Wykład IV

w **PL/SQL bloki nazwane** to:
funkcje, procedury, pakiety, wyzwalacze

Cechy bloków nazwanych:

- w postaci skompilowanej trwale przechowywane na serwerze wraz z danymi
- wykonywane na żądanie użytkownika lub w wyniku zajścia zdarzenia
- funkcje i procedury mogą (nie muszą) posiadać parametry
  posiadają specyfikację i ciało
- część deklaracyjna funkcji, procedur i pakietów NIE rozpoczyna się od słowa DECLARE

Parametry

- umożliwiają przekazywanie wartości między środowiskiem wywołującym a programem
- w deklaracji programu występują parametry formalne, dla których nie podaje się rozmiaru typu
- definicja parametru formalnego
  <*nazwa*> [<tryb>] typ [{ :=| DEFAULT } <*wartość* >]

Tryby przekazywania parametru

- **IN** (domyślnie) - przekazywanie wartości zmiennej do programu, parametr aktualny to literał lub zmienna, parametr traktowany wewnątrz programu jak stała
- **OUT** – parametr aktualny jest zmienną, do której (na zewnątrz) będzie przekazywana wartość, wewnątrz programu jest traktowany jako niezainicjalizowana zmienna
- **IN OUT** – parametr aktualny jest zmienną posiadającą wartość, która jest przekazywana do programu, wewnątrz jest traktowany jak zainicjalizowana zmienna, przez którą można zwrócić wartość na zewnątrz programu

CREATE [OR REPLACE] FUNCTION <*Nazwa*>
[(<*par1***> [IN | OUT | IN OUT]** <typ> **[, <***par2***> . . .])]**
RETURN< typ>
{ IS | AS }
[< *deklaracje* **;>]**
BEGIN
<*blok_instrukcji* **;>**
RETURN < *wyrażenie* **;>**
[EXCEPTION
<obsługa_*wyjątków* **; >]**
END [<*Nazwa*>**] ;**

przykłady:

```
CREATE  OR  REPLACE  FUNCTION Dwa_imiona
                        ( p_Nr_albumu IN Studenci.Nr_albumu%TYPE )
RETURN BOOLEAN
AS
v_Robo NUMBER:=0;
BEGIN
  SELECT  instr(imiona,',')  INTO  v_Robo FROM  Studenci
         WHERE Nr_albumu = p_Nr_albumu;
  IF v_Robo <> 0  THEN
    RETURN FALSE;
  ELSE
   RETURN  TRUE;
  END  IF;
END Dwa_imiona;


BEGIN
  FOR v_Rec IN ( SELECT nr_albumu, nazwisko FROM Studenci WHERE  rok = 1)
  LOOP
    IF Dwa_imiona( v_Rec.nr_albumu) THEN
    DBMS_output.put_line(v_Rec.nazwisko ||' ma dwa imiona ');
    END  IF;
  END  LOOP;
END;


CREATE  OR  REPLACE  FUNCTION Odsetki(kwota NUMBER, okres NUMBER)
RETURN NUMBER
IS
kwota_odsetek NUMBER:=0 ;
BEGIN
  IF  okres between 1 and 30 THEN  kwota_odsetek := kwota*0.01 ;
  ELSIF  okres between 31 and 60 THEN  kwota_odsetek := kwota*0.02 ;
    ELSE  kwota_odsetek := kwota*0.05 ;
  END  IF ;
  RETURN kwota_odsetek ;
END Odsetki ;
```

Wywołanie funkcji:

```
SELECT Odsetki( 325, 21 ) FROM  Dual ;

SELECT  nazwisko_imie, r.kwota, Odsetki(r.kwota,(sysdate-data_do_zaplaty))
FROM  Odbiorcy  JOIN  Rachunki r  ON ( kod_odbiorcy=dla_kogo )
                        WHERE  . . . ;
```

CREATE [OR REPLACE] PROCEDURE <*Nazwa*>
[(<*par_1*> **[IN | OUT | IN OUT]** <typ> **[,** <*par_2*> **. . .])]**
{ IS | AS }
[<*deklaracje* **;>]**
BEGIN
<*blok_instrukcji*> **;**
[EXCEPTION
<*obsługa_wyjatków* **; >]**
END [<*Nazwa*> **] ;**

przykład:

CREATE TABLE Wyniki
(Kol1 **VARCHAR2(30) ,**
Kol2 **VARCHAR2(10) ,**
Kol3 **NUMBER(4)) ;**

CREATE OR REPLACE PROCEDURE Losowanie**(** Ilu **NUMBER,** p_Rezyser **VARCHAR2)**
IS
CURSOR Bufor **(p**_Rezyser **VARCHAR2) IS**
SELECT Tytul, Kraj, Rok **FROM** Filmy
WHERE upper(Rezyser) = p_Rezyser
ORDER BY DBMS_RANDOM.value() ;

TYPE T_rec IS RECORD
(Tytul **VARCHAR2(40),** Kraj **VARCHAR2(10),** Rok **NUMBER(4));**
Robo **T_rec;**

BEGIN
DELETE FROM Wyniki **;**
OPEN Bufor**(** p_Rezyser**) ;**
LOOP
FETCH Bufor **INTO** Robo **;**
EXIT WHEN (Bufor**%ROWCOUNT** >Ilu **) OR (** Bufor**%NOTFOUND) ;**
INSERT INTO Wyniki **VALUES(** Robo.Tytul, Robo.Kraj, Robo.Rok **) ;**
END LOOP ;
CLOSE Bufor **;**
END Losowanie **;**

