PL/SQL blokk

```
[címke]
[DECLARE
   deklarációs utasítás(ok)]
BEGIN
   végrehajtható utasítás(ok)
   [EXCEPTION
      kivételkezelő]
END  [név];
```

PL/SQL alprogramok

- Blokkba ágyazva
- Séma szinten
- Csomagban

PL/SQL alprogramok

Eljárás:

```
PROCEDURE név [(formális
   paraméterlista)]
IS
[deklarációs utasítás(ok)]
BEGIN
   végrehajtható utasítás(ok)
   [EXCEPTION
        kivételkezelő]
END [név];
```

Függvény:

```
FUNCTION név [(formális
   paraméterlista)]
RETURN típus IS
[deklarációs utasítás(ok)]
BEGIN
   végrehajtható utasítás(ok)
   [EXCEPTION
        kivételkezelő]
END [név];
```

```
declare
  b number(2);
  procedure ures is
    a varchar2(20);
    begin
      a:='';
    end ures;
  function ketto return number is
     begin
       return 2;
     end;
begin
  ures;
  b:=ketto;
end;
```

Formális paraméterlista

- Formális paraméterekből áll

név [{IN|{OUT|IN OUT} [NOCOPY]}] *típus*
    [{:=|DEFAULT} *kifejezés*]

- IN: érték szerinti (de nincs értékmásolás)
- OUT: eredmény szerinti
- IN OUT: érték-eredmény szerinti paraméterátadás
- NOCOPY: tanács (hint) a fordítónak, hogy OUT/IN OUT esetben se másoljon értéket
- A paraméterösszerendelés történhet pozíció és/vagy név szerint
- Lokális és csomagbeli alprogramnevek túlterhelhetők

```
DECLARE
  v_Szam          NUMBER;
  PROCEDURE inp(p_In IN NUMBER) IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE('in:' || p_In);
    -- p_in:=0;
  END inp;
BEGIN
  v_Szam := 10;
  DBMS_OUTPUT.PUT_LINE('1:' || v_Szam);
  inp(v_Szam);
  inp(v_Szam + 1000);
END;
/
```

```
DECLARE
  v_szam NUMBER;
 PROCEDURE outp(p_Out OUT NUMBER) IS
  BEGIN
   DBMS_OUTPUT.PUT_LINE(NVL(p_Out, -1));
   p_Out := 20;
  END outp;
BEGIN
  v_Szam := 10;
  DBMS_OUTPUT.PUT_LINE('2:' || v_Szam);
  outp(v_Szam);
  -- outp(v_Szam + 1000);
END;
/
```

```
DECLARE
  v_Szam          NUMBER;
PROCEDURE inoutp(p_Inout IN OUT NUMBER) IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE('inout:' || p_Inout);
    p_Inout := 30;
  END inoutp;
 BEGIN
  v_Szam := 10;
  DBMS_OUTPUT.PUT_LINE('3:' || v_Szam);
  inoutp(v_Szam);
  -- inoutp(v_Szam + 1000);
END;
/
```

```
DECLARE
  v_Szam          NUMBER;
PROCEDURE outp_kivetel(p_Out IN OUT NUMBER) IS
 BEGIN
  p_Out := 40;
  DBMS_OUTPUT.PUT_LINE('kiv. előtt:'|| p_Out);
  RAISE VALUE_ERROR;
 END outp_kivetel;
BEGIN
  v_Szam := 10;
  DBMS_OUTPUT.PUT_LINE('4:' || v_Szam);
  outp_kivetel(v_Szam);
EXCEPTION
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('kiv.kezelőben'|| v_Szam);
END;
/
```

Formális és aktuális paraméterek

- Egymáshoz rendelés:
  - pozíció szerinti (sorrendi kötés)
  - név szerinti kötés
  - Keverve a kettő, először a pozíció szerintiek majd a név szerintiek
- Fix paraméterű alprogramok
- Az aktuális paraméterek száma kevesebb lehet, mint a formális paraméterek száma
- Alprogramok túlterhelése

```
DECLARE
  v_Datum DATE:=TO_DATE('2006-04-10 20:00:00',
  'YYYY-MM-DD HH24:MI:SS');
FUNCTION to_char2(
    p_Datum   DATE DEFAULT SYSDATE,
    p_Format  VARCHAR2 DEFAULT 'YYYY-MON-DD
  HH24:MI:SS'
  ) RETURN VARCHAR2 IS
  BEGIN
    return TO_CHAR(p_Datum, p_Format);
  END to_char2;
…
```

```
…
BEGIN
DBMS_OUTPUT.PUT_LINE(to_char2(v_Datum, 'MM.DD'));
DBMS_OUTPUT.PUT_LINE(
     to_char2(p_Format => 'YYYY-MON-DD‘,
                p_Datum  => v_Datum));
DBMS_OUTPUT.PUT_LINE(
   to_char2(p_Format => ‘YY-MM-DD'));
DBMS_OUTPUT.PUT_LINE(to_char2(v_Datum,
                      p_Format => 'YYYY-MON-DD'));
DBMS_OUTPUT.PUT_LINE(to_char2);
END;
/
```

