SYSTEMY BAZ DANYCH
SQL

TEMAT 25-02: Zmienne i instrukcje sterujące w skryptach SQL.

Autor dokumentu: Wojciech Galiński	sobota, 19 września 2015 r.	351203 Technik informatyk
ŹRÓDŁA WIEDZY:	http://dev.mysql.com/doc/refman/5.7/en/flow-control-statements.html,	

Zagadnienia obowiązkowe

1. **Zmienne w skryptach SQL** – służą do zapamiętywania wyników zapytań SQL m. in. w celu powtórnego wykorzystania w skrypcie SQL. Akcje wykonywane na zmiennych to: tworzenie zmiennych, przypisywanie im wartości oraz wyświetlanie wartości zmiennych.
   Oto wzorce używania zmiennych:

   zmienne globalne:
   ```
   SET @zmienna = wartosc;
   SELECT @zmienna;
   ```
   PRZYKŁAD: `SET @x1 = 5;  SELECT @x1;`

   zmienne lokalne:
   ```
   DECLARE zmienna TYP_ZMIENNEJ;
   SET zmienna = wartosc;
   SELECT zmienna;
   ```
   PRZYKŁAD:
   ```
   DECLARE x FLOAT;  SET x = 2.72;  SELECT x;
   ```

   Wartości zmiennych można przypisywać także do innych zmiennych.
   ```
   PRZYKŁAD 1:  SET @x2 = @x1 * @x1;  SELECT @x1, @x2;
   PRZYKŁAD 2:  SET @lp=0;  SELECT @lp:=@lp+1 AS "L. p.", * FROM osoba;
                -- wstawienie dodatkowej kolumny numerującej wiersze w tabeli (nie mylić z ID)
   ```
2. **Instrukcje warunkowe w skryptach SQL** – wyróżniamy:
   - ***instrukcja „IF”*** – oto wzorzec (w nawiasach kwadratowych znajdują się składniki opcjonalne):
     ```
     IF warunek_wyszukiwania THEN lista_instrukcji_1
          [ELSEIF warunek_wyszukiwania THEN lista_instrukcji_2] …
          [ELSE lista_instrukcji_3]
     END IF
     ```
     PRZYKŁAD:
     ```
     DELIMITER //
     CREATE PROCEDURE sortuj_2_wartosci(n FLOAT)
     BEGIN
       DECLARE x1 FLOAT;  DECLARE x2 FLOAT;
       SET x1 = n;  SET x2 = n * n;
       IF x1 < x2 THEN  SELECT x1, x2;  ELSE SELECT x2, x1;  END IF;
     END //
     DELIMITER ;
     CALL sortuj_2_wartosci(2);  CALL sortuj_2_wartosci(0.5);
     DROP PROCEDURE sortuj_2_wartosci;
     ```
   - ***instrukcja „CASE”*** – to odpowiednik instrukcji „**switch**” (PHP/JavaScript/C/C++) lub „**case**” (Pascal). Oto wzorce:
     ```
     CASE wartosc_sprawdzana
       WHEN wartosc_1
         THEN lista_instrukcji_1
       [WHEN wartosc_2
         THEN lista_instrukcji_2] …
       [ELSE lista_instrukcji_3]
     END CASE
     ```
     ```
     CASE
       WHEN warunek_wyszukiwania_1
         THEN lista_instrukcji_1
       [WHEN warunek_wyszukiwania_2
         THEN lista_instrukcji_2] …
       [ELSE lista_instrukcji_3]
     END CASE
     ```
     PRZYKŁAD:
     ```
     DELIMITER //
     CREATE FUNCTION dzien_tygodnia(dzien INT) RETURNS VARCHAR(25)
     BEGIN
       CASE dzien  WHEN 5 THEN RETURN 'sob';  WHEN 6 THEN RETURN 'nie';  ELSE RETURN '?';
       END CASE;
     END //
     DELIMITER ;
     SELECT dzien_tygodnia(1), dzien_tygodnia(5), dzien_tygodnia(6), dzien_tygodnia(100);
     DROP FUNCTION dzien_tygodnia;
     ```

