

Haladó DBMS ismeretek 1

Hasznos információk

A tantárgy weboldala:

it.inf.unideb.hu/honlap/halado_oracle1

Oracle Junior képzés

Gyakorlatok és a neptun

Gyakorlat követelmények

Ajánlott irodalom

- Juhász István - Gábor András:
PL/SQL-programozás
- www.oracle.com/pls/db112/homepage

A tárgy előfeltétele

Adatbázisrendszerek című tantárgy

Emellett szükséges előismeretek:

deklaráció, kifejezések, vezérlési
szerkezetek, alprogramok, kivételkezelés,
hatáskör, élettartam, stb.

PL/SQL

- PL/SQL:
 - Procedurális nyelv (Procedural Language for SQL)
 - Kombinálja az SQL adatmanipulációt a procedurális nyelvi feldolgozással.
 - Ada-szerű szintakszis



PL/SQL

- Blokkszerkezetű nyelv
- A következő konstrukciókkal egészíti ki az SQL-t:
 - változók és típusok
 - vezérlési szerkezetek
 - alprogramok és csomagok
 - kurzorok és kurzorváltozók
 - kivételkezelés
 - objektumorientált eszközök

PL/SQL blokk

[*címke*]

[DECLARE

deklarációs utasítás(ok)]

BEGIN

végrehajtható utasítás(ok)

[EXCEPTION

kivételkezelő]

END [*név*];

PL/SQL blokk példák

1. Példa

```
BEGIN  
    null;  
END;  
/
```

2. Példa

```
<<pelda>>  
BEGIN  
    delete from orszagok;  
END;  
/
```


Deklarációs rész

- típus definíció
- változó deklaráció
- nevesített konstans deklaráció
- kivétel deklaráció
- kurzor definíció
- alprogram definíció
- (pragma)

Nevesített konstans és változó

- **Változó**

név típus [[NOT NULL] {:= | DEFAULT} *kifejezés*];

Pl:

```
szuletesi_ido DATE;
```

```
dolgozok_szama NUMBER NOT NULL DEFAULT 0;
```

- **Nevesített konstans**

név CONSTANT *típus* [NOT NULL] {:= | DEFAULT} *kifejezés*;

Pl:

```
max_napok_szama CONSTANT NUMBER := 366;
```

Típusok

- Az eddig tanult SQL típusok:

VARCHAR2

CHAR

NUMBER

DATE

- Később lesz szó a többiről

DBMS_OUTPUT csomag

- A PL/SQL nem tartalmaz I/O utasításokat.
- A DBMS_OUTPUT csomag segítségével üzenetet helyezhetünk el egy belső pufferbe. A PUT_LINE eljárással helyezhetünk el üzenetet a pufferben.
- A puffer tartalmát megjeleníthetjük a képernyőn a következő SQL*Plus utasítással:
`SET SERVEROUTPUT ON`

DBMS_OUTPUT példa

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('Sziaztok');
END;
/
```

Utasítások

- Üres

`NULL;`

- Értékadó

`x := 10;`

- Ugró

`GOTO cimke;`

- Elágaztató

- Feltételes
- Case utasítás

- Ciklusok

- SQL utasítások

Feltételes elágaztató utasítás

```
IF feltétel
  THEN utasítás [utasítás]
  [ELSIF feltétel
    THEN utasítás [utasítás]...]...
  [ELSE utasítás [utasítás]...]
END IF;
```

Feltételes elágaztató utasítás

```
DECLARE
  v_Nagyobb NUMBER;
  x          NUMBER:=7;
  y          NUMBER:=6;
BEGIN
  v_Nagyobb := x;
  IF x < y
    THEN v_Nagyobb := y;
  END IF;
END;
/
```


Feltételes elágaztató utasítás

```
DECLARE
  v_Nagyobb NUMBER;
  x          NUMBER:=5;
  y          NUMBER:=6;
BEGIN
  IF x < y
    THEN v_Nagyobb := y;
    ELSE v_Nagyobb := x;
  END IF;
END;
/
```

Feltételes elágaztató utasítás

```
DECLARE
  v_Nagyobb NUMBER;
  x          NUMBER:=5;
  y          NUMBER:=7;
BEGIN
  IF x < y THEN
    DBMS_OUTPUT.PUT_LINE('x kisebb, mint y');
    v_Nagyobb:= x;
  ELSIF x > y THEN
    DBMS_OUTPUT.PUT_LINE('x nagyobb, mint y');
    v_Nagyobb:= y;
  ELSE
    DBMS_OUTPUT.PUT_LINE('x és y egyenlők');
    v_Nagyobb:= x;
  END IF;
END;
/
```

Feltételes elágaztató utasítás

```
DECLARE
  v_Nagyobb NUMBER;
  x          NUMBER:=5;
  y          NUMBER:=7;
  z          NUMBER:=6;
BEGIN
  IF x > y
    THEN IF x > z
           THEN v_Nagyobb := x;
           ELSE v_Nagyobb := z;
         END IF;
    ELSE IF y > z
           THEN v_Nagyobb := y;
           ELSE v_Nagyobb := z;
         END IF;
    END IF;
END;
/
```

A CASE utasítás

```
CASE [szelektor_kifejezés]
    WHEN {kifejezés | feltétel}
        THEN utasítás [utasítás]..
    [WHEN {kifejezés | feltétel}
        THEN utasítás [utasítás]...]...
    [ELSE utasítás [utasítás]...]
END CASE;
```

A CASE utasítás 1. példa

