Laborator 1 PL/SQL

Tipuri de date scalare în PL/SQL. Declararea variabilelor. Blocuri. Instrucțiuni.

PL/SQL include atât instrucțiuni *SQL* pentru prelucrarea datelor și pentru gestiunea tranzacțiilor, cât și instrucțiuni proprii.

PL/SQL extinde *SQL* prin construcții specifice limbajelor procedurale (definirea variabilelor, declararea tipurilor, utilizarea structurilor de control, implementarea procedurilor și funcțiilor, introducerea tipurilor obiect și metodelor etc).

Tipurile de date scalare

- tipurile de date care stochează valori numerice
  - *NUMBER* cu subtipurile *DEC*/*DECIMAL*/*NUMERIC*, *FLOAT*/*DOUBLE PRECISION*, *INT*/*INTEGER*/*SMALLINT*, *REAL*
  - *BINARY_FLOAT* și *BINARY_DOUBLE*
  - *BINARY_INTEGER*/*PLS_INTEGER* cu subtipurile *NATURAL*, *NATURALN*, *POSITIVE*, *POSITIVEN*, *SIGNTYPE*, *SIMPLE_INTEGER*
- tipurile de date care stochează caractere
  - *VARCHAR2* cu subtipurile *STRING*/*VARCHAR*
  - *CHAR* cu subtipul *CHARACTER*
- tipurile de date globalizare ce stochează date *unicode*
  - tipurile *NCHAR* și *NVARCHAR2*
- tipurile de date care stochează data calendaristică și ora
  - tipurile *DATE*, *TIMESTAMP*, *TIMESTAMP WITH TIME ZONE*, *TIMESTAMP WITH LOCAL TIME ZONE*, *INTERVAL YEAR TO MONTH*, *INTERVAL DAY TO SECOND*.
- tipul de date *BOOLEAN* stochează valori logice (*true*, *false* sau *null*)

Declararea variabilelor PL/SQL

- Identificatorii *PL/SQL* trebuie declarați înainte să fie referiți în blocul *PL/SQL*. Dacă în declarația unei variabile apar referiri la alte variabile, acestea trebuie să fi fost declarate anterior. Orice variabilă declarată într-un bloc este accesibilă blocurilor conținute sintactic în acesta.
- La declararea variabilelor în *PL/SQL* pot fi utilizate atributele *%TYPE* și *%ROWTYPE*, care reprezintă tipuri de date implicite.
- Atributul *%TYPE* permite definirea unei variabile cu același tip de date ca al altei variabile sau al unei coloane dintr-un tabel.
- Atributul *%ROWTYPE* permite definirea unei variabile de tip înregistrare cu aceeași structură ca a altei variabile de tip înregistrare, a unui tabel sau cursor.

Sintaxa declarării unei variabile este următoarea:

identificator [*CONSTANT*]{*tip_de_date* | *identificator%TYPE* | *identificator%ROWTYPE*} [*NOT NULL*] [{:= | *DEFAULT*} *expresie_PL/SQL*];

- Constantele și variabilele *NOT NULL* trebuie inițializate atunci când sunt declarate, altfel apare eroare la compilare.
- Afișarea valorii variabilelor se face cu ajutorul procedurilor:
  DBMS_OUTPUT.PUT(sir_caractere);
  DBMS_OUTPUT.PUT_LINE(sir_caractere);
  ***Obs*:** Dacă se lucrează în SQL*Plus, atunci pentru activarea modului afișare se utilizează comanda SET SERVEROUTPUT ON.

Blocuri PL/SQL

PL/SQL este un limbaj cu structură de bloc, adică programele sunt compuse din blocuri care pot fi complet separate sau încuibărite unul în altul.

Tipuri de blocuri:

- anonime - sunt blocuri fără nume, care nu sunt stocate în baza de date și sunt compilate de fiecare dată când sunt executate; acest tip de bloc nu permite parametrii și nu poate întoarce un rezultat;
- neanonime - sunt blocuri cu nume care sunt compilate o singură dată, sunt stocate în baza de date și pot fi apelate din alte aplicații.

Un bloc PL/SQL are structura:

```
[<<nume_bloc>>]
[DECLARE
        variabile, cursoare]
BEGIN
        instrucțiuni SQL și PL/SQL
[EXCEPTION
        tratarea erorilor]
END[nume_bloc]
```

Instrucțiuni PL/SQL

PL/SQL dispune de comenzi ce permit controlul execuției unui bloc.

Instrucțiunile limbajului pot fi: iterative (*LOOP*, *WHILE*, *FOR*), de atribuire (:=), condiționale (*IF*, *CASE*), de salt (*GOTO*, *EXIT*) și instrucțiunea vidă (*NULL*).

Comentarii în PL/SQL

- pe o singură linie, prefixate de simbolurile "--", care încep în orice punct al liniei și se termină la sfârșitul acesteia;
- pe mai multe linii, care sunt delimitate de simbolurile "/*" și "*/".

Caracterul ";" este separator pentru instrucțiuni.

Observație Pentru a nu se vedea codul PL/SQL la rularea unui script se setează parametrul ECHO la valoarea OFF.

1. Evaluați următoarele declarații de variabile:

a.
```
DECLARE
    v_nume, v_prenume VARCHAR2(35);  -- greșit
Corect:
DECLARE
    v_nume    VARCHAR2(35);
    v_prenume VARCHAR2(35);
```

b.
```
DECLARE
   v_nr  NUMBER(5);     --corect
```

c.
```
DECLARE
   v_nr NUMBER(5,2) = 10;     --greșit
```
Corect:
```
DECLARE
   v_nr NUMBER(5,2) := 10;
```

d.
```
DECLARE
   v_test   BOOLEAN:= SYSDATE;    --greșit
```
Corect:
```
DECLARE
   v_test   BOOLEAN:=TRUE;
```

e.
```
DECLARE
   v1  NUMBER(5) :=10;
   v2  NUMBER(5) :=15;
   v3  NUMBER(5) := v1< v2;     --greșit
```
Corect:
```
DECLARE
   v1  NUMBER(5) :=10;
   v2  NUMBER(5) :=15;
   v3  BOOLEAN := v1< v2;
```

2. Se dă următorul bloc PL/SQL:

```
<<principal>>
DECLARE
  v_client_id     NUMBER(4):= 1600;
  v_client_nume   VARCHAR2(50):= 'N1';
  v_nou_client_id NUMBER(3):= 500;
BEGIN
 <<secundar>>
 DECLARE
   v_client_id       NUMBER(4) := 0;
   v_client_nume     VARCHAR2(50) := 'N2';
   v_nou_client_id   NUMBER(3) := 300;
   v_nou_client_nume VARCHAR2(50) := 'N3';
 BEGIN
   v_client_id:= v_nou_client_id;
   principal.v_client_nume:=
             v_client_nume ||' '|| v_nou_client_nume;
   --poziția 1
 END;
 v_client_id:= (v_client_id *12)/10;
 --poziția 2
END;
/
```

Determinați:
- valoarea variabilei v_client_id la poziția 1;
- valoarea variabilei v_client_nume la poziția 1;
- valoarea variabilei v_nou_client_id la poziția 1;
- valoarea variabilei v_nou_client_nume la poziția 1;
- valoarea variabilei v_id_client la poziția 2;
- valoarea variabilei v_client_nume la poziția 2.

Soluție:

```
Poz1 v_client_id 300
Poz1 v_client_nume N2
Poz1 v_nou_client_id 300
Poz1 v_nou_client_nume N3
Poz2 v_client_id 1920
Poz2 v_client_nume N2 N3
```

3. Creați un bloc anonim care să afișeze propoziția "Invat PL/SQL" pe ecran.

Varianta 1 - Afișare folosind variabile de legătură

```
VARIABLE g_mesaj VARCHAR2(50)
BEGIN
  :g_mesaj := 'Invat PL/SQL';
END;
/
PRINT g_mesaj
```

Varianta 2 - Afișare folosind procedurile din pachetul standard DBMS_OUTPUT

```
BEGIN
  DBMS_OUTPUT.PUT_LINE('Invat PL/SQL');
END;
/
```

4. Definiți un bloc anonim în care să se afle numele departamentului cu cei mai mulți angajați. Comentați cazul în care există cel puțin două departamente cu număr maxim de angajați.

```
DECLARE
  v_dep departments.department_name%TYPE;
BEGIN
  SELECT department_name
  INTO   v_dep
  FROM   employees e, departments d
  WHERE  e.department_id=d.department_id
  GROUP BY department_name
  HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                     FROM   employees
                     GROUP BY department_id);
  DBMS_OUTPUT.PUT_LINE('Departamentul '|| v_dep);
END;
/
```

5. Rezolvați problema anterioară utilizând variabile de legătură. Afișați rezultatul atât din bloc, cât și din exteriorul acestuia.

```
VARIABLE rezultat VARCHAR2(35)
BEGIN
  SELECT department_name
  INTO   :rezultat
```

```
  FROM   employees e, departments d
  WHERE  e.department_id=d.department_id
  GROUP BY department_name
  HAVING COUNT(*) = (SELECT MAX(COUNT(*))
                     FROM   employees
                     GROUP BY department_id);
  DBMS_OUTPUT.PUT_LINE('Departamentul '|| :rezultat);
END;
/
PRINT rezultat
```

6. Modificați exercițiul anterior astfel încât să obțineți și numărul de angajați din departamentul respectiv.

7. Determinați salariul anual și bonusul pe care îl primește un salariat al cărui cod este dat de la tastatură. Bonusul este determinat astfel: dacă salariul anual este cel puțin 200001, atunci bonusul este 20000; dacă salariul anual este cel puțin 100001 și cel mult 200000, atunci bonusul este 10000, iar dacă salariul anual este cel mult 100000, atunci bonusul este 5000. Afișați bonusul obținut. Comentați cazul în care nu există niciun angajat cu codul introdus.

Obs. Se folosește **instrucțiunea *IF*.**

```
IF condiție1 THEN
   secvența_de_comenzi_1
[ELSIF condiție2 THEN
   secvența_de_comenzi_2]
…
[ELSE
   secvența_de_comenzi_n]
END IF;
```

```
SET VERIFY OFF
DECLARE
   v_cod           employees.employee_id%TYPE:=&p_cod;
   v_bonus         NUMBER(8);
   v_salariu_anual NUMBER(8);
BEGIN
   SELECT salary*12 INTO v_salariu_anual
   FROM   employees
   WHERE  employee_id = v_cod;
   IF v_salariu_anual>=200001
      THEN v_bonus:=20000;
   ELSIF v_salariu_anual BETWEEN 100001 AND 200000
      THEN v_bonus:=10000;
   ELSE v_bonus:=5000;
END IF;
DBMS_OUTPUT.PUT_LINE('Bonusul este ' || v_bonus);
END;
/
SET VERIFY ON
```

8. Rezolvați problema anterioară folosind instrucțiunea CASE.

```
CASE test_var
    WHEN valoare_1 THEN secvența_de_comenzi_1;
    WHEN valoare_2 THEN secvența_de_comenzi_2;
```

```
        …
        WHEN valoare_k THEN secvența_de_comenzi_k;
        [ELSE altă_secvență;]
    END CASE;

    sau

    CASE
        WHEN condiție_1 THEN secvența_de_comenzi_1;
        WHEN condiție_2 THEN secvența_de_comenzi_2,
         …
        WHEN condiție_k THEN secvența_de_comenzi_k;
        [ELSE alta_secvența;]
    END CASE [eticheta];
```

Clauza *ELSE* este opțională.
Dacă aceasta este necesară în implementarea unei probleme, dar practic lipsește, iar *test_var* nu ia nici una dintre valorile ce apar în clauzele *WHEN*, atunci se declanșează eroarea predefinită *CASE_NOT_FOUND* (*ORA - 6592*).

```
DECLARE
   v_cod           employees.employee_id%TYPE:=&p_cod;
   v_bonus         NUMBER(8);
   v_salariu_anual NUMBER(8);
BEGIN
   SELECT salary*12 INTO v_salariu_anual
   FROM   employees
   WHERE  employee_id = v_cod;
   CASE WHEN v_salariu_anual>=200001
             THEN v_bonus:=20000;
        WHEN v_salariu_anual BETWEEN 100001 AND 200000
             THEN v_bonus:=10000;
        ELSE v_bonus:=5000;
   END CASE;
DBMS_OUTPUT.PUT_LINE('Bonusul este ' || v_bonus);
END;
/
```

9. Scrieți un bloc PL/SQL în care stocați prin variabile de substituție un cod de angajat, un cod de departament și procentul cu care se mărește salariul acestuia. Să se mute salariatul în noul departament și să i se crească salariul în mod corespunzător. Dacă modificarea s-a putut realiza (există în tabelul *emp_*** un salariat având codul respectiv) să se afișeze mesajul "Actualizare realizata", iar în caz contrar mesajul "Nu exista un angajat cu acest cod". Anulați modificările realizate.

```
DEFINE p_cod_sal= 200
DEFINE p_cod_dept = 80
DEFINE p_procent =20
DECLARE
  v_cod_sal   emp_***.employee_id%TYPE:= &p_cod_sal;
  v_cod_dept  emp_***.department_id%TYPE:= &p_cod_dept;
  v_procent   NUMBER(8):=&p_procent;
```

```
BEGIN
  UPDATE emp_***
  SET department_id = v_cod_dept,
      salary=salary + (salary* v_procent/100)
  WHERE employee_id= v_cod_sal;
  IF SQL%ROWCOUNT =0 THEN
     DBMS_OUTPUT.PUT_LINE('Nu exista un angajat cu acest cod');
  ELSE DBMS_OUTPUT.PUT_LINE('Actualizare realizata');
  END IF;
END;
/
ROLLBACK;
```

10. Creați tabelul *zile_****(id, data, nume_zi). Introduceți în tabelul *zile_**** informațiile corespunzătoare tuturor zilelor care au rămas din luna curentă.

```
LOOP
   secvența_de_comenzi
END LOOP;
```

Comanda se execută cel puțin o dată.
Dacă nu este utilizată comanda *EXIT*, ciclarea ar putea continua la infinit.

```
DECLARE
  contor  NUMBER(6) := 1;
  v_data  DATE;
  maxim   NUMBER(2) := LAST_DAY(SYSDATE)-SYSDATE;
BEGIN
  LOOP
    v_data := sysdate+contor;
    INSERT INTO zile_***
    VALUES (contor,v_data,to_char(v_data,'Day'));
    contor := contor + 1;
    EXIT WHEN contor > maxim;
  END LOOP;
END;
/
```

11. Rezolvați cerința anterioară folosind instrucțiunea WHILE.

```
WHILE condiție LOOP
   secvența_de_comenzi
END LOOP;
```

Dacă condiția este evaluată ca fiind *FALSE* sau *NULL*, atunci secvența de comenzi nu este executată și controlul trece la instrucțiunea imediat următoare după *END LOOP*.

```
DECLARE
  contor  NUMBER(6) := 1;
  v_data  DATE;
  maxim   NUMBER(2) := LAST_DAY(SYSDATE)-SYSDATE;
```

```
BEGIN
  WHILE contor <= maxim LOOP
    v_data := sysdate+contor;
    INSERT INTO zile_***
    VALUES (contor,v_data,to_char(v_data,'Day'));
    contor := contor + 1;
  END LOOP;
END;
/
```

12. Rezolvați cerința anterioară folosind instrucțiunea FOR.

```
FOR contor_ciclu IN [REVERSE] lim_inf..lim_sup LOOP
   secvența_de_comenzi;
END LOOP;
```

Variabila *contor_ciclu* nu trebuie declarată, ea fiind implicit de tip *BINARY_INTEGER.*
Aceasta este neidentificată în afara ciclului.
Pasul are implicit valoarea 1 și nu poate fi modificat.
Limitele domeniului pot fi variabile sau expresii, dar care pot fi convertite la întreg.

```
DECLARE
  v_data  DATE;
  maxim   NUMBER(2) := LAST_DAY(SYSDATE)-SYSDATE;
BEGIN
  FOR contor IN 1..maxim LOOP
    v_data := sysdate+contor;
    INSERT INTO zile_***
    VALUES (contor,v_data,to_char(v_data,'Day'));
  END LOOP;
END;
/
```

13. Să se declare și să se inițializeze cu 1 variabila *i* de tip *POZITIVE* și cu 10 constanta *max_loop* de tip *POZITIVE*. Să se implementeze un ciclu *LOOP* care incrementează pe *i* până când acesta ajunge la o valoare > *max_loop*, moment în care ciclul *LOOP* este părăsit și se sare la instrucțiunea i:=1.

Obs. Se utilizează **instrucțiunile GOTO/EXIT**.

Instrucțiunea *EXIT* permite ieșirea dintr-un ciclu. Controlul trece fie la prima instrucțiune situată după *END LOOP*-ul corespunzător, fie la instrucțiunea având eticheta *nume_eticheta*.

```
EXIT [nume_eticheta] [WHEN condiție];
```

Numele etichetelor urmează aceleași reguli ca și cele definite pentru identificatori. Eticheta se plasează înaintea comenzii, fie pe aceeași linie, fie pe o linie separată. Etichetele se definesc prin intercalare între “<<” și “>>”.

```
GOTO nume_eticheta;
```

Nu este permis saltul:

- în interiorul unui bloc (subbloc);
- în interiorul unei comenzi *IF*, *CASE* sau *LOOP*;

- de la o clauză a comenzii *CASE*, la altă clauză aceleași comenzi;
- de la tratarea unei excepții, în blocul curent;
- în exteriorul unui subprogram.

Varianta 1

```
DECLARE
   i         POSITIVE:=1;
   max_loop CONSTANT POSITIVE:=10;
BEGIN
LOOP
   i:=i+1;
   IF i>max_loop THEN
     DBMS_OUTPUT.PUT_LINE('in loop i=' || i);
     GOTO urmator;
   END IF;
END LOOP;
<<urmator>>
i:=1;
DBMS_OUTPUT.PUT_LINE('dupa loop i=' || i);
END;
/
```

Varianta 2

```
DECLARE
  i         POSITIVE:=1;
  max_loop CONSTANT POSITIVE:=10;
BEGIN
  i:=1;
  LOOP
    i:=i+1;
    DBMS_OUTPUT.PUT_LINE('in loop i=' || i);
    EXIT WHEN i>max_loop;
  END LOOP;
  i:=1;
  DBMS_OUTPUT.PUT_LINE('dupa loop i=' || i);
END;
/
```

Exerciții

E1. Se dă următorul bloc:

```
DECLARE
 numar number(3):=100;
mesaj1 varchar2(255):='text 1';
 mesaj2 varchar2(255):='text 2';
BEGIN
  DECLARE
   numar number(3):=1;
   mesaj1 varchar2(255):='text 2';
   mesaj2 varchar2(255):='text 3';
```

```
  BEGIN
   numar:=numar+1;
   mesaj2:=mesaj2||' adaugat in sub-bloc';
  END;
 numar:=numar+1;
 mesaj1:=mesaj1||' adaugat un blocul principal';
 mesaj2:=mesaj2||' adaugat in blocul principal';
END;
```

a) Valoarea variabilei *numar* în subbloc este:
b) Valoarea variabilei *mesaj1* în subbloc este:
c) Valoarea variabilei *mesaj2* în subbloc este:
d) Valoarea variabilei *numar* în bloc este:
e) Valoarea variabilei *mesaj1* în bloc este:
f) Valoarea variabilei *mesaj2* în bloc este:
Verificați răspunsul.

E2. Se dă următorul enunț: Pentru fiecare zi a lunii octombrie (se vor lua în considerare și zilele din lună în care nu au fost realizate împrumuturi) obțineți numărul de împrumuturi efectuate.

a. Încercați să rezolvați problema în SQL fără a folosi structuri ajutătoare.
b. Definiți tabelul *octombrie_*** (id, data). Folosind PL/SQL populați cu date acest tabel. Rezolvați în SQL problema dată.

E3. Definiți un bloc anonim în care să se determine numărul de filme (titluri) împrumutate de un membru al cărui nume este introdus de la tastatură. Tratați următoarele două situații: nu există nici un membru cu nume dat; există mai mulți membrii cu același nume.

E4. Modificați problema anterioară astfel încât să afișați și următorul text:
- Categoria 1 (a împrumutat mai mult de 75% din titlurile existente)
- Categoria 2 (a împrumutat mai mult de 50% din titlurile existente)
- Categoria 3 (a împrumutat mai mult de 25% din titlurile existente)
- Categoria 4 (altfel)

E5. Creați tabelul *member_*** (o copie a tabelului *member*). Adăugați în acest tabel coloana *discount*, care va reprezenta procentul de reducere aplicat pentru membrii, în funcție de categoria din care fac parte aceștia:
- 10% pentru membrii din Categoria 1
- 5% pentru membrii din Categoria 2
- 3% pentru membrii din Categoria 3
- nimic

Actualizați coloana *discount* pentru un membru al cărui cod este dat de la tastatură. Afișați un mesaj din care să reiasă dacă actualizarea s-a produs sau nu.

E6. Adaptați cerințele exercițiilor 5, 7 și 9 pentru diagrama proiectului prezentată la materia Baze de Date din anul I. Rezolvați aceste exerciții în PL/SQL, folosind baza de date proprie.