OBRADA GREŠAKA U POHRANJENIM PROCEDURAMA

XI auditorne vježbe

Teme

- Rukovanje uslovima (*condition handling*)
- Deklarisanje uslova (*condition*)
- Deklarisanje rukovatelja (*handler*)
- SIGNAL naredba
- RESIGNAL naredba

Rukovanje uslovima

- Tokom izvođenja pohranjenih programa mogu se pojaviti uslovi (*condition*) koji zahtjevaju posebno rukovanje (*handling*)
- Rukovatelji (*handlers*) se mogu definisati za opšte uslove kao što su upozorenja ili greške, ili za posebne uslove kao što je određeni kod greške
- Posebnim uslovima mogu biti pridružena imena putem kojih se može rukovati korištenjem rukovatelja

Deklarisanje uslova

```
DECLARE condition_name CONDITION FOR condition_value
condition_value:
    mysql_error_code
  | SQLSTATE [VALUE] sqlstate_value
```

- Ova naredba deklariše uslov (*condition*) za imenovanu grešku (*condition_value*), čime se pridružuje ime uslovu (*condition_name*) koji treba određeno rukovanje
- Imenu se može pristupiti u sljedećoj DECLARE ... HANDLER naredbi
- Deklaracija uslova se mora pojaviti prije deklaracije kursora i rukovatelja

Deklarisanje uslova

- ***condition_value*** može biti kod MySQL greške (broj) ili SQLSTATE vrijednost (string od 5 karaktera)
- Korištenje imena za uslove pomaže pojašnjenju koda pohranjenih programa

```
DECLARE no_such_table CONDITION FOR 1051 [ili SQLSTATE '42S02‘];
DECLARE CONTINUE HANDLER FOR no_such_table
   BEGIN
         -- body of handler
   END;
```

- Imena uslova kojima se pristupa iz SIGNAL ili RESIGNAL naredbi moraju koristiti SQLSTATE vrijednosti, a ne kod MySQL greške

Deklarisanje rukovatelja

DECLARE ***handler_action*** HANDLER

FOR ***condition_value*** [, ***condition_value***] ...

statement

handler_action:

CONTINUE | EXIT | UNDO

condition_value:

mysql_error_code

| SQLSTATE [VALUE] ***sqlstate_value***

| ***condition_name*** | SQLWARNING | NOT FOUND | SQLEXCEPTION

- Ova naredba specificira rukovatelj koji rukuje jednim ili više uslova
- Ukoliko se pojavi jedan od ovih uslova, izvršava se specificirani ***statement*** koji može biti jednostavan iskaz kao što je SET naredba ili složeni iskaz napisan korištenjem BEGIN ... END

Deklarisanje rukovatelja

- Deklaracija rukovatelja se mora pojaviti nakon deklaracije varijabli ili uslova
- ***handler_action*** vrijednost indicira koju akciju rukovatelj poduzima nakon izvršavanja naredbi rukovatelja:
  - CONTINUE – izvršavanje programa se nastavlja
  - EXIT – prekida se izvršavanje BEGIN ... END složenog izraza u kojem je rukovatelj deklarisan (čak i ako se uslov pojavi u ugniježdenom bloku)
  - UNDO: nije podržano

Deklarisanje rukovatelja

- ***condition_value*** indicira određeni uslov ili klasu uslova koji aktiviraju rukovatelj
  - kod MySQL greške (broj) ili SQLSTATE vrijednost (string od 5 karaktera)
  - Ime uslova prethodno specificirano sa DECLARE ... CONDITION
  - SQLWARNING je skraćenica za klasu SQLSTATE vrijednosti koje počinju sa '01'
  - NOT FOUND je skraćenica za klasu SQLSTATE vrijednosti koje počinju sa '02'
  - SQLEXCEPTION je skraćenica za klasu SQLSTATE vrijednosti koje ne počinju sa '00', '01' i '02'

Deklarisanje rukovatelja

- Ukoliko se pojavi uslov za koji nije deklarisan rukovatelj, akcija koja se poduzima ovisi o klasi uslova
  - Za SQLEXCEPTION uslove, pohranjeni program se prekida na naredbi koja je izazvala grešku, kao da postoji EXIT handler
  - Za SQLWARNING uslove, program nastavlja sa izvršavanjem kao da postoji CONTINUE handler
  - Za NOT FOUND uslove, ukoliko se uslov pojavio normalno, akcija je CONTINUE. Ukoliko je izazvan sa SIGNAL ili RESIGNAL, akcija je EXIT.
- Za ignorisanje uslova, deklarišite CONTINUE rukovatelj za njega i pridružite ga praznom bloku, npr.

```
DECLARE CONTINUE HANDLER FOR SQLWARNING BEGIN END;
```

