

Napredne baze podataka

Conditions & Handlers



•Conditions & Handlers

- Služe da bismo preusmjerili tok koda u trenutku pojave nekog uvjeta – **CONDITION**
- Pokrivaju što se događa u trenutku pojave više događaja tj. grešaka – **HANDLER**
 - **TIP**
 - CONTINUE – nastavlja izvršavanje bloka bez prekida
 - EXIT – prekida se izvršavanje bloka u kojem se *handler* nalazi
 - Pretpostavljena akcija za sve pogreške neobuhvaćene handlerom
 - UNDO – bit će implementiran u nadolazećim verzijama
 - **UVJET**
 - može biti SQLSTATE vrijednost ili
 - mysql_error_code

• Conditions & Handlers - deklaracija

1. odrediti na koju se pogrešku odnosi:

- **DECLARE ime CONDITION FOR uvjet**

```
DECLARE err_table_exists CONDITION FOR 1050;
```

```
DECLARE err_table_exists CONDITION FOR SQLSTATE '42S01';
```

2. odrediti *handler* za zadani tip pogreške (sadrži naredbe koje će se izvesti u slučaju nastanka definirane pogreške)

- **DECLARE tip HANDLER FOR uvjeti**

```
DECLARE EXIT HANDLER FOR err_table_exists
```

```
BEGIN
```

```
    SET @table=-1;
```

```
END;
```

- **NAPOMENA:** moguće je definirati tip greške i unutar same deklaracije *handlera* (donji je kod ekvivalent gornjim koracima)

```
DECLARE EXIT HANDLER FOR 1050
```

```
BEGIN
```

```
    SET @table=-1;
```

```
3 / 33 END;
```

•Uvjeti

- SQLSTATE vrijednost – string od 5 znamenaka
 - ANSI SQL standardiziran
 - Ne preporuča se koristiti pogreška '00000' tj '0' – kod za uspješno izvršavanje
 - SQLWARNING = SQLSTATE koji počinje sa '01'
 - NOT FOUND = SQLSTATE koji počinje sa '02'
 - SQLEXCEPTION = sve što ne počinje sa '00', '01' ili '02'
- MySQL error code – broj pogreške
 - Error codes:
<http://dev.mysql.com/doc/refman/5.7/en/error-messages-server.html>

•Conditions & Handlers

```
DELIMITER //
```

```
CREATE PROCEDURE handlerdemo()  
BEGIN  
    DECLARE CONTINUE HANDLER FOR SQLSTATE '23000'  
        SET @x=1;  
    SET @x=1;  
    INSERT INTO test.t VALUES(1);  
    SET @x=2;  
    INSERT INTO test.t VALUES(1);  
    SET @x=3;  
END;  
//  
DELIMITER ;  
  
CALL handlerdemo();  
SELECT @x  
→ vraća 3
```

Ide na sljedeću naredbu

Definira slučaj
pojave greške
Duplicate key

Glatko prolazi

Uzrokuje grešku, ali nastavlja
sa izvođenjem – dokaz da
nastavlja

•Primjer

- Napisati proceduru koja će za zadani poštanski broj pronaći sve nazive mjesta u županiji u kojoj se nalazi zadano mjesto. Neka procedura ispiše nazive tih mjesta na ekran.

```
DELIMITER //
```

```
CREATE PROCEDURE pbr(ulaz INT)
```

```
BEGIN
```

```
    DECLARE p_zup INT;
```

```
    DECLARE naziv VARCHAR(50);
```

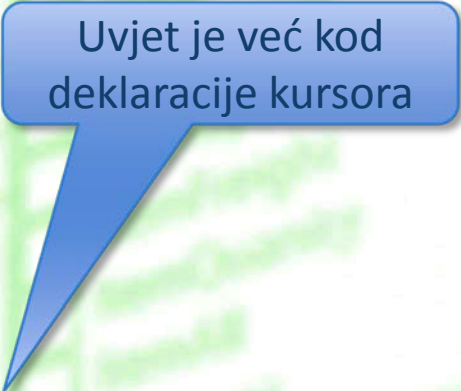
```
    DECLARE error INT DEFAULT 0;
```

```
    DECLARE kursor CURSOR FOR SELECT nazivMjesto
```

```
        FROM mjesto WHERE mjesto.sifZupanija=p_zup;
```

```
    DECLARE CONTINUE HANDLER FOR NOT FOUND
```

```
        SET error=1;
```



