PL/SQL – Kontrollstrukturen

Stephan Karrer

Ablauf-Konstrukte: IF - THEN

```
DECLARE
  sales NUMBER(8,2) := 10100;
  quota NUMBER(8,2) := 10000;
  bonus NUMBER(6,2);
  emp_id NUMBER(6) := 120;
BEGIN
  IF sales > (quota + 200)
    THEN
        bonus := (sales - quota)/4;
        UPDATE employees
            SET salary = salary + bonus
            WHERE employee_id = emp_id;
  END IF;
END;
```

Ablauf-Konstrukte: IF - THEN - ELSE

```
DECLARE
   sales NUMBER(8,2) := 12100;
   quota NUMBER(8,2) := 10000;
   bonus NUMBER(6,2);
   emp_id NUMBER(6) := 120;
BEGIN
   IF sales > (quota + 200)
      THEN bonus := (sales - quota)/4;
      ELSE
         IF sales > quota
            THEN bonus := 50;
            ELSE bonus := 0;
         END IF;
   END IF;
   UPDATE employees SET salary = salary + bonus
      WHERE employee_id = emp_id;
END;
```

Ablauf-Konstrukte: IF - THEN - ELSIF

```
DECLARE
  grade CHAR(1);
BEGIN
  grade := 'B';
  IF grade = 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
    ELSIF grade = 'B' THEN DBMS_OUTPUT.PUT_LINE('Very Good');
    ELSIF grade = 'C' THEN DBMS_OUTPUT.PUT_LINE('Good');
    ELSIF grade = 'D' THEN DBMS_OUTPUT. PUT_LINE('Fair');
    ELSIF grade = 'F' THEN DBMS_OUTPUT.PUT_LINE('Poor');
    ELSE  DBMS_OUTPUT.PUT_LINE('No such grade');
  END IF;
END;
```

Partielle Auswertung von logischen Ausdrücken

```
DECLARE
 on_hand INTEGER := 0;
 on_order INTEGER := 100;
BEGIN
 -- Does not cause divide-by-zero error;
 -- evaluation stops after first expression

 IF (on_hand = 0) OR ((on_order / on_hand) < 5)
     THEN
     DBMS_OUTPUT.PUT_LINE('On hand quantity is zero.');
 END IF;
END;
```

Ablauf-Konstrukte: Mehrfachauswahl mit CASE

```
DECLARE
  grade CHAR(1);
BEGIN
  grade := 'B';
  CASE grade
    WHEN 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
    WHEN 'B' THEN DBMS_OUTPUT.PUT_LINE('Very Good');
    WHEN 'C' THEN DBMS_OUTPUT.PUT_LINE('Good');
    WHEN 'D' THEN DBMS_OUTPUT.PUT_LINE('Fair');
    WHEN 'F' THEN DBMS_OUTPUT.PUT_LINE('Poor');
    ELSE DBMS_OUTPUT.PUT_LINE('No such grade');
  END CASE;
END;
```

Spezialfall: CASE-Expression (SQL-CASE)

```
DECLARE
  grade CHAR(1) := 'B';
  appraisal VARCHAR2(20);
BEGIN
  appraisal :=
  CASE grade
    WHEN 'A' THEN 'Excellent'
    WHEN 'B' THEN 'Very Good'
    WHEN 'C' THEN 'Good'
    WHEN 'D' THEN 'Fair'
    WHEN 'F' THEN 'Poor'
    ELSE 'No such grade'
    END;                -- Hier SQl-Syntax!!
  DBMS_OUTPUT.PUT_LINE
       ('Grade ' || grade || ' is ' || appraisal);
END;
```

Allgemeine CASE-Anweisung (Searched Case)

```
DECLARE
  grade CHAR(1);
BEGIN
  grade := 'B';

  CASE
    WHEN grade = 'A' THEN
        DBMS_OUTPUT.PUT_LINE('Excellent');
    WHEN grade = 'B' THEN
         DBMS_OUTPUT.PUT_LINE('Very Good');
    WHEN grade = 'C' THEN DBMS_OUTPUT.PUT_LINE('Good');
    WHEN grade = 'D' THEN DBMS_OUTPUT.PUT_LINE('Fair');
    WHEN grade = 'F' THEN DBMS_OUTPUT.PUT_LINE('Poor');
    ELSE DBMS_OUTPUT.PUT_LINE('No such grade');
  END CASE;
END;
```

Ablauf-Konstrukte: Einfache Schleifen

```
DECLARE
   s PLS_INTEGER := 0;
   i PLS_INTEGER := 0;
   j PLS_INTEGER;
BEGIN
   <<outer_loop>>
   LOOP
      i := i + 1;
      j := 0;
      <<inner_loop>>
      LOOP
         j := j + 1;
         s := s + i * j; -- sum a bunch of products
         EXIT inner_loop WHEN (j > 5);
         EXIT outer_loop WHEN ((i * j) > 15);
      END LOOP inner_loop;
   END LOOP outer_loop;
   DBMS_OUTPUT.PUT_LINE('The sum of products equals: '
                                          || TO_CHAR(s));
END;
```

Ablauf-Konstrukte: CONTINUE (ab Version 11.1)

```
DECLARE
    x NUMBER := 0;
BEGIN
    LOOP -- After CONTINUE statement, control resumes here
        DBMS_OUTPUT.PUT_LINE ('Inside loop: x = ' ||
                                        TO_CHAR(x));
        x := x + 1;
        CONTINUE WHEN x < 3;
        DBMS_OUTPUT.PUT_LINE
            ('Inside loop, after CONTINUE: x = ' || TO_CHAR(x));
        EXIT WHEN x = 5;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE (' After loop: x = ' || TO_CHAR(x));
END;
```

Ablauf-Konstrukte: WHILE-Schleife

```
DECLARE
 sal employees.salary%TYPE := 0;
 mgr_id employees.manager_id%TYPE;
 lname employees.last_name%TYPE;
 starting_empid employees.employee_id%TYPE := 120;

BEGIN
 SELECT manager_id INTO mgr_id
    FROM employees
    WHERE employee_id = starting_empid;

 WHILE sal <= 15000 LOOP
 SELECT salary, manager_id, last_name INTO sal, mgr_id,
                                          lname
 FROM employees
 WHERE employee_id = mgr_id;
 END LOOP;

END;
```

Ablauf-Konstrukte: Gezählte Schleifen

```
DECLARE
        p NUMBER := 0;

BEGIN
        FOR k IN 1..500 LOOP -- calculate pi with 500 terms
        p := p + ( ( (-1) ** (k + 1) ) / ((2 * k) - 1) );
        END LOOP;

        p := 4 * p;
        DBMS_OUTPUT.PUT_LINE( 'pi is approximately : ' || p );
END;
```

Es gibt auch die Möglichkeit zu dekrementieren mit dem Schlüsselwort REVERSE

Ablauf-Konstrukte: Sprünge

```
DECLARE
    v_last_name VARCHAR2(25);
    v_emp_id NUMBER(6) := 120;
BEGIN
    <<get_name>>
    SELECT last_name INTO v_last_name
        FROM employees
        WHERE employee_id = v_emp_id;
    BEGIN
        DBMS_OUTPUT.PUT_LINE (v_last_name);
        v_emp_id := v_emp_id + 5;
        IF v_emp_id < 120 THEN
            GOTO get_name;
        END IF;
    END;
END;
```