Deklarisanje rukovatelja

- Vidljivost labele bloka ne uključuje kod za rukovatelje deklarisane unutar bloka
- Zbog toga, naredbe pridružene rukovatelju ne mogu koristiti ITERATE ili LEAVE za referisanje na labele blokova koji obuhvataju deklaraciju rukovatelja

```
CREATE PROCEDURE p ()
BEGIN
    DECLARE i INT DEFAULT 3;
    retry: REPEAT
            BEGIN
                    DECLARE CONTINUE HANDLER FOR SQLWARNING
                        BEGIN
                            ITERATE retry; # illegal
                        END;
                    IF i < 0 THEN
                            LEAVE retry; # legal
                    END IF;
                    SET i = i - 1;
            END;
    UNTIL FALSE END REPEAT;
END;
```

Deklarisanje rukovatelja

- Kako bi se izbjeglo referenciranje na vanjske labele u rukovateljima koristite jednu od strategija:
  - Da bi se napustio blok koristite EXIT rukovatelj
  - Da bi se nastavilo izvršavanje, postavite status varijablu u CONTINUE rukovatelju koja može biti provjerena u bloku kako bi se odredilo da je rukovatelj pozvan, npr

```
CREATE PROCEDURE p ()
BEGIN
    DECLARE i INT DEFAULT 3;
    DECLARE done INT DEFAULT FALSE;
    retry: REPEAT
            BEGIN
                    DECLARE CONTINUE HANDLER FOR SQLWARNING
                            BEGIN
                                    SET done = TRUE;
                            END;
                    IF done OR i < 0 THEN
                            LEAVE retry;
                    END IF;
                    SET i = i - 1;
            END;
    UNTIL FALSE END REPEAT;
END;
```

Dijagnostičke informacije

- SQL naredbe proizvode dijagnostičke informacije
- GET DIAGNOSTIC naredba omogućava aplikacijama provjeru ovih informacija
- Obično se koristi u rukovatelju unutar pohranjenih rutina. Ukratko, sadrži dvije vrste informacija:
  - Informacije o naredbi, kao što je broj uslova koji se javljaju ili broj zapisa na koje utiče
  - Informacije o uslovu, kao što su kod greške i poruka

Dijagnostičke informacije

- GET DIAGNOSTIC može dohvatiti informacije ili o naredbi ili o uslovu, ali ne oboje u istoj naredbi
  - Da bi se dohvatile informacije o naredbi, dohvatite željene stavke naredbe u ciljane varijable.

```
GET DIAGNOSTICS @p1 = NUMBER, @p2 = ROW_COUNT;
```

  - Da bi se dohvatile informacije o uslovu, specificirajte broj uslova i dohvatite željene stavke uslova u ciljane varijable.

```
GET DIAGNOSTICS CONDITION 1
   @p3 = RETURNED_SQLSTATE, @p4 = MESSAGE_TEXT;
```

Dijagnostičke informacije

- Lista za dohvat specificira jedno ili više ***target = item_name*** dodjeljivanja razdvojenih zarezima
- Svako dodjeljivanje navodi ciljanu varijablu i oznaku ili ***statement_information_item_name*** ili ***condition_information_item_name*** u zavisnosti da li se dohvataju informacije o naredbi ili uslovu
- Validne target oznake za pohranu informacija o stavki mogu biti parametri procedure ili funkcije, lokalne varijable u pohranjenim procedurama ili session-specific varijable
- Validne ***condition_number*** oznake mogu biti i sistemske varijable ili literali

```
CREATE PROCEDURE do_insert(value INT)
BEGIN
    -- Daklaracija varijabli koje će čuvati informacije iz dijagnostičkog područja
    DECLARE kod CHAR(5) DEFAULT '00000';
    DECLARE poruka TEXT;
    DECLARE redova INT;
    DECLARE rezultat TEXT;
    -- Deklaracija rukovatelja greškom za neuspio unos
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
            BEGIN
                    GET DIAGNOSTICS CONDITION 1
                    kod = RETURNED_SQLSTATE, poruka = MESSAGE_TEXT;
            END;
    -- Izvedi unos
    INSERT INTO t1 (int_col) VALUES(value);
    -- Provjeri da li je unos bio uspješan
    IF kod = '00000' THEN
            GET DIAGNOSTICS redova = ROW_COUNT;
            SET rezultat = CONCAT('unos uspio, broj redova = ', redova);
    ELSE
            SET rezultat = CONCAT('unos nije uspio, greska = ',kod,', poruka = ',poruka);
    END IF;
     -- Dojavi šta se dogodilo
    SELECT rezultat;
END;
```

SIGNAL naredba

SIGNAL
    [***condition_value***]
    [SET ***signal_information_item***
    [, ***signal_information_item***] ...]