Uvjet je već kod deklaracije kursora

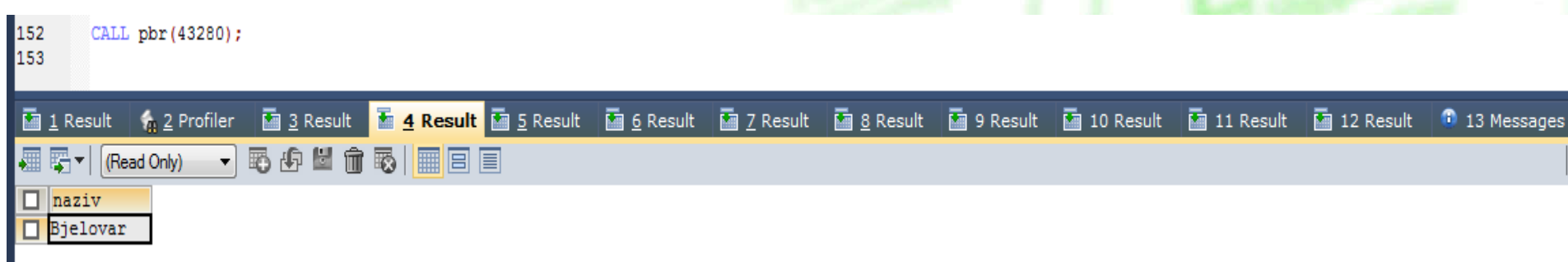
- Ili umjesto NOT FOUND napisati SQLSTATE '02000'

•Primjer (nastavak)

```
SELECT sifzupanija INTO p_zup FROM mjesto
      WHERE mjesto.pbrmjesto=ulaz;
OPEN kursor;
petlja: LOOP
      FETCH kursor INTO naziv;
      IF error=1 THEN
            LEAVE petlja;
      END IF;
      SELECT naziv;
END LOOP;
CLOSE kursor;
END;
//
DELIMITER ;
CALL pbr(43280);
```

• 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



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

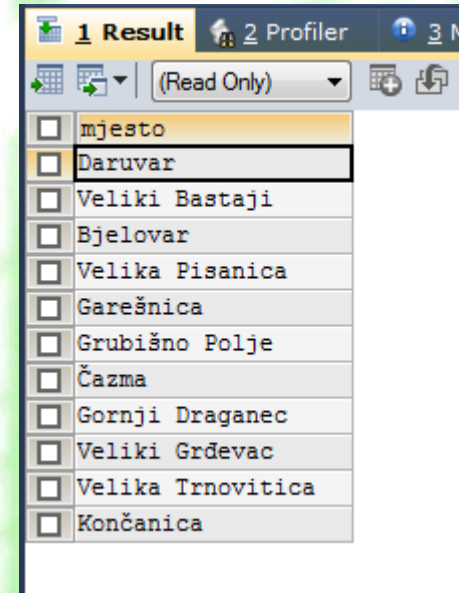
•Primjer (nastavak) – privremena tablica

- Napisati proceduru koja će za zadani poštanski broj pronaći sve nazive mjesta u županiji u kojoj se nalazi zadano mjesto. Neka procedura ispiše nazive tih mjesta na ekran.

```
DROP PROCEDURE pbr;  
  
DELIMITER //  
  
CREATE PROCEDURE pbr(ulaz INT)  
BEGIN  
  
    DECLARE p_zup INT;  
  
    DECLARE naziv VARCHAR(50);  
  
    DECLARE error INT DEFAULT 0;  
  
    DECLARE kursor CURSOR FOR SELECT nazivMjesto  
        FROM mjesto WHERE mjesto.sifZupanija=p_zup;  
  
    DECLARE CONTINUE HANDLER FOR NOT FOUND  
        SET error=1;
```

•Primjer (nastavak) – privremena tablica

```
DROP TEMPORARY TABLE IF EXISTS bbzup;  
CREATE TEMPORARY TABLE bbzup(mjesto VARCHAR(50));  
SELECT sifzupanija INTO p_zup FROM mjesto  
      WHERE mjesto.pbrmjesto=ulaz;  
OPEN kursor;  
petlja: LOOP  
      FETCH kursor INTO naziv;  
      IF error=1 THEN  
          LEAVE petlja;  
      END IF;  
      SELECT naziv; → INSERT INTO bbzup VALUES(naziv);  
END LOOP;  
SELECT * FROM bbzup;  
CLOSE kursor;  
END; //  
DELIMITER ;  
CALL pbr(43280);
```



The screenshot shows a database application interface with a '1 Result' tab. Below the tab is a table with a single column 'mjesto'. The table contains the following rows:

mjesto
Daruvar
Veliki Bastaji
Bjelovar
Velika Pisanica
Garešnica
Grubišno Polje
Čazma
Gornji Draganec
Veliki Grđevac
Velika Trnovitica
Končanica

•Primjer – privremena tablica&funkcija

- Možemo li isto rješenje realizirati izradom funkcije? (Pazi: funkcija ne može ispisati sadržaj privremene tablice)

```
DROP FUNCTION IF EXISTS pbr;  
DELIMITER //  
CREATE FUNCTION pbr(ulaz INT) RETURNS INT DETERMINISTIC  
BEGIN  
    DECLARE p_zup INT;  
    DECLARE naziv VARCHAR(50);  
    DECLARE error INT DEFAULT 0;  
    DECLARE kursor CURSOR FOR SELECT nazivMjesto  
        FROM mjesto WHERE mjesto.sifZupanija=p_zup;  
    DECLARE CONTINUE HANDLER FOR NOT FOUND  
        SET error=1;
```

