SGBD - Anul 3
Laborator 1

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

PL/SQL este extensia procedurală a limbajului SQL, cu trăsături specifice limbajelor de programare.

I. Tipuri de date scalare

Nu au componente interne (conțin valori atomice). Se împart în 5 clase.

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

Obs : Mai multe informații despre tipurile de date PL/SQL la

http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28370/datatypes.htm#CJAEDAEA

II. Variabile de legătură *PL/SQL*

- **O variabilă de legătură** (globală sau *bind variable*) este variabila care se declară într-un mediu gazdă şi este folosită pentru transferul la execuție al valorilor numerice sau de tip caracter în/din unul sau mai multe programe *PL/SQL*.
- Variabilele declarate în mediul gazdă sau în cel apelant pot fi referite în instrucțiuni *PL/SQL* dacă acestea nu sunt în cadrul unei proceduri, funcții sau pachet.
- În *SQL*Plus*, variabilele de legătură se declară folosind comanda ***VARIABLE*,** iar pentru tipărirea acestora se utilizează comanda ***PRINT***. Ele sunt referite prin prefixare cu simbolul "**:**", pentru a putea fi deosebite de variabilele *PL/SQL* declarate.

III. 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.

- În declararea variabilelor în *PL/SQL* pot fi utilizate atributele ***%TYPE*** şi ***%ROWTYPE***, care reprezintă tipuri de date implicite.
- Atributul *%TYPE* permite definirea unei variabile având tipul unei variabile declarate anterior sau tipul unei coloane dintr-un tabel.
- Atributul *%ROWTYPE* permite definirea unei variabile având tipul unei înregistrări dintr-un tabel.

Sintaxa declarării unei variabile este următoarea:

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

***Exemplu*:**

```
v_valoare          NUMBER(15) NOT NULL := 0;
v_data_achizitie DATE DEFAULT SYSDATE;
v_material         VARCHAR2(15) := 'Matase';
c_valoare          CONSTANT NUMBER := 100000;
v_stare            VARCHAR2(20) DEFAULT 'Buna';
v_clasificare      BOOLEAN  DEFAULT FALSE;
v_cod_opera        opera.cod_opera%TYPE;
v_opera            opera%ROWTYPE;
int_an_luna        INTERVAL YEAR TO MONTH :=
                   INTERVAL '3-2' YEAR TO MONTH; --interval de 3 ani si 2 luni
```

***Observații*:**

- Pentru pentru uşurința referirii se convine prefixarea numelor de variabile astfel:
- prefixarea cu litera *v* (*v_valoare*) pentru varibilele PL/SQL
- prefixarea cu litera *c* (*c_valoare*) pentru constante
- parametrii de substituție (variabilele de substituție din SQL*Plus) se prefixează cu litera *p*
- variabilele globale (*bind variables*) se prefixează cu *g*.
- Variabilele pot fi inițializate, iar dacă o variabilă nu este inițializată, valoarea implicită a acesteia este *NULL*. Dacă o variabilă este declarată *NOT NULL*, atunci ea va fi obligatoriu inițializată.
- Constantele trebuie inițializate când sunt declarate, altfel apare eroare la compilare.

!!! Afişarea valorii variabilelor se face cu ajutorul procedurilor:

DBMS_OUTPUT.PUT(sir_caractere);

DBMS_OUTPUT.PUT_LINE(sir_caractere);
Obs: se utilizează **SET SERVEROUTPUT ON** pentru activarea modului afişare.

IV. Instrucțiuni PL/SQL

- iterative (*LOOP*, *WHILE*, *FOR*),
- de atribuire (:=),
- condiționale (*IF*, *CASE*),
- de salt (*GOTO*, *EXIT*),
- instrucțiunea vidă (*NULL*).

Observații

- Comentariile sunt ignorate de compilatorul *PL/SQL*. Există două tipuri de comentarii:

- pe o singură linie, prefixate de simbolurile “--”, care încep în orice punct al liniei şi se termină la sfârşitul acesteia şi
- pe mai multe linii, care sunt delimitate de simbolurile “/*” şi “*/”.