Hatáskör, élettartam

- Statikus hatáskörkezelés
  - egy név csak deklarációjától kezdve látszik
- Kezeli a lyuk a hatáskörben problémát
- Dinamikus élettartam-kezelés (blokkok és alprogramok esetén)

```
<<blokk>>
DECLARE
  i     NUMBER := 1; p     VARCHAR2(10) := 'Hello';
PROCEDURE alprg1(p NUMBER DEFAULT 22) IS
    i    NUMBER := 2;
 PROCEDURE alprg2(p NUMBER DEFAULT 555) IS
      i    NUMBER := 3;
  BEGIN
  FOR i IN 4..4 LOOP
   DBMS_OUTPUT.PUT_LINE(blokk.i||', '|| blokk.p);
   DBMS_OUTPUT.PUT_LINE(alprg1.i||', '||alprg1.p);
   DBMS_OUTPUT.PUT_LINE(alprg2.i||', '||alprg2.p);
   DBMS_OUTPUT.PUT_LINE(i||', '||p);
      END LOOP;
    END alprg2;
 BEGIN
    alprg2;
  END alprg1;
BEGIN
  alprg1;
END blokk;
```

```
DECLARE
  PROCEDURE elj1(p NUMBER);
  PROCEDURE elj2(p NUMBER) IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE('elj2: ' || p);
    elj1(p+1);
  END elj2;

  PROCEDURE elj1(p NUMBER) IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE('elj1: ' || p);
    IF p = 0 THEN
      elj2(p+1);
    END IF;
  END elj1;
BEGIN
  elj1(0);
END;
```

Tárolt alprogramok

```
CREATE [OR REPLACE] eljárásfej
[AUTHID {DEFINER|CURRENT_USE}]
eljárás_törzs

CREATE [OR REPLACE] függvényfej
[AUTHID {DEFINER|CURRENT_USER}]
[DETERMINISTIC]
függvénytörzs
```

Információ a tárolt alprogramkról

- Adatszótárnézetek:
  - USER_OBJECTS
  - USER_SOURCE
  - USER_ERRORS
- SQL*Plus:
  - SHOW ERRORS

Tárolt alprogram újrafordítása és eldobása

```
ALTER PROCEDURE eljárásnév
  COMPILE [DEBUG];
ALTER FUNCTION függvénynév
  COMPILE [DEBUG];


DROP PROCEDURE eljárásnév;
DROP FUNCTION függvénynév;
```

Tárolt alprogram meghívása

```
CALL alprogram_név
  ([aktuális_paraméter_lista])
     [INTO gazdaváltozó];
```

```
create or replace procedure
      dopl(s varchar2) is
begin
dbms_output.put_line(s);
end;
/
show errors
```

```
create or replace function
  hatvany(p_a number, p_b number)
  return number is
begin
return p_a**p_b;
end;
/
show errors
```

Tárolt alprogramok meghívása

```
begin
   dopl(hatvany(2,3));
end;
/
```

Alprogramnevek túlterhelése

```
DECLARE
 v_Marad        NUMBER;

PROCEDURE marad( p_Ugyfel_id ugyfel.id%TYPE,
       p_Kolcsonozhet  OUT NUMBER) IS
 BEGIN
  SELECT max_konyv - db
   INTO p_Kolcsonozhet
   FROM (SELECT max_konyv FROM ugyfel WHERE id =
  p_Ugyfel_id),
        (SELECT COUNT(1) AS db FROM kolcsonzes
         WHERE kolcsonzo = p_Ugyfel_id);
  END marad;
```

……

```
…
PROCEDURE marad(p_Ugyfel_nev ugyfel.nev%TYPE,
    p_Kolcsonozhet  OUT NUMBER) IS
    v_Id         ugyfel.id%TYPE;
  BEGIN
    SELECT id
      INTO v_Id
      FROM ugyfel
      WHERE nev = p_Ugyfel_nev;
    marad(v_Id, p_Kolcsonozhet);
  END marad;
BEGIN
  marad(15, v_Marad);
  DBMS_OUTPUT.PUT_LINE('1: ' || v_Marad);

  marad('Komor Ágnes', v_Marad);
  DBMS_OUTPUT.PUT_LINE('2: ' || v_Marad);
END;
/
```