•Primjer – privremena tablica&funkcija

```
DROP TEMPORARY TABLE IF EXISTS bbzup;
CREATE TEMPORARY TABLE bbzup(mjesto VARCHAR(50));
SELECT sifzupanija INTO p_zup FROM mjesto WHERE mjesto.pbrmjesto=ulaz;
OPEN kursor;
petlja: LOOP
    FETCH kursor INTO naziv;
    IF error=1 THEN
        LEAVE petlja;
    END IF;
    INSERT INTO bbzup VALUES(naziv);
END LOOP;
CLOSE kursor;
RETURN 1;
END; //
```

```
DELIMITER ;

SELECT pbr(43280); /*poziv funkcije*/

SELECT * FROM bbzup; /*ispis rezultata 'ručno' izvan funkcije*/
```

•Handleri – primjeri uporabe

- DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
SET I_error=1;
- DECLARE EXIT HANDLER FOR SQLEXCEPTION
BEGIN
ROLLBACK;
SELECT 'Error occurred – terminating';
END;
- DECLARE CONTINUE HANDLER FOR SQLSTATE '23000'
SELECT 'Duplicate key in index';
- DECLARE CONTINUE HANDLER FOR NOT FOUND
SET I_done=1;
- DECLARE CONTINUE HANDLER FOR SQLSTATE '02000'
SET I_done=1;

•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;
```

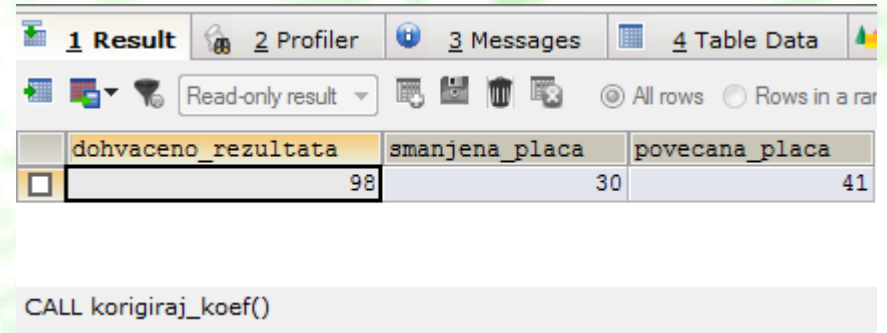
```
CLOSE kursor;
```

```
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()

•Zadatak

- Napisati proceduru koja će primiti šifre dvaju radnika (rOduzmiID i rDodajID) te iznos. Prvom radniku je potrebno smanjiti iznosOsnovice za zadani iznos, a drugom ga uvećati.
- Potrebno je provjeriti da li prvi radnik ima iznosOsnovice veći od zadanog iznosa. Također, ne smije se dogoditi da je prvom iznos oduzet, a drugome nije dodan.

```
DROP PROCEDURE IF EXISTS obaviTransakciju ;  
DELIMITER &&  
CREATE PROCEDURE obaviTransakciju  
    (IN rOduzmiID INT,  
    IN rDodajID INT,  
    IN iznos DOUBLE,  
    OUT poruka VARCHAR(50))
```

•Zadatak (nastavak)

```
BEGIN
  DECLARE rltrenutnoStanje DOUBLE(10,2) DEFAULT 0;
  BEGIN
    DECLARE EXIT HANDLER FOR NOT FOUND
      SET poruka='Radnik nije pronađen pod
zadanim brojem!';
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
      BEGIN
        ROLLBACK;
        SET poruka='Greska!';
      END;
    SET autocommit=0;
    START TRANSACTION;
    SELECT iznosOsnovice INTO rltrenutnoStanje
      FROM radnik WHERE sifRadnik=rOduzmiID;
```

•Zadatak (nastavak)

```
/*'ako ima dovoljno', obavi transakciju*/
IF rltrenutnoStanje>iznos THEN
    UPDATE radnik SET iznosOsnovice=iznosOsnovice-iznos
        WHERE sifRadnik=rOduzmiID;
    UPDATE radnik SET iznosOsnovice=iznosOsnovice+iznos
        WHERE sifRadnik=rDodajID;

ELSE
    SET poruka='Radnik nema dovoljan iznos osnovice!';
END IF;
END;
IF poruka != '' THEN ROLLBACK;
    ELSE COMMIT;
    SET poruka='SUCCESS! Transakcija je uspješno obavljena!';
END IF;
SELECT poruka;
SET autocommit=1;
END &&
DELIMITER ;

CALL obaviTransakciju(518,514,1000,@a);
CALL obaviTransakciju(518,514,5000,@a);
```