condition_value:
    SQLSTATE [VALUE] ***sqlstate_value***
    | ***condition_name***

signal_information_item:
    condition_information_item_name = ***simple_value_specification***

condition_information_item_name:
    CLASS_ORIGIN | SUBCLASS_ORIGIN
    | MESSAGE_TEXT | MYSQL_ERRNO
    | CONSTRAINT_CATALOG | CONSTRAINT_SCHEMA
    | CONSTRAINT_NAME | CATALOG_NAME
    | SCHEMA_NAME | TABLE_NAME
    | COLUMN_NAME | CURSOR_NAME

SIGNAL naredba

- SIGNAL naredba predstavlja način da se "vrati" greška
- SIGNAL rukovatelju obezbjeđuje informacije o grešci
- Također obezbjeđuje kontrolu nad karakteristikama greške (broj greške, SQLSTATE vrijednost, poruka)
- SQLSTATE za SIGNAL naredbu ne bi trebalo početi sa '00' jer takve vrijednosti indiciraju uspjeh i nisu validne za signaliziranje greške
- Kako bi se signalizirala generička SQLSTATE vrijednost koristite '45000' što znači "nerukovana korisnički definisana greška"
- SIGNAL naredba opciono uključuje SET klauzulu koja sadrži više stavki signala, u listi ***condition_information_item_name*** = ***simple_value_specification*** dodjeljivanja razdvojenih zarezom

```
CREATE PROCEDURE p (pval INT)
BEGIN
    DECLARE specialty CONDITION FOR SQLSTATE '45000';
    IF pval = 0 THEN
            SIGNAL SQLSTATE '01000';
    ELSEIF pval = 1 THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'An error
    occurred';
    ELSEIF pval = 2 THEN
            SIGNAL specialty SET MESSAGE_TEXT = 'An error occurred';
    ELSE
            SIGNAL SQLSTATE '01000' SET MESSAGE_TEXT = 'A warning
    occurred', MYSQL_ERRNO = 1000;
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'An error
    occurred', MYSQL_ERRNO = 1001;
    END IF;
END;
```

SIGNAL naredba

- Ukoliko SIGNAL naredba indicira određenu SQLSTATE vrijednost, ta vrijednost se koristi za signaliziranje određenog uslova
- Ukoliko SIGNAL naredba koristi imenovani uslov, uslov mora biti deklarisan u istom području vidljivosti u kojem se primjenjuje SIGNAL naredba i mora biti definisan korištenjem SQLSTATE vrijednosti, a ne broja MySQL greške
- Ukoliko je uslov sa istim imenom deklarisan više puta na različitim područjima vidljivosti, primjenjuje se deklaracija u najlokalnijem području
- Signali mogu biti izazvani i unutar rukovatelja greškama

RESIGNAL naredba

RESIGNAL
    [***condition_value***]
    [SET ***signal_information_item***
    [, ***signal_information_item***] ...]

condition_value:
    SQLSTATE [VALUE] ***sqlstate_value***
    | ***condition_name***

signal_information_item:
    condition_information_item_name = ***simple_value_specification***

condition_information_item_name:
    CLASS_ORIGIN | SUBCLASS_ORIGIN
    | MESSAGE_TEXT | MYSQL_ERRNO
    | CONSTRAINT_CATALOG | CONSTRAINT_SCHEMA
    | CONSTRAINT_NAME | CATALOG_NAME
    | SCHEMA_NAME | TABLE_NAME
    | COLUMN_NAME | CURSOR_NAME

RESIGNAL naredba

- Prosljeđuje informacije o stanju greške koje su dostupne tokom izvršavanja rukovatelja uslova unutar složenog iskaza u okviru pohranjene procedure ili funkcije, okidača ili događaja.
- RESIGNAL može promijeniti sve ili neke informacije prije prosljeđivanja
- RESIGNAL omogućava i rukovanje greškom i vraćanje informacija o grešci
- U suprotnom, izvršavanjem SQL naredbe u okviru rukovatelja uništavaju se informacije koje su uzrokovale aktiviranje rukovatelja
- RESIGNAL također može i skratiti neke procedure, ako dati rukovatelj rukuje dijelom situacije, a onda prosljeđuje uslov drugom rukovatelju iznad

RESIGNAL naredba

- RESIGNAL naredba ima condition_value i SET klauzule koje su obje opcione, zbog čega postoji nekoliko načina upotrebe
  - RESIGNAL zasebno:

    ```
    RESIGNAL;
    ```

  - RESIGNAL sa novim informacijama o signalu:

    ```
    RESIGNAL SET signal_information_item [,
    signal_information_item] ...;
    ```

  - RESIGNAL sa vrijednošću uslova i mogućim novim informacijama o signalu:

    ```
    RESIGNAL condition_value [SET
    signal_information_item [, signal_information_item]
    ...];
    ```

