Laborator 3

Variabile. Tipuri de date. Instructiuni

Variabile

- PL/SQL acceptă toate tipurile de date din SQL
- Nu sunt permise referințe anticipate
- 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: nume_var [CONSTANT]{tip_de_date|identificator%TYPE| identificator%ROWTYPE} [NOT NULL] [{:=|DEFAULT} expresie_PL/SQL];**

Variabile

***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
```

Tipuri de date

Tipuri de date scalare

- Nu au componente interne (conţin valori atomice).

  - 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 ce stochează **date în format unicode** includ tipurile *NCHAR, NVARCHAR2*
  - Tipul de date *BOOLEAN* stochează **valori logice** (*true*, *false* sau *null*).

Tipuri de date

- Tipuri de date corelate
  - Tipurile corelate sunt folosite pentru a declara variabile de tipul unui rând de tabel sau al unei coloane de tabel. Avantajul folosirii unor astfel de tipuri este că atunci când se modifică structura coloanelor dintr-un tabel nu mai trebuie să fie modificat și codul **PL/SQL**.
  - Tipuri corelate:
    - %TYPE - variabilă de tipul unei coloane de tabel
    - %ROWTYPE - vector de tipul unui rând de tabel
- Tipuri de date compuse (lab. urmator)

Tipuri de date

- Tipuri de date LOB
  - Tipuri LOB (Large OBject) sunt folosite pentru stocarea de date binare, text, imagini și fișiere video de dimensiuni mari (până la 4GB).

Data Type	Description
CLOB	A character large object containing single-byte or multibyte characters. Both fixed-width and variable-width character sets are supported, both using the database character set. Maximum size is (4 gigabytes - 1) * (database block size).
NLOB	A character large object containing Unicode characters. Both fixed-width and variable-width character sets are supported, both using the database national character set. Maximum size is (4 gigabytes - 1) * (database block size). Stores national character set data.
BLOB	A binary large object. Maximum size is (4 gigabytes - 1) * (database block size).
BFILE	Contains a locator to a large binary file stored outside the database. Enables byte stream I/O access to external LOBs residing on the database server. Maximum size is 4 gigabytes.

Instructiuni

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

- Obs**!!!** Pentru evaluarea unei condiţii logice care apare în comenzile limbajului, trebuie remarcat că orice expresie ce conţine o valoare *null* este evaluată *null*. Singura excepţie o constituie operatorul de concatenare.

Instructiuni

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*.

Instructiuni

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.

Instructiuni

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 “>>”.

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.

Instructiuni

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*;

Exercitii (1)

1. Care dintre urmatoarele declaratii nu sunt corecte si explicati de ce:

a) DECLARE v_id NUMBER(4);

b) DECLARE v_x, v_y, v_z VARCHAR2(10);

c) DECLARE v_birthdate DATE NOT NULL;

d) DECLARE v_in_stock BOOLEAN := 1;

e) DECLARE name VARCHAR2(20) NOT NULL := 'JoHn SmItH';
same_name name%TYPE;

2. Determinati tipul de date al rezultatului in fiecare din atribuirile urmatoare:

a) v_days_to_go := v_due_date - SYSDATE;

b) v_sender := USER || ': '||TO_CHAR(v_dept_no);

c) v_sum := $100,000 + $250,000;

d) v_flag :=TRUE;

e) v_n1 := v_n2 > (2 * v_n3);

f) v_value :=NULL;

Exercitii (2)

1. Să se creeze un bloc anonim în care se declară o variabilă *v_oras* de tipul coloanei *city*. Atribuiţi acestei variabile numele oraşului în care se află departamentul având codul 30.
2. Să se creeze un bloc anonim în care să se afle media salariilor pentru angajaţii al căror departament este 50. Se vor folosi variabilele *v_media_sal* de tipul coloanei *salary* şi *v_dept* (de tip NUMBER).
3. Să se specifice dacă un departament este mare, mediu sau mic după cum numărul angajaţilor săi este mai mare ca 30, cuprins între 10 şi 30 sau mai mic decât 10. Codul departamentului va fi cerut utilizatorului.
4. Creati un bloc PL/SQL care calculeaza castigul total pentru un an; salariul lunar si procentul care reprezinta bonusul sunt transmise blocului PL/SQL prin variabile de substitutie. Bonusul se va introduce ca numar intreg (pentru bonus de 15% se va introduce 15). Daca salariul este null, va fi setat la 0 inainte de a calcula castigul total. Executati blocul PL/SQL. Se va folosi functia NVL pentru manipularea valorilor NULL.
5. Stocaţi într-o variabilă de substituţie p_cod_dep valoarea unui cod de departament. Definiţi şi o variabilă p_com care reţine un număr din intervalul [0, 100]. Pentru angajaţii din departamentul respectiv care nu au comision, să se atribuie valoarea lui p_com câmpului commission_pct. Afişaţi numărul de linii afectate de această actualizare. Dacă acest număr este 0, să se scrie « Nicio linie actualizata ». (SQL%ROWCOUNT)
6. În funcţie 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) (ambele variante pentru comanda CASE).

7. Să se acorde un comision de 10%, managerilor companiei care nu au primit comision, dar președintele să nu primească comision. (pe tabelul emp – copie employees)
8. 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.

8. 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.
9. 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.
10. 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.
11. Sa se creeze un bloc PL/SQL care sterge departamentul creat la exercitiul 9. 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?

12. Să se calculeze impozitul pe salariu al unui angajat, după algoritmul:

- Din tabela JOBS se află salariul minim, respectiv maxim, pentru funcția pe care o deține;
- Dacă angajatul are salariul minim, atunci impozitul este de 10% din salariu.
- Dacă angajatul are salariul maxim, atunci impozitul este de 30% din salariu.
- Altfel impozitul este de 20% din salariu.
- Se afișează numele și impozitul angajatului.
- Sa se trateze cazurile care ar putea da erori.

Alte exerctii (3)

1. Calculate the moment when someone has lived for 10^9 seconds.
A gigasecond is 10^9 (1,000,000,000) seconds.

```
Tests:
```

- `2011-04-25, 1977-06-13, 1959-07-19, 1959-07-19 23:59:59`

2. Given a DNA strand, return its RNA complement (per RNA transcription).
Both DNA and RNA strands are a sequence of nucleotides.
The four nucleotides found in DNA are adenine (**A**), cytosine (**C**), guanine (**G**) and thymine (**T**).
The four nucleotides found in RNA are adenine (**A**), cytosine (**C**), guanine (**G**) and uracil (**U**).
Given a DNA strand, its transcribed RNA strand is formed by replacing each nucleotide with its complement:

- `G` -> `C`
- `C` -> `G`
- `T` -> `A`
- `A` -> `U`

```
Optional: reverse transcription
Tests:
```

- `C, G, T, A, ACGTGGTCTTAA`
- `C, G, A, U, UGAACCCGACAUG`

- `Hint: REGEXP_LIKE, TRANSLATE`