- Caracterul “**;”** este separator pentru instrucțiuni.
- Operatorii din *PL/SQL*, ca şi ordinea de execuție a acestora, sunt identici cu cei din *SQL*. În *PL/SQL* este introdus un nou operator (**).
- Un identificator este vizibil în blocul în care este declarat şi în toate subblocurile, procedurile şi funcțiile încuibărite în acesta. Dacă blocul nu găseşte identificatorul declarat local, atunci îl caută în secțiunea declarativă a blocurilor care includ blocul respectiv şi niciodată nu caută în blocurile încuibărite în acesta.
- Comenzile *SQL*Plus* **nu** pot să apară într-un bloc *PL/SQL*.
- În comanda *SELECT* trebuie specificate variabilele care recuperează rezultatul acțiunii acestei comenzi. În clauza *INTO*, care este obligatorie, pot fi folosite variabile *PL/SQL* sau variabile de legătură.
- Referirea la o variabilă de legătură se face în *PL/SQL* prin prefixarea acestei variabile utilizând caracterul “**:**”.
- Cererea *SELECT* trebuie să întoarcă ca rezultat o singură linie. Dacă întoarce mai multe linii, atunci apare eroarea *TOO_MANY_ROWS*, iar în cazul în care comanda nu găseşte date se generează eroarea *NO_DATA_FOUND*.

!!! Pentru evaluarea unei condiții logice care apare în comenzile limbajului, trebuie remarcat că orice expresie ce conține o valoare *null* este evaluată *null*. Singura excepție o constituie operatorul de concatenare.

1) Instrucțiunea de atribuire

variabila := expresie;

Obs: Nu poate fi asignată valoarea *NULL* unei variabile care a fost declarată *NOT NULL*.

2) Instrucțiunea *IF*

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

Este permis un număr arbitrar de opțiuni *ELSIF*, dar poate fi cel mult o clauză *ELSE*. Aceasta se referă la ultimul *ELSIF*.

3) Instrucțiunea *CASE*

Comanda *CASE* permite implementarea unor condiții multiple. Instrucțiunea are următoarea formă sintactică:

[<<*eticheta*>>]
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*** *alta_secvența*;]
END CASE [*eticheta*];

Sau următoarea formă, în care fiecare clauză *WHEN* conține o expresie booleană.

[<<*eticheta*>>]
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*];

4) Instrucțiuni iterative

Instrucțiunile de ciclare pot fi:
- încuibărite pe multiple niveluri;
- etichetate;
- ieşirea din ciclare se poate realiza cu ajutorul comenzii *EXIT*.

a) 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.

b) WHILE *condiție* ***LOOP***
secvența_de_comenzi
END LOOP;

În cazul în care condiția este evaluată ca fiind *FALSE* sau *NULL*, atunci secvența de comenzi nu este executată şi controlul trece la instrucțiunea după *END LOOP*.

Instrucțiunea repetitivă *FOR* (ciclare cu pas) permite executarea unei secvențe de instrucțiuni pentru valori ale variabilei *contor* cuprinse între două limite, *lim_inf* şi *lim_sup*. Dacă este prezentă opțiunea *REVERSE*, iterația se face (în sens invers) de la *lim_sup* la *lim_inf*.

c) 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*** şi este neidentificată în afara ciclului. Pasul are implicit valoarea 1 şi nu poate fi modificat. Limitele domeniului pot fi variabile sau expresii, dar care pot fi convertite la întreg.

5) Instrucțiuni de salt

Instrucțiunea *EXIT* permite ieşirea 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 aceleaşi reguli ca cele definite pentru identificatori. Eticheta se plasează înaintea comenzii, fie pe aceeaşi linie, fie pe o linie separată. Etichetele se definesc prin intercalare între “<<” şi “>>”.

Exemplu:

```
DECLARE
  v_contor    BINARY_INTEGER := 1;
  raspuns     VARCHAR2(10);
  alt_raspuns VARCHAR2(10);
BEGIN
  …
  <<exterior>>
  LOOP
    v_contor := v_contor + 1;
  EXIT WHEN v_contor > 70;
    <<interior>>
    LOOP
      …
      EXIT exterior WHEN raspuns = 'DA';
      -- se parasesc ambele cicluri
      EXIT WHEN alt_raspuns = 'DA';
      -- se paraseste ciclul interior
      …
    END LOOP interior;
    …
  END LOOP exterior;
END;
```

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 a aceleaşi comenzi;
- de la tratarea unei excepții, în blocul curent;
- în exteriorul unui subprogram.