3. **Pętle w skryptach SQL** – wyróżniamy:
   - ***instrukcja „ITERATE”*** – służy do przerwania bieżącej iteracji w pętli i natychmiastowego rozpoczęcia kolejnej iteracji. Oto wzorzec:

     ```
     ITERATE etykieta
     ```

     Instrukcja ta jest odpowiednikiem instrukcji „**continue**” m. in. w języku Pascal.
   - **instrukcje „RETURN” i „LEAVE”** – przerywają pętlę albo podprogram. W języku SQL w funkcjach używamy „**RETURN**”, a w procedurach używamy „**LEAVE**”. Oto wzorce:

     ```
     LEAVE etykieta
     ```

     ```
     RETURN etykieta
     ```
   - ***pętla „LOOP”*** – to najprostsza pętla języka SQL (nie ma odpowiednika w językach wysokiego poziomu). Używa się jej łącznie z instrukcją „**ITERATE**” oraz „**LEAVE**”. Oto wzorce:

     ```
     LOOP
          lista_instrukcji
          instrukcja_wyjscia_z_petli
     END LOOP
     ```

     ```
     etykieta: LOOP
          lista_instrukcji
          instrukcja_wyjscia_z_petli
     END LOOP etykieta
     ```

     PRZYKŁAD:

     ```
     DELIMITER //
     CREATE PROCEDURE przyklad_loop()
     BEGIN
       DECLARE n INT;
       SET n = 1;
       etykieta: LOOP
         SET n = n + 1;
         IF n < 10 THEN ITERATE etykieta;   END IF;
         LEAVE etykieta;
       END LOOP etykieta;
       SELECT n;
     END //
     DELIMITER ;
     CALL przyklad_loop;   DROP PROCEDURE przyklad_loop;
     ```
   - ***pętla „REPEAT”*** – to odpowiednik pętli „**repeat … until**” w Pascalu. Warunek sprawdzany jest na końcu, a co za tym idzie, pętla wykona się zawsze co najmniej 1 raz. Oto wzorzec:

     ```
     REPEAT
         lista_instrukcji
     UNTIL wyrazenie_warunkowe
     END REPEAT
     ```

     ```
     etykieta: REPEAT
         lista_instrukcji
     UNTIL wyrazenie_warunkowe
     END REPEAT etykieta
     ```
   - ***pętla „WHILE”*** – to pętla, w której warunek sprawdzany jest na początku. Może się ona nie wykonać ani razu. Oto wzorzec:

     ```
     WHILE wyrazenie_warunkowe DO
         lista_instrukcji
     END WHILE
     ```

     ```
     etykieta: WHILE wyrazenie_warunkowe DO
         lista_instrukcji
     END WHILE etykieta
     ```

     PRZYKŁADY:

     ```
     DELIMITER //
     CREATE PROCEDURE moj_repeat()
     BEGIN
       DECLARE n INT;
       SET n = 1;
       REPEAT
         SET n = n + 1;
         UNTIL n>0
       END REPEAT;
       SELECT n;
     END //
     DELIMITER ;
     CALL moj_repeat; DROP PROCEDURE moj_repeat;
     ```

     ```
     DELIMITER //
     CREATE PROCEDURE moj_while()
     BEGIN
       DECLARE n INT;
       SET n = 1;
       WHILE n<0 DO
         SET n = n + 1;
       END WHILE;
       SELECT n;
     END //
     DELIMITER ;
     CALL moj_while; DROP PROCEDURE moj_while;
     ```

Pytania kontrolne

1. Jakie rodzaje zmiennych można definiować w języku SQL i jak używamy zmiennych?
2. Wymień i opisz instrukcje warunkowe oraz pętle wykorzystywane w języku SQL.

Zadania

1. Wykonaj powyższe przykłady i przeanalizuj ich działanie. Utwórz własną procedurę i funkcję zawierającą powyższe instrukcje.