RESIGNAL naredba

- RESIGNAL zasebno jednostavno znači "proslijedi grešku bez promjene"
- Restaurira posljednje dijagnostičko područje i čini ga trenutnim dijagnostičkim područjem
- Unutar rukovatelja koji obuhvata uslov, jedan način upotrebe RESIGNAL zasebno je da izvede neke druge akcije, a potom proslijedi bez promjene orginalne informacije o uslovu (informacije koje su postojale prije ulaska u rukovatelj)

```
DROP TABLE IF EXISTS xx;
delimiter //
CREATE PROCEDURE p ()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
            BEGIN
                    SET @error_count = @error_count + 1;
                    IF @a = 0 THEN RESIGNAL; END IF;
            END;
    DROP TABLE xx;
END//
delimiter ;
SET @error_count = 0;
SET @a = 0;
CALL p();
```

RESIGNAL naredba

- RESIGNAL sa SET klauzulom obezbjeđuje nove informacije o signalu, pa naredba znači "proslijedi grešku sa izmjenama"
- Kao i sa RESIGNAL zasebno, ideja je da se uradi pop na steku dijagnostičkog područja tako da se vrati originalna informacija

```
DROP TABLE IF EXISTS xx;
delimiter //
CREATE PROCEDURE p ()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
            BEGIN
                    SET @error_count = @error_count + 1;
                    IF @a = 0 THEN RESIGNAL SET MYSQL_ERRNO = 5;
                    END IF;
            END;
    DROP TABLE xx;
END//
delimiter ;
SET @error_count = 0;
SET @a = 0;
CALL p();
```

RESIGNAL naredba

- RESIGNAL sa vrijednošću uslova znači "potisni uslov u trenutno dijagnostičko područje"
- Ukoliko je pristuna SET klauzula također mijenja informacije o grešci
- Ova forma RESIGNAL naredbe restaurira posljednje dijagnostičko područje i čini ga trenutnim dijagnostičkim područjem
- Ona radi pop steka dijagnostičkog područja što je isto što bi zasebni RESIGNAL uradio
- Međutim, također mijenja dijagnostičko područje u ovisnosti o vrijednosti uslova ili informaciji o signalu

```
DROP TABLE IF EXISTS xx;
delimiter //
CREATE PROCEDURE p ()
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
            BEGIN
                    SET @error_count = @error_count + 1;
                    IF @a = 0 THEN
                            RESIGNAL SQLSTATE '45000' SET MYSQL_ERRNO=5;
                    END IF;
            END;
    DROP TABLE xx;
END// delimiter ;
SET @error_count = 0;
SET @a = 0;
SET @max_error_count = 2;
CALL p();
SHOW ERRORS;
```

Pravila vidljivosti za rukovatelje

- Primjenjivost bilo kojeg rukovatelja ovisi o njegovoj lokaciji u okviru definicije programa i o uslovu ili uslovima kojima rukuje
- Rukovatelj koji je deklarisan u BEGIN ... END bloku je u području vidljivosti samo za SQL naredbe koje slijede iza deklaracije rukovatelja u bloku
- Ukoliko sam rukovatelj izaziva grešku, ne može rukovati tim uslovom, niti može bilo koji drugi rukovatelj deklarisan u bloku
- Rukovatelj je u području vidljivosti samo za blok u kojem je deklarisan i ne može biti aktiviran uslovima koji se javljaju izvan tog bloka
- Rukovatelj može biti specifični ili opšti. Specifični rukovatelj je za kod MySQL greške, SQLSTATE vrijednost ili naziv uslova. Opšti rukovatelj je za uslov u SQLWARNING, SQLEXCEPTION ili NOT FOUND klasi

Pravila vidljivosti za rukovatelje

- Više rukovatelja može biti deklarisano u različitim područjima vidljivosti i sa različitim specifičnostima
- Da li se rukovatelj aktivira ovisi ne samo o vlastitom području vidljivosti i vrijednosti uslova, nego i o tome koji su drugi rukovatelji prisutni
- Kada server pronađe jedan ili više primjenjivih rukovatelja u datom području među njima bira na osnovu sljedećih prioriteta:
  - Rukovatelj kodom MySQL greške ima prednost nad rukovateljem SQLSTATE vrijednosti
  - Rukovatelj SQLSTATE vrijednosti ima prednost nad opštim SQLWARNING, SQLEXCEPTION i NOT FOUND rukovateljima
  - SQLEXCEPTION rukovatelj ima prednost nad SQLWARNING rukovateljem
  - Prednost NOT FOUND zavisi kako je uslov nastao
    - Normalno, uslov u NOT FOUND klasi može biti obuhvaćen SQLWARNING ili NOT FOUND rukovateljem, gdje SQLWARNING ima prednost ako su oba prisutna
    - Ako je NOT FOUND uslov izazvan SIGNAL naredbom može biti obuhvaćen samo NOT FOUND rukovateljem