7) Instrucțiunea vidă. *NULL* este instrucțiunea care nu are niciun efect. Nu trebuie confundată instrucțiunea *NULL* cu valoarea *null*!

V. Blocuri PL/SQL

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

Un program poate cuprinde unul sau mai multe blocuri. Un bloc poate fi anonim sau neanonim.

- **Blocurile anonime** sunt blocuri *PL/SQL* fără nume, care sunt construite dinamic şi sunt executate o singură dată. Acest tip de bloc nu are argumente şi nu returnează un rezultat.
- **Blocurile neanonime** sunt fie blocuri având un nume (etichetate), care sunt construite static sau dinamic şi sunt executate o singură dată, fie subprograme, pachete sau declanşatori..

Structura unui bloc *PL/SQL* este compusă din trei secțiuni distincte:

Blocul *PL/SQL* are următoarea structură generală:

[<<*nume_bloc*>>]
[***DECLARE***
  instrucțiuni de declarare]
BEGIN
  instrucțiuni executabile (*SQL sau PL/SQL*)

[***EXCEPTION***
tratarea erorilor]
END [*nume_bloc*];

Dacă blocul *PL/SQL* este executat fără erori, va apărea mesajul:

PL/SQL procedure successfully completed

Compatibilitate *SQL*

Din punct de vedere al compatibilității *PL/SQL versus SQL* există următoarele reguli de bază:

- *PL/SQL* furnizează toate comenzile *LMD* ale lui *SQL*, comanda *SELECT* cu clauza *INTO*, comenzile *LCD*, funcțiile, pseudo-coloanele şi operatorii *SQL*;
- ***PL/SQL* nu include comenzile *LDD***.
- Majoritatea funcțiilor *SQL* sunt disponibile în *PL/SQL*.

- Există funcții noi, specifice *PL/SQL*, cum sunt funcțiile *SQLCODE* şi *SQLERRM*.
- Există funcții *SQL* care nu sunt disponibile în instrucțiuni procedurale (de exemplu, ***DECODE*, NULLIF, funcțiile grup**), dar care sunt disponibile în instrucțiunile *SQL* dintr-un bloc *PL/SQL*. *SQL* nu poate folosi funcții sau atribute specifice *PL/SQL*.

! Funcțiile grup trebuie folosite cu atenție, deoarece instrucțiunea *SELECT … INTO* nu poate conține clauza *GROUP BY*.

Exercitii

1. Creați un bloc anonim care să afişeze propoziția "Invat PL/SQL" pe ecran.

2. Să se creeze un bloc anonim în care se declară o variabilă *v_oras* de tipul coloanei *city* (*locations.city%TYPE*). Atribuiți acestei variabile numele oraşului în care se află departamentul având codul 30.

3. Să se creeze un bloc anonim în care să se afle media salariilor pentru angajații al căror departament este 50. Se vor folosi variabilele *v_media_sal* de tipul coloanei *salary* şi *v_dept* (de tip NUMBER).

4. Să se specifice dacă un departament este mare, mediu sau mic după cum numărul angajaților săi este mai mare ca 30, cuprins între 10 şi 30 sau mai mic decât 10. Codul departamentului va fi cerut utilizatorului.

5. Stocați într-o variabilă de substituție p_cod_dep valoarea unui cod de departament. Definiți şi o variabilă p_com care reține un număr din intervalul [0, 100]. Pentru angajații din departamentul respectiv care nu au comision, să se atribuie valoarea lui p_com câmpului commission_pct. Afişați numărul de linii afectate de această actualizare. Dacă acest număr este 0, să se scrie « Nicio linie actualizata ».

Obs: (vom reveni în laboratorul despre cursoare)
Atributele cursoarelor implicite :

- SQL%ROWCOUNT – Numarul de linii afectate de cea mai recenta comanda SQL;
- SQL%FOUND – Atribut boolean ce returneaza TRUE daca ultima comanda SQL a afectat cel putin o linie;
- SQL%NOTFOUND – Atribut boolean ce returneaza TRUE daca ultima comanda SQL nu a afectat nici o linie
- SQL%ISOPEN – Atribut boolean ce returneaza TRUE daca cursorul implicit asociat ultimei comenzi a ramas deschis. Nu e niciodata true pentru ca serverul inchide automat cursorul la terminarea comenzii SQL.

