VARIJABLE, NAREDBE KONTROLE TOKA I UPOTREBA KURSORA

X auditorne vježbe

Teme

- Deklarisanje varijabli u MySQL
- Petlje u MySQL
  - LOOP, REPEAT, WHILE
  - Izlazak iz petlje
- LIMIT naredba (TopN analiza)
- Upotreba kursora
  - Deklaracija, otvaranje, pomjeranje kroz kursor i zatvaranje

Deklarisanje varijabli u MySQL

- U MySQL vrijednost se u jednom iskazu može pohraniti u korisnički definisanu varijbalu, a potom joj kasnije pristupiti iz drugog iskaza
- Omogućava se prosljeđivanje vrijednosti iz jednog iskaza u drugi
- Varijable koje se koriste na ovaj način su specifične za jednu korisničku sjednicu (*session-specific*)
- Varijabli koju je definisao jedan klijent ne može vidjeti niti koristiti drugi klijent
- Postoje samo za vrijeme trajanja korisničke sjednice
- Nakon završetka sjednice varijable se brišu

Deklarisanje varijabli u MySQL

- *Session-specific* varijable se pišu sa *@ime_var* gdje se *ime_var* sastoji od alfanumeričkih karaktera, '.', '_' i '$'
- Od vezije MySQL 5.0 nazivi varijabli nisu case sensitive
- Jedan način postavljanja *session-specific* varijable je izvršavanjem SET iskaza

```
SET @var_name = expr [, @var_name = expr] ...
```

- U SET iskazu se kao operator dodjeljivanja može koristiti operator = ili :=
- U drugim izrazima u kojima se dodjeljuje vrijednost varijabli mora se koristiti operator :=

```
SET @t1=1, @t2=2, @t3:=4;
SELECT @t1, @t2, @t3, @t4 := @t1+@t2+@t3;
```

Deklarisanje varijabli u MySQL

- Korisničkim varijablama se mogu dodjeliti vrijednosti iz ograničenog skupa tipova podataka:
  - integer, decimal, floating-point, binary ili nonbinary string ili NULL vrijednost
- Ukoliko se dodjeljuje vrijednost drugog tipa ona se pretvara u dozvoljeni tip
  - Npr. vrijednost vremenskog ili prostornog tipa se pretvara u binary string
- Vrijedi opšte pravilo:
  - U jednom iskazu ne bi trebalo dodjeljivati varijabli vrijednost i čitati vrijednost

Deklarisanje varijabli u MySQL

- Ukoliko se koristi varijabla koja nije inicijalizirana, ima vrijednost NULL i tipa je string
- Koriste se prvenstveno za pohranu vrijednosti i ne mogu direktno biti korištene u SQL iskazima kao identifikator

```
SET @col = "c1";
SELECT @col FROM t;
```

- Izuzetak je kada se konstruiše string koji će se koristiti kao pripremljeni iskaz za kasnije izvršavanje

```
SET @c = "c1";
SET @s = CONCAT("SELECT ", @c, " FROM t");
PREPARE stmt FROM @s;
EXECUTE stmt;
```

Deklarisanje varijabli u MySQL

- Primjer dvije funkcije koje dijele jednu *session-specific* varijablu.

```
CREATE FUNCTION func1()
  RETURNS INT
  BEGIN
  SET @svar = 2;
  SET @svar = @svar + 1;
  RETURN @svar;
END;
CREATE FUNCTION func2()
  RETURNS INT
  BEGIN
  SET @svar = 5;
  SET @svar = @svar + 1;
  RETURN @svar;
END;
```

Deklarisanje varijabli u MySQL

- *Session-specific* varijabla nema osobinu globalne varijable koja se inicijalizira samo pri prvoj upotrebi za vrijeme jedne sjednice.
- Ako se funkcije iz prethodnog primjera izvrše sljedećim redom, vrijednost **svar** će biti 6.

```
SELECT func1();
SELECT func2();
SELECT @svar;
```

- Ako se izvrše obrnutim redom, vrijednost **svar** će biti 3.

```
SELECT func2();
SELECT func1();
SELECT @svar;
```

CASE naredba

- CASE naredba u pohranjenim programima implementira konstrukciju sa složenim uslovima

```
CASE case_value
    WHEN when_value THEN statement_list
    [WHEN when_value THEN statement_list] ...
    [ELSE statement_list]
END CASE
```

- Ili

```
CASE
    WHEN search_condition THEN statement_list
    [WHEN search_condition THEN statement_list] ...
    [ELSE statement_list]
END CASE
```

CASE naredba

- Za prvu sintaksu, *case_value* je izraz.
- Ova vrijednost se poredi sa *when_value* izrazom u svakoj WHEN klauzuli dok se ne podudari sa nekom, kada se izvršava *statement_list* odgovarajuće THEN klauzule
- Ukoliko nijedna *when_value* nije jednaka, izvršava se *statement_list* ELSE klauzule ukoliko ona postoji
- Za drugu sintaksu, *search_condition* izraz svake WHEN klauzule se evaluira dok jedan ne bude istinit, kada se izvršava *statement_list* pripadajuće THEN klauzule
- Ukoliko nijedan *search_condition* nije istinit, izvršava se *statement_list* ELSE klauzule ukoliko ona postoji
- Ukoliko se ne izvršava nijedna WHEN klauzula, a CASE naredba ne sadrži ELSE klauzulu, javlja se greška *Case not found for CASE statement*
- Prazna *statement_list* nije dozvoljena