```
BEGIN
  CASE 2
    WHEN 1 THEN
      DBMS_OUTPUT.PUT_LINE('2 = 1');
    WHEN 1+2 THEN
      DBMS_OUTPUT.PUT_LINE('2 = 1 + 2 ');
    ELSE DBMS_OUTPUT.PUT_LINE('egyik sem');
  END CASE;
END;
/
```

A CASE utasítás 2. példa

```
DECLARE
  v_Szam          NUMBER;
BEGIN
  v_Szam := 10;
  CASE
    WHEN v_Szam MOD 2 = 0
      THEN DBMS_OUTPUT.PUT_LINE('Páros. ');
    WHEN v_Szam < 5
      THEN DBMS_OUTPUT.PUT_LINE('Kisebb 5-nél. ');
    WHEN v_Szam > 5
      THEN DBMS_OUTPUT.PUT_LINE('Nagyobb 5-nél. ');
    ELSE DBMS_OUTPUT.PUT_LINE('Ez csak az 5 lehet. ');
  END CASE;
END;
/
```

Ciklusok

- Alapciklus (végtelen ciklus)
 - WHILE ciklus
 - FOR ciklus
 - Kurzor FOR ciklus (később)
-
- ❖ Kilépni: `EXIT [WHEN feltétel];`
 - ❖ Folytatni: `CONTINUE [WHEN feltétel];`

Alapciklus (végtelen ciklus)

Alakja:

```
[címke] LOOP
    utasítás [utasítás]...
END LOOP [címke];
```

Példa:

```
declare
    a number(5) := 10;
    b number(5) := 10;
begin
    loop
        b := b/a;
        a := a-1;
    end loop;
end;
/
```


WHILE ciklus

Alakja:

```
[címke] WHILE feltétel  
        LOOP utasítás [utasítás]...  
        END LOOP[címke];
```

WHILE ciklus példa

```
DECLARE
  v_Faktorialis NUMBER(5);
  i              PLS_INTEGER;
BEGIN
  i := 1;
  v_Faktorialis := 1;
  WHILE v_Faktorialis * i < 10**5 LOOP
    v_Faktorialis := v_Faktorialis * i;
    i := i + 1;
  END LOOP;
END;
/
```

FOR ciklus

- **Alakja:**

[címke]

```
FOR ciklusváltozó
```

```
    IN [REVERSE] alsó_határ..felső_határ
```

```
    LOOP utasítás [utasítás]...
```

```
END LOOP [címke];
```

FOR ciklus példa 1.

```
DECLARE
  v_Osszeg NUMBER(10) :=0;
BEGIN
  FOR i IN 1..100 LOOP
    v_Osszeg := v_Osszeg + i;
  END LOOP;
END;
/
```

FOR ciklus példa 1.

```
DECLARE
    v_Osszeg NUMBER(10) :=0;
BEGIN
    FOR i IN REVERSE 1..100 LOOP
        v_Osszeg := v_Osszeg + i;
    END LOOP;
END;
/
```

EXIT utasítás

Alakja:

```
EXIT [címke] [WHEN feltétel];
```

EXIT utasítás 1. példa

```
DECLARE
    v_Osszeg PLS_INTEGER;
    i        PLS_INTEGER;
BEGIN
    i := 1;
    v_Osszeg := 0;
    LOOP
        v_Osszeg := v_Osszeg + i;
        IF v_Osszeg >= 100 THEN
            EXIT; -- elértük a célt
        END IF;
        i := i+1;
    END LOOP;
END;
/
```

EXIT utasítás 2. példa

```
DECLARE
    v_Osszeg PLS_INTEGER;
    i        PLS_INTEGER;
BEGIN
    i := 1;
    v_Osszeg := 0;
    LOOP
        v_Osszeg := v_Osszeg + i;
        EXIT WHEN v_Osszeg >= 100; -- elértük a célt
        i := i+1;
    END LOOP;
END;
/
```


EXIT utasítás 3. példa

```
DECLARE
  v_Osszeg PLS_INTEGER := 0;
BEGIN
  <<kulso>>
  FOR i IN 1..100 LOOP
    FOR j IN 1..i LOOP
      v_Osszeg := v_Osszeg + i;
      EXIT kulso WHEN v_Osszeg > 100;
    END LOOP;
  END LOOP;
END;
/
```

CONTINUE utasítás

Alakja:

```
CONTINUE [címke] [WHEN feltétel];
```

CONTINUE utasítás 1. példa

```
DECLARE
  x NUMBER := 0;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE ('x=' || x);
    x := x + 1;
    IF x < 3
      THEN CONTINUE;
    END IF;
    DBMS_OUTPUT.PUT_LINE ('After CONTINUE:x=' || x);
    EXIT WHEN x = 5;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE ('After loop:x=' || x);
END;
/
```

CONTINUE utasítás 2. példa

```
DECLARE
  x NUMBER := 0;
BEGIN
  LOOP
    DBMS_OUTPUT.PUT_LINE ('x=' || x);
    x := x + 1;
    CONTINUE WHEN x < 3;
    DBMS_OUTPUT.PUT_LINE ('After CONTINUE:x=' || x);
    EXIT WHEN x = 5;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE ('After loop:x=' || x);
END;
/
```

SQL utasítások PL/SQL-ben

- DML
 - INSERT, DELETE, UPDATE
 - kiegészülnek egy RETURNING utasításrészsel, segítségével az érintett sorok alapján számított értéket kaphatunk meg
 - MERGE
 - „UPSERT” funkcionalitás
 - SELECT
 - kiegészül egy INTO (ill. BULK COLLECT INTO) utasításrészsel
- DCL
 - COMMIT, ROLLBACK, SAVEPOINT