE PROCEDURE etiketiraj_2()  
    BEGIN  
        DECLARE trenutni_naziv VARCHAR(255) DEFAULT NULL;  
        DECLARE trenutni_sati, trenutni_id, dohvaceno INT DEFAULT NULL;  
        DECLARE i INT DEFAULT 0;  
        DECLARE kur CURSOR FOR SELECT sifKvar, nazivKvar, satiKvar FROM kvar;  
  
        DROP TEMPORARY TABLE IF EXISTS tmp;  
        CREATE TEMPORARY TABLE  
            tmp(naziv VARCHAR(50),  
                id INT(11),  
                etiketa VARCHAR(10));  
  
        OPEN kur;  
        SELECT FOUND_ROWS() INTO dohvaceno;
```

# •Primjer (nastavak) – privremena tablica

```
WHILE i<dohvaceno DO
    FETCH kur INTO trenutni_id, trenutni_naziv, trenutni_sati;
    IF trenutni_sati>3
        /*THEN SELECT trenutni_naziv AS ime,
        trenutni_id AS sifra, 'veliki' AS velicina;*/
        THEN INSERT INTO tmp (naziv, id, etiketa) VALUES
            (trenutni_naziv, trenutni_id, 'veliki');
    ELSE
        /*SELECT trenutni_naziv AS ime,
        trenutni_id AS sifra, 'mali' AS velicina;*/
        INSERT INTO tmp (naziv, id, etiketa) VALUES
            (trenutni_naziv, trenutni_id, 'mali');
    END IF;
    SET i=i+1;
END WHILE;
CLOSE kur;
SELECT 'kraj'; /*naredba dodana za provjeru izvođenja procedure*/
SELECT * FROM tmp;

END; $$
DELIMITER ;
CALL etiketiraj_2();
```

# •Zadatak

- Napisati proceduru koja će svim radnicima koji imaju koeficijent plaće manji od 1.00 povišati ga **ZA** 1.00. Ostalim radnicima čiji je koeficijent plaće veći od 2.00 smanjiti ga **ZA** 0.50. Procedura mora vratiti *broj n-torki koje je obradila, broj radnika kojima je plaća uvećana te broj radnika kojima je plaća smanjena*.

```
DROP PROCEDURE IF EXISTS korigiraj_koef;
DELIMITER //
CREATE PROCEDURE korigiraj_koef()
BEGIN
    DECLARE koef DECIMAL(3,2);
    DECLARE sif, dohvaceno, smanjena, povecana INT;
    DECLARE flag BOOL;
    DECLARE kursor CURSOR FOR SELECT sifRadnik, koefPlaca FROM radnik;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET flag=TRUE;
    SET flag=FALSE;
    SET smanjena=0;
    SET povecana=0;

    OPEN kursor;
    SELECT FOUND_ROWS() INTO dohvaceno;
```

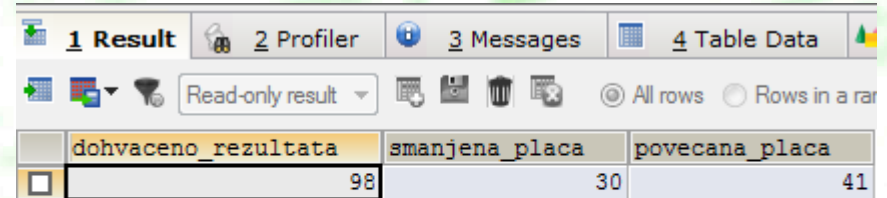


# •Zadatak (nastavak)

```
petlja:LOOP
    FETCH kursor INTO sif, koef;
    IF flag=TRUE THEN
        LEAVE petlja;
    END IF;
    IF koef<1.00 THEN
        SET koef=koef+1;
        UPDATE radnik SET koefPlaca=koef WHERE sifRadnik=sif;
        SET povecana=povecana+1;
    ELSEIF koef>=2.00 THEN
        SET koef=koef-0.5;
        UPDATE radnik SET koefPlaca=koef WHERE sifRadnik=sif;
        SET smanjena=smanjena+1;
    END IF;

END LOOP;
CLOSE kursor;
SELECT dohvaceno AS dohvaceno_rezultata,
       smanjena AS smanjena_placa, povecana AS povecana_placa;

END; //
DELIMITER ;
CALL korigiraj_koef();
```



	dohvaceno_rezultata	smanjena_placa	povecana_placa
<input type="checkbox"/>	98	30	41

CALL korigiraj\_koef()