LOOP petlja

- LOOP implementira jednostavnu konstrukciju petlje, što omogućava ponavljanje jedne ili više naredbi, od kojih se svaka završava sa (;)

```
[begin_label:] LOOP
    statement_list
END LOOP [end_label]
```

- Naredbe unutar petlje se izvršavaju dok se petlja ne okonča
- Ovo se obično postiže upotrebom LEAVE iskaza, ili RETURN naredbe ako se radi o pohranjenoj funkciji
- LOOP petlja može biti labelirana

REPEATE petlja

```
[begin_label:] REPEAT
    statement_list
UNTIL search_condition
END REPEAT [end_label]
```

- Naredbe unutar REPEATE petlje se ponavljaju sve dok je *search_condition* izraz istinit
- Zbog toga se REPEAT petlja izvršava bar jednom
- *statement_list* sadrži jednu ili više naredbi, od kojih se svaka završava sa (;)
- REPEAT petlja može biti labelirana

WHILE petlja

```
[begin_label:] WHILE search_condition DO
    statement_list
END WHILE [end_label]
```

- Naredbe unutar WHILE petlje se ponavljaju sve dok je *search_condidtion* izraz istinit
- *statement_list* sadrži jednu ili više naredbi, od kojih se svaka završava sa (;)
- WHILE petlja može biti labelirana
- Unutar LOOP, REPEAT i WHILE petlji se može koristiti ITERATE naredba
- ITERATE znači "počni petlju ponovo"

```
ITERATE label
```

- LEAVE naredba se koristi da se izađe iz konstrukcije koja ima datu labelu
- Može se koristiti unutar BEGIN...END bloka ili petlji

```
LEAVE label
```

LIMIT naredba

- LIMIT klauzula se koristi u SELECT naredbi da ograniči broj redova u rezultujućem skupu
- LIMIT klauzula prihvata jedan ili dva argumenta koji moraju biti 0 ili pozitivni cijeli brojevi

```
LIMIT offset, count
```

- *Offset* specificira redni broj prvog reda koji će biti vraćen. *Offset* prvog reda je 0, a ne 1
- *Count* specificira maksimalan broj redova koji će biti vraćen
- Kada se LIMIT koristi sa jednim argumentom, ovaj argument se koristi da specificira maksimalan broj redova koji će biti vraćen sa početka rezultujućeg skupa

Upotreba kursora

- MySQL podržava kursore unutar pohranjenih programa koji imaju sljedeće osobine:
  - Server može a i ne mora praviti kopiju njegove rezultujuće tabele
  - Nije ih moguće koristiti za ažuriranje
  - Mogu se prelaziti samo u jednom smjeru i ne mogu se preskakati redovi
- Deklaracija kursora se mora pojaviti prije *handler* deklaracija a nakon deklaracija varijabli i uslova

```
DECLARE cursor_name CURSOR FOR select_statement
```

- Ova naredba deklariše kursor i pridružuje ga SELECT naredbi koja dohvaća zapise kroz koje će se kretati kursor
- Za dohvat redova koristi se FETCH naredba
- Broj kolona koje se dohvate SELECT naredbom mora odgovarati broju varijabli koje se koriste u FETCH naredbi
- SELECT naredba ne može imati INTO iskaz
- Pohranjeni program može sadržavati više deklaracija kursora, ali svaki deklarisani kursor u datom bloku mora imati jedinstven naziv

Upotreba kursora

```
OPEN cursor_name
```

- Ova naredba otvara deklarisani kursor

```
FETCH [[NEXT] FROM] cursor_name INTO var_name [, var_name]
```

- Ova naredba dohvata naredni red SELECT naredbe pridružene određenom kursoru (koji mora biti otvoren), te pomjera pokazivač kursora
- Ukoliko red postoji, dohvaćeni atributi se pohranjuju u imenovane varijable
- Ukoliko nema više dostupnih redova, pojavljuje se *No Data* uslov sa SQLSTATE vrijednošću '02000'.
- Za detektovanje ovog uslova, može se postaviti *handler* za njega (ili za NOT FOUND uslov)

```
CLOSE cursor_name
```

- Ova naredba zatvara prethodno otvoreni kursor
- Ukoliko kursor nije otvoren javlja se greška

Prijmjer procedure sa kursorima

```
CREATE PROCEDURE curdemo()
BEGIN
   DECLARE done INT DEFAULT FALSE;
   DECLARE a CHAR(16);
   DECLARE b, c INT;
   DECLARE cur1 CURSOR FOR SELECT id,data FROM test.t1;
   DECLARE cur2 CURSOR FOR SELECT i FROM test.t2;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;
   OPEN cur1;
   OPEN cur2;
   read_loop: LOOP
         FETCH cur1 INTO a, b;
         FETCH cur2 INTO c;
         IF done THEN
                  LEAVE read_loop;
         END IF;
         IF b < c THEN
                  INSERT INTO test.t3 VALUES (a,b);
         ELSE
                  INSERT INTO test.t3 VALUES (a,c);
         END IF;
   END LOOP;
   CLOSE cur1;
   CLOSE cur2;
END;
```