Wywołanie procedury

CALL Losowanie (5,’Wajda Andrzej') ;
EXECUTE Losowanie (3,’ ') ;

Pakiet – zbiór logicznie powiązanych zmiennych, stałych, kursorów, wyjątków, procedur, funkcji itp., tworzących jeden nazwany, przechowywany trwale w bazie danych

Składowe pakietu:

- specyfikacja (interfejs) - dostępna dla aplikacji
- ciało (implementacja) - ukryte, opcjonalne

Specyfikacja pakietu

CREATE [OR REPLACE] PACKAGE <*Nazwa*>
{ IS | AS }
< *publiczne-deklaracje* >;
< *specyfikacja_podprogramów*>;
END [<*Nazwa*>];

Ciało pakietu

CREATE [OR REPLACE] PACKAGE BODY <Nazwa>
{ IS | AS }
<*prywatne_deklaracje*>;
< *definicje_podprogramów* >;
[BEGIN
< *instrukcje_inicjalizujące*>; **]**
END [< *Nazwa*>];

Przykład z dokumentacji Oracle
http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_6007.htm#i2065383

```
CREATE OR REPLACE PACKAGE emp_mgmt AS
   FUNCTION hire (last_name VARCHAR2, job_id VARCHAR2,
      manager_id NUMBER, salary NUMBER,
      commission_pct NUMBER, department_id NUMBER)
      RETURN NUMBER;
   FUNCTION create_dept(department_id NUMBER, location_id NUMBER)
      RETURN NUMBER;
   PROCEDURE remove_emp(employee_id NUMBER);
   PROCEDURE remove_dept(department_id NUMBER);
   PROCEDURE increase_sal(employee_id NUMBER, salary_incr NUMBER);
   PROCEDURE increase_comm(employee_id NUMBER, comm_incr NUMBER);
   no_comm EXCEPTION;
   no_sal EXCEPTION;
END emp_mgmt;
/
```

```
CREATE OR REPLACE PACKAGE BODY emp_mgmt AS
   tot_emps NUMBER;
   tot_depts NUMBER;
FUNCTION hire
   (last_name VARCHAR2, job_id VARCHAR2,
    manager_id NUMBER, salary NUMBER,
    commission_pct NUMBER, department_id NUMBER)
   RETURN NUMBER IS new_empno NUMBER;
BEGIN
   SELECT employees_seq.NEXTVAL
      INTO new_empno
      FROM DUAL;
   INSERT INTO employees
      VALUES (new_empno, 'First', 'Last','first.example@oracle.com',
              '(415)555-0100','18-JUN-02','IT_PROG',90000000,00,
              100,110);
      tot_emps := tot_emps + 1;
   RETURN(new_empno);
END;
FUNCTION create_dept(department_id NUMBER, location_id NUMBER)
   RETURN NUMBER IS
      new_deptno NUMBER;
   BEGIN
      SELECT departments_seq.NEXTVAL
         INTO new_deptno
         FROM dual;
      INSERT INTO departments
         VALUES (new_deptno, 'department name', 100, 1700);
      tot_depts := tot_depts + 1;
      RETURN(new_deptno);
   END;
PROCEDURE remove_emp (employee_id NUMBER) IS
   BEGIN
      DELETE FROM employees
      WHERE employees.employee_id = remove_emp.employee_id;
      tot_emps := tot_emps - 1;
   END;
PROCEDURE remove_dept(department_id NUMBER) IS
   BEGIN
      DELETE FROM departments
      WHERE departments.department_id = remove_dept.department_id;
      tot_depts := tot_depts - 1;
      SELECT COUNT(*) INTO tot_emps FROM employees;
   END;
PROCEDURE increase_sal(employee_id NUMBER, salary_incr NUMBER) IS
   curr_sal NUMBER;
   BEGIN
      SELECT salary INTO curr_sal FROM employees
      WHERE employees.employee_id = increase_sal.employee_id;
      IF curr_sal IS NULL
         THEN RAISE no_sal;
      ELSE
         UPDATE employees
         SET salary = salary + salary_incr
         WHERE employee_id = employee_id;
      END IF;
   END;
```

```
PROCEDURE increase_comm(employee_id NUMBER, comm_incr NUMBER) IS
   curr_comm NUMBER;
   BEGIN
      SELECT commission_pct
      INTO curr_comm
      FROM employees
      WHERE employees.employee_id = increase_comm.employee_id;
      IF curr_comm IS NULL
         THEN RAISE no_comm;
      ELSE
         UPDATE employees
         SET commission_pct = commission_pct + comm_incr;
      END IF;
   END;
END emp_mgmt;
/
```

DROP FUNCTION <*Nazwa*>;
DROP PROCEDURE <*Nazwa*>;
DROP PACKAGE [BODY] <*Nazwa*>;

Pakiety wbudowane:

DBMS_DATA_MINIG – umożliwia drążenie danych
DBMS_DATAPUMP – ułatwia przenoszenie danych (metadanych) między bazami danych
DBMS_DIMISION – używany głównie do obsługi hurtowni danych
DBMS_JAVA – umożliwia dostęp do mechanizmów języka Java w języku PL/SQL
DBMS_OUTPUT – narzędzie do diagnozowania
DBMS_RANDOM – zawiera generator liczb losowych bazujący na języku C
DBMS_XMLGEN – umożliwia transformację wyniku SELECT na format XML