6. În funcție de o valoare introdusă de utilizator, utilizând comanda *CASE* se va afişa un mesaj prin care este specificată ziua săptămânii (a cărei abreviere este chiar valoarea respectivă).

7. În structura tabelului *emp* se va introduce un nou câmp (*vechime* de tip *VARCHAR2(200)*). Să se creeze un bloc *PL/SQL* care va reactualiza acest câmp, introducând un "#" pentru fiecare an de vechime al unui angajat al cărui cod este specificat de către utilizator.

8. Folosind o instructiune *LOOP* calculați n!.

Exerciții propuse:

1. Se consideră următorul bloc PL/SQL:

```
<<bloc>>
DECLARE
   v_cantitate NUMBER(3) := 300;
   v_mesaj     VARCHAR2(255) := 'Produs 1';
BEGIN
   <<subbloc>>
   DECLARE
      v_cantitate NUMBER(3) := 1;
      v_mesaj     VARCHAR2(255) := 'Produs 2';
      v_locatie   VARCHAR2(50) :=  'Europa';
   BEGIN
      v_cantitate := v_cantitate + 1;
      v_locatie   :=  v_locatie || 'de est';
    END;
v_cantitate:= v_cantitate + 1;
v_mesaj := v_mesaj ||' se afla in stoc';
v_locatie :=  v_locatie || 'de est' ;
END;
/
```

Evaluați:
- valoarea variabilei v_cantitate în subbloc;
- valoarea variabilei v_locatie la poziția în subbloc ;
- valoarea variabilei v_cantitate în blocul principal ;
- valoarea variabilei v_mesaj în blocul principal ;
- valoarea variabilei v_locație în blocul principal.

2. Creati si executati un bloc PL/SQL care cere de la tastatura 2 numere (prin variabile de substitutie SQL* Plus). Calculati valoarea functiei :

$$f(x,y)=\begin{cases}\dfrac{x}{y}+y, \text{daca } y\neq 0\\ x^2, \text{daca } y=0\end{cases}$$

Rezultatul va fi retinut intr-o variabila PL/SQL si va fi tiparit pe ecran.

3. Să se calculeze suma salariilor pentru un job al cărui cod este introdus de utilizator. Căutarea se va face case-insensitive.

4. Sa se creeze un bloc PL/SQL care calculeaza si modifica valoarea comisionului pentru un angajat al carui cod este dat de la tastatura, pe baza salariului acestuia, astfel:
- daca salariul este mai mic decat 1000$, comisionul va fi 10% din salariu;
- daca salariul ete intre 1000 si 1500$, comisionul va fi 15% din salariu;
- daca salariul depaseste 1500$, comisionul va fi 20% din salariu;
- daca salariul este NULL, comisionul va fi 0.

Modificările se fac în tabelul emp.

5. Creați structura tabelului *org_tab* constând din două coloane, *cod_tab* de tip *INTEGER* ce conține un contor al înregistrărilor şi *text_tab* de tip *VARCHAR2* ce conține un text asociat fiecărei înregistrări. Să se introducă 70 de înregistrări în acest tabel. Se cer 2 metode (LOOP si WHILE).

6. Scrieți un bloc PL/SQL care actualizează conținutul tabelului anterior, indicând pe coloana text_tab dacă numărul cod_tab este par sau impar.

7. Sa se creeze un bloc PL/SQL care selecteaza codul maxim de departament din tabelul DEPT si il stocheaza intr-o variabila SQL*Plus. Se va tipari rezultatul pe ecran.

8. Sa se creeze un bloc PL/SQL care insereaza un nou departament in tabelul DEPT. Se va folosi parametru de substitutie pentru numele departamentului. Codul este dat de valoarea variabilei calculate anterior +10. Locatia va avea valoarea null. Sa se listeze continutul tabelului DEPT.

9. Sa se creeze un bloc PL/SQL care reactualizeaza locatia pentru un departament existent (în tabelul DEPT). Se vor folosi parametri de substitutie pentru numarul departamentului si locatia acestuia. Sa se listeze codul, numele si locatia pentru departamentul reactualizat.

10. Sa se creeze un bloc PL/SQL care sterge departamentul creat la exercitiul 8. Se va folosi un parametru de substitutie pentru numarul departamentului. Se va tipari pe ecran numarul de linii afectate. Ce se intampla daca se introduce un cod de